Object Manager Manual

Table of Contents

I. System structure

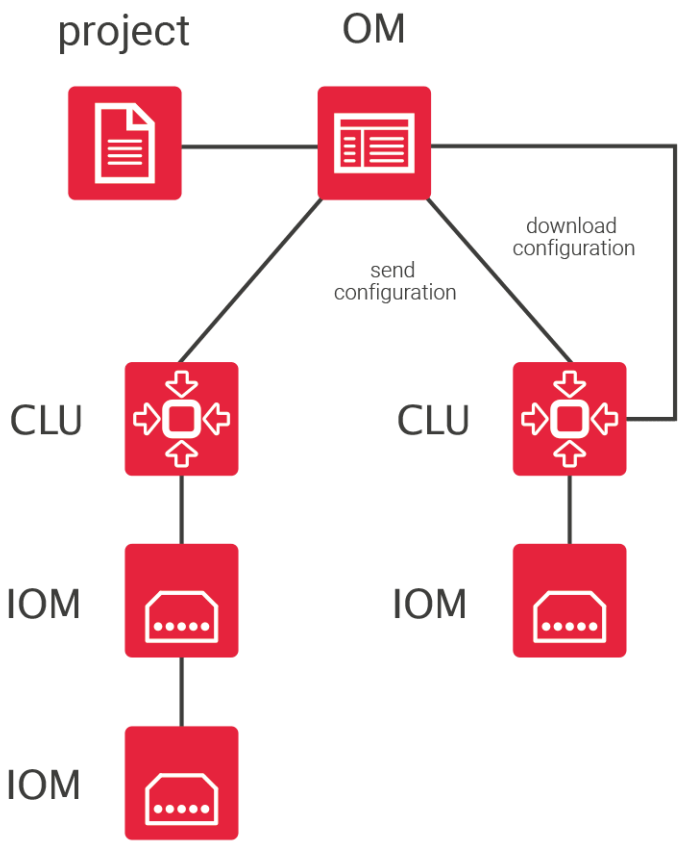
GRENTON Smart Building System was designed to operate small, medium, and large objects. System created on its basis can be easily modified, expanded, and integrated with other systems.

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 alsocommunication 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 assosiated with losses of: graphic view of scripts, containers, mobile interfaces and objects types (source/load).



II. Foundation - Grenton Logical Interface

1. Introduction

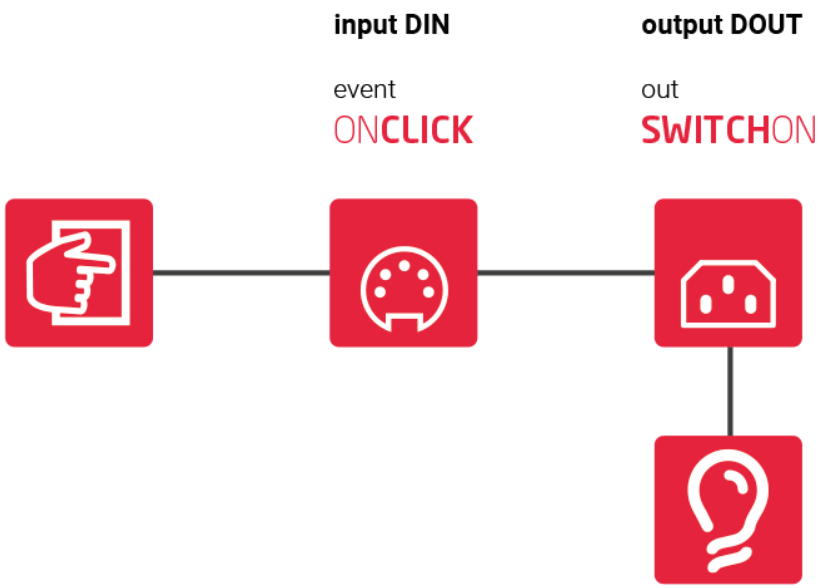
GRENTON system works on the basis of event driven model. Household members and their environment cause generation of events in the system, to which system reactions are connected, e. g. turning on the lamp in response to pressing a switch.

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

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

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

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



2. Features

2.1. Built-in features

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

2.2. User features

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

3. Methods

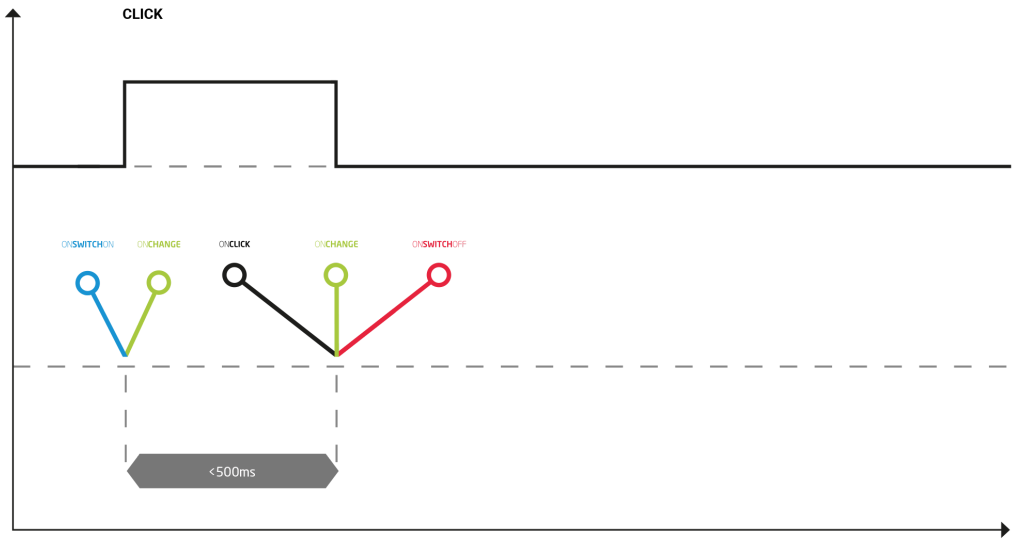
Methods are commands which can be given to a specific object. Each object has its own characteristic methods. For relay output, it can be methods SwitchOn and SwitchOff. Additionally, the methods can include required or optional parameters, which specify details of method invocation e. g. switch-on time.

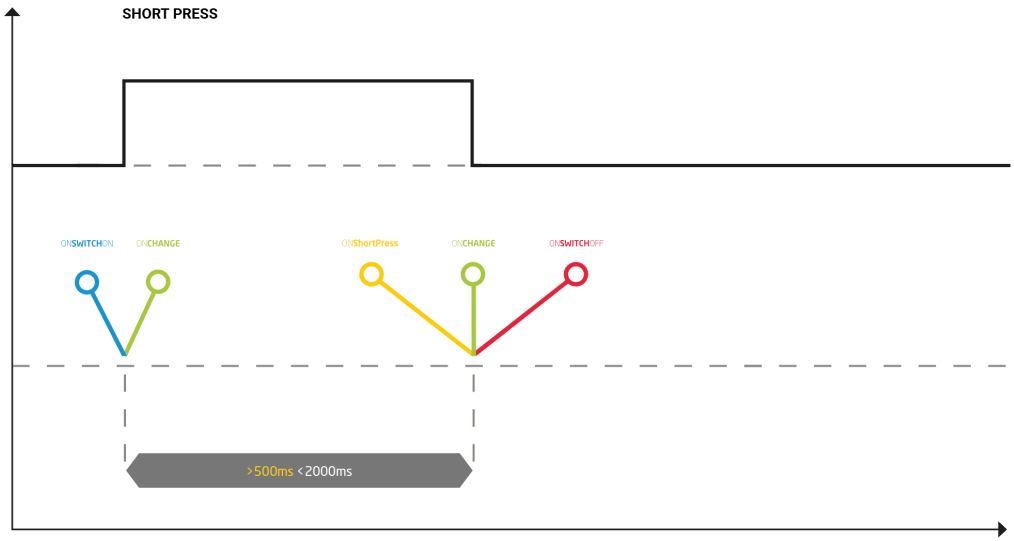
4. Events

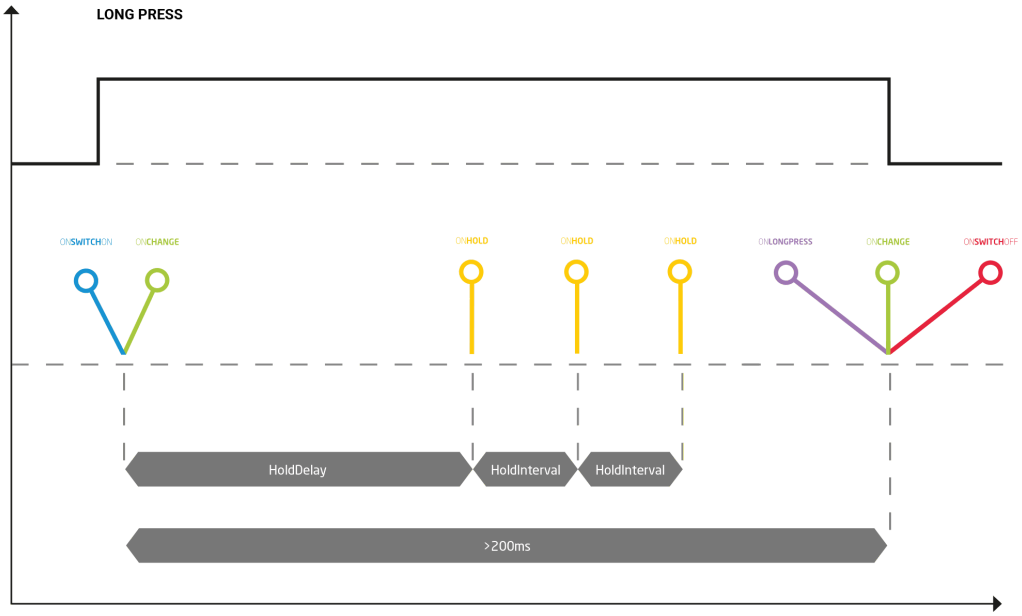
Events are elements of logical interface, invoked in reaction to the changes occurring in relation to an object (e.g. pressing a button, temperature change, etc.). We can connect one or more methods to each event, which will be invoked when an event occurs, e.g. when the button is pressed, the lights will be switched on. By connecting events of one object (mainly inputs, but sometimes also outputs) with methods of other object, we create logical configuration of the system. Each type of object (type of input/output) has its own list of events, which are invoked in a precisely specified way, depending on the actions performed by the user. For instance, binary input has the following list of events:

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

which are invoked according to the following scheme:







5. Features and methods addresses

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

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

Przykładowo: `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 each other using serial connection. They can also be connected to the field bus using taps. 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 diameter 0.5 mm 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.

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

C. Useful tips

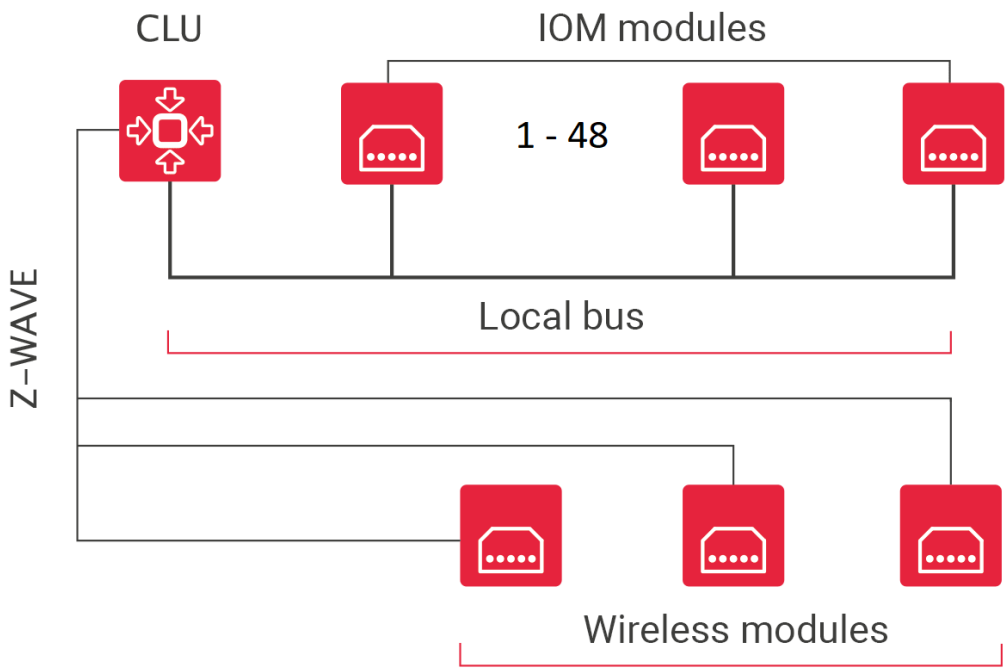
- Before implementing electrical system project, prepare smart house system project
- If you don't know yet, which devices will be controlled by the system, make sure the wiring reaches all possible places
- For light switches, any thin wire may be used, e.g. YTDY – it will allow saving on wire
- Remember 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 ofobjects - the system is fully scalable. Depending on scale and needs, there are several available configurations:

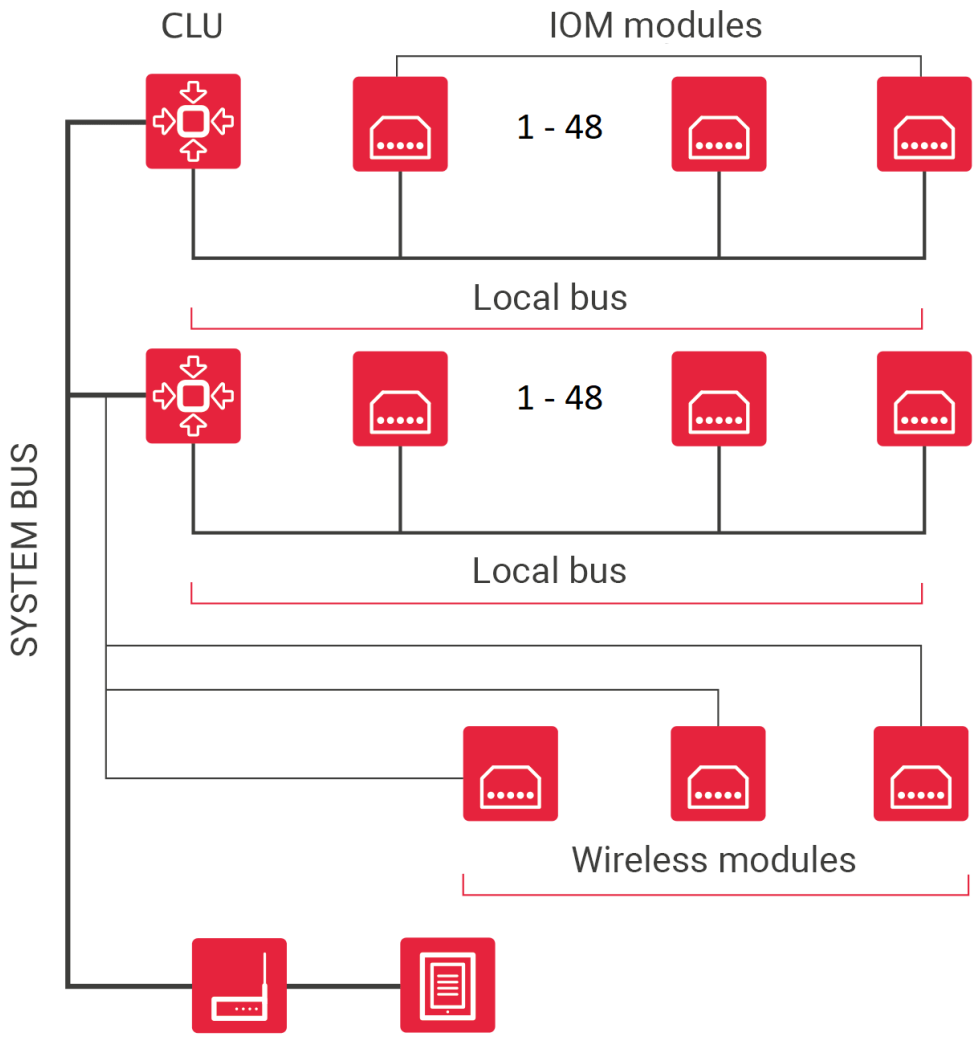
A. BASIC CONFIGURATION – CENTRALIZED SYSTEM WITH ONE CLU

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 128 inputs / outputs). Remember to provide the bus with power adequate to its load.



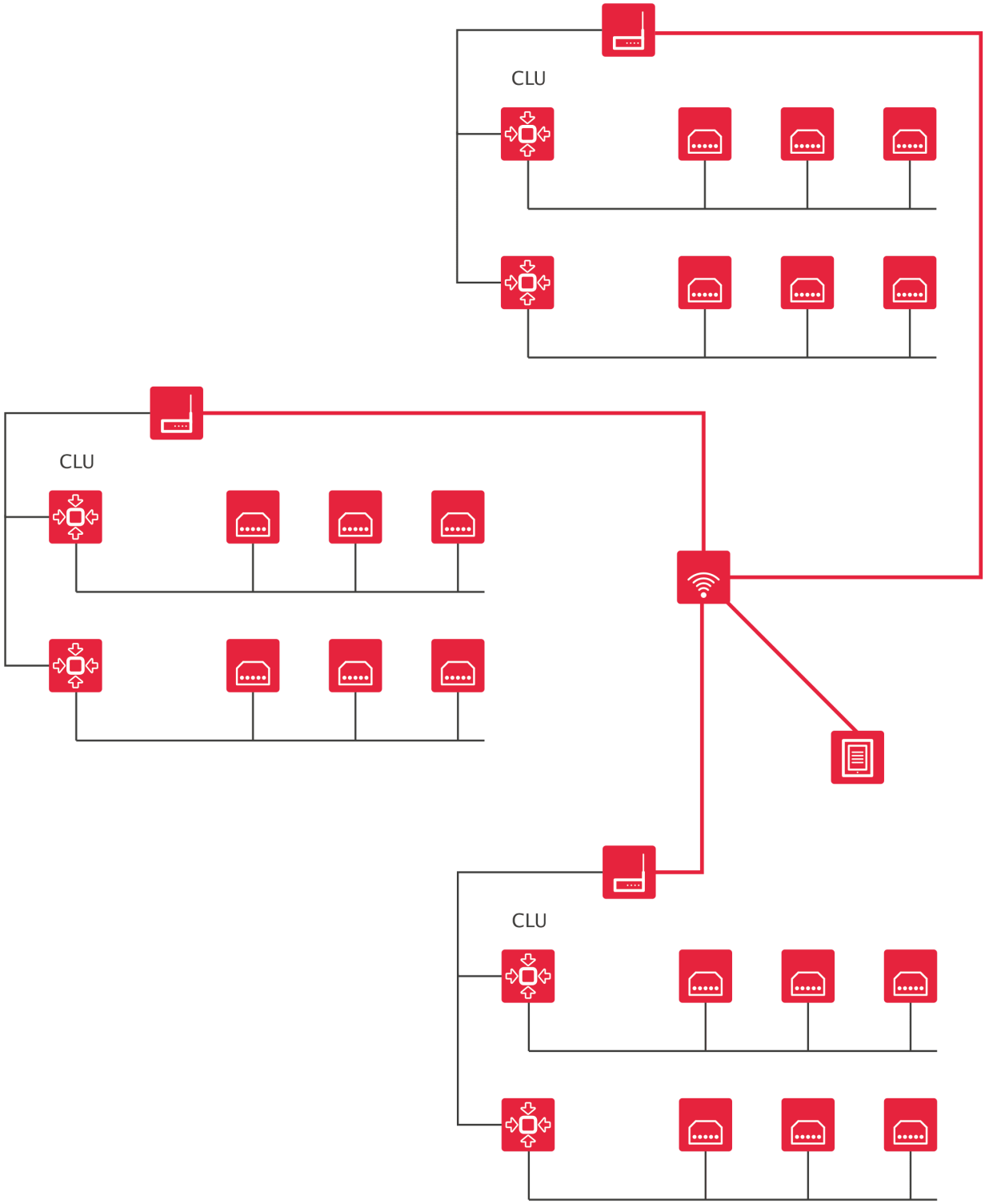
B. ADVANCED CONFIGURATION – TABLET-CONTROLLED DISTRIBUTED SYSTEM WITH MANY CLU

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

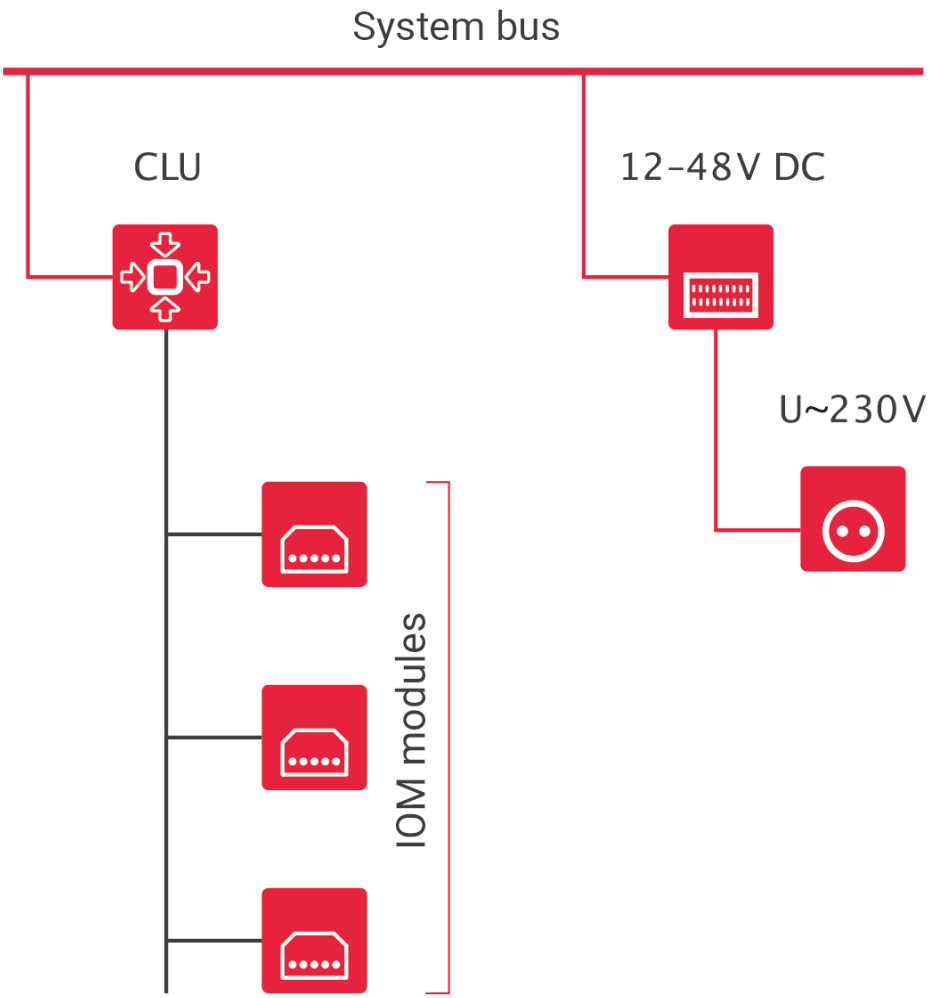
System expansion is practically unlimited. Several objects can be connected to one system. Thanks to that, you can have central control using only one system



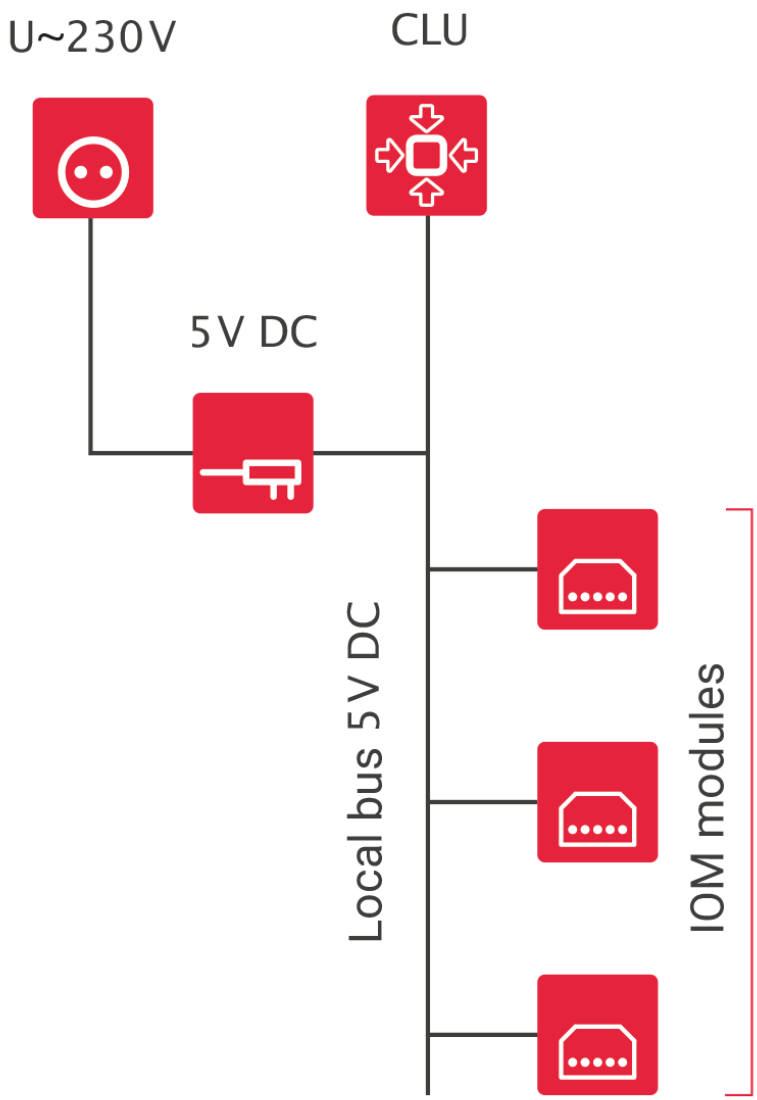
3. Modules power supply

A. THERE ARE THREE WAYS OF POWERING CLU AND IOM MODULES

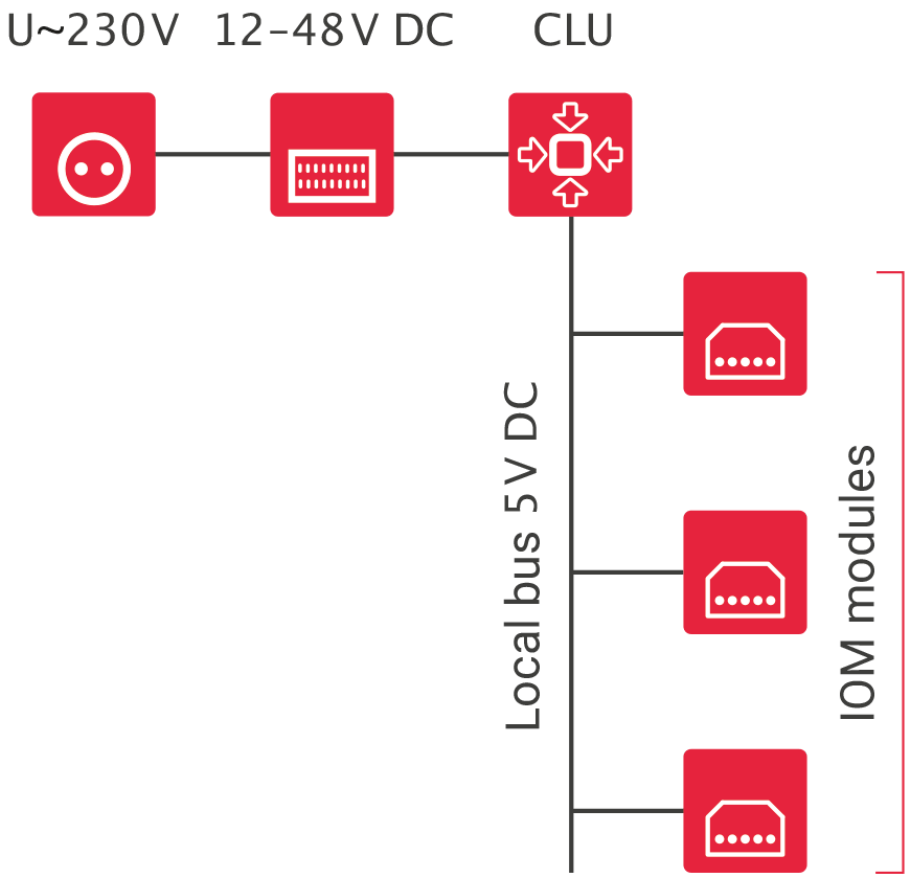
1. By connecting power supply to 12–48V DC system bus – in this case, CLU module will be powering IOM modules connected to it using field bus. The amperage of CLU built-in power supply is 1000 mA.



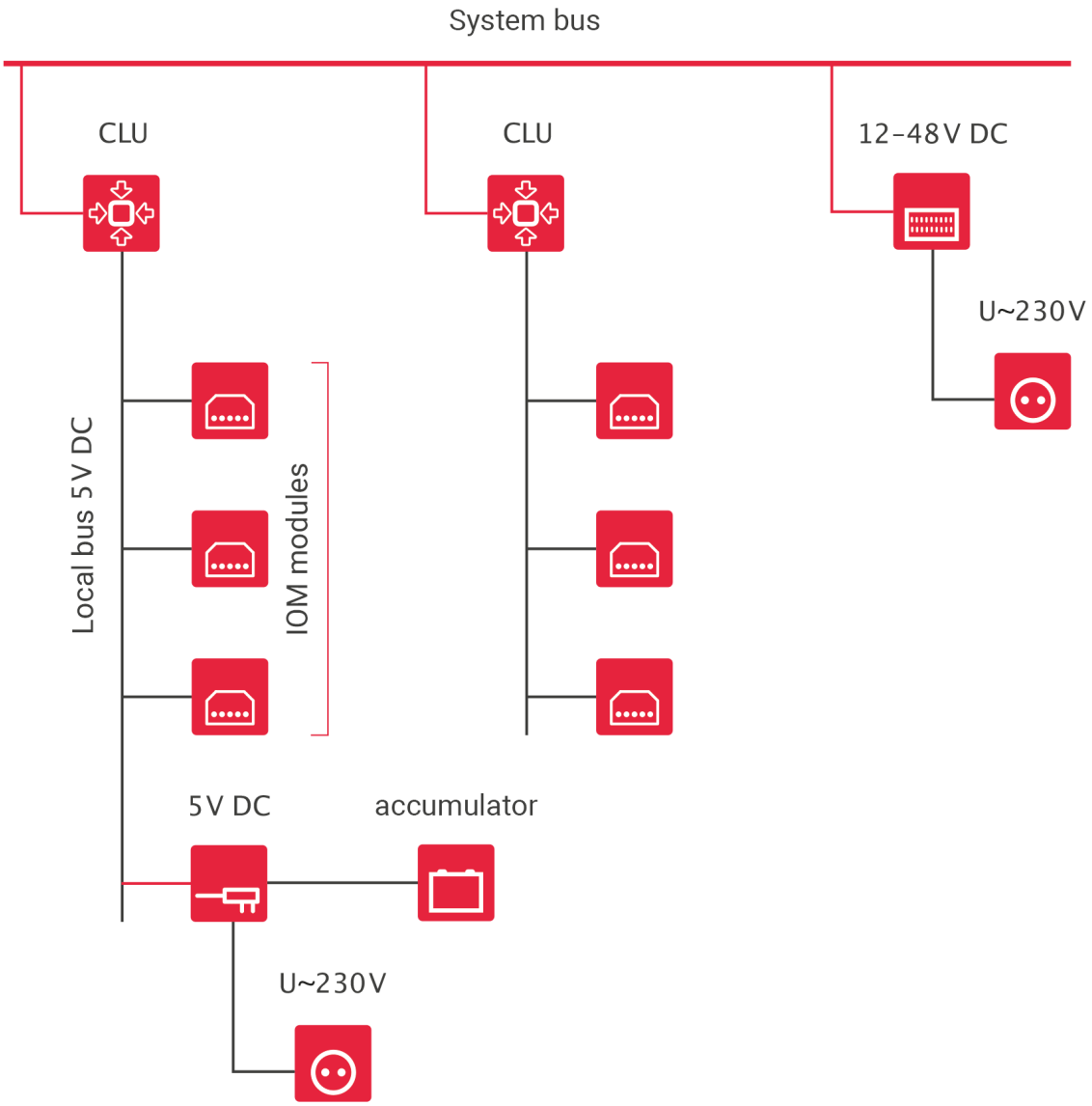
2. By connecting 5 V DC power supply to the field bus. In this case, CLU will be powered by the field bus.



3. By connecting 12-48V DC power supply to power outlet in CLU on the DIN rail. Similarly to point 1, CLU will be powering IOM modules using field bus



In the case of flush-mounted modules, there is a possibility of using an optional flushmounted 5V DC.



> **NOTE!** CLU may be simultaneously connected to power supply through system bus and field 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 switchiong 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.

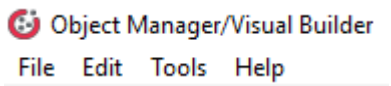
Current Object Manager version could be downloaded from https://www.grenton.com/wsparcie/materialy-do-pobrania.html

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

2. Menu structure

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

- **MAIN MENU**

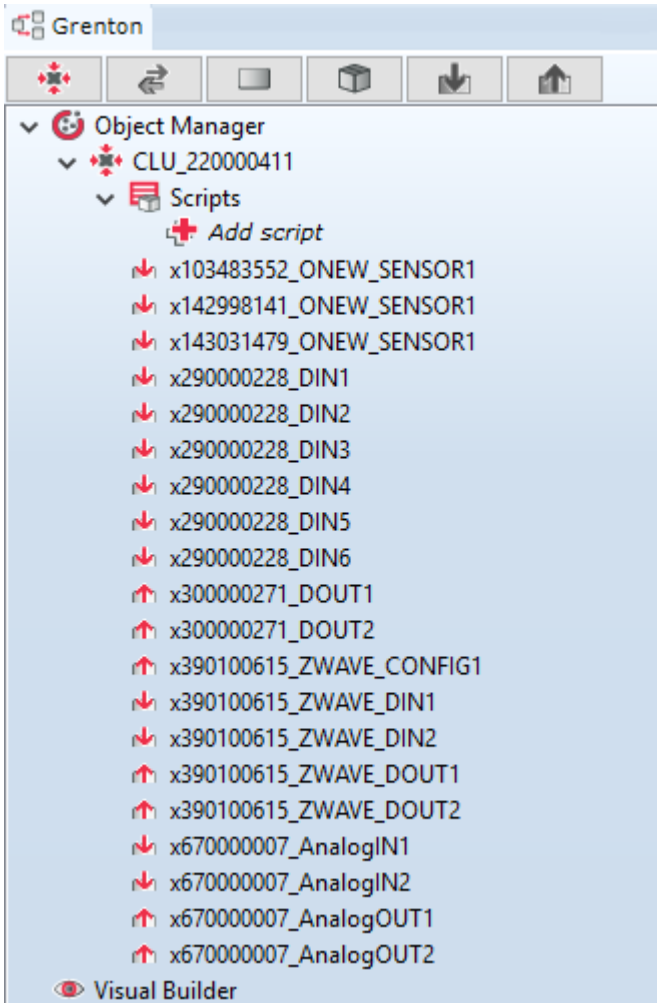


Zawiera podstawowe komendy służące do obsługi projektu.

- **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 two parts: Object (CLU, inputs, outputs) list and Visual Builder.

Składa się z dwóch części: listy obiektów (CLU, wejść, wyjść) oraz Visual Buildera.

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.

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`

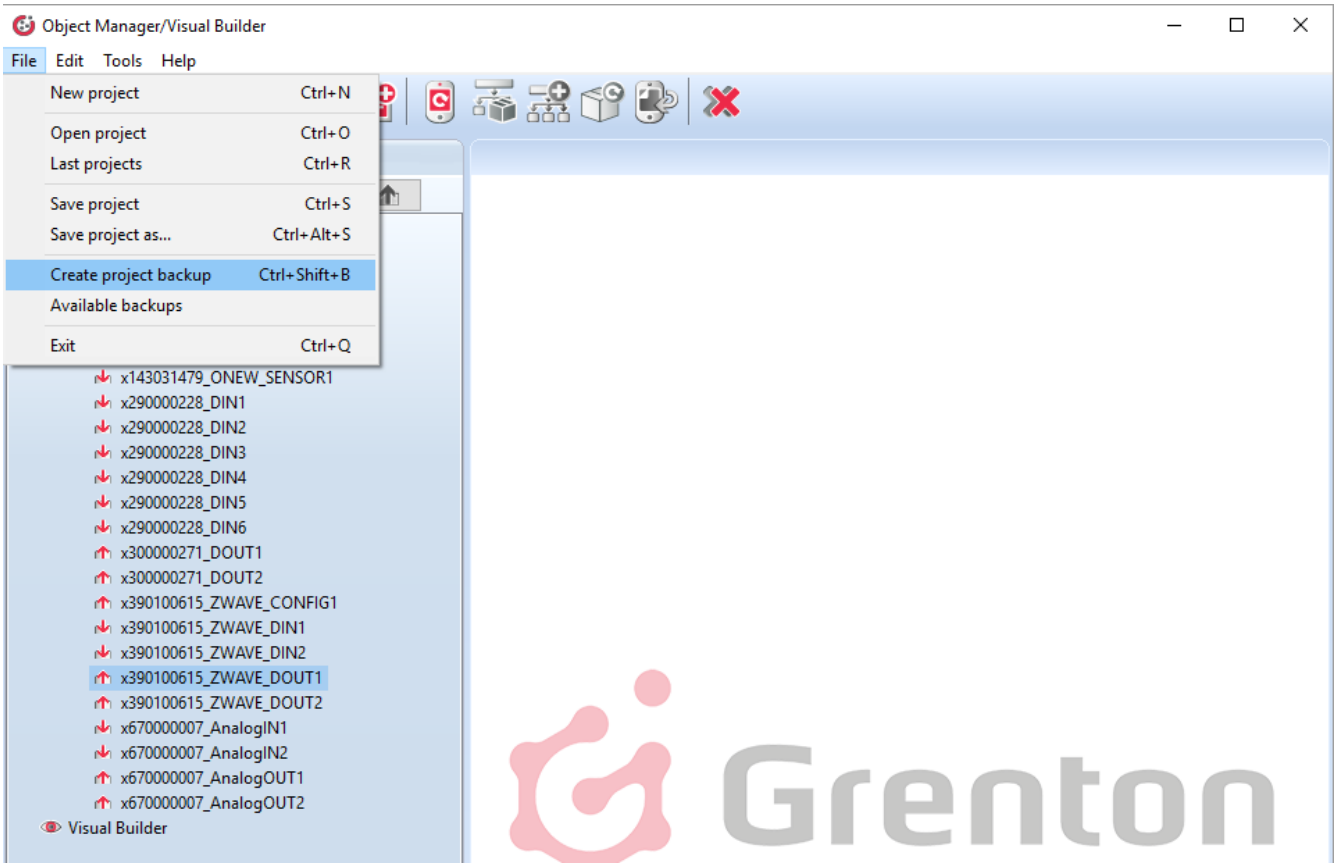
All files of created and saved projects are saved in the catalogue with `*.omp` extension. (e.g. `projekt.omp`).

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`).



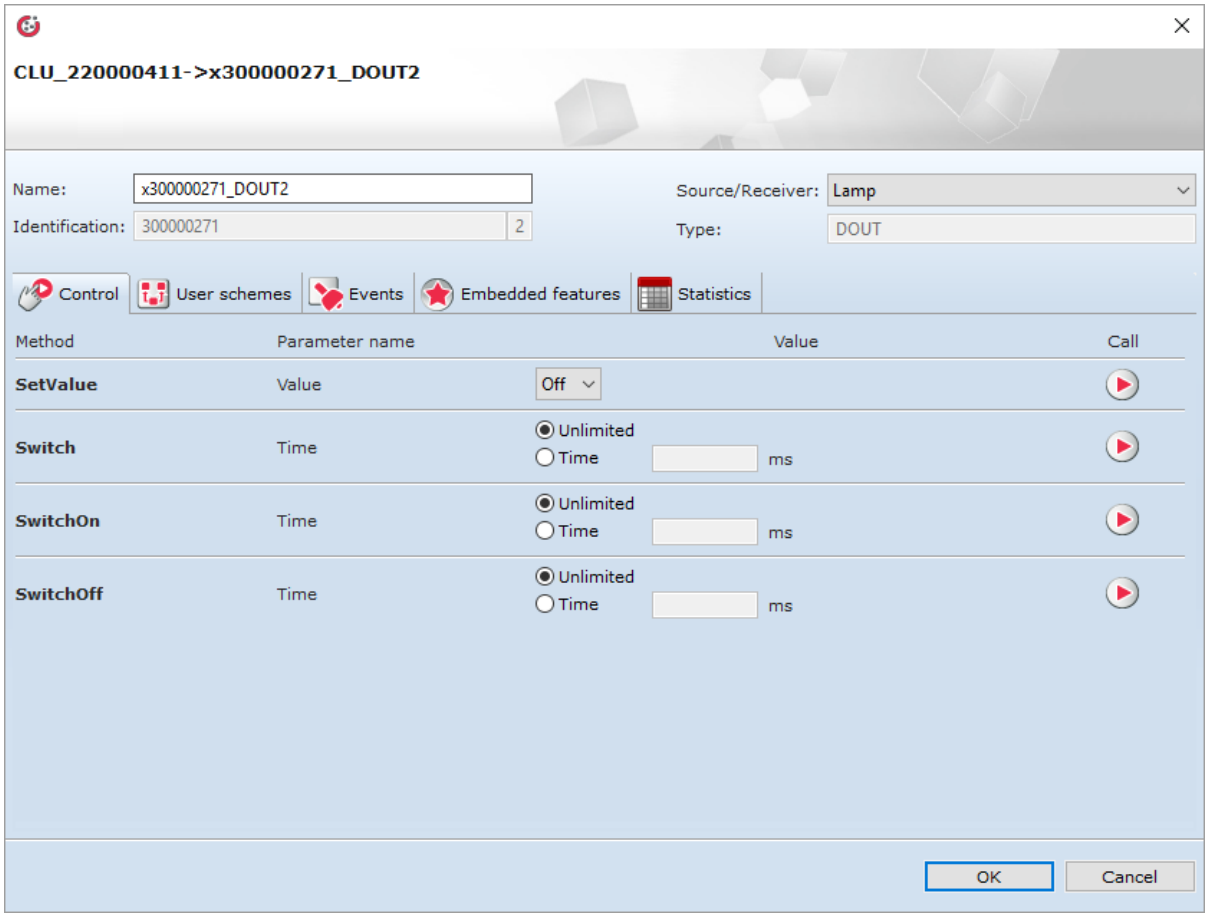
Saved backups are available in the list opened by clicking `Available backups`, or in the project opening window in a tab `Backups`.

> **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

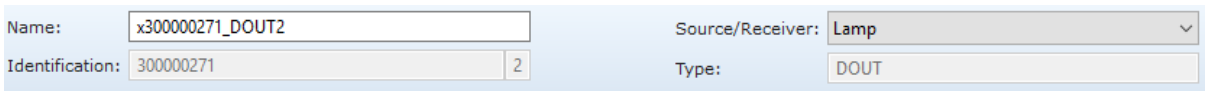
4.1. Objects configurator

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.



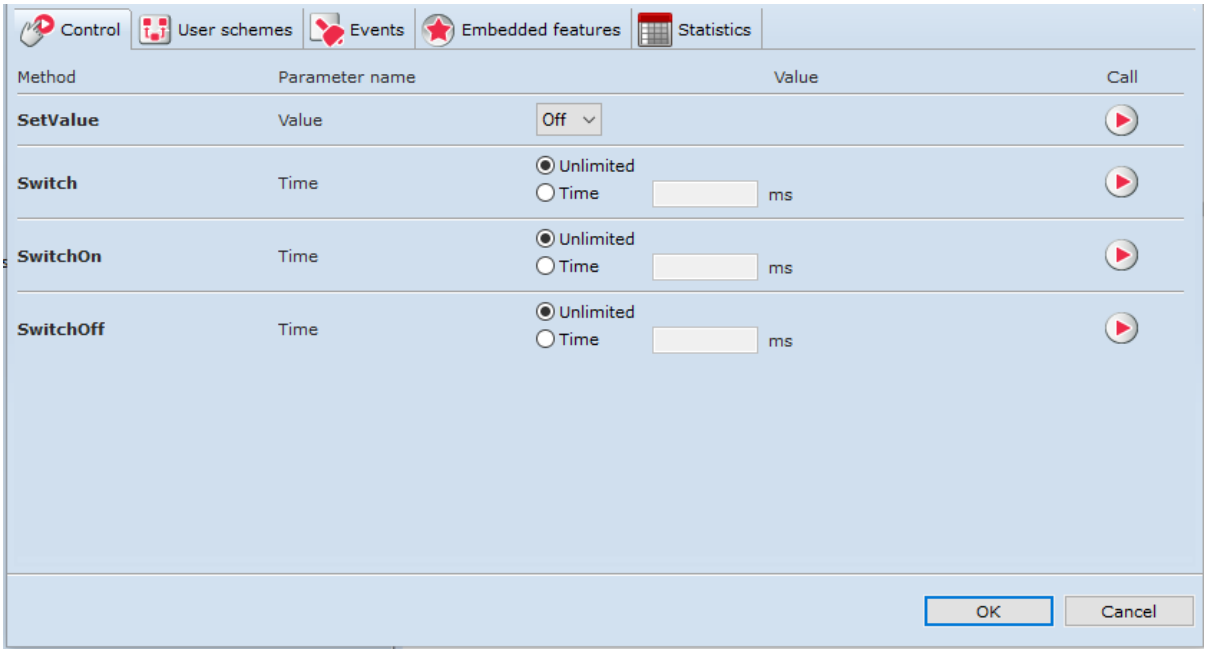
The form above consists of the following sections:

1. Basic information



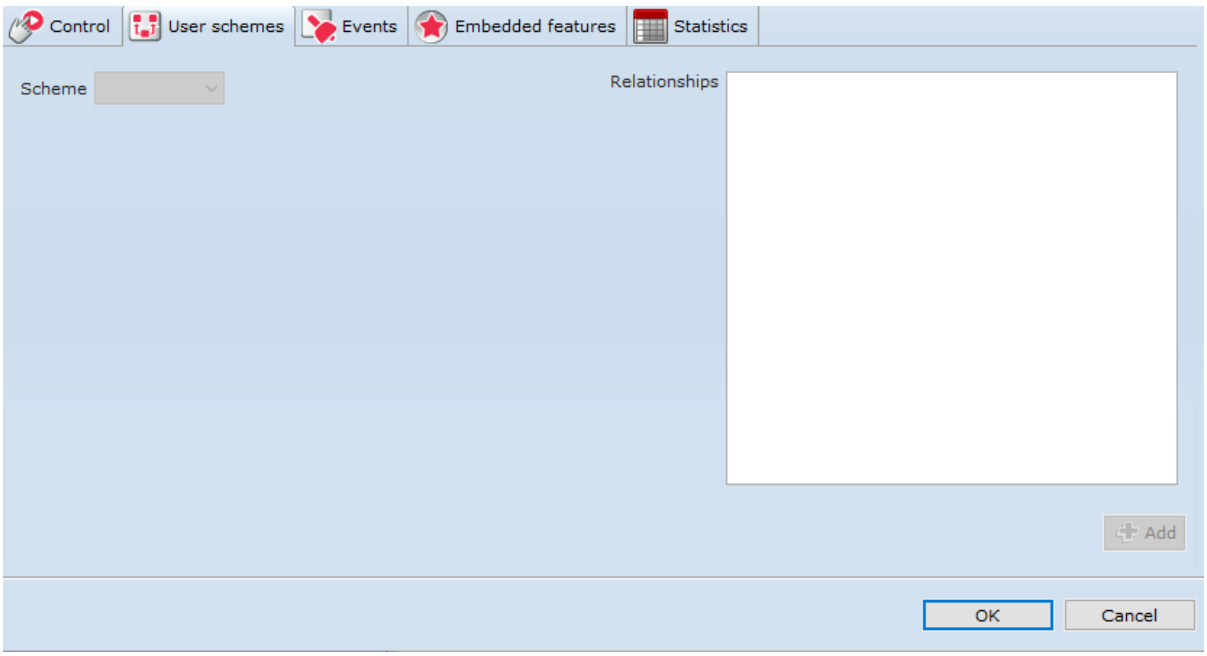
This section is located in the upper part of the form and contains basic information on each object, e. g. IP address, name, module type, serial number, input / output number within the module. In this section the user can also define type of source or receiver, physically connected to the object.

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. Configuration chart

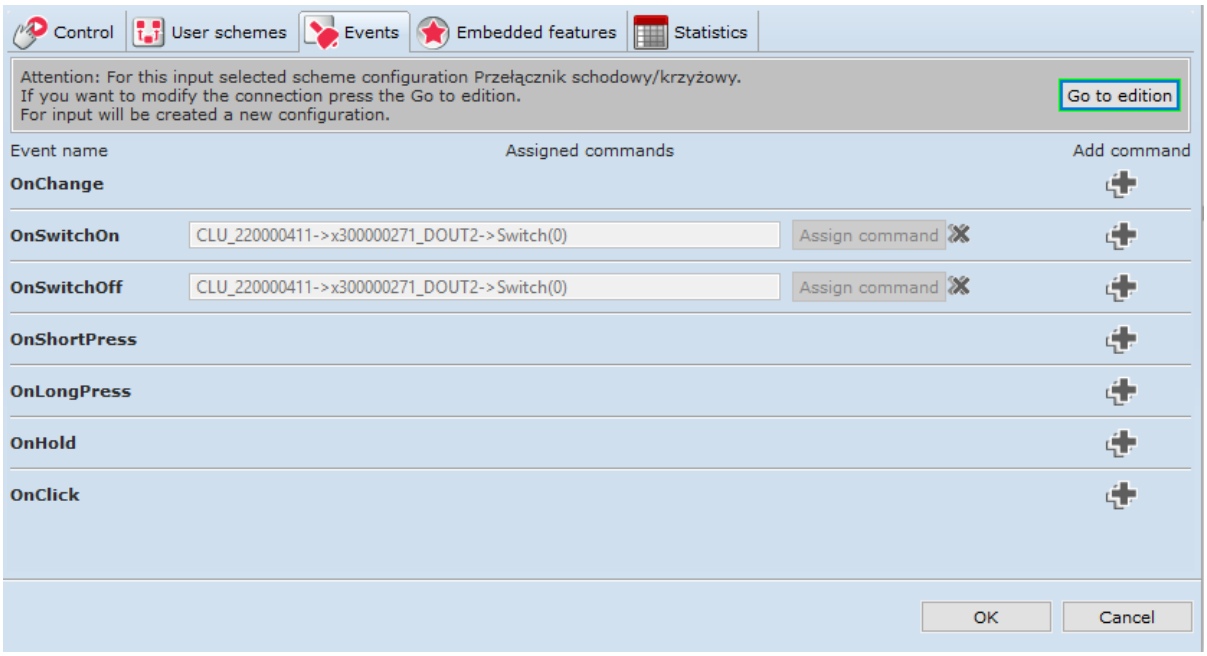


Configurations charts define object behaviour and allow simplified logic configuration. After selecting configuration chart for the input and adding objects connections, Object Manager will automatically create connections between appropriate events and connected objects methods.

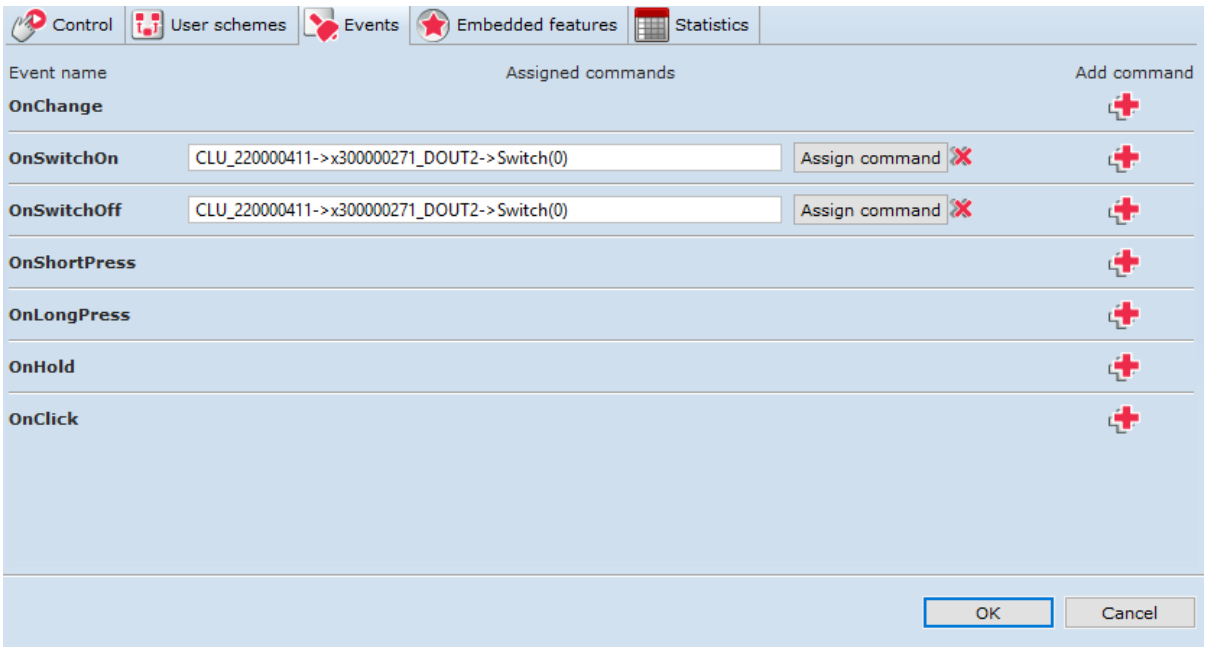
If the user created individual event-method connections using `Events` tab, they are visible on the list as `User chart`.

4. Events - tab description

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



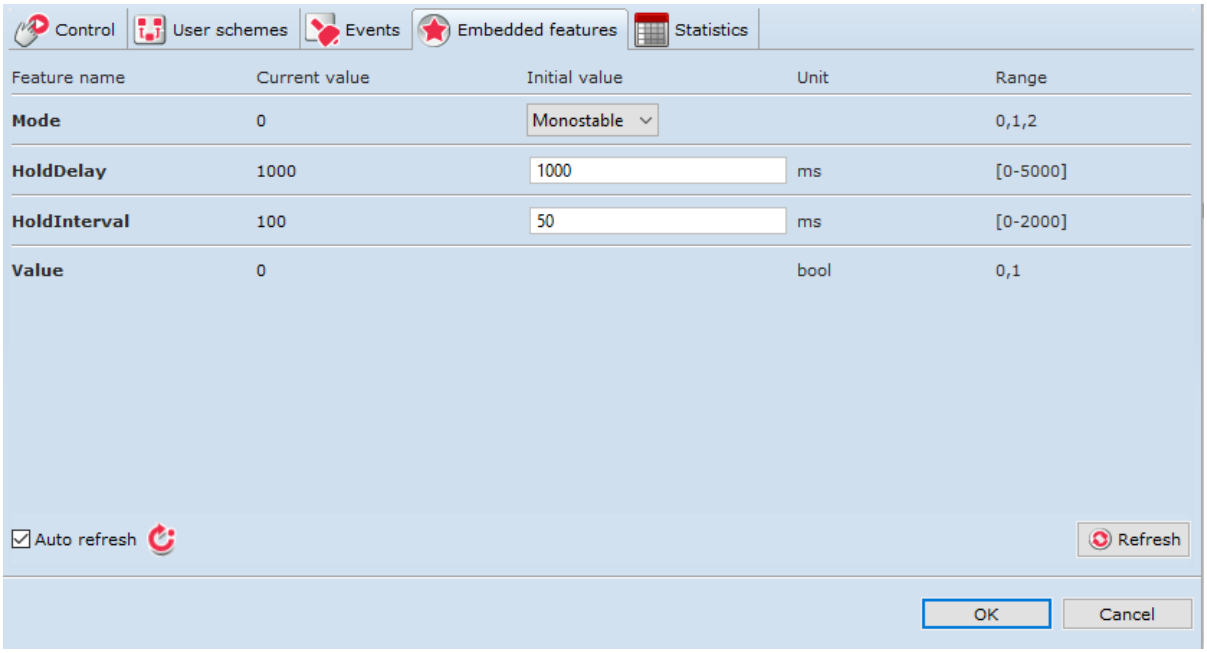
You can go to event-method connections edition any time by clicking "Go do edition". In this case, "user chart" will be created, which will show up on the list in the configurations charts tab.



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.

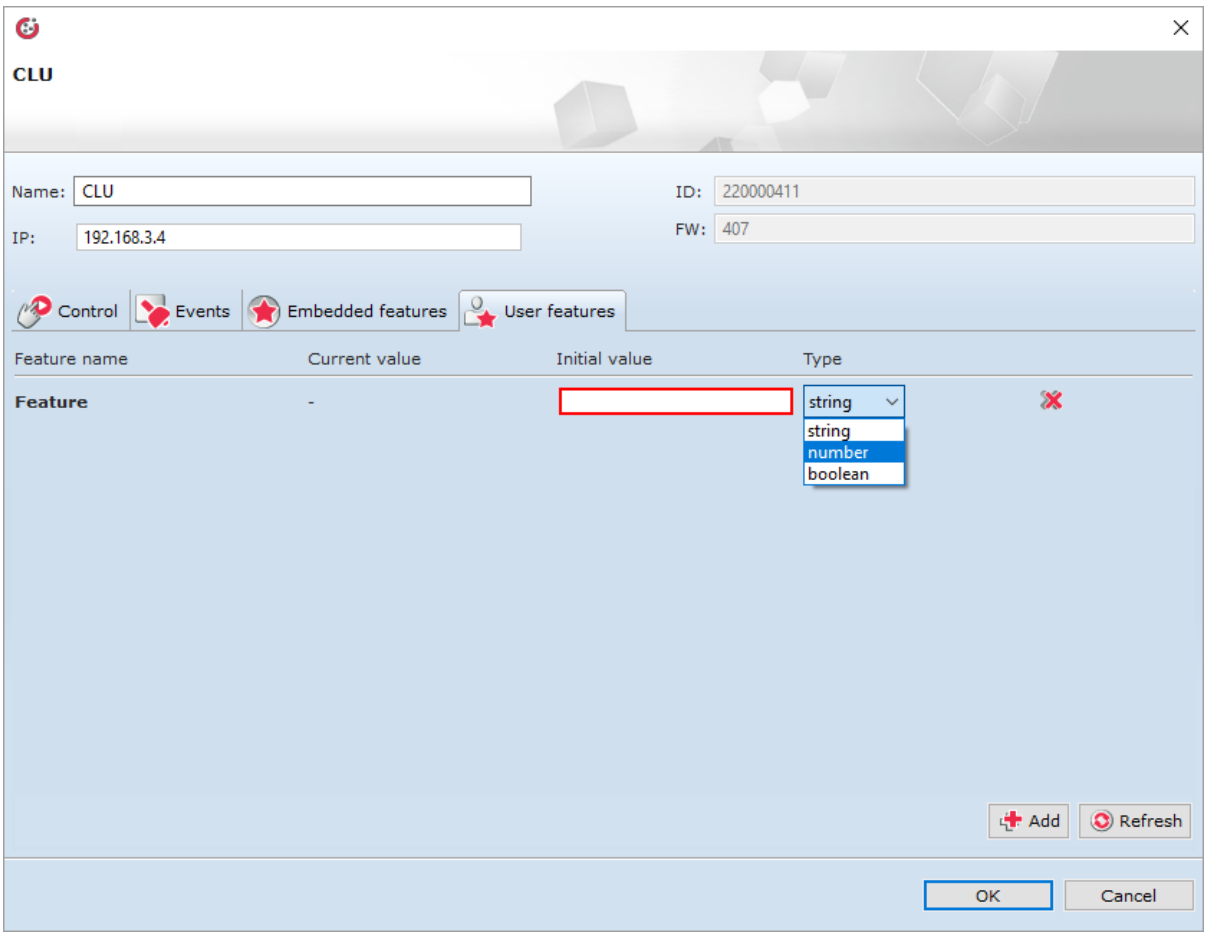
5. Built-in features

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.



6. User features (only CLU)

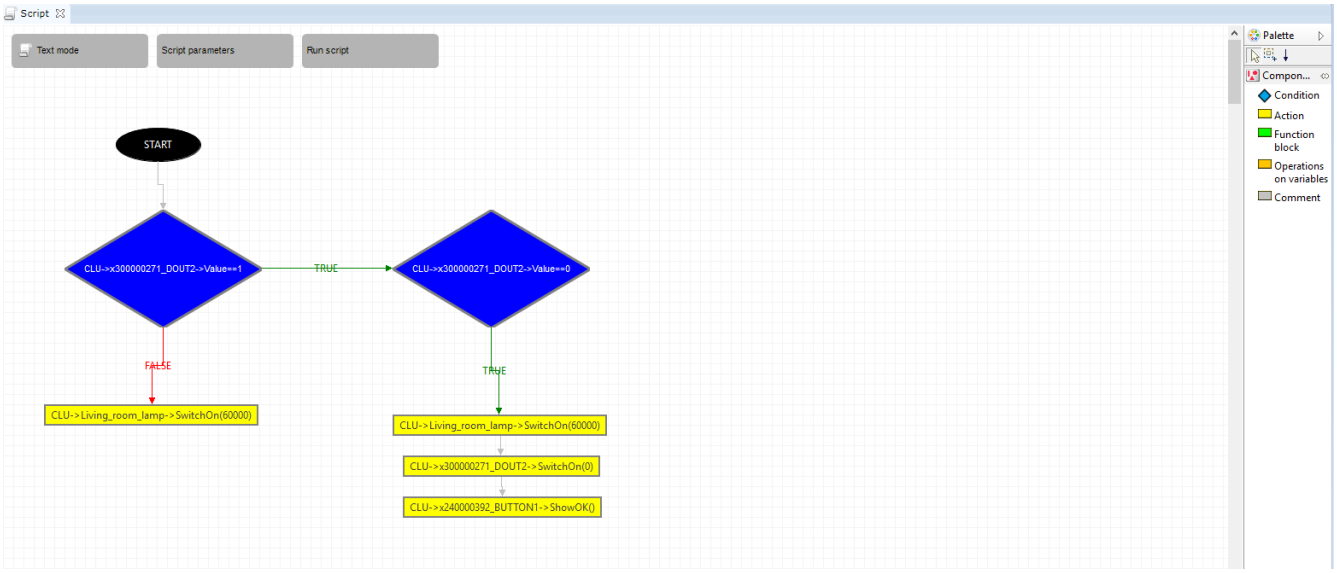
The tab allows user to define in the CLU his own list of features, which then can be used to store various type of data (counters, markers). Adding user feature happens after clicking the "add" button and entering feature name. Then, feature's initial value and type (text, numeric, boolean) have to be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



4.2. Script builder

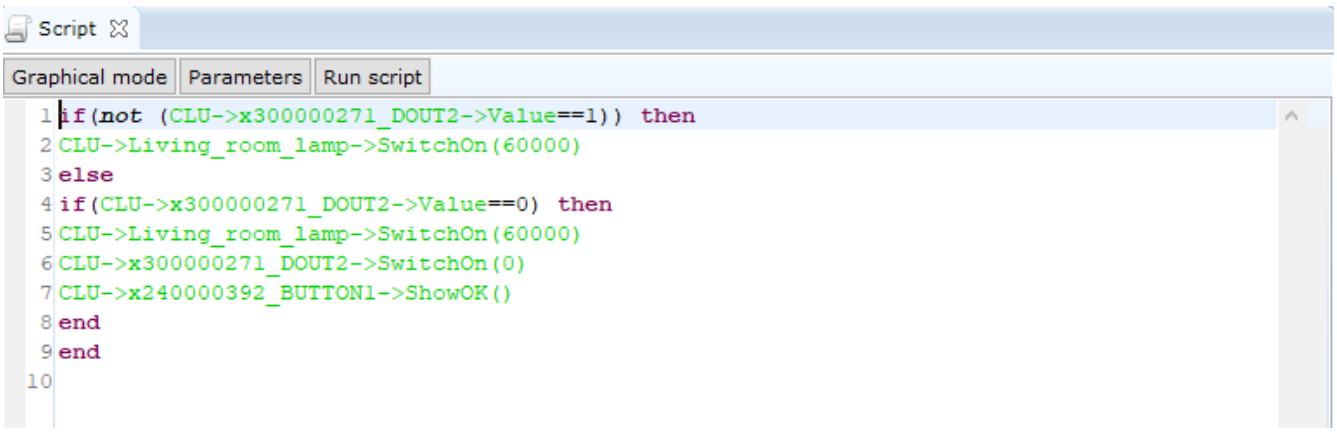
It's a tool for script creation, which can work in two modes:

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

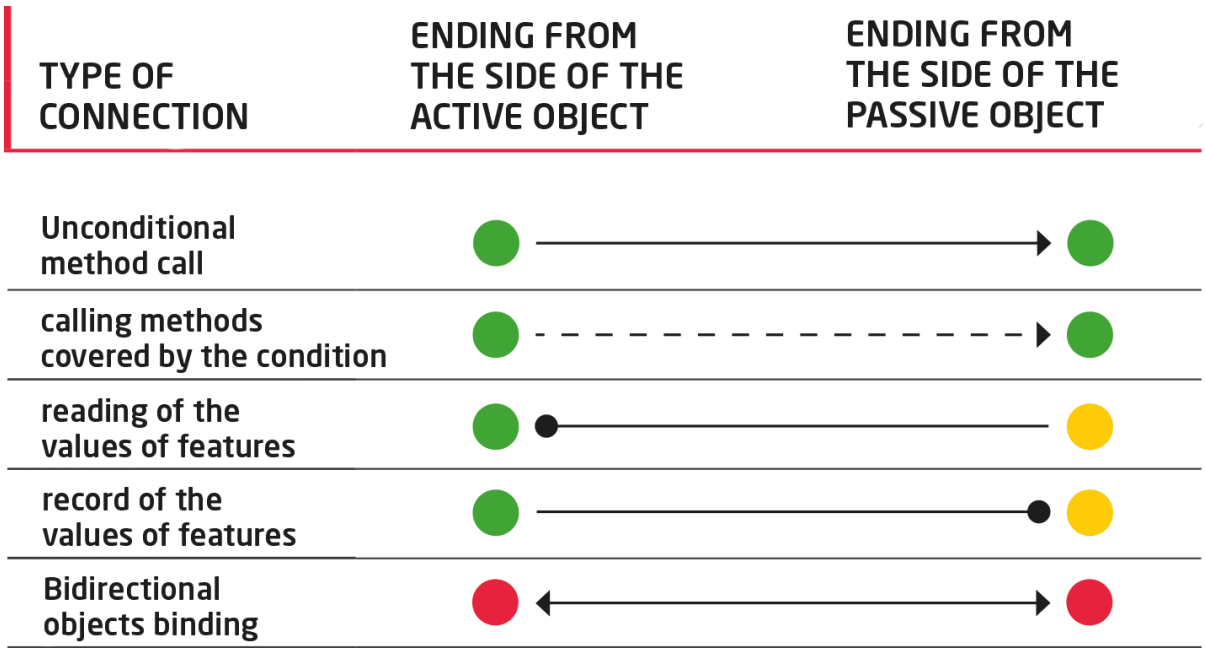
A tool showing dependencies and connections between all objects in the system. Thanks to that tool, you can easily and quickly find a dependency that interests you, or check dependencies of a specific module without going through configurations.

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.

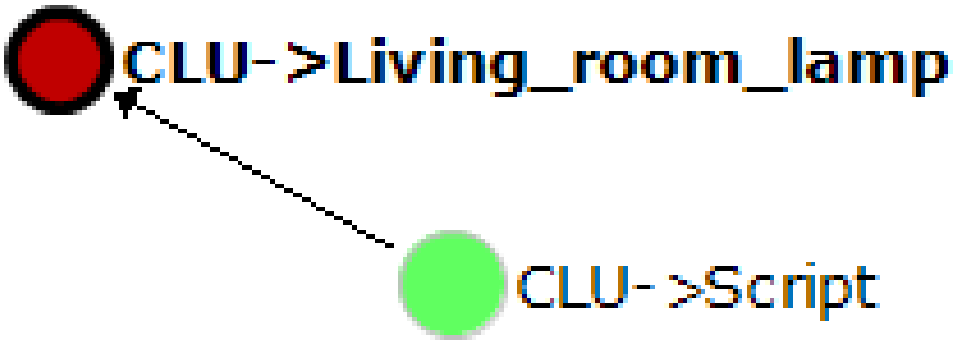


Connections are displayed on three different levels:

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.

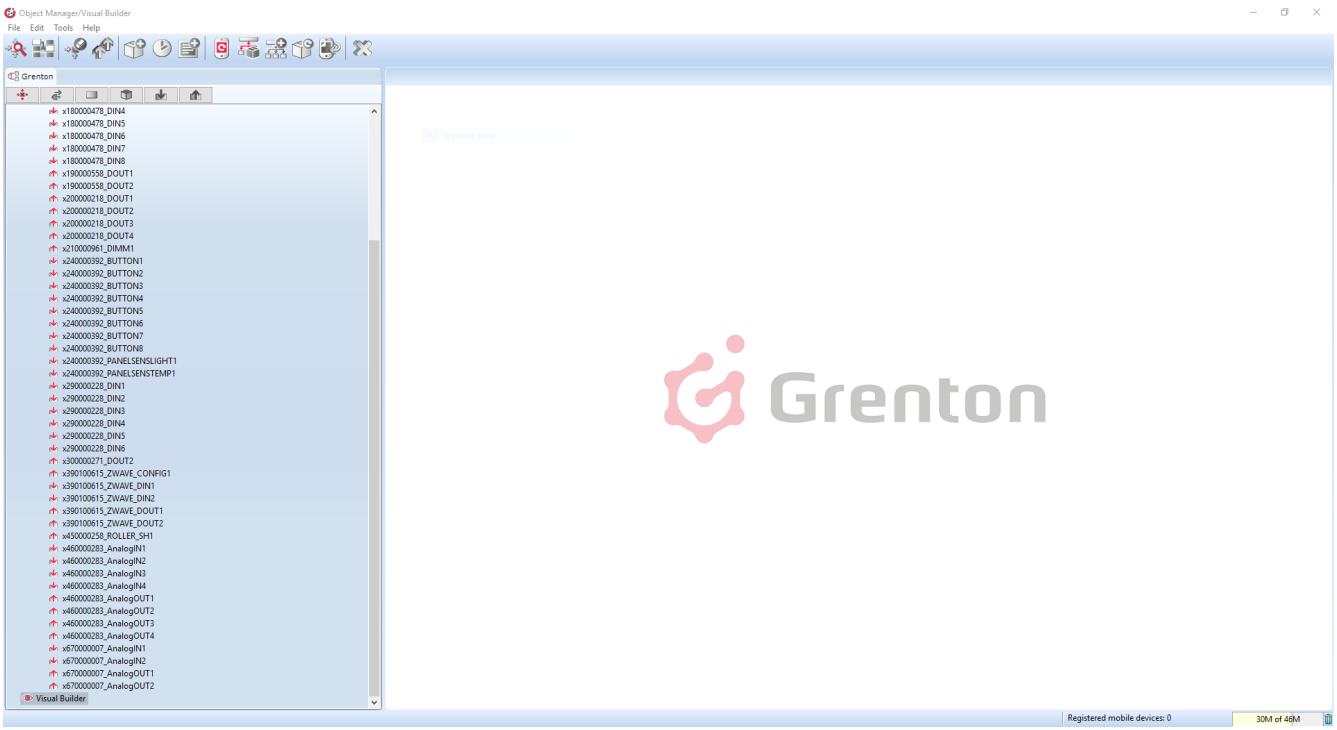
Navigation also happens in two planes:

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

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. Bin

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

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

Objects can be irretrievably removed from the bin by selecting "delete" from the context menu.

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

VI. Basic system configuration

1. Connecting OM to CLU

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

There are two connection methods:

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

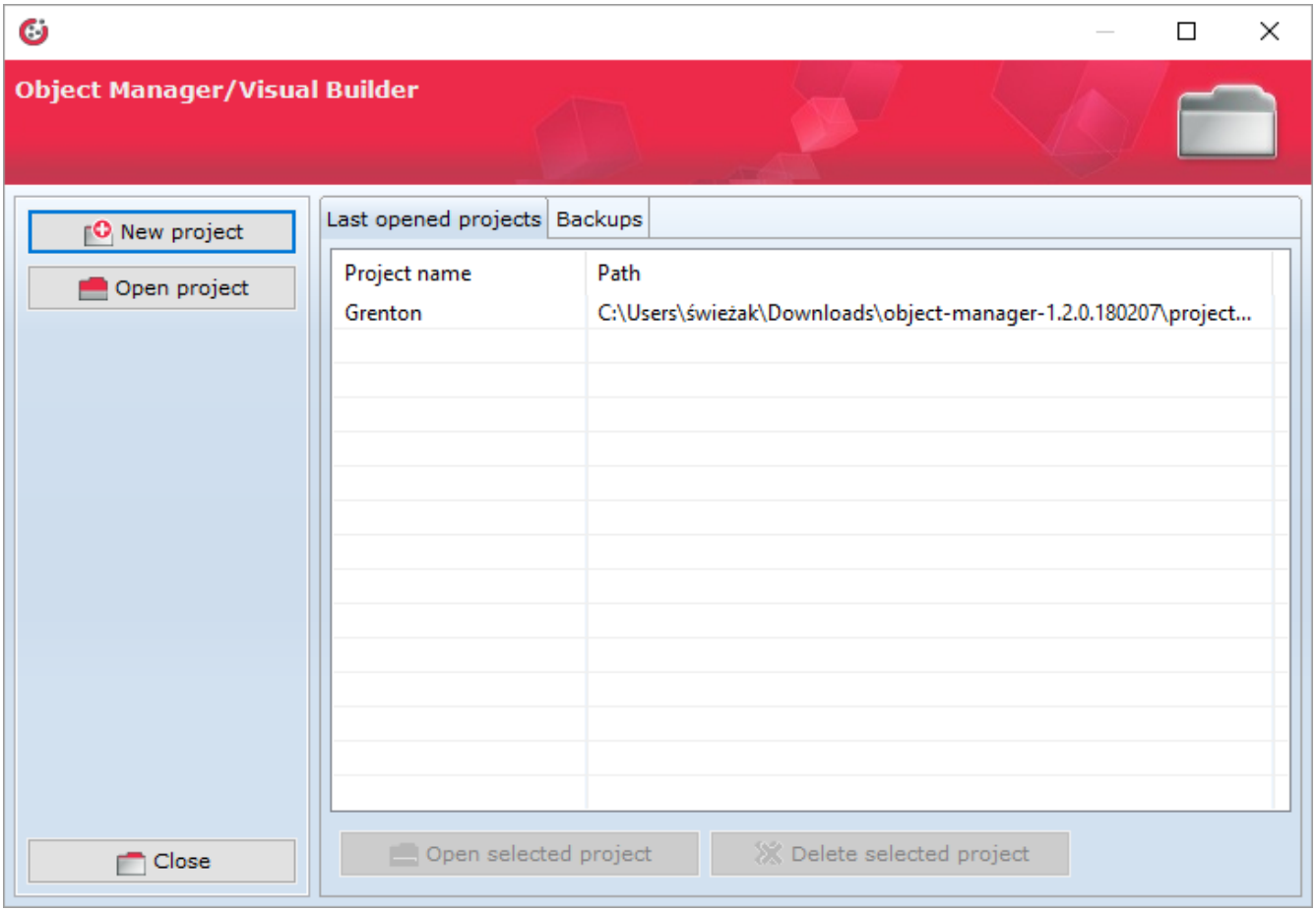
2. IP adressess

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

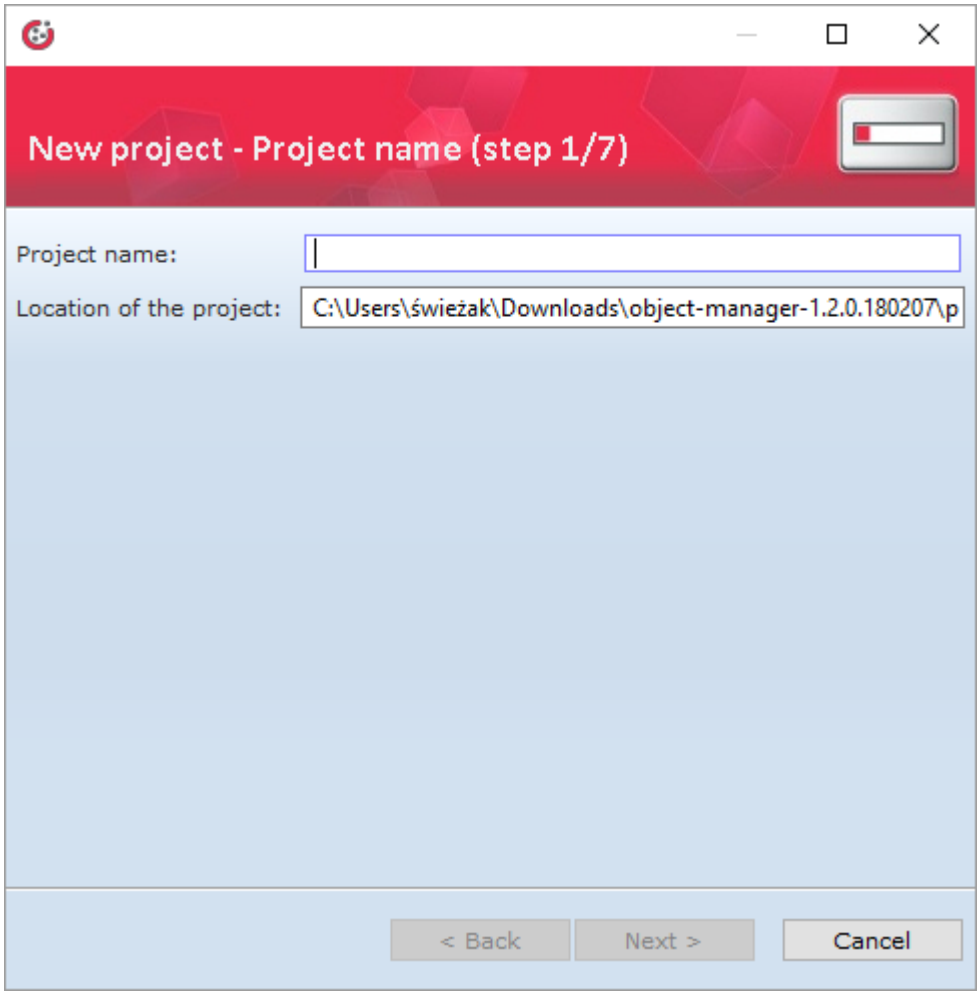
> **NOTE!** After connecting CLU (or several CLU) to computer's network card, it will receive a new IP address consistent with pool of addresses in which computer's network card is.

3. Opening new project

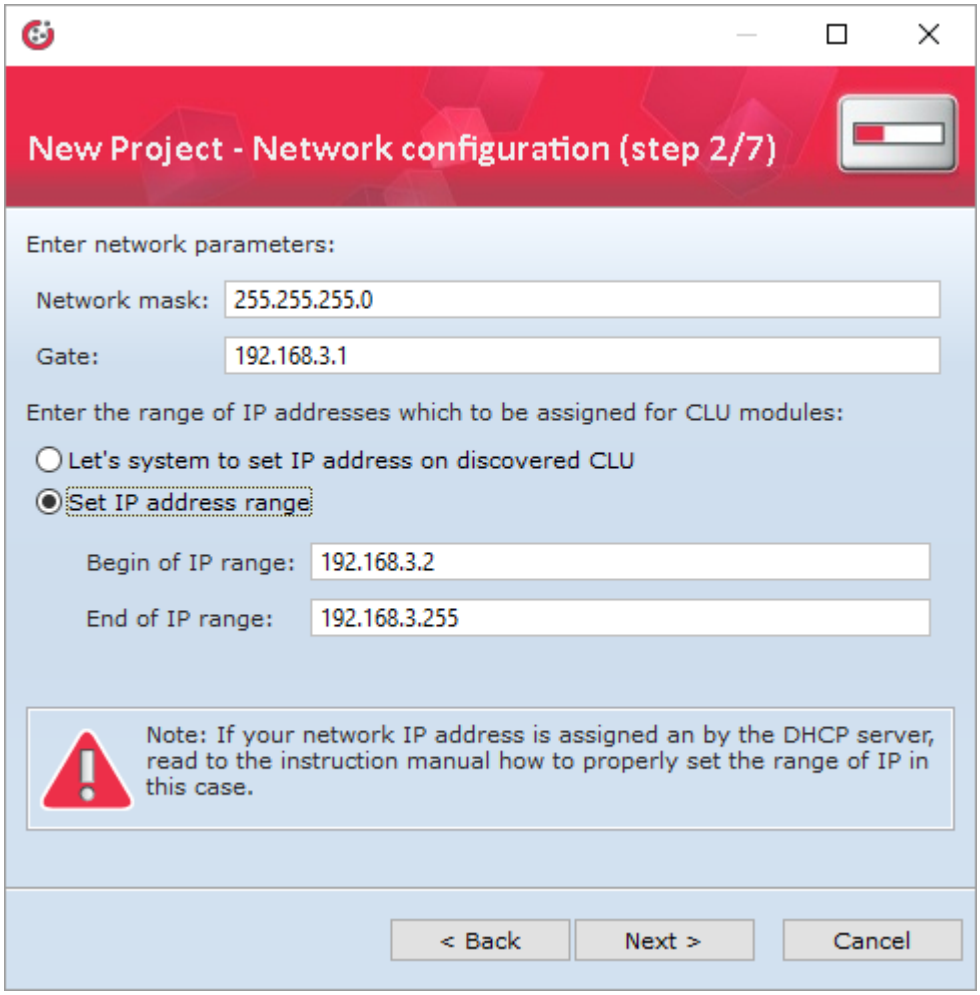
After opening Object Manager, a new window with two options appears: opening a saved project and creating a new project.



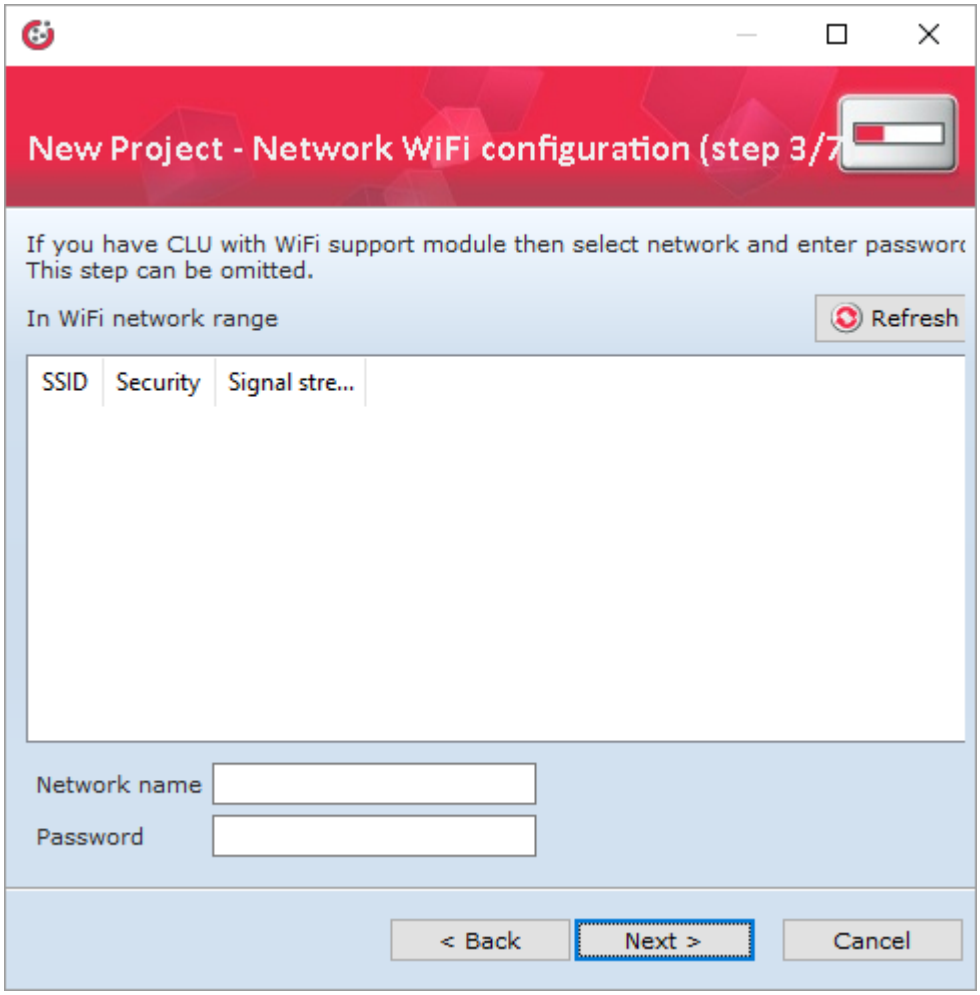
1. Select new project creation, then name the created project.



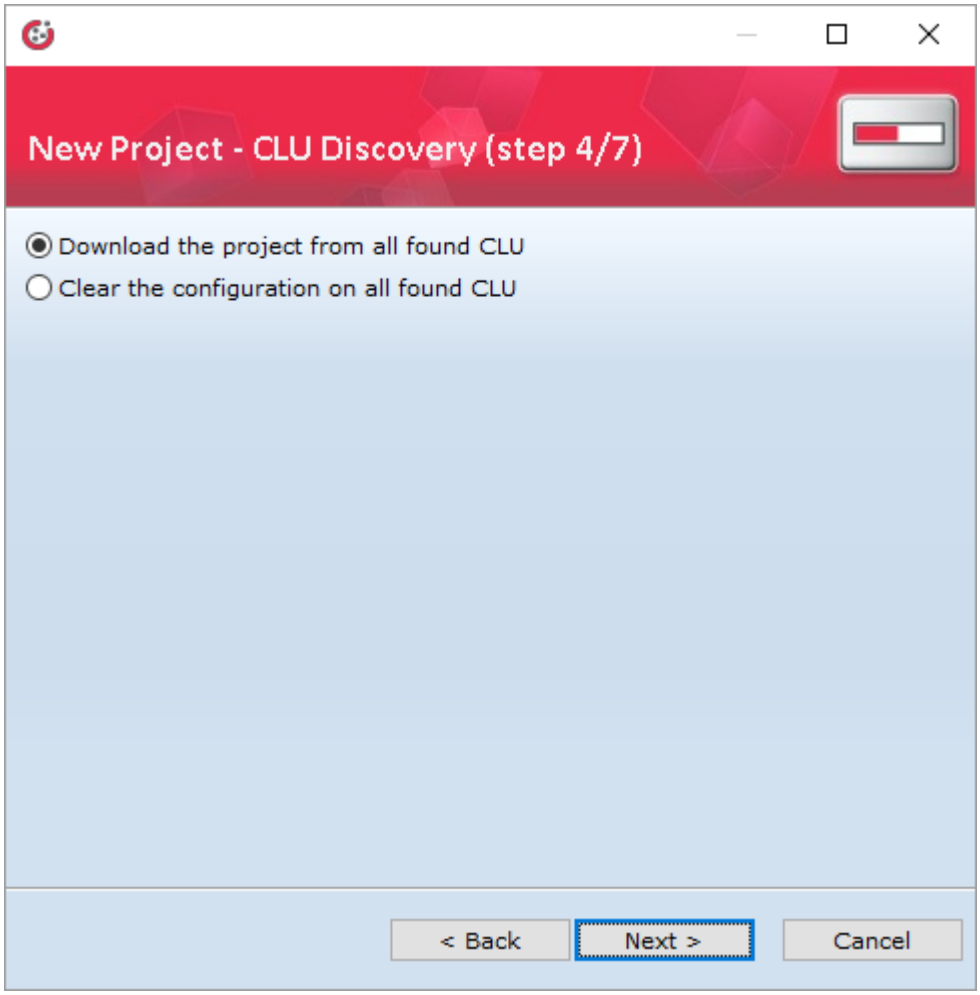
2. Object Manager software will display network configuration window, in which you need to specify range of available IP addresses. You can also allow the system to give IP address to CLU automatically.



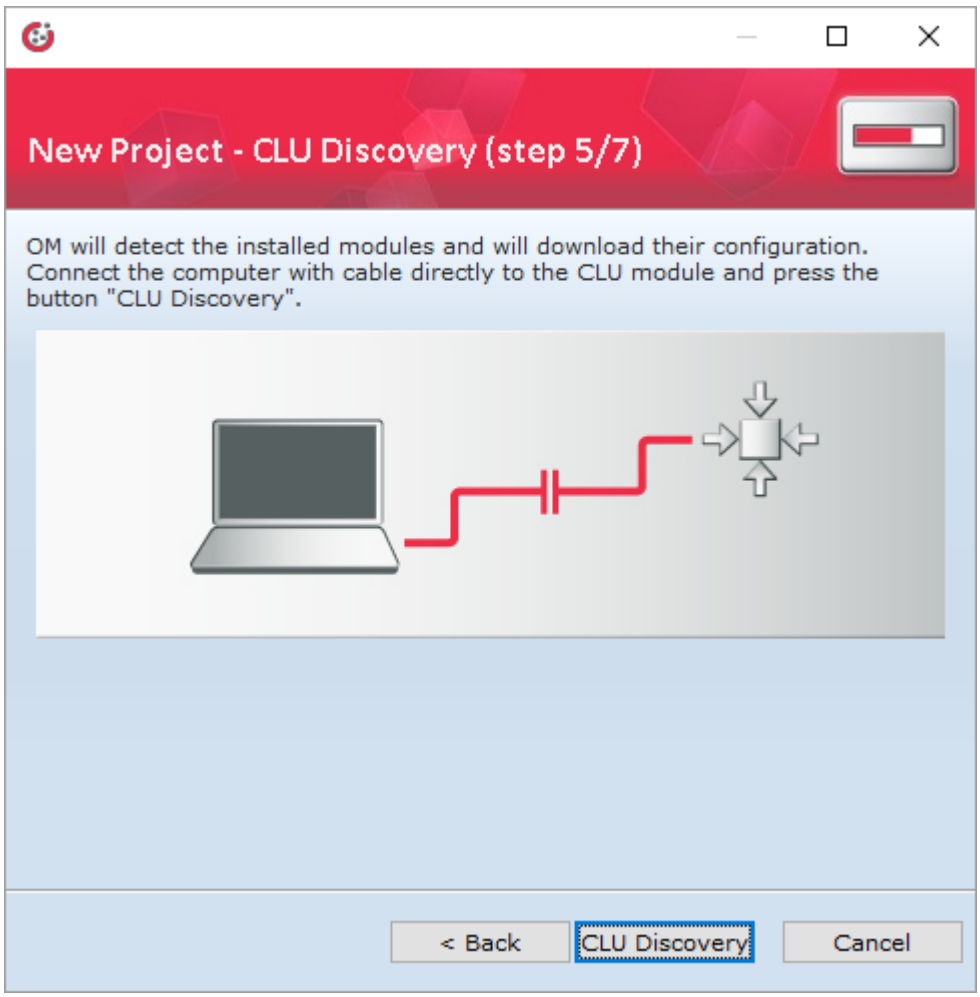
3. In the next window you will see the step for *WiFi network configuration*, which should be omitted.



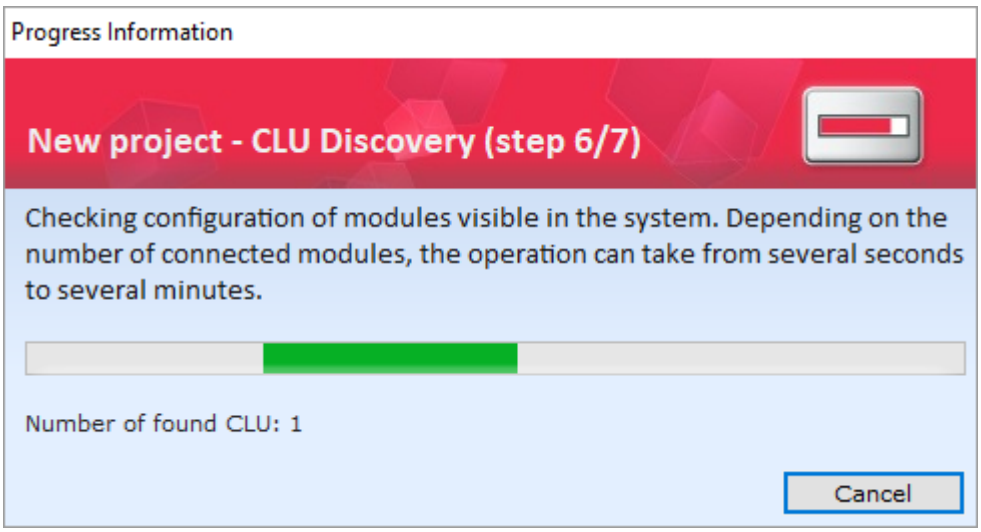
4. In the fourth step, you may choose between downloading existing system configuration to the newly created project, and complete configuration reset and starting a project from scratch. The first option is useful when necessity of recreating configuration after loss of project file occurs.



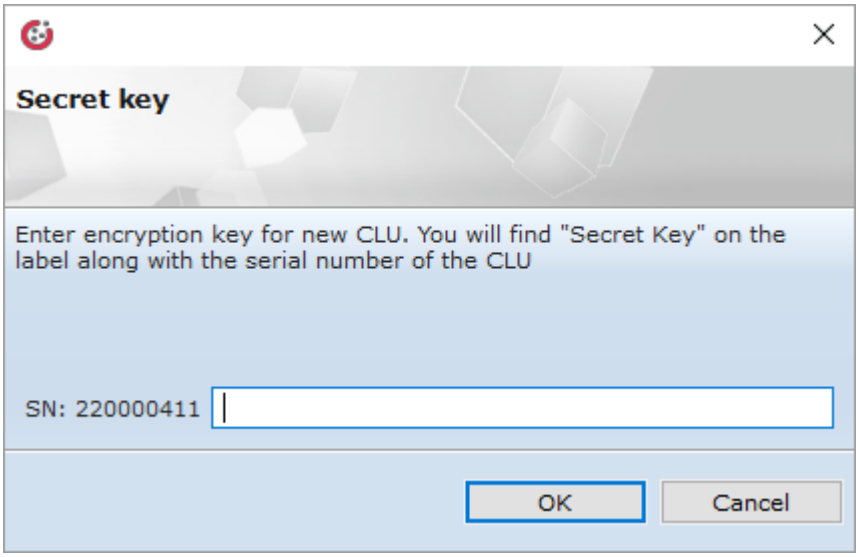
5. In next step, available modules search procedure named CLU DISCOVERY should be launched.



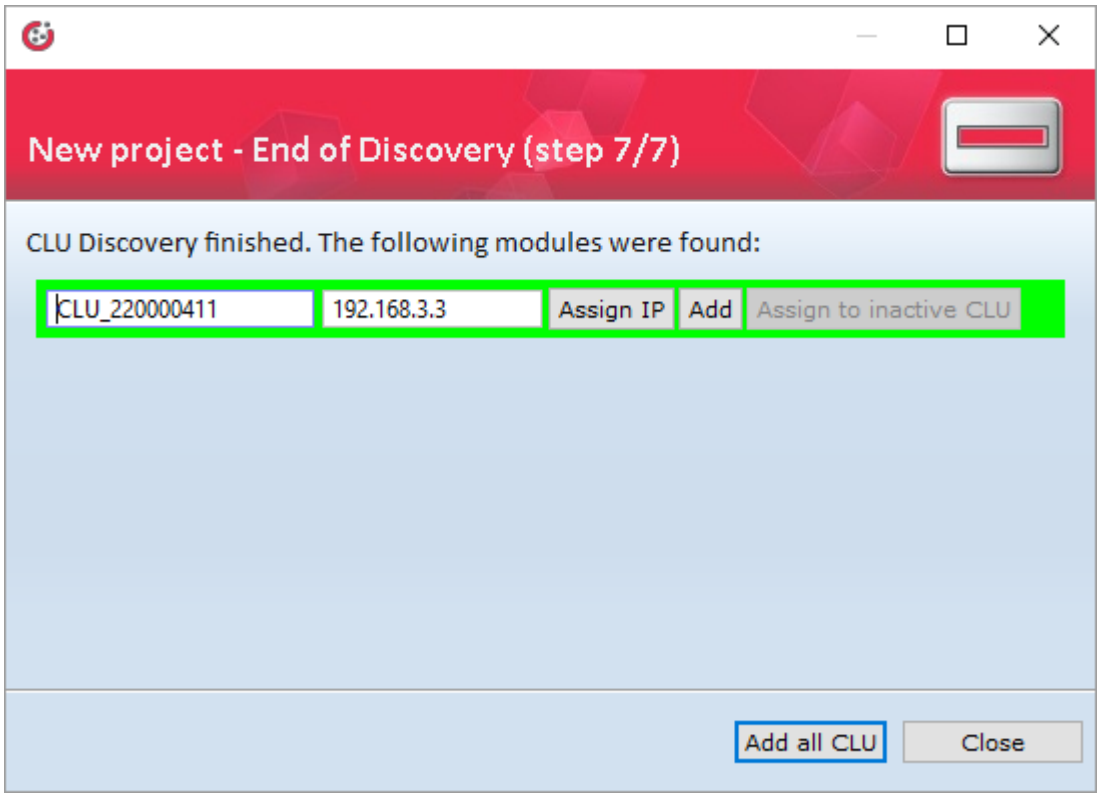
6. In the sixth step, OM starts searching available CLU modules.



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.



7. After finishing, OM will display a list of all found CLU modules. In this window, you can add all or only selected modules to the created project.



4. CLU Discovery function

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

Use CLU DISCOVERY function when:

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

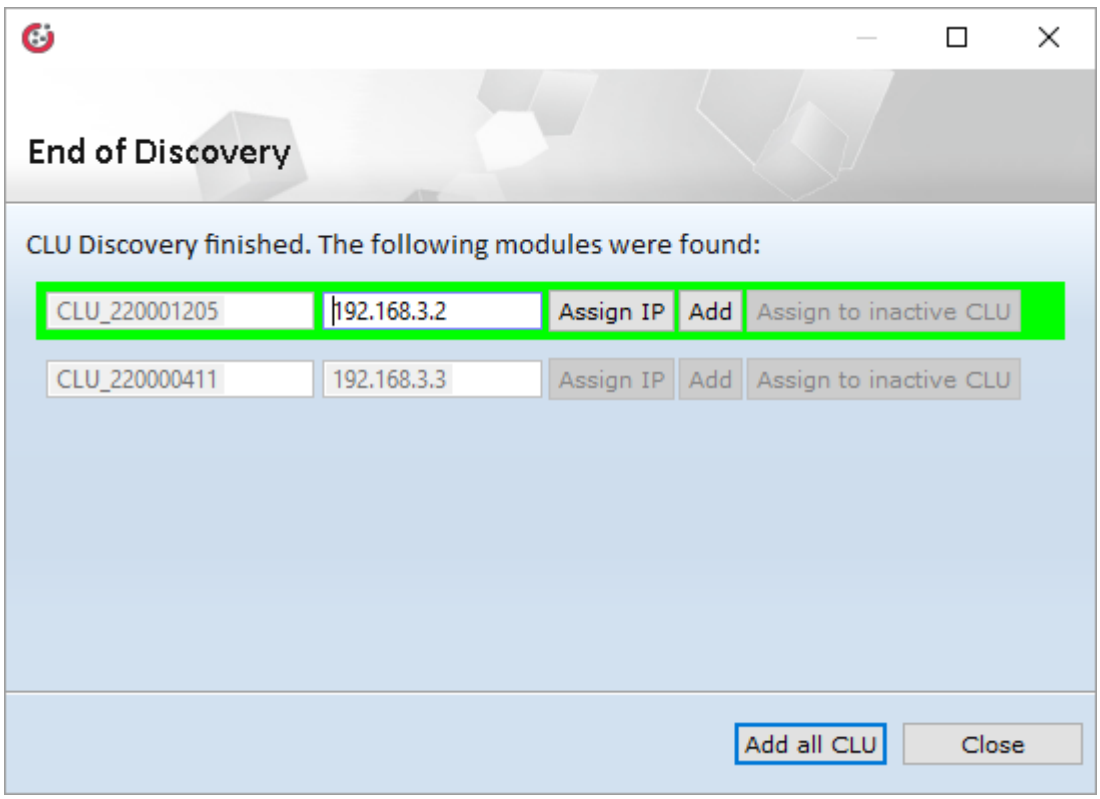
After properly conducted CLU DISCOVERY run, all changes will be found and added to the project.

Before running CLU DISCOVERY, make sure that:

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

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

All found modules will be displayed as a List.



Colour of the position means:

- **Green** – newly found CLU, which can be added to the project
- **Red** – CLU, which for various reasons can't be added to the project (version not operated by OM etc.)

- **Blue** – 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" button

After doing the above, the project consists of a list of objects present in the system and you may begin their configuration

5. CLU status

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.

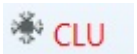
Normal mode

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

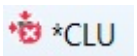
If there is no connection between CLU module and OM (no physical connection or error in LAN configuration), the name of CLU will be displayed in red, and the icon marking this status will look like this:



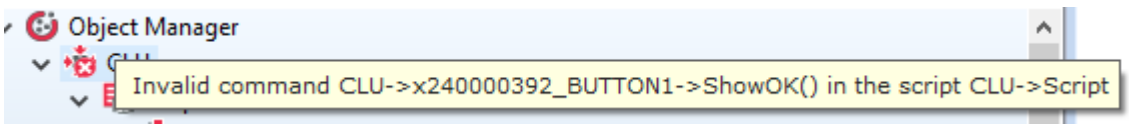
If the CLU is in disconnected mode, the user has an option of making and saving changes in the project, but the new configuration won't be sent to CLU – that is only possible in normal mode.

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 containig 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 colour to orange, and the failure symbol will appear next to its icon:

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

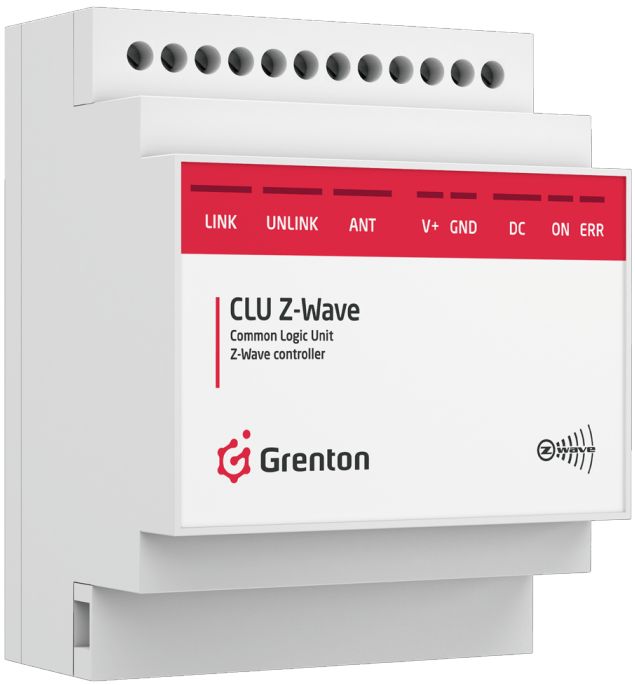
> **NOTE!** The CLUs taken out from the box (in the delivered condition) are in Emergency mode!

6. Connecting Z-Wave modules

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

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

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



6.1. Adding Z-Wave modules

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

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

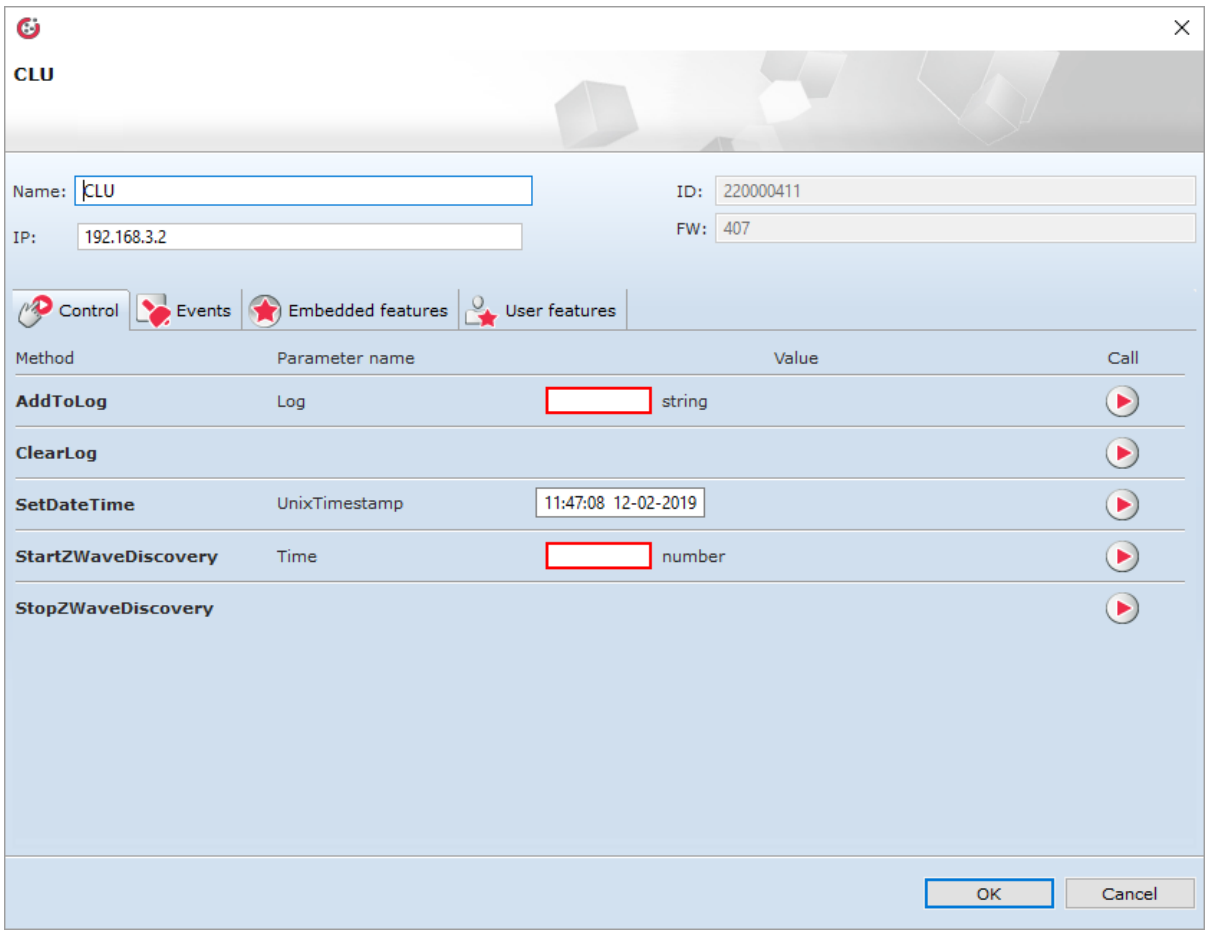
After pressing the button, the CLU switches to the mode of adding modules - the ON diode blinks all the time at intervals of 200ms.

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

2. **Using Object Manager software**

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

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



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

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

performed - new Z-Wave modules will be added to the project.

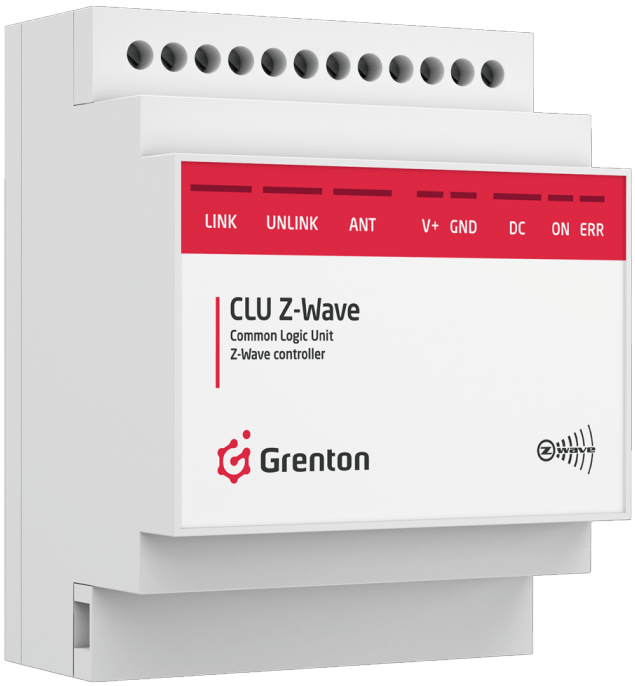
> **NOTE!** Calling the `StopZwaveDiscovery` method interrupts the search for Z-Wave modules.

> **NOTE!** Do not add modules to the system that have already been connected to it. If you are not sure whether a module has been added before, you should first perform the removal procedure for this module.
>
> The situation is similar when the Z-Wave module was connected and was not removed from another controller - the procedure of removing the module should be performed first.

6.2. Removal of Z-Wave modules

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

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



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

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

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

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

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

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

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

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

The following features are available for a given object:

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

Failure counting mechanism in communication:

- In the event of failure of communication with the module (no response, confirmation, etc.), the **FailCount** feature of the ZWAVE_CONFIG object of the Z-Wave device is incremented.
- Another attempt to send an order to the retry device is every 15 seconds - 3 attempts are made to communicate with the device.
- In the case of 3 attempts to communicate with the module, the **Banned** feature is set to 1 and all communication with the module is blocked.

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,
    - 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. 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

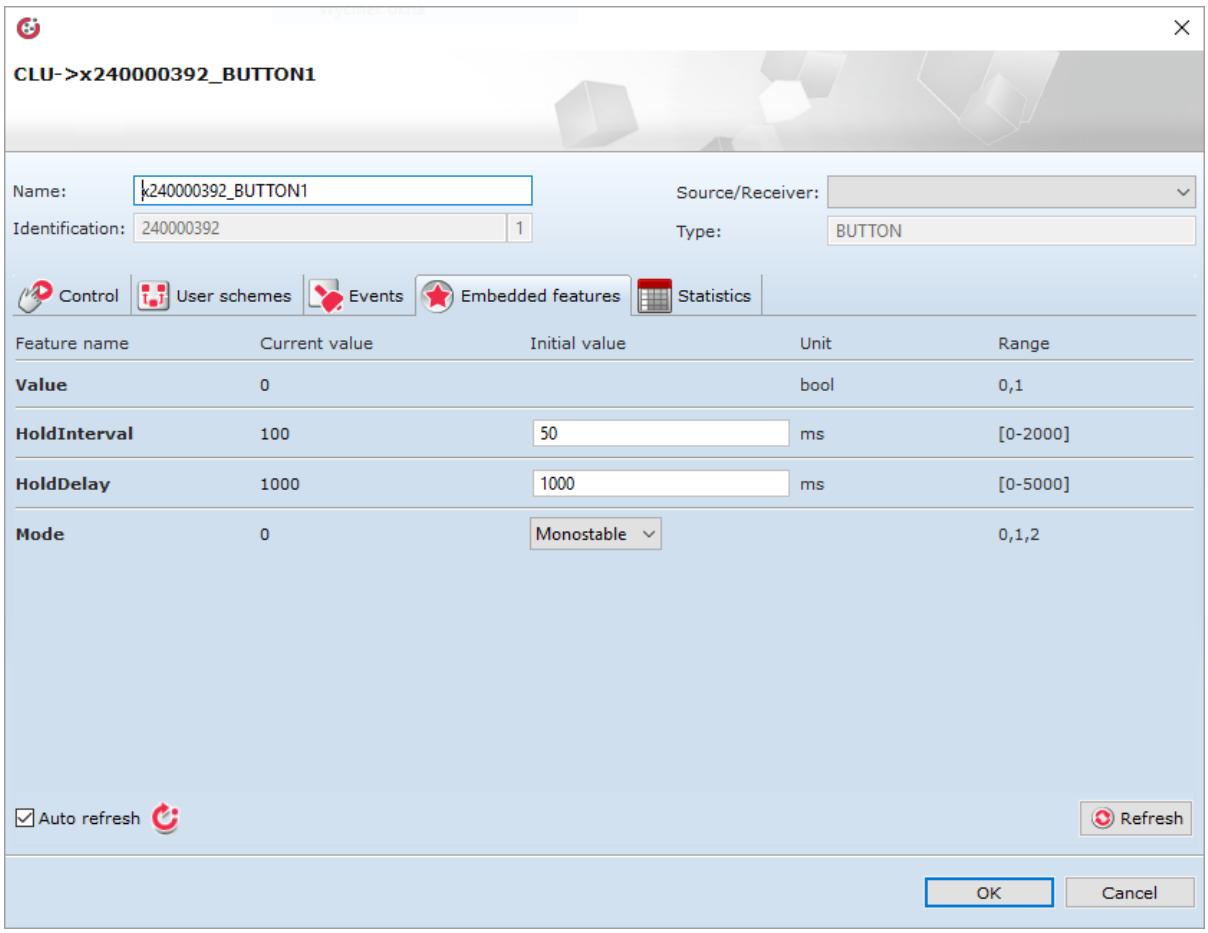
The configuration is stored in the OM and until it is sent to the CLU, it is not taken into account in the operation of the system. To send the configuration to the CLU, press the 'Send configuration' button in the menu.

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.

Associations can be created in two ways:

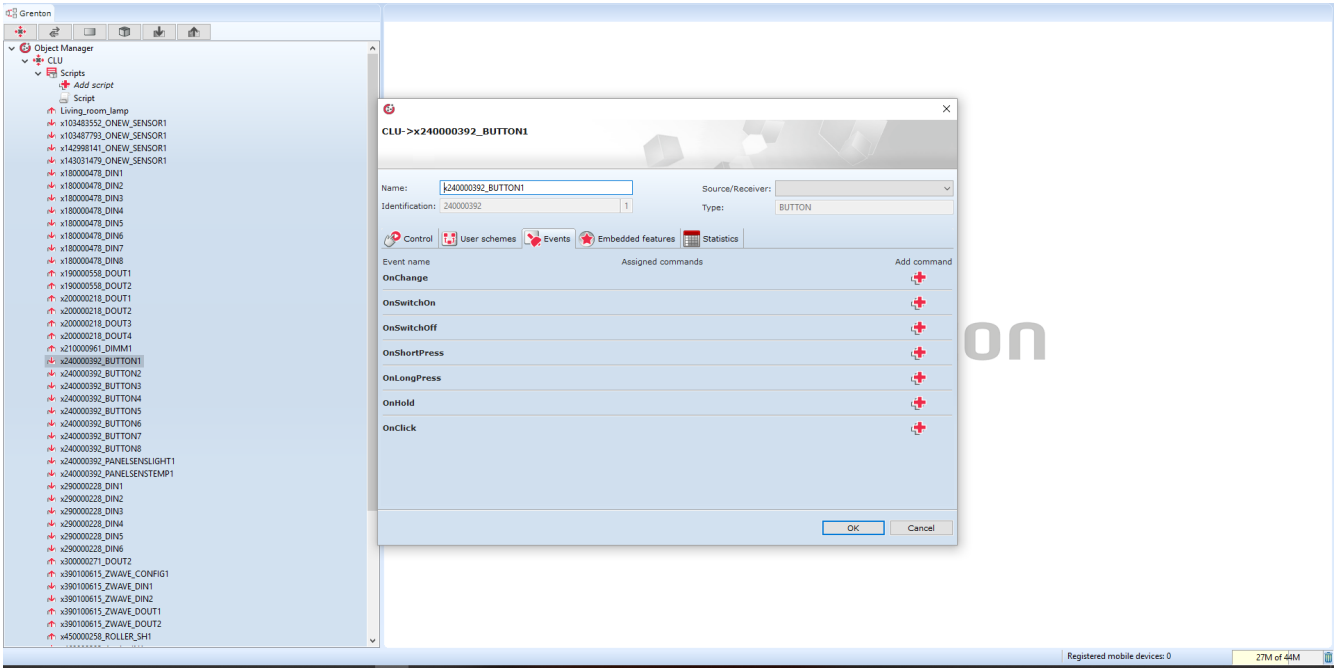
- By using configuration diagrams - it allows quick creation of typical switch-lamp connections;
- By manually creating event-method bindings - which will provide great flexibility in creating system logic.

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

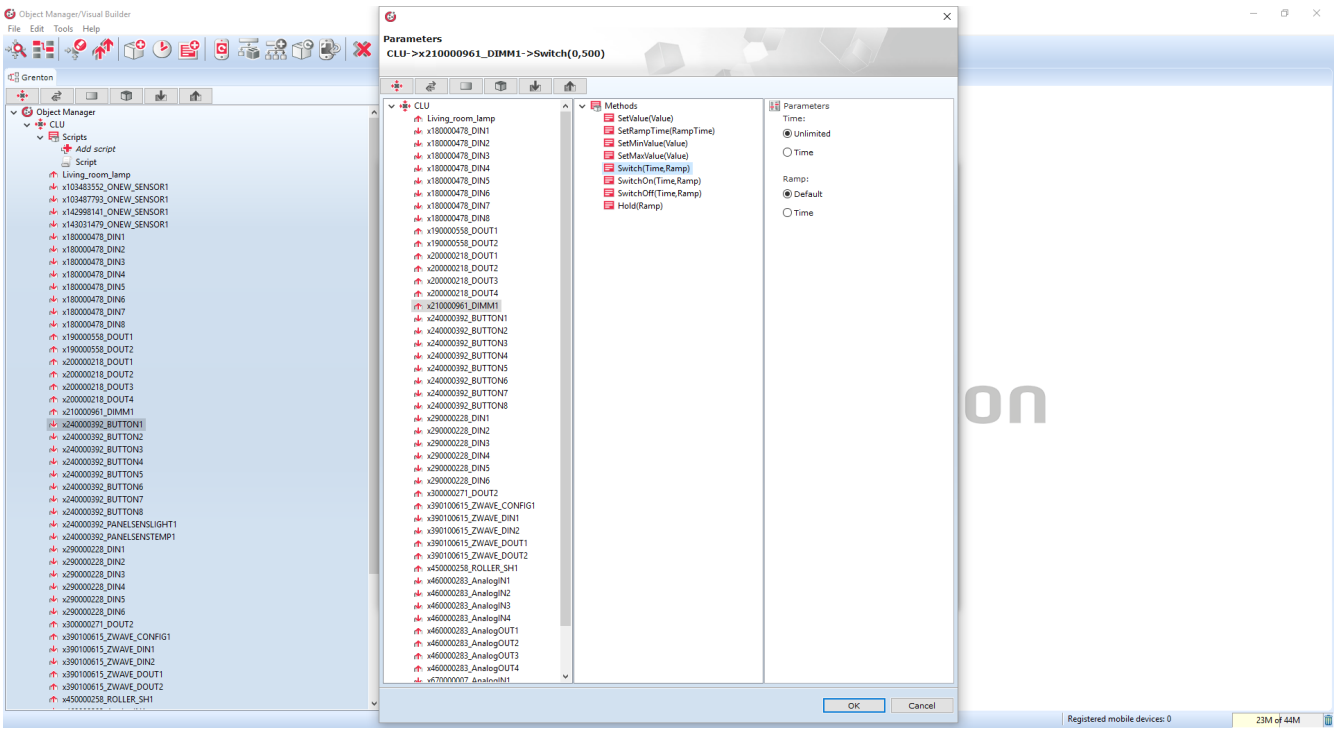
- Click on the input to be linked to the output;
- Go to the schemas tab, select an interesting scheme from the list;
- By clicking `Add relations`, select the outputs to be triggered;
- Configure the remaining inputs and send the configuration to the CLU.

To manually create an event-method binding:

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

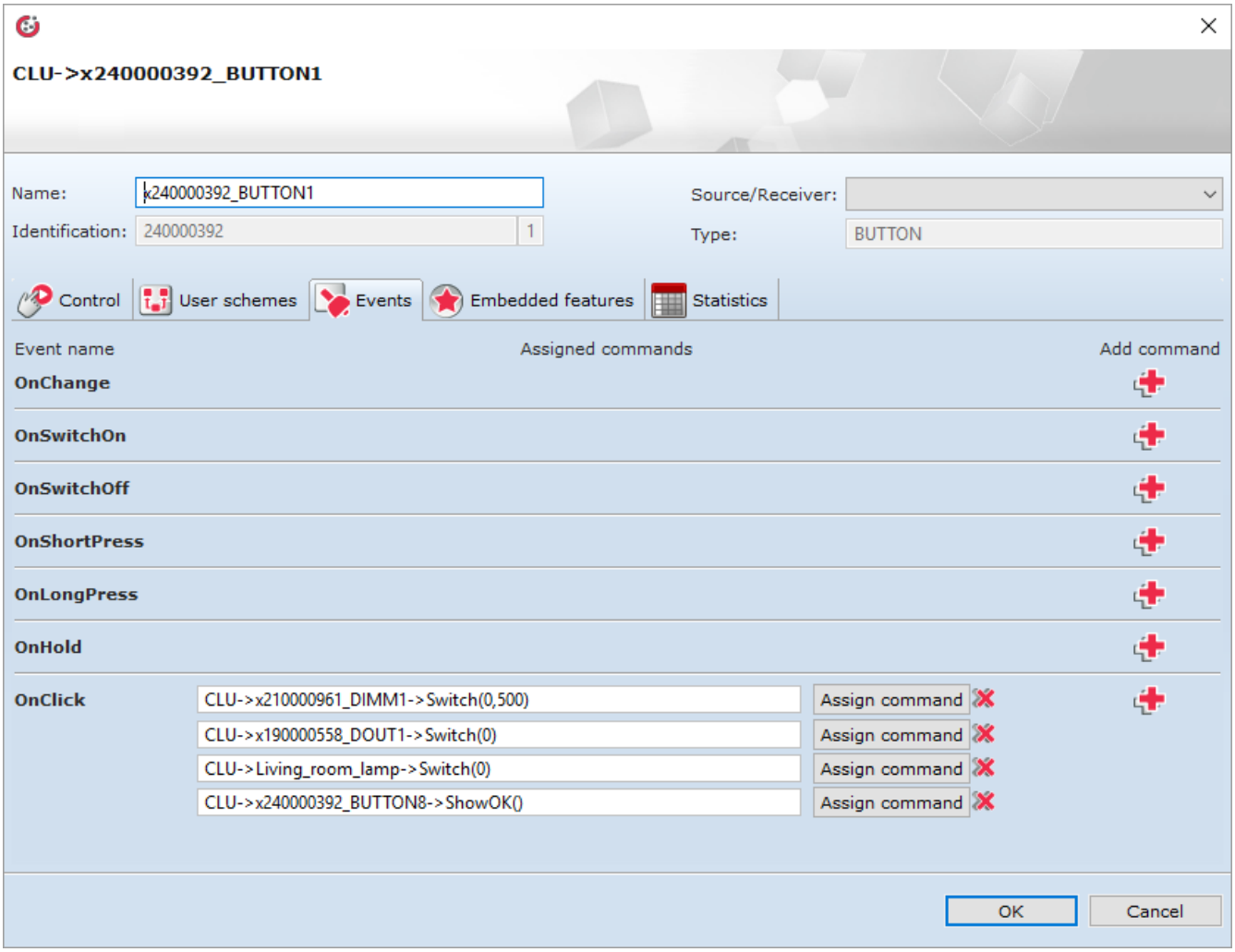


- Find the event to be linked from the list and click `+`;
- In the method selection format, select the object, method and parameters in sequence:



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

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



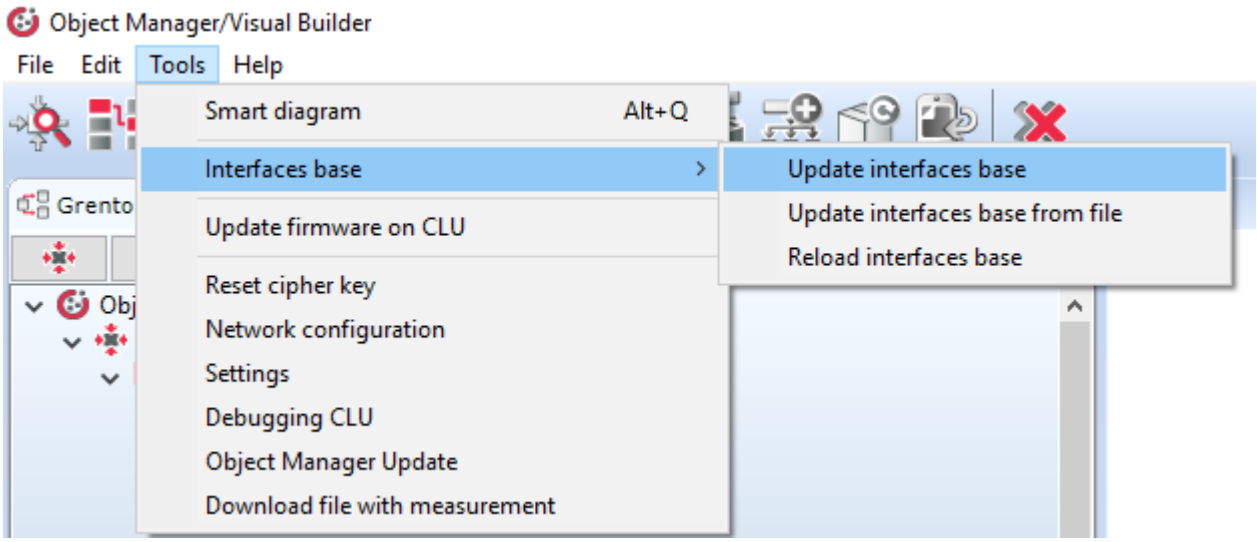
10. Performing an update

10.1. The process of updating the interface database

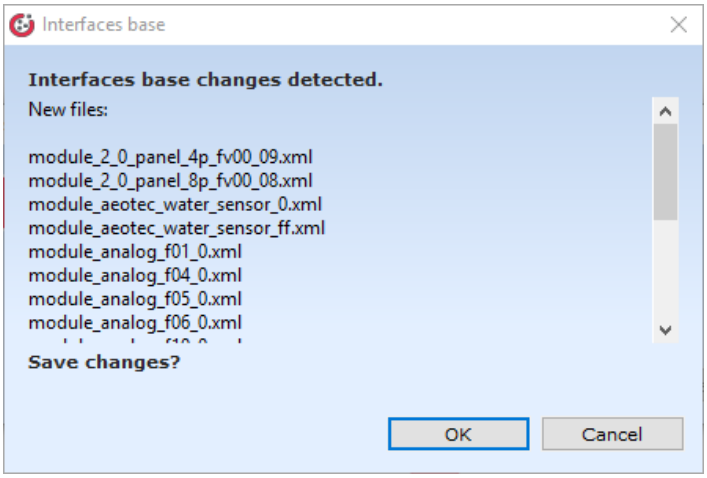
If the option *to automatically update the interfaces database* is marked when the Object Manager is started for the first time, there is no need to run it again. Otherwise, remember to update regularly. Updating the interfaces database should be done always before updating the software of a given Grenton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

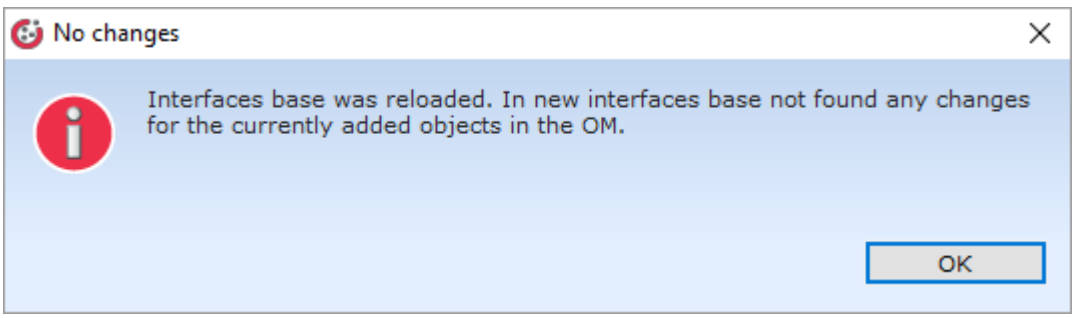
- Select *Tools* from the menu bar;
- Select the item *Interfaces base*;
- Select *Update interfaces database* from the list displayed:



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

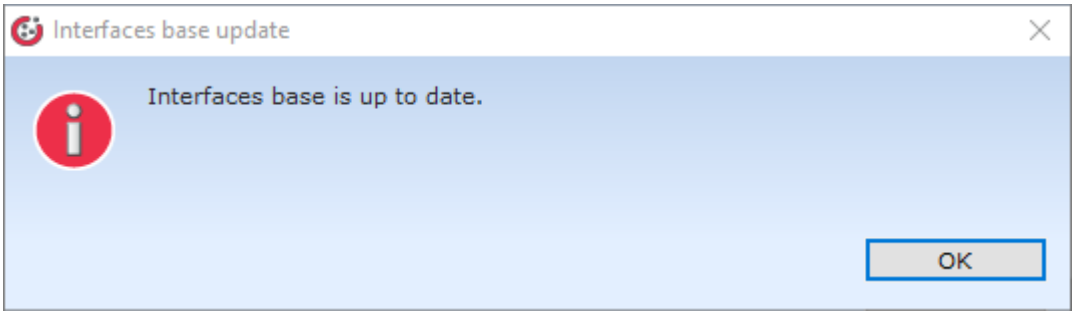


- Then a window will be displayed informing you that the interfaces base has been reloaded:



- 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:

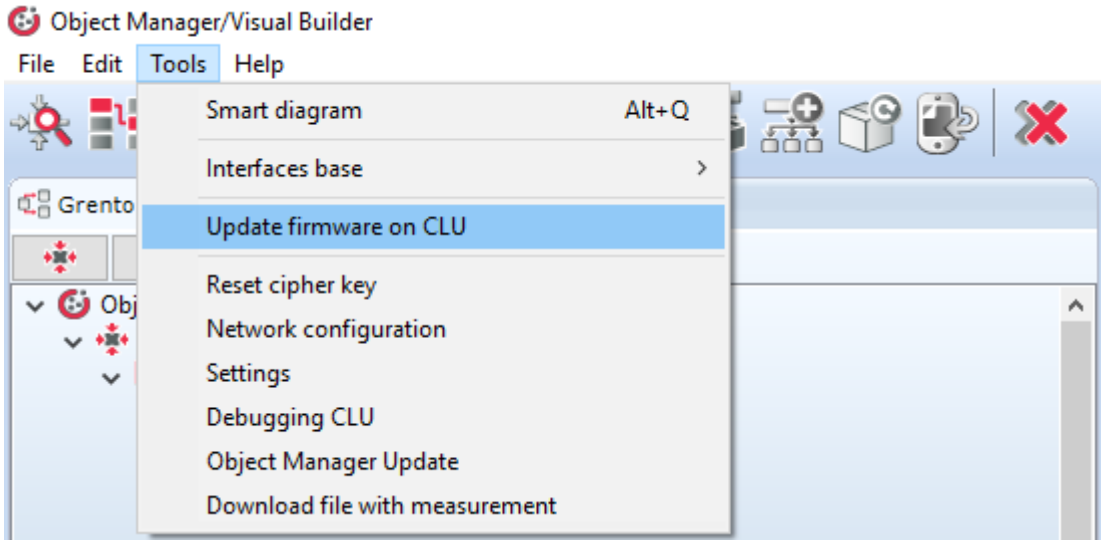


10.2. The process of updating the firmware on the CLU

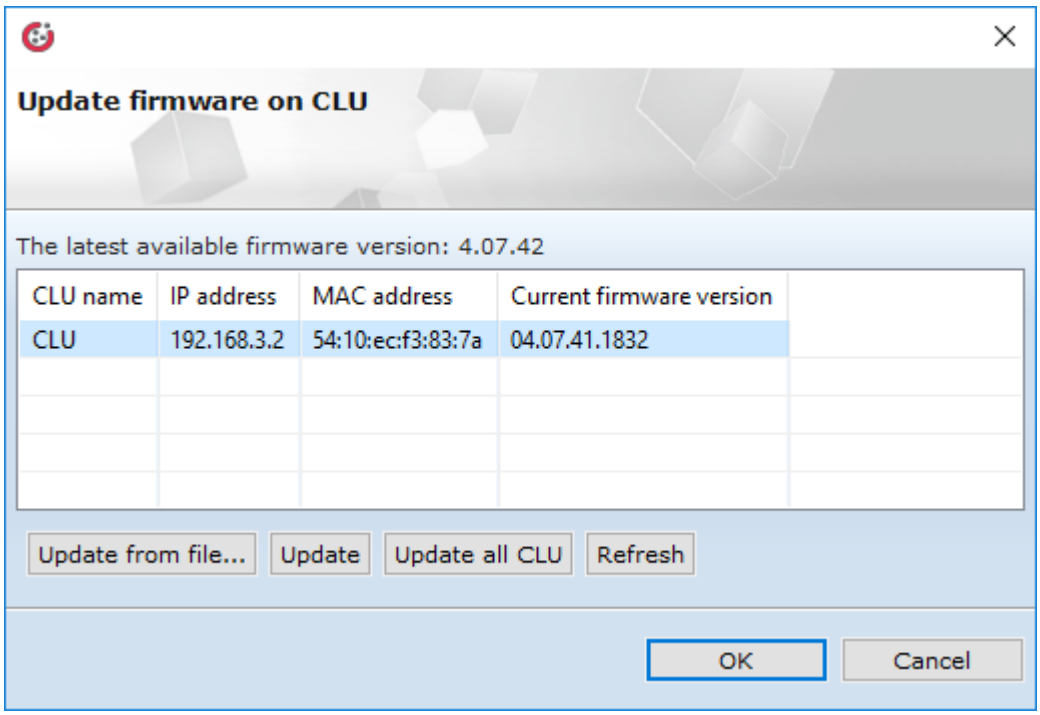
The firmware update on the CLU is carried out in order to: add support for new devices and increase the capabilities of the system. More details can be found in the Release Notes.

If you want to update the firmware on CLU you should:

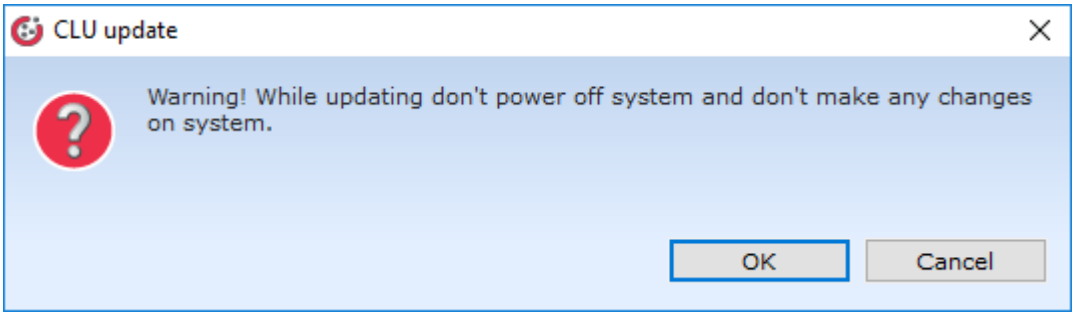
- Select *Tools* from the menu bar;
- Select item *Update firmware on CLU*:



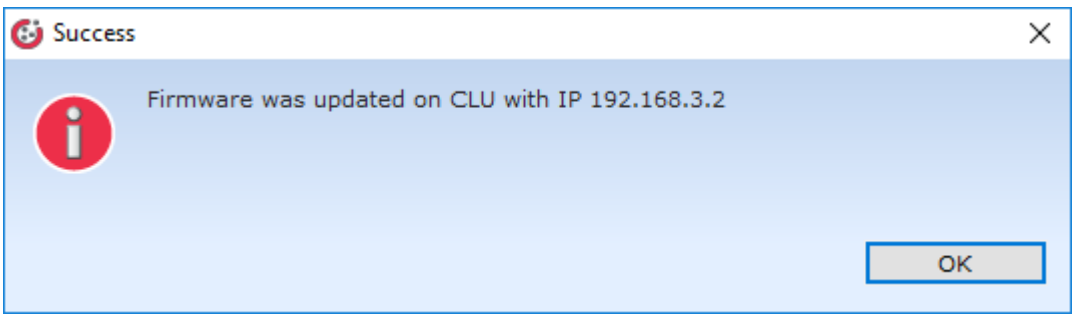
- After a moment a window will appear with all available CLUs along with their firmware versions;
- In the window you can select *Update all CLU* or select specific CLU, for which the firmware is to be updated to the latest version, and then select *Update*:



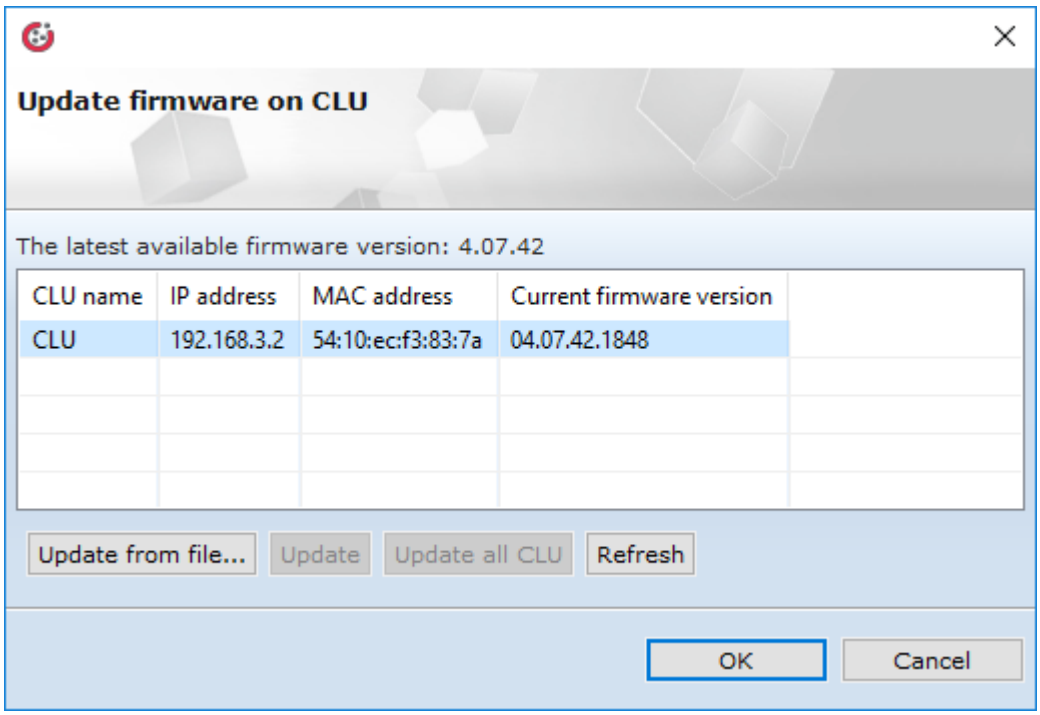
- then the information will be displayed, after which the update process begins:



- the completion of the update process will be confirmed by the following message:



> **NOTE!** If the firmware on the CLU is up to date, after choosing the option *Update firmware on CLU* in the window displayed, the *Update* option for the given CLU will not be available!



11. Other operations on the system

Cleaning configuration

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

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

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

Downloading configuration from an existing object

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

Adding a new CLU or IOM module

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

Replacing the IOM module (inputs / outputs)

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

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

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

VII. Advanced configuration functions

1. Containers

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

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

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

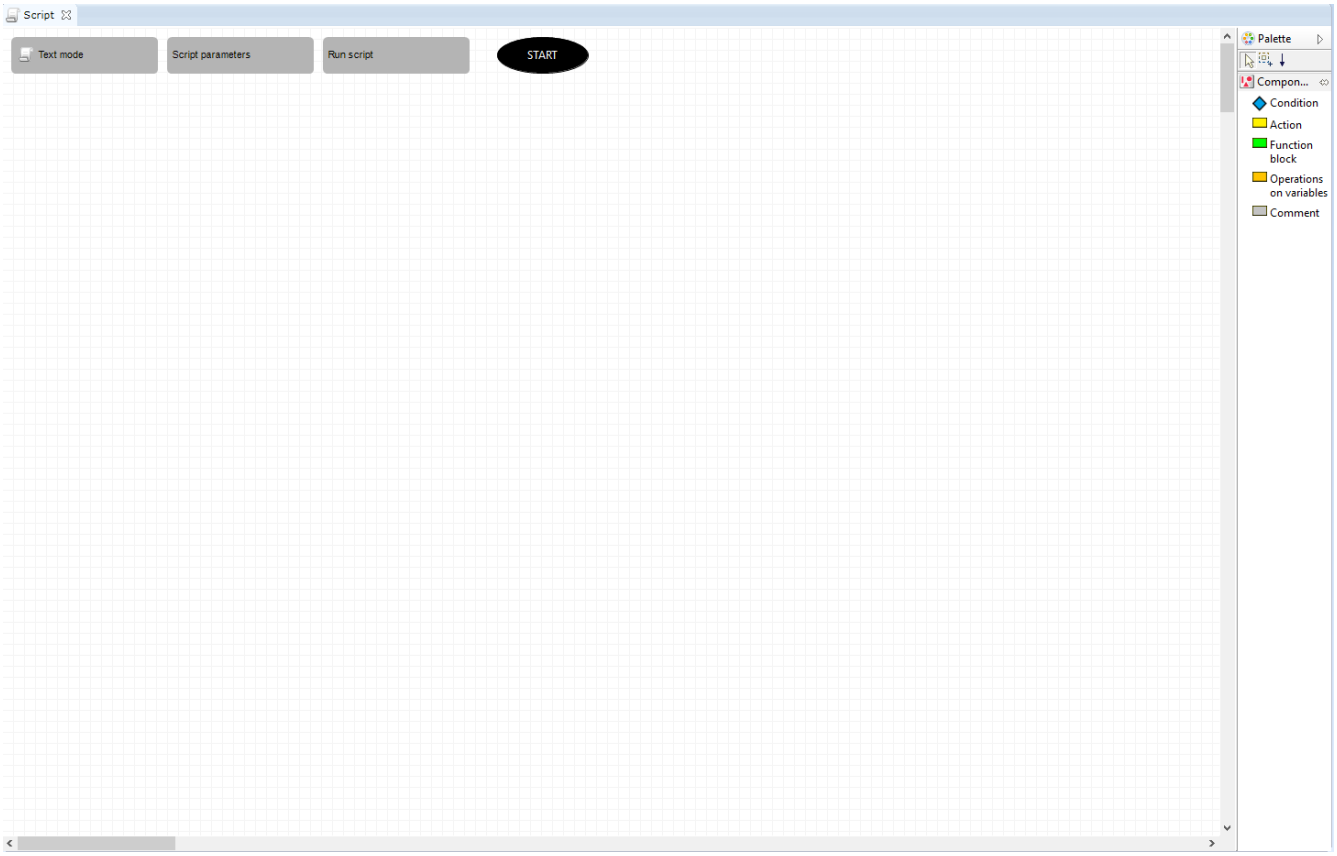
2. Scripts

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

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

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

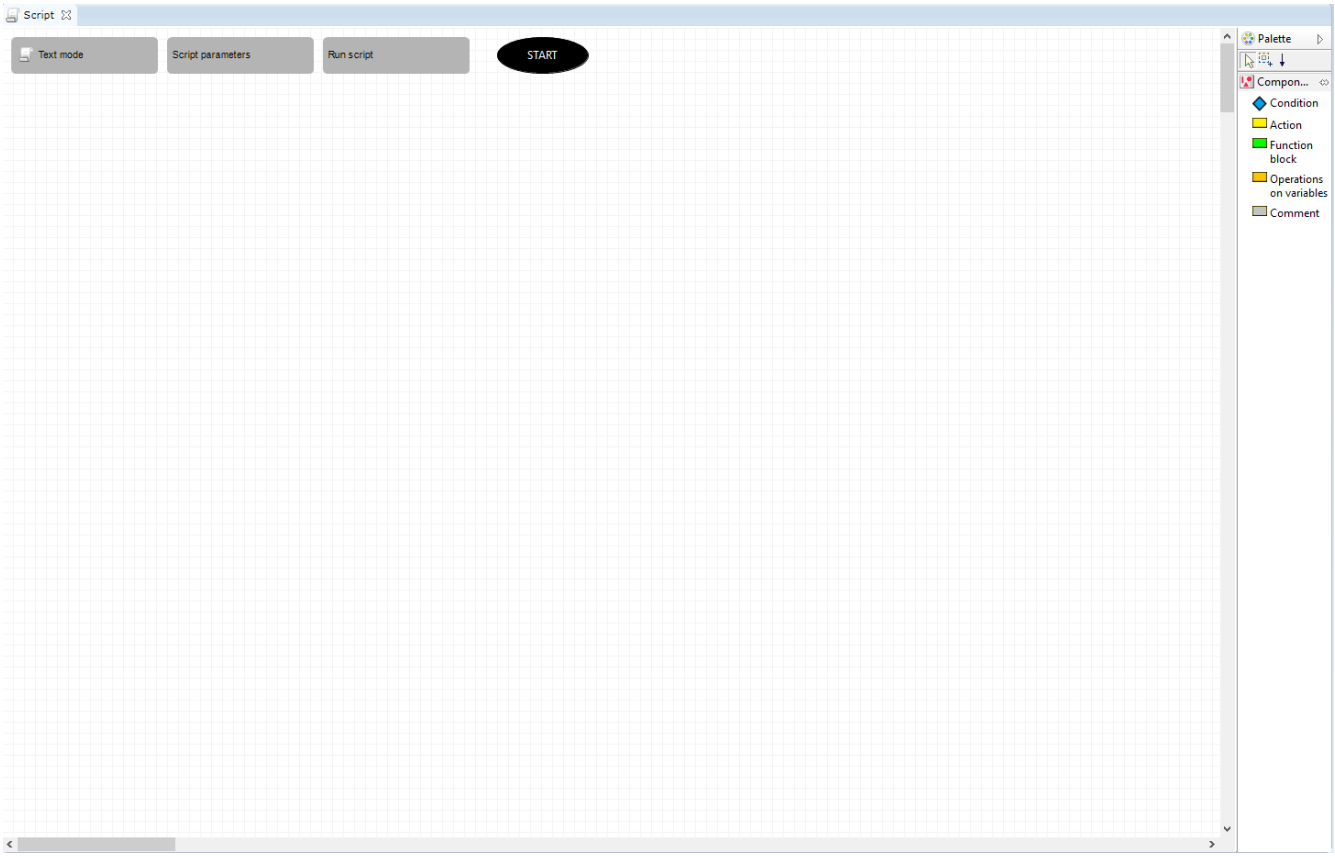
After naming the script (no Polish letters allowed), script builder that enables script creation will open in a tab. Script builder can work in two modes: graphic and text. After new script creation, script builder automatically launches in the graphic mode. You can switch to text mode by clicking `Text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

A. Script creation in the graphic mode

After opening, a clear worksheet appears.

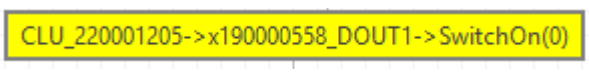


There is components list on the right of the worksheets. Drag commands from the list to the worksheet to add them. After dropping a command on the worksheet, a dialogue box open which allows to determine command parameters and conditional instructions. After adding a new component to the worksheet, a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

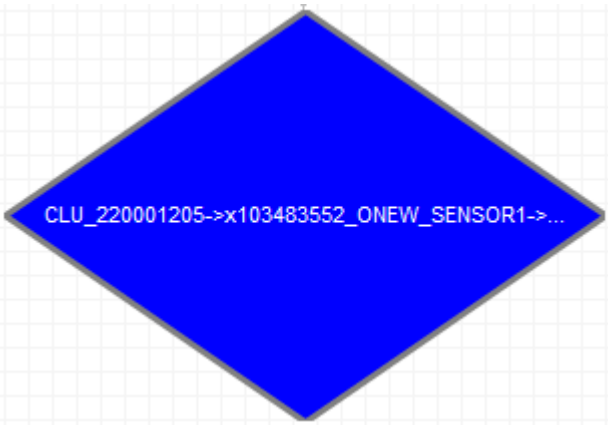
Script Builder uses the following components:

Action



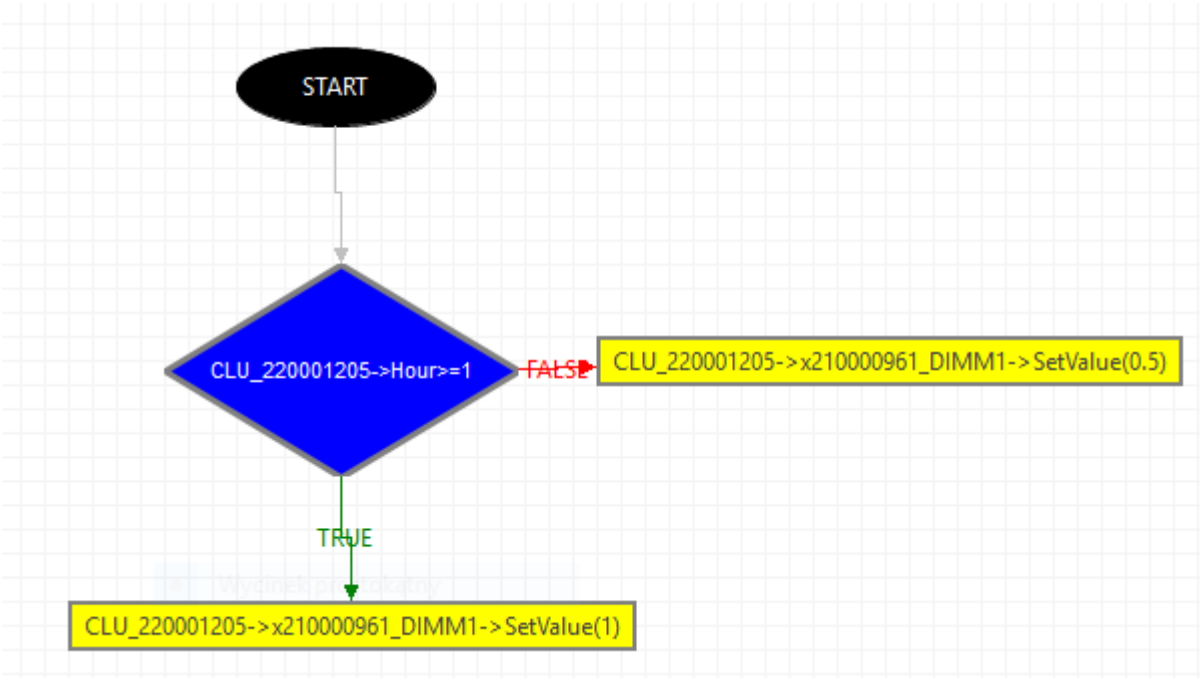
The block in which the order to be executed is entered. The command might be not only method invocation, but also value change or script invocation. After dragging action icon into the worksheet, a window with objects list and their methods opens. Scripts are available on the list as CLU methods after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

Condition

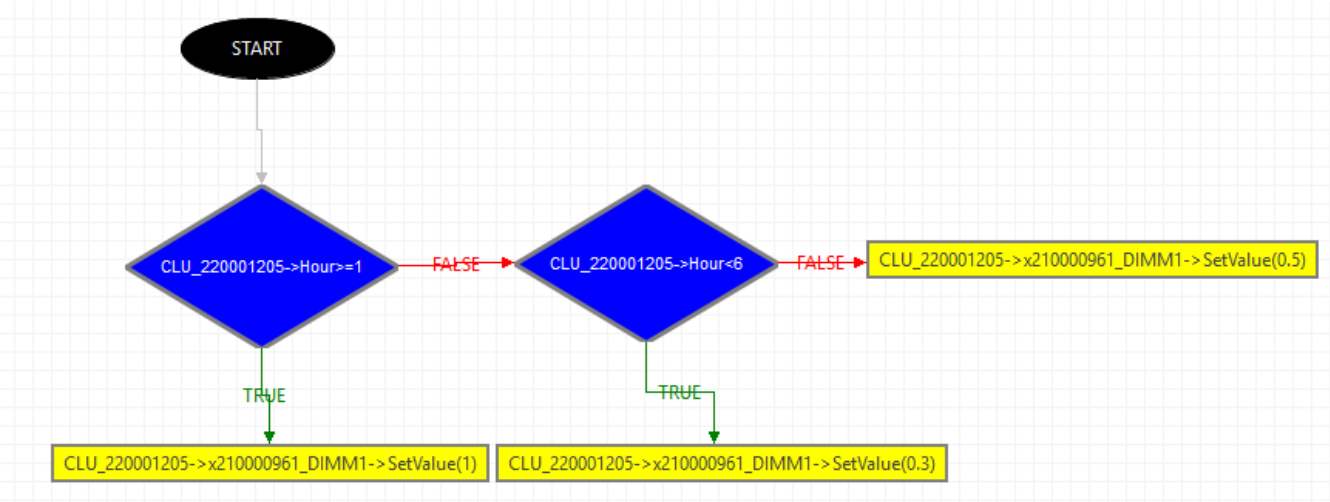


Logic block realising `IF then ELSE` function. Using this block makes it possible to make the action dependent on the conditions, eg if it is dark, turn on the light, if not, turn it off. After dragging block to the worksheet, enter condition which needs to be fulfilled in its parameters. After adding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

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.

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 realisation
- **PRINT** Command causing displaying predefined text on the command list.

Operation on variables

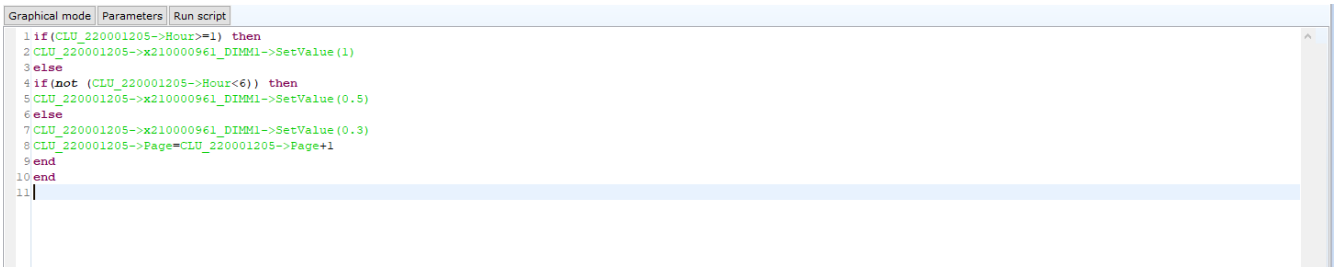
The block enables creation of complex logical functions using variables. The variables must be declared first so they can be used in the script. Variables can be declared in scrip parameters and CLU user features. A variable declared as script parameter can be used within the script to make calculations 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.

B. 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:



C. Passing parameters to the script

Scripts can have initial parameters, which are sent during their invocation (e.g. in the event) and then can be used within script, e.g. in conditional instructions. Script parameters are created in the editor by clicking `script parameters`, then defining parameters by entering name, default value, type, and restrictions.

Default value is a value parameter which will be set if the parameter is not set during script invocation.

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

- **string** – for text data;
- **num** – for numerical data;
- **boolean** – for `true` / `false` variables.

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.

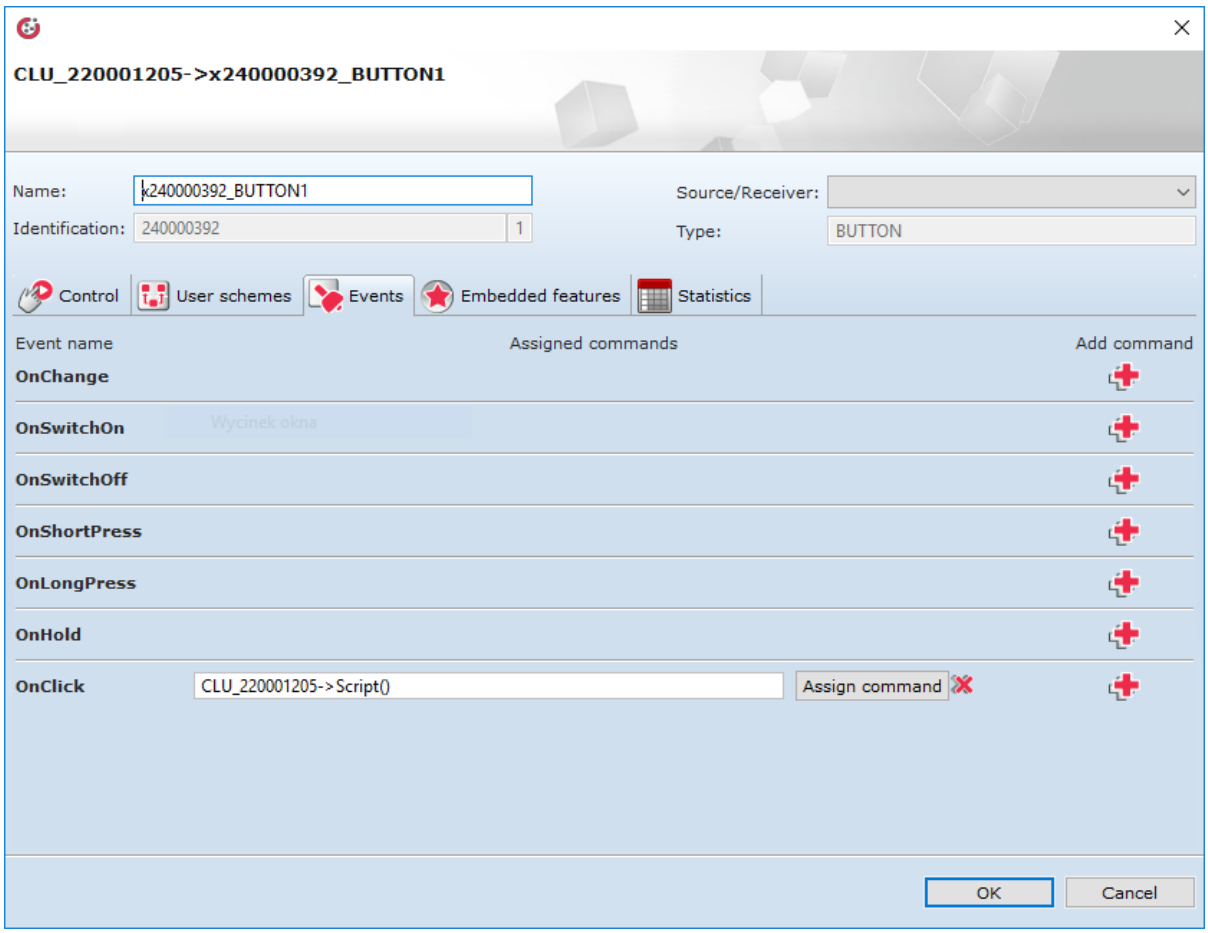
Defined parameters are available in command choice form in the script, and in the block og operations on variables

> **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.

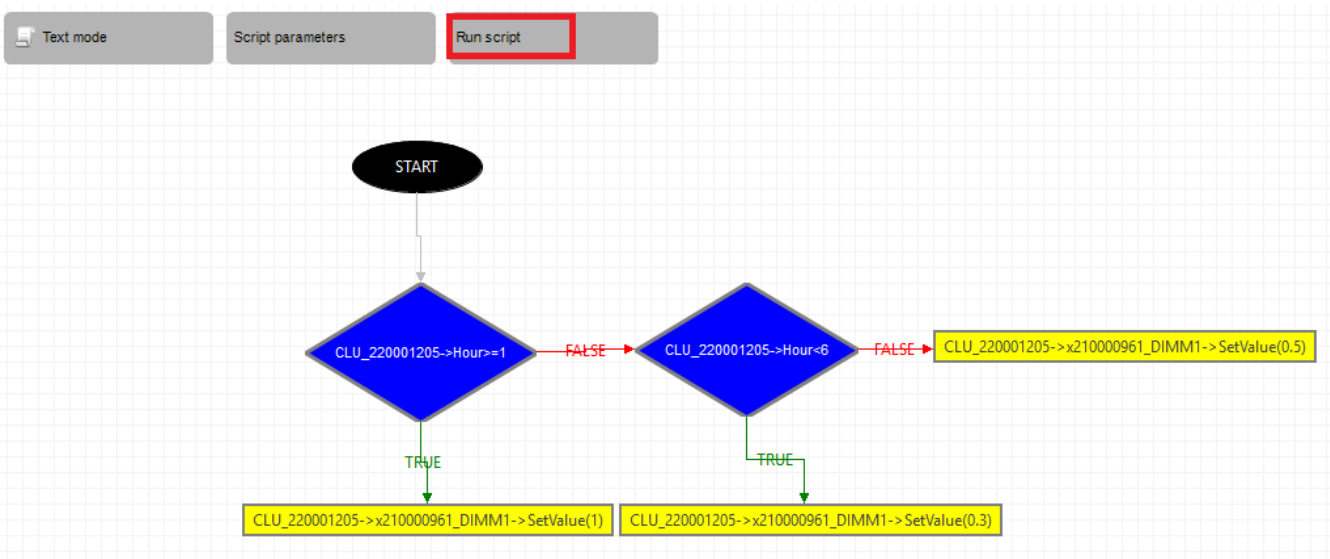
D. Scripts invocation

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.

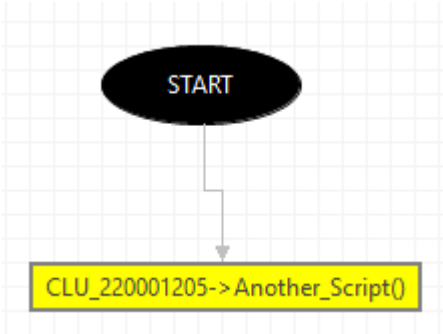
- **Invocation by an event** Picture below shows adding script to a switch. The script will be started after pressing the switch..



- **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.



3. Date and time

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

Name	Description
`Uptime`	Work time of the device since the last reset (in seconds)
`Log`	Inner log of the device
`State`	State of the device (states list)
`IsLocalPower`	`True` - if it's powered by the field bus, `false` - if it's powered by the system bus
`Date`	Shows current date
`Time`	Shows current time (hh:mm:dd)
`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
`UnixTime`	Shows current unix time maker
`FirmwareVersion`	Shows current firmware version

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

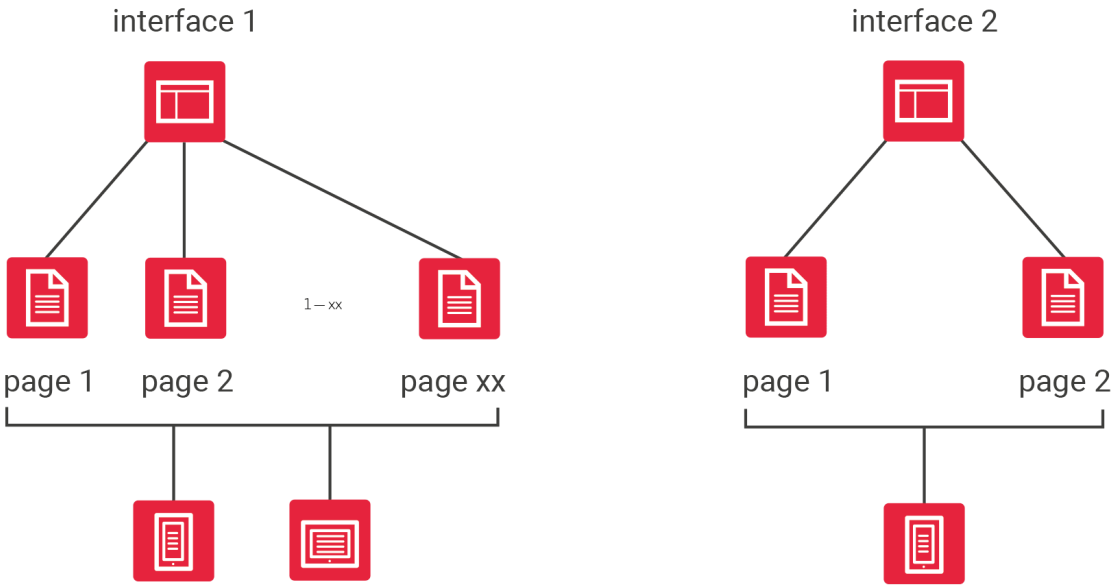
Time is set using `SetTimeDate` method. You can manually enter current time to CLU, or use automatic setting time option. After checking `Auto refresh` box, current date and time will be downloaded from the operational system

VIII. Visual Builder – Smartphone control

1. System control on the level of smartphone

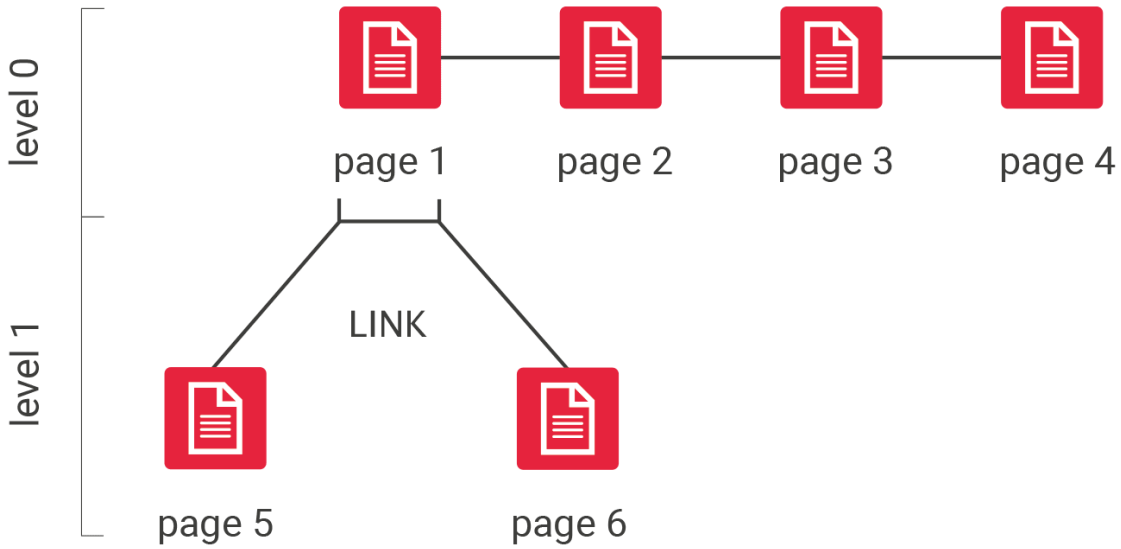
The system enables control using any devices working on the basis of both android and iOS operational systems (tablets, mobile phones, media players). For each system, one or numerous interfaces can be prepared, each of which can contain many subpages. It enables creation of various interfaces for various 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

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.

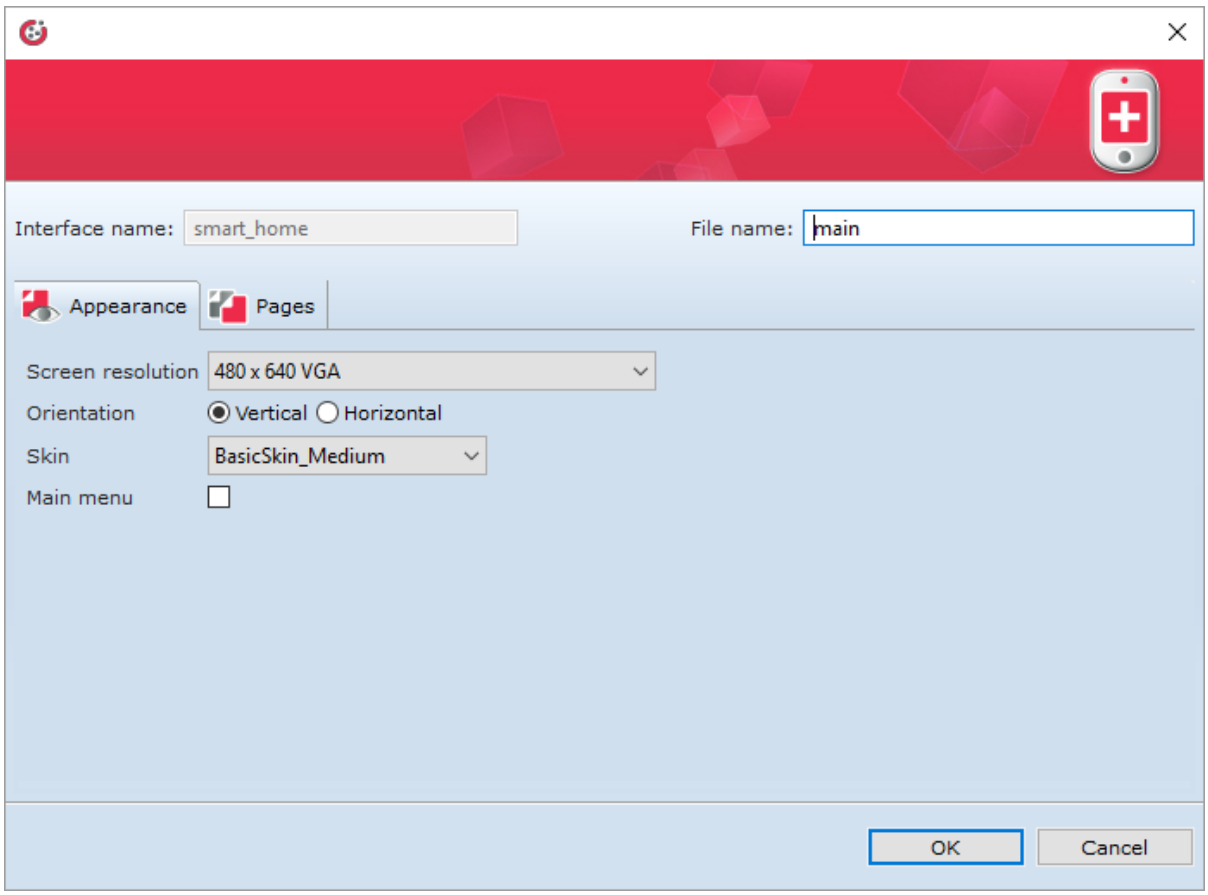
Depending on the created interface, the GRENTON HOME MANAGER application allows to check current system status and control of all functions available in the system.

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 VPN connection created in the WAN network.

4. New interface creation

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

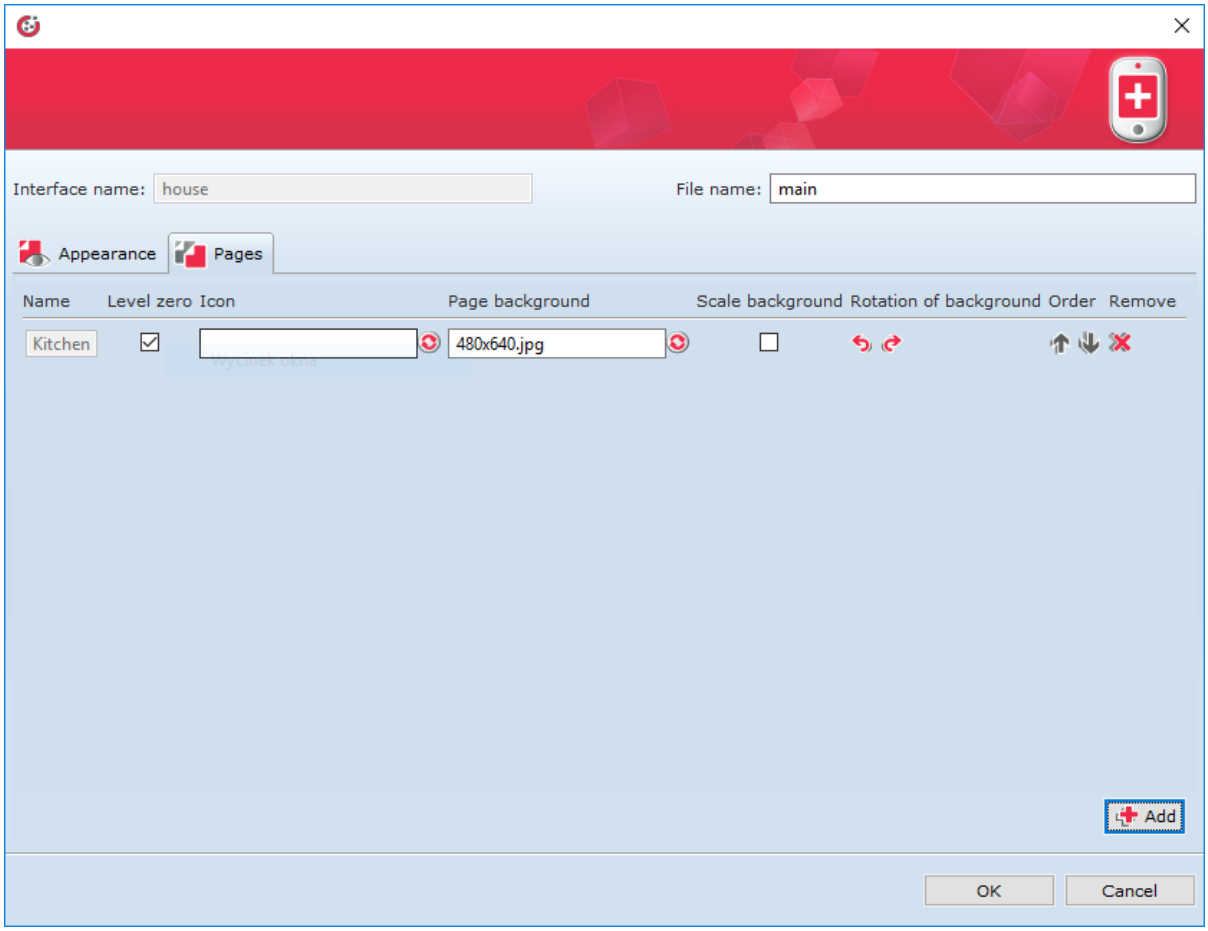
After entering name, new interface window will open. It contains two tabs: appearance and pages (interface window is also available through double-clicking icon of created interface in the objects menu). `Appearance` tab:



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 scalling. Selecting this option adjusts any background resolution to the resolution of the interface being created.

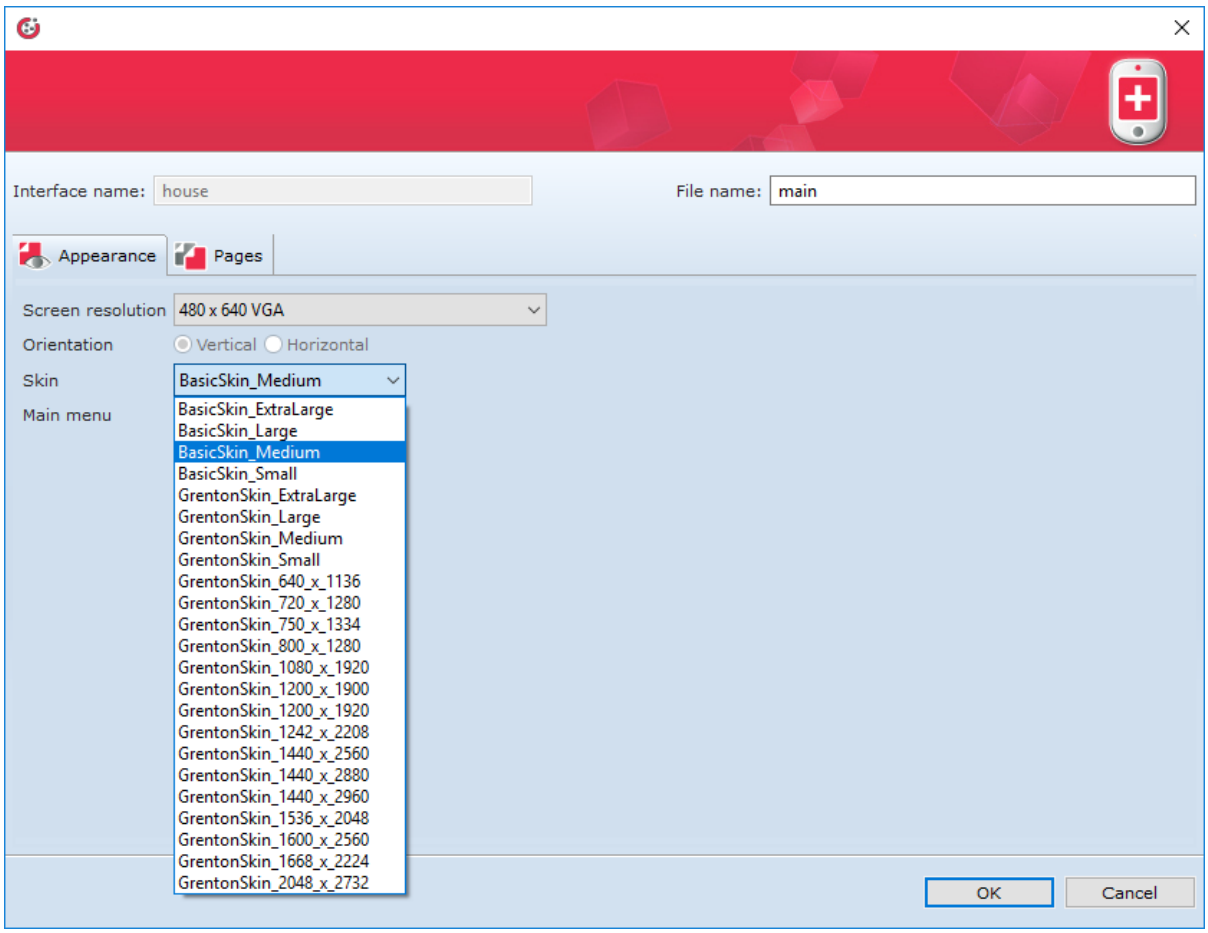
> **NOTE!** Newly created project has no information in the "Pages" tab, new information appears whenever interface page is created:

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.

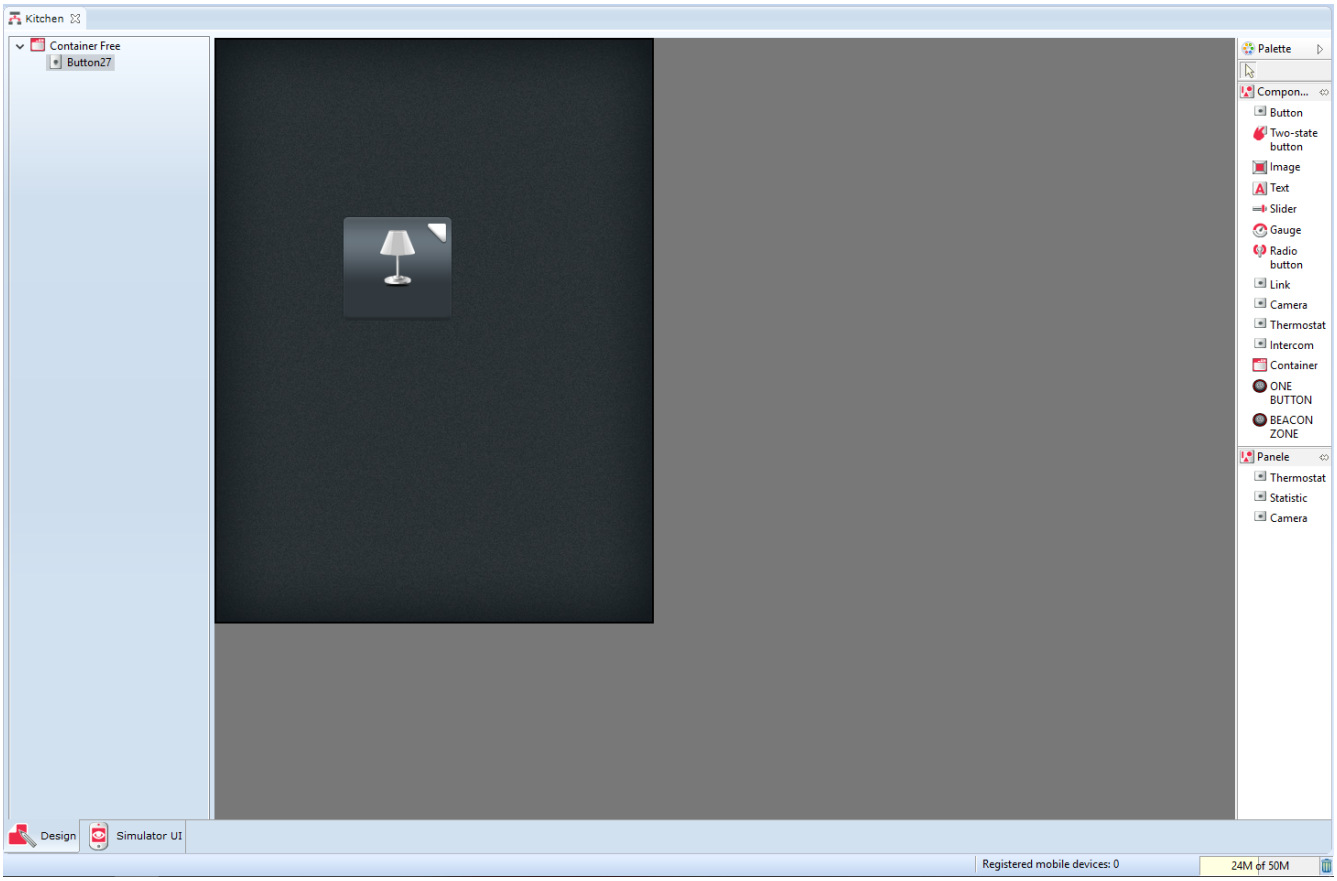


4.2. Interface pages creation

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.

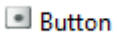


Objects list displays all objects used in the current worksheet.

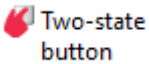
4.3. Components

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

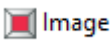
- **Button** – works as a monostable button



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



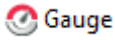
- **Picture** – enables adding picture from an external file



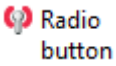
- **Text** – enables adding text box

- **Slider** – enables fluid regulation

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

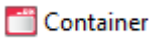


- **Radio** – displays object state in digital (on/off) way

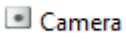


- **Link** – enables creating links to other pages within the same interface

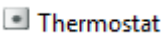
- **Container** – arranges components in the workspace in specific way



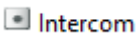
- **Camera** – allows to display image from an IP camera in the Home Manager application



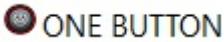
- **Thermostat** - allows displaying the virtual object Thermostat in the Home Manager application



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



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

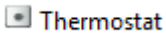


- **BEACON ZONE** - allows you to configure BEACON in BEACON ZONE mode and assign specific methods to events (after adding BEACON ZONE to the page visible at the bottom).

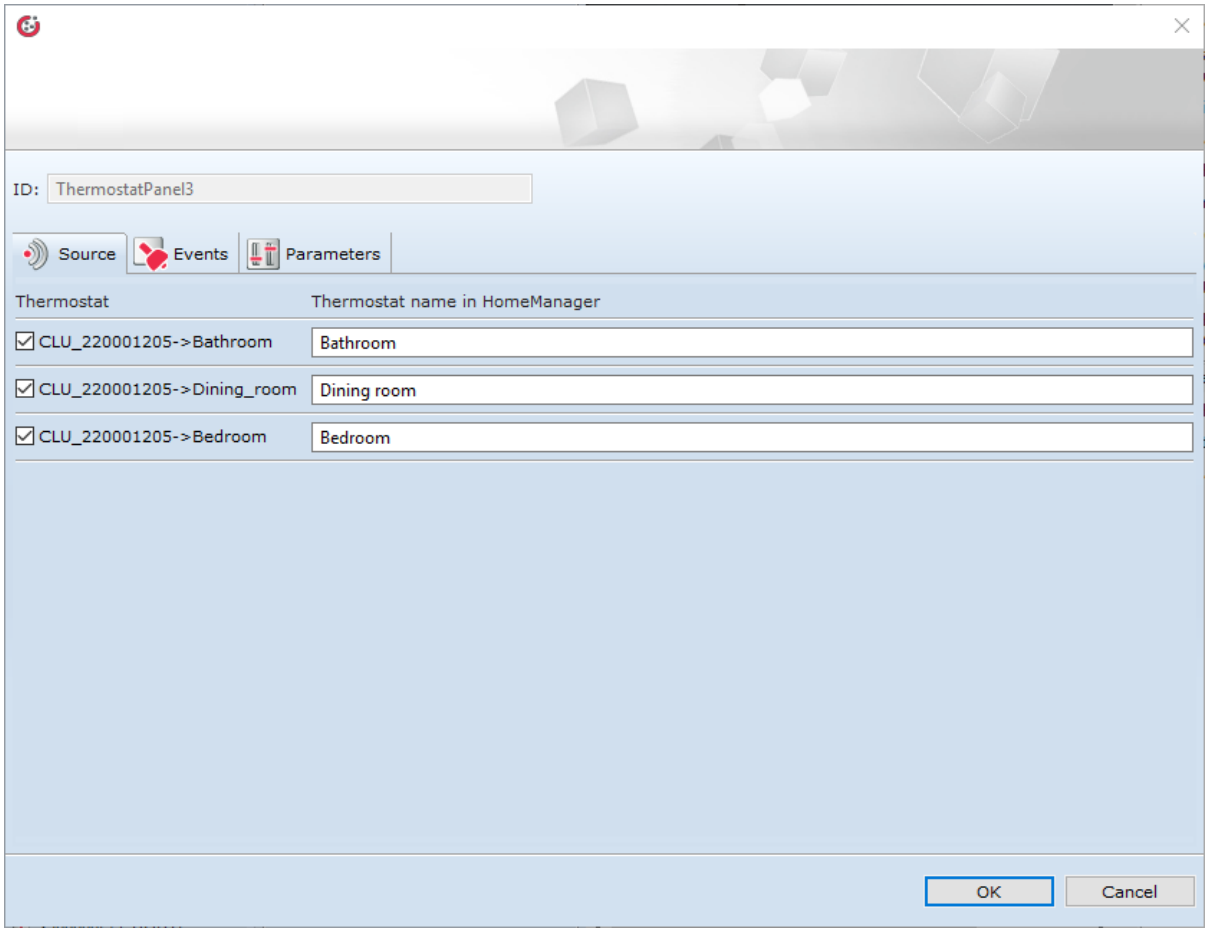
Selected objects are put in the container by dragging them from the components list and their arrangement depends on type of used main container.

4.4. Panels

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

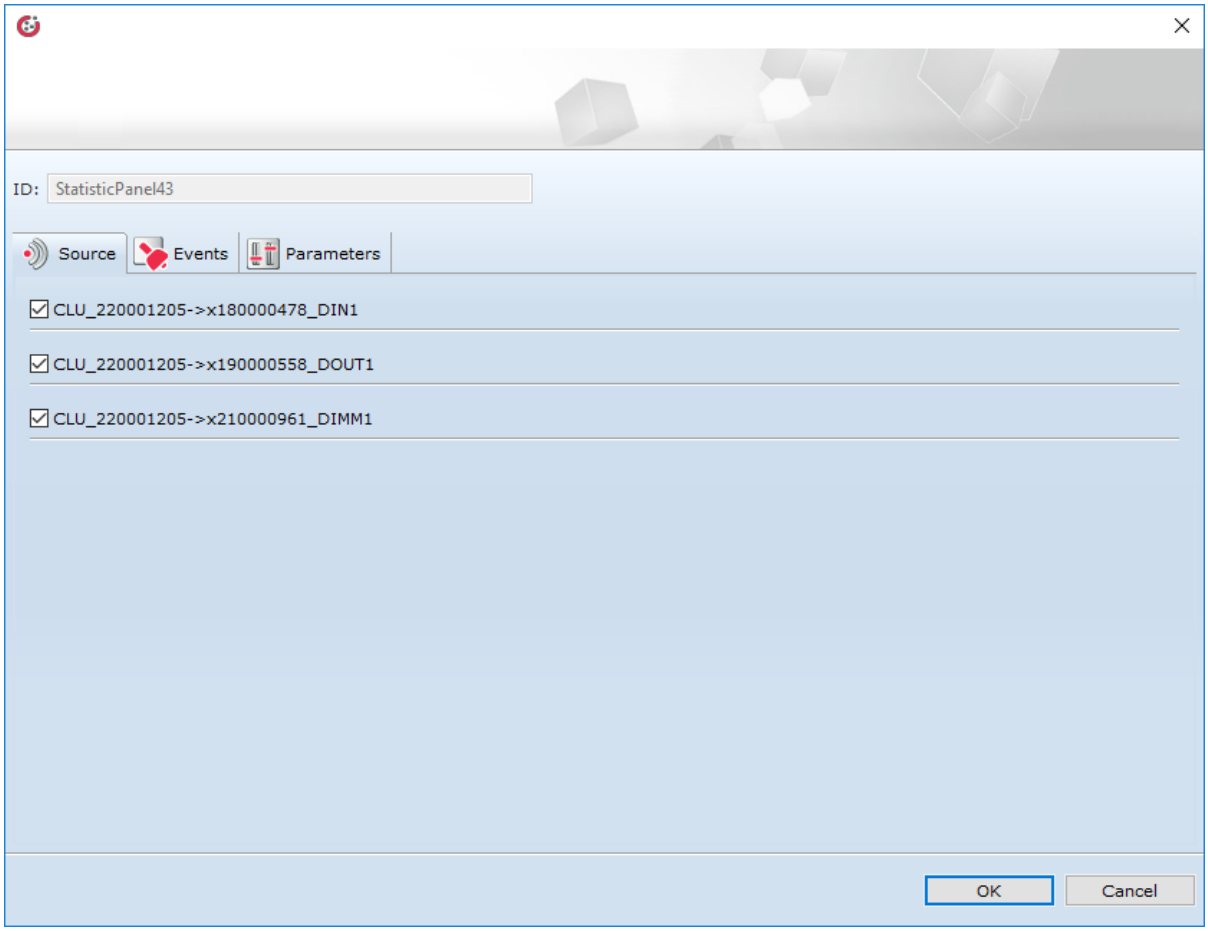
- **Thermostat** - creates a panel for the thermostat on the entire interface page in the HM. 

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



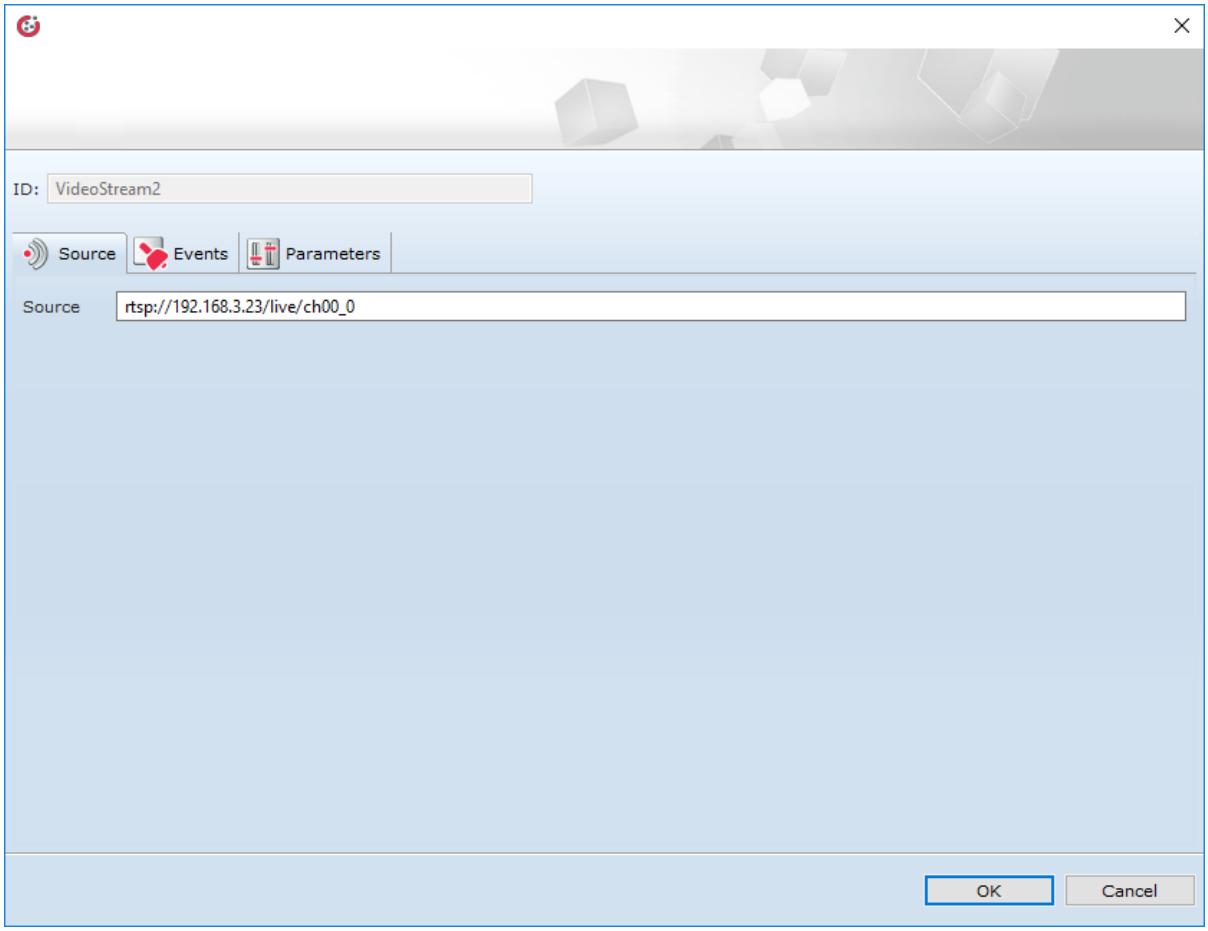
- **Statistics** - creates a panel for media measurement on the interface page in HM.

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.

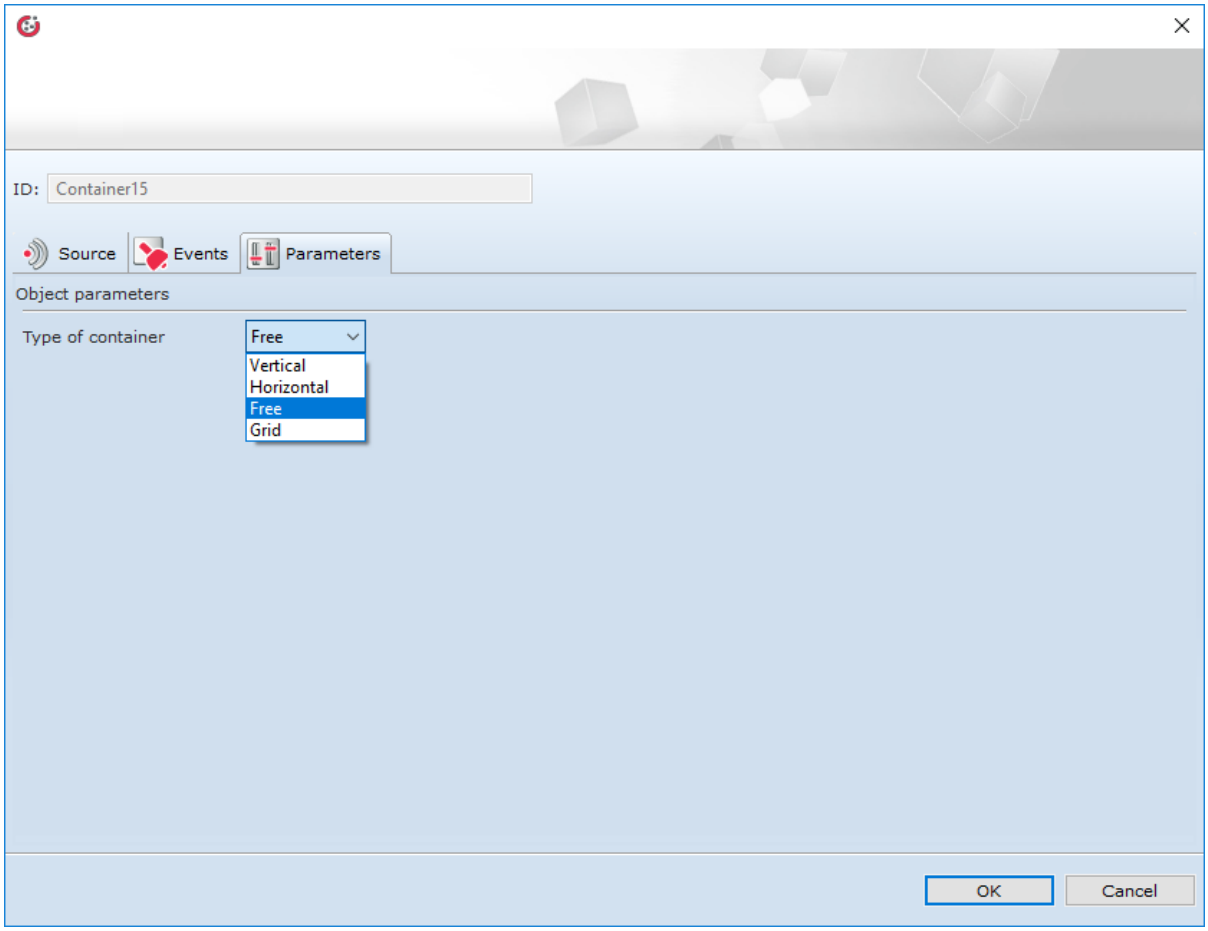


4.5. Containers

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

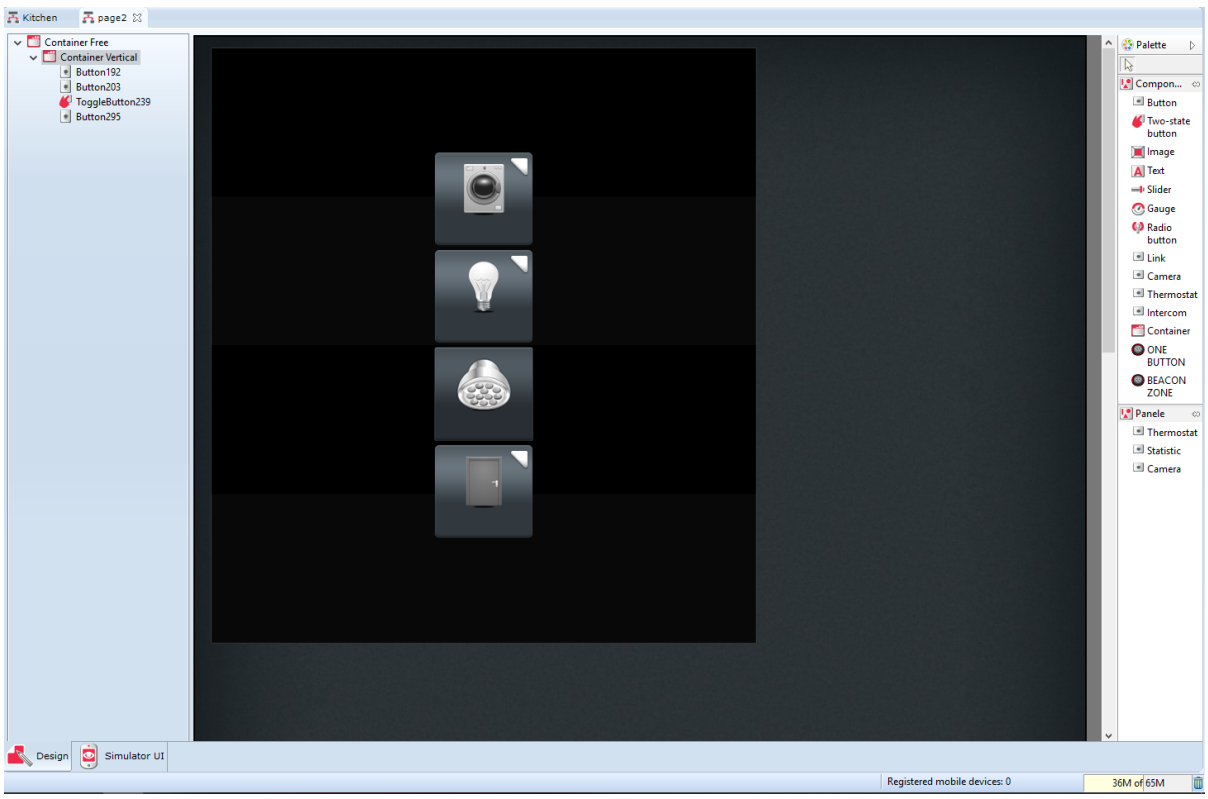
Objects within the workspace are arranged accordingly to the type of the selected container.

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.

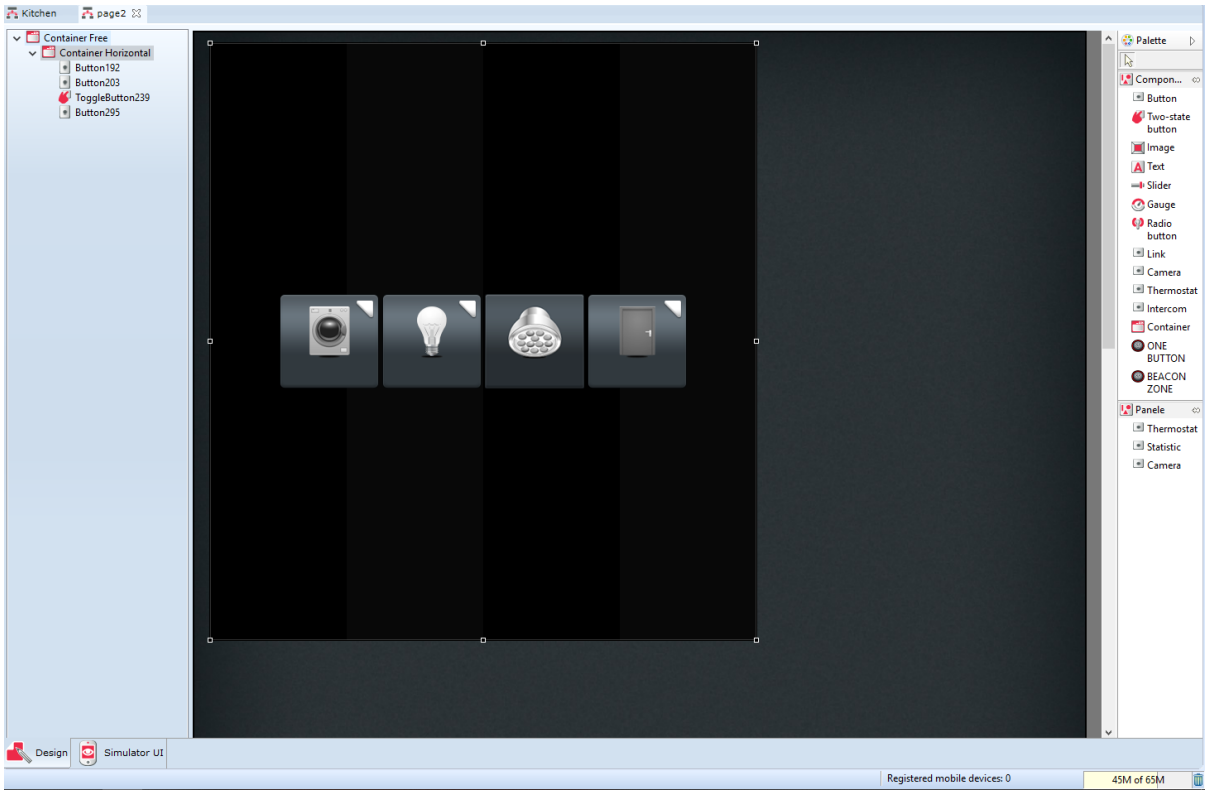


There four types of containers:

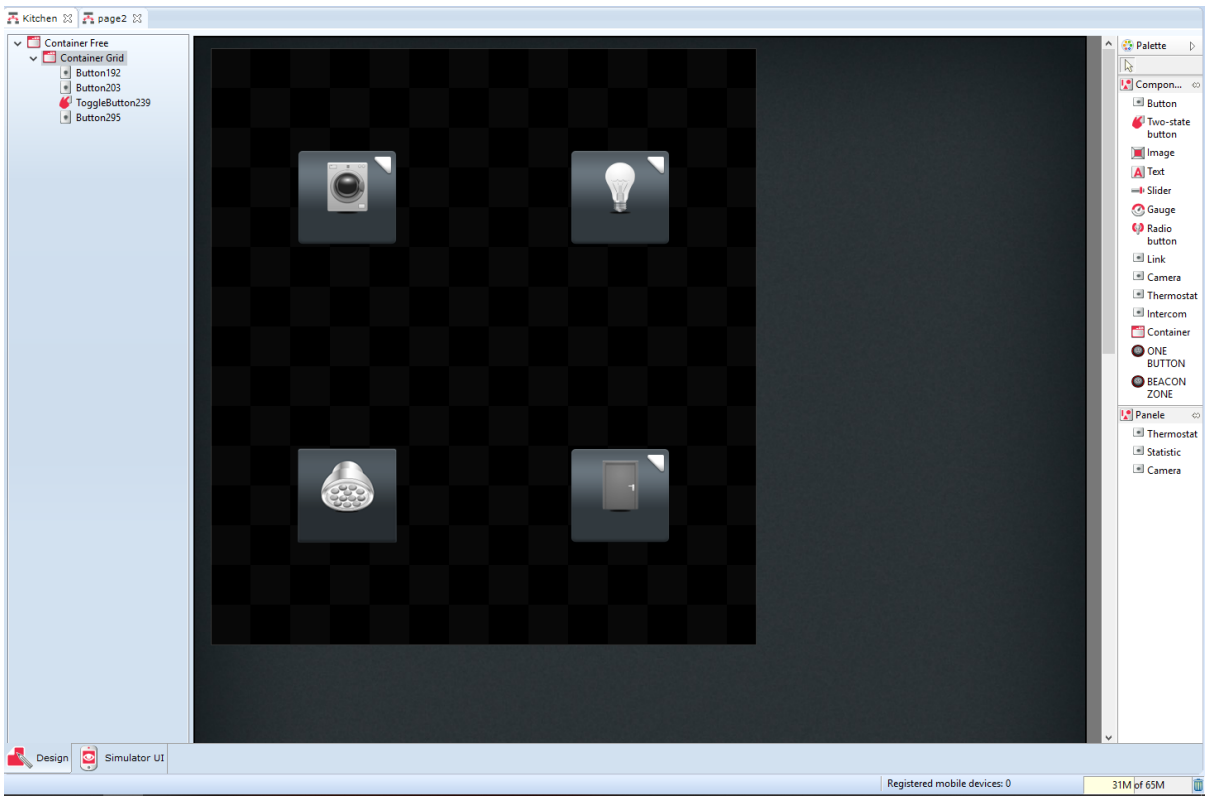
1. **Vertical** – the elements are arranged vertically in equal, automatically created sections



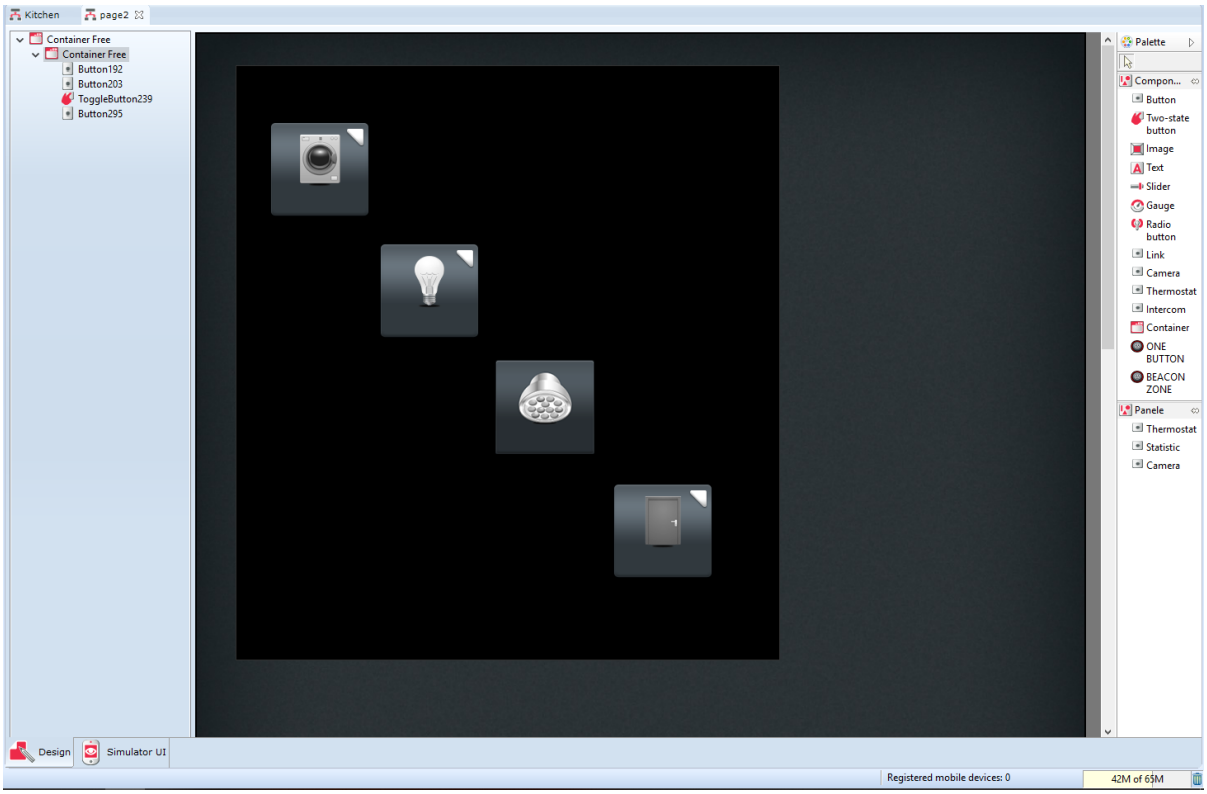
2. **Horizontal** – the elements are arranged in horizontal sections



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



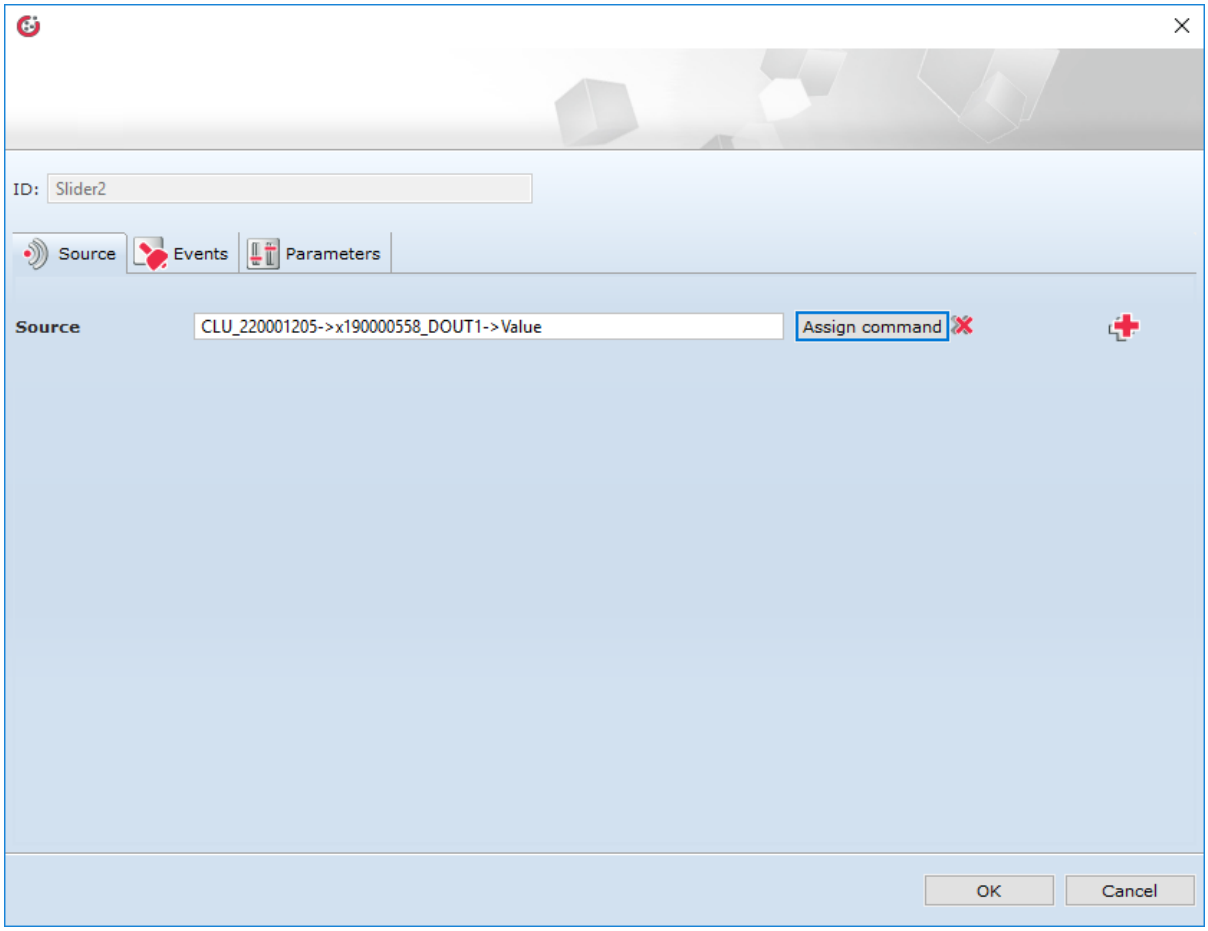
4. **Random** – enables any arrangement of the elements within the whole container area



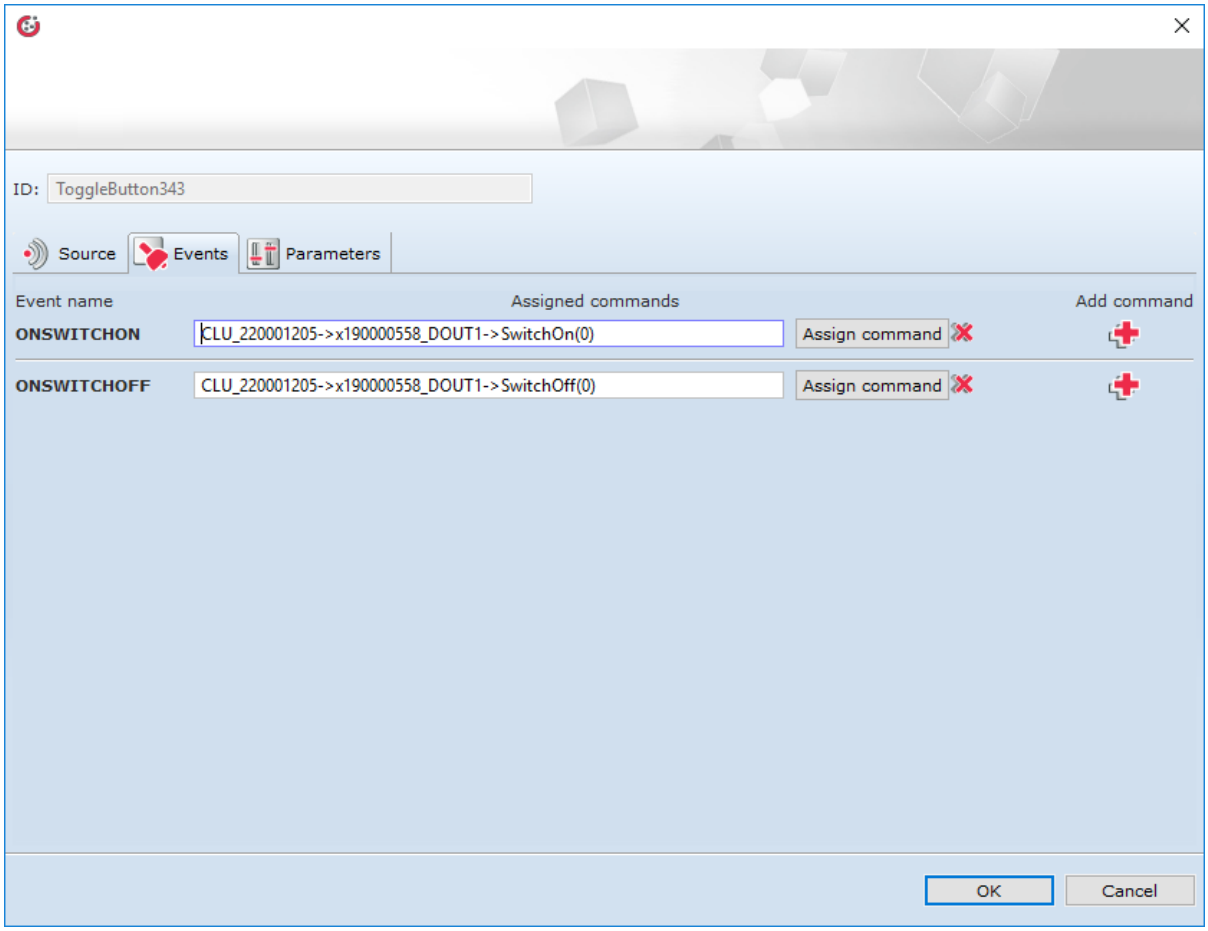
4.6. Adding components and connecting to the system objects.

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:

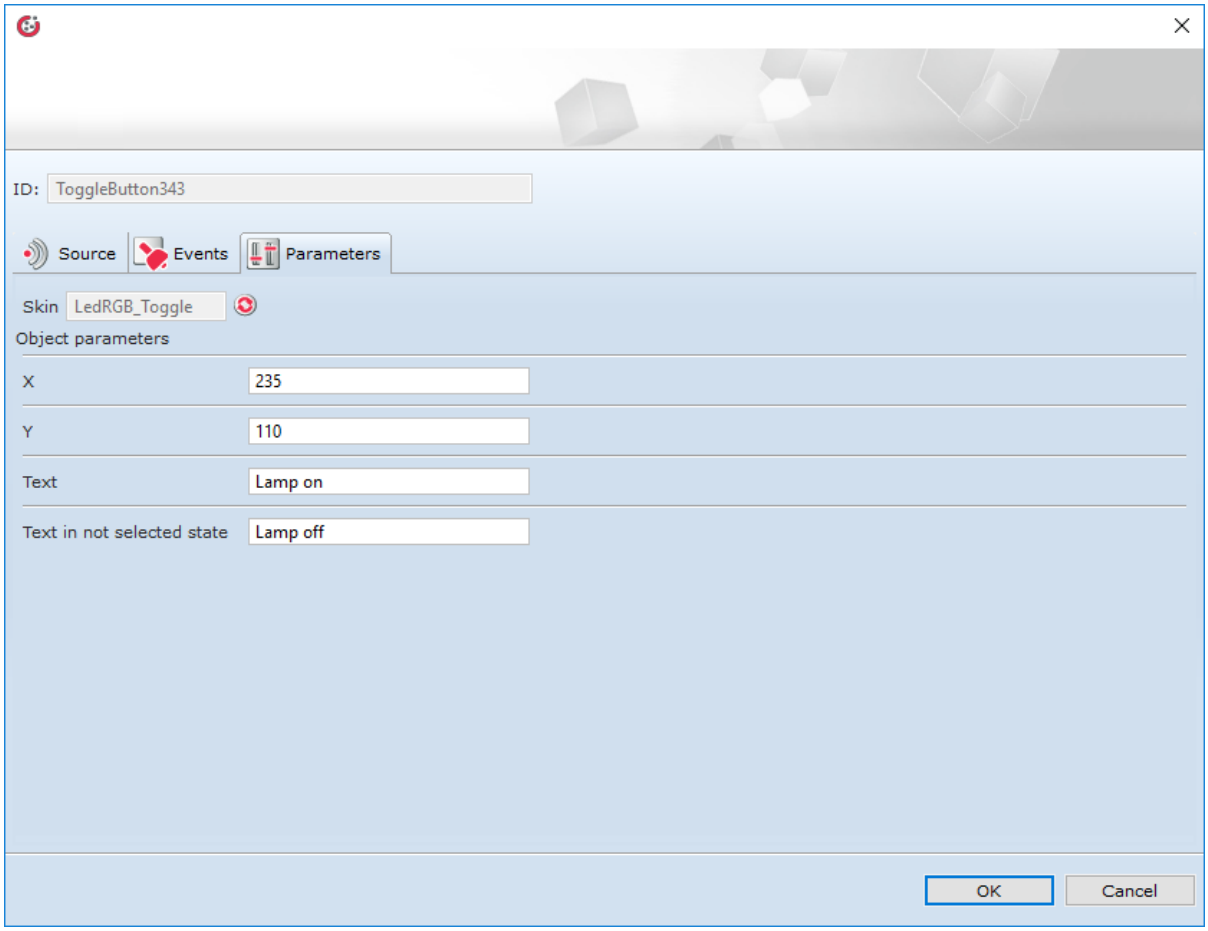
1. In the `Source` tab, select an object which value should be mirrored, and time of refreshing the value, e.g. if you put a slider controlling the dimmer in the interface, then the controlled dimmer must be set as a source so the current value of light can be displayed on the smartphone.



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.

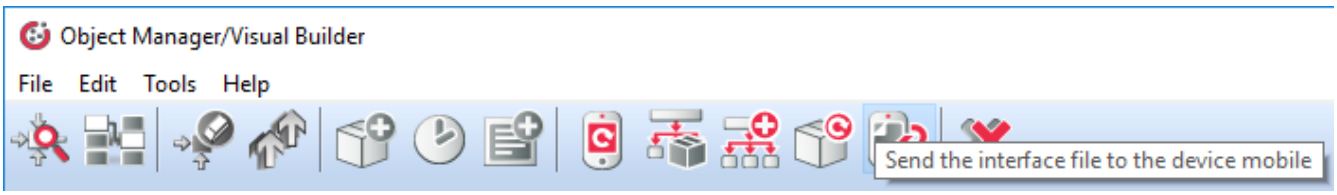
After and during interface creation the user has an option of checking its functionality and appearance. To do that, launch UI Simulator (second tab on the bottom of the page).

4.7. Sending interface to mobile device

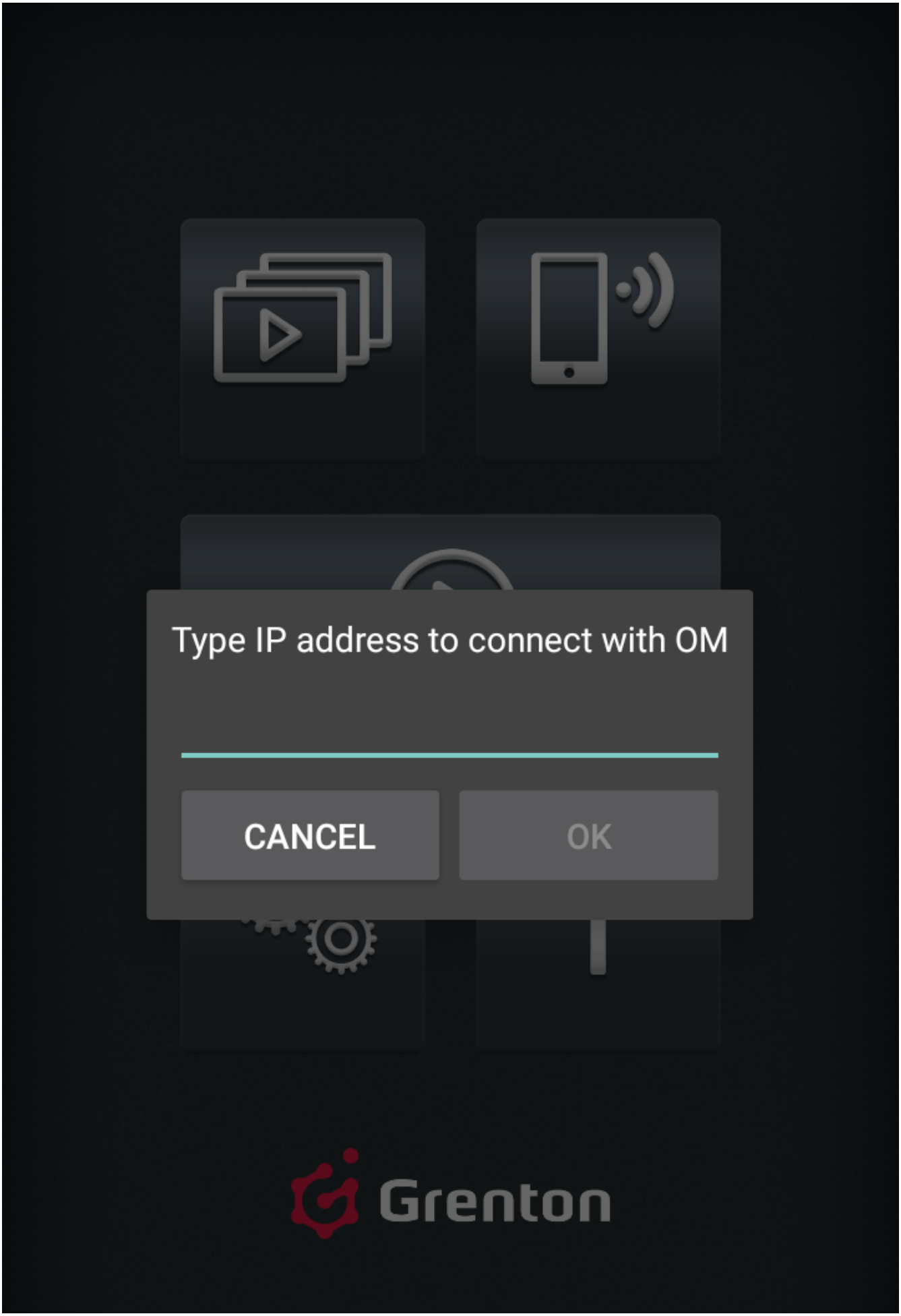
To enable system control by a mobile device, the created interface must be sent to GRENTON HOME MANAGER application installed on the device.

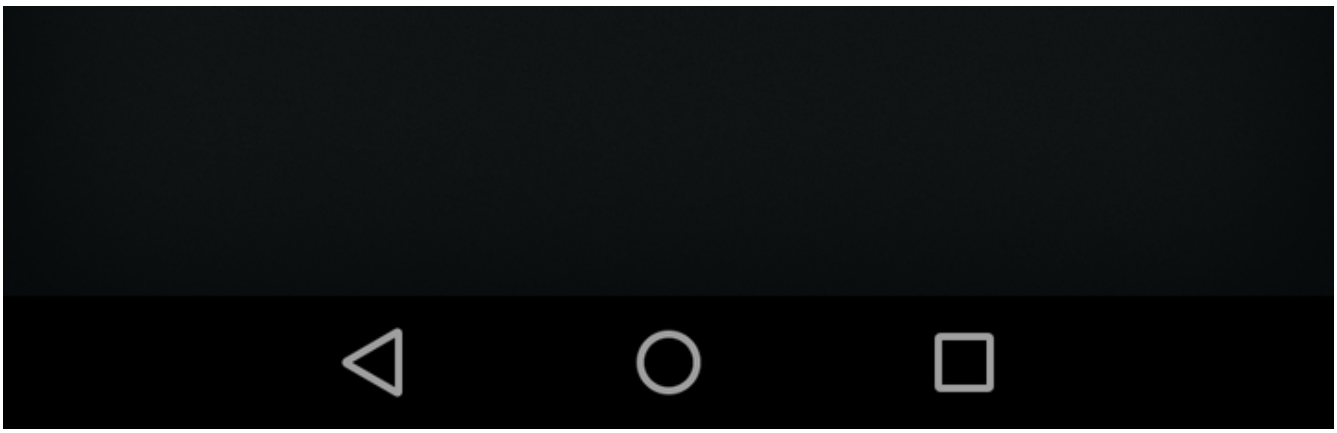
To do that:

- 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*:

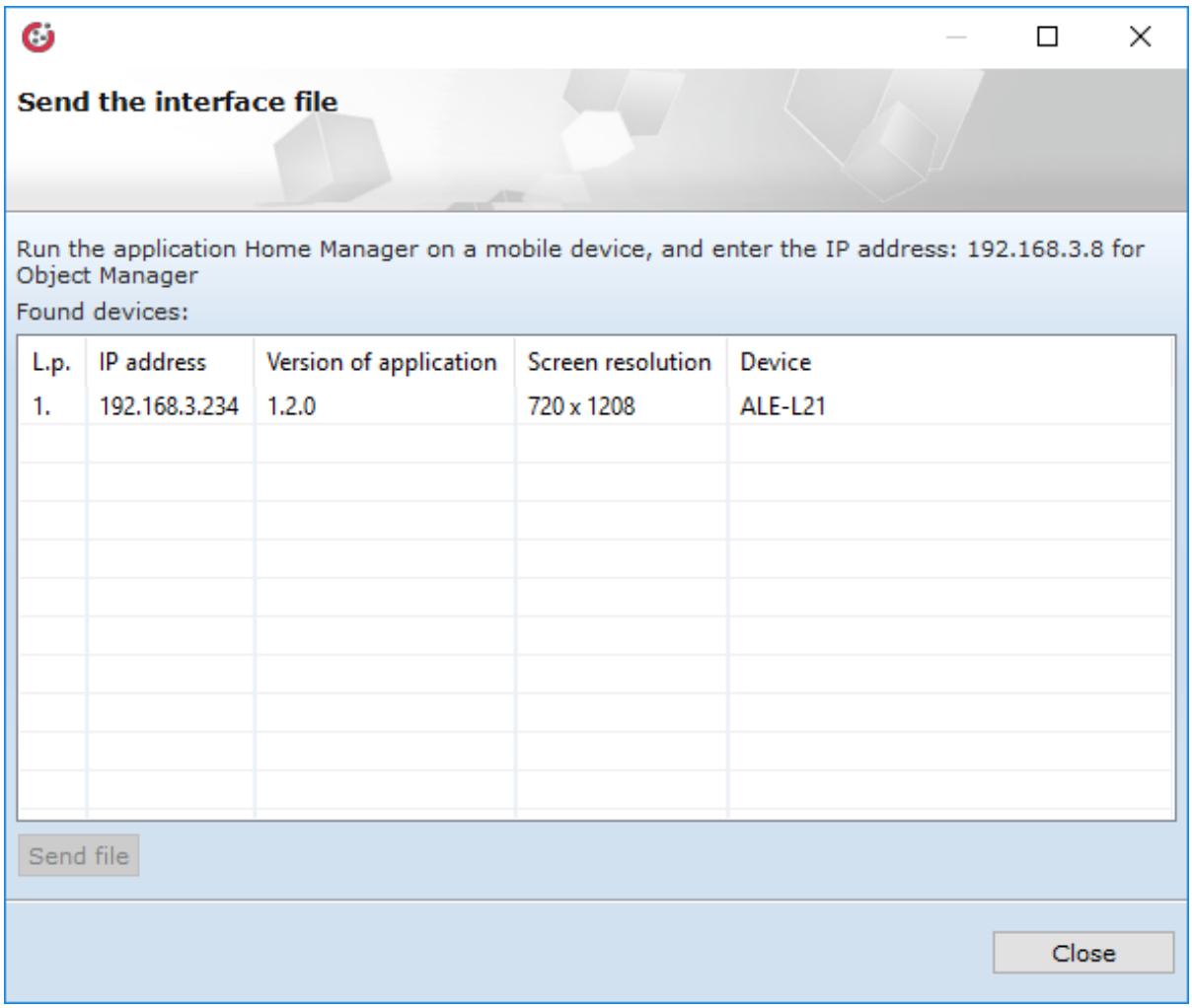

Type IP address to connect with OM
CANCEL
OK
Grenton



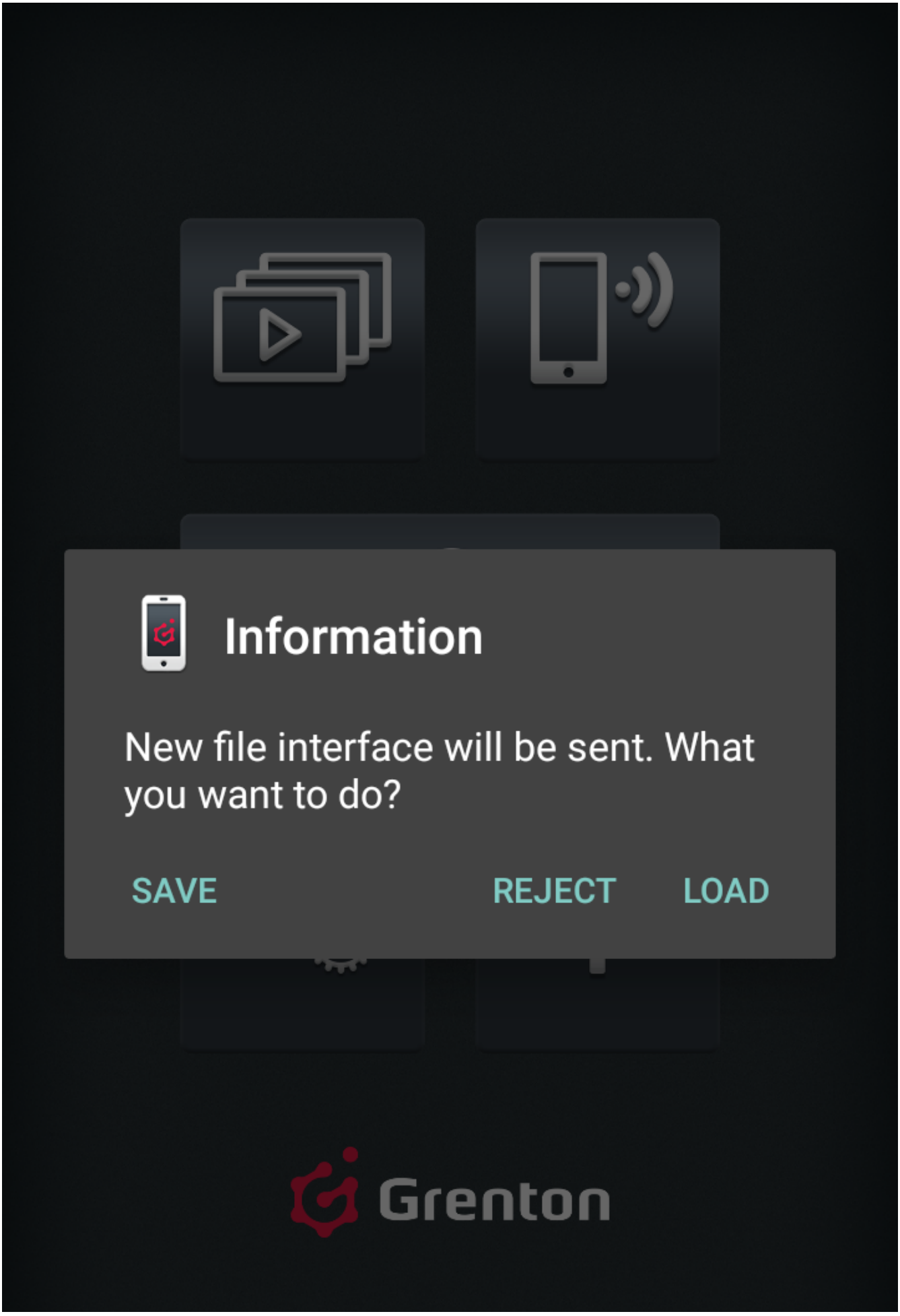
- The mobile device will appear in the send window that was displayed in the Object Manager;

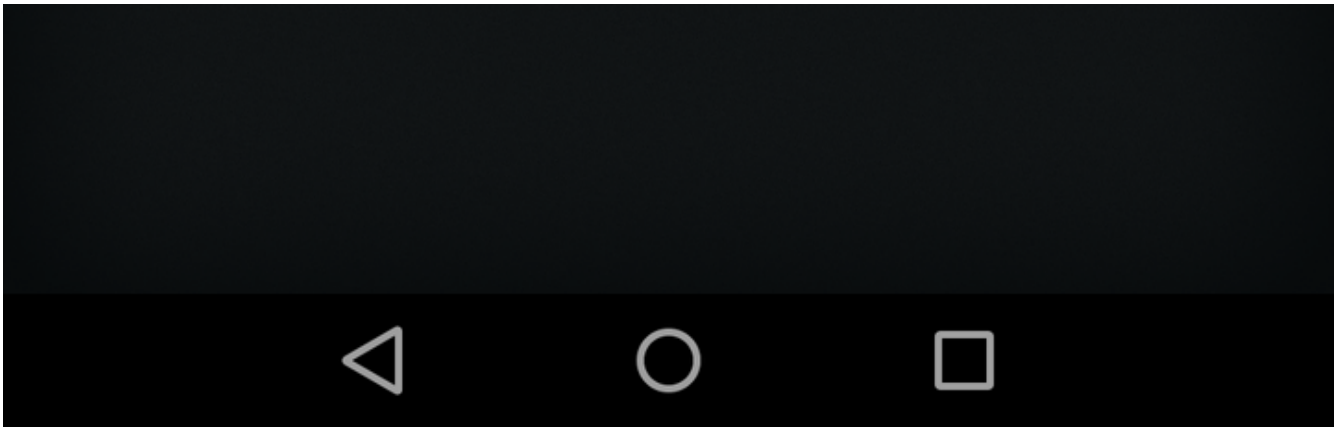
> **NOTE!** The list displays the devices which have GRENTON HOME MANAGER application running, and have `connect to OM` option turned on in the application settings.

- 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*:

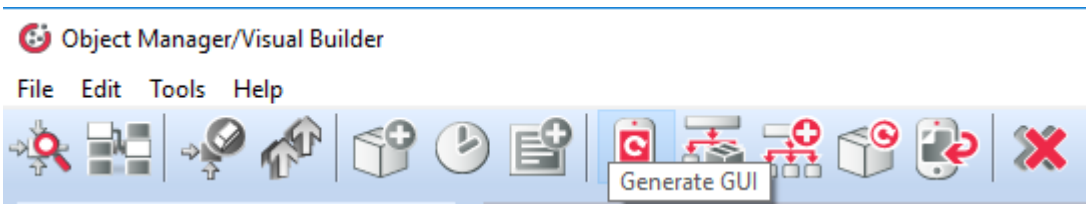

Information
New file interface will be sent. What you want to do?
SAVE
REJECT
LOAD
Grenton



- 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

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:

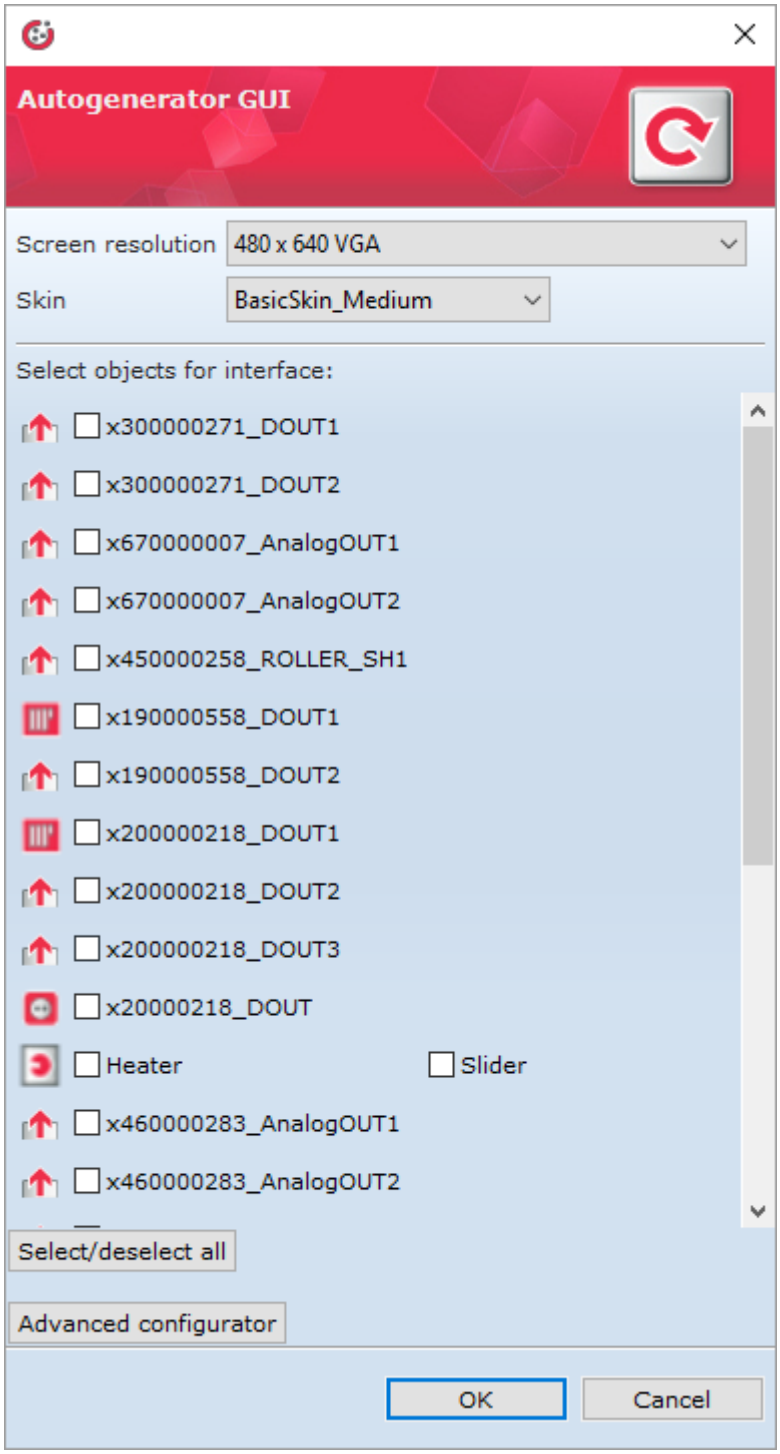


5.1. Creating an interface with available resolution

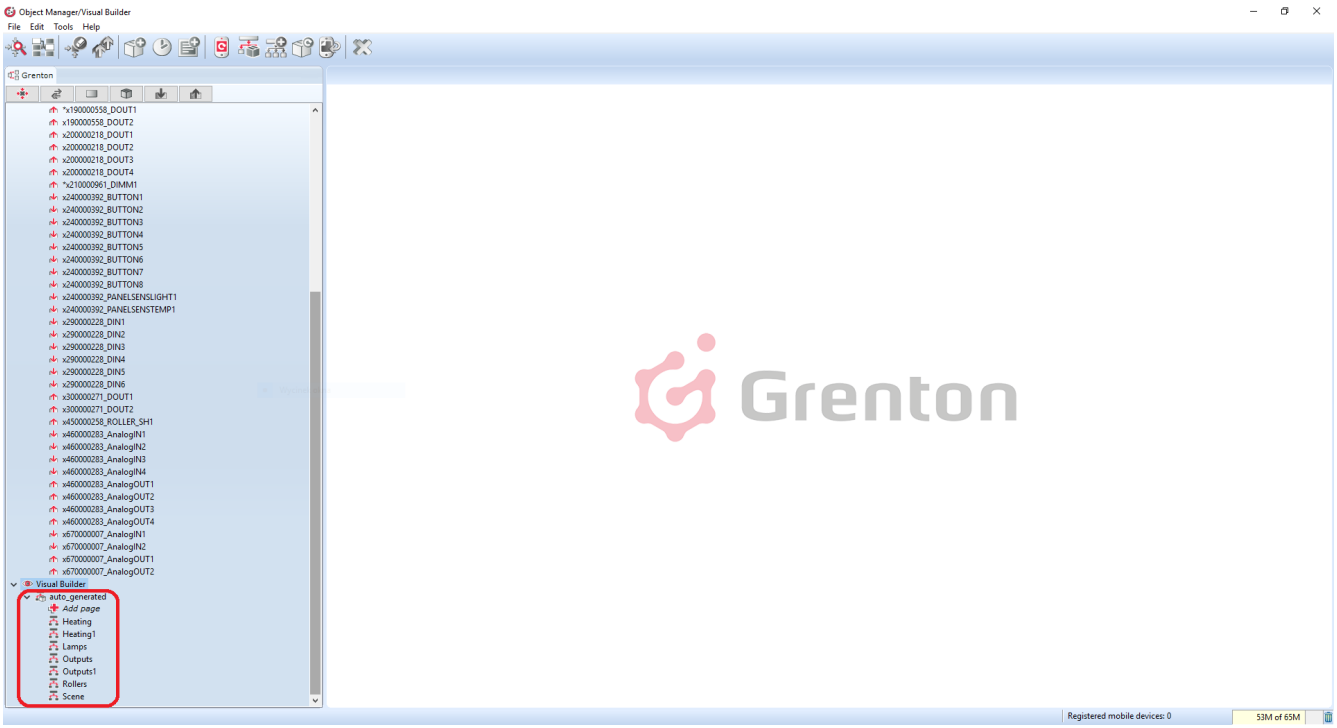
A. Simple configurator

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

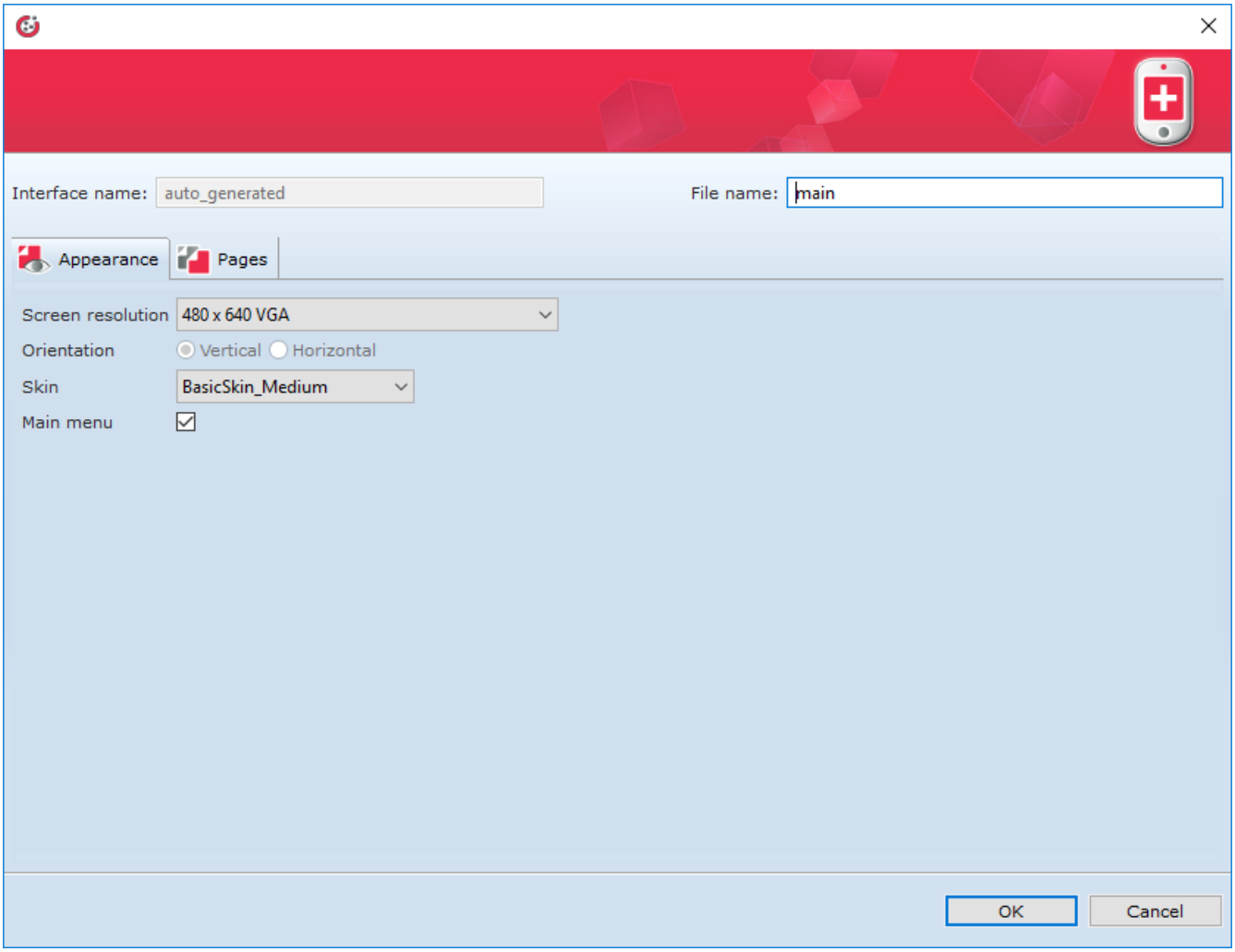


After selecting the objects of interest, click `OK`. As a result, the newly created pages appear in the list of objects (under the icon of the created interface) according to the following figure:



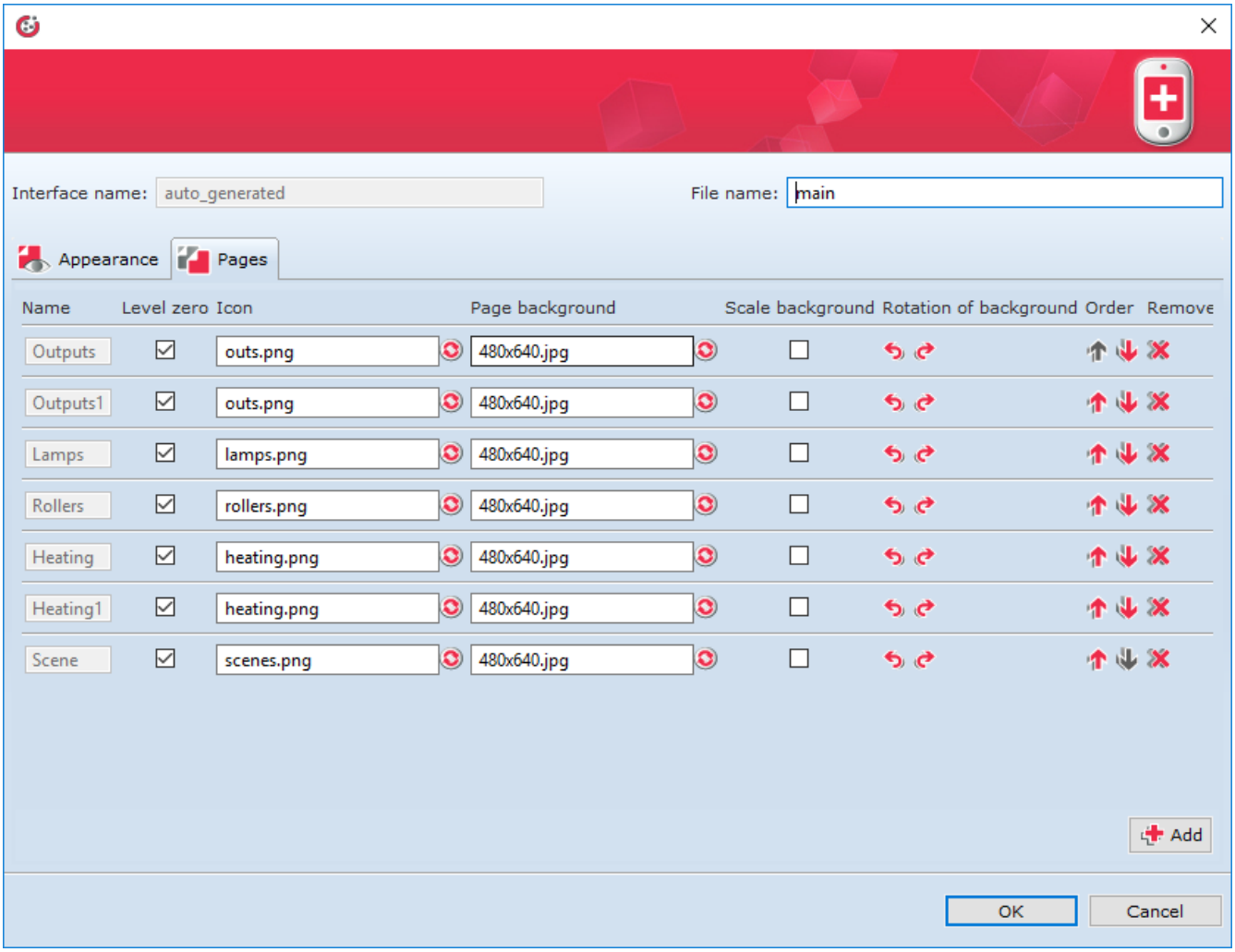
At any time, you can change the interface settings - just double click on its name, and a window will open with two tabs: `Appearance` and `Pages`.

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.



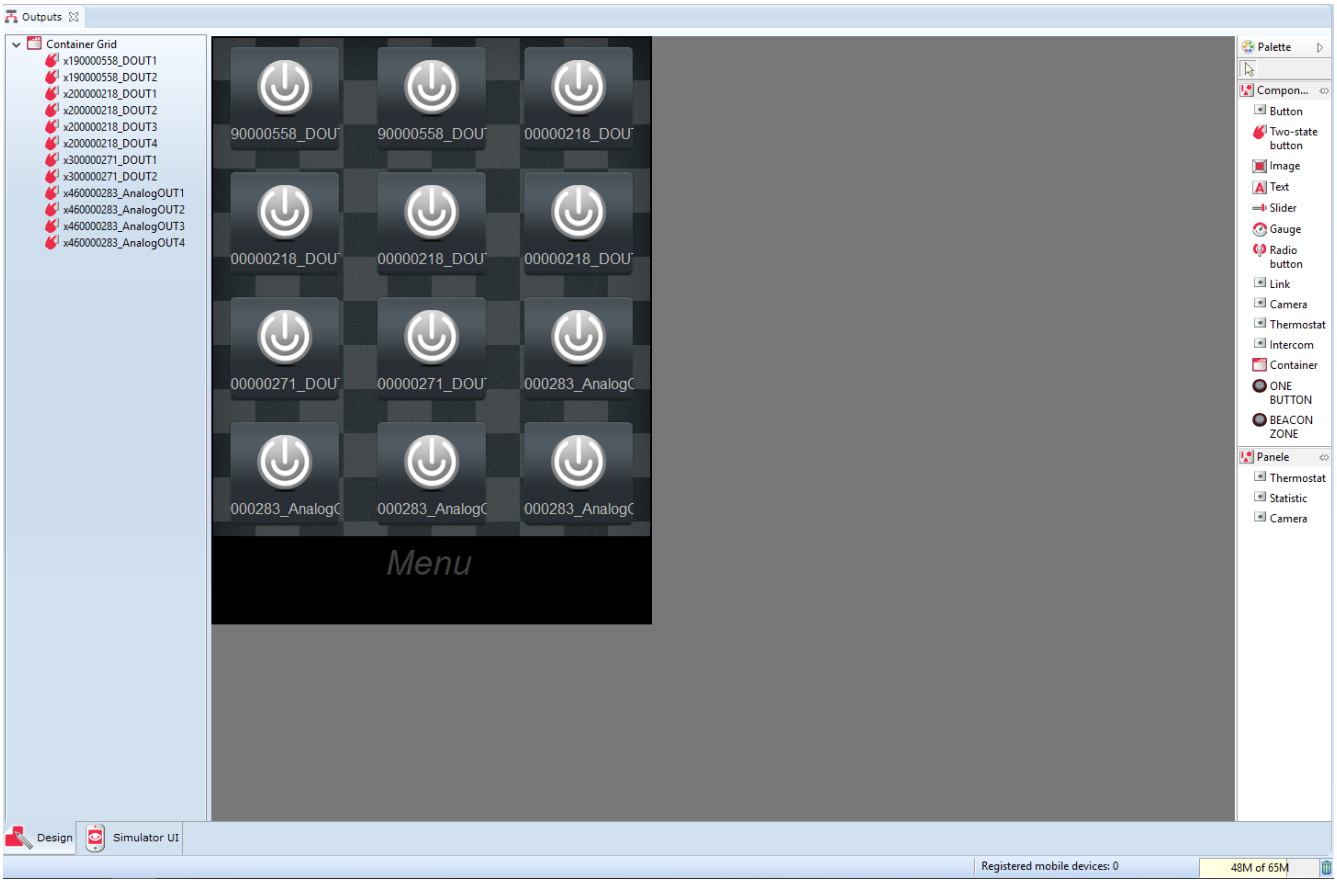
The `Pages` tab contains a list of pages created and allows you to change their parameters, such as:

- **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.

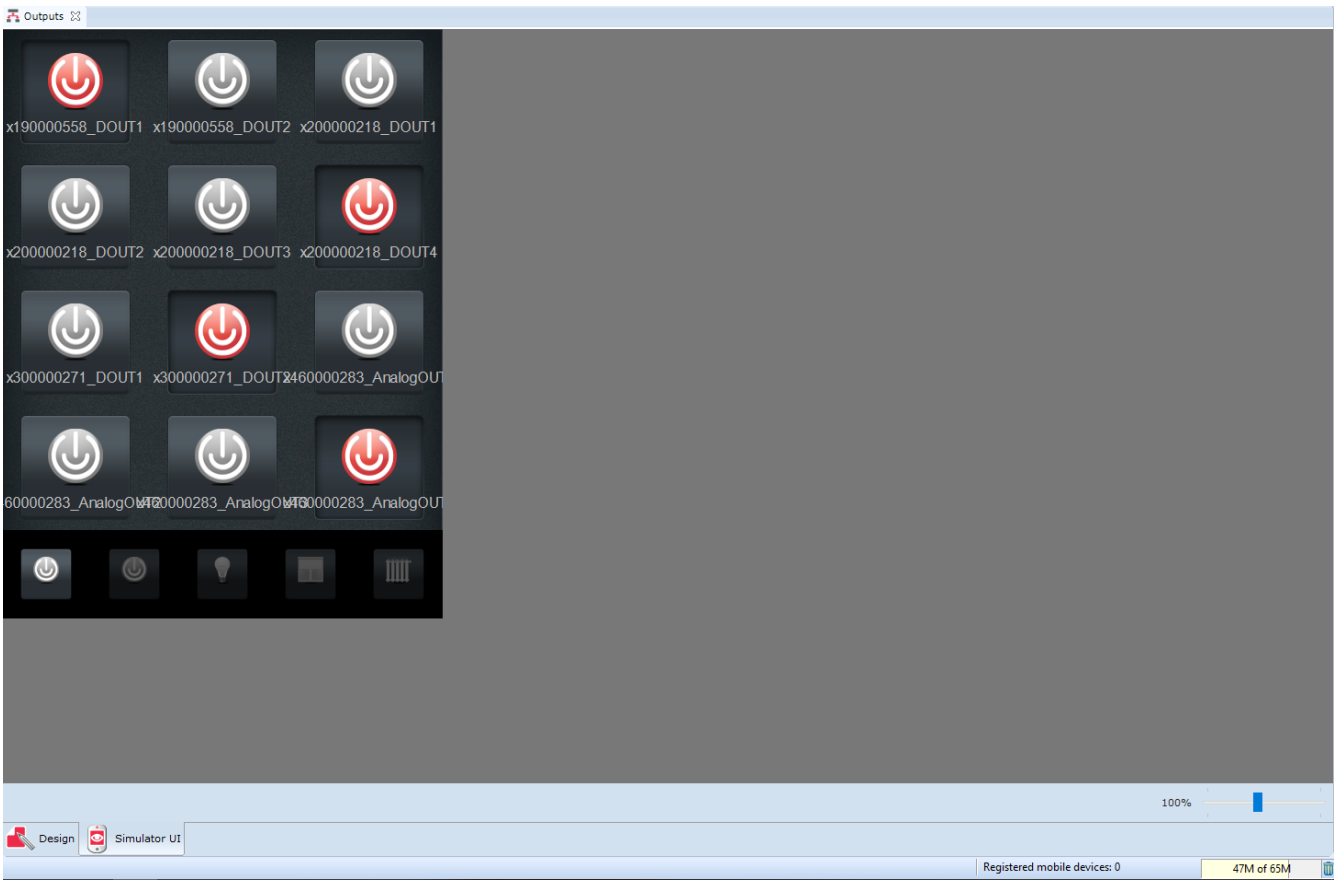


The user also has the option of making changes to the generated pages. Double-clicking on the page icon will open the edit sheet, containing the two tabs `Design` and `Simulator`.

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



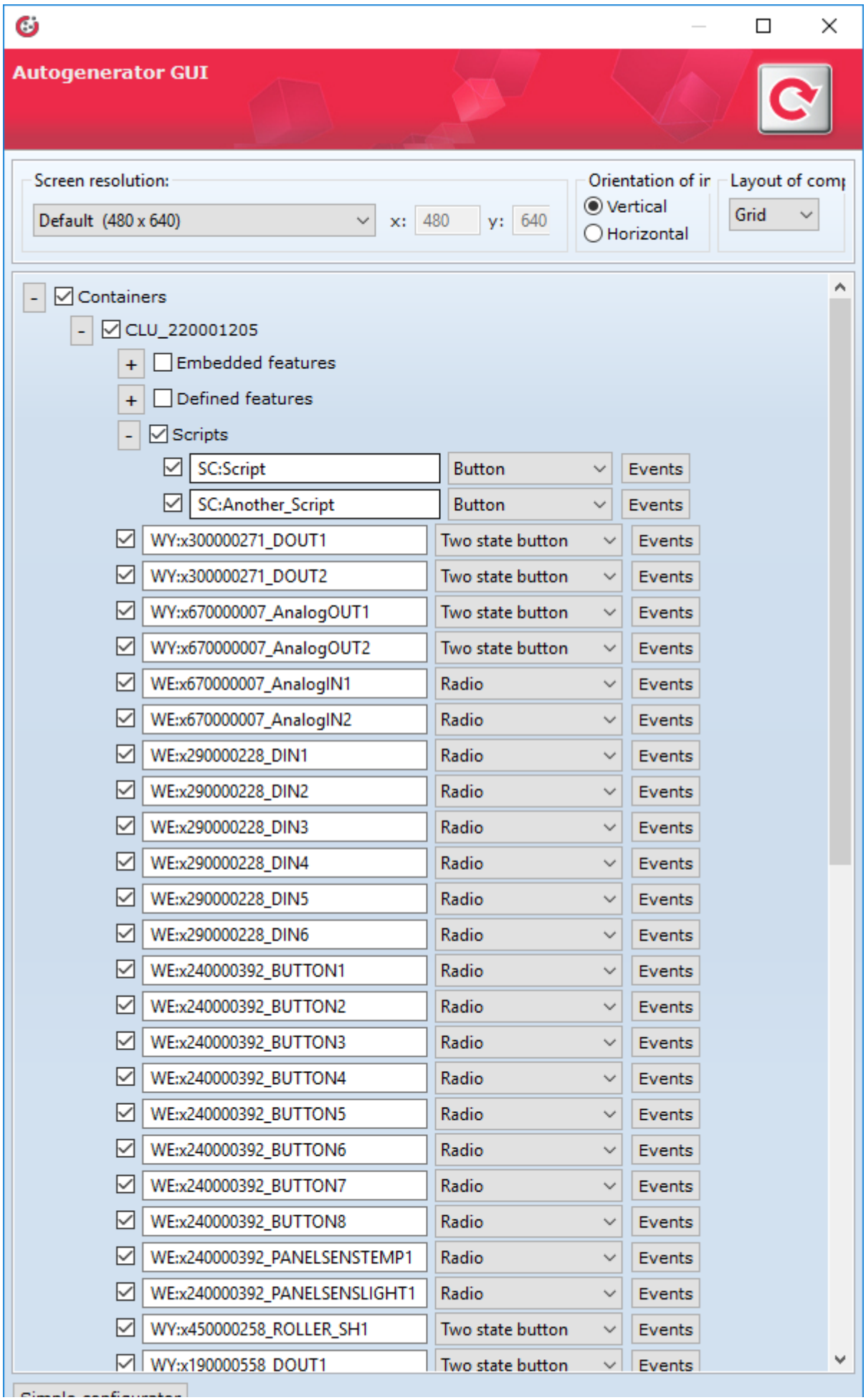
The `Simulator` tab - gives the user the ability to check the appearance and operation of the created interface from the computer screen (before it is sent to the mobile device).

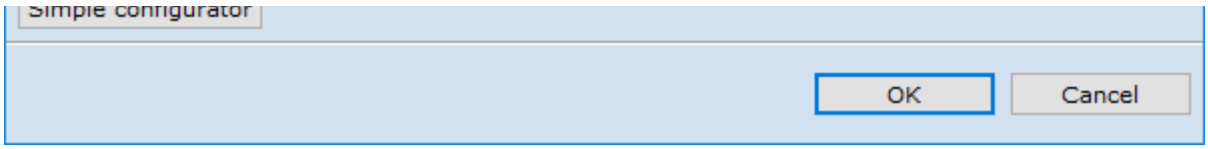


B. Advanced configurator

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:

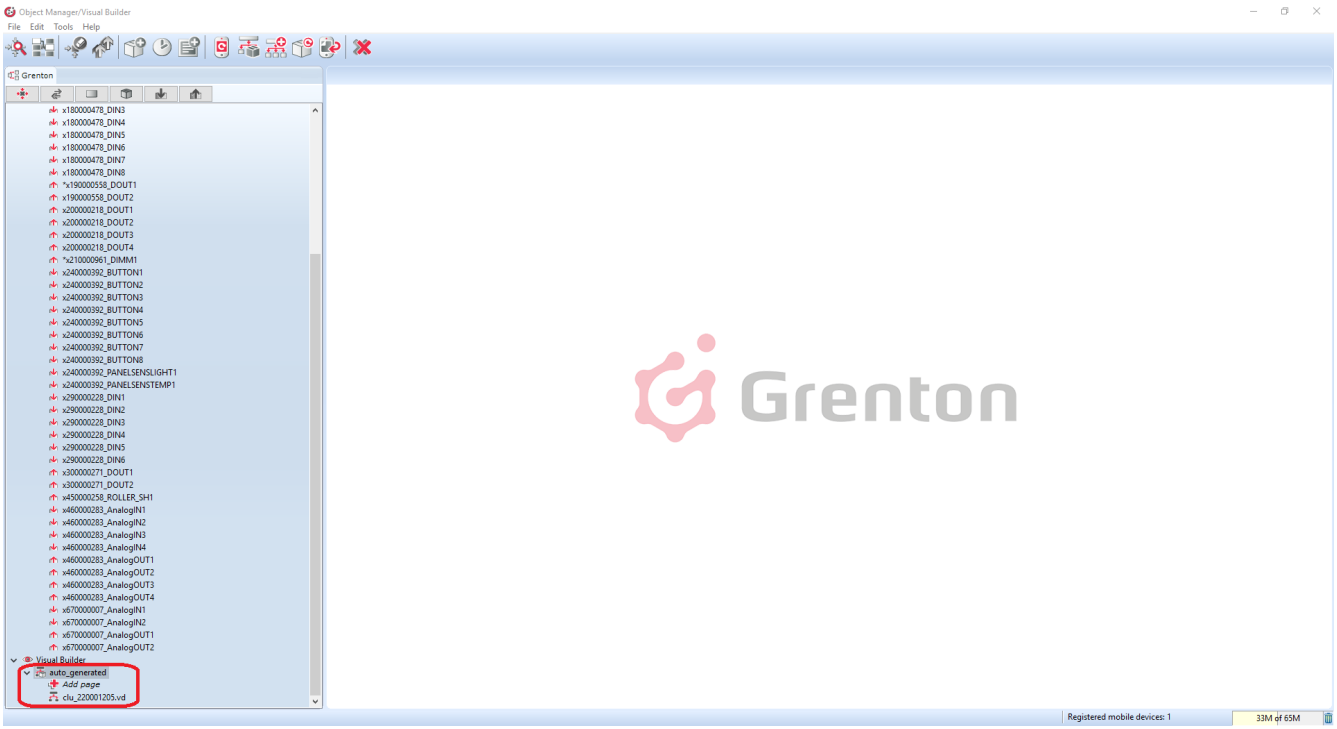
- the resolution with which the mobile device is working;
- interface orientation (vertical or horizontal);
- arrangement of components (grid or list);
- objects and features (from the list of objects) to be included in the created interface;
- displayed icon and events for each object.


Autogenerator GUI
Screen resolution:
Default (480 x 640)
x: 480
y: 640
Orientation of ir
Vertical
Horizontal
Layout of comp
Grid
Containers
CLU_220001205
Embedded features
Defined features
Scripts
SC:Script
Button
Events
SC:Another_Script
Button
Events
WY:x300000271_DOUT1
Two state button
Events
WY:x300000271_DOUT2
Two state button
Events
WY:x670000007_AnalogOUT1
Two state button
Events
WY:x670000007_AnalogOUT2
Two state button
Events
WE:x670000007_AnalogIN1
Radio
Events
WE:x670000007_AnalogIN2
Radio
Events
WE:x290000228_DIN1
Radio
Events
WE:x290000228_DIN2
Radio
Events
WE:x290000228_DIN3
Radio
Events
WE:x290000228_DIN4
Radio
Events
WE:x290000228_DIN5
Radio
Events
WE:x290000228_DIN6
Radio
Events
WE:x240000392_BUTTON1
Radio
Events
WE:x240000392_BUTTON2
Radio
Events
WE:x240000392_BUTTON3
Radio
Events
WE:x240000392_BUTTON4
Radio
Events
WE:x240000392_BUTTON5
Radio
Events
WE:x240000392_BUTTON6
Radio
Events
WE:x240000392_BUTTON7
Radio
Events
WE:x240000392_BUTTON8
Radio
Events
WE:x240000392_PANELSENSTEMP1
Radio
Events
WE:x240000392_PANELSENSLIGHT1
Radio
Events
WY:x450000258_ROLLER_SH1
Two state button
Events
WY:x190000558_DOUT1
Two state button
Events



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:

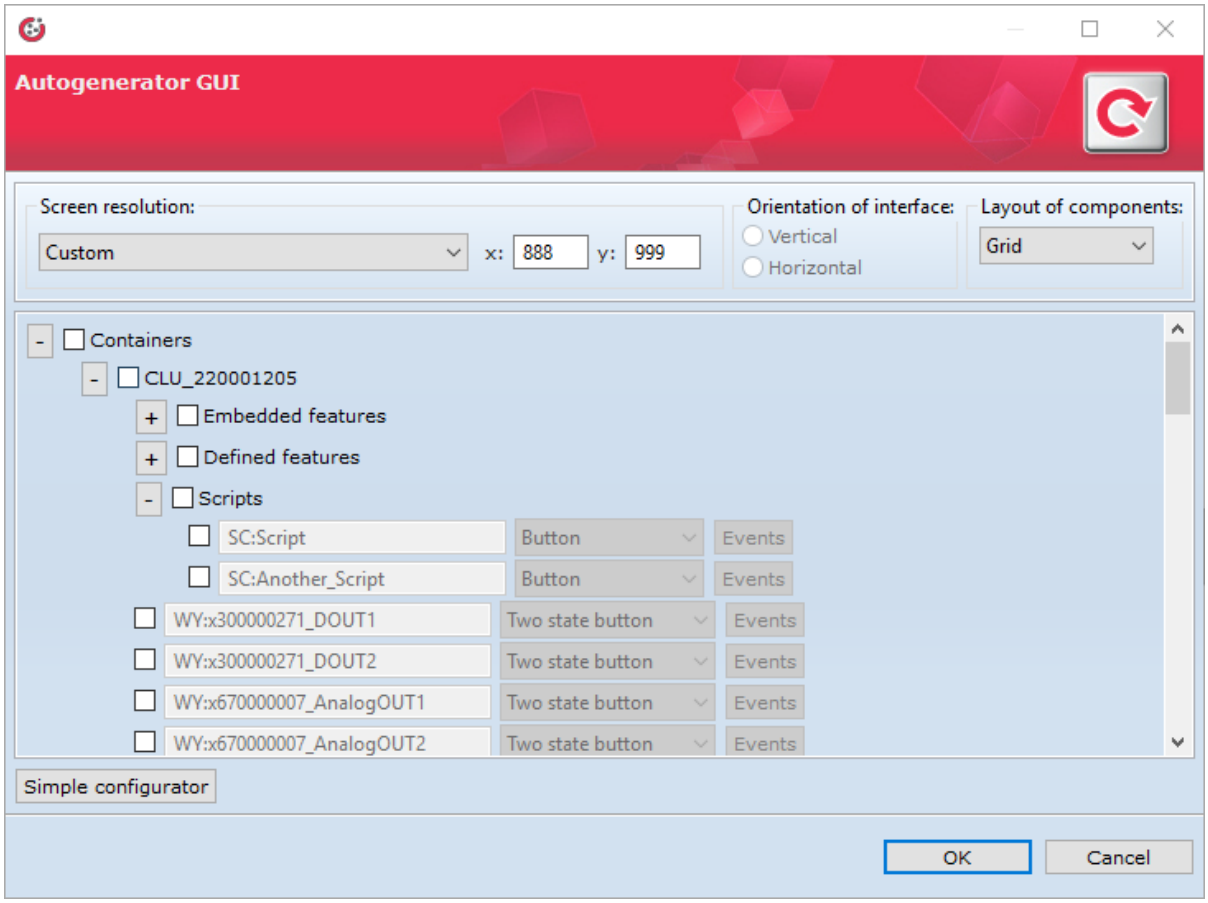


As with the simple configurator - the user has the option of making changes to the generated pages. Double-clicking on the page icon will open the edit sheet, containing the two tabs `Design` and `Simulator`.

5.2. Creating an interface with its own resolution

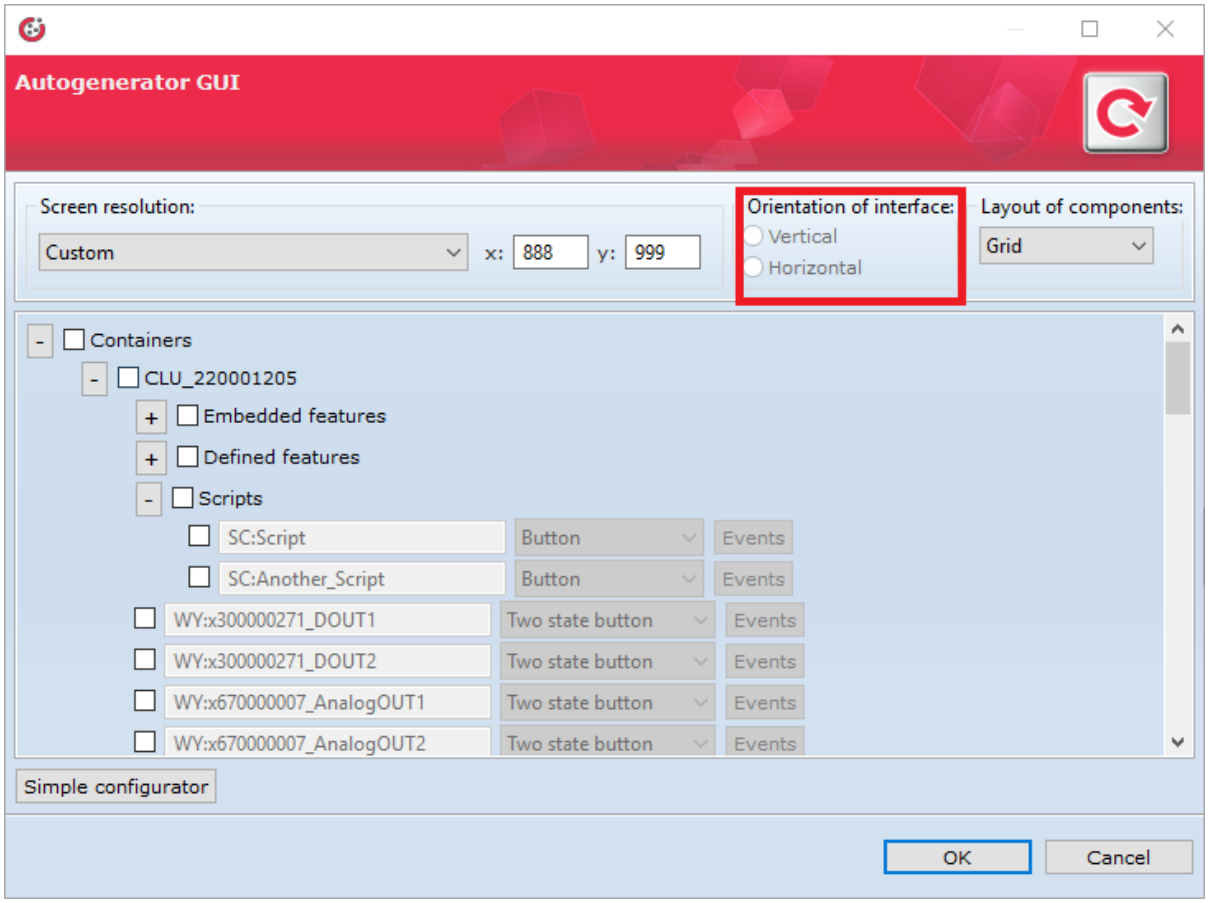
In the case of the advanced configurator, it is possible to create an interface with its own, selected resolution. To do this:

- 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 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. Android screen widgets

GRENTON system can be controlled directly from the desktop of a mobile device thanks to widgets. Widgets are designed for tablets and smartphones with ANDROID operational system. To enable this function, the device must have GRENTON HOME MANAGER application installed.

Widgets use connections between objects in the interface sent to application and devices in the system. Thus, to create new widget, there must be at least one savedinterface in the application.

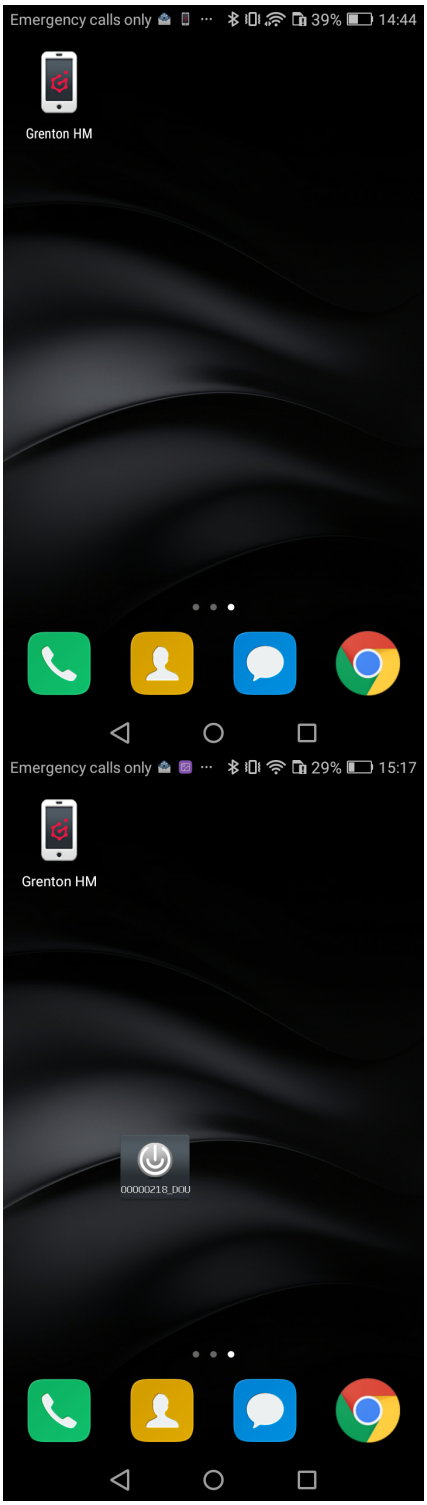
A. Widget creation

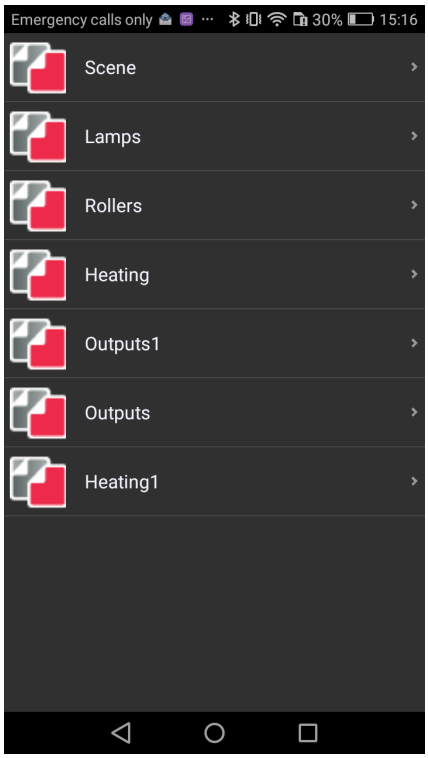
You have to start adding widgets on your mobile device and then find a widget named *HOME MANAGER* on the list.

> **NOTE!** Widgets list may be displayed in various ways on different devices.

List of interfaces saved in the HOME MANAGER application will be displayed on screen. When creating a widget you can select only objects from the saved interfaces.

- Select an interface, in which the object you're interested in is placed, from the list.
- List of available objects will be displayed on screen.
- After selecting one of available objects, it will be placed on the desktop.





> **NOTE!** Widgets are connected to the objects within interfaces. In the case of interface deletion or changing its configuration, all widgets connected to it will stop working.. One needs to delete not working widgets from pulpit and create new ones with current connections in their place.

7. Video intercom configuration

7.1. 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 doorphone - it will be needed when configuring the *SIP* server.

C. SIP configuration:

- To create a video intercom configuration you need at least two *SIP* accounts;
- Using the browser, log in to the video intercom [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.

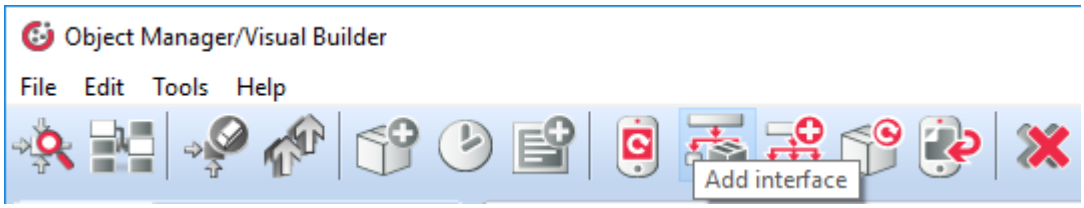
> **NOTE!** If in the setting of Intercom, it is necessary to select one account from several configurable, select the previously selected one - in the example **Account_1**!

7.2. Creation and configuration of the application interface

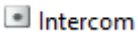
A. Adding a door intercom to the application interface in the Object Manager

In order to add a video intercom to the interface:

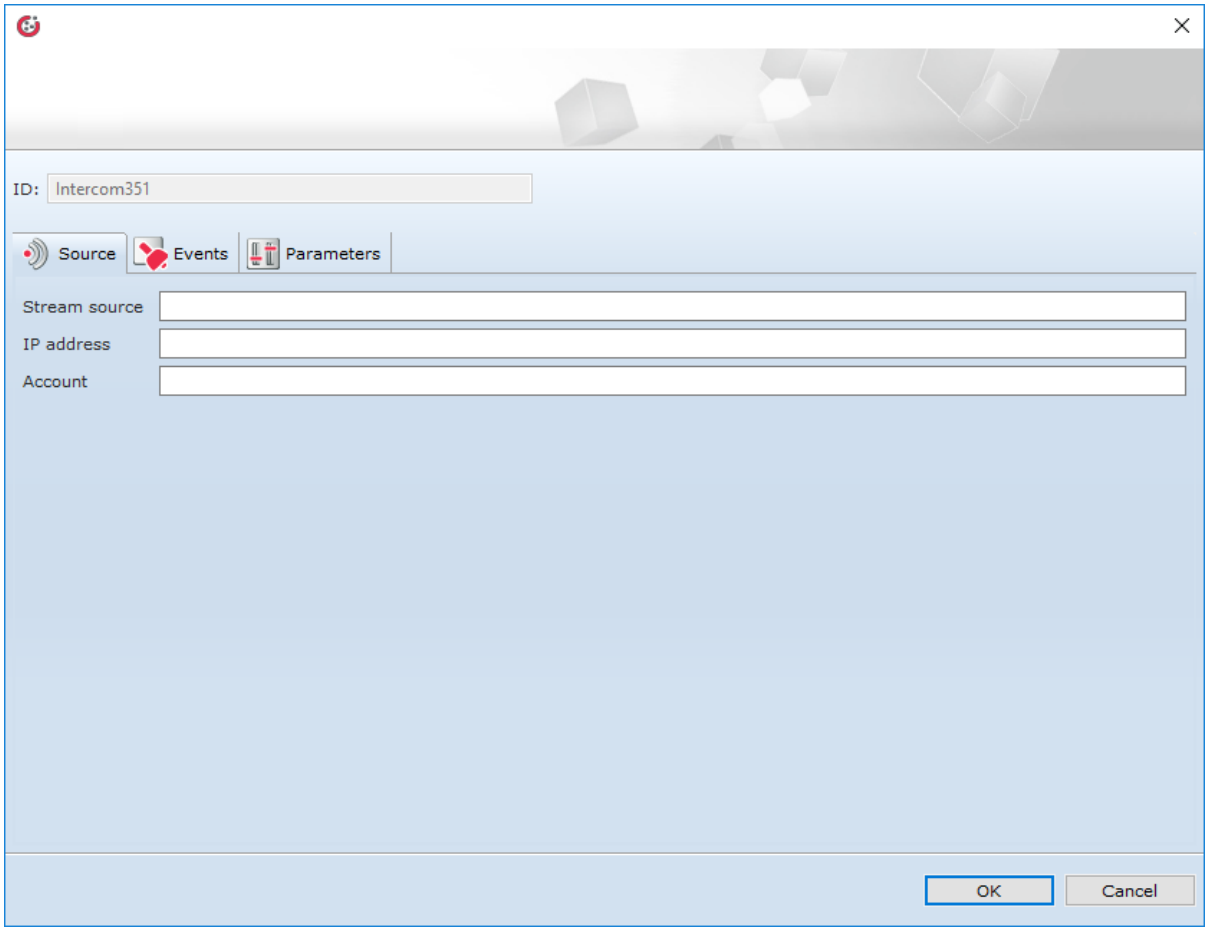
- From the main menu, click *Add interface*:



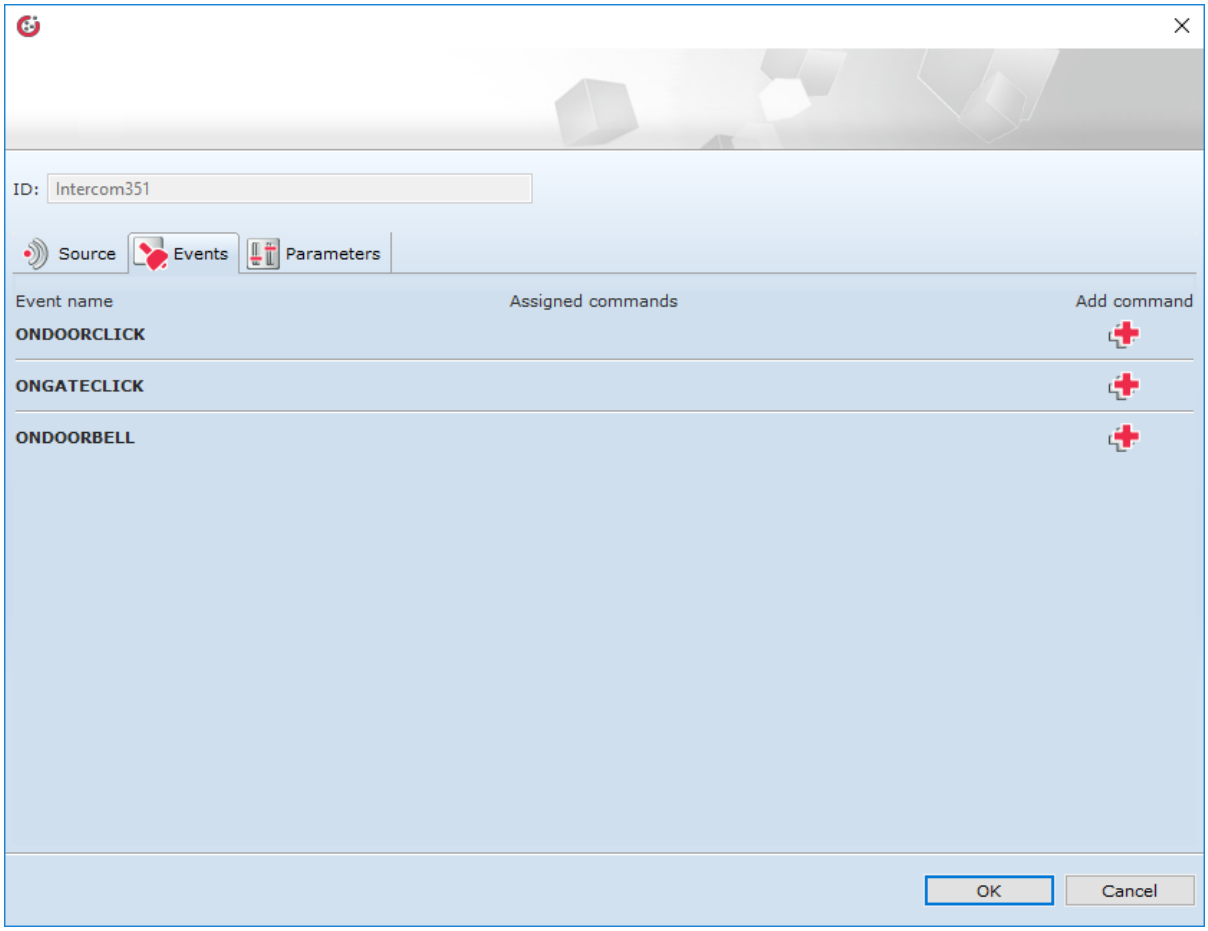
- Configure interface settings - choose: resolution, name, skin, add at least one page;
- To the created page - from the component palette - add the button *Intercom*:



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

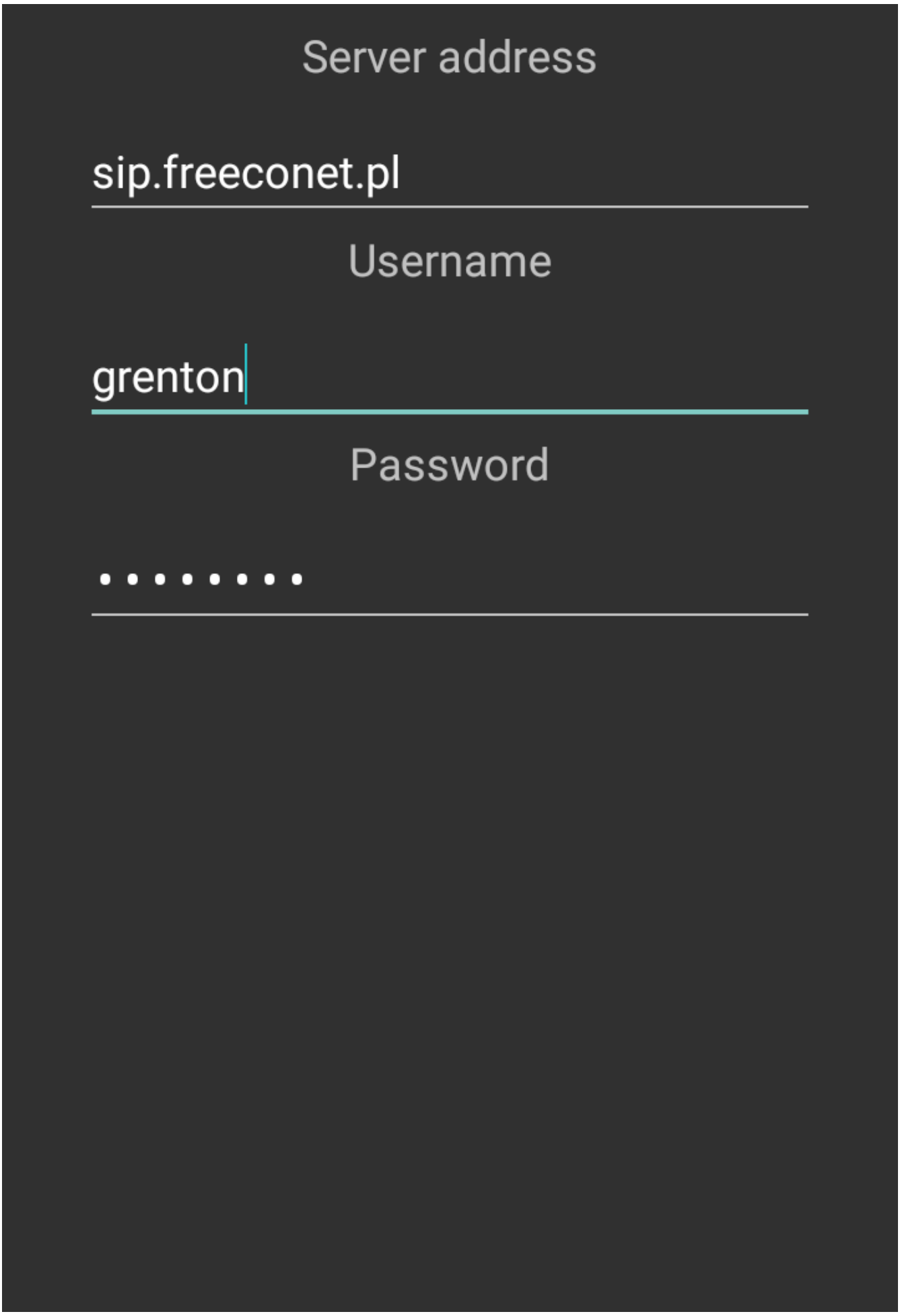


- Click OK;
- Send the interface to the mobile device - look up VIII.4.7.

B. Home Manager application configuration

In order to carry out the configuration:

- Open the Home Manager application;
- Select *Settings* from the main menu (gearstick pictogram);
- From section *Intercom* select *SIP configuration* [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:

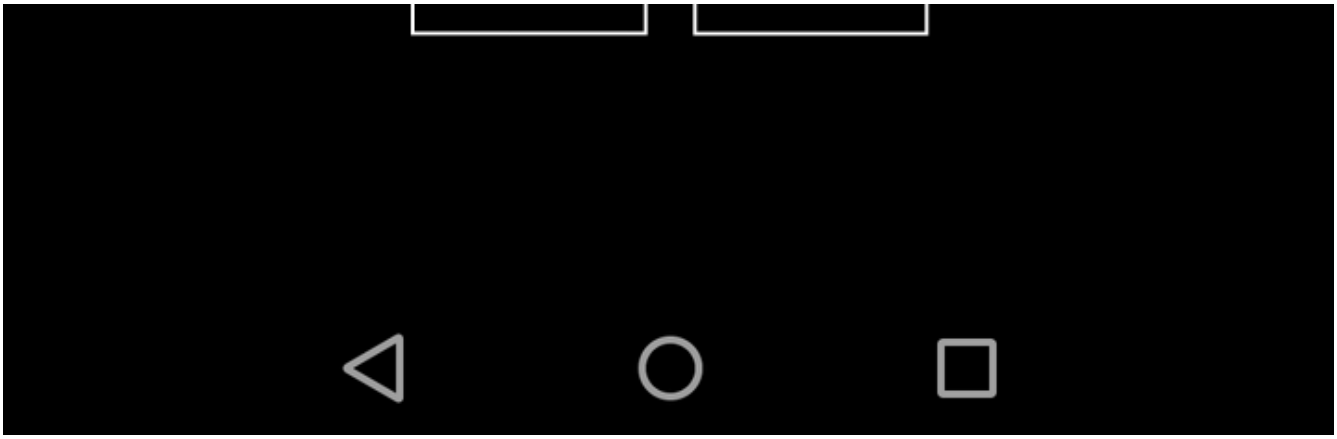

Server address
sip.freeconet.pl
Username
grenton
Password
••••••••

- 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.

7.3. Making a call from the intercom

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

INTERCOM



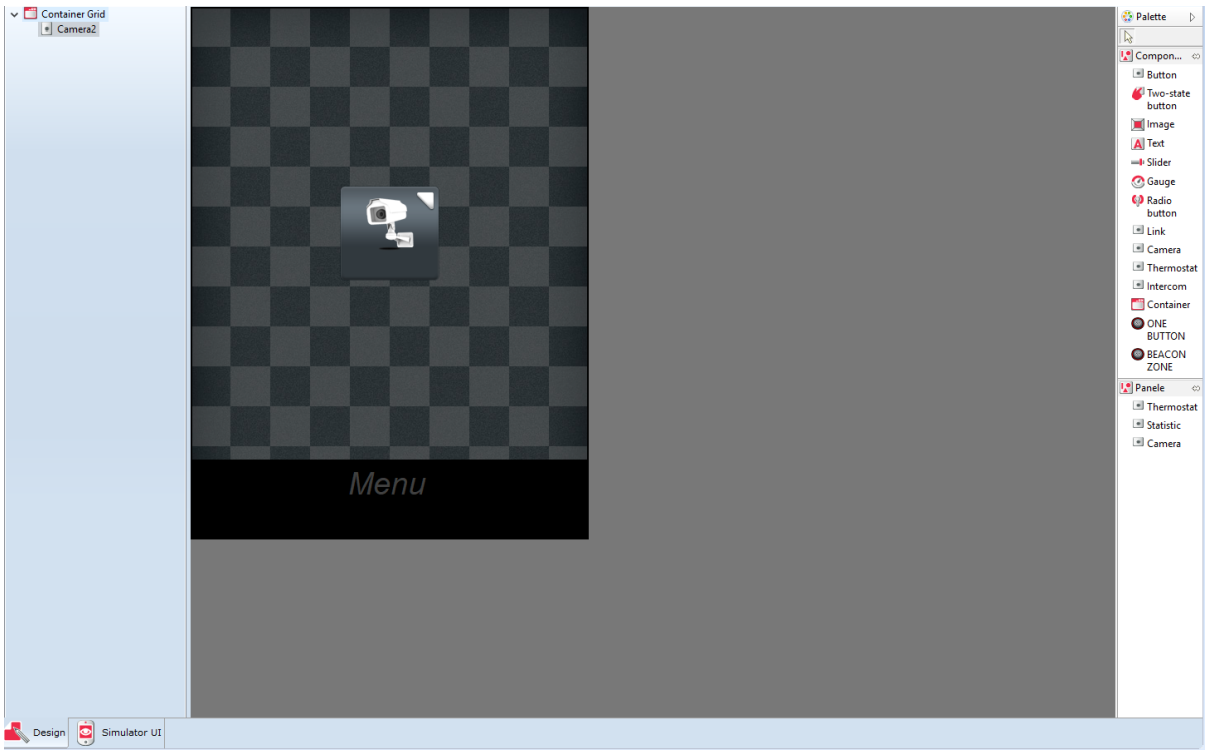
8. IP cameras image operation

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.

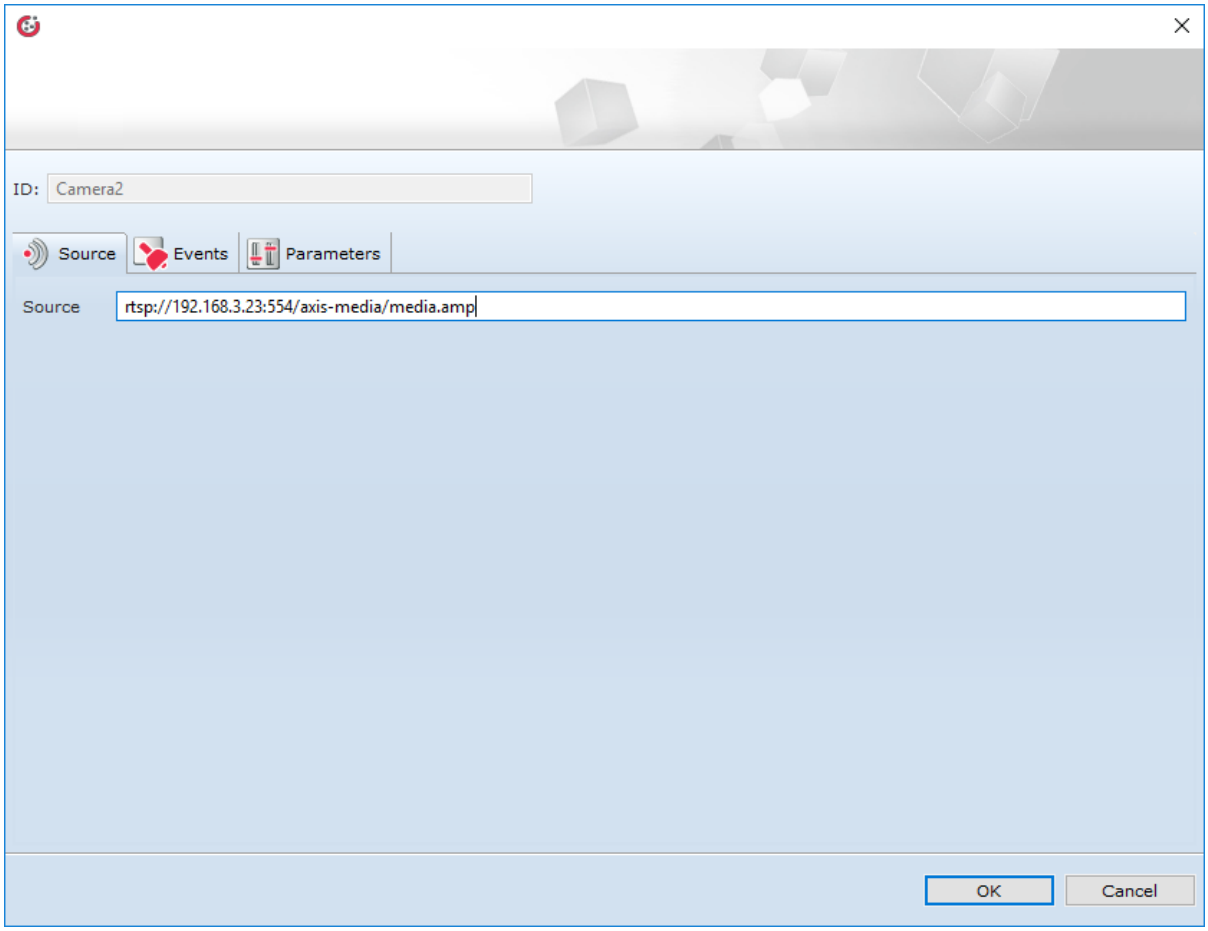
> **NOTE!** Home Manager application displays correctly images of cameras supporting RTSP protocol and h264 codec in MPEG standard.

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.

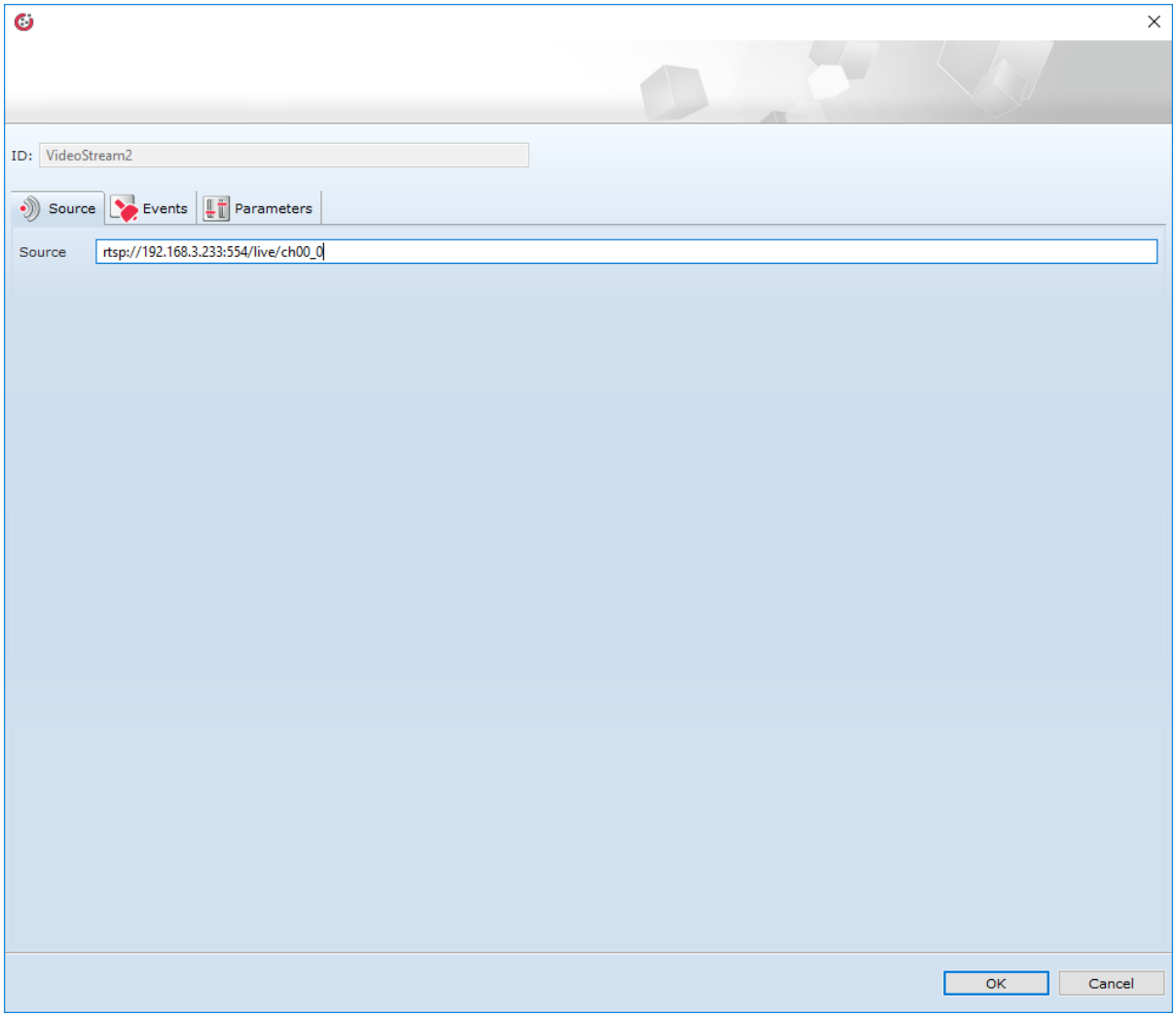


After sending created interface, image from the camera will be displayed on the screen of the mobile device after clicking added object.

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*.

9. Remote access of the mobile application to the system

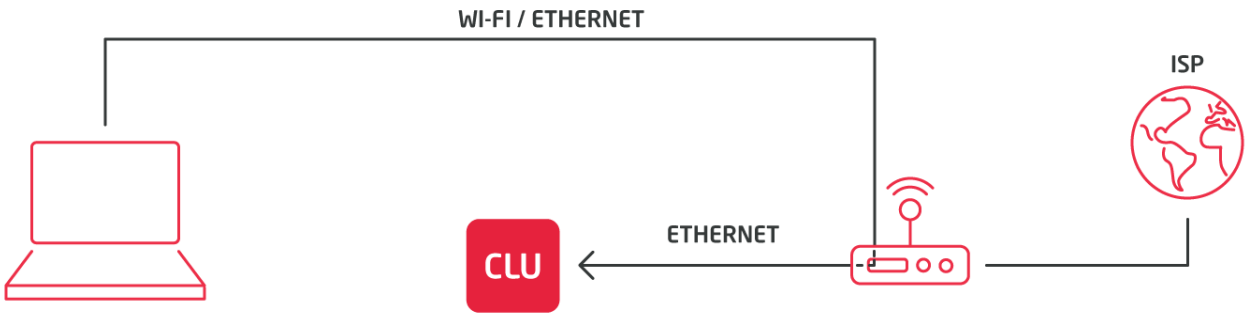
The Grenton system gives you the freedom to control your home from anywhere in the world. Sitting at work, or being on a business trip, we can easily control the state of our investment and manage its functions in a very simple way.

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.

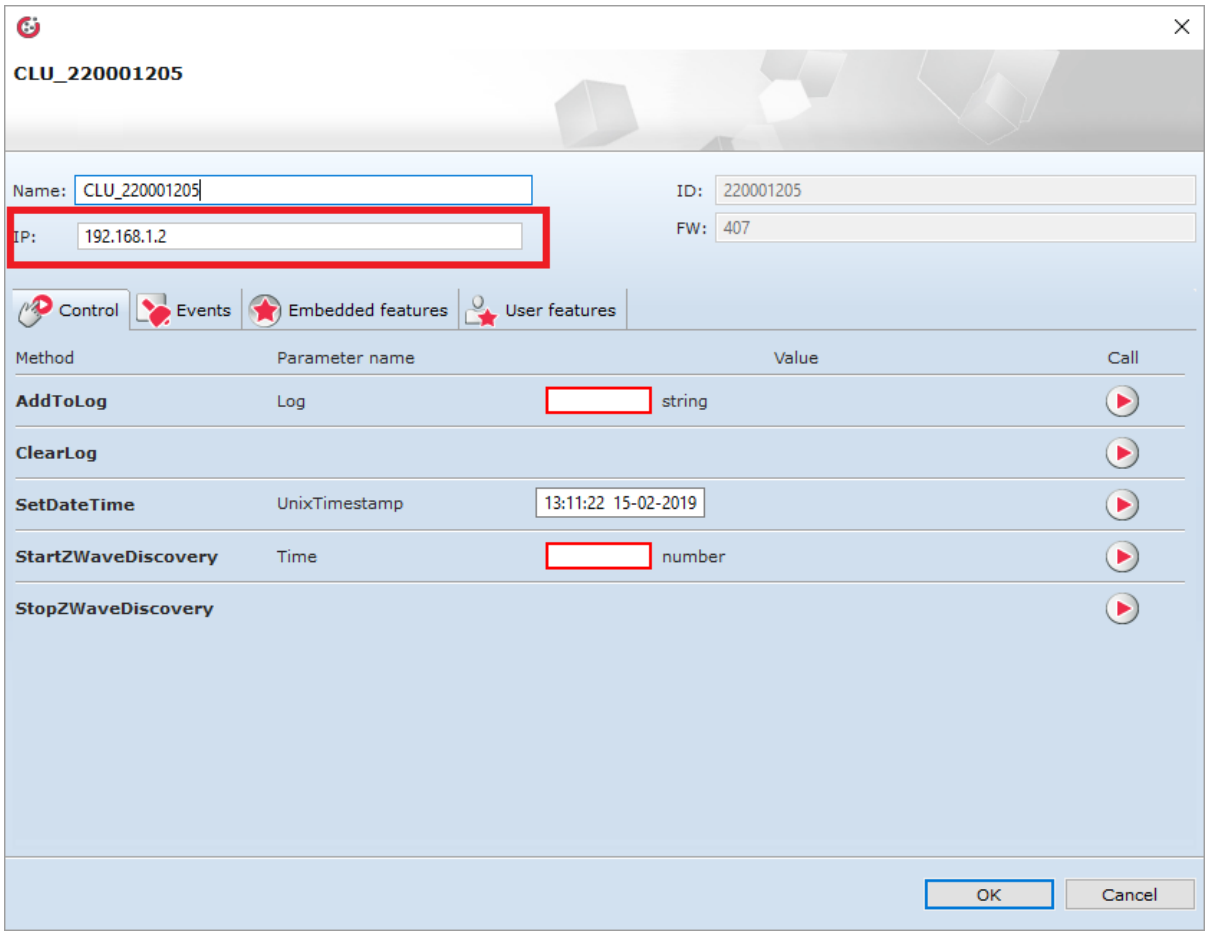
9.1. System configuration

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.

9.2. Port routing setting on the local network router

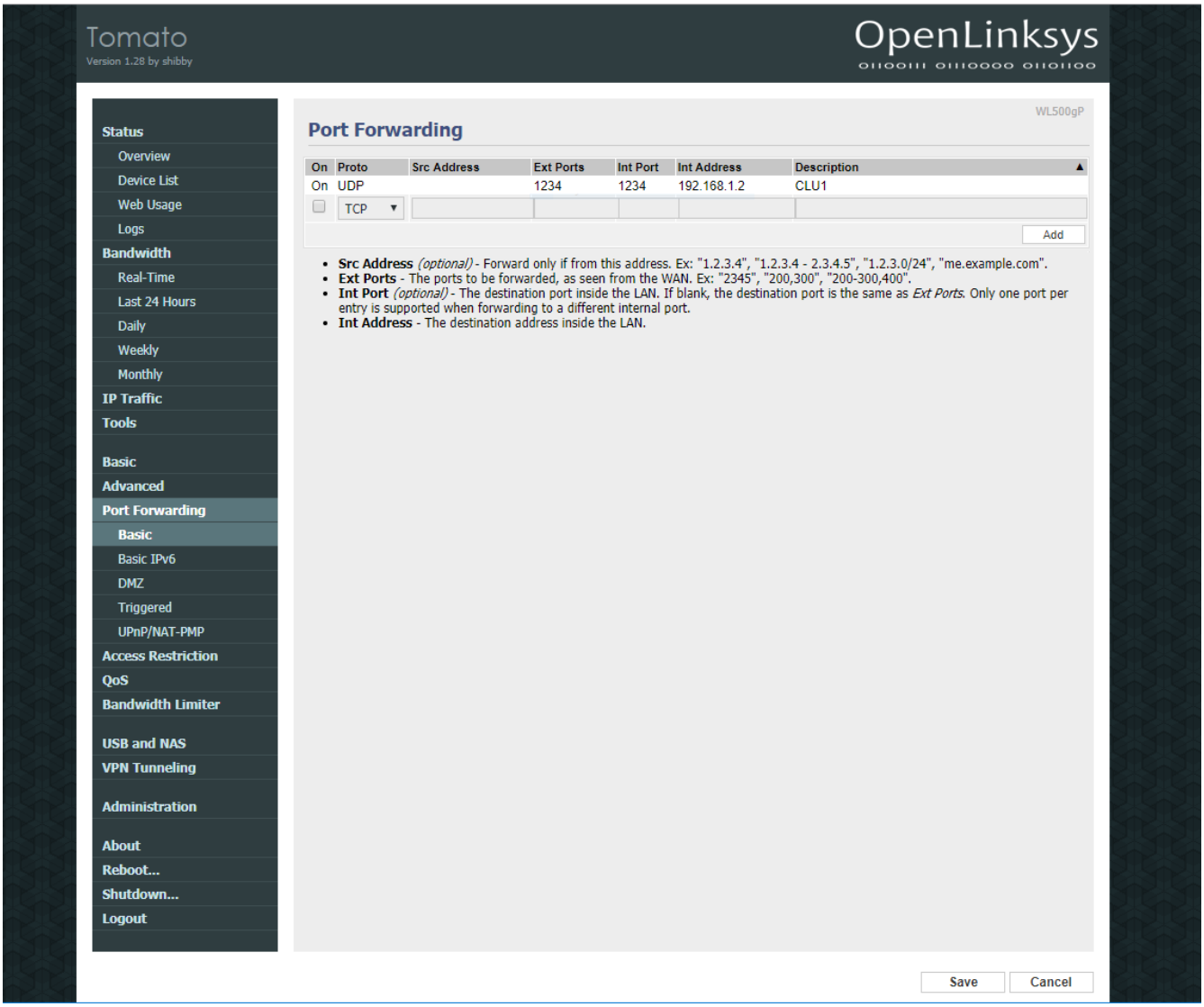
> **NOTE!** The port routing settings for each router may vary! The general procedure is presented below.

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.

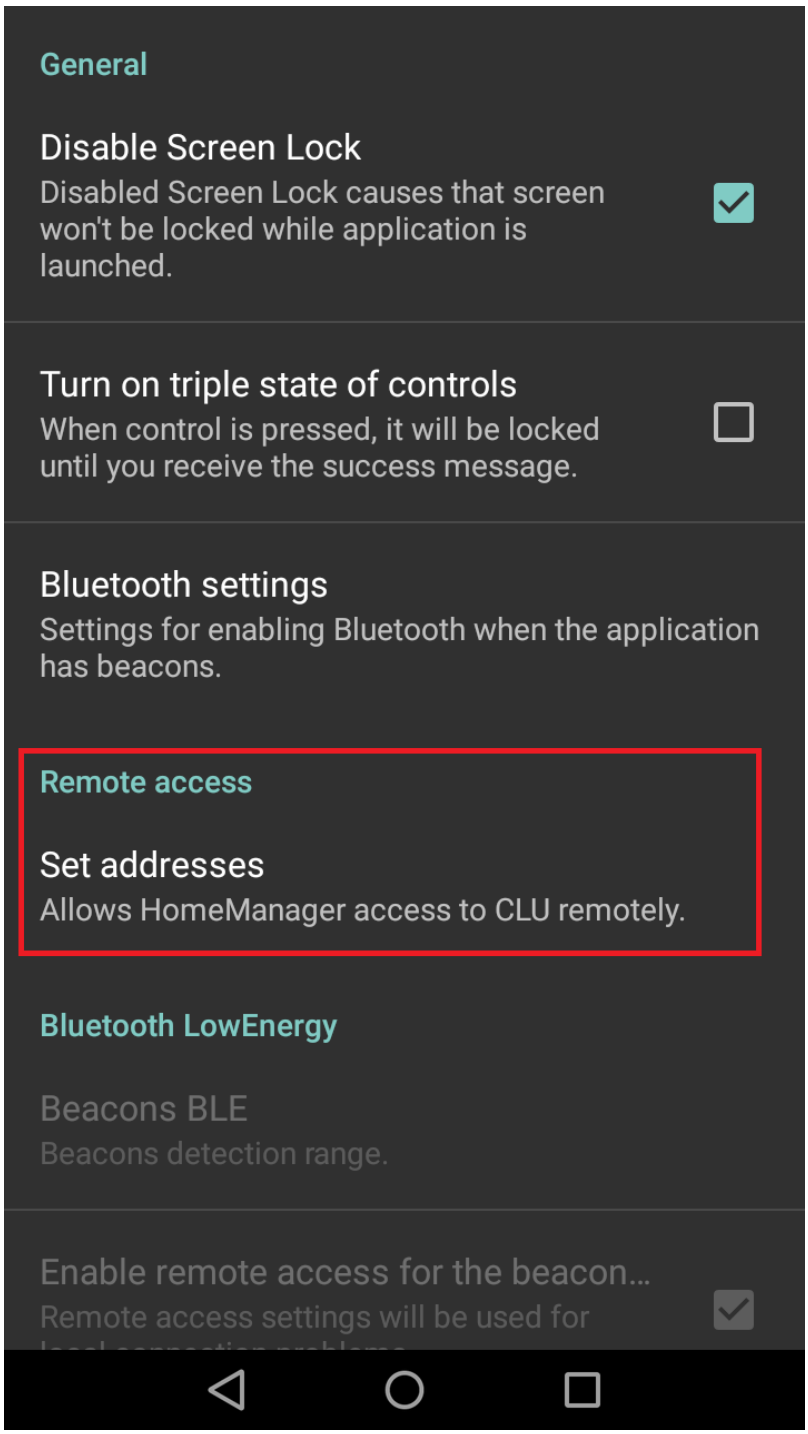
> **NOTE!** Make sure external communication is not blocked by internal router settings.

9.3. Configuration of the Home Manager mobile application

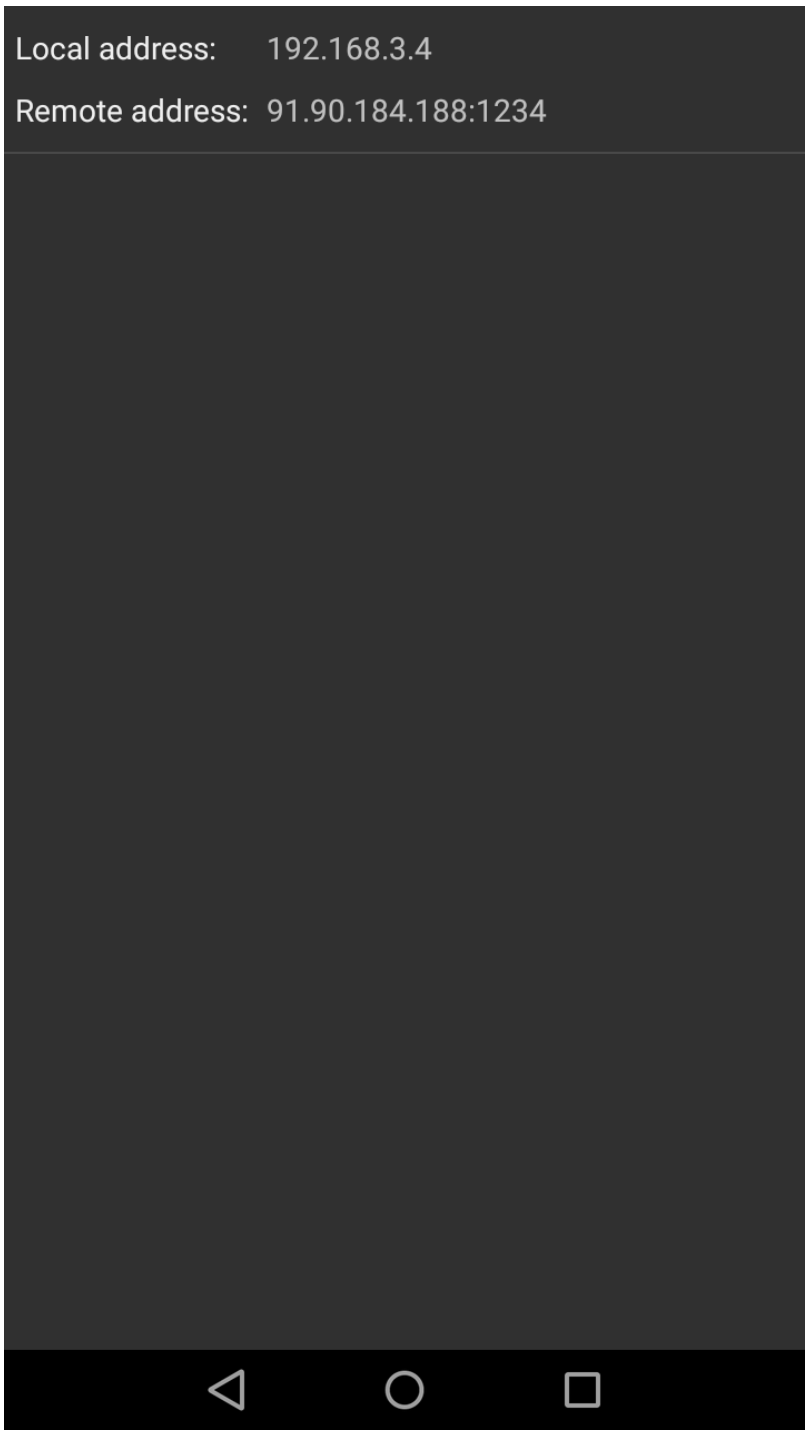
When creating a configuration, you must:

- 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):

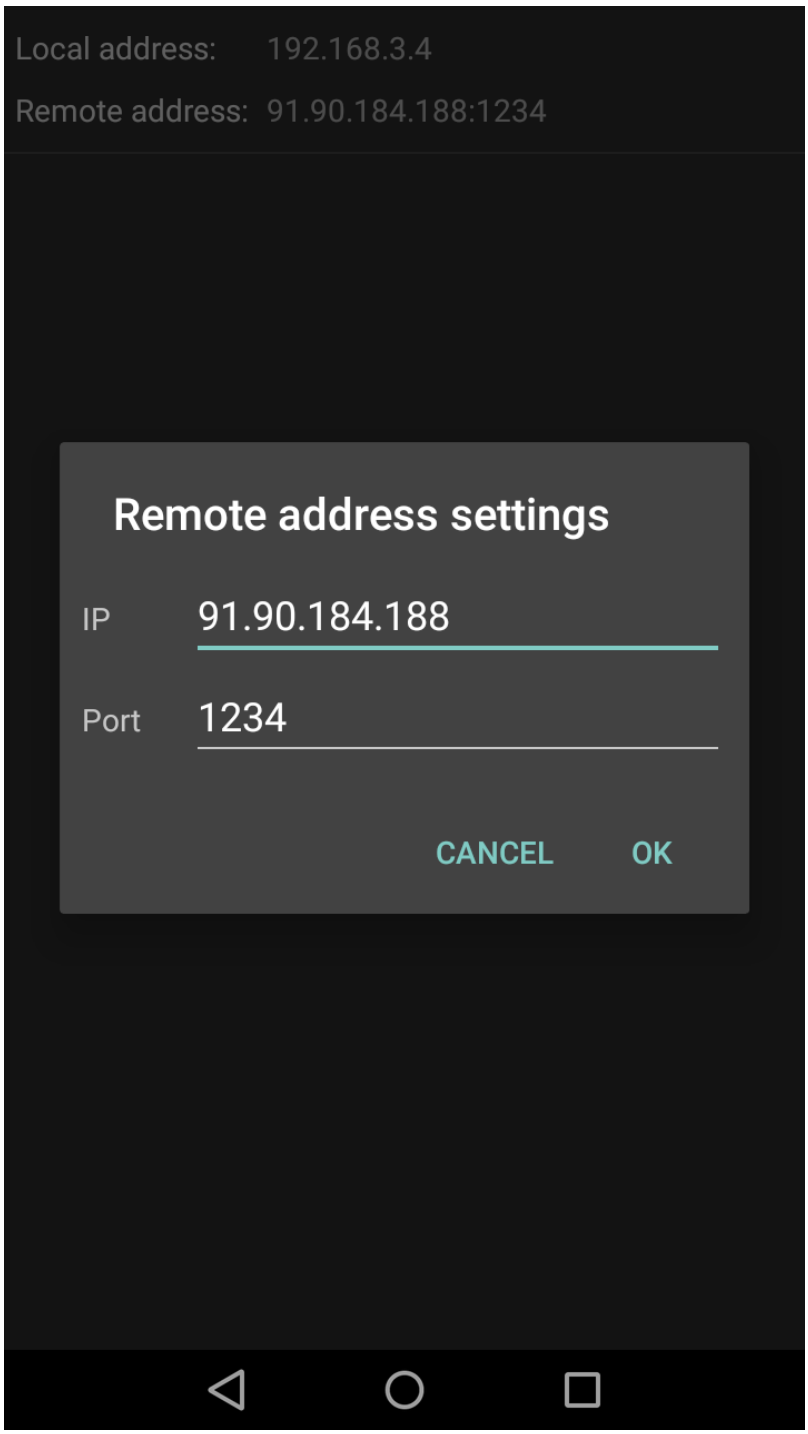
- in the settings, click *Remote access*, *Set addresses*:



- 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*.

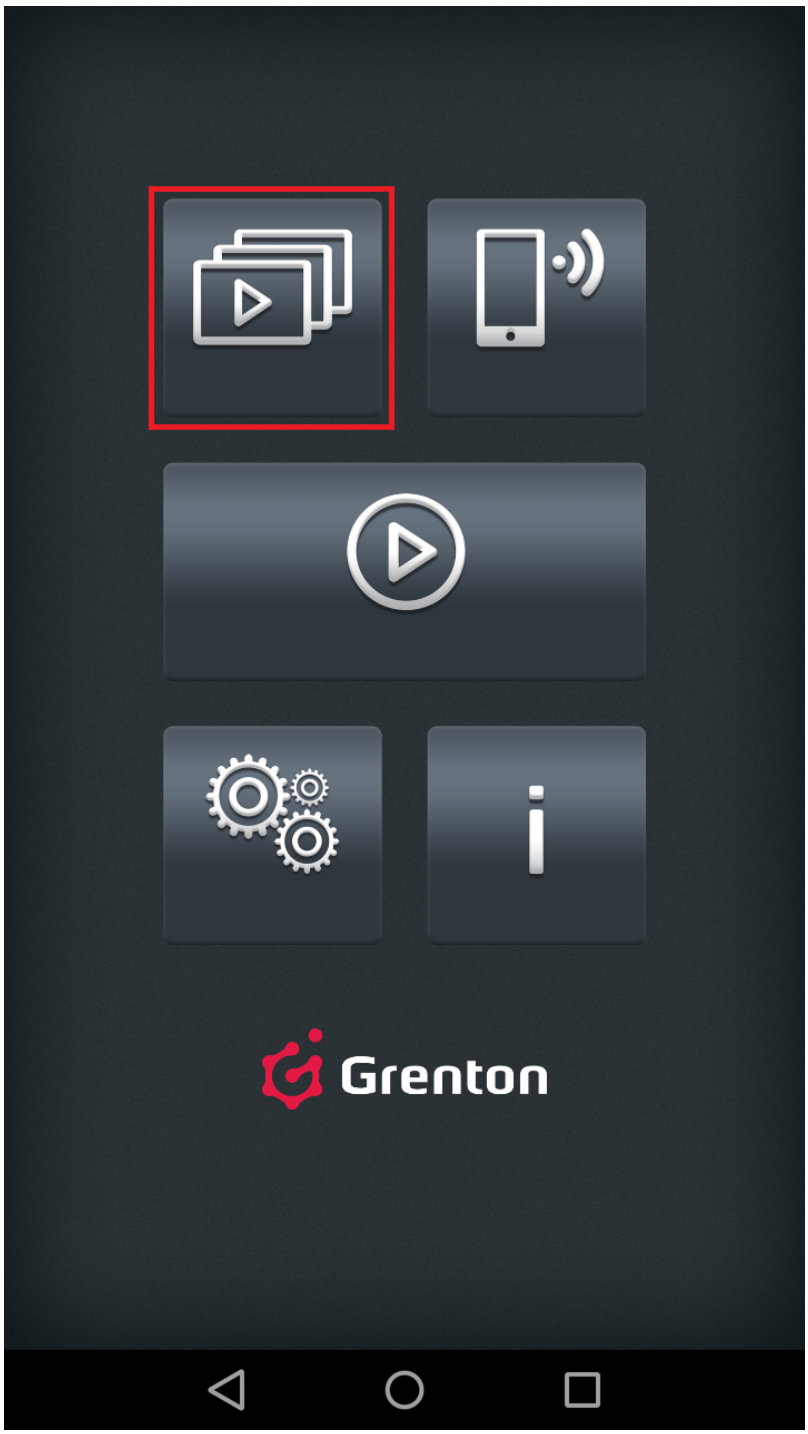


9.4. Starting remote access

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

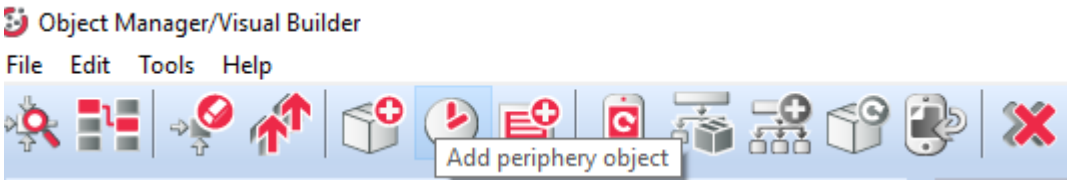
Timers are virtual objects created within specific CLU. Object Manager enables creation of maximum 64 timers. Timers can be used wherever there is need for method invocation after specific time or cyclical method invocation.

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

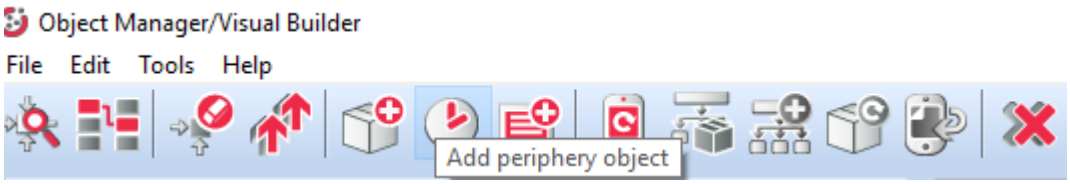
Name	Description
`OnTimer`	The event is called when the timer is counted
`OnStart`	The event is triggered when the timer starts
`OnStop`	Event triggered when the timer stops
`OnPause`	The event is called when the timer is paused

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.

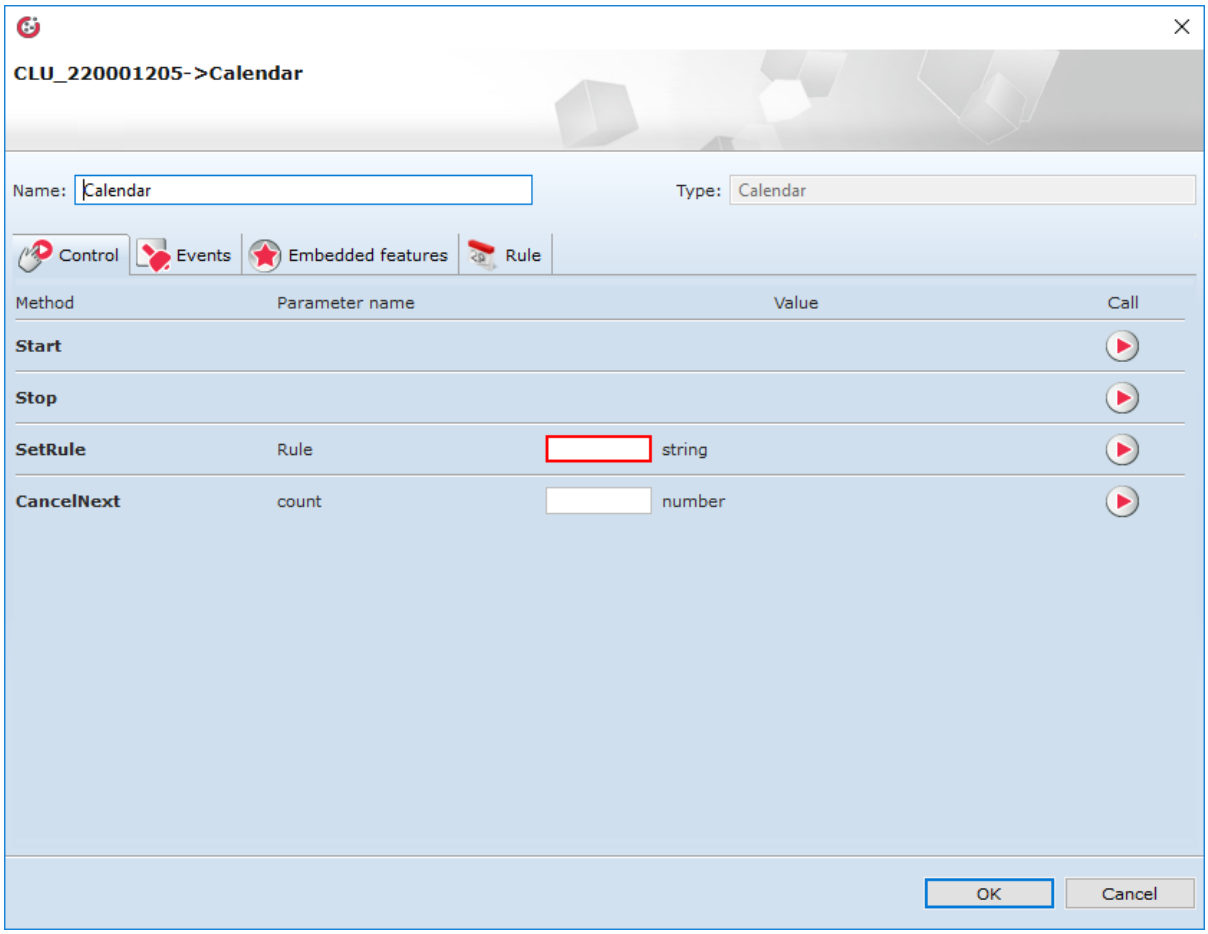


In the window that opens, select `Calendar`. After clicking `OK`, enter the name for the calendar you are creating. The Object Manager will display the properties window of the created object.

> **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

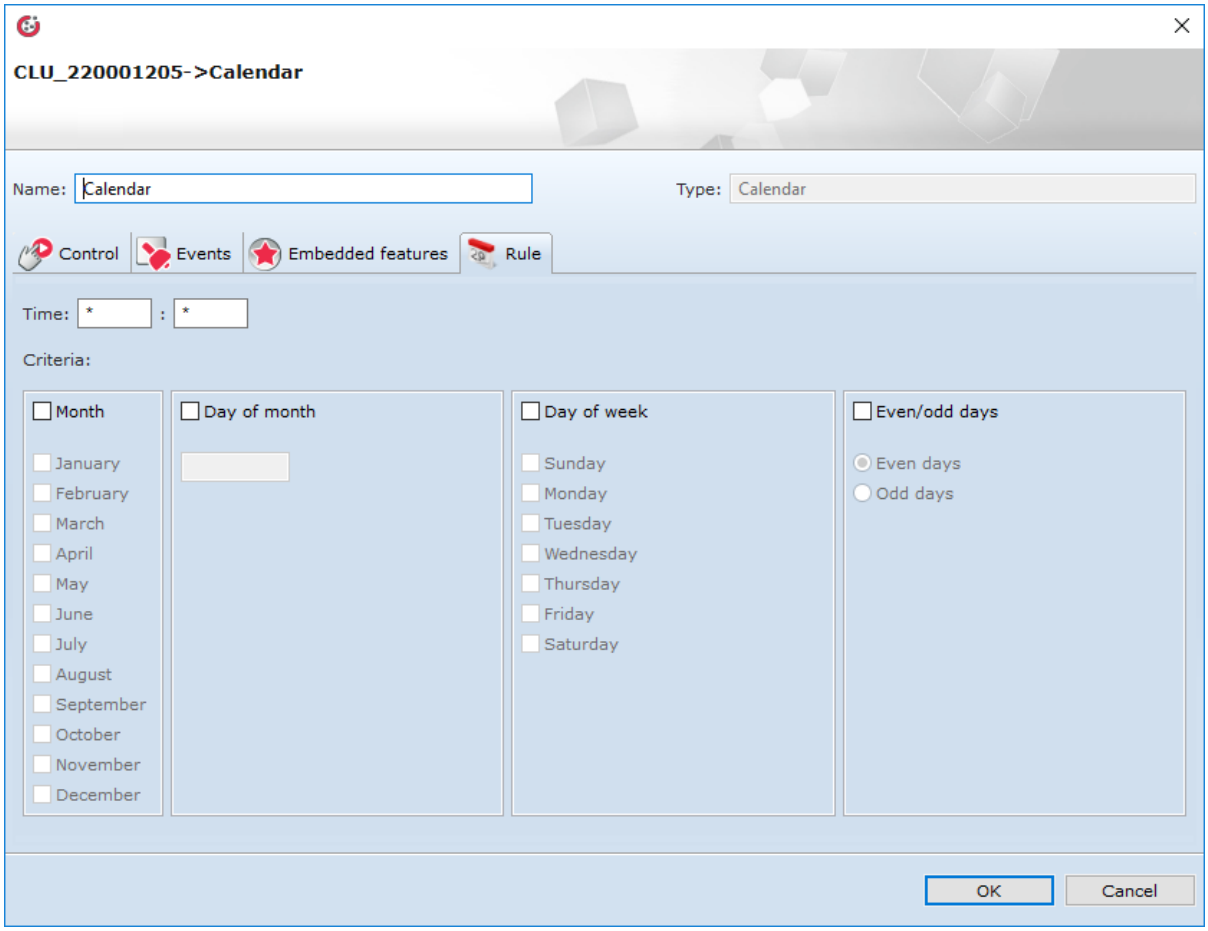
There are two ways of entering rules for the calendar:

• 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 easly 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

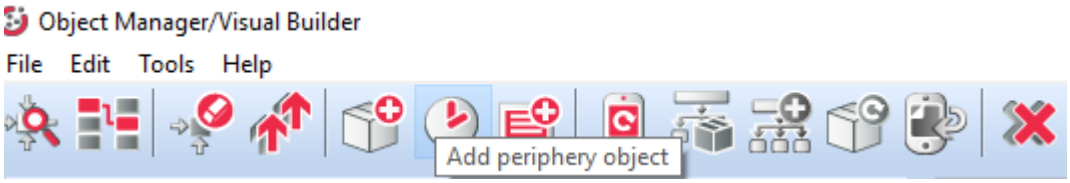
3. Schedule

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.



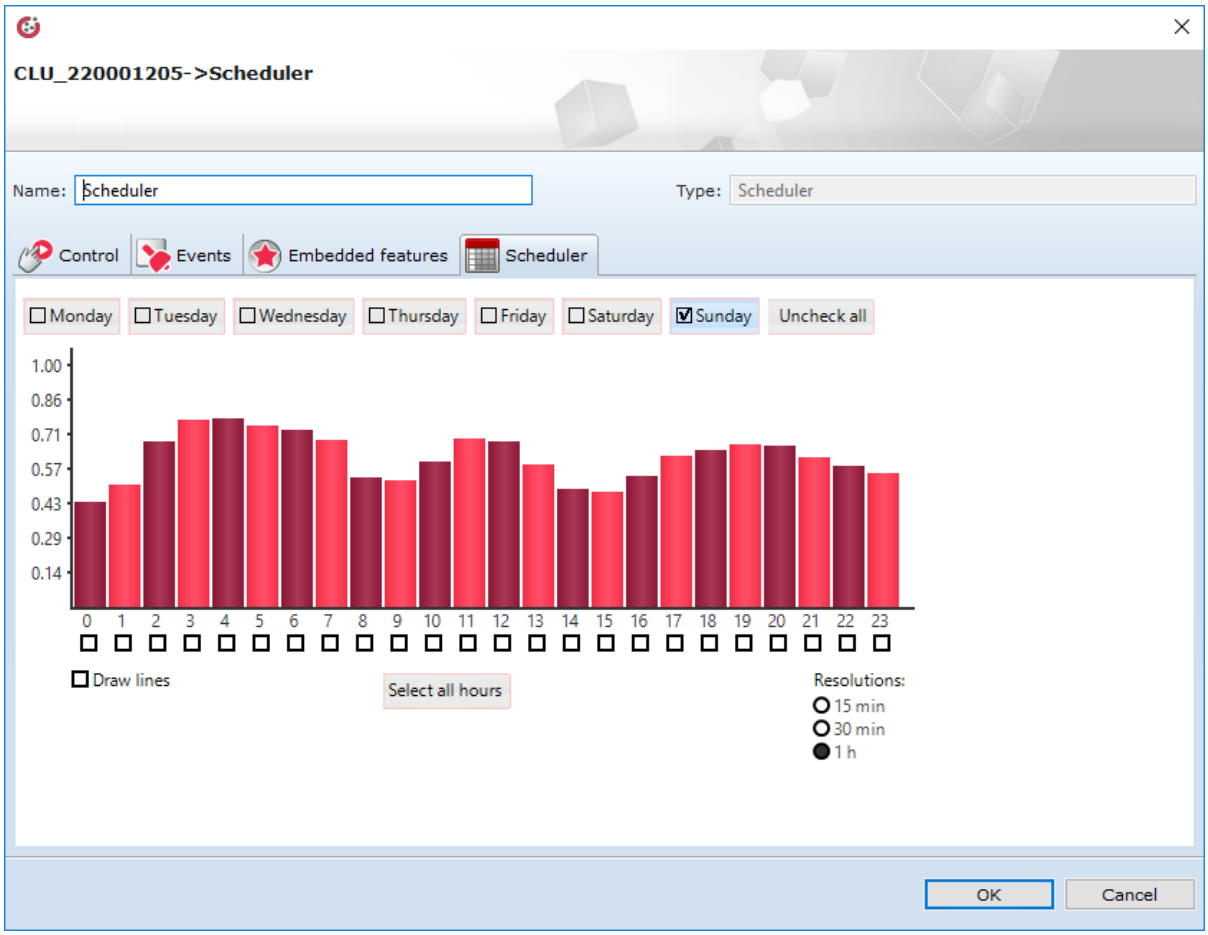
In the opened selection window, it is necessary to find and select the `Scheduler` object. After entering the name, the schedule properties window will open on the screen.

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.

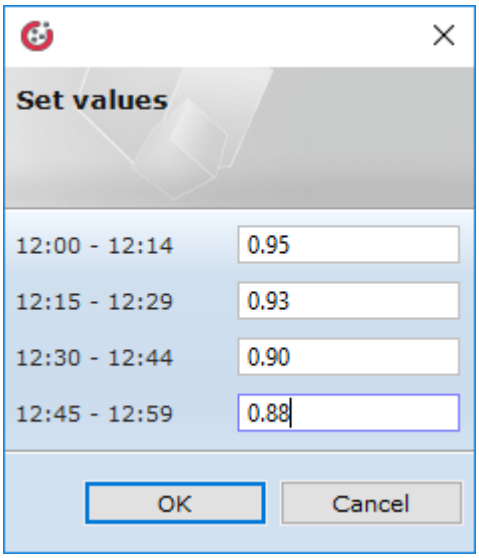
B. Setting values for the schedule

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.

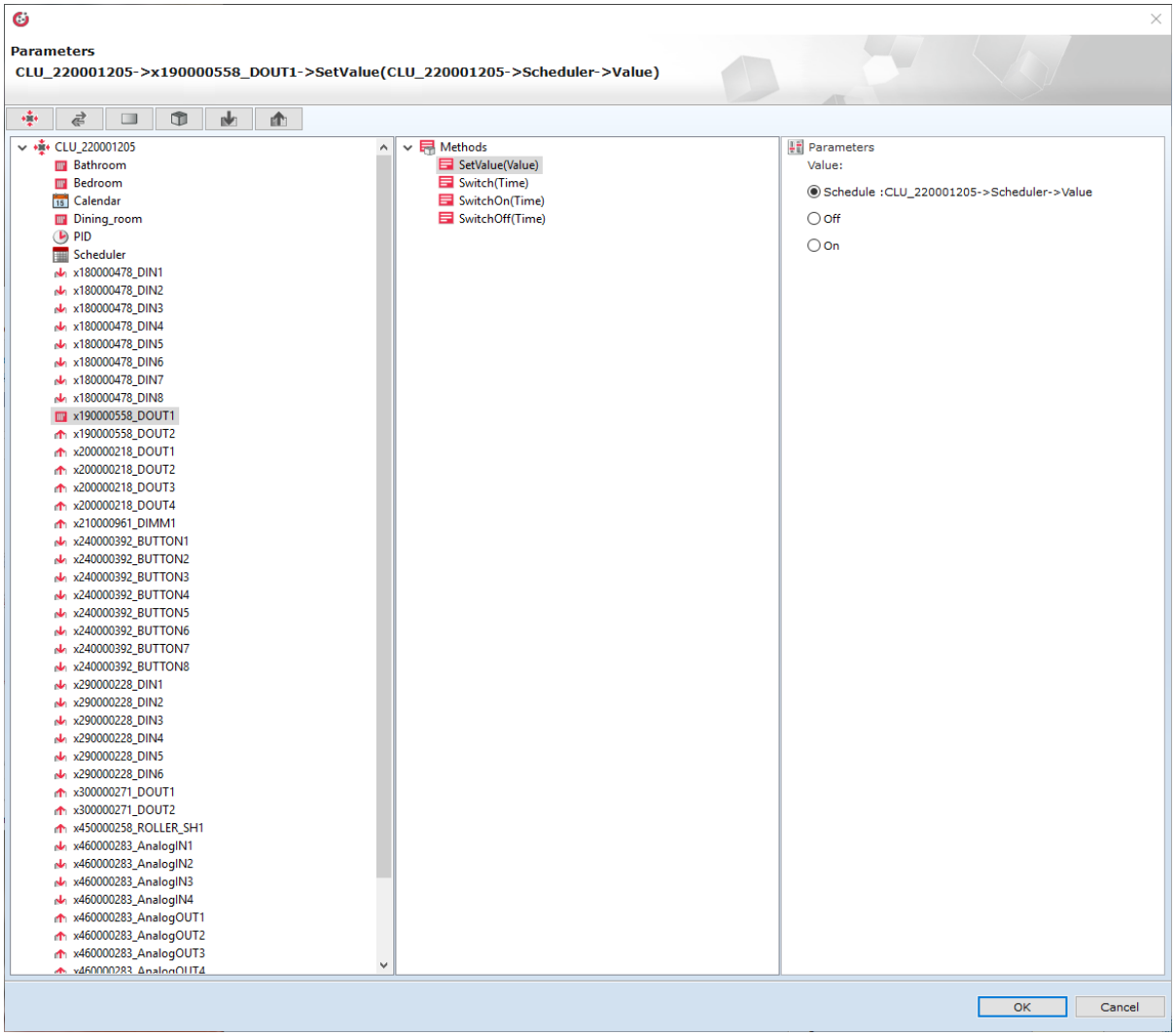
To simultaneously enter values for several days, mark days for which you will set values (by clicking marker), then begin entering the values. You can add values directly on the chart using mouse, or enter them in the values window which appears after clicking a specific hour.



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`	Harmonogram 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. PID controller

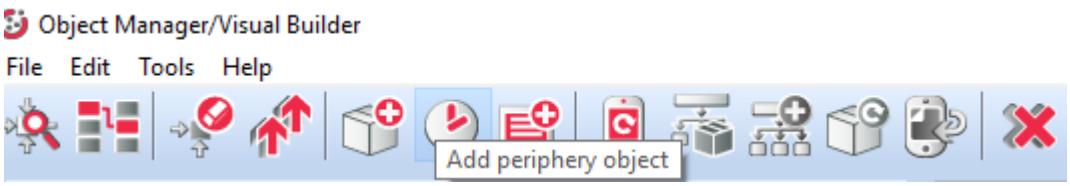
CLU enables creation of up to 64 PID controllers (proportional-integratingdifferentiating), used to maintain set output value on the constant level depending on the input value.

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

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

A. PID controller creation

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



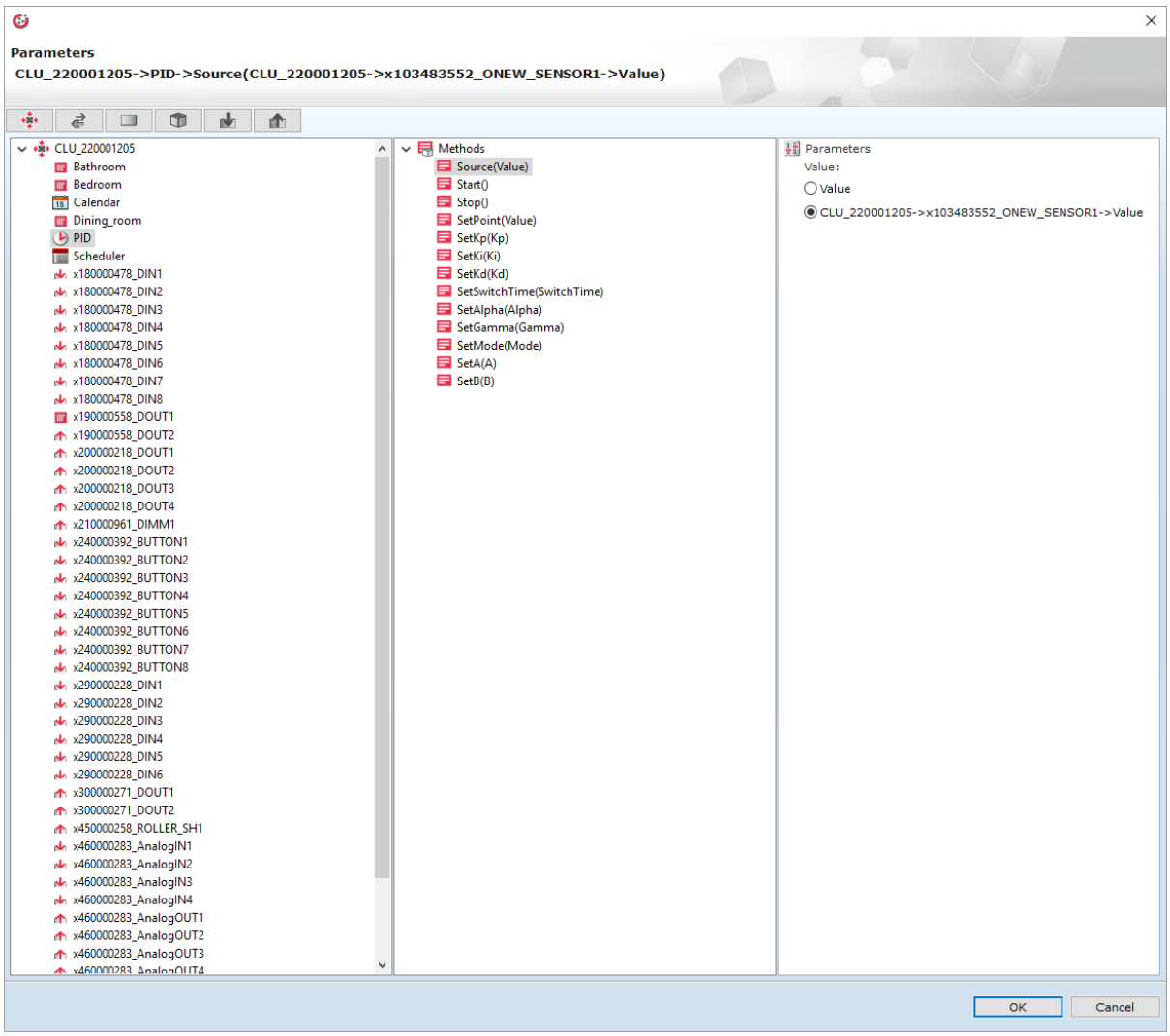
In the open window find and select `PIDcontroler`, then name it. Windows of features of the newly created controller will appear on the screen. It contains three tabs

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

B. Control using the controller

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

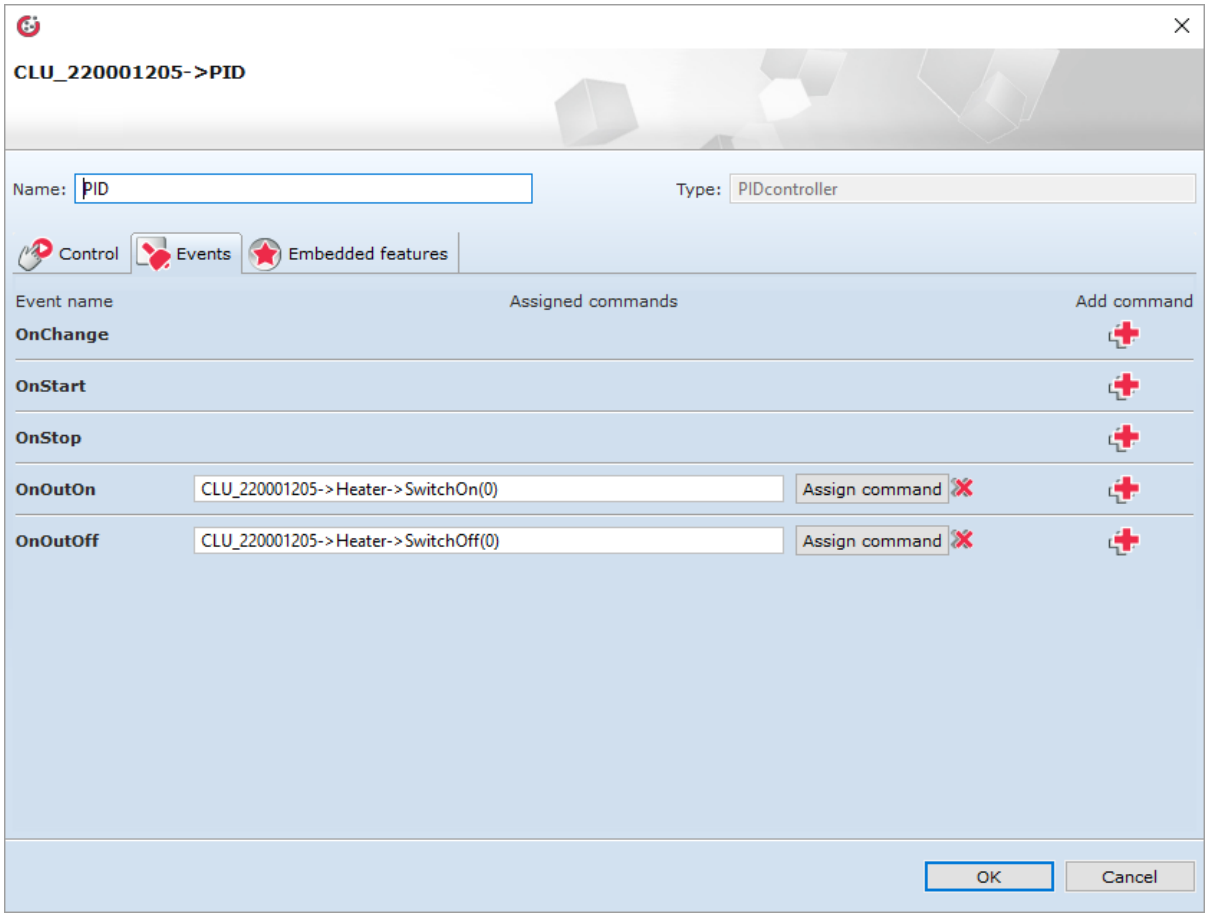
- Add the source value to the Source method, eg the temperature sensor's 'Value' feature (in the temperature sensor to the `OnChange` event, select the PID controller, and assign the value from the temperature sensor to the `Source` method as a parameter).



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

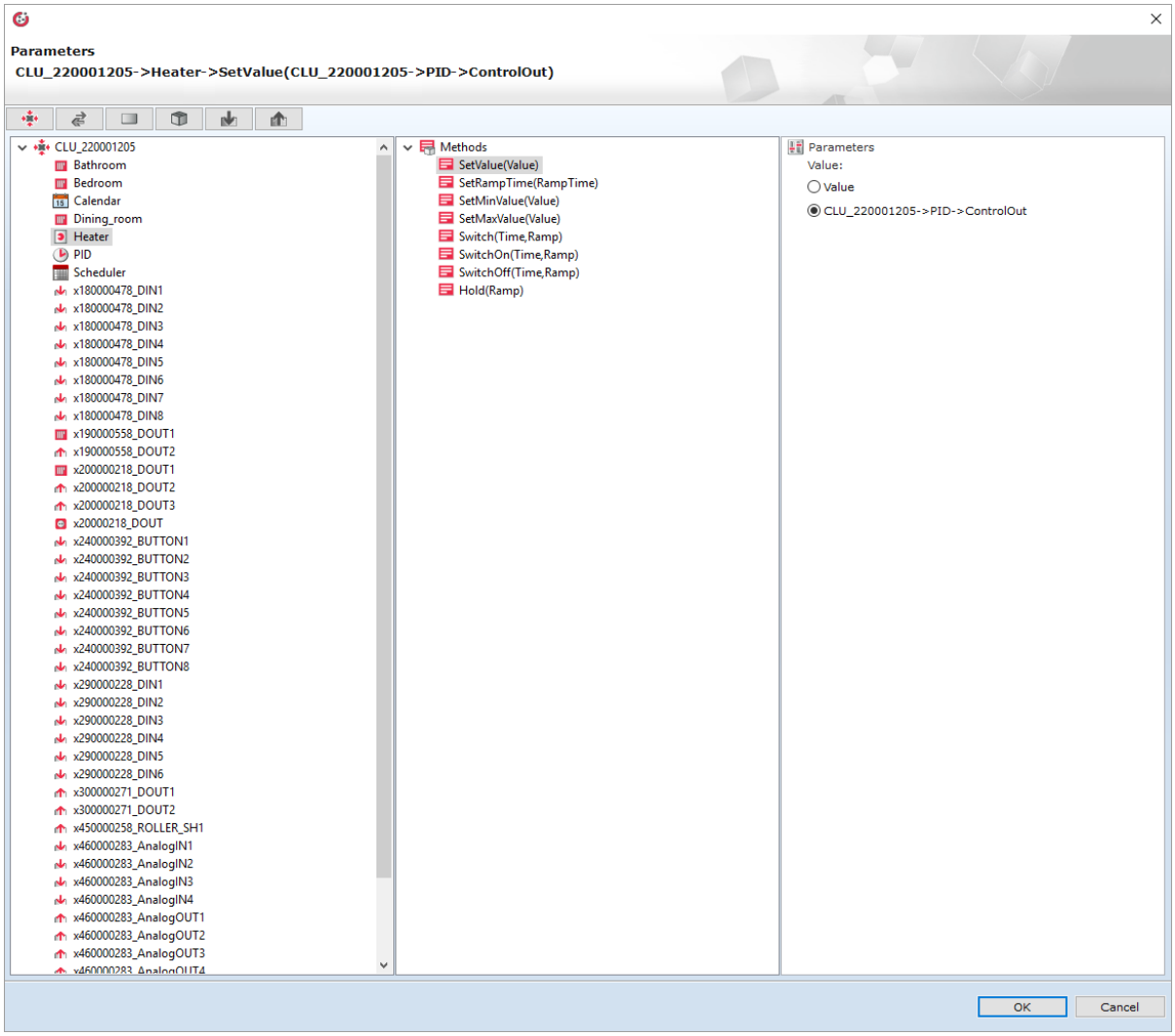
  To do this, assign control methods to the object to the `OnOutOn` and `OnOutOff` events.

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



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

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



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

C. Work modes

The controller has two possible work modes:

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

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

- `Normal` – for manual mode;
- `Auto` – for automatic mode.

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

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

D. PID Controller operational design

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

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

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

E. PID Controller configuration parameters

FEATURES

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

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

METHODS

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

EVENTS

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

5. Thermostat

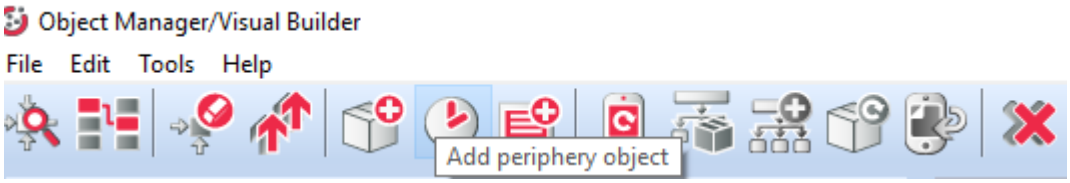
A thermostat is a virtual object that is used to create a heating or cooling control configuration depending on the given temperature sensor and the heating or cooling schedule introduced in the weekly schedule. Temperature values are set using the graphical interface for each day and hour with a 15-minute,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

You can create up to 64 thermostats in one CLU.

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

A. Thermostat creation

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



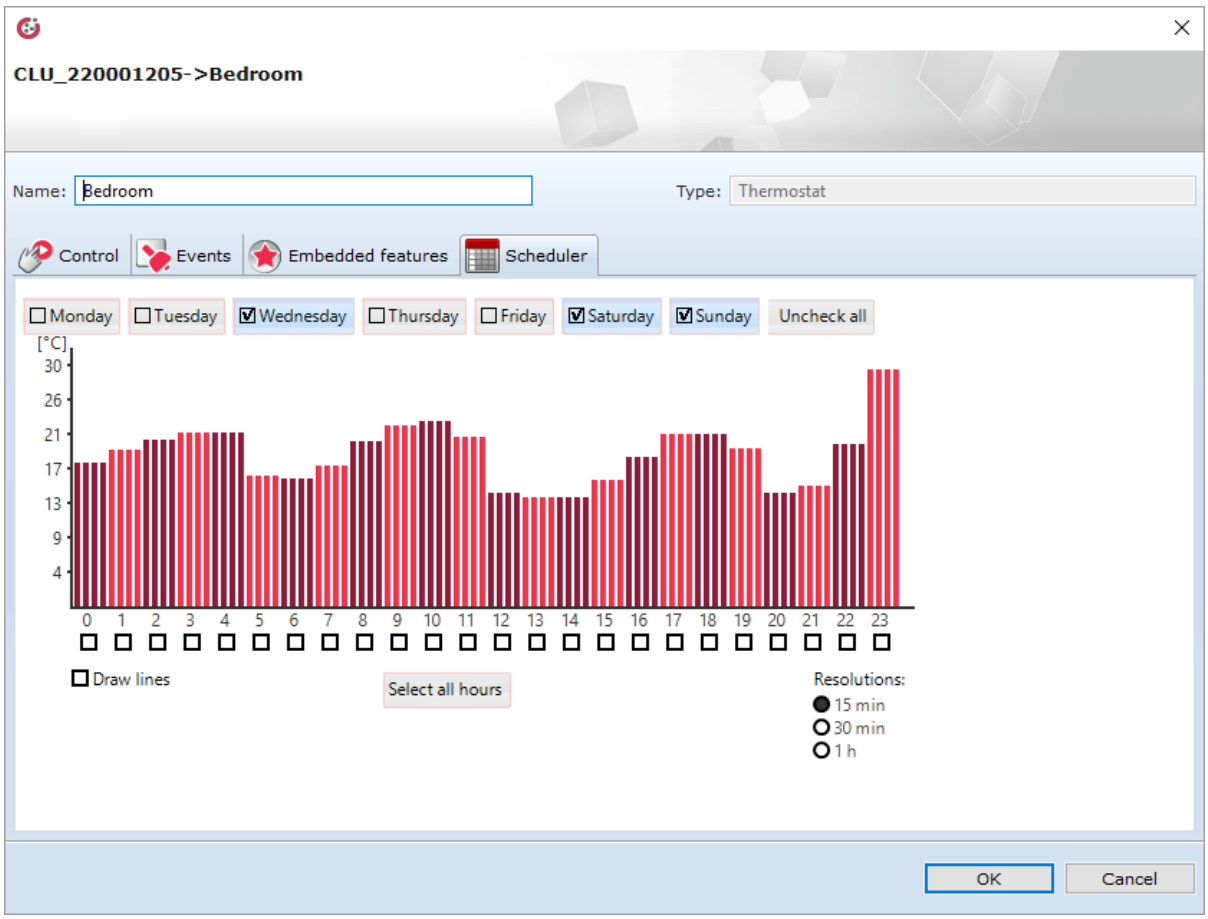
In the opened selection window, search for and select `Thermostat`. After entering the name, entering the source (which should be the temperature sensor responsible for a given heating zone) and selecting the receiver (which is the output to which the device responsible for a given heating zone is connected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

In this window there are four tabs:

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

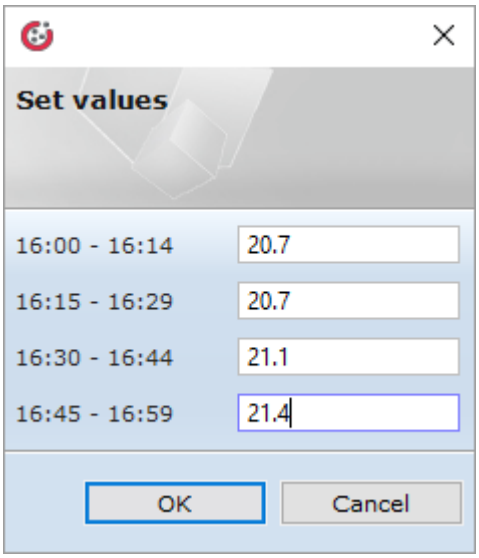
B. Formulating values for a thermostat

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



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

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



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

C. Configuration parameters of the Thermostat object

FEATURES

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

METHODS

Nazwa	Opis
`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` - autodetection, `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

Nazwa	Opis
`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

X. Media measurement

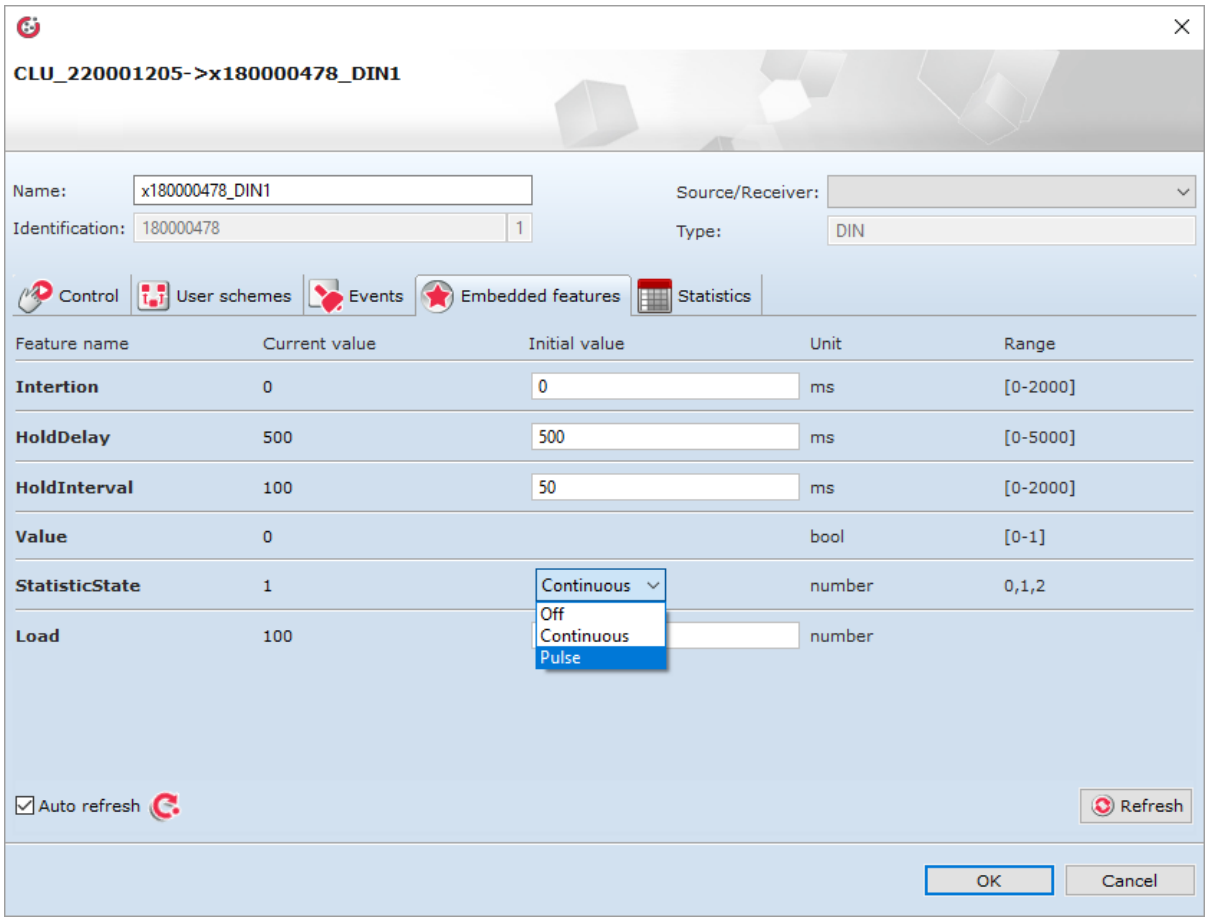
1. Launching media measurement on the Object Manager page

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

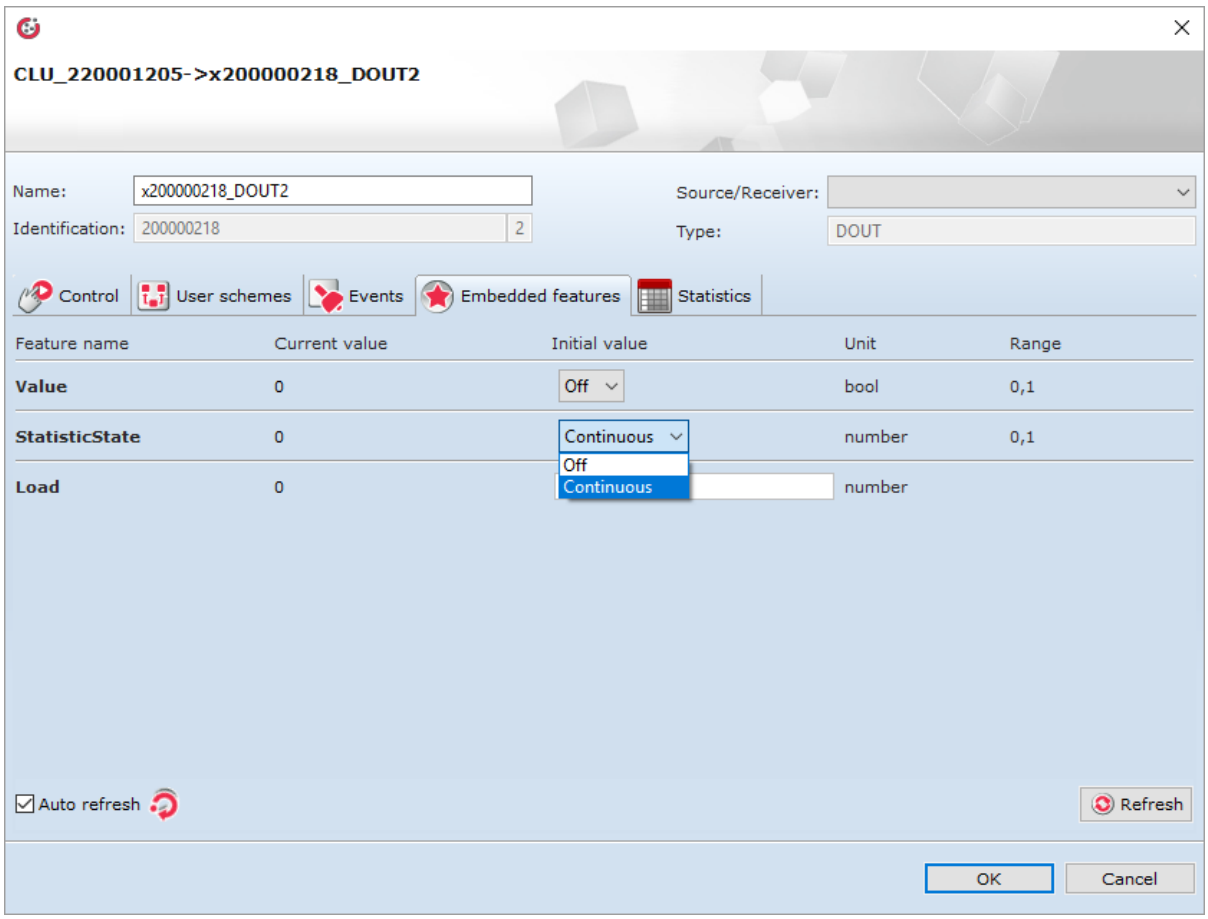
> **NOTE!** Media measurement is available for Object Manager version 1.2.0.180202 and higher, and for CLU with firmware 04.07.29-1802 and higher.

Media measurement can be run for modules:

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



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

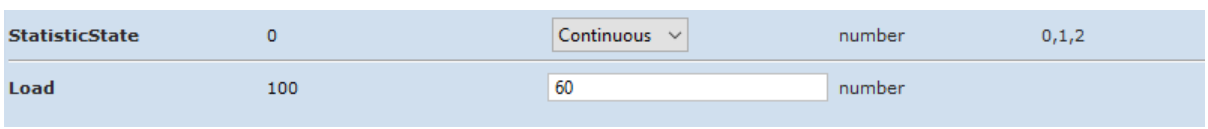


> **NOTE!** The media measurement of the above-mentioned modules applies to modules for DIN rail and flush-mounted Tf-bus! The measurement setting is not available for Z-Wave modules!

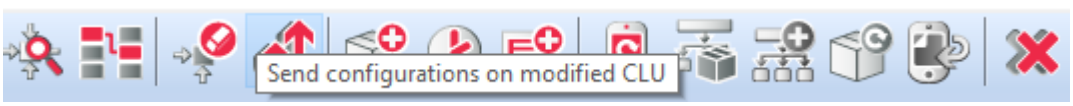
A. Creating a configuration

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):



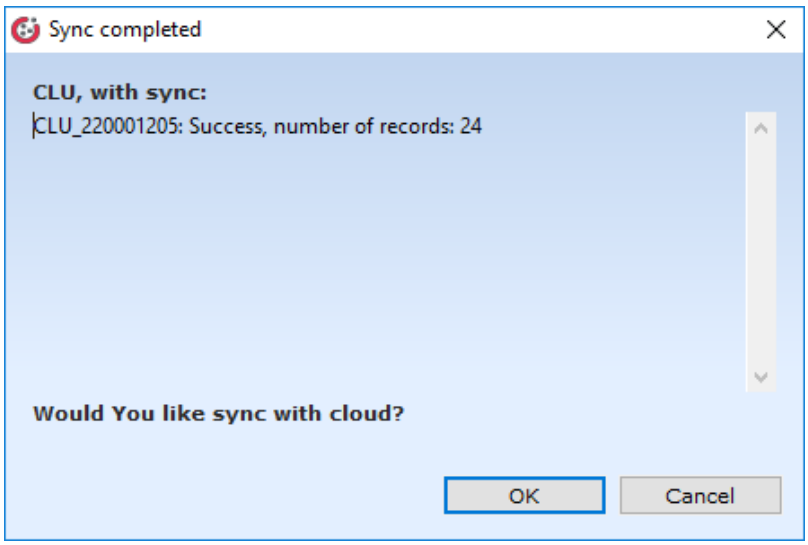
- Confirm with *OK*;
- Add media measurement settings for subsequent modules - repeat steps 2b-2e;
- Send the configuration to the CLU.



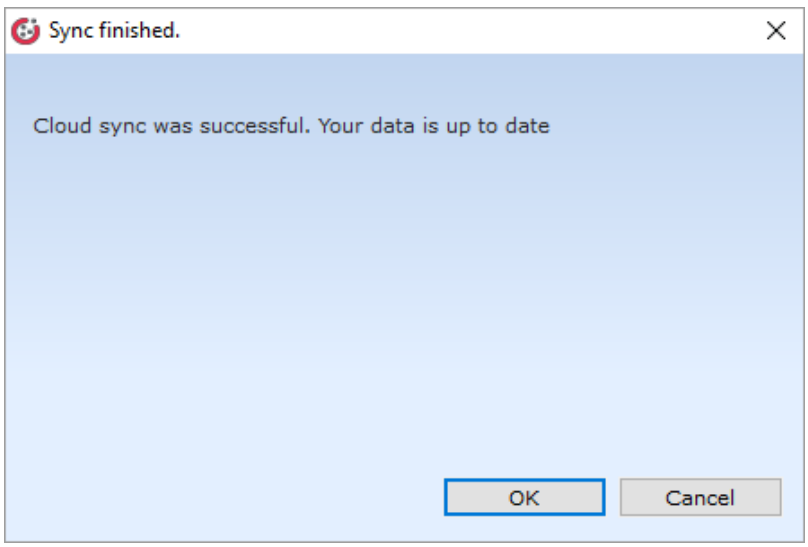
B. Read media measurement in the Object Manager

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:



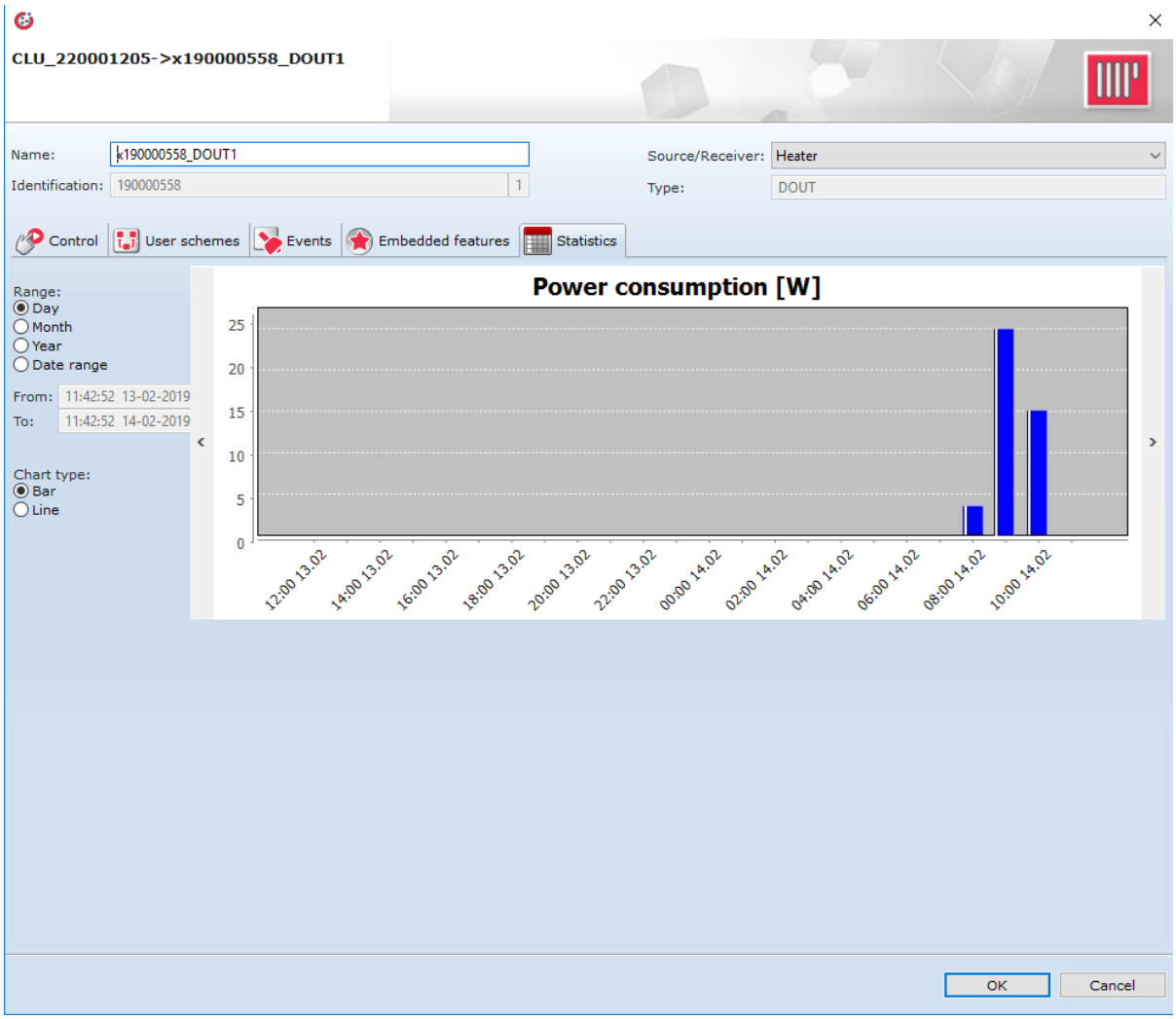
- Click *OK*;
- The Object Manager will then synchronize the downloaded data with the cloud;
- After the synchronization is completed, press *OK*:



> **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.

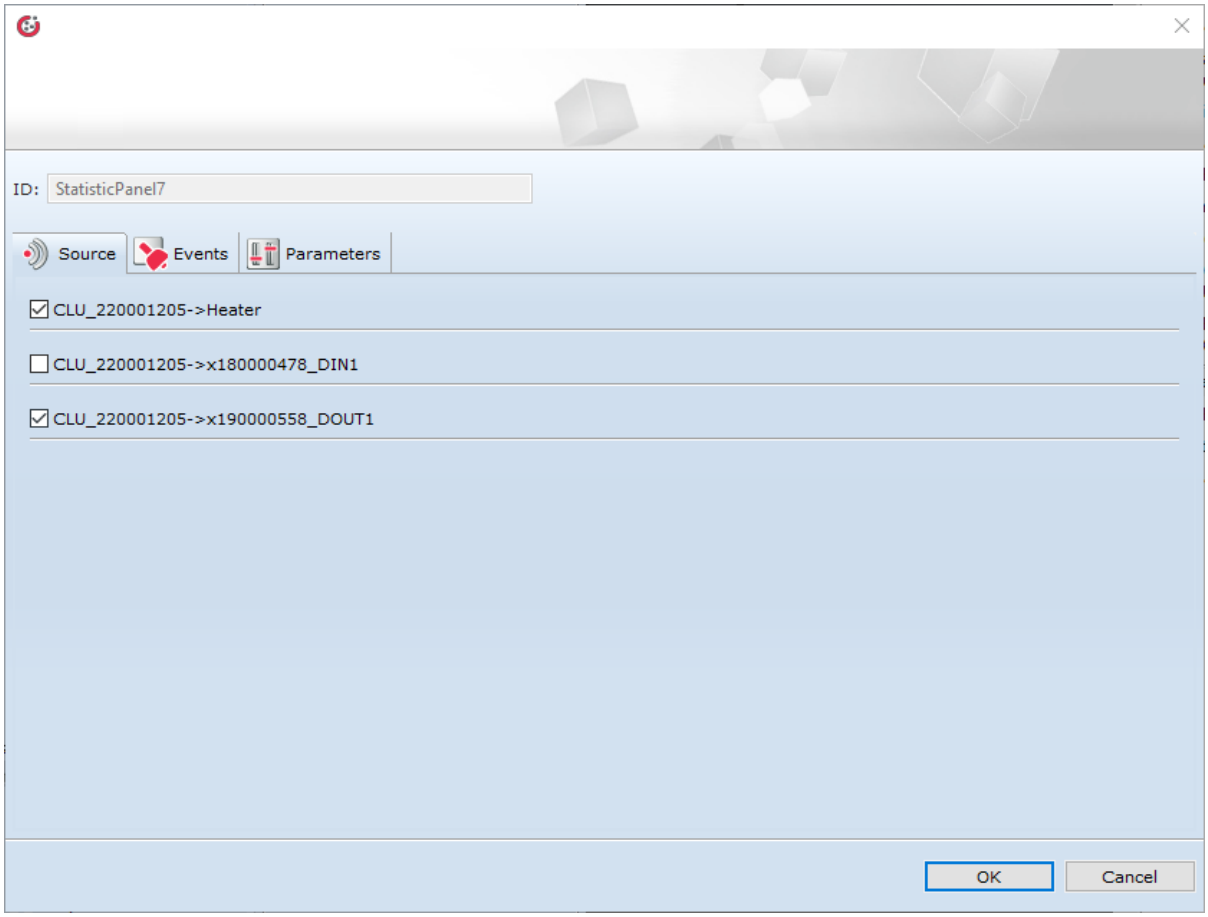


C. Configure the media measurement for the application interface

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..

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.

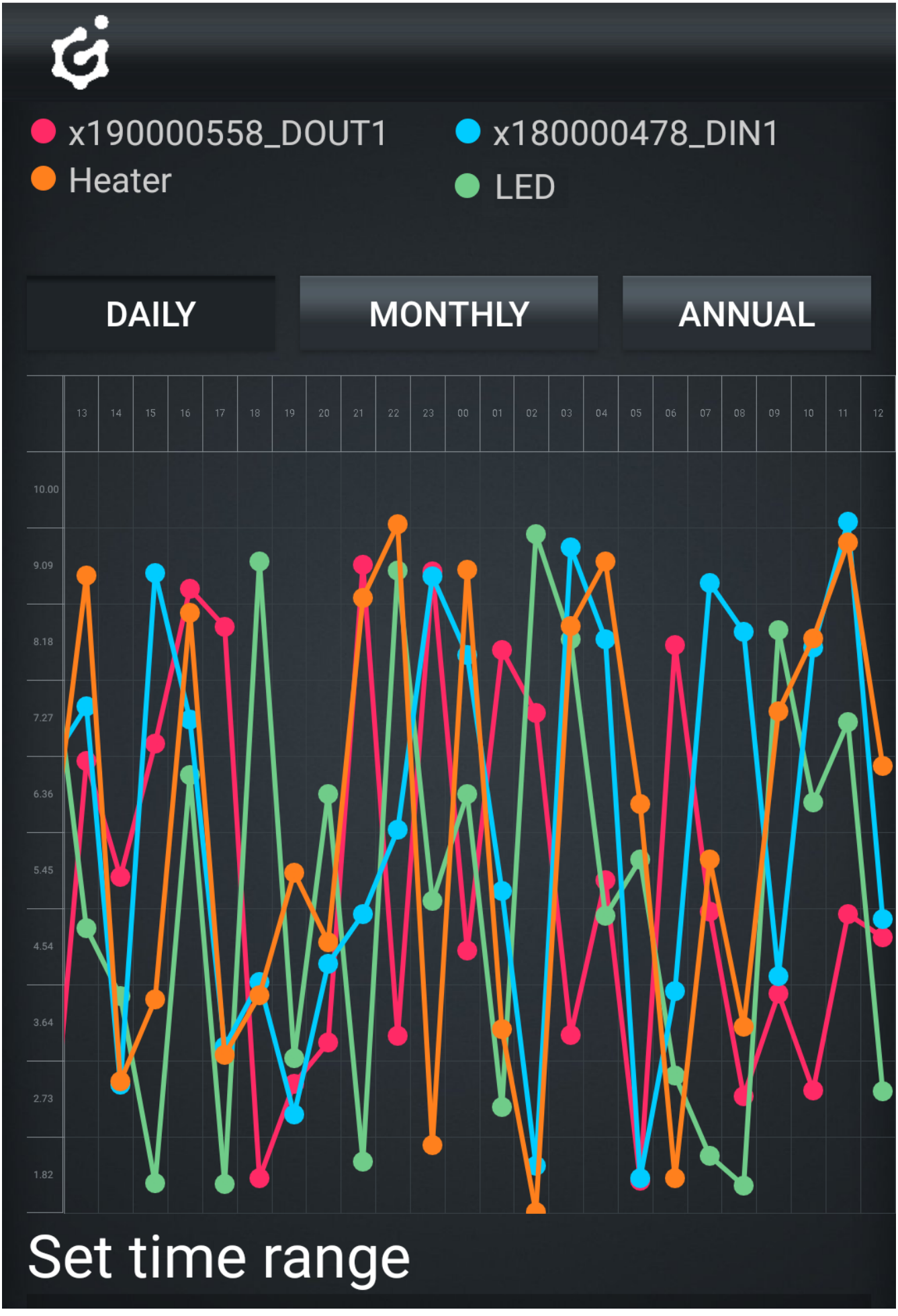
A. Taking measurements:

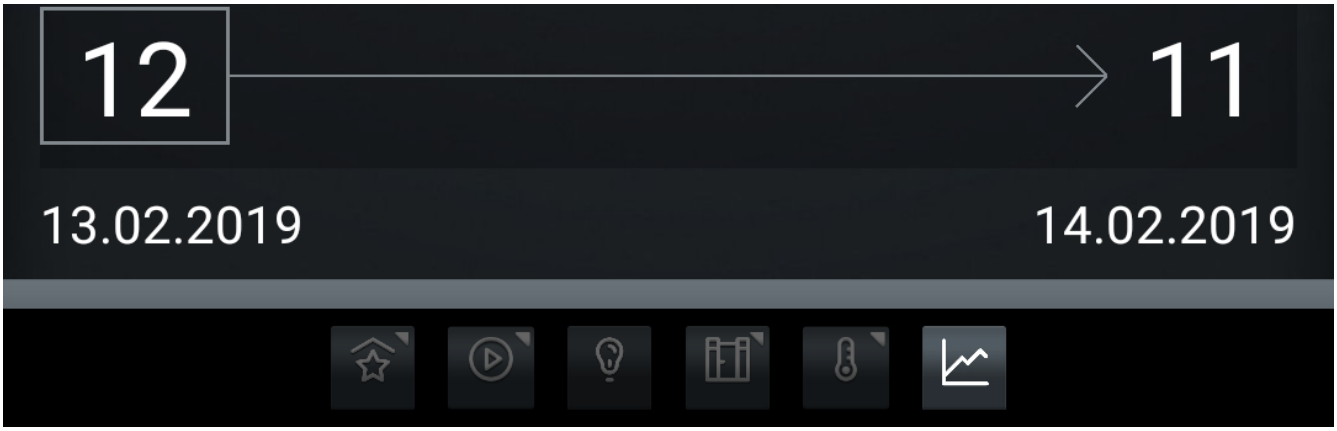
- Enter the application settings from the main menu (gear icon).
- Select from the list of settings: *Download measurements from the CLU*.

- 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.

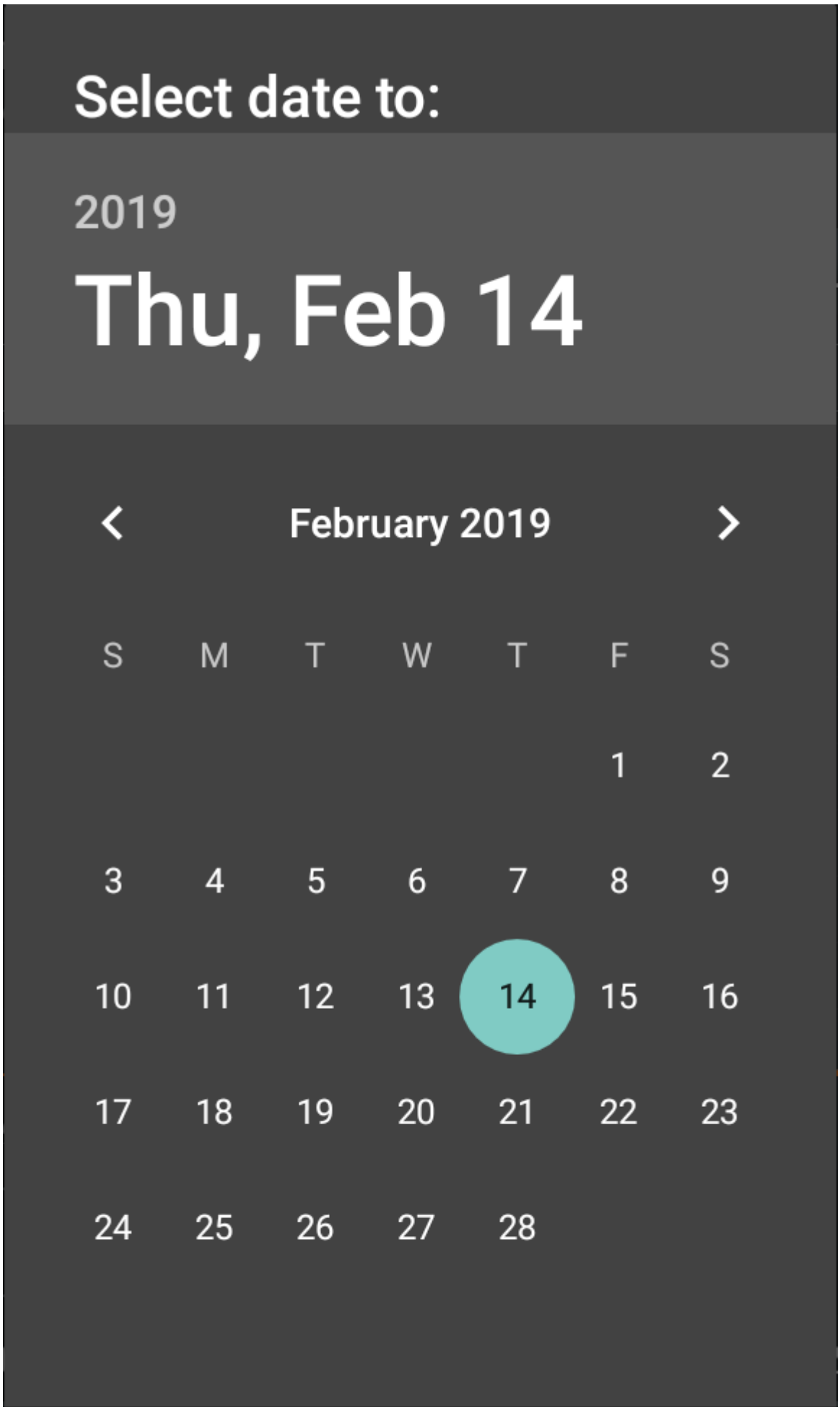
B. Media panel view options:

- 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*:


x190000558_DOUT1
x180000478_DIN1
Heater
LED
DAILY
MONTHLY
ANNUAL
13 14 15 16 17 18 19 20 21 22 23 00 01 02 03 04 05 06 07 08 09 10 11 12
10.00
9.09
8.18
7.27
6.36
5.45
4.54
3.64
2.73
1.82
Set time range



- 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:


Select date to:
2019
Thu, Feb 14
February 2019
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28

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*.

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.

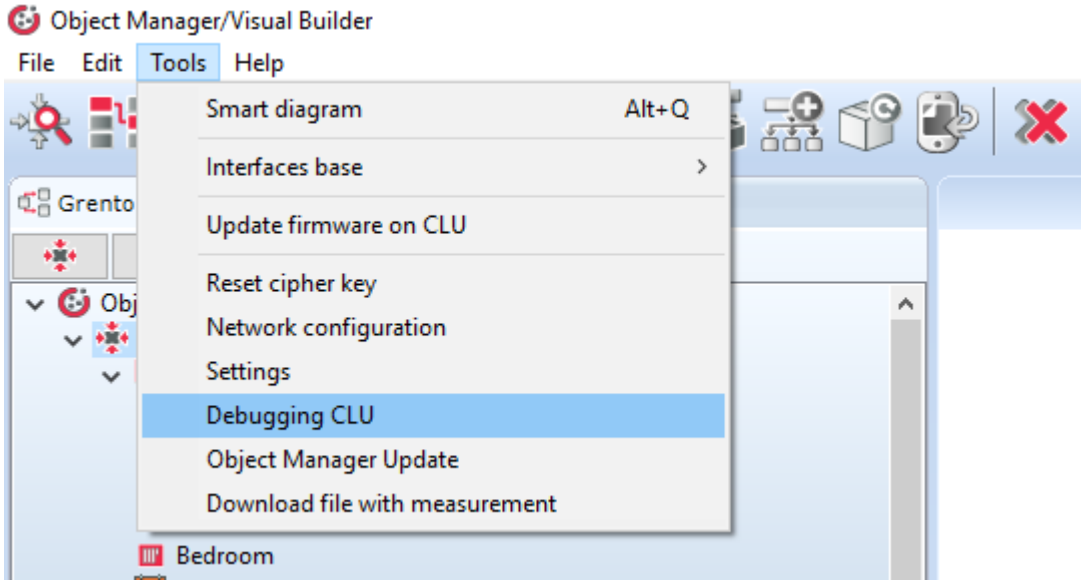
> **NOTE!** If Z-Wave modules were added to the CLU before starting the *Hard Reset* function, after performing the reset it will be necessary to perform the procedure of deleting and re-adding each Z-Wave module!

2. System self-diagnosis - *Debbuging CLU*

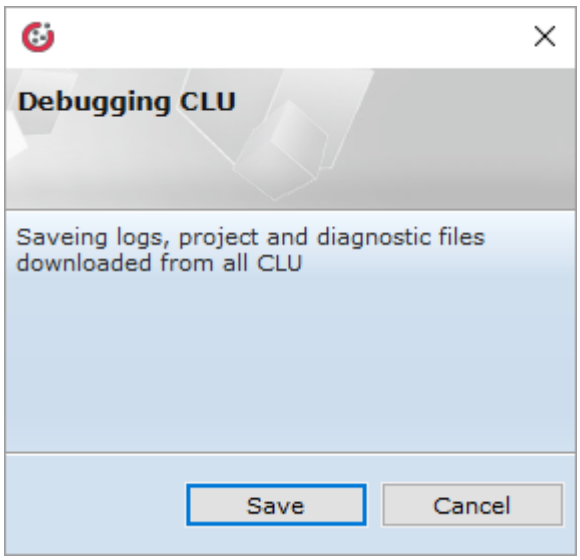
- *Debugging CLU* is used to diagnose the CLU central unit and to quickly find any problems in the created project.

In order to carry out the self-diagnosis of the system:

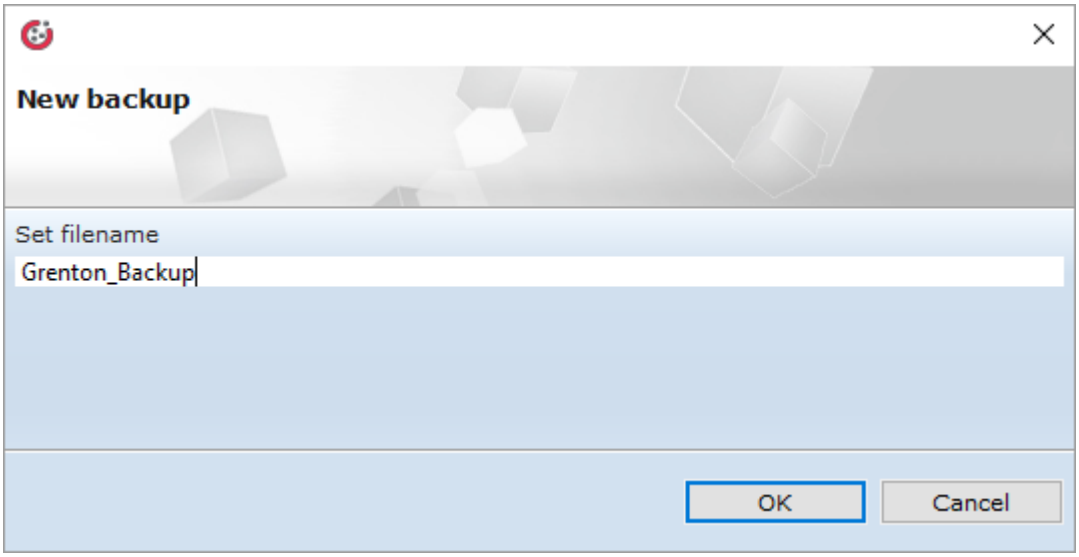
- Open the project in the Object Manager;
- Choose *Tools* from the taskbar and *Debugging CLU* from the expanded list:



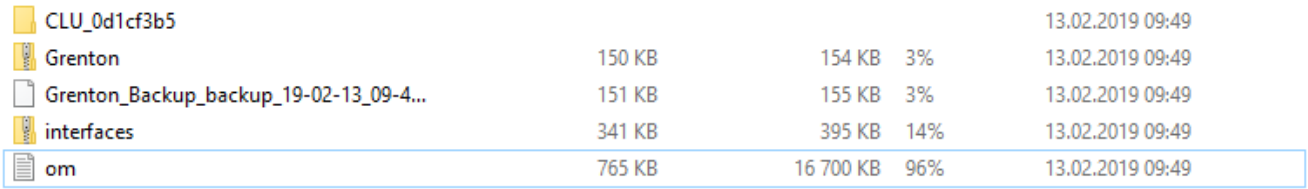
- In the window that opens, select *Save*:



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



- Then a folder in *.zip* format will appear in the selected location, the contents of which will be as follows:



- The folder created in this way contains:
  - CLU configuration files;
  - the current interface database used in the project;
  - file with specified application logs;
  - information about the project and its backup.

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.

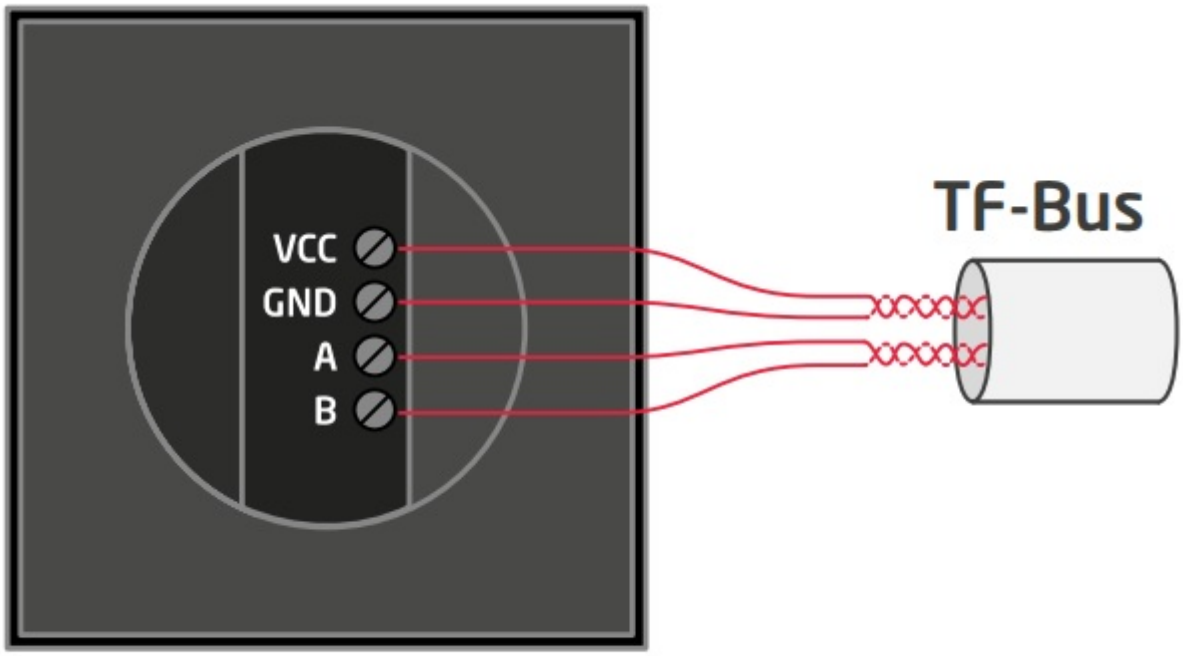
2. Connection of the Smart Panel to the CLU

> **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.

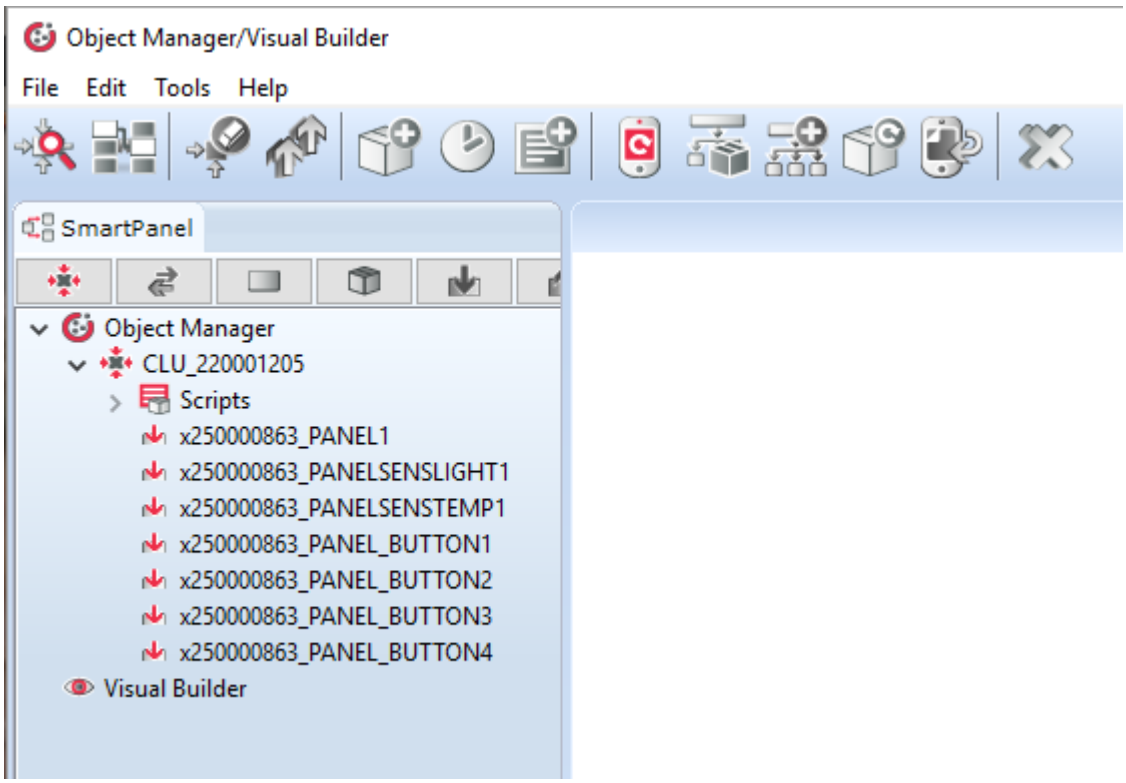
> **NOTE**! Smart Panel version v4 is available for Object Manager in version 1.2.1.190201 and higher, and for CLU with firmware 04.07.49-1912 and higher.

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:

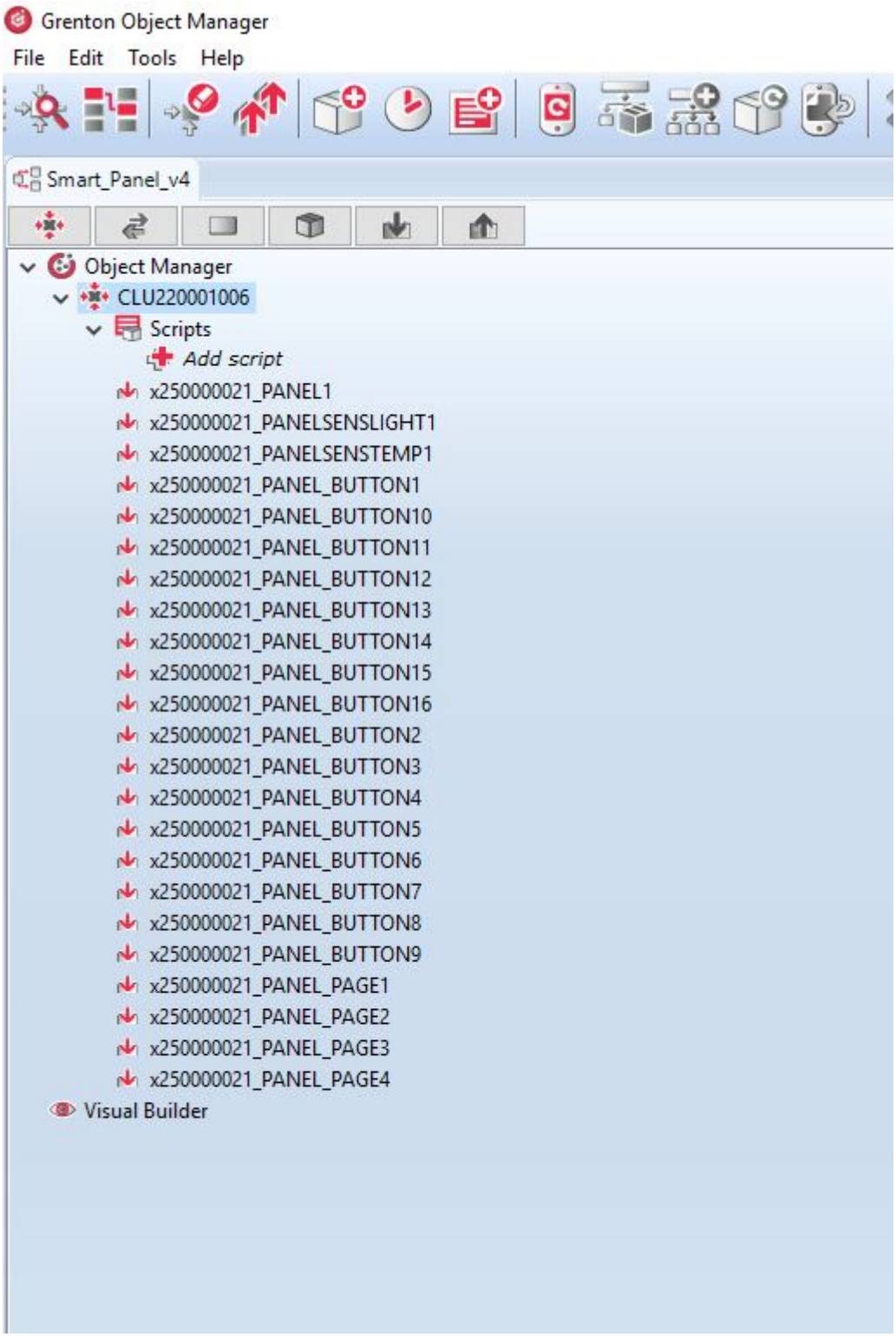
- Connect one lead from the first twisted pair (eg UTP cable) to the Vcc terminal;
- Connect the other wire from the pair to the GND terminal;
- Connect one cable from the other pair to terminals A and B.



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%)

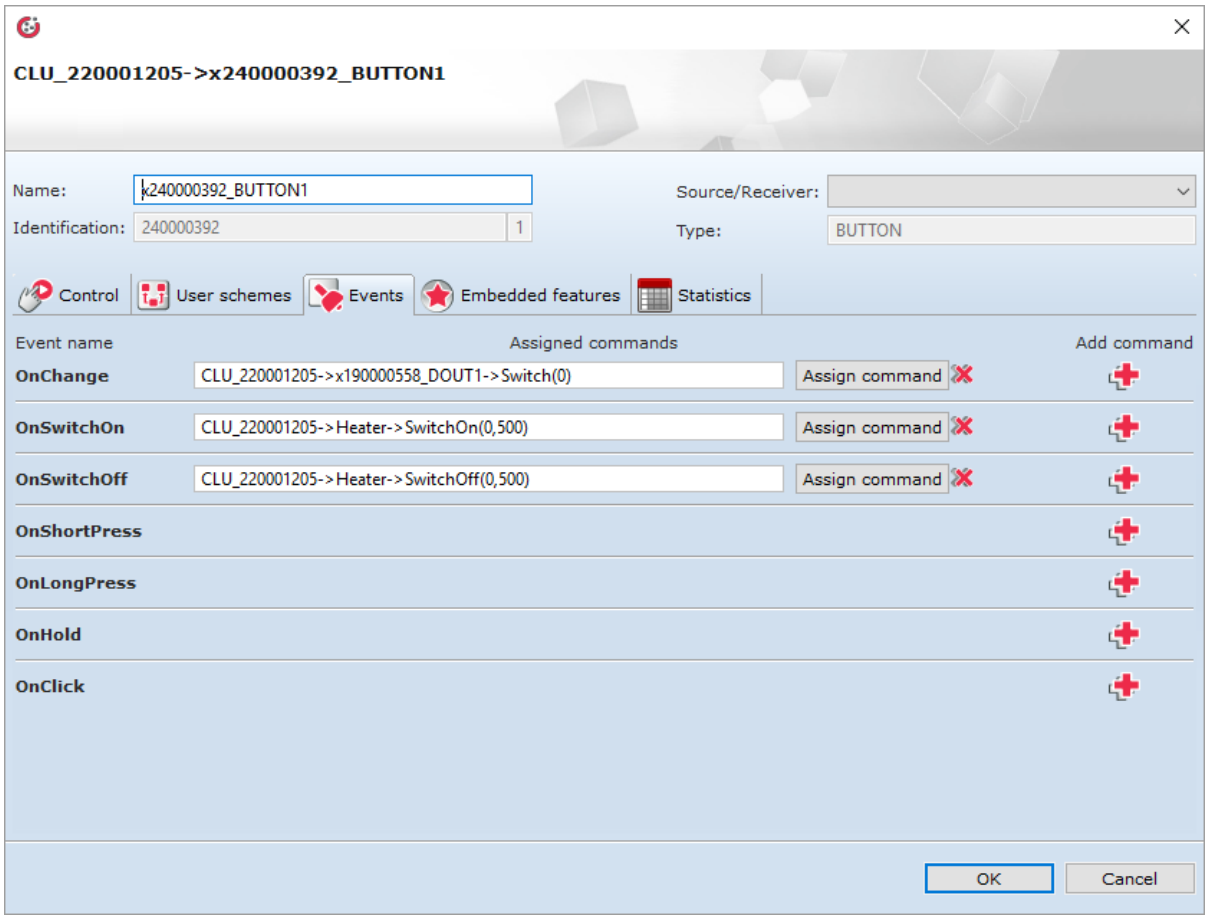
EVENTS

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

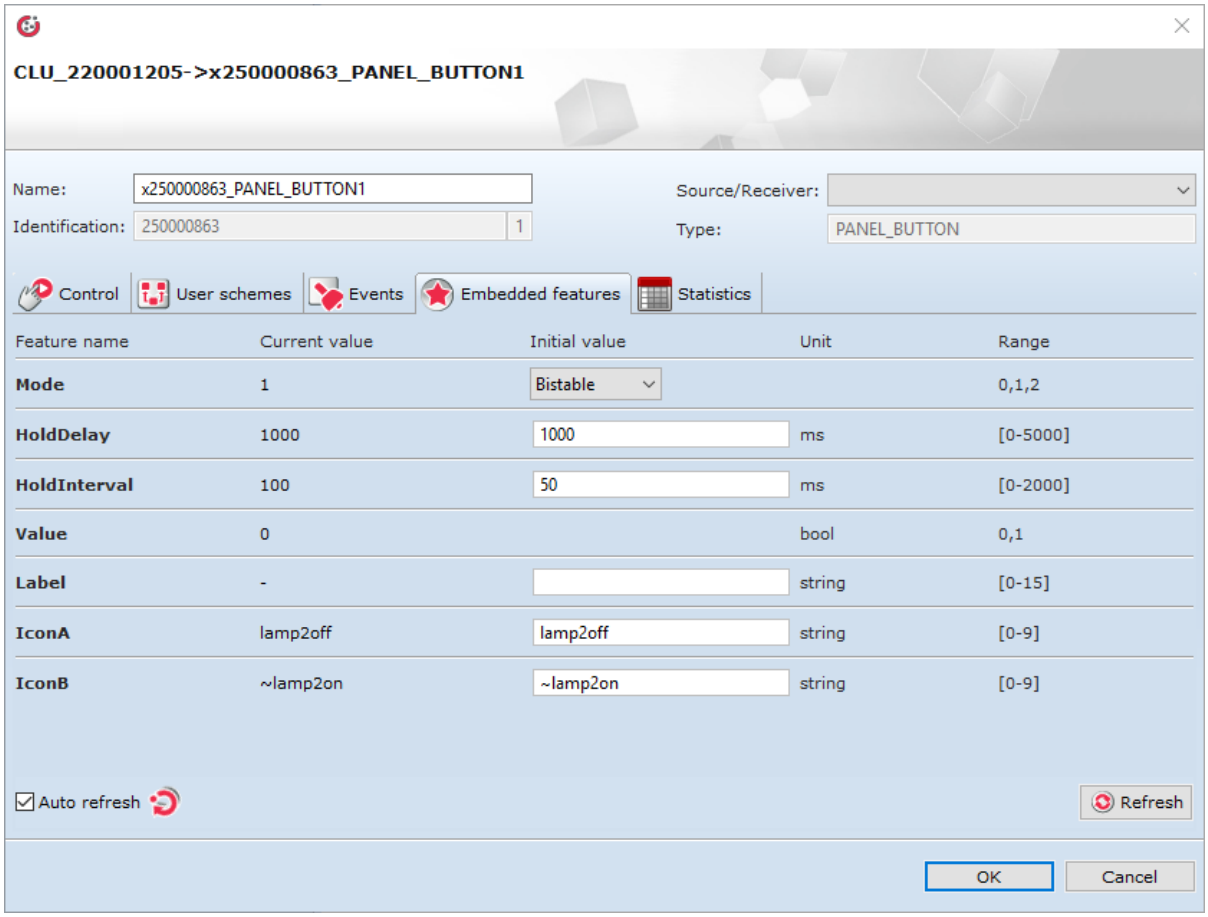
4.2 Creating button and display configurations

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`.

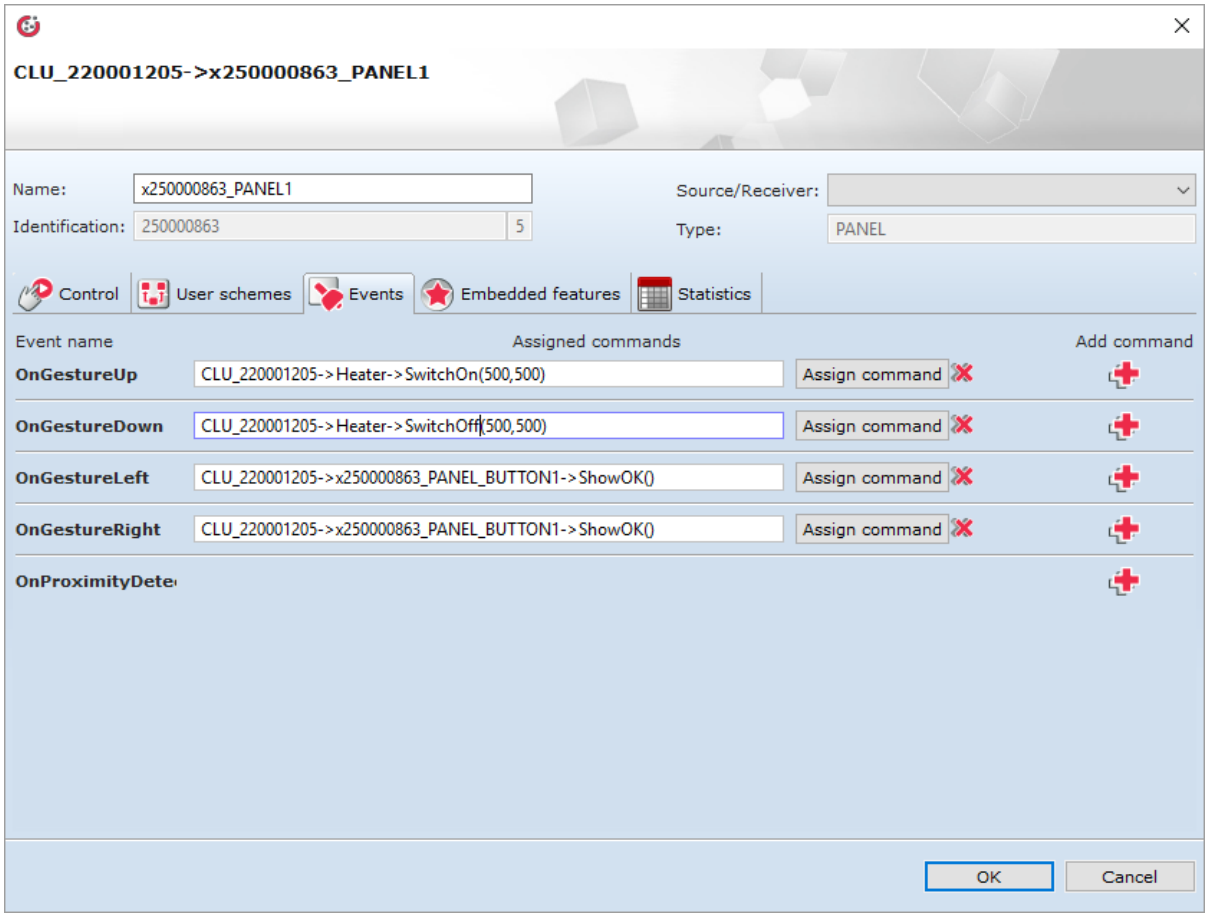
> **NOTE!** The `SetIcon` method has a higher priority in the system than the `SetLabel` method!

- Send the configuration to the CLU.

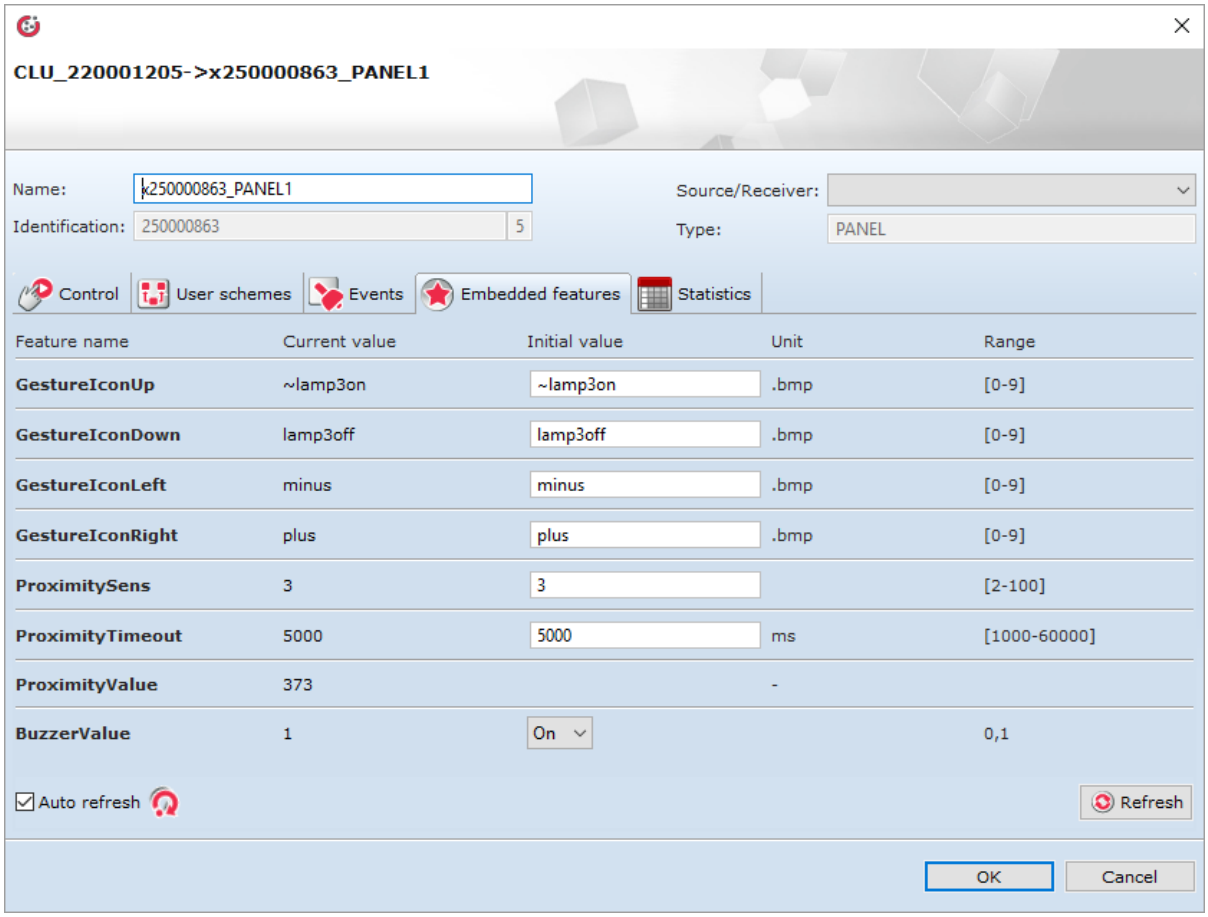
4.3 Creating a gesture sensor configuration

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:



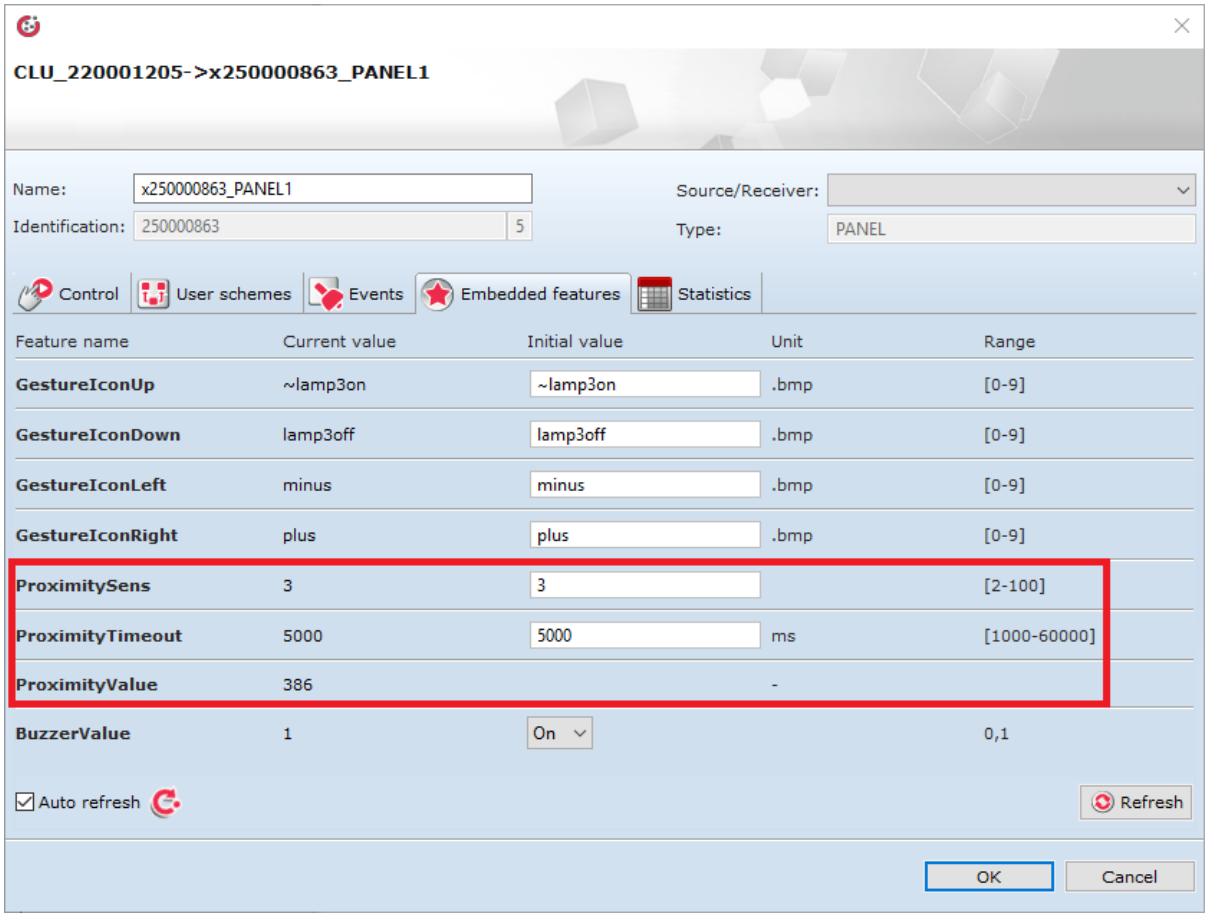
The use of icons will be possible when they will be loaded onto a microSD card with the extension *.bmp*.

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

4.4 Configuration of the proximity sensor

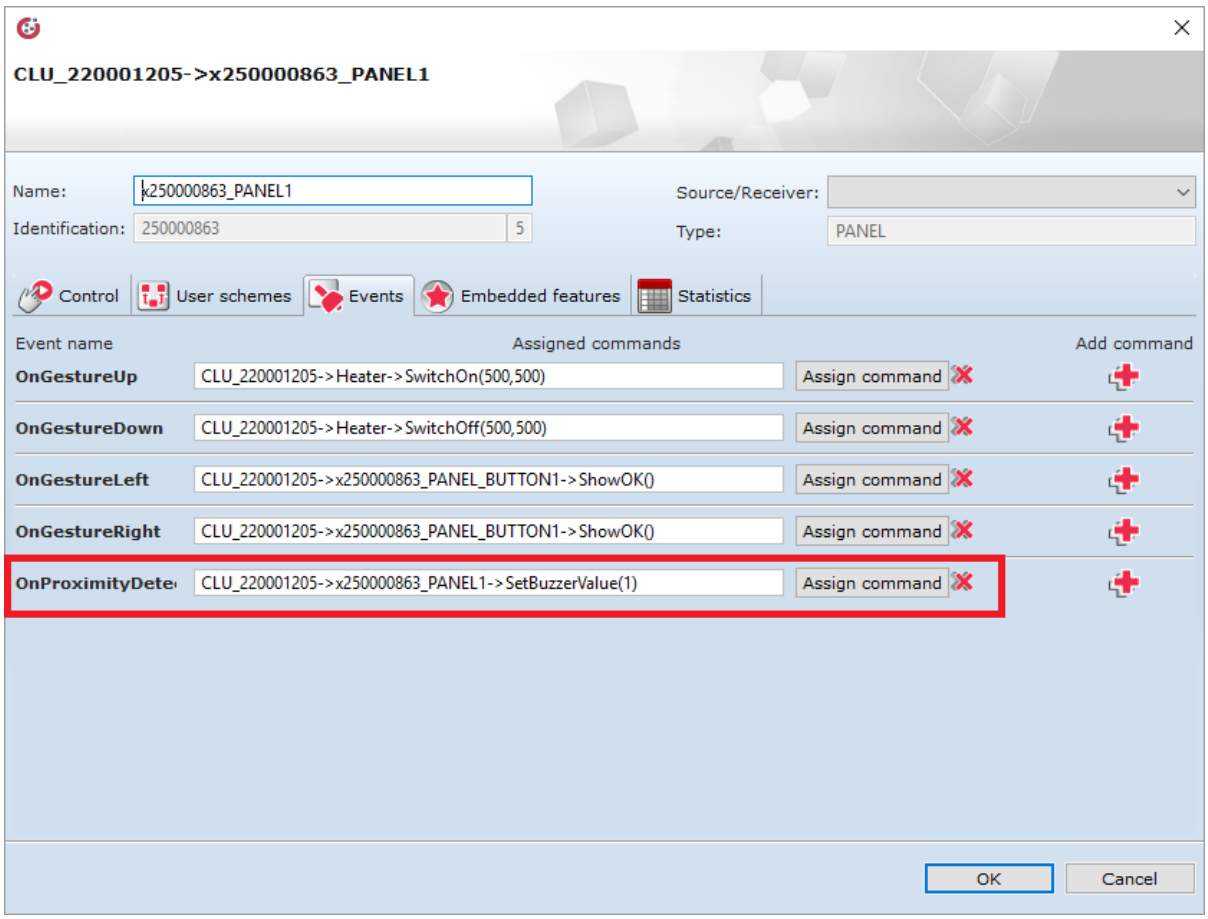
To set the proximity sensor parameters:

- 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:

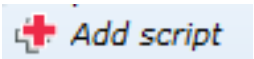


- Send the configuration to the CLU.

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!

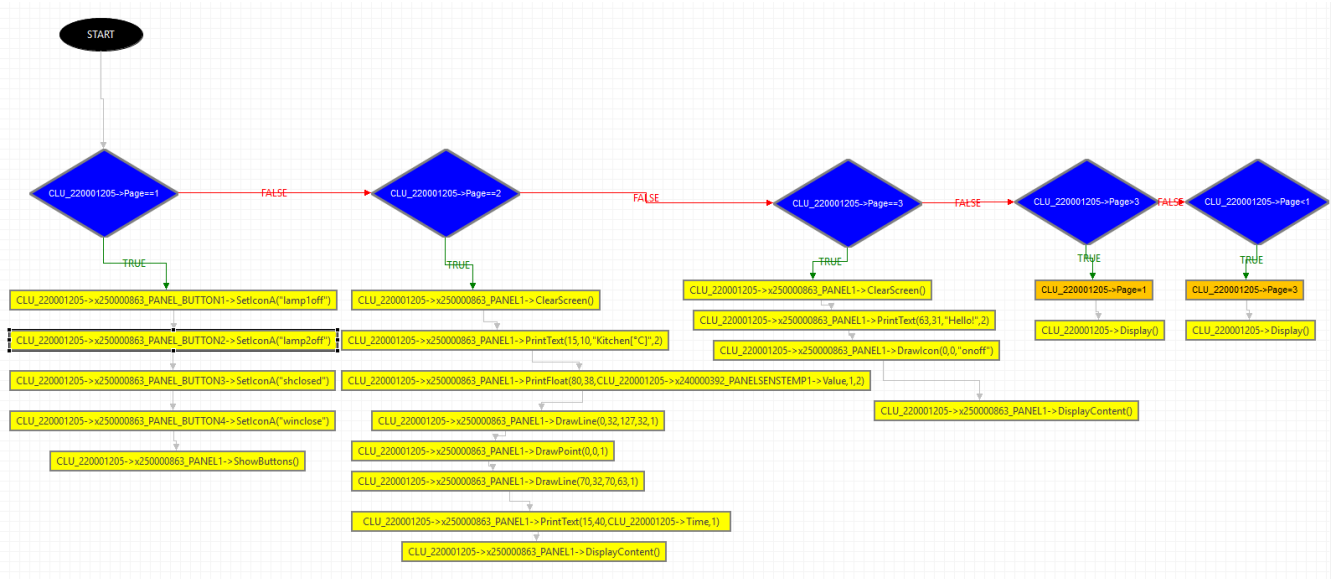
> **NOTE!** In the case of creating multiple pages, setting the button in the bistable mode - using the feature / method - will not correctly read the state of the relay (due to the different functionality of the buttons when changing pages)!

- **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.

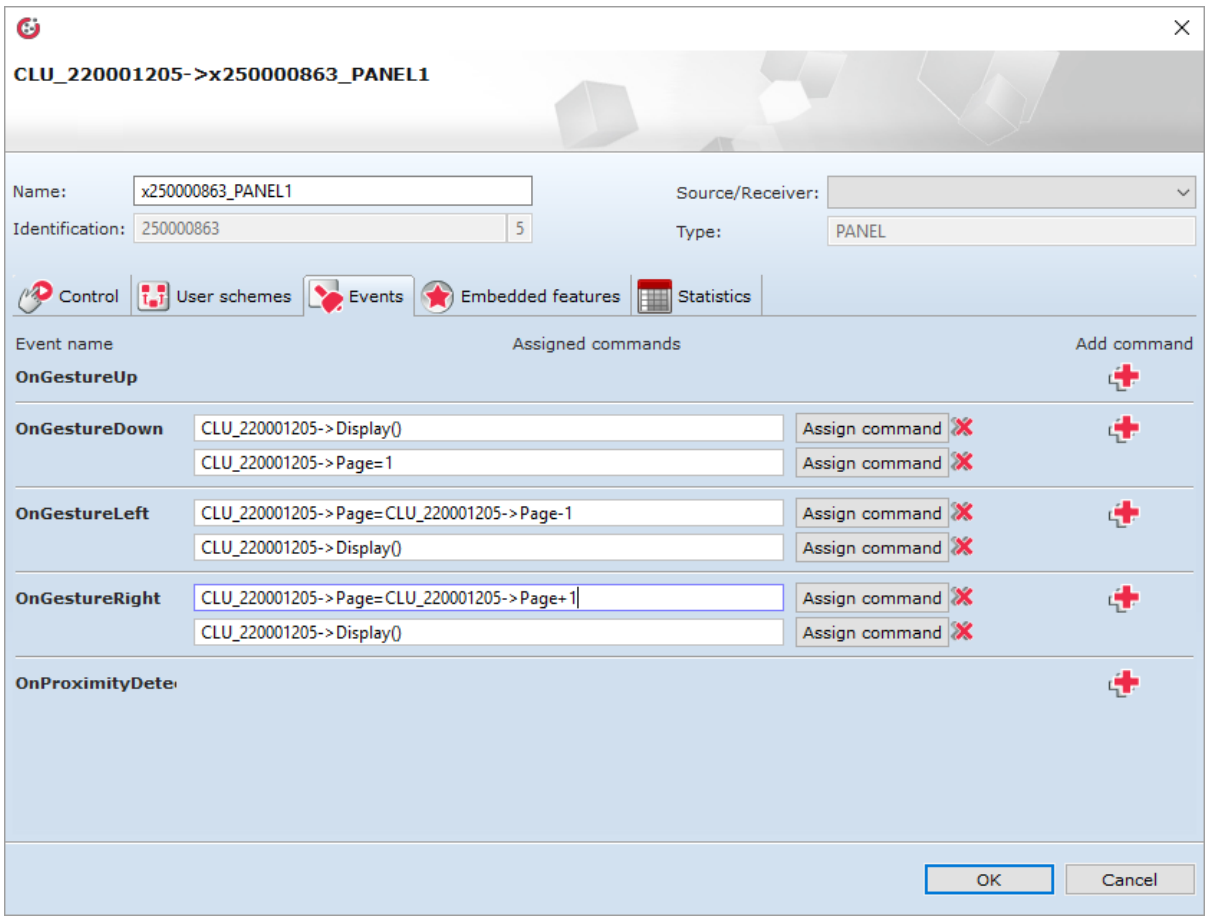
The implementation of all the methods described above is presented in the screen shot of the sample script:



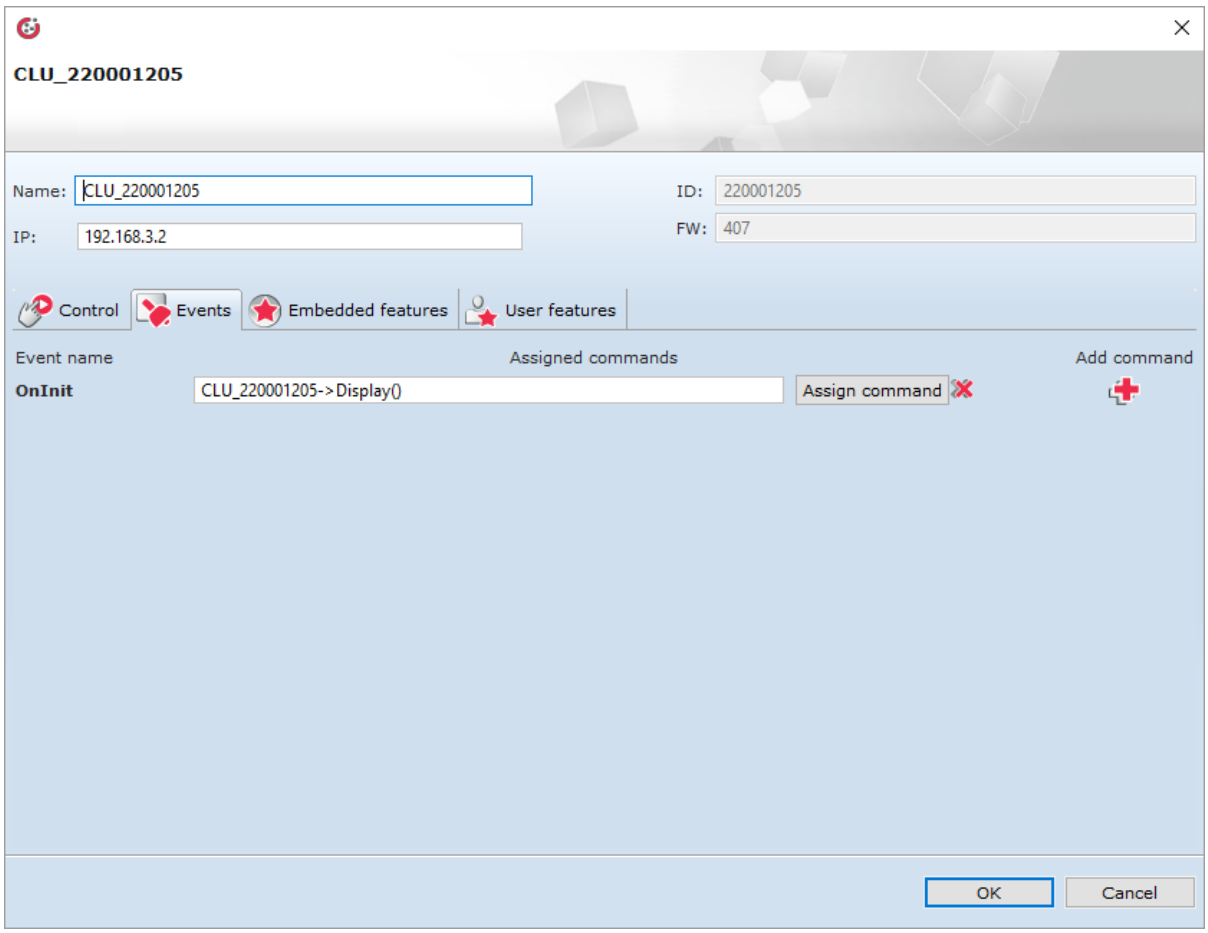
The above script is placed at the end of the document in the text version (point 3).

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:

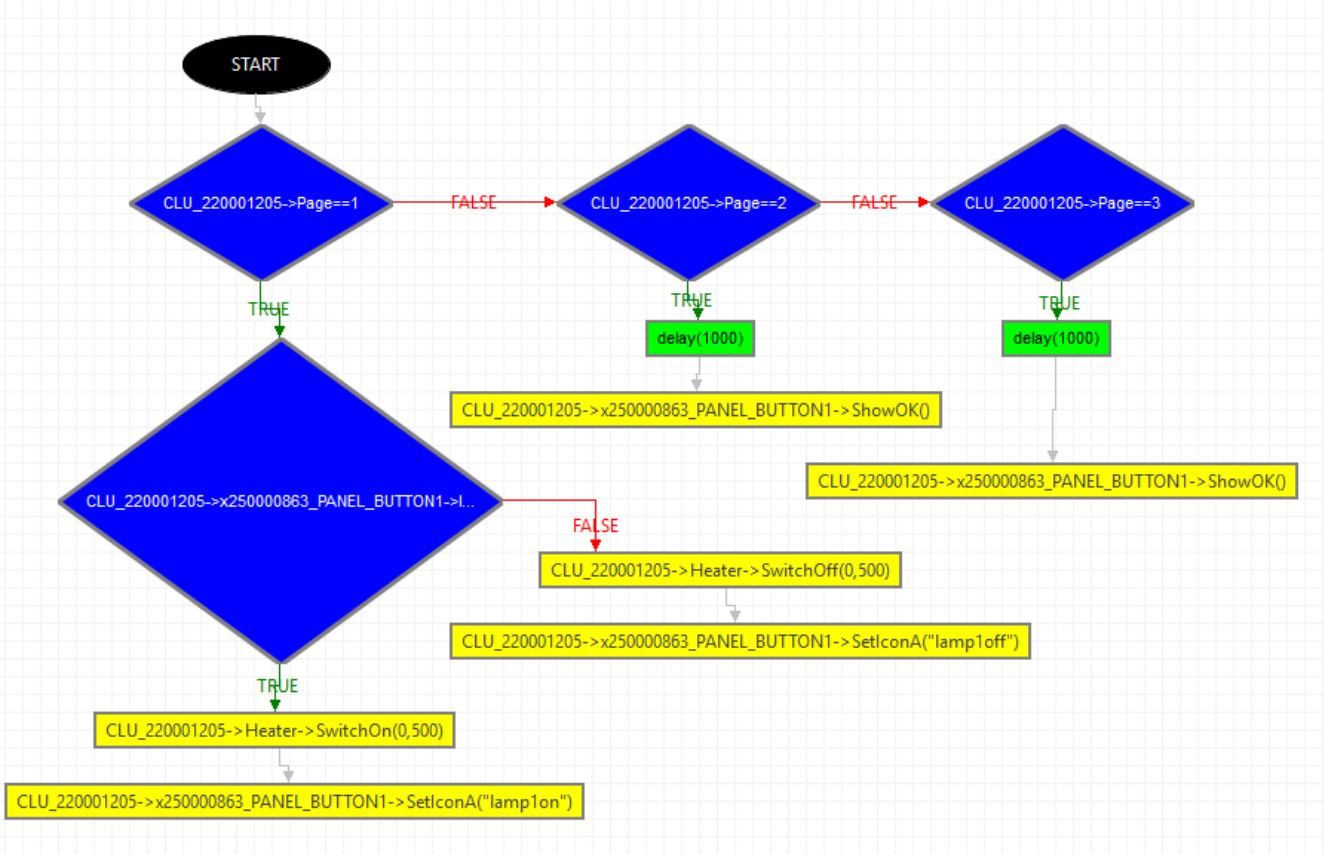


- Assign *CLU* -> `OnInit` to the script call *Display*:



- 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)
`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

Name	Description
`SetProximityTimeout`	Sets the time in seconds after which the display goes out
`SetBuzzerValue`	Control of sound signaling (On / Off)
`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 triggeredwhen 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

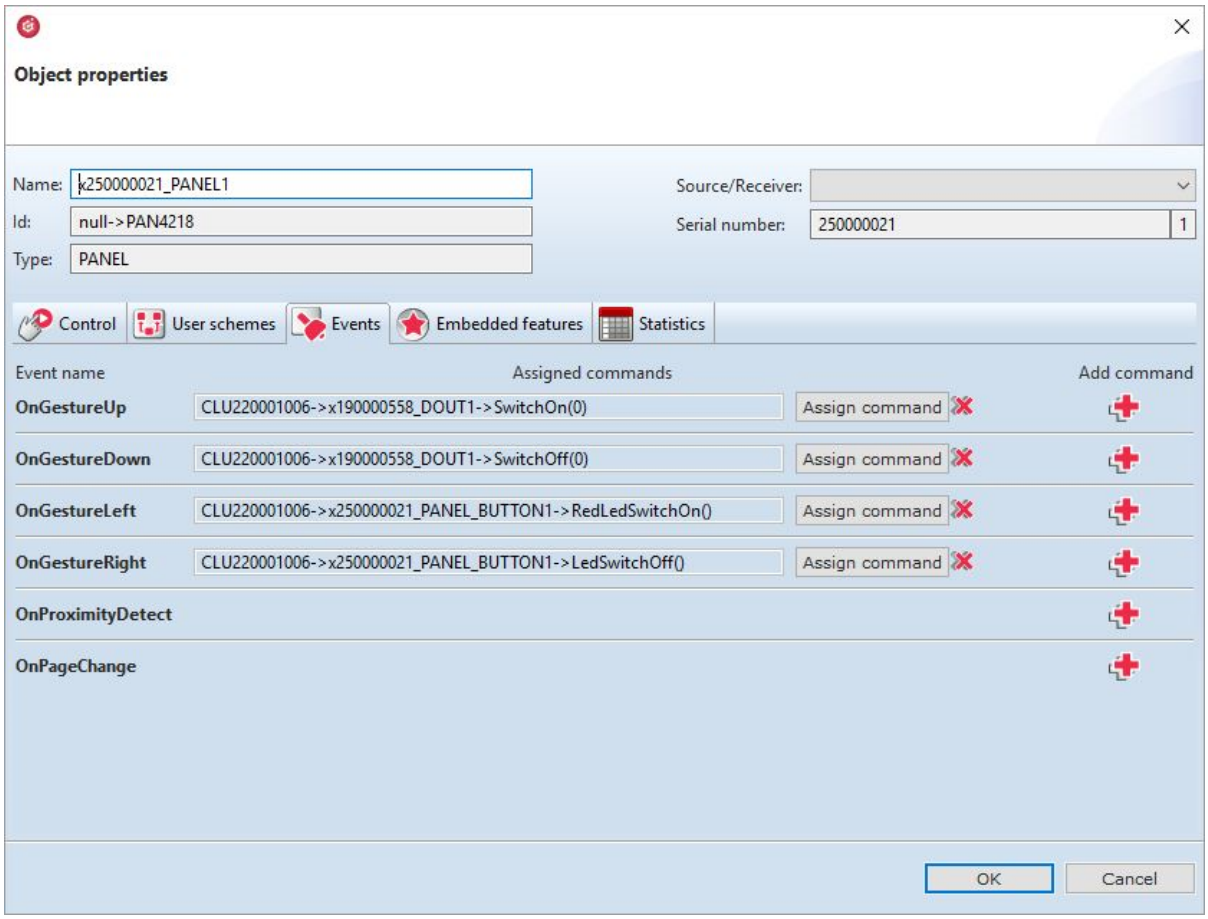
Name	Description
`Threshold`	Hysteresis size (accuracy 0.1 ° C / 0.1%) defining the sensitivity at which events are generated: `OnChange`, `OnLowerValue`, `OnRaiseValue`
`Sensitivity`	The period (in ms) at which the sampled values are averaged
`MinValue`	The minimum value of the `Value` feature, exceeded 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

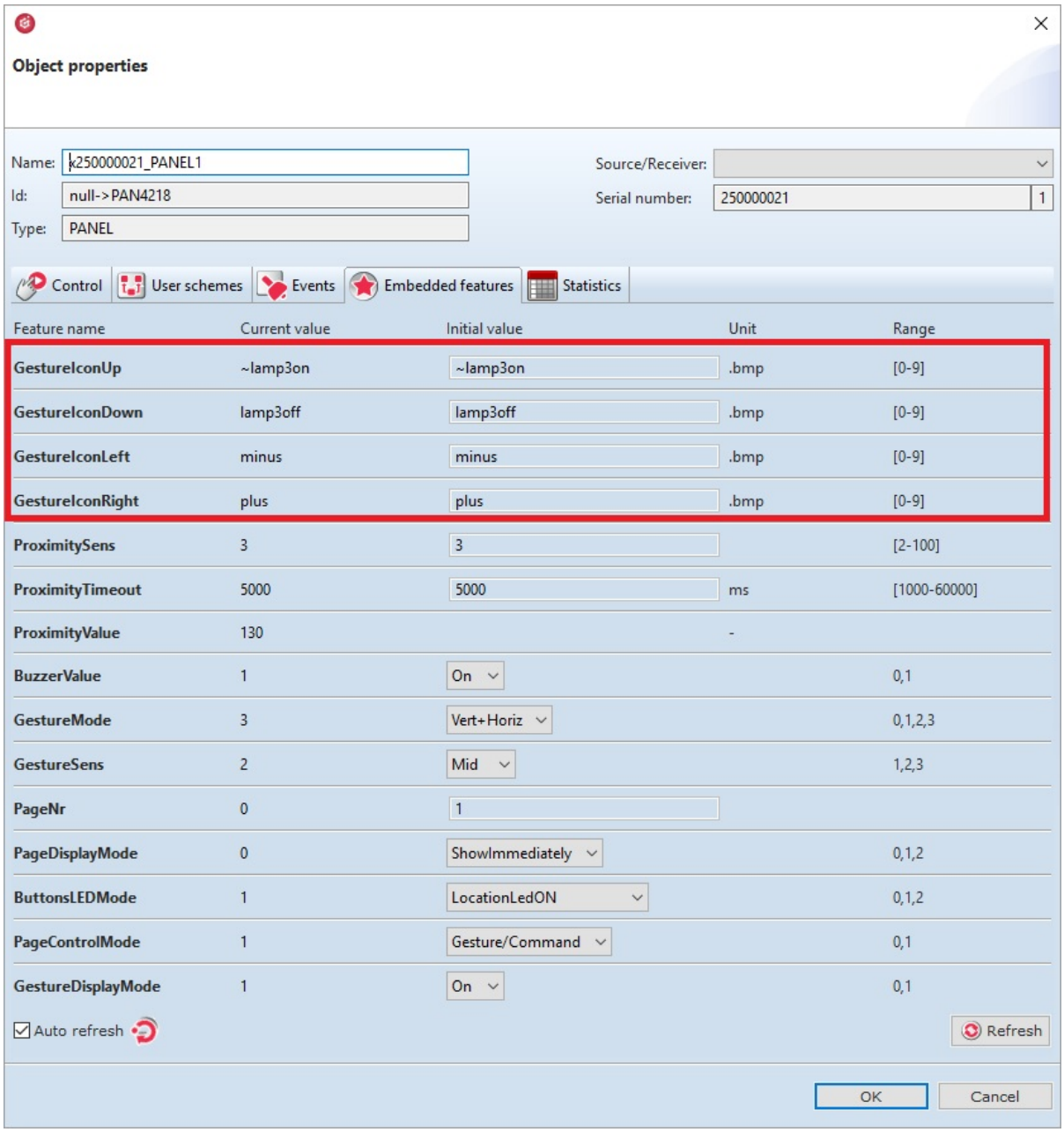
- To create a configuration for a gesture sensor:
  - Open - by double click - object *Panel*;
  - Go to the tab *Events*;
  - Assign methods to the events `OnGestureUp`, `OnGestureDown`, `OnGestureLeft`, `OnGestureRight` (clicking on the '+' on the right of each method):



> **NOTE!**

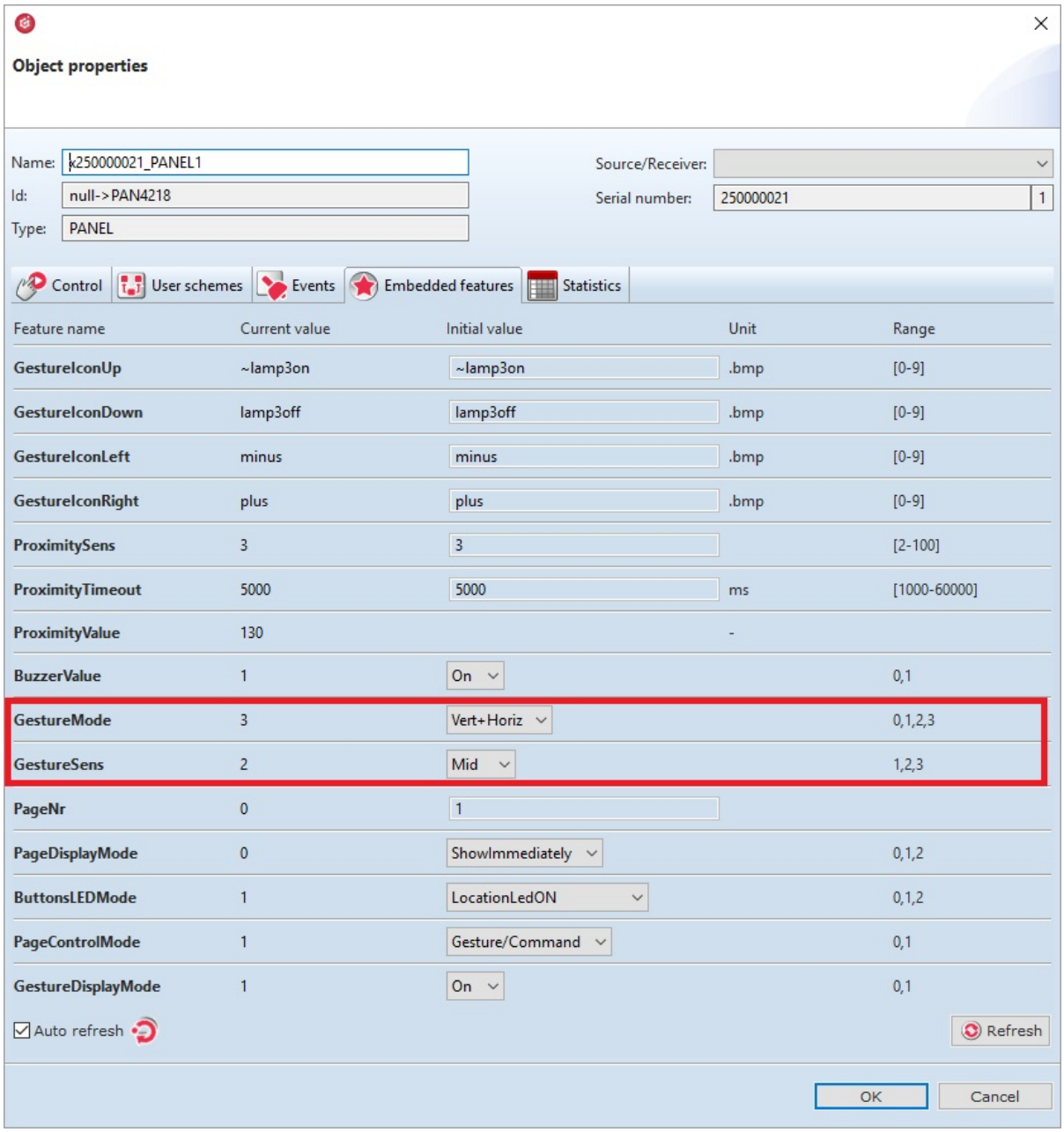
> In the case of configurations containing the configuration of pages (Buttons / FreeDraw / Thermostats), the methods assigned to the OnGestureLeft and OnGestureRight events will not be executed. This is related to the predefined functionality of switching between pages. You can change the way pages 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.

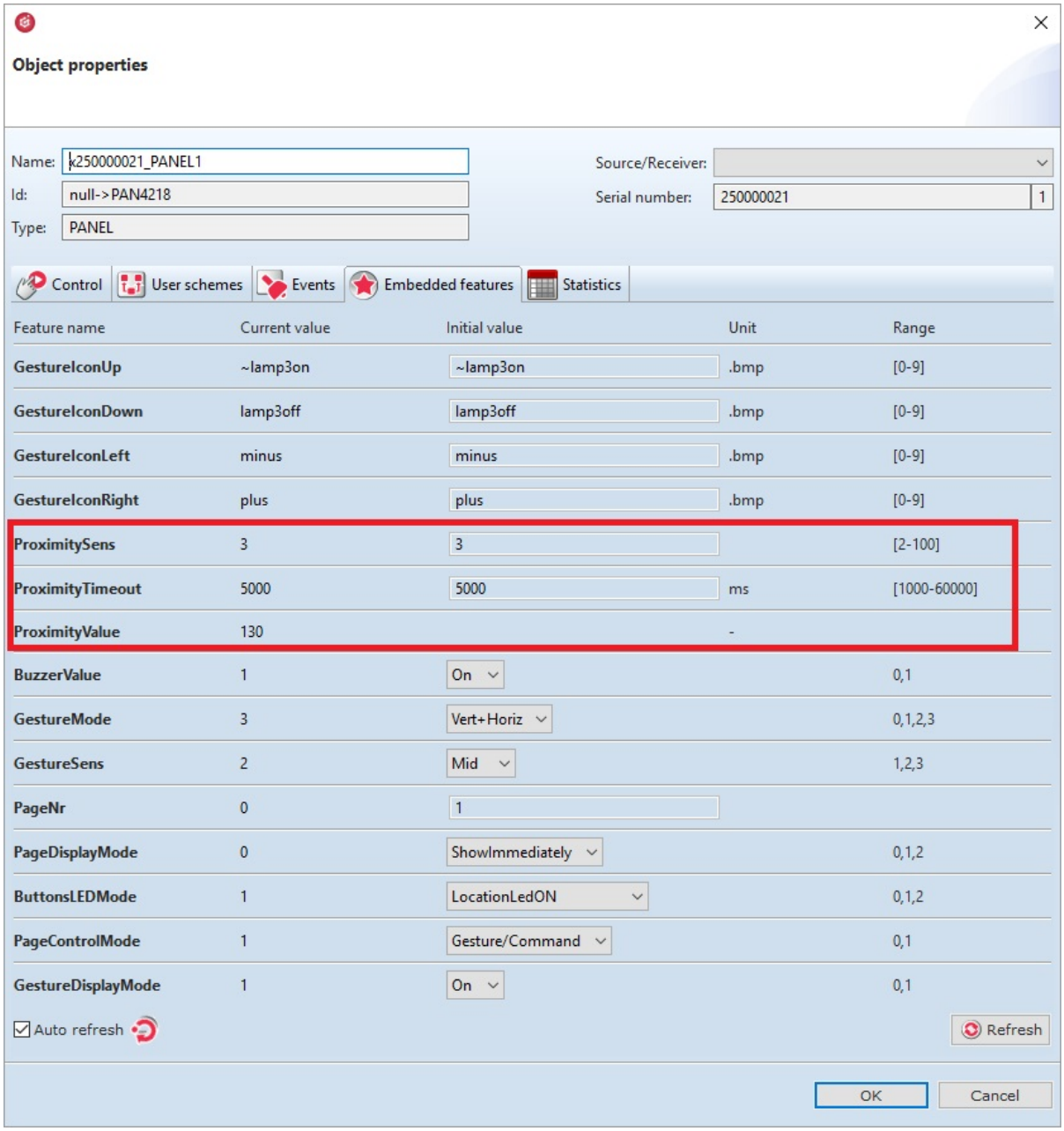
The above features can be set both in the tab *Built-in features*, as well as using methods: `SetGestureIconUp`, `SetGestureIconDown`, `SetGestureIconLeft`, `SetGestureIconRight`, `SetGestureMode`, `SetGestureSens` (in the methods of the Panel object).

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

5.3. Configuration of the proximity sensor

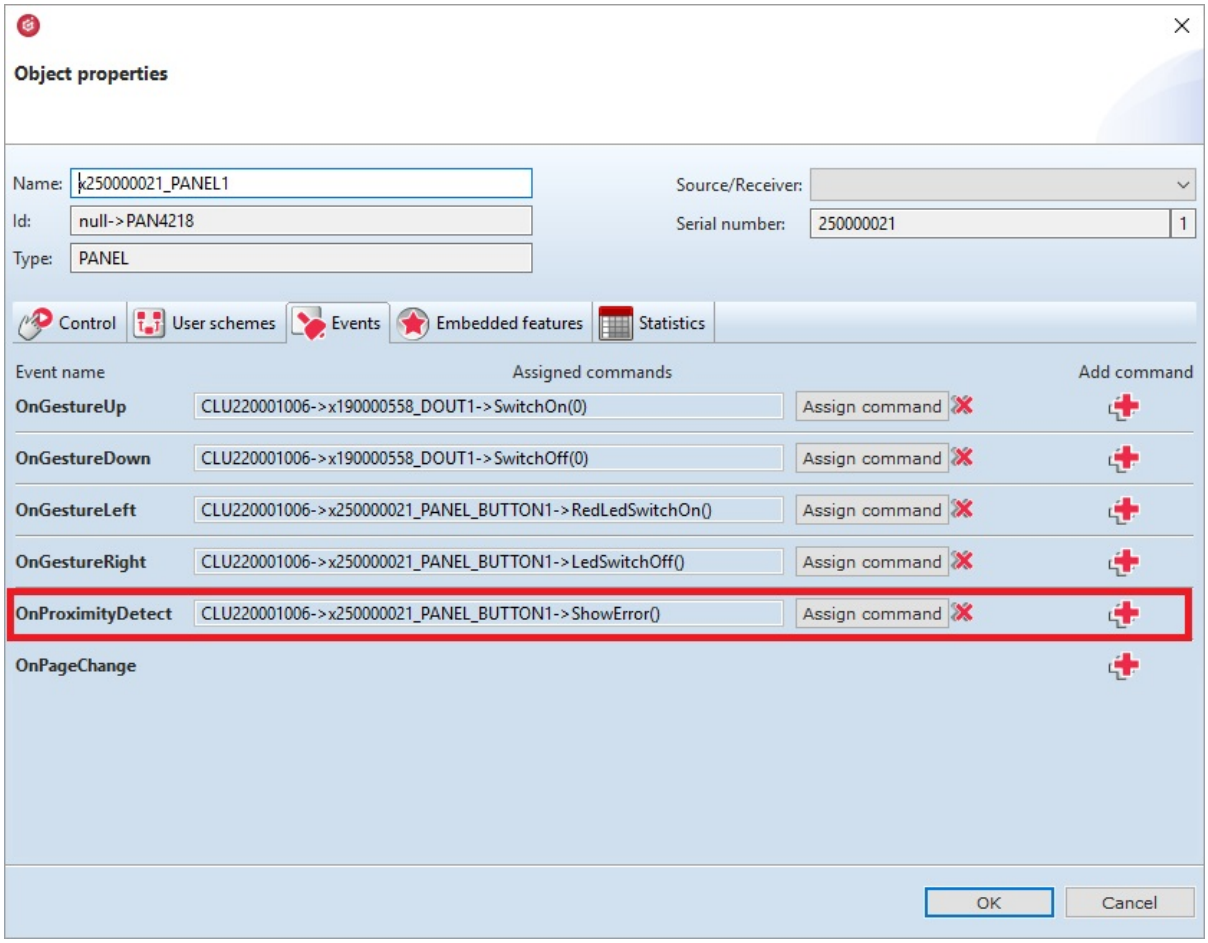
To set the proximity sensor parameters:

- 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:



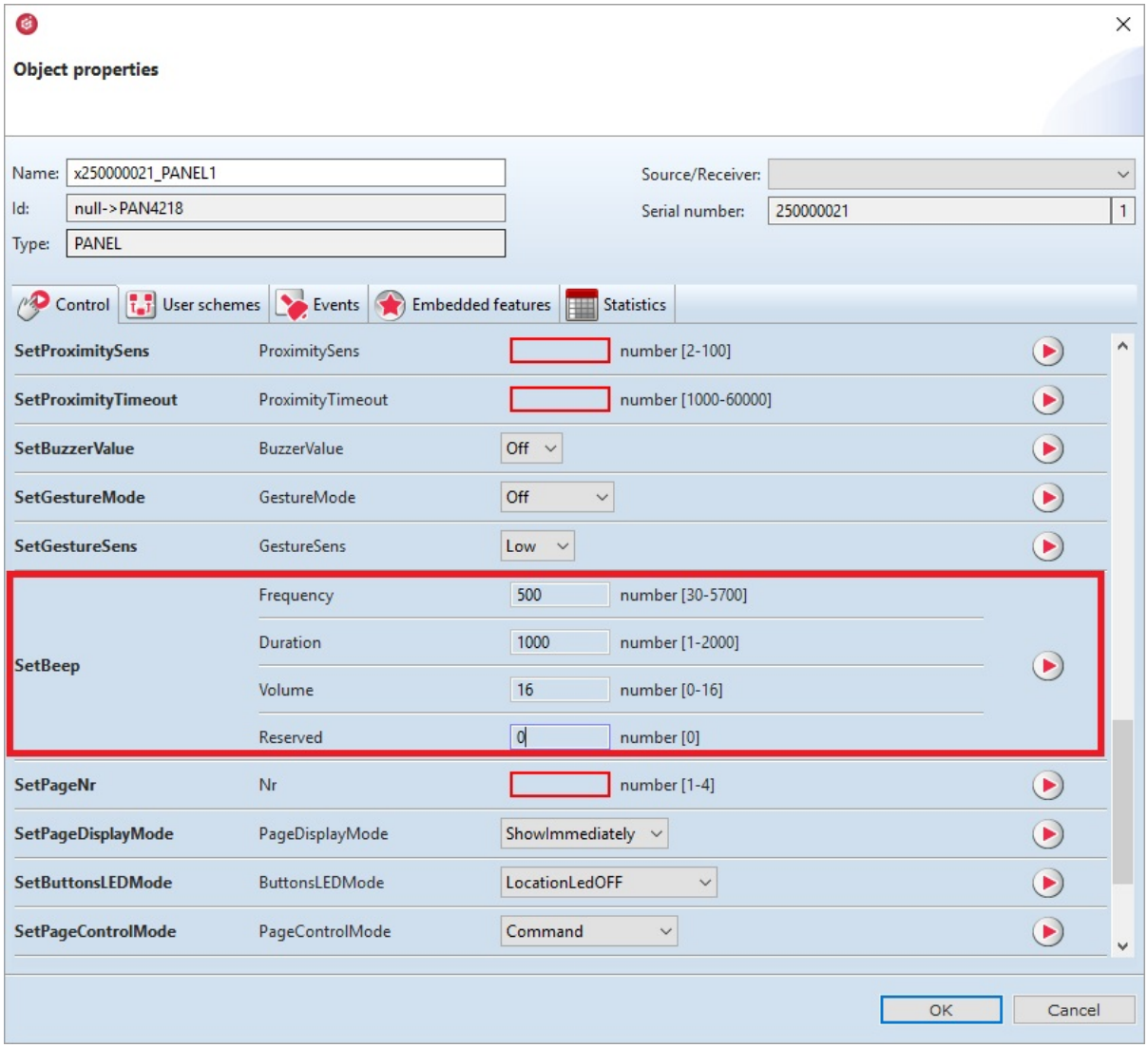
- Send the configuration to the CLU Z-Wave.

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:

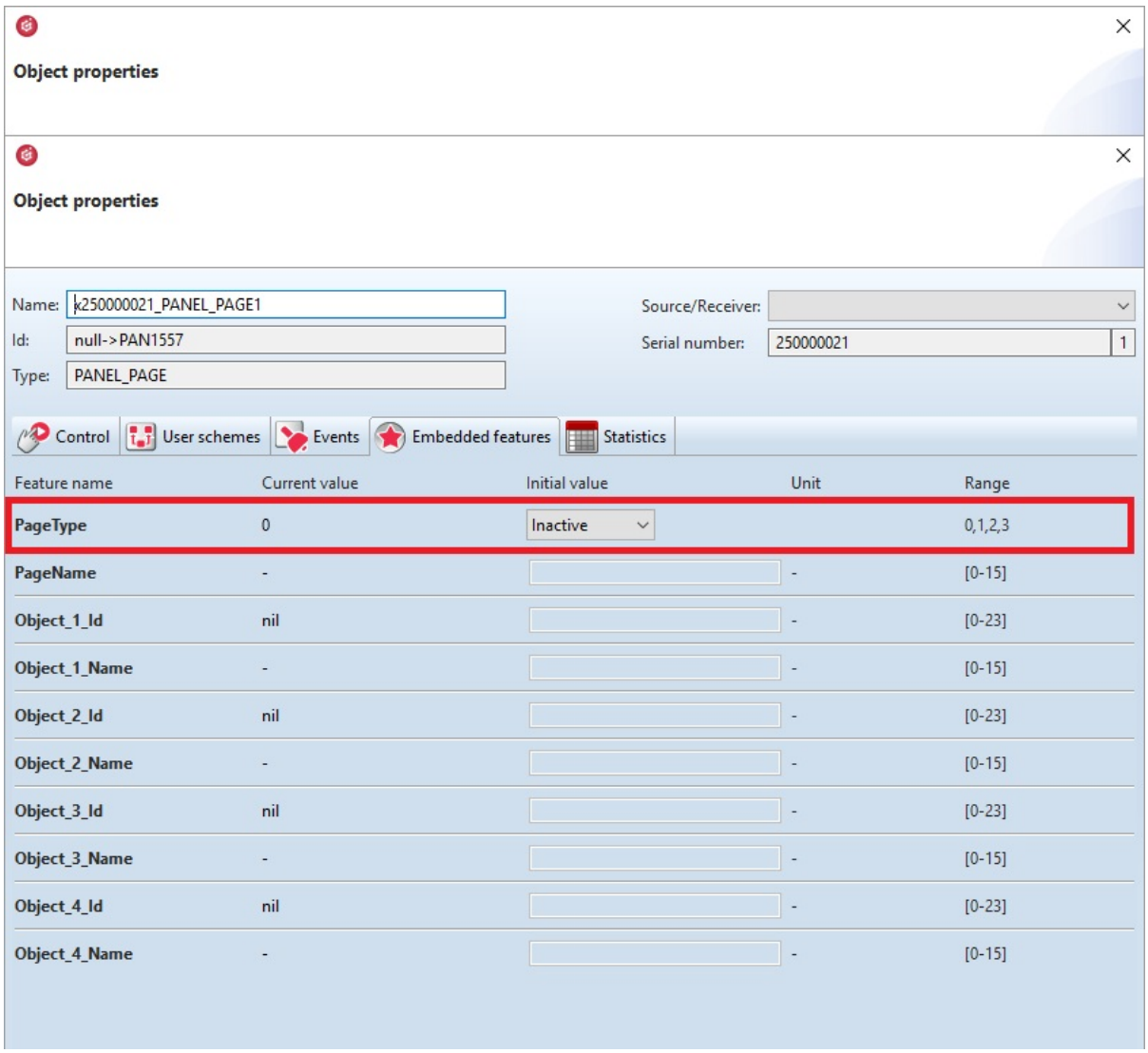
  Methods / Features:

  - `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

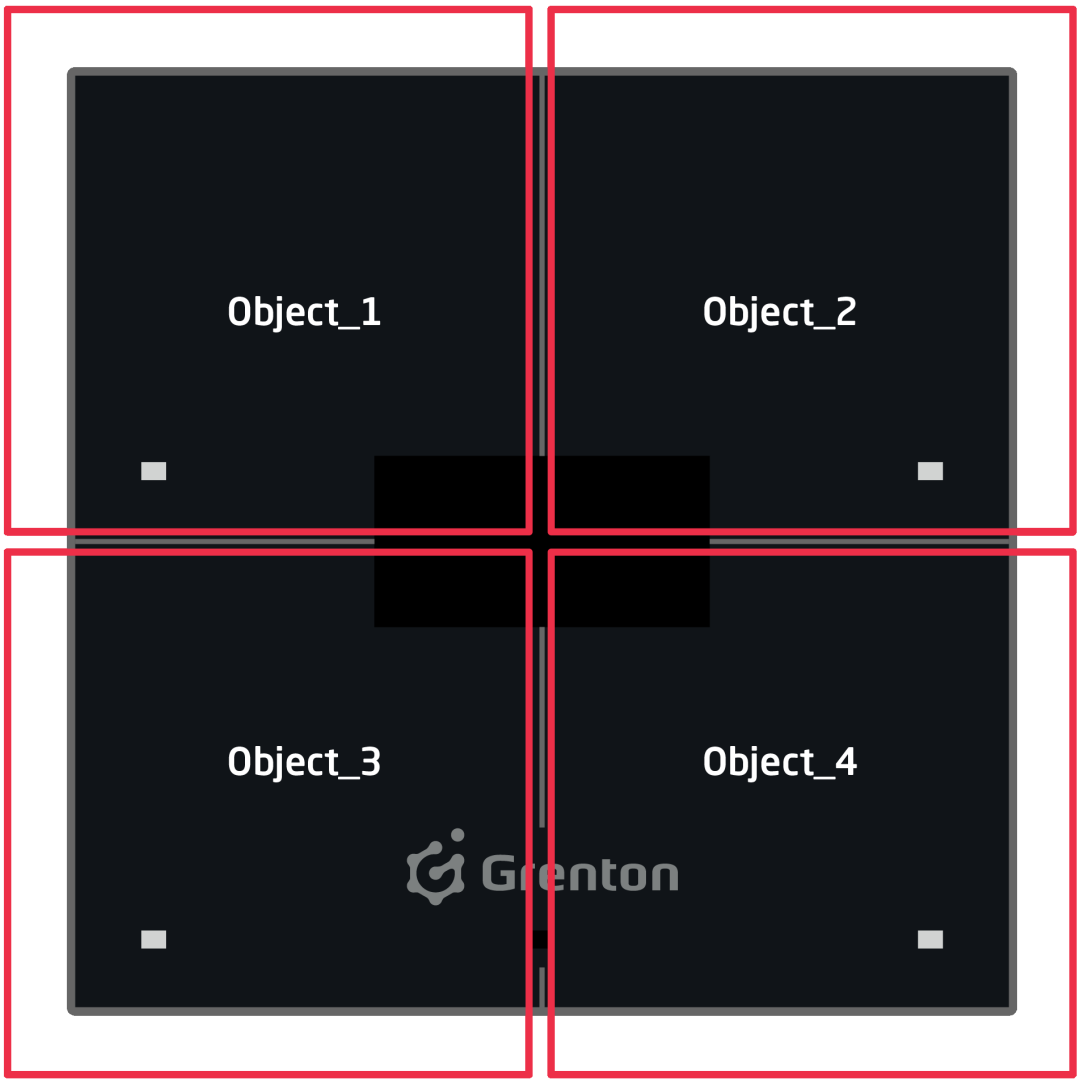
When starting work with the new version of the Smart Panel module, the device is in the default configuration, which is backward compatible. All four Panel_Page objects have the built-in feature `PageType` set to *Inactive*. This allows you to work with the panel in the same way as before (in version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

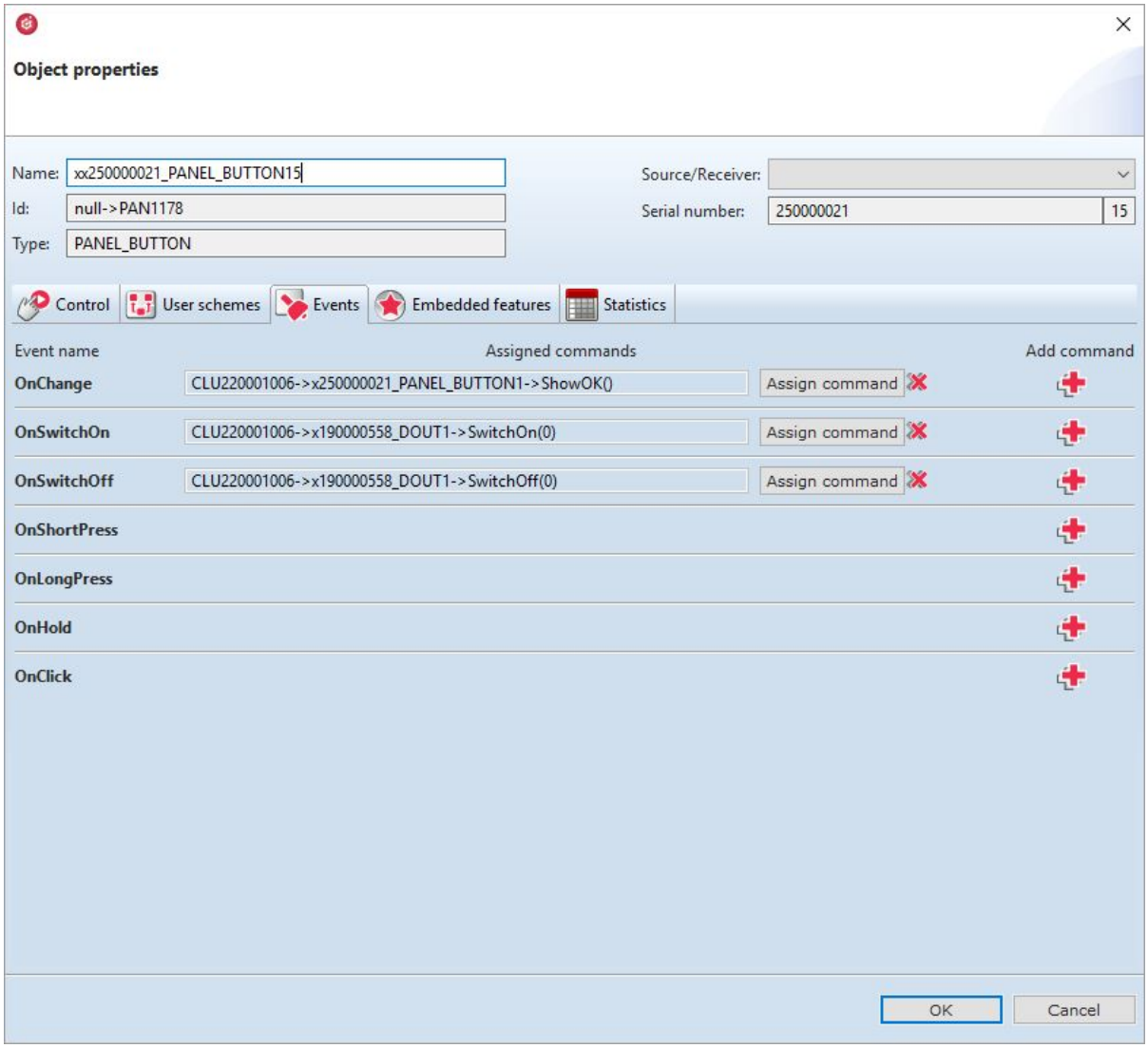
In the *Buttons* operating mode, there are 4 physical touch buttons and up to 16 virtual buttons spread over 4 pages, each of which can perform independent functions. It is also possible to combine / merge 2,3,4 objects into one button (described in more detail in subsection XII.5.10).

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

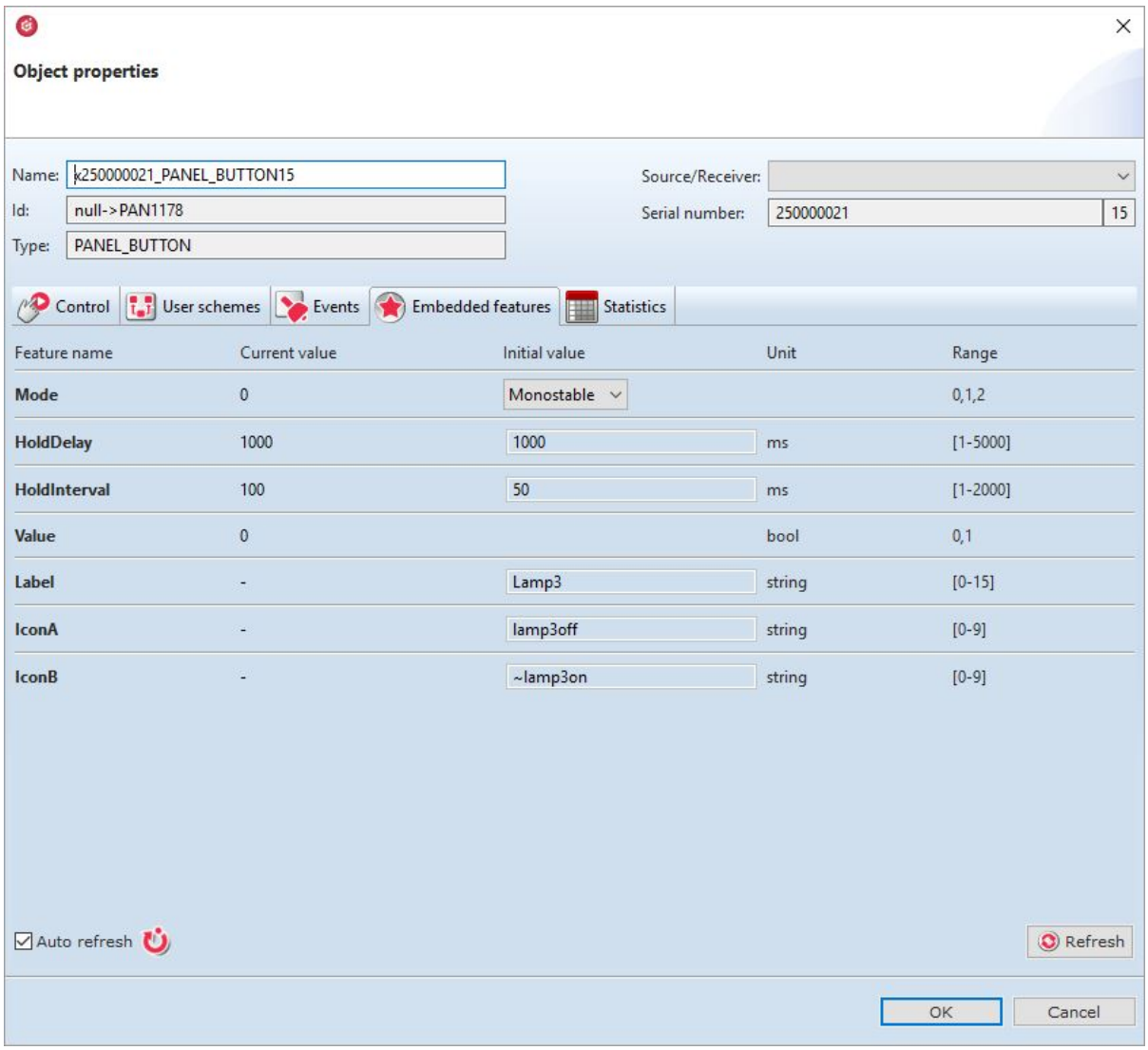


Page type „Buttons/FreeDraw”

- 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`):



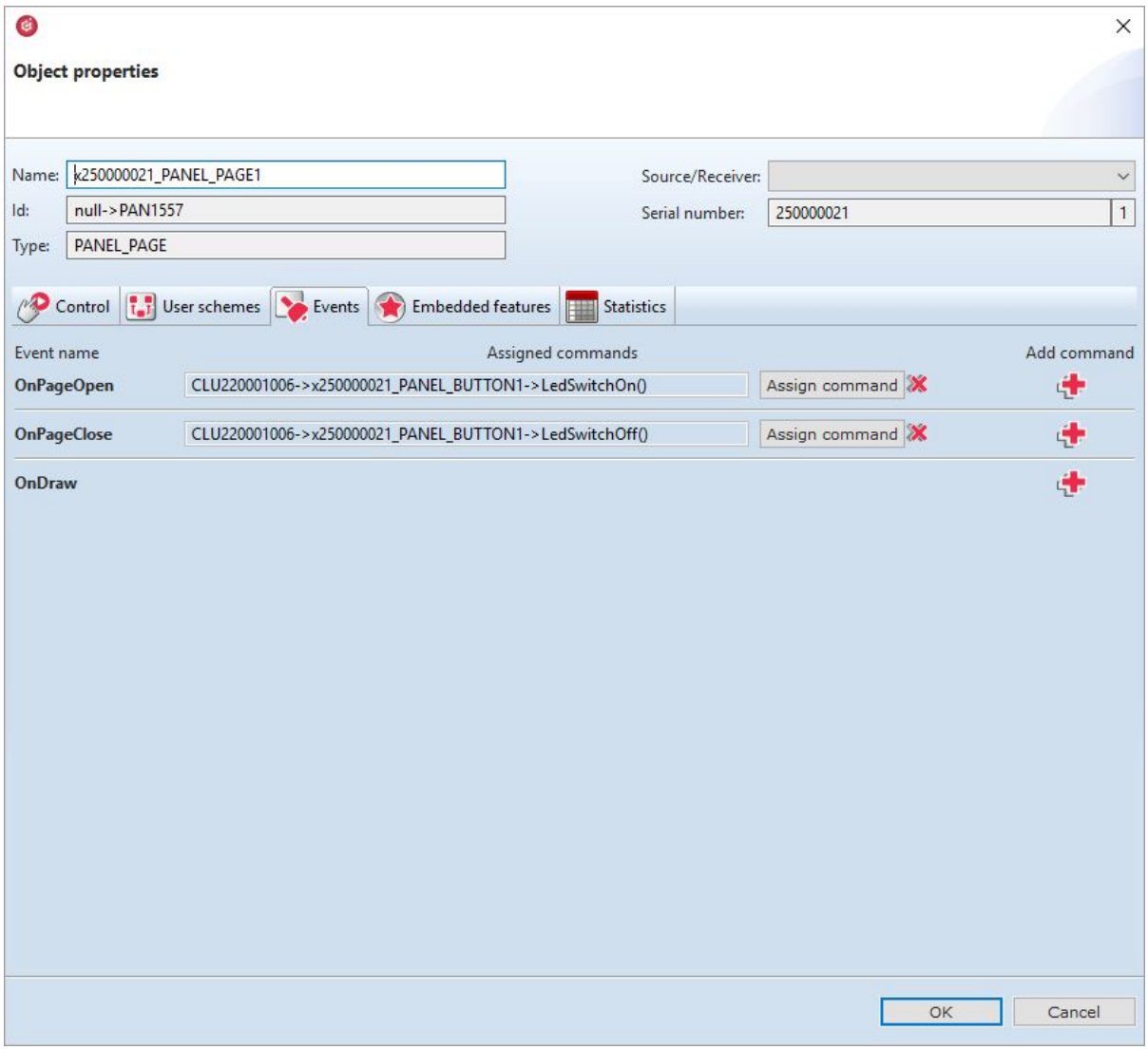
The above built-in features can be set both in the tab *Built-in features*, as well as via the methods: `SetLabel`, `SetIconA`, `SetIconB`.

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

- Send the configuration to the CLU Z-Wave.

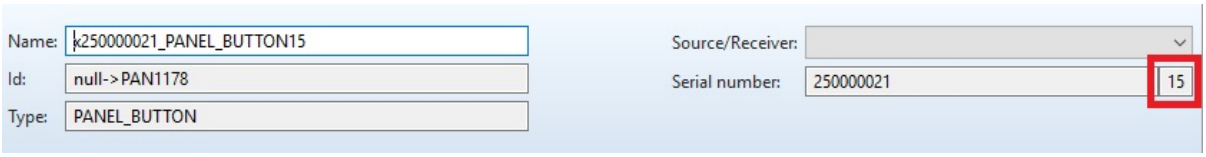
The next step in creating the configuration is configuring Panel_Page objects depending on the number of buttons. One Panel_Page object supports up to 4 buttons. To do this:

- 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):

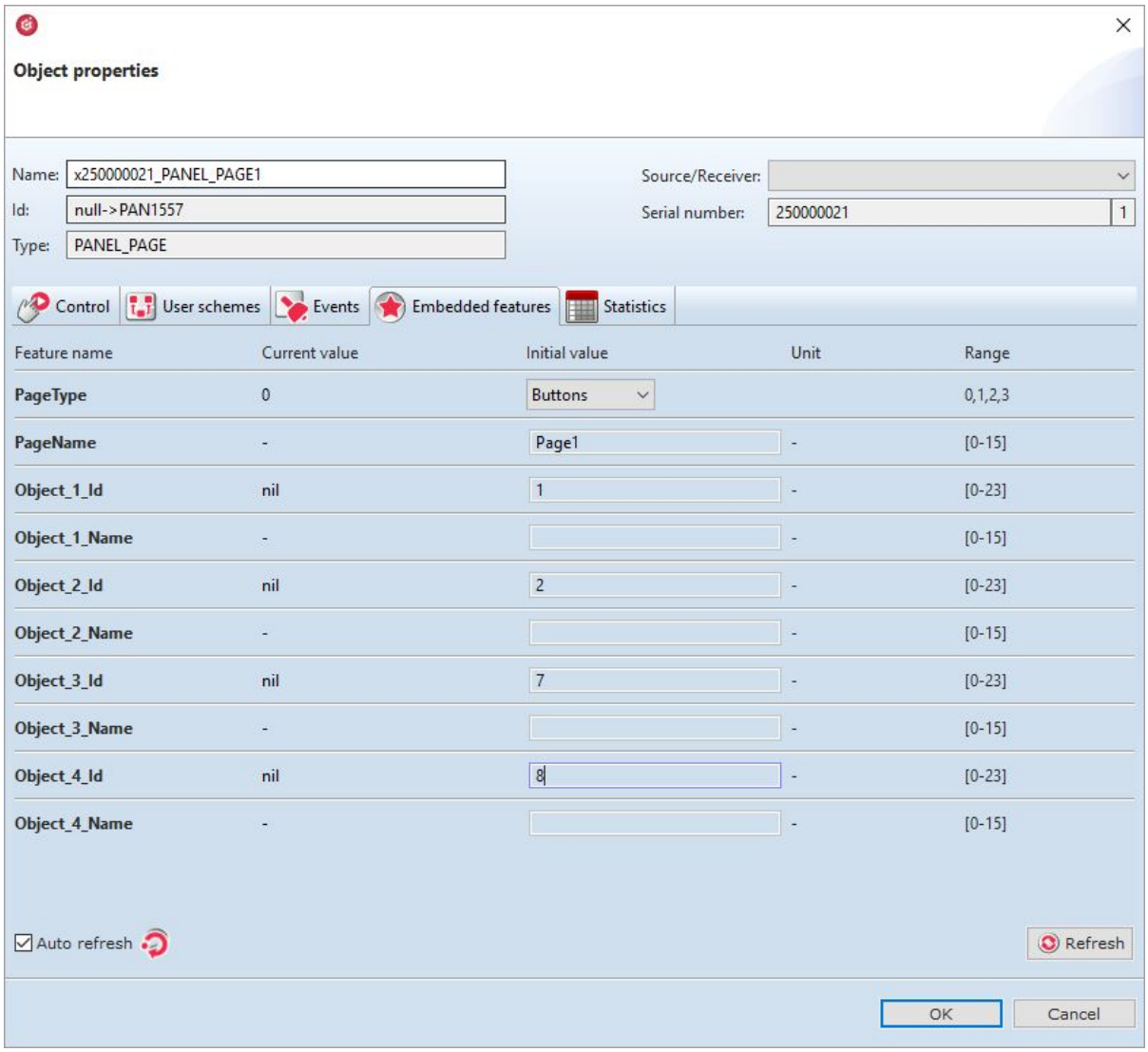


> **NOTE!**
>
> For page type *Buttons*, the `OnDraw` event is not generated.

- 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;

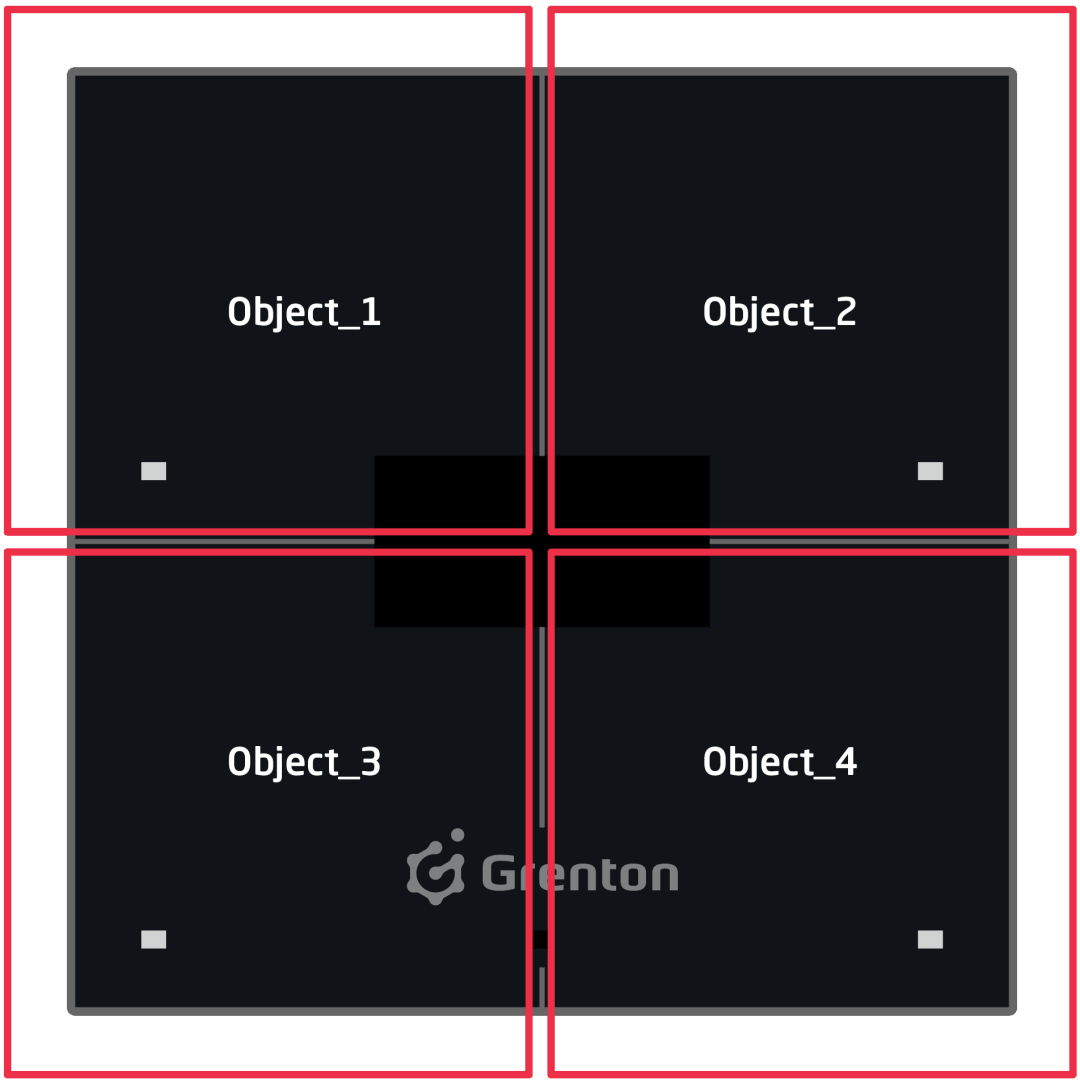


> **NOTE!**
>
> Sending the configuration only with the defined page type, without setting the binding of objects with the buttons is connected with starting the panel operation mode as *Buttons*. However, the buttons on the module will be inactive. This is related to the non-complementing of the Object_X_Id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

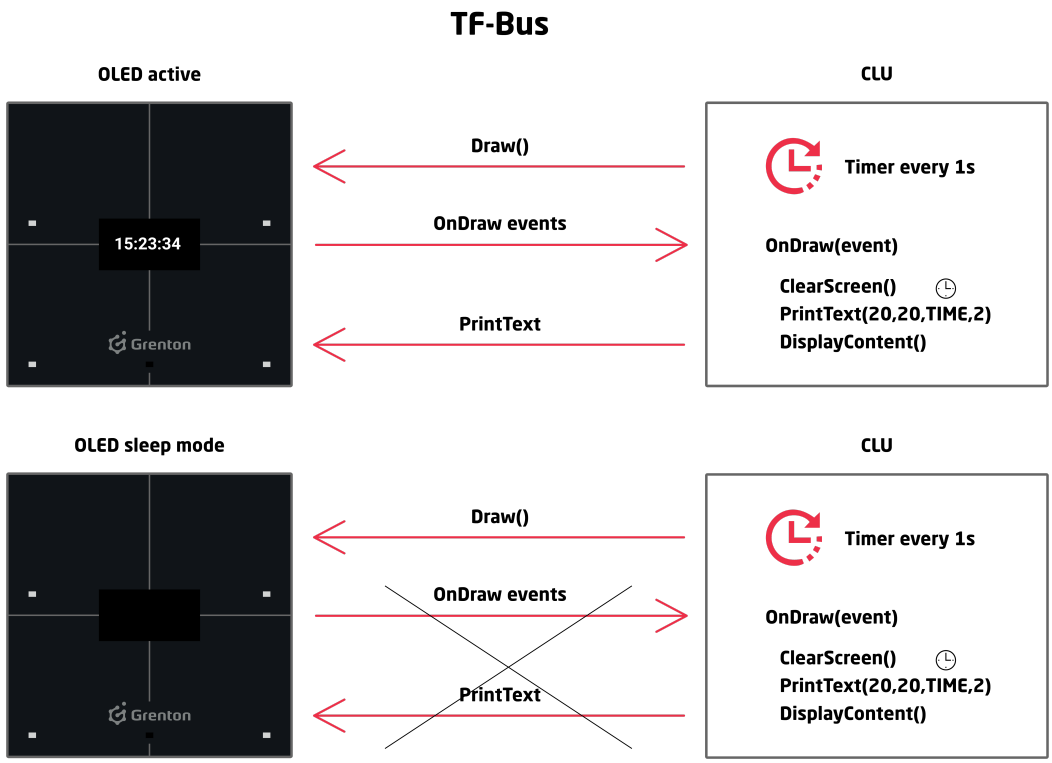
Example of a script that draws content on the display (*Page1*):

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()
```

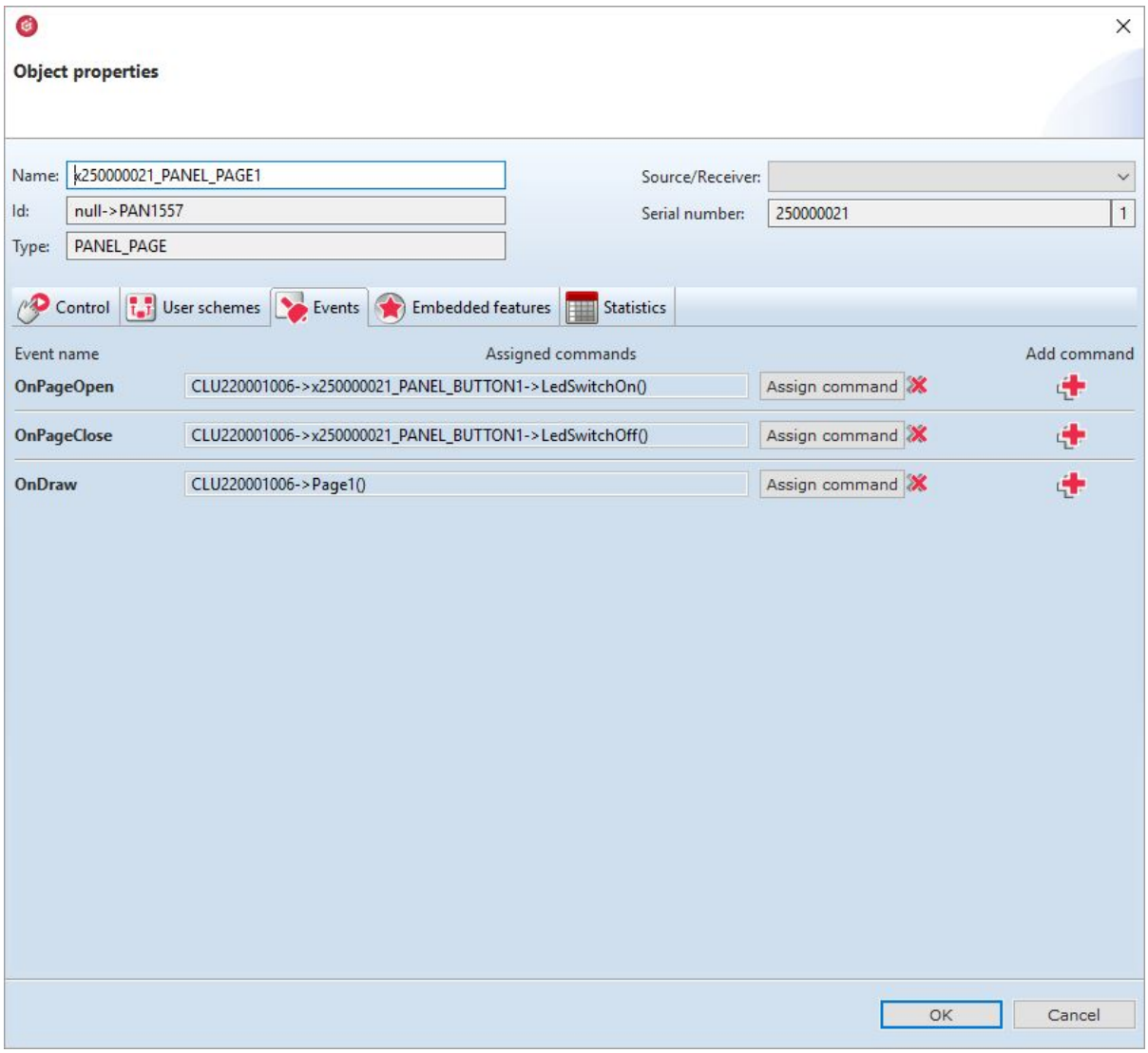
> **UWAGA!**
>
> A restriction has been introduced in the drawing mechanism. CLU Z-Wave expects 2 seconds to finish drawing with the DisplayContent method. Otherwise, the following message will be displayed on the screen:
>
> "page: `PageName`
>
> free draw
>
> ! TIMEOUT !"

The following figure shows the current drawing mechanism.



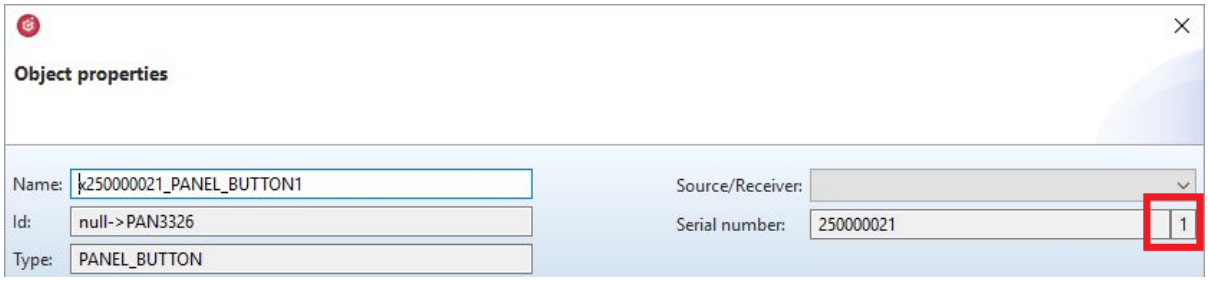
The next step in creating the configuration is configuring Panel_Page objects depending on the number of buttons. One Panel_Page object supports up to 4 buttons. To do this:

- 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):

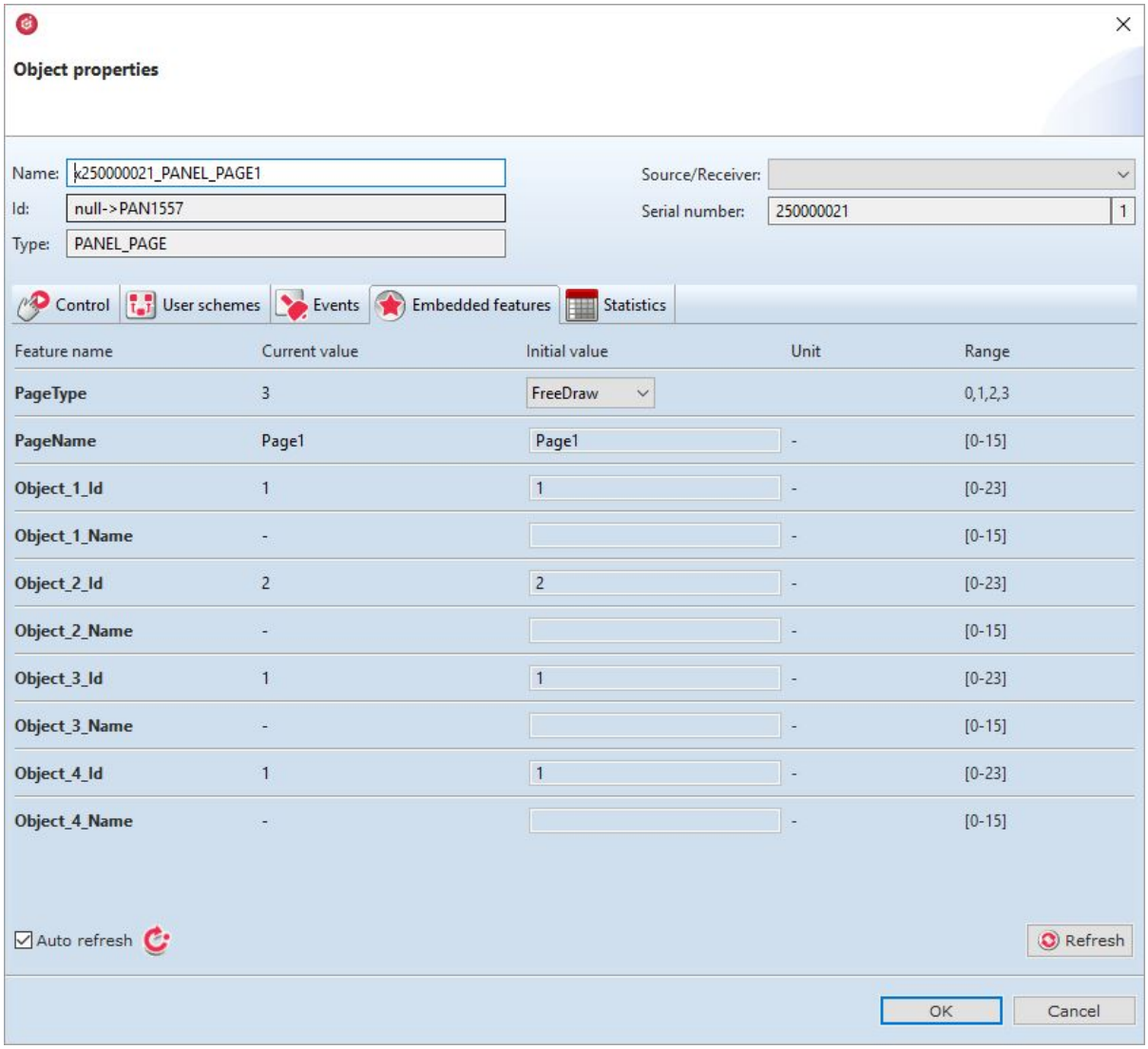


> **NOTE!**
>
> For the page type *FreeDraw*, complete the `OnDraw` event.

- 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.

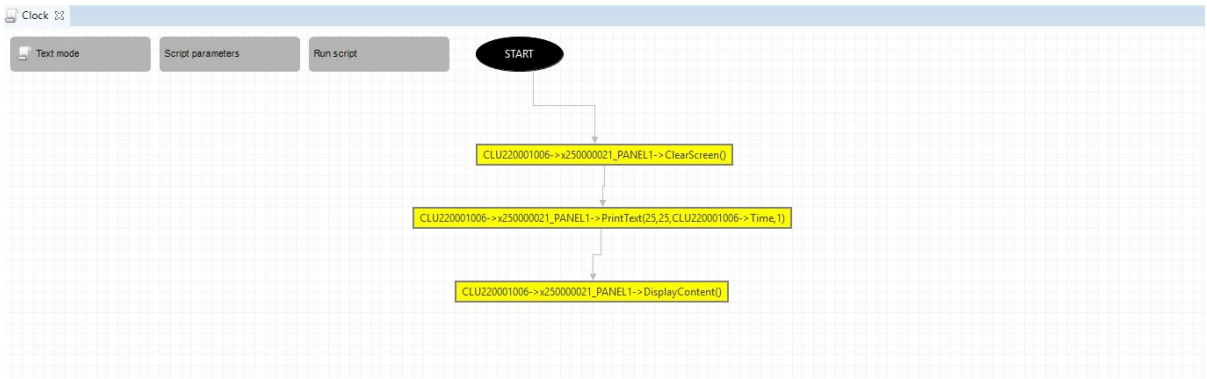
> **NOTE!**

> 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.

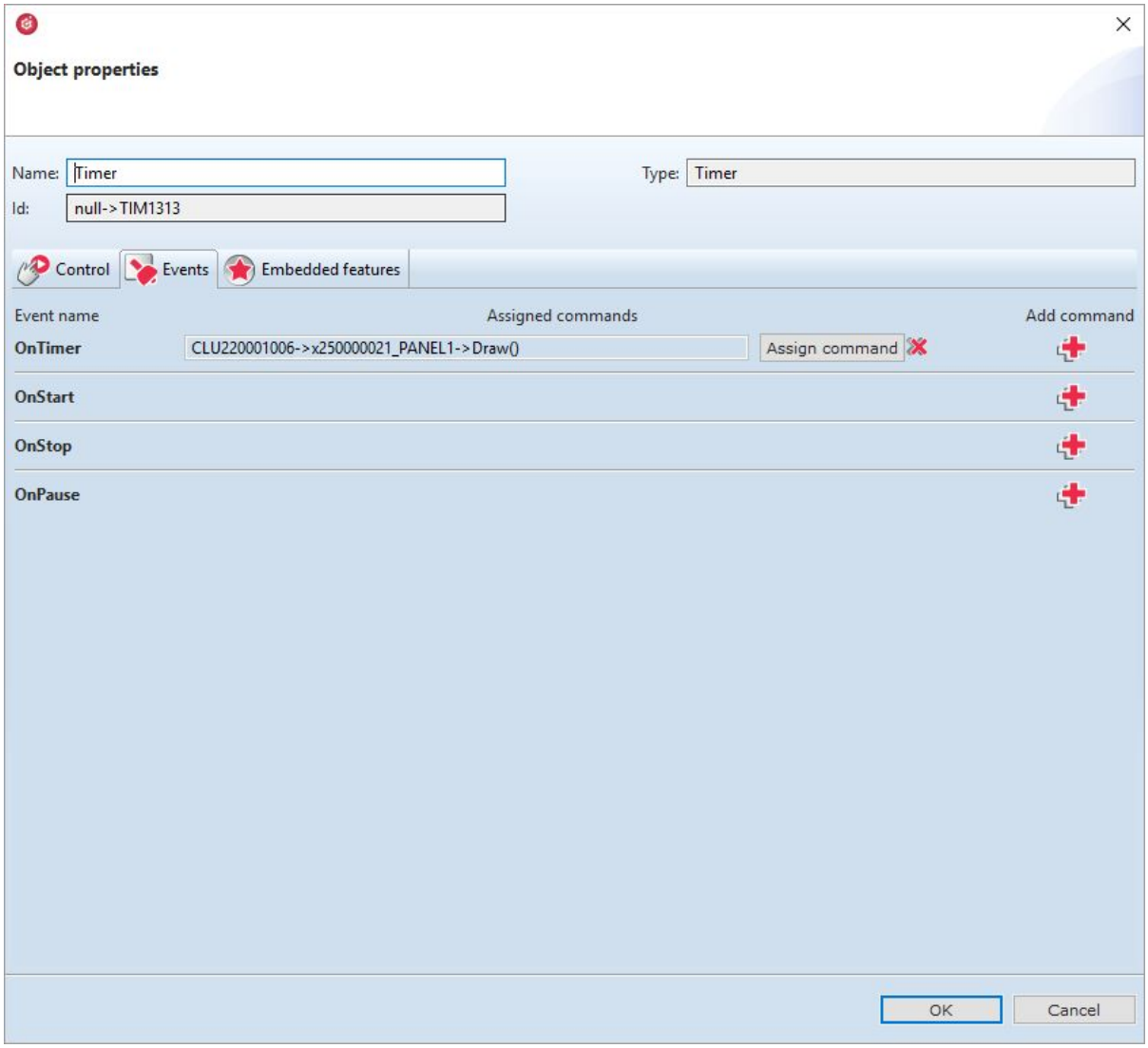
B. Set up the site as a clock

To configure the site as a clock:

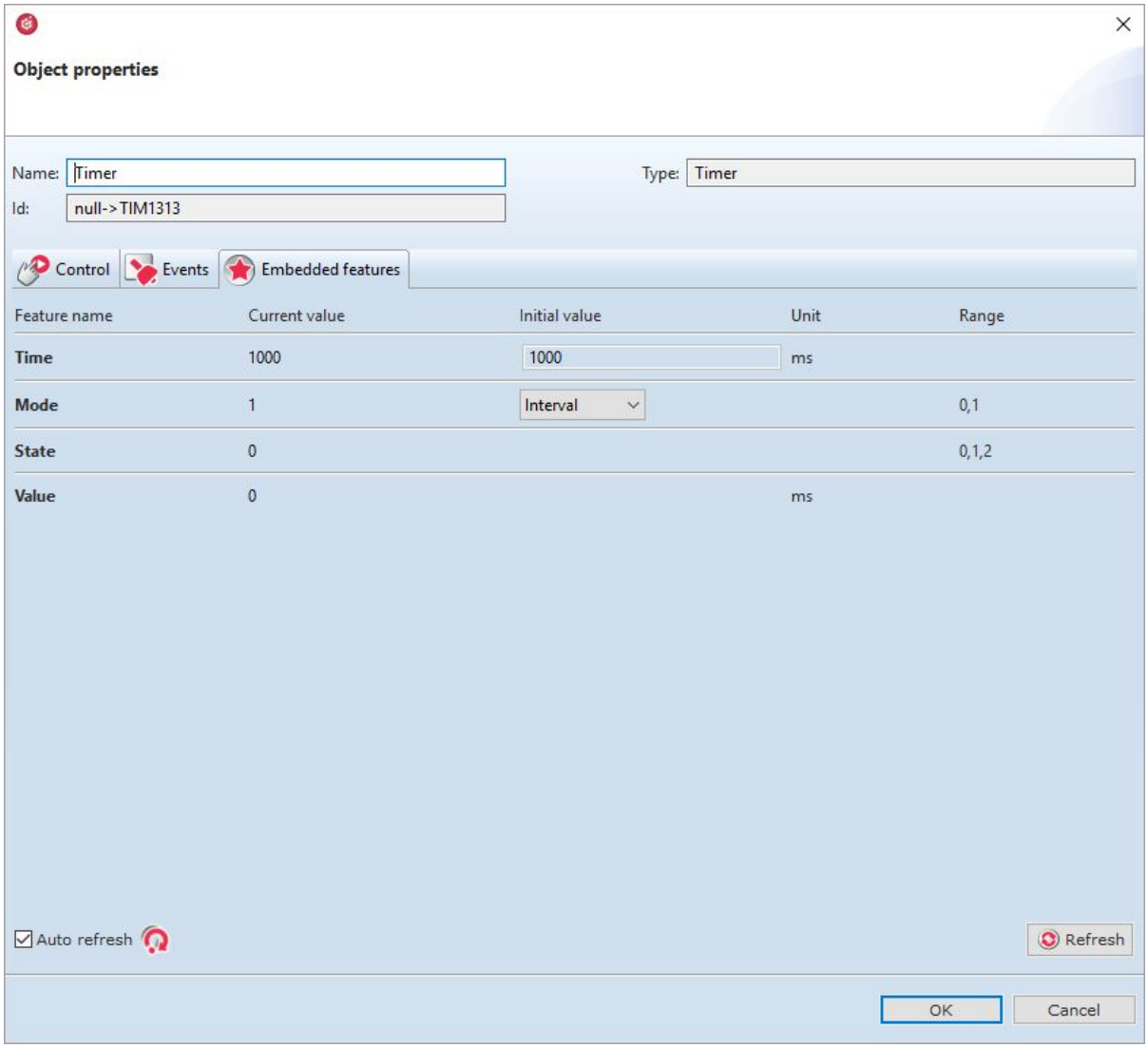
- Create a script displaying the current time (*Clock*);



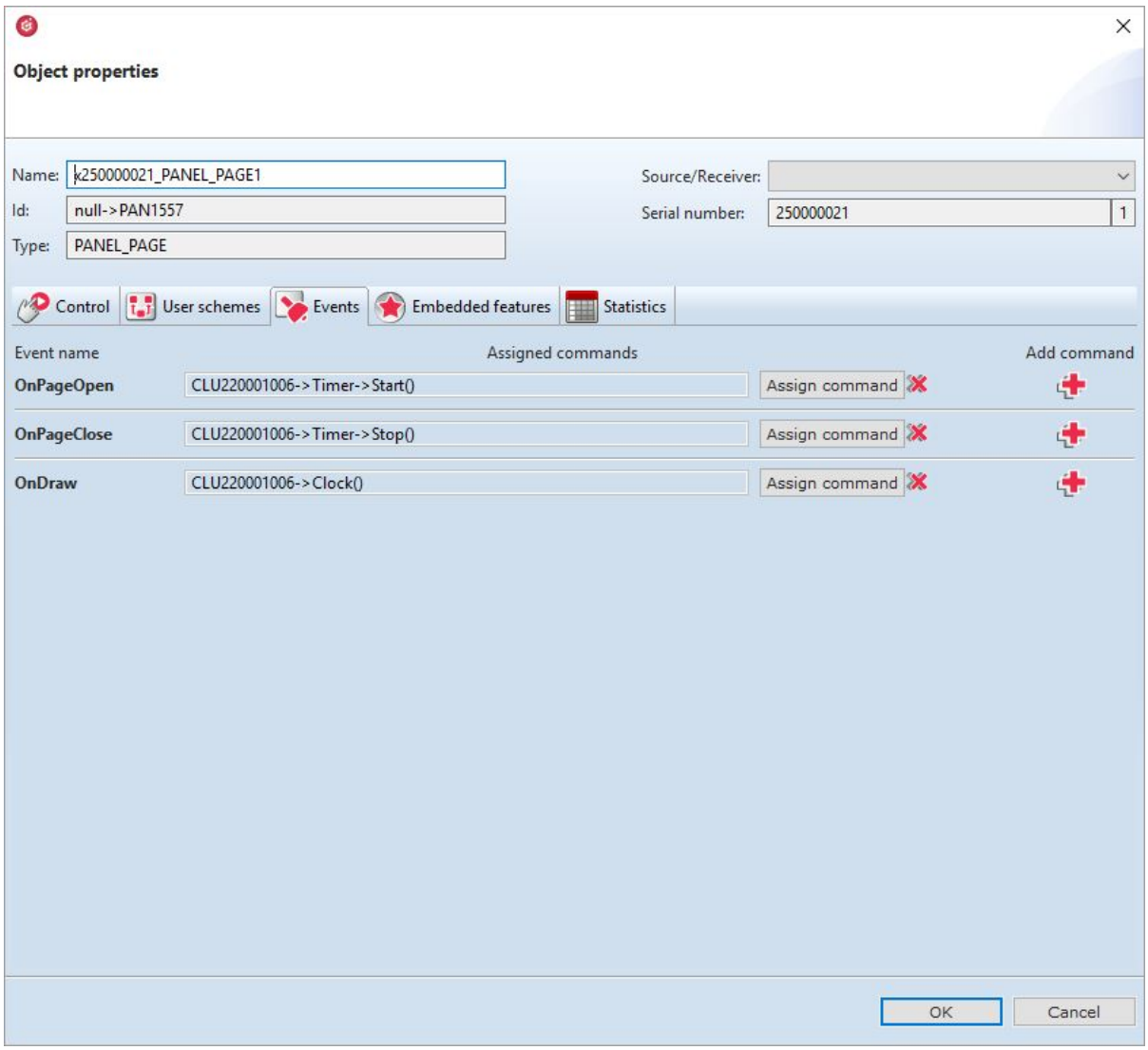
- - Create a virtual object Timer:
    - Go to the tab *Events*;
    - Configure the operation of the virtual object by assigning the `Draw` method of the *Panel* object to the `OnTimer` event:



- 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.

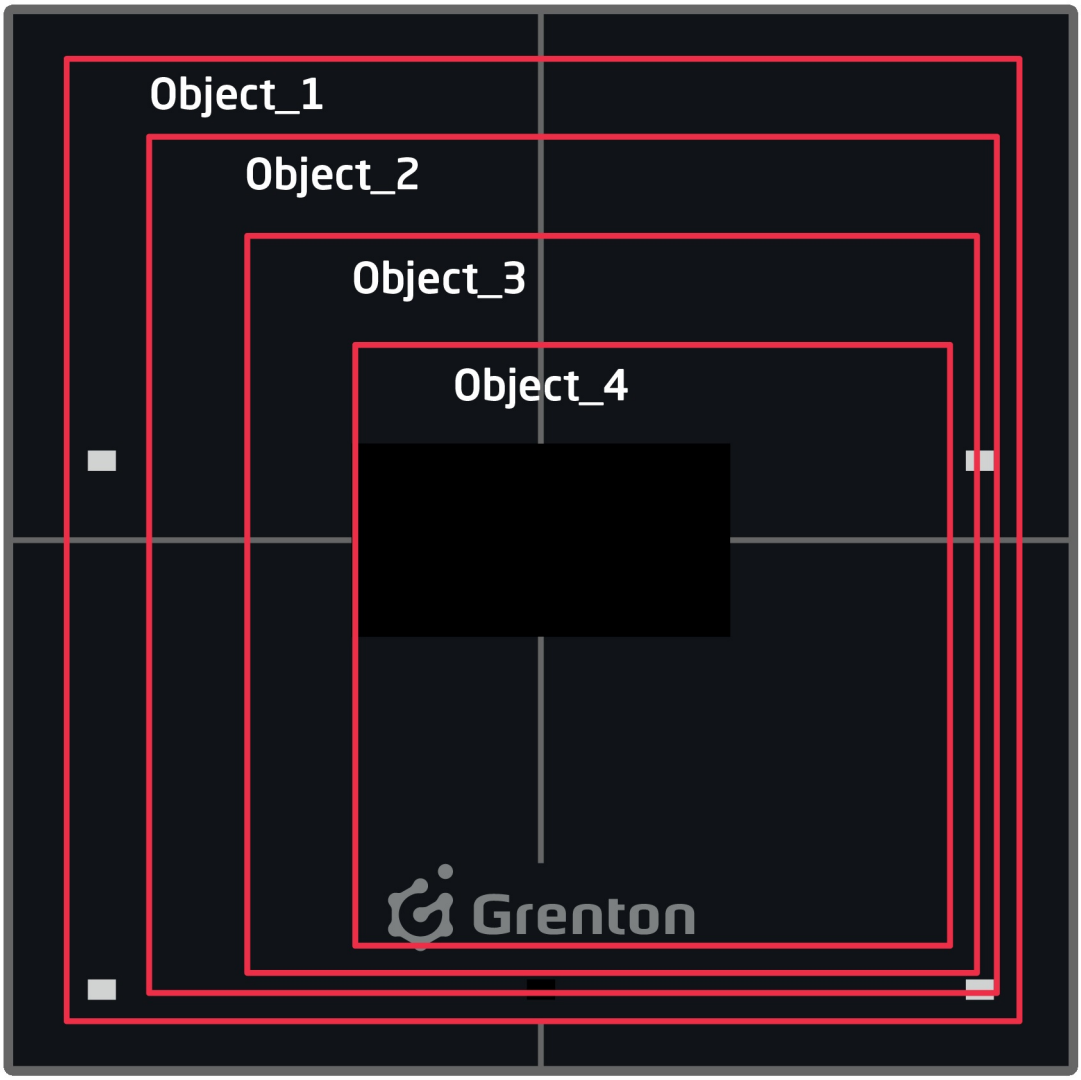
Script *Clock* in text version:

```
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:

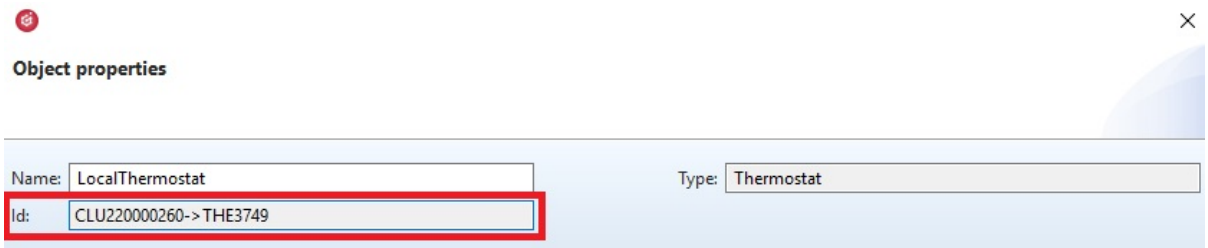
- `PointValue` - preset temperature, the ability to read the currently set temperature as well as the change to a new value;
- `Mode` - thermostat mode:

    - 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)`).

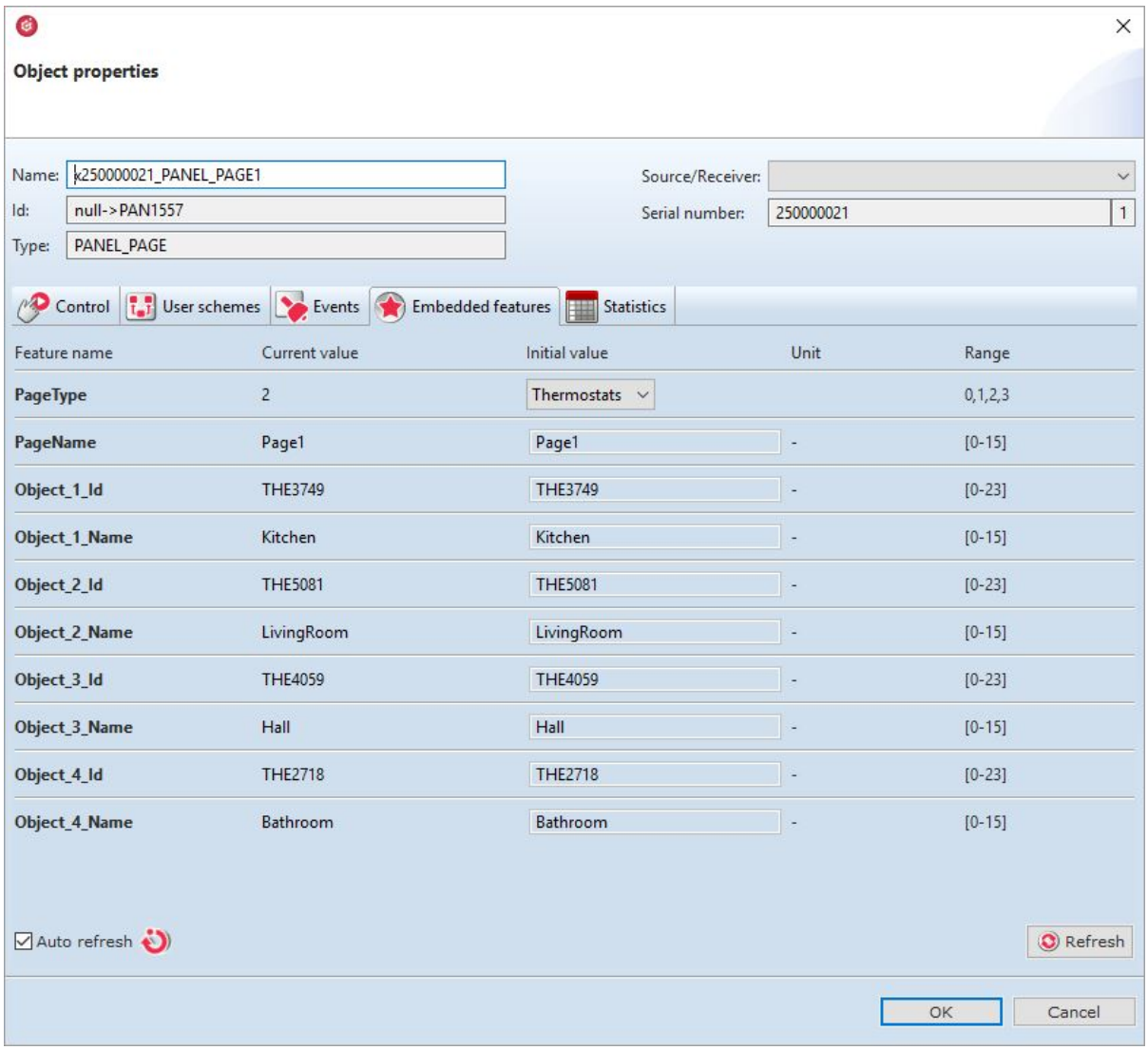
A. Creating a configuration with a local thermostat

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

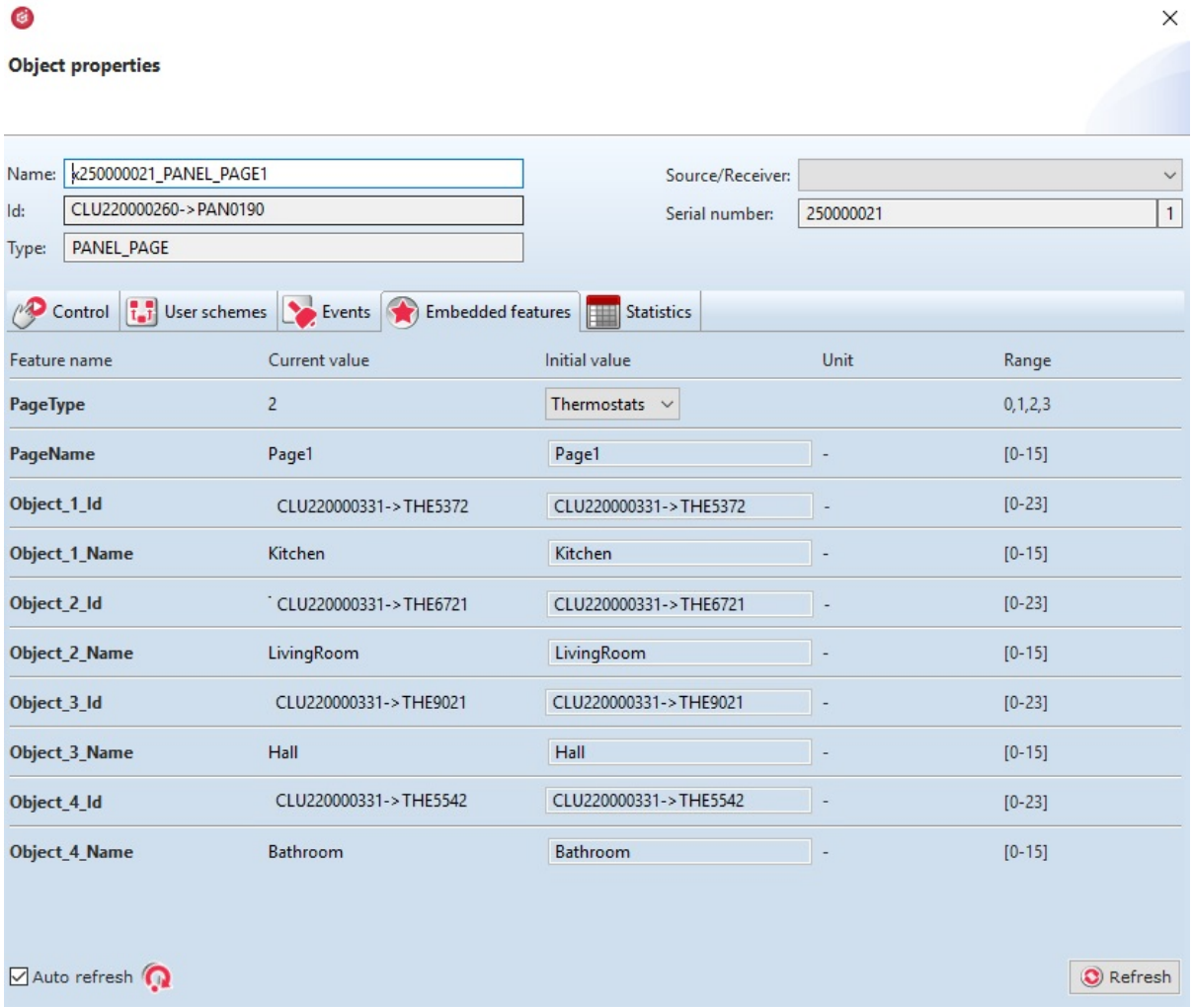
To create a configuration using a remote thermostat you should:

- 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:

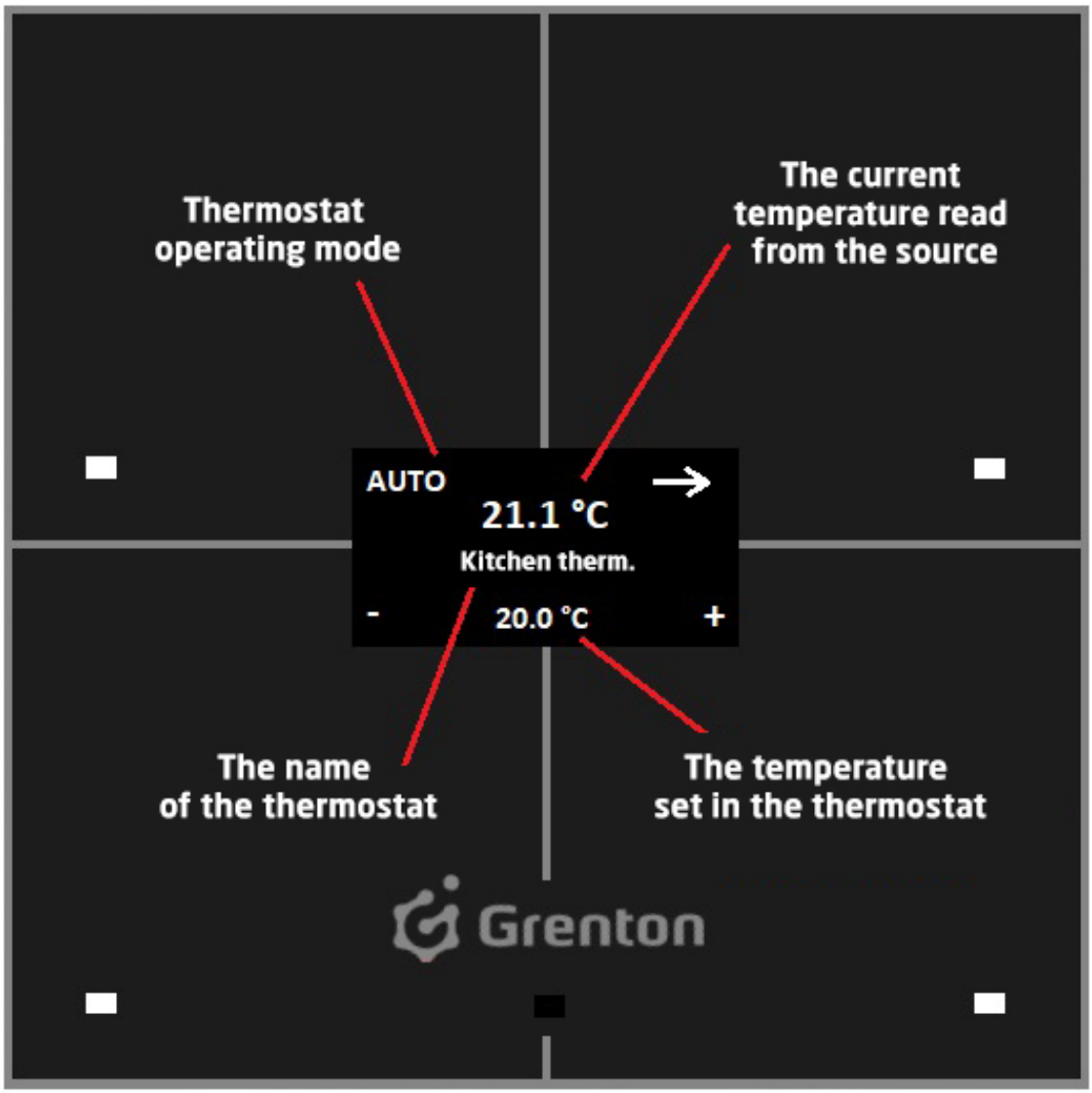
- `Object_X_Name` - name of the thermostat. The lack of a thermostat name in the parameter causes that the thermostat is not displayed;



> **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)	Zmiana stanu termostatu: 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)	Reducition 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_BUTTOX* objects (where X is the button number):
  - In the *Events tab* configure the operation of the button by assigning methods to specific events;
  - In the *Built-in features*, define objects displayed on the screen of a given button;
- Open object *PANEL_PAGEX* (where X is page number);
- Go to the tab *Events*;
- Set up the website by assigning methods to specific events;
- Go to the tab *Embedded features*;
- Set the `PageType` feature to *Buttons* or *FreeDraw*;
- Set the `Object_X_Id` features according to any version of the join:
  - Merging 2 objects into one horizontal button - the icon set for the button is displayed in the middle at the top of the screen (for objects `Object_1_Id` and `Object_2_Id`) or lower part of the screen (for objects `Object_3_Id` and `Object_4_Id`);
  - merging 2 objects into one button vertically - the icon set for the button is displayed in the middle on the left part of the screen (for objects `Object_1_Id` and `Object_3_Id`) or on the right part of the screen (for objects `Object_2_Id` and `Object_4_Id`);

- Merging 3 objects into one button - two identical icons are displayed, depending on how the objects are connected;
- Merging 4 objects into one button - the icon set for the button is displayed in the center of the screen

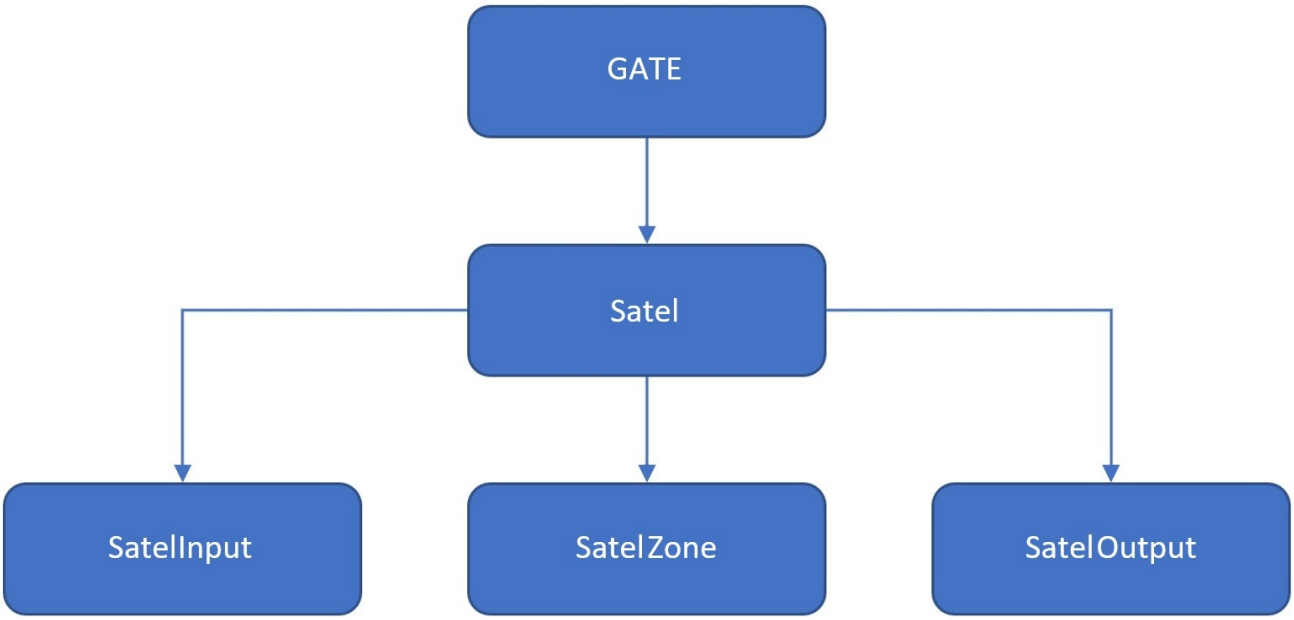
XIII. GATE Alarm Module

1. Integration with the Satel alarm control panel

1.1. General information

Integration of the Grenton system with the Satel alarm control panel is possible via the ETHM-1 module. You can create up to 64 virtual objects of the type: `*SatelZone*, *SatelInput*, *SatelOutput*. It is also possible to use the integration coding offered by Satel.

The configuration structure is as follows:



Virtual objects:

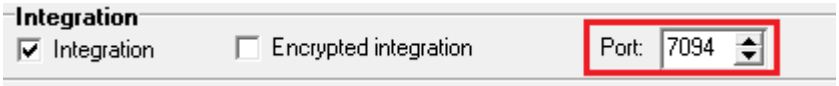
- **Satel** - allows you to perform a configuration that allows you to integrate the system with the Grenton alarm panel;
- **SatelZone** - allows to create a zone to which access will be possible after entering the password of one of the users or the password of the administrator himself;
- **SatelInput** - gives the ability to monitor the status of the selected input;
- **SatelOutput** - allows you to monitor and set the status of the selected output after entering the user or administrator password.

1.2. Configuration of 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:

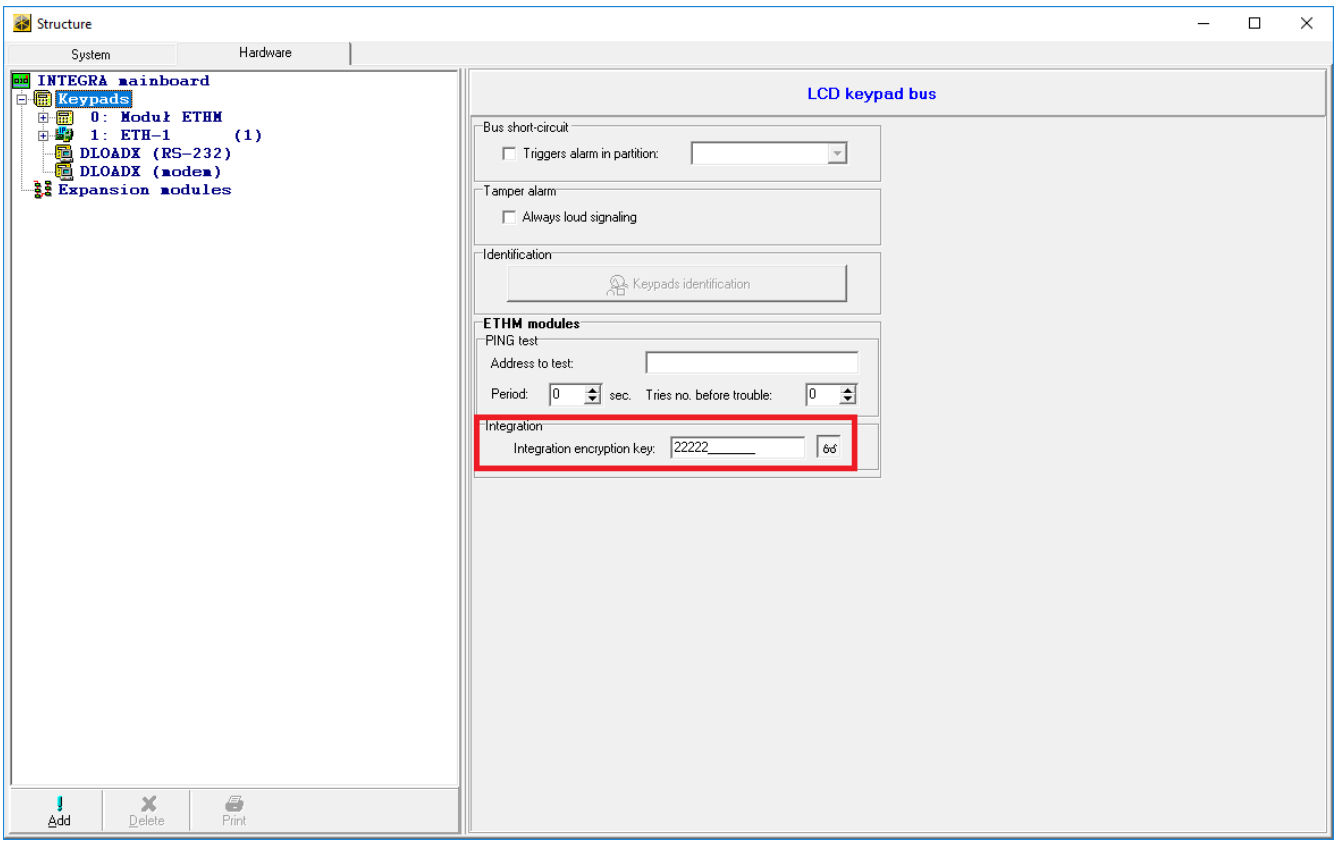
- IP address of the ETHM module (Satel) - available in the Satel configuration *(DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads -> ETHM-1 -> section Server IP address)*;
- ETHM integration port - available in the Satel configuration *(DLOADX -> Data -> Structure and Hardware -> 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)*.



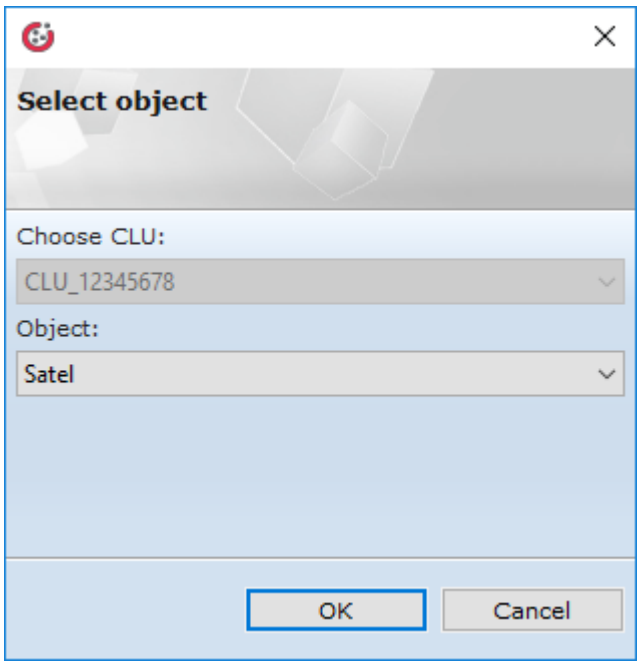
1.3. Virtual Objects

> **NOTE!** Before you start working with the GATE Alarm module, you must update the interface database!

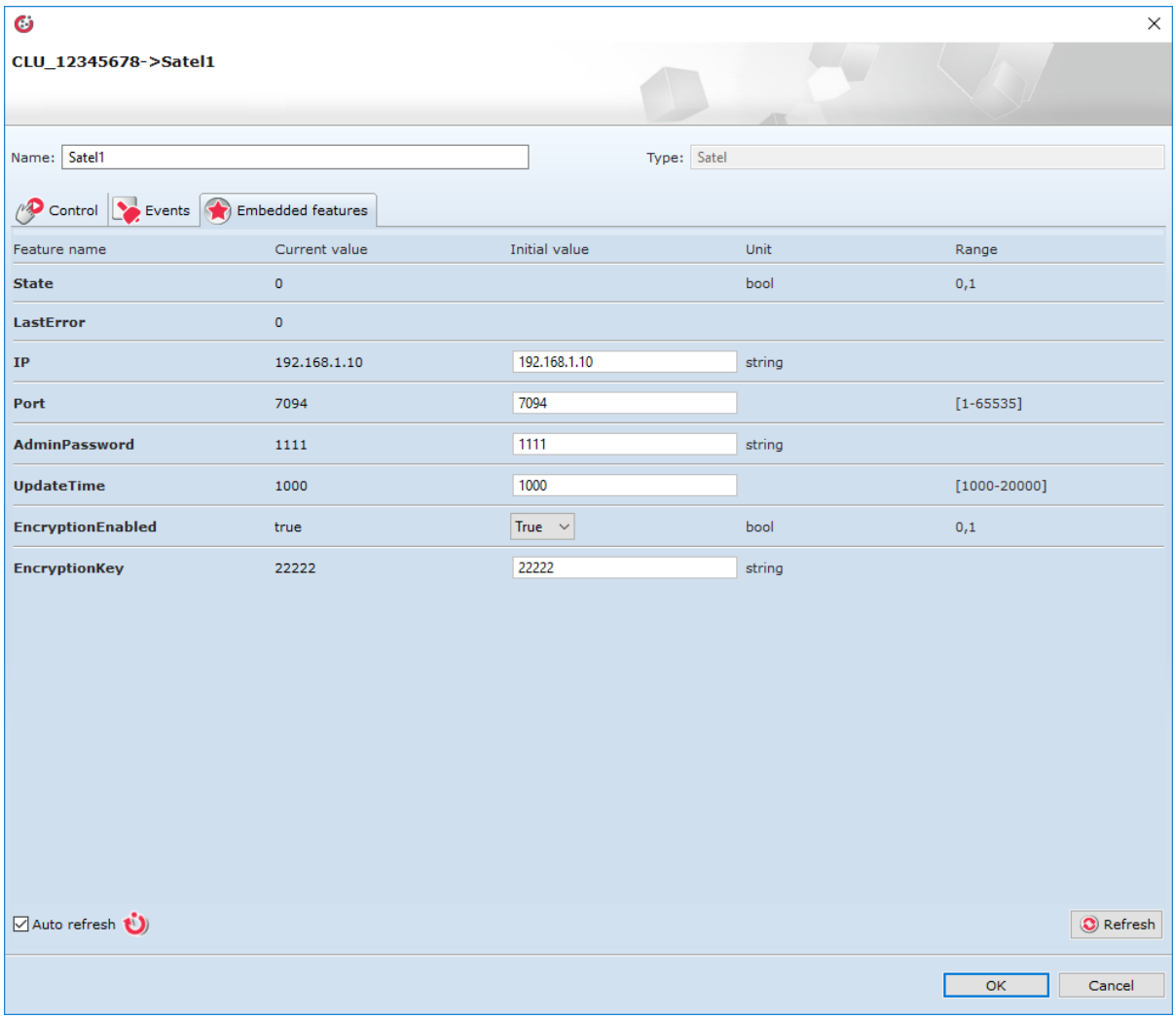
A. Satel

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

- Create a virtual object *Satel*:

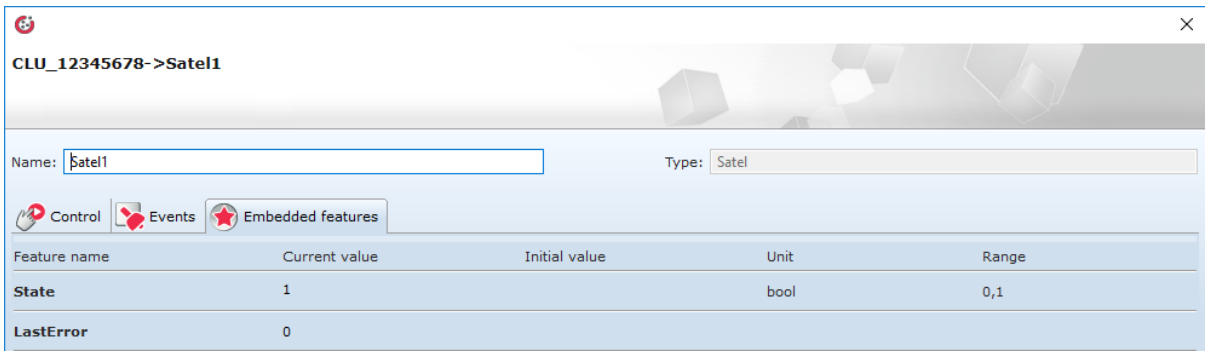


- Go to configuration - tab *Embedded features* and enter:
  - **IP** - IP address of the ETHM module (Satel);
  - **Port** - ETHM integration port;
  - **AdminPassword** - administrator password;
  - **EncryptionEnabled** - enable encoding - set if the integration on the ETHM module is marked with *Integration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

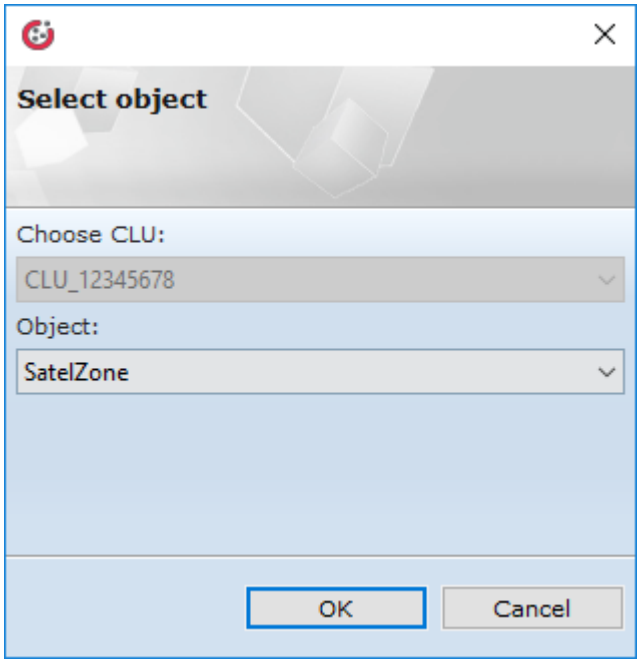
- Send configuration and verify connection - tab *Embedded features*, 'State' feature (1 - correctly connected to the control panel, 0 - no connection):



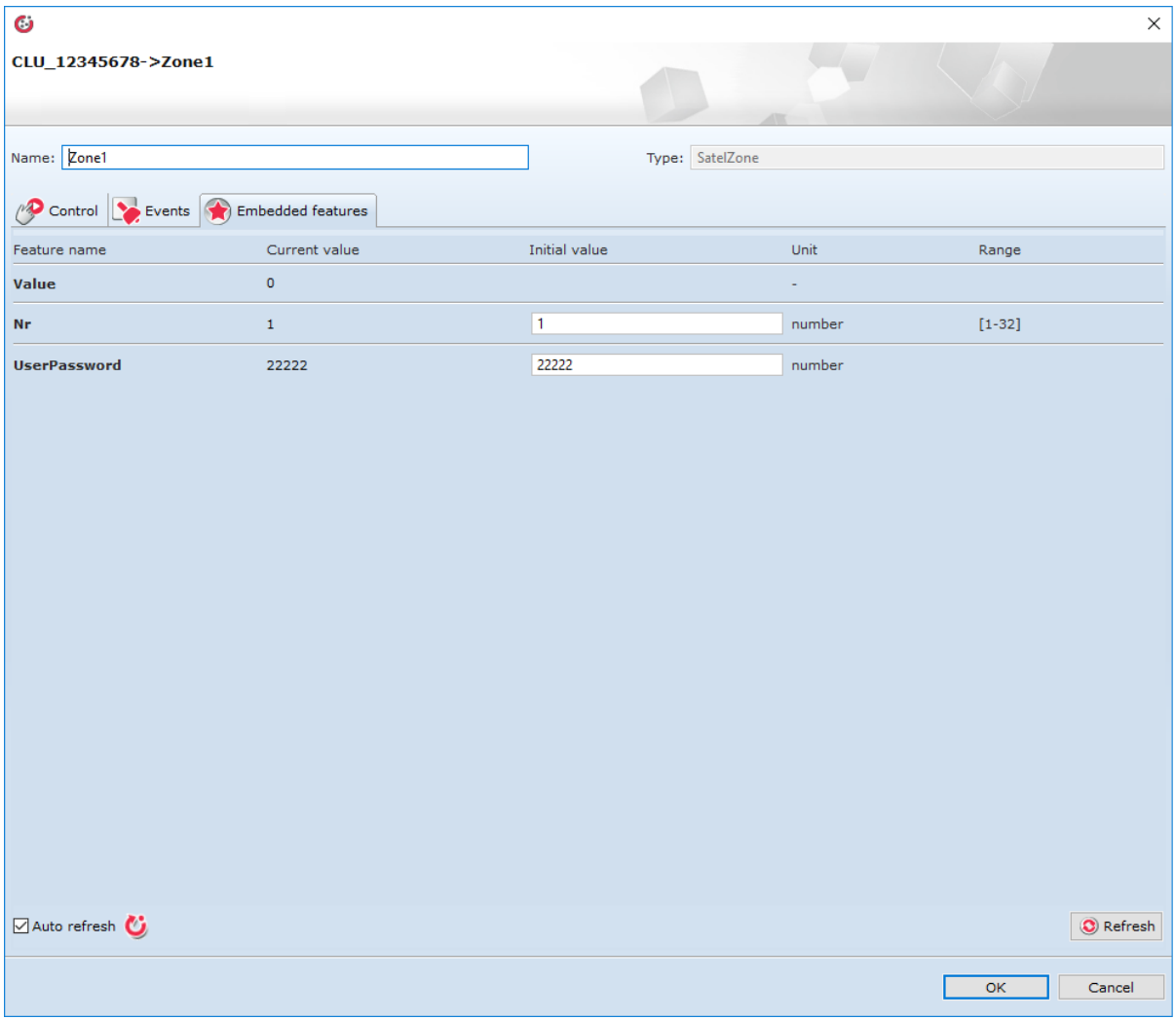
B. Zone

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:



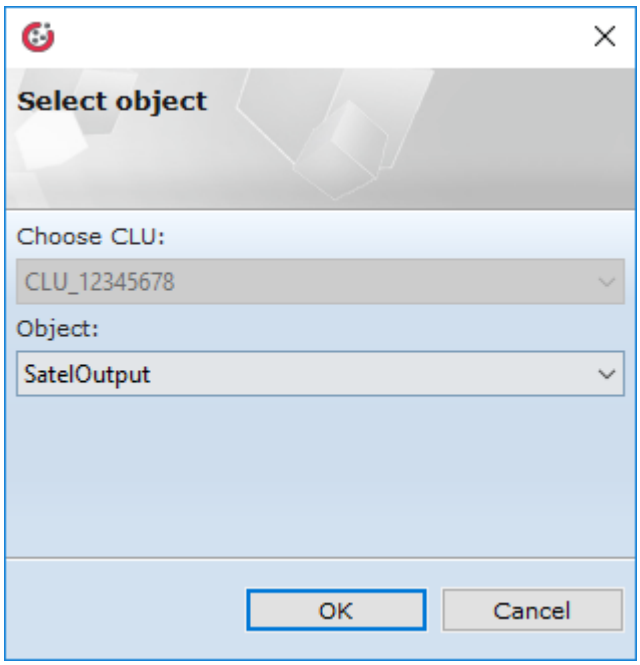
- Send configuration and verify the connection - tab *Built-in features*, the `value` feature (-1 means no connection to the control panel, the others mean a correct connection and the state of the zone is returned: 0 or 1);

- Arm / disarm the zone - the `ArmZone` and `DisarmZone` methods.

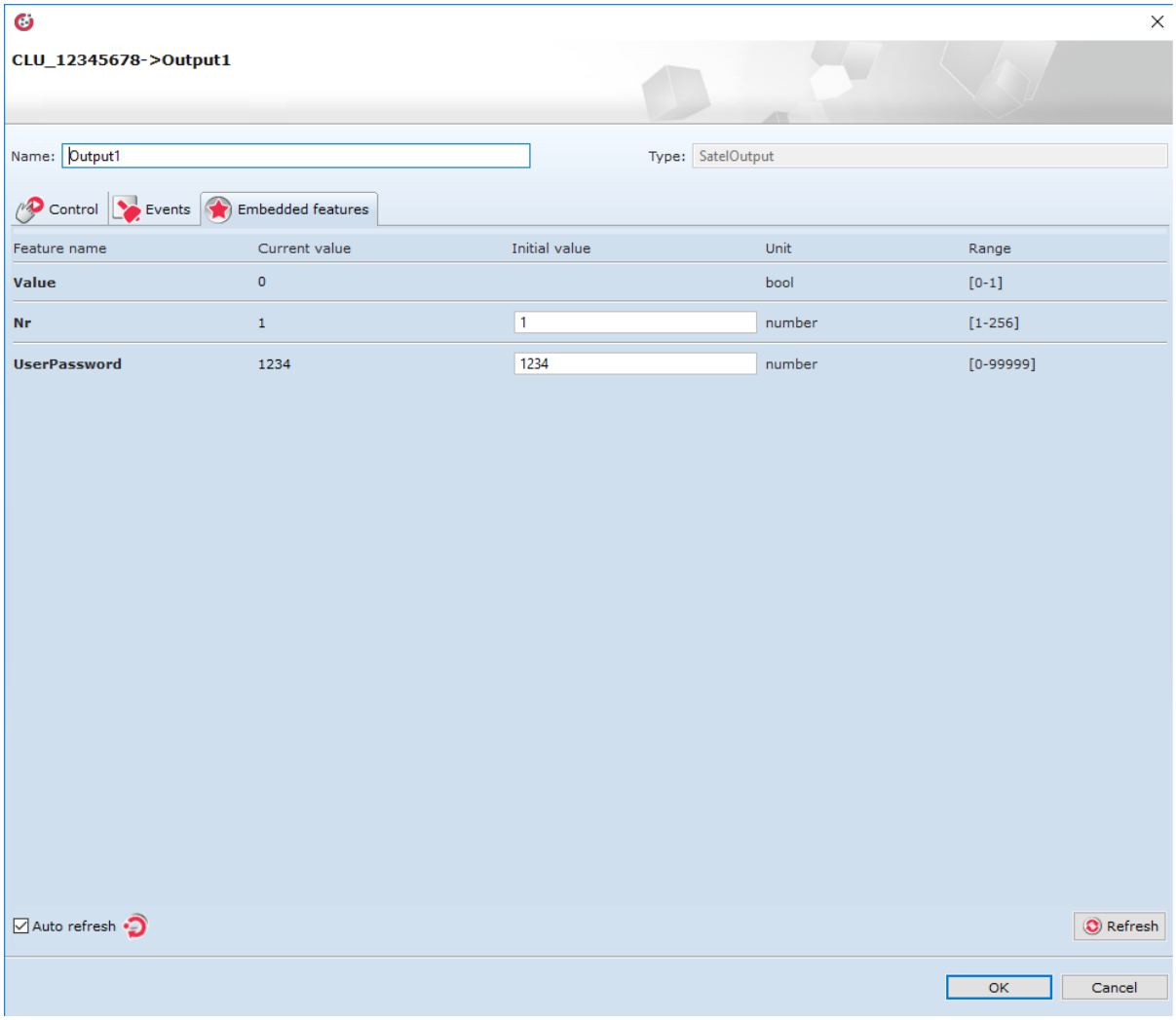
C. Output

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:

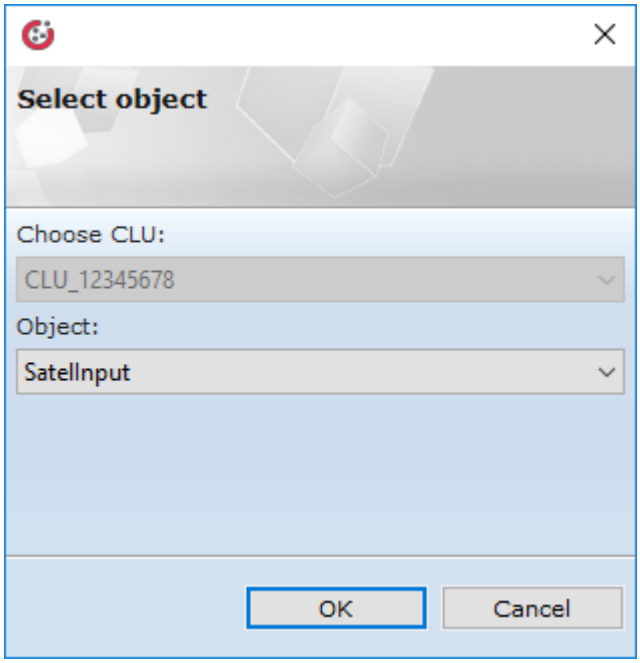


- Send configuration and verify the connection - tab *Embedded features*, the `Value` feature (-1 means no connection to the central unit, the others mean a correct connection and the status of the zone is returned: 0 or 1);
- Switch on / off the output - methods `SwitchOn` and `SwitchOff`.

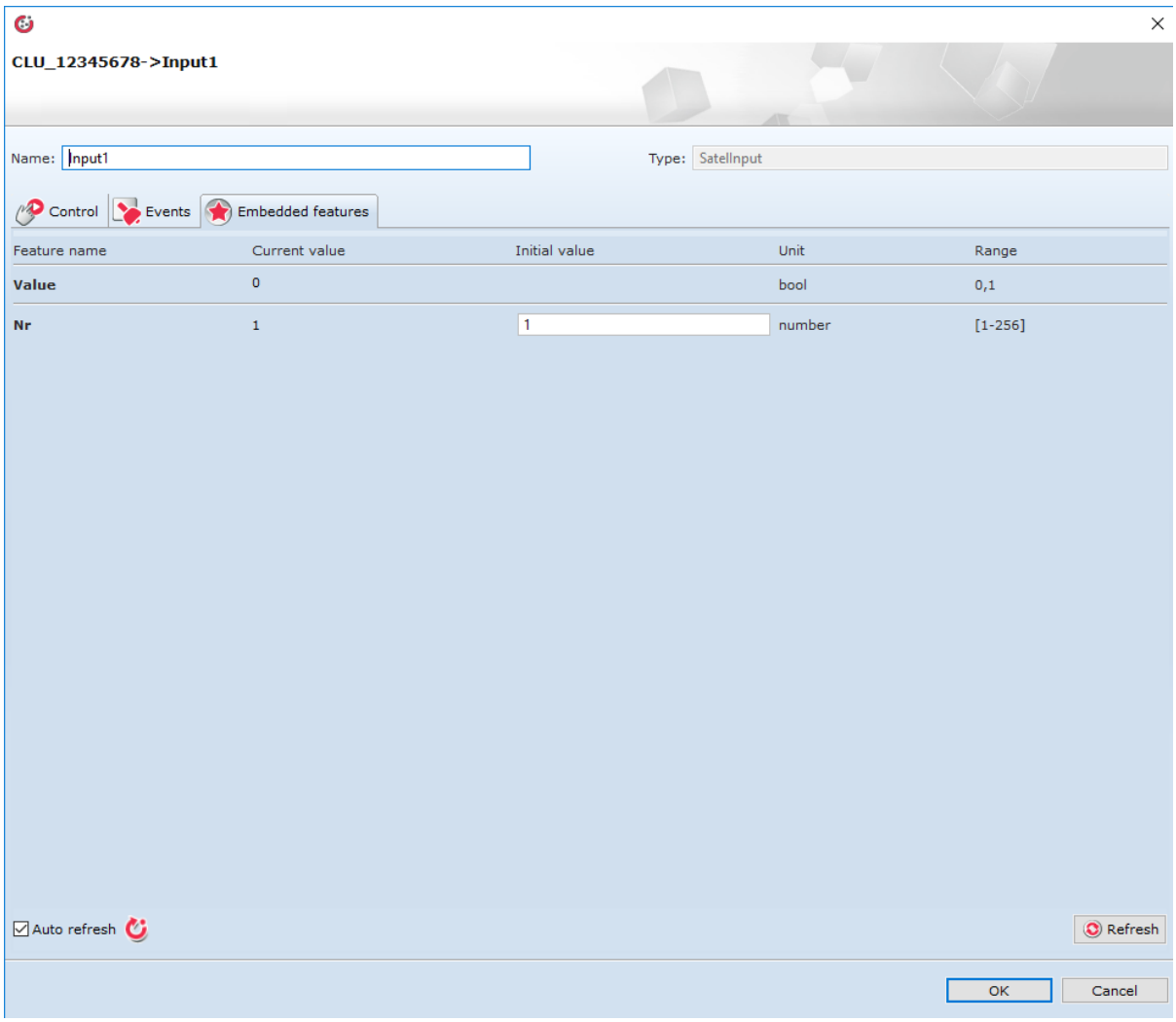
D. Input

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).

2. Restoring factory settings - *Hard Reset*

> **NOTE!** The procedure steps apply to the GATE Alarm module in the fw version. **18.23.4.1457** or lower

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 3 seconds - the correct reset will be confirmed by a 3-blink green LED.
- Release the *Reset* button after 3 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.

3. Configuration parameters

FEATURES

Name	Description
`Uptime`	Operation time of the device since the last reset (in seconds)
`UnixTime`	The current Unix time stamp
`FirmwareVersion`	Gate software version
`ClientReportInterval`	Reporting period about changes in features
`State`	Status of the central unit (0 - no connection to the central unit, 1 - connected to the central unit)
`LastError`	Last error code of the ETHM module (0 - ok, 1 - incorrect password)
`IP`	IP address of the ETHM module (Satel)
`Port`	ETHM module port (Satel)
`AdminPassword`	Satel administrator password
`UpdateTime`	The time of updating the status of the central unit
`EncryptionEnabled`	Encryption status (*true* - enabled, *false* - disabled)
`EncryptionKey`	Satel encryption key
`Value`	Returns the current status (1 - for armed zone, violated input, connected output, 0 - for disarmed zone, intact input, disabled output, -1 - no status information due to lack of connection)
`Nr`	Parameter defining the zone, entrance or exit to which the object refers
`UserPassword`	User's password (for "_" will use the administrator's password)

METHODS

Name	Description
`SetDateTime`	Sets the date and time
`SetClientReportInterval`	Sets the reporting period for feature changes
`SetUpdateTime`	Sets the update time of the central unit
`SetIP`	Sets the IP address of the ETHM module (Satel)
`SetPort`	Sets the ETHM module port (Satel)
`SetAdminPassword`	Sets the administrator password
`SetEncryptionEnabled`	Enables / disables encryption
`SetEncryptionKey`	Sets the Satel encryption key
`ArmZone`	Arm the zone
`DisarmZone`	Disarms the zone
`SetNr`	Sets the parameter defining which zone, entry or exit the object refers to
`SetUserPassword`	Sets the user's password (for "_" he will use the administrator's password)
`SwitchOn`	Switches output on
`SwitchOff`	Switches output off

EVENTS

Name	Description
`OnInit`	An event dispatched when the device initializes
`OnConnected`	The event is triggered after connecting to the alarm control panel
`OnDisconnected`	An event triggered after losing connection with the alarm control panel
`OnError`	Event triggered after an error occurred in the central unit (`LastError`)
`OnChange`	An event dispatched when the state changes (regardless of the value)
`OnSwitchOn`	An event triggered when an output is turned on or an input is violated
`OnSwitchOff`	An event triggered when the output is turned off or the normal state is set at the input
`OnArm`	Event triggered when arming the zone
`OnDisarm`	Event triggered when disarming the zone

XIV. GATE Modbus module

1. General information

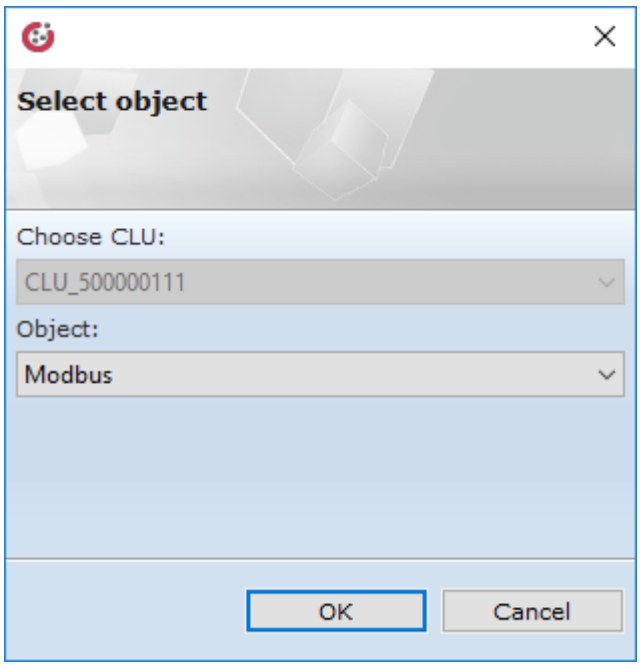
The GATE Modbus module enables the integration of the Grenton system with all devices supporting the MODBUS RTU standard. A maximum of 64 Modbus virtual objects can be created. Before starting the configuration one must obtain information about the selected Slave device supporting the MODBUS RTU standard - it will be necessary to know, among other things: device address, type and address of registers, as well as transmission speed.

2. Configuration of the GATE Modbus module

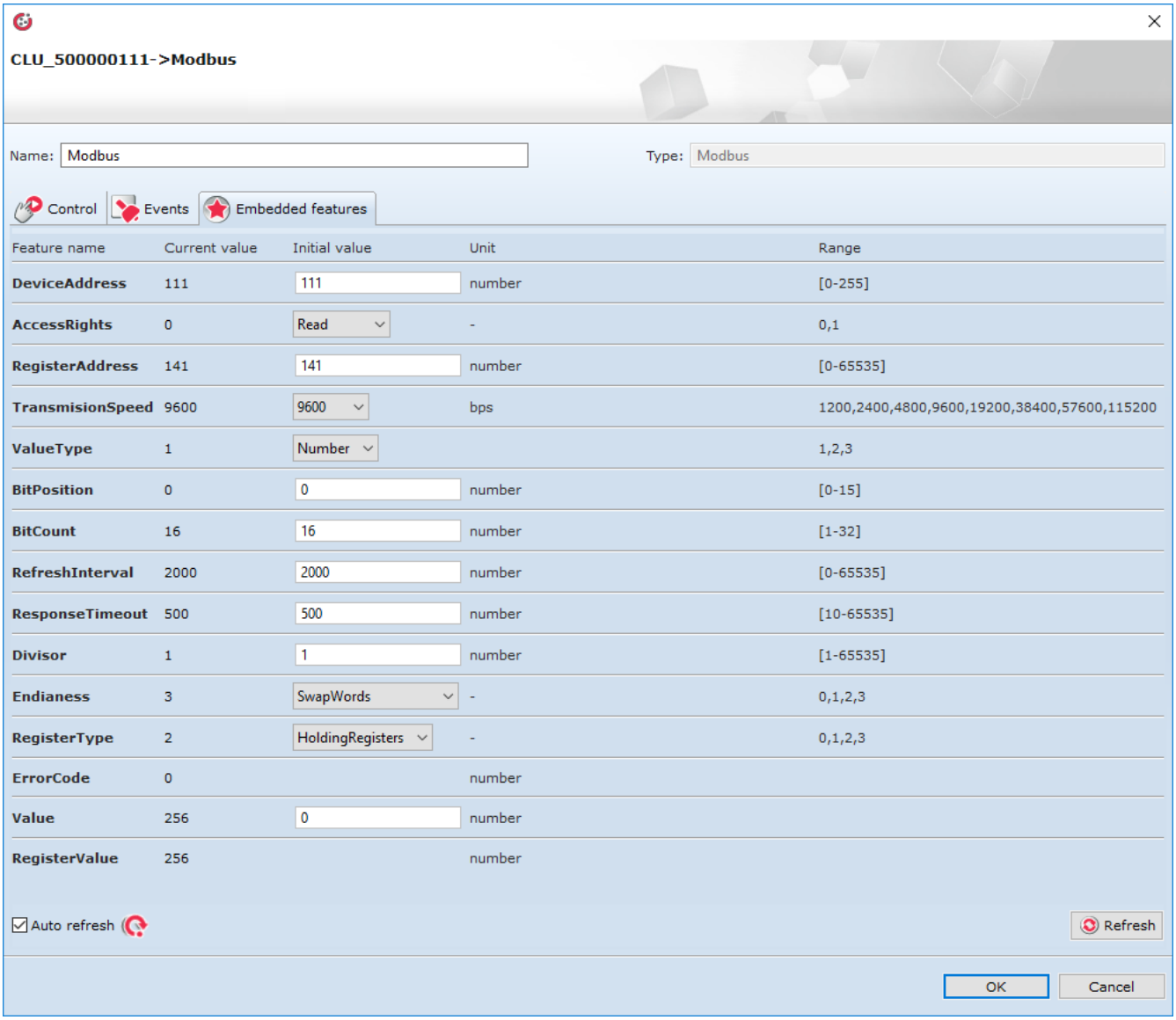
> **NOTE!** Before starting any work with the GATE Modbus module, it is necessary to update the interface database!

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

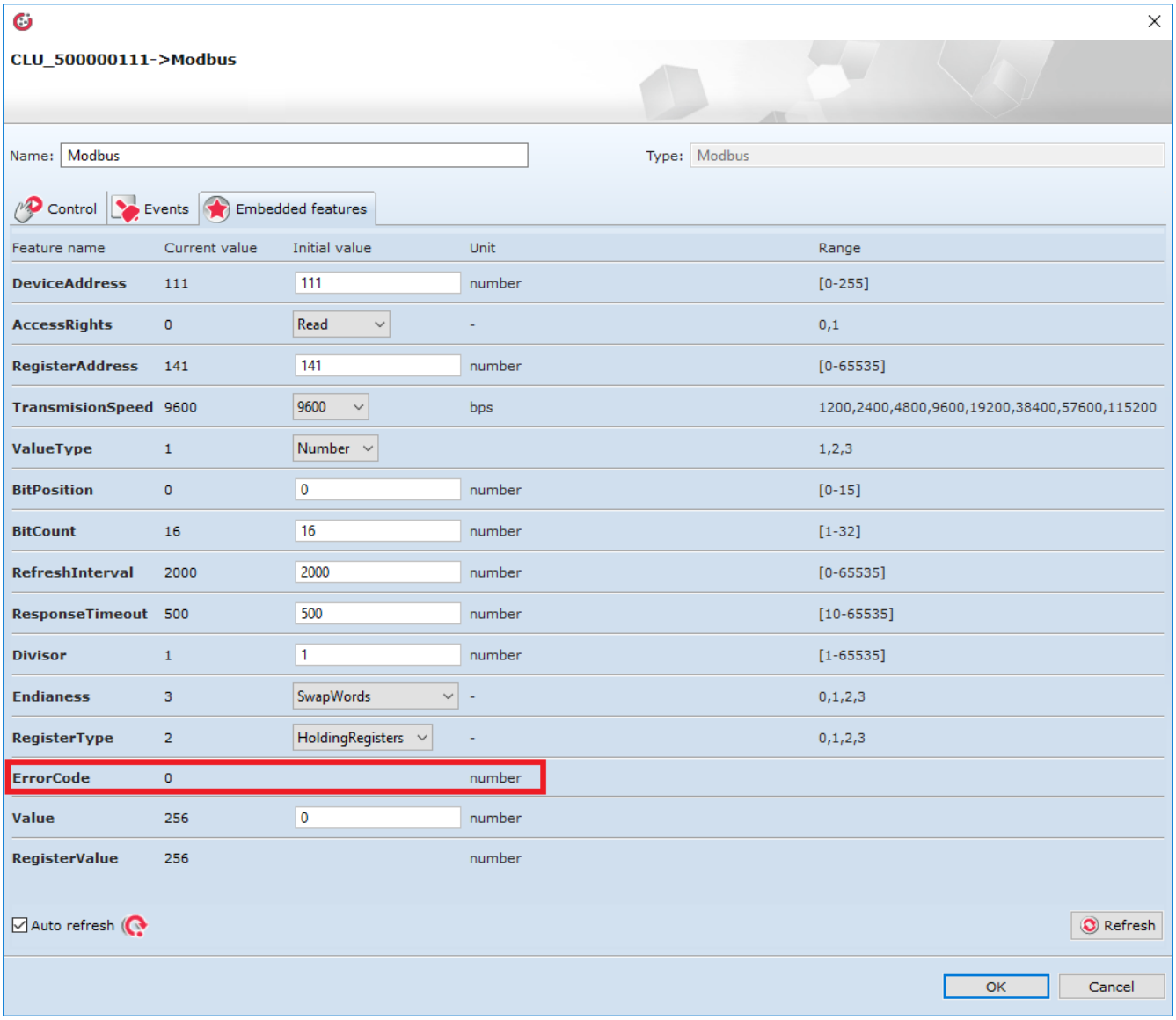
- Create a virtual object *Modbus*:



- Go to configuration - tab *Embedded features* and enter:
  - **DeviceAddress** - address of the slave device;
  - **AccessRights** - operating mode (*read* - reading the value from the register; *read / write* - allows saving the value to the set register);
  - **RegisterAddress** - address of the supported registry;
  - **TransmissionSpeed** - transmission speed;
  - **RefreshInterval** - the period of polling the slave device register by GATE Modbus;
  - **ResponseTimeout** - Slave device time for response (if it is exceeded, `ErrorCode` = **- 2** is returned);
  - **Divisor** - divisor (for `ValueType` = *number / float*);
  - parameters appropriate for the selected slave device type - look up XIV.3..



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



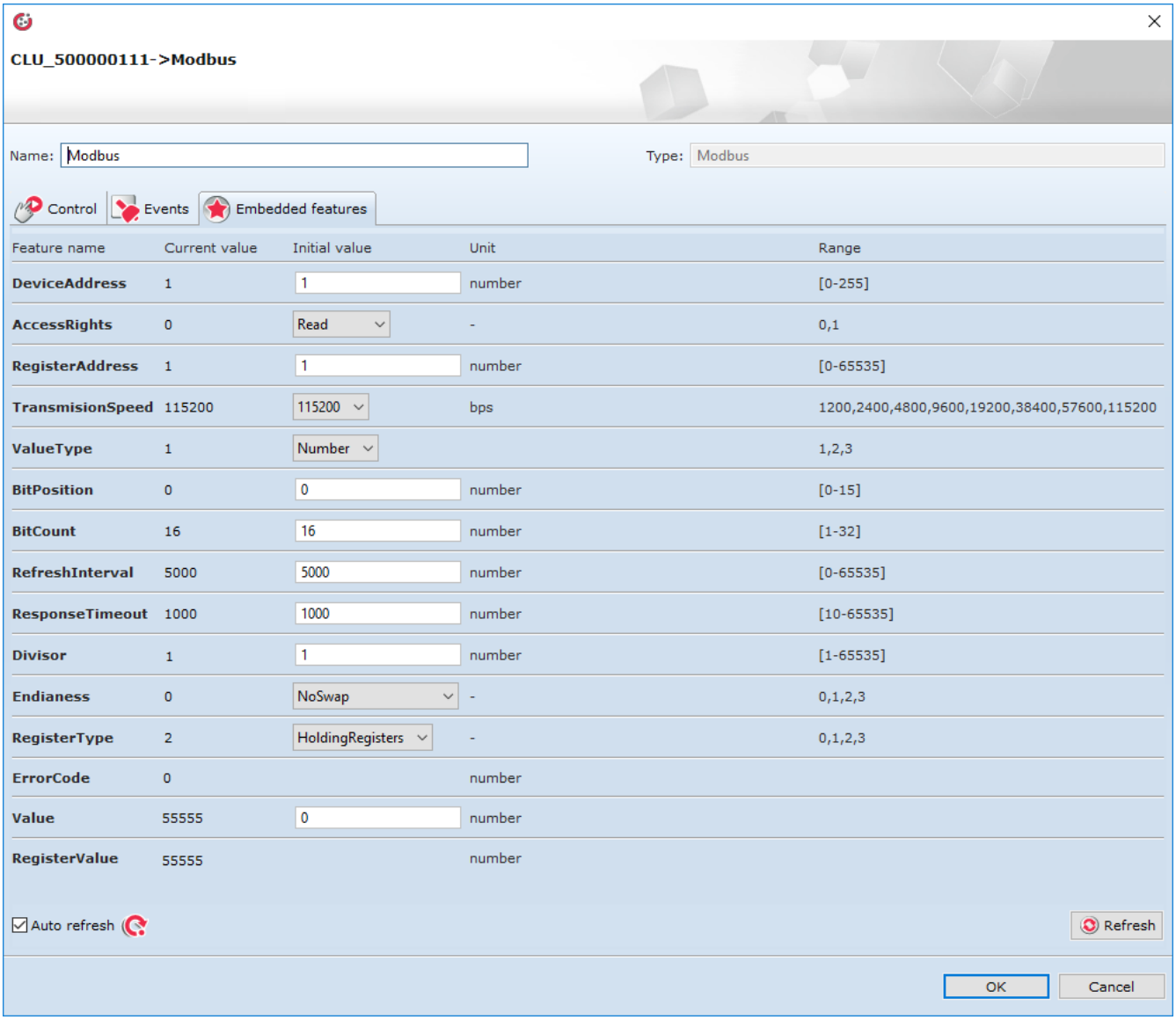
3. Parameters of registers

Depending on the type of slave register, the next available parameters must be set accordingly.

A. 16-bit registers

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

- `AccessRights`: read;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 16;
- `Endianness`: default value;
- `RegisterType`: Registers that remember:



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

- `AccessRights`: read;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 16;
- `Endianness`: default value;
- `RegisterType`: input registers

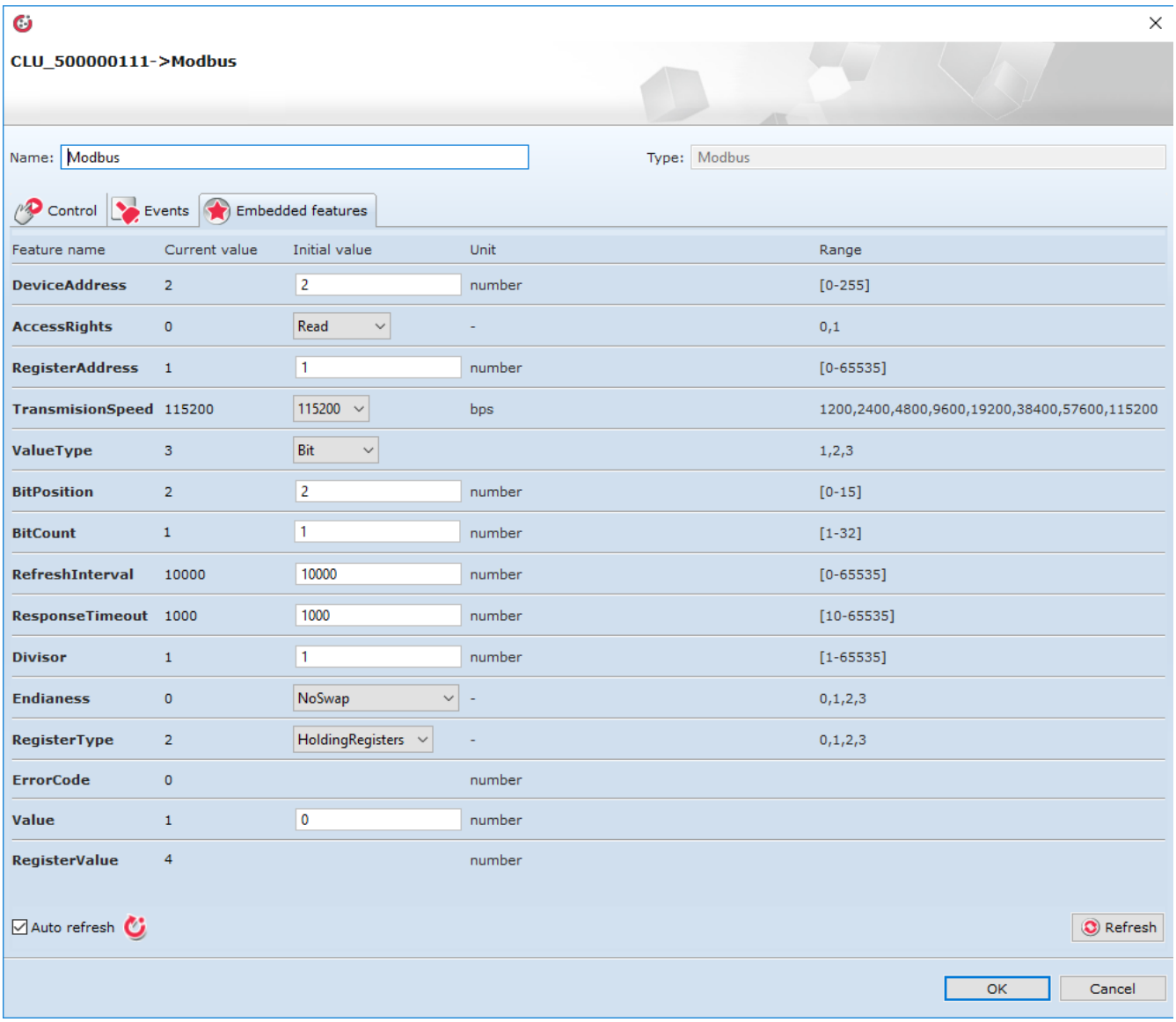
Records of 16-bit holding registers (`Preset / Write Single Holding Register`, *FunctionCode = 06*):

- `AccessRights`: read / write;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 16;
- `Endianness`: default value;
- `RegisterType`: remembering registers.

B. Fields in 16-bit registers

Reading of bit fields in a 16-bit remembering register (`Read Holding Registers`, *FunctionCode = 03*):

- `AccessRights`: read;
- `ValueType`: bit;
- `BitPosition`: 0-15 (position of the first interesting bit);
- `BitCount`: 1-16 (number of bits read sequentially);
- `Endianness`: default value;
- `RegisterType`: Registers that remember:



Reading of bit fields in a 16-bit input register (`Read Input Registers`, *FunctionCode = 04*):

- `AccessRights`: read;
- `ValueType`: bit;
- `BitPosition`: 0-15 (position of the first interesting bit);
- `BitCount`: 1-16 (number of bits read sequentially);

- `Endianness`: default value;
- `RegisterType`: input registers.

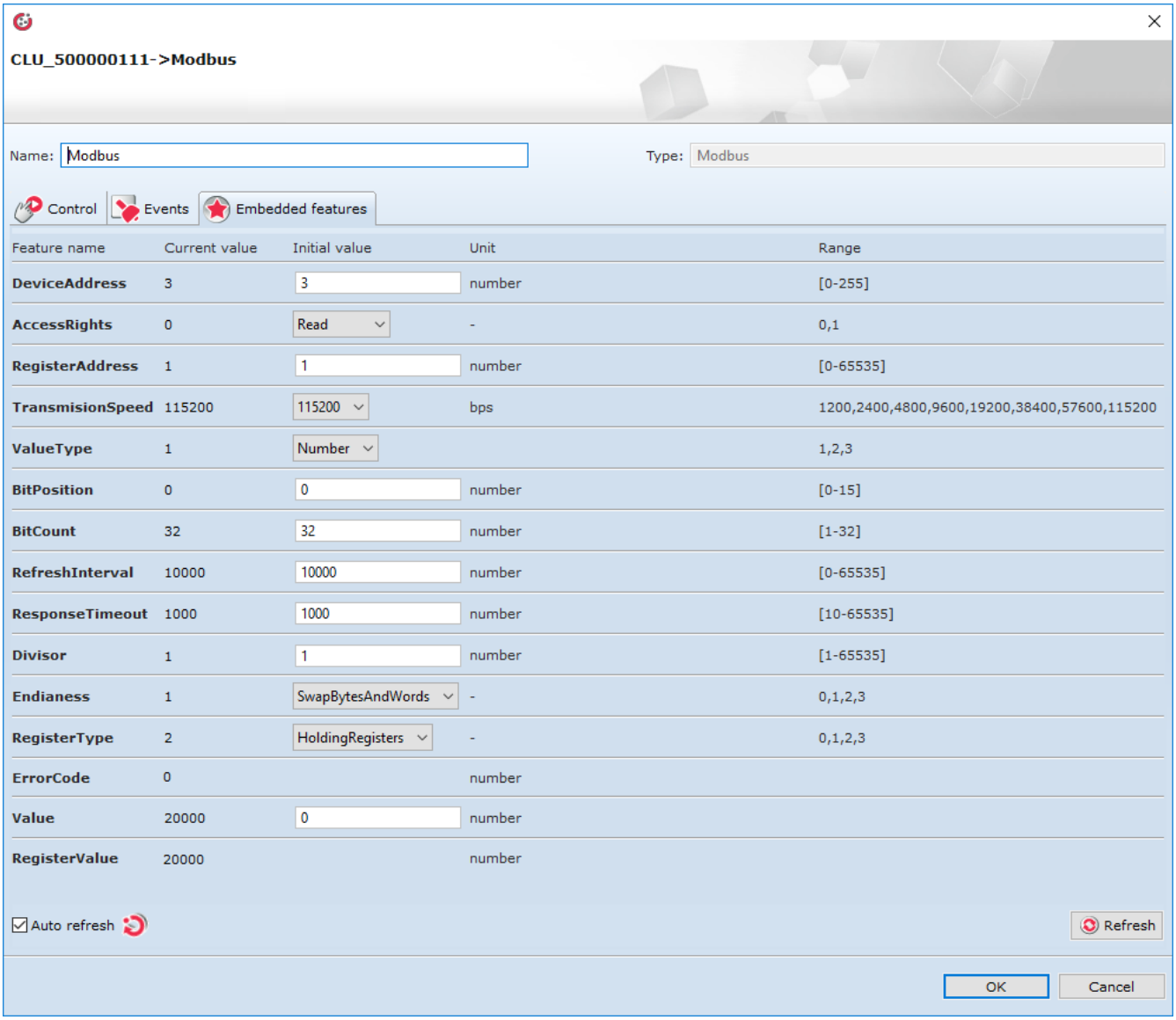
Writing bit fields in a 16-bit reminder register (`Preset / Write Single Holding Register`, *FunctionCode = 06*):

- `AccessRights`: read / write;
- `ValueType`: bit;
- `BitPosition`: 0-15 (position of the first interesting bit);
- `BitCount`: 1-16 (number of bits read sequentially);
- `Endianness`: default value;
- `RegisterType`: remembering registers.

C. 32-bit integer values of registers

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

- `AccessRights`: read;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 32;
- `Endianness`: in the case of 32-bit registers, slaves usually require reordering of bytes and words - *Swap bytes and words*; detailed information should be found in the slave device card;
- `RegisterType`: Registers that remember:



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

- `AccessRights`: read;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 32;
- `Endianness`: in the case of 32-bit registers, slaves usually require reordering of bytes and words - *Swap bytes and words*; detailed information should be found in the slave device card;
- `RegisterType`: input registers.

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

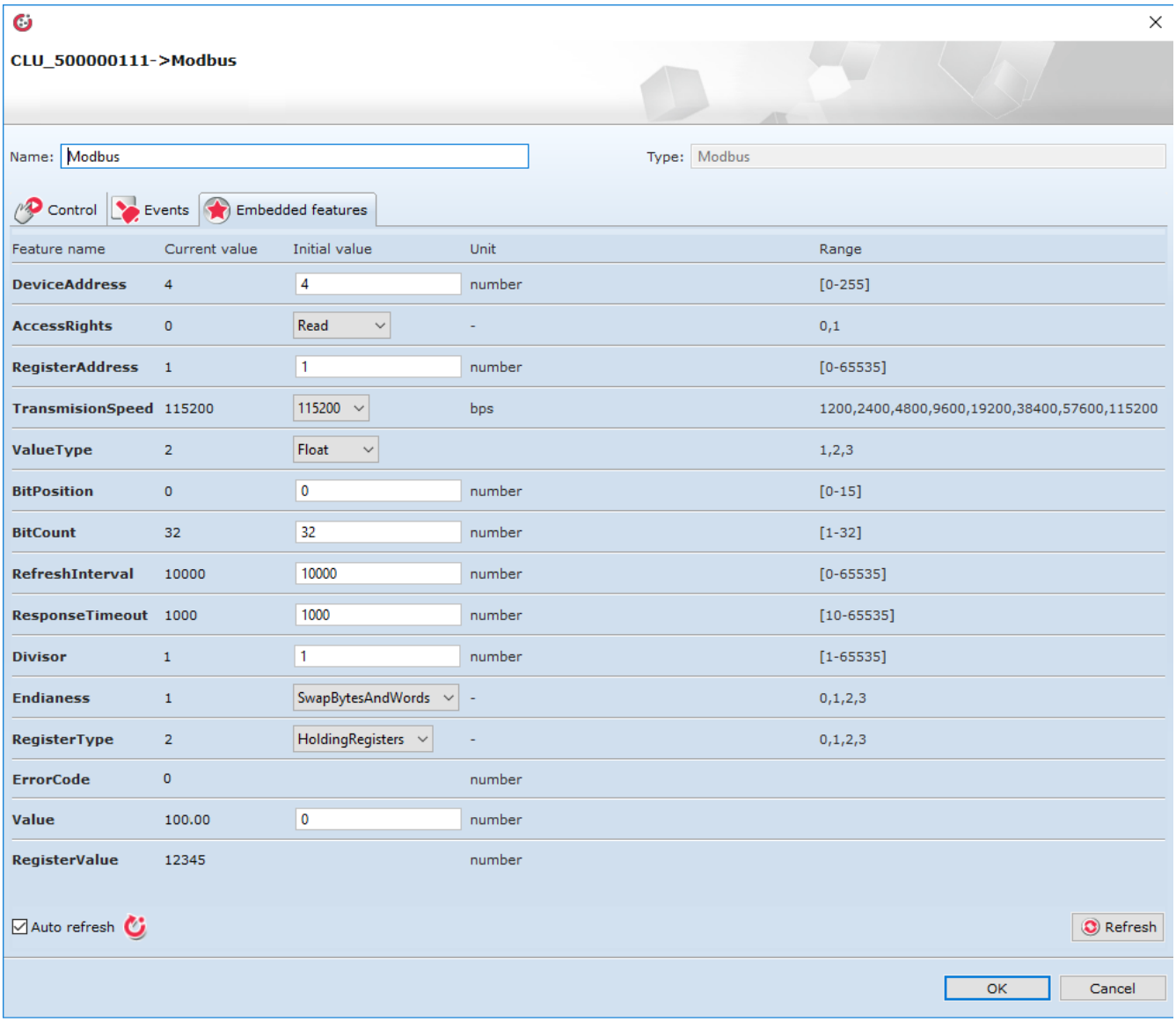
- `AccessRights`: read / write;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 32;

- `Endianness`: in the case of 32-bit registers, slaves usually require reordering of bytes and words - *Swap bytes and words*; detailed information should be found in the slave device card;
- `RegisterType`: remembering registers.

D. 32-bit floating point values of registers

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

- `AccessRights`: read;
- `ValueType`: float;
- `BitPosition`: default value;
- `BitCount`: 32;
- `Endianness`: in the case of 32-bit registers, slaves usually require reordering of bytes and words - *Swap bytes and words*; detailed information should be found in the slave device card;
- `RegisterType`: remembering registers



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

- `AccessRights`: read;
- `ValueType`: float;
- `BitPosition`: default value;
- `BitCount`: 32;
- `Endianness`: in the case of 32-bit registers, slaves usually require reordering of bytes and words - *Swap bytes and words*; detailed information should be found in the slave device card;
- `RegisterType`: input registers.

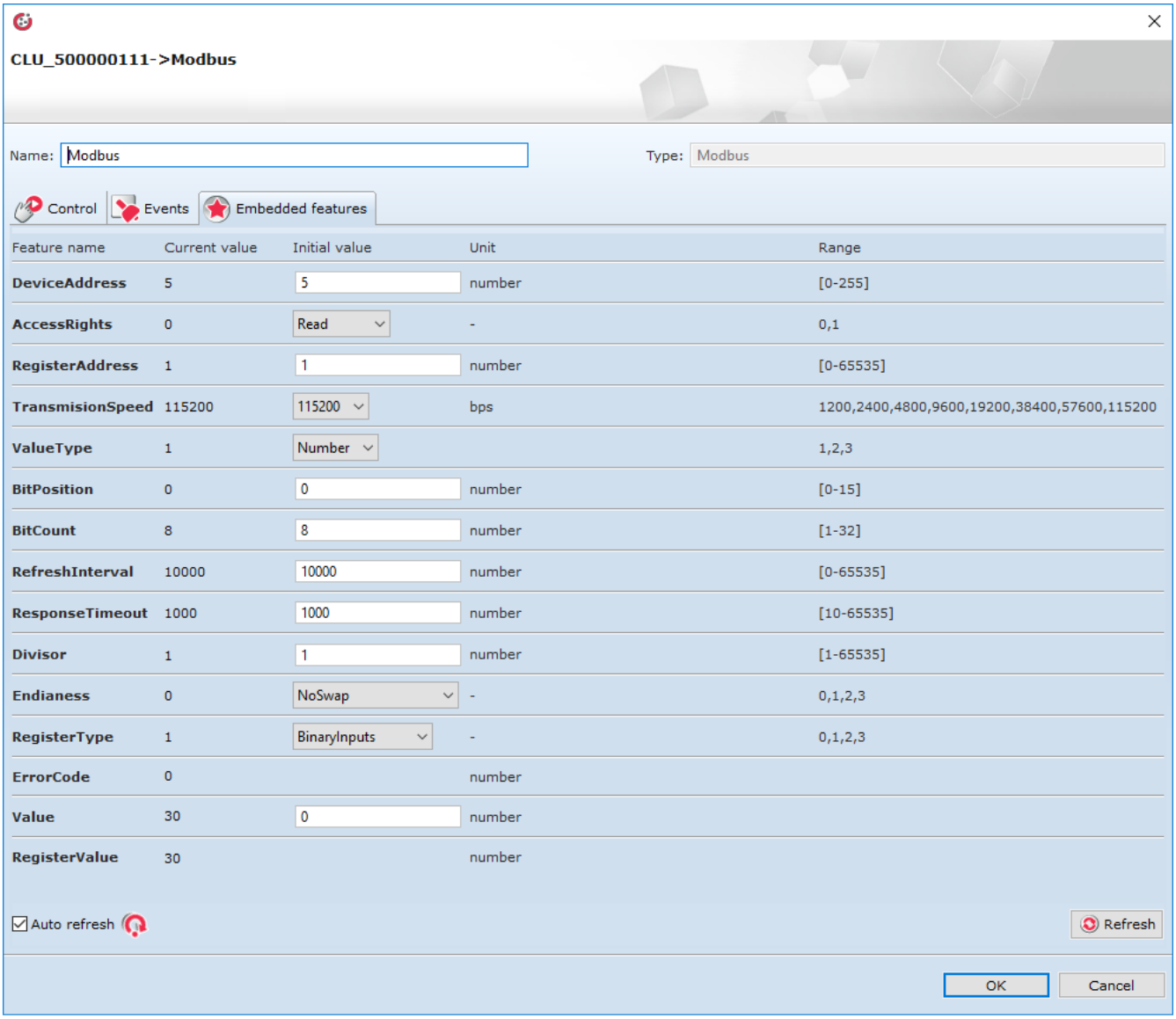
Record of 32-bit floating point values in the reminder register (`Preset / Write Multiple Holding Registers`, *FunctionCode* = *16*):

- `AccessRights`: read / write;
- `ValueType`: float;
- `BitPosition`: default value;
- `BitCount`: 32;
- `Endianness`: in the case of 32-bit registers, slaves usually require reordering of bytes and words - *Swap bytes and words*; detailed information should be found in the slave device card;
- `RegisterType`: remembering registers.

E. Discrete inputs / outputs

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

- `AccessRights`: read;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 1-32;
- `Endianness`: default value;
- `RegisterType`: bit outputs:



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

- `AccessRights`: read;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 1-32;
- `Endianness`: default value;
- `RegisterType`: two-state inputs.

Writing of discrete outputs / bit inputs (`Force / Write Single Coil`, *FunctionCode = 05*; `Force / Write Multiple Coils`, *FunctionCode = 15*):

- `AccessRights`: read / write;
- `ValueType`: number;
- `BitPosition`: default value;
- `BitCount`: 1-32;

- `Endianness`: default value;
- `RegisterType`: bit outputs.

4. Restoring factory settings - *Hard Reset*

> **NOTE!** The steps of the procedure apply to the GATE Modbus module fw. **18.21.4.1453** or lower

Running the *Hard Reset* function on the GATE Modbus module causes:

- Removal of the saved configuration;
- Formatting the flash memory partition;
- Removal of all created LUA objects;
- Loss of communication between OM / HM and Gate module.

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

- Disconnect power from the Gate module;
- Press and hold the *Reset* button on the module (the button is located under the bottom end of the module);
- Connect the power supply to the Gate module;
- Keep the *Reset* button pressed for at least 3 seconds - the correct reset will be confirmed by a 3-blink green LED.
- Release the *Reset* button after 3 seconds
- Wait about 60 seconds until the LED module - green and red - blink alternately (*Emergency* mode)

After the procedure the module will be cleared, but the module will no longer be visible (no response to *Keep-Alive*) in the project from the Object Manager level. To restore the module again, perform CLU Discovery and then send the configuration.

5. Configuration parameters

FEATURES

Name	Description
`Uptime`	Operation time of the device since the last reset (in seconds)
`UnixTime`	The current Unix time stamp
`FirmwareVersion`	Gate software version
`ClientReportInterval`	Reporting period about changes in features
`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 [7] : *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

METHODS

Name	Description
`SetDateTime`	Sets the date and time
`SetClientReportInterval`	Sets the reporting period for feature changes
`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
`SetResponseTimeout`	Sets the waiting time for a response
`SetDivisor`	Sets the divisor
`SetEndianness`	Sets the byte order type
`SetRegisterType`	Sets the Modbus register type
`SetValue`	Sets the read / write value

EVENTS

Name	Description
`OnInit`	An event dispatched when the device initializes
`OnChange`	An event dispatched when the state changes (regardless of the value)
`OnError`	An event dispatched when the slave device reports an error

XV. GATE HTTP Module

1. General information

The GATE Http module is a device enabling system integration with external sites using the HTTP protocol, as well as a wide group of devices and external / third-party systems - eg AV devices with HTTP interfaces.

2. Configuration of the HTTP GATE module

> **NOTE!** Before starting any work with the GATE HTTP module, the interface database update is required!

2.1 Virtual objects

2.1.1. HTTP Request

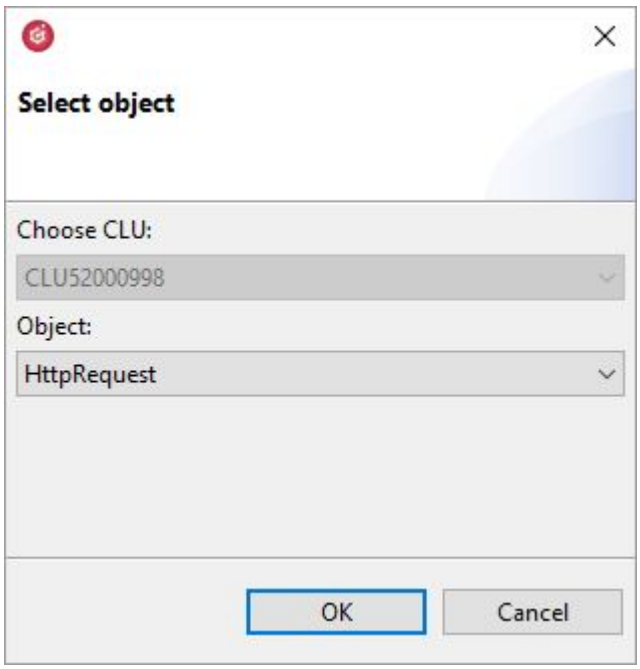
For the HttpRequest, for example, the weather service http://api.openweathermap.org is used

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

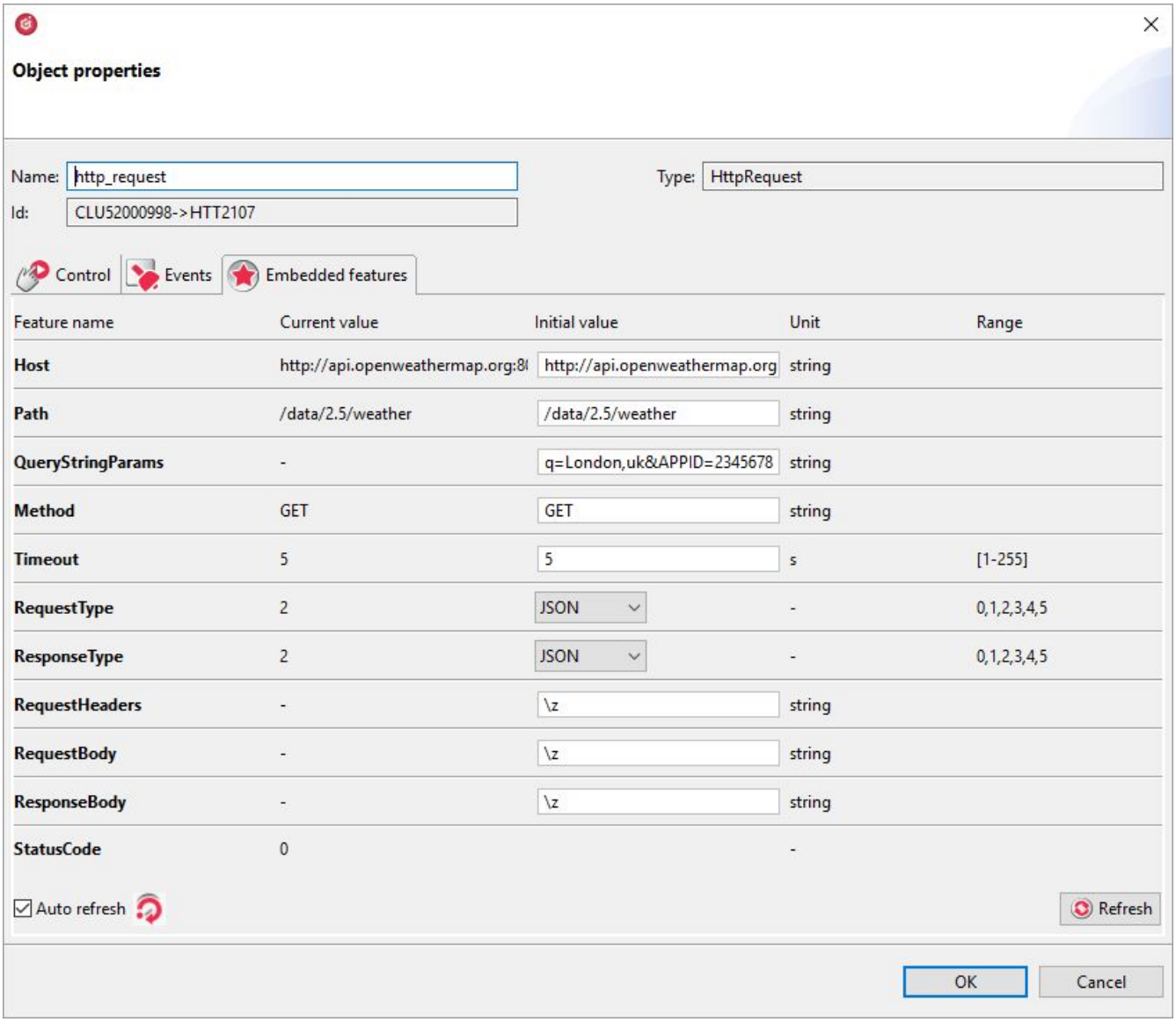
API call: http://api.openweathermap.org/data/2.5/weather?q=London&APPID={APIKEY}

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!

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":295.37},
"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,"sunset":143565087
0},
"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":295.37},
```

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":295.37},
```

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 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

```
<current>
 <city id="2643741" name="City of London">
    <coord lon="-0.09" lat="51.51">
    <country>GB</country>
    <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
  </city>
```

In a script:

```
local nameTag = resp[1][2].xmlTag
```

After calling the script, the value of "country" will be assigned to the local variable (script variable).

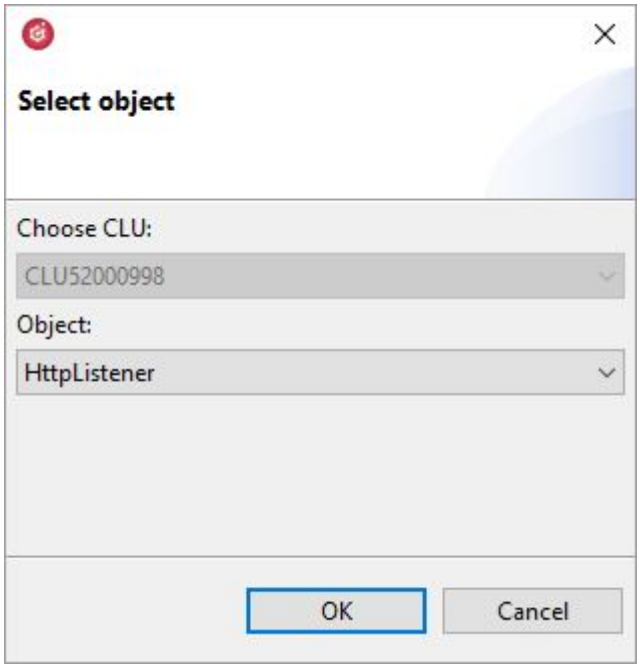
2.2.1. HttpListener

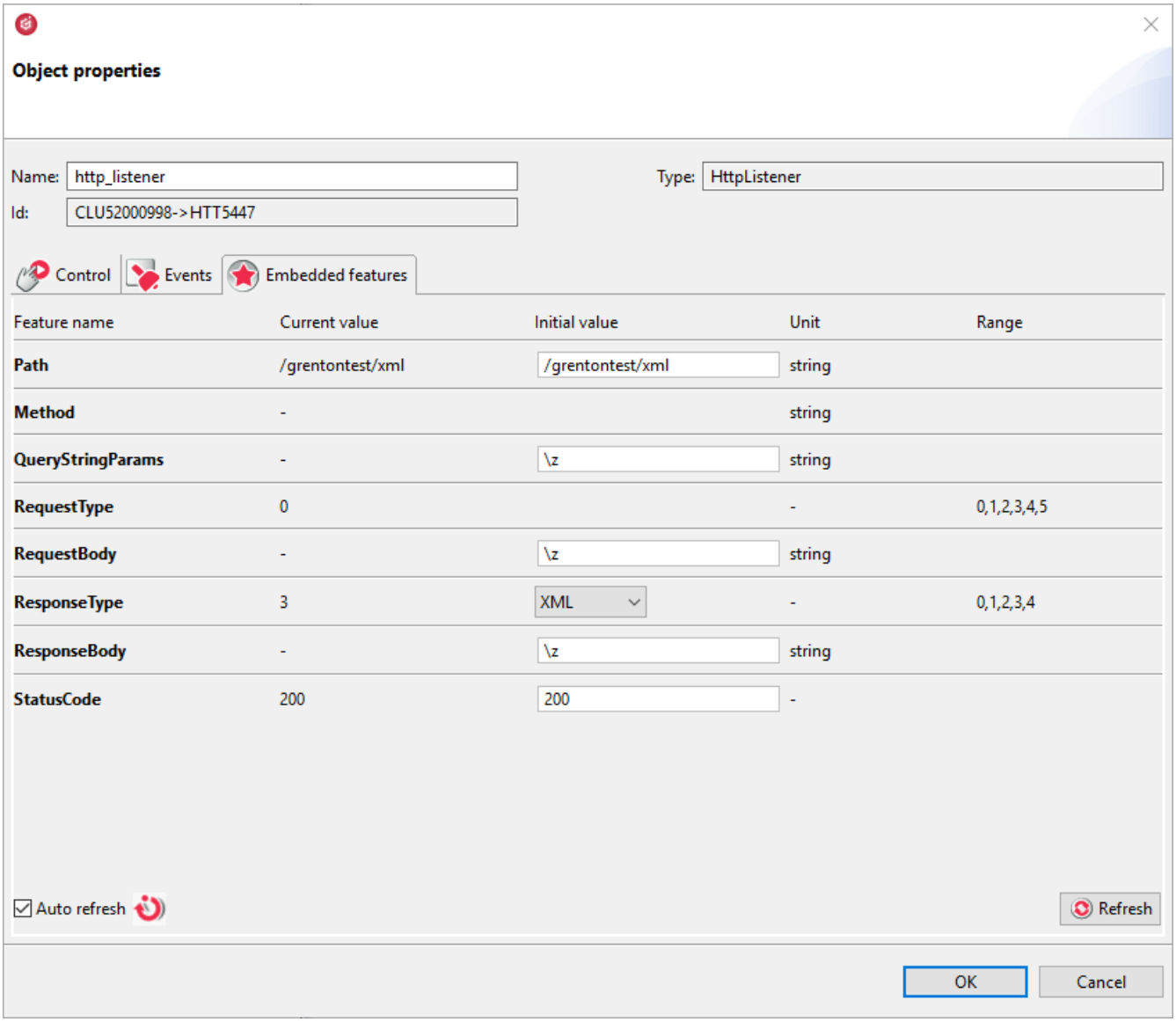
The HttpListener object is used for receiving HTTP (GET, POST) requests. The returned response can be serialized to one of the standard types including JSON, XML. In the HttpListener object, it is important to return the response to every incoming Request.

In the case of listening to Request from the Gate module on the query - for example (using an internet browser):

GET 192.168.4.12/grentontest/xml

You must create the HttpListener virtual object.





The following parameters must be set in the HttpListener object:

- **Path:** /grentontest/xml
- **ResponseType:** XML
- **StatusCode:** 200

> **NOTE!**
>
> Features described as not settable are features containing answers. The initial values of these features should be left unchanged. All operations on these variables should be performed on scripts (and local variables)

You must create a script for the OnRequest event that will create the correct answer and send it back.

2.2.2. Preparation of the response sent to the server

The response is created in the local resp variable.

After preparing the response, set it for ResponseBody (resp) and then send it using the SendResponse () method.

A. XML:

To send the value of a given attribute in response:

```
local resp ="<clu><temperature>" ..CLUZ->x103478262_ONEW_SENSOR1->Value.."</temperature>
</clu>"
GATE_2->Listener_XML->SetResponseBody(resp)
GATE_2->Listener_XML->SendResponse()
```

The answer you've provided is as follows:

```
<clu>
  <temperature>22.5</temperature>
</clu>
```

B.JSON:

```
local resp = {
Temp = CLUZ->x103478262_ONEW_SENSOR1->Value
}
GATE_2->Listener_JSON->SetResponseBody(resp)
GATE_2-> Listener_JSON->SendResponse()
```

The answer you've provided is as follows:

```
{"Temp":22.6}
```

2.2.3. Reading key values from the querystringparams parameter

According to the description of the QueryStringParams feature, its value is not settable, it can be read in the script. If querystring with keys (keys) is sent in the query, the given value can be read from the script level - it is saved in the form of a table.

Individual key values can be obtained on the basis of:

```
value1 = qs.klucz1
```

For the query received:

192.168.1.12/grentontest/query?light1=on&light2=off&light3=on

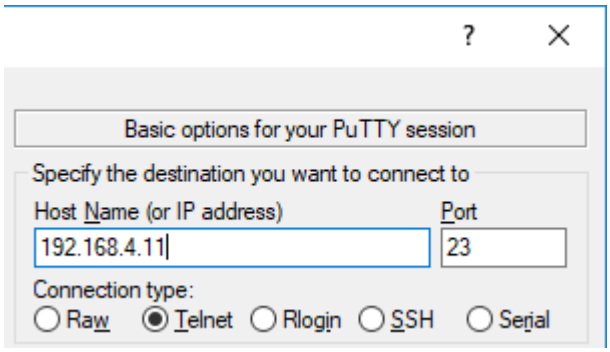
You must create a script:

```
local qs = HTTP_L->grentontest_query_listener->QueryStringParams

local test0 = qs.light1
local test1 = qs.light2
local test2 = qs.light3

HTTP_L->grentontest_query_listener->SetResponseBody()
HTTP_L->grentontest_query_listener->SendResponse()
```

All key values will be saved in local variables (test0, test1, test2).

3. The ability to connect to the Gate using TELNET

For the Gate Http module it is possible to view Lua scripts. In case of configuration error (emergency mode), it is possible to view the error location in the LUA configuration created. The connection is established using the Telnet protocol - for this purpose, for example, the PuTTY program can be used. Examples of parameters to establish a connection:



Two methods can be used to call a connection on the Gate side:

- `StartConsole` – Launches the Lua console. When the method is called, the user has 10s to set the connection to Gate. If the connection is correct, the information about the correct connection will be returned on the terminal (client):

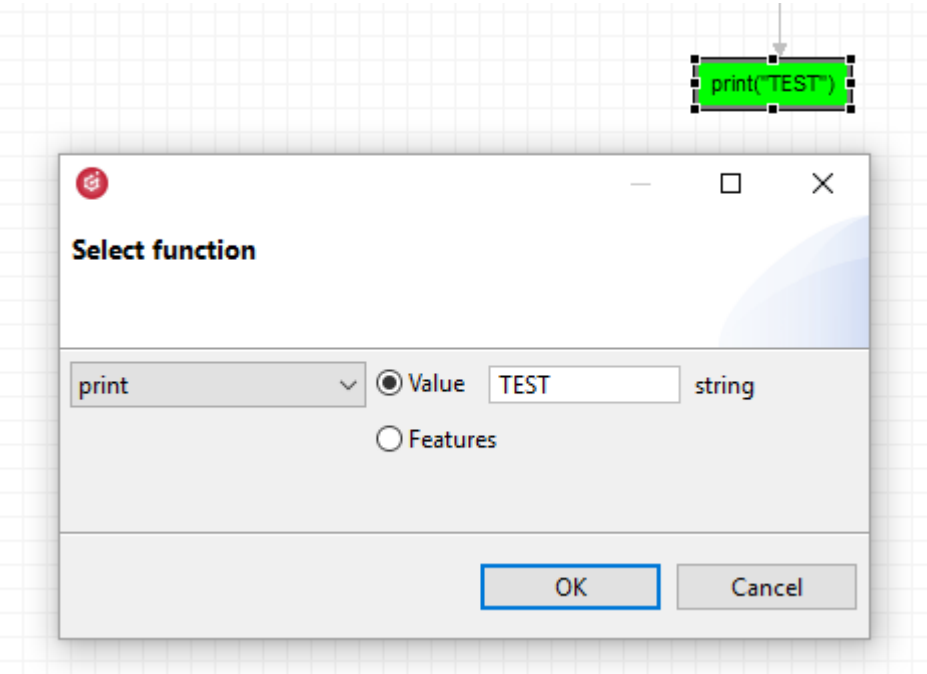
```
CLU SN Telnet session started.
```

- `StartConsoleOnReboot` – allows you to establish a connection the next time Gate reboots. After reboot, the user has 10s to set the connection to Gate. If the connection is correct, the information about the correct connection will be returned on the terminal (client)

```
CLU SN initializing...
CLU: running user.lua...
CLU: running om.lua...
CLU: running OnInit...
CLU: Project loaded.
```

> **NOTE!** It is not recommended to assign the `StartConsole` and `StartConsoleOnReboot` methods to the `OnInit` event of the GATE Http module.

To display eg the value of a given feature on the console, use the *Function block* component and select the *Print* method, and then select the desired feature.



4. 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.

5. Configuration parameters

A. GATE Object

FEATURES

Name	Description
`Uptime`	Operation time of the device since the last reset (in seconds)
`UnixTime`	The current Unix time stamp
`Firmwareversion`	Gate software version
`ClientReportInterval`	Reporting period about changes in features

METHODS:

Name	Description
`SetDateTime`	Sets the date and time
`SetClientReportInterval`	Sets the reporting period for feature changes
`SetUpdateTime`	Sets the date and time of what the state of the central unit is updated
`StartConsole`	Launches the Lua console
`StartConsoleOnReboot`	Launches the Lua console the next time the module is started

EVENTS:

Name	Description
`OnInit`	An event called once at the time of device initialization

B. HttpRequest Object

> **Uwaga!** 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 adress
`Timeout`	Permitted response time
`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
`ResponseBody`	The content of the message received after sending the query. (feature used for reading in scripts - not settable)
`StatusCode`	HTTP response status

METHODS

Name	Description
`SendRequest`	Sends the query
`AbortRequest`	Aborts query handling
`Clear`	Removes the content of the query
`SetHost`	Sets the host address
`SetPath`	Sets the query path
`SetQueryStringParams`	Sets the query parameters
`SetMethod`	Sets the type of variable
`SetTimeout`	Sets the position of the bit
`SetResponseType`	Sets the number of registry bits to read
`SetResponseBody`	Sets the refresh time
`SetRequestHeaders`	Sets the waiting time for a response
`SetRequestBody`	Sets the divisor

EVENTS

Name	Description
`OnRequestSent`	An event triggered when the query is sent
`OnResponse`	The event is triggered when the response is received

C. HttpListener Object

> **NOTE!** Features described as not settable are features containing answers. The initial values of these features should be left unchanged. All operations on these variables should be performed on scripts (and local variables).

FEATURES

Name	Description
`Path`	Query path
`Method`	The type of method obtained in the query, e.g. `GET`, `POST`
`QueryStringParams`	Returns HTTP query parameters (feature used for reading in scripts - not settable)
`RequestType`	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).
`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).
`ResponseBody`	Returns the contents of the HTTP response (a feature used to read in scripts).

Name	Description
`StatusCode`	Status wysyłanej odpowiedzi HTTP. Obsługiwane statusy: `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 a response to the query
`Clear`	Removes the contents of the answer
`SetPath`	Sets the query path
`SetResponseType`	Sets the response type
`SetResponseBody`	Sets the content of the response
`SetStatusCode`	Sets the status of the response

EVENTS

Name	Description
`OnRequest`	The event is triggered when the request is received

XVI. 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 RGBW

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

1.1. General information

The Z-Wave Fibaro RGBW module enables reading and setting the status of single output channels R, G, B, W in the range from 0 to 255. In addition, it gives the possibility to change the configuration parameters (Fibaro configuration interface).

1.2. 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 - eg set directly from the button connected to the module.

> **NOTE!** The value from the attached button is sent when 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 color
`RampTime`	Time of rise / fall of change of dimmer value in milliseconds. The value of this feature affects the actions triggered by the 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 triggered when the dimmer value is changed
`OnSwitchOn`	An event triggered when the dimmer status is changed to on
`OnSwitchOff`	An event is triggered when the dimmer status is changed to off

B. 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).

> **NOTE!** In the case of 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 the CLU Discovery.

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 blocked (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	Desctiption
`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. Fibaro UBS

Module version: FGBS-001 v2.1.

2.1. 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.

2.2. 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

Nazwa	Opis
`SetHoldDelay`	Sets `HoldDelay` value
`SetHoldInterval`	Sets `HoldInterval` value

EVENTS

Name	Description
`OnChange`	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
`OnChange`	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	Desctiption
`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

3. NEO Coolcam Motion Sensor (PIR)

Module version: NAS-PD01ZE HW: 66 FW: 3.80

3.1. 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.

3.2. 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
`OnChange`	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
`OnChange`	An event triggered when the sensor value is changed
`OnValueRaise`	An event is triggered when the sensor value changes to a higher one than the previous one
`OnValueDrop`	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 is 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

4. NEO Coolcam Door / Window Sensor

Module version: NAS-DS01Z

4.1. 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.

4.2. 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
`OnChange`	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

5. INFIBITY Motion Sensor (PIR) [NEO Coolcam]

Module version: NAS-PD01ZE HW: 66 FW: 3.80

5.1. 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.

5.2. 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
`OnChange`	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
`OnChange`	An event triggered when the sensor value is changed
`OnValueRaise`	An event triggered when the sensor value changes to a higher one than the previous one
`OnValueDrop`	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

6. INFIBITY Door/Window Sensor [NEO Coolcam]

Module version: NAS-DS01Z HW: 65 FW: 3.61

6.1. 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.

6.2. Obiekty

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
`OnChange`	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`	The 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

7. INFIBITY Water Sensor [NEO Coolcam]

Module version: NAS-WS02ZU HW: 32 FW: 2.133

7.1. 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!

7.2. 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
`OnChange`	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`	Okres samoczynnego wybudzania modułu Z-Wave z trybu uśpienia (w sekundach)
`LastWakeUp`	Czas ostatniego wybudzenia modułu Z-Wave z trybu uśpienia

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

8. Heiman Smart Smoke Sensor

Module version: ***HS1SA-Z (HS1SA-Z HW: 255 FW: 1.10)***

8.1. 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.

8.2. 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)

EVENTS

Name	Description
`OnChange`	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

9. INFIBITY Siren Alarm [NEO Coolcam]

Module version: ***NAS-AB01Z HW:48 FW: 2.90***

9.1. 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.

> **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!** After CLU reboot (sending configuration), wait 10s before the first attempt to turn on the Siren Alarm module.

9.2. 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
`OnChange`	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 device is banned
`OnLowBattery`	An event triggered when a battery drop is detected below the warning level
`OnBatteryGood`	An event triggered when the battery level returns to a value above the warning level

C. ZWAVE_WAKEUP

An object enabling setting and reading of the battery Wake Up time of the Z-Wave module. The default setting value for the CLU is 3600s (5 minutes). The minimum value is 60s (1 minute); maximum 16777200s (about 194 days).

FEATURES

Name	Description
`Interval`	The period of self-awakening of the Z-Wave module from the sleep mode in seconds
`LastWakeUp`	Time of the last awakening of the Z-Wave module from sleep mode

METHODS

Name	Description
`SetInterval`	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

EVENTS

Name	Description
`OnWakeUp`	An event triggered when the Z-Wave module wakes up from sleep mode

D. ZWAVE_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It also allows setting advanced configuration parameters of a given module (specified individually in the manual).

Setting register 7 changes the siren mode:

- As an **Alarm** – the siren operates according to the parameter settings: 1,2,5,8
- As a **DoorBell** – the siren operates according to the parameter settings: 3,4,6,9

FEATURES

Name	Description
`NodeID`	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
`Banned`	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) **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

10. Danfoss Living Connect

Module version: EU HW: 00 FW: 1.1

10.1. 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.

> **NOTE!** 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.

10.2. 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 device is banned
`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

11. POPP Z-Weather

Module version: EU HW: 01 FW: 1.0

11.1. 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.

> **NOTE!** 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. To wake up the device, click 1x on the device.

11.2. 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`	Ustawia poziom ostrzegawczy baterii modułu Z-Wave

EVENTS

Name	Description
`OnChange`	An event triggered when the device is banned
`OnLowBattery`	An event triggered when a battery drop is detected below the warning level
`OnBatteryGood`	An event triggered when the battery level returns to a value above the warning level

C. ZWAVE_WAKEUP

An object enabling setting and reading of the battery Wake Up time of the Z-Wave module. The default setting 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

12. FAKRO AMZ Solar

Module version: Type ID 0x0005 Manuf. ID 0x0085 Product ID 0x0112

12.1. 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).

> **NOTE!** Adding / removing the device is done by pressing the 'P' button on the device during inclusion / exclusion (called on the CLU).

12.2. 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

> **Uwaga!** The value of the set configuration parameters is refreshed at the time of `WakeUp` of the given device (values are taken from the Z-Wave device).

METHODS

Name	Description
`Up`	Awning up
`Down`	Awning down
`Stop`	Stop if the awning is in motion
`Start`	Awning up if previously move down, awning down if previously move up
`SetPercent`	Sets the percentage, where 100% - awning opened
`SetMode`	Sets the device's operating mode
`SetSeasonMode`	Sets the seasonal mode
`SetOpeningTime`	Sets the awning opening time
`SetSensitivity`	Sets the sensitivity of the sun exposure level

EVENTS

Name	Description
`OnChange`	An event triggered when the window controller state changes
`OnUp`	An event will trigger at the time of changing from Stop to Up
`OnDown`	An event triggered when the state changes from Stop to Down
`OnStart`	An event triggered when the Start command is called
`OnStop`	An event triggered when the Stop command is issued

ZWAVE_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

FEATURES

Name	Description
`NodeID`	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
`Banned`	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

13. FAKRO ARF

Module version: Type ID 0x0004 Manuf. ID 0x0085 Product ID 0x0011

13.1. 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.

> **NOTE!** 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 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

14. FAKRO FTP_V

Module version: Type ID 0x0004 Manuf. ID 0x0085 Product ID 0x0011

14.1. General information

FAKRO FTP_V module support includes window control - both through maximum opening / closing and setting the percentage of window opening.

> Adding / removing the device is done by pressing the 'P' button on the device during inclusion / exclusion (called on the CLU).

14.2. 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

15. Remotec ZXT-310

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

15.1. General information

The support of the ZXT-310 Remotex 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.

> **The way of adding / removing:** 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.

> **The method of restoring 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.

15.2. Device configuration

A. The way of teaching IR codes

1. Uczenie kodów odbywa się za pomocą 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.

15.3. 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 `EmmitterPower` 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* `EmmitterPower` *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 `EmmitterPower` parameter is not configurable for port 1*
`TransmissionMode`	Returns the IR code transmission mode: `0` – continous 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 `EmmitterPower` 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

METHODY

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

16. Remotex ZXT-120

Module version: ZXT-120EU V1.0

16.1. General information

The handling of the Remotex 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.

> **The way of adding / removing:** 1x click the PROG button in the module during inclusion / exclusion - the red LED will flash 1x and then it will turn on continuously.

> **Method of restoring 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).

16.2. Description of device configuration

1. 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.

16.3. 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 device is banned
`OnLowBattery`	An event triggered when a battery drop is detected below the warning level
`OnBatteryGood`	An event triggered when the battery level returns to a value above the warning level

C. ZWAVE_WAKEUP

An object that allows setting and reading the reading time of the Z-Wave module parameters. The default setting value for the CLU is 3600s (60 minutes). The minimum value is 10s, maximum 16777200s (about 194 days). It is possible to set the value in step 5s.

> **NOTE!** It is not recommended to set the value of the `WakeUp` feature less than 60s during normal device operation. Decreasing the value may be useful only in the case of 'teaching' codes by the device (generating changes in the status of learning mode, as well as reading the `LearningStatus` feature), as well as in setting configuration parameters

FEATURES

Name	Description
`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

---

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.↩
7. Within the meaning of the instructions, the word consists of two bytes.↩