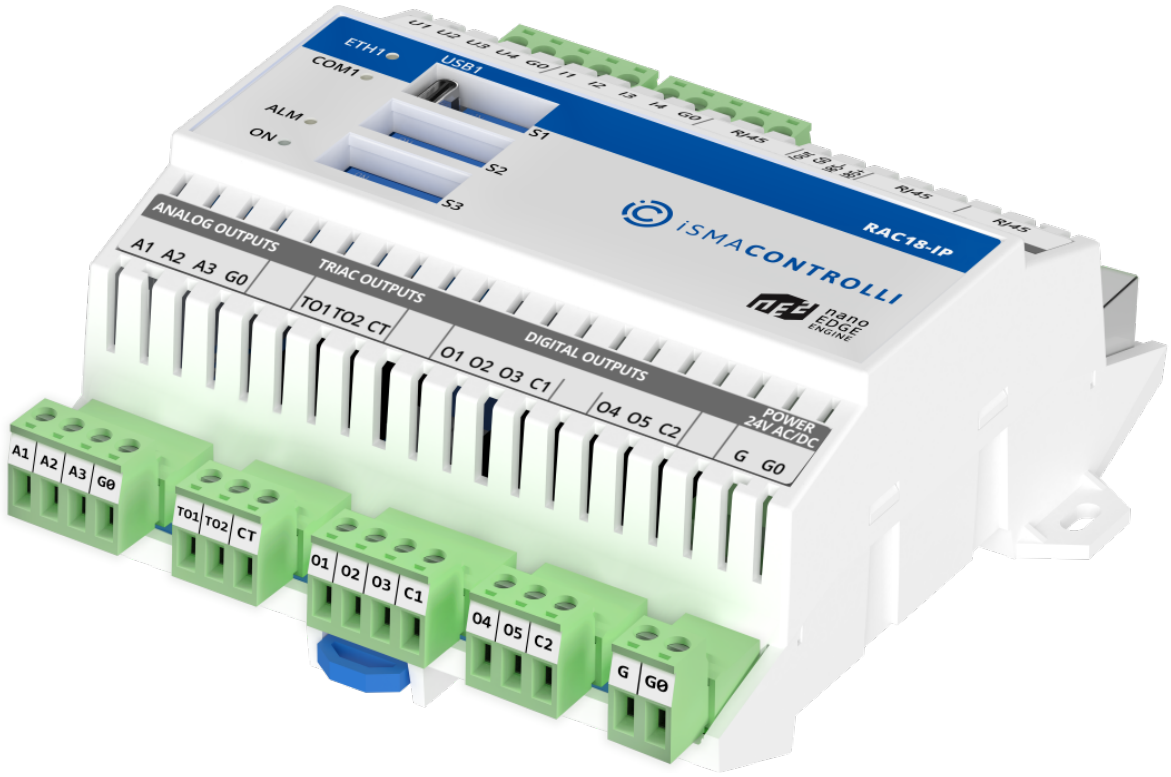
RAC18-IP

User Manual

Software



iSMA CONTROLLI S.p.A. - Via Carlo Levi 52, 16010 Sant'Olcese (GE) - Italy | support@ismacontrolli.com

Table of Contents

1 Introduction 7
1.1 Revision History 7
2 Basic Concept of the nano EDGE ENGINE 8
2.1 Structure of Containers and Their Functionalities 8
3 Licensing 11
4 Linking 12
4.1 Reference Linking 12
5 Emergency Mode 20
5.1 Operation in Emergency Mode 20
5.2 Possible Actions 21
6 Applications Container 22
6.1 Applications Libraries 23
6.2 Basic Concept of Applications 23
6.2.1 Applications 24
6.2.2 Data Points 27
6.3 Core (for Applications) 28
6.3.1 Application 29
6.3.2 Analog Data Point 31
6.3.3 Binary Data Point 39
6.3.4 Equipment 46
6.3.5 Folder 47
6.4 Logic 48
6.4.1 And 49
6.4.2 Between 50
6.4.3 Binary2Numeric 51
6.4.4 Equal 52
6.4.5 GreaterOrEqual 53
6.4.6 GreaterThan 54
6.4.7 JKFlipFlop 54
6.4.8 LessOrEqual 56
6.4.9 LessThan 56
6.4.10 Nand 57
6.4.11 Nor 58
6.4.12 Not 59
6.4.13 NotEqual 59

6.4.14 Or 60
6.4.15 RSLatch 61
6.4.16 Select 62
6.4.17 Switch 63
6.4.18 Toggle 64
6.4.19 Xor 65
6.5 Math 66
6.5.1 Absolute 68
6.5.2 Add 68
6.5.3 ArcCosine 69
6.5.4 ArcSine 69
6.5.5 ArcTangent 70
6.5.6 Average 70
6.5.7 Cosine 71
6.5.8 Divide 72
6.5.9 Exponential 72
6.5.10 Factorial 73
6.5.11 LogBase10 73
6.5.12 LogNatural 74
6.5.13 Maximum 74
6.5.14 Minimum 75
6.5.15 MinMaxAverage 76
6.5.16 Modulus 77
6.5.17 Multiply 78
6.5.18 Negative 79
6.5.19 Power 79
6.5.20 RootY 80
6.5.21 Round 81
6.5.22 Sine 81
6.5.23 Square 82
6.5.24 SquareRoot 82
6.5.25 Subtract 83
6.5.26 Tangent 84
6.5.27 Truncate 84
6.6 Process 84
6.6.1 Curve 85
6.6.2 Floating 87

6.6.3 Hysteresis 89
6.6.4 Latch 90
6.6.5 Limit 92
6.6.6 Linear 93
6.6.7 Loop 94
6.6.8 Thermostat 96
6.7 Time 98
6.7.1 Counter 98
6.7.2 MinOnOff 100
6.7.3 OneShot 101
6.7.4 OnOffDelay 102
6.7.5 Timer 104
6.8 FCU 105
6.8.1 EffectiveSetpointCalculator 106
6.8.2 FanControl 108
6.8.3 HeatingCoolingSwitch 116
6.8.4 MasterSlave 117
6.8.5 ModeCalculator 119
6.8.6 OccupancyCalculator 121
6.8.7 OutputsSwitch 123
6.8.8 PWM 127
6.8.9 TemperatureAnalog 128
6.8.10 TemperatureBinary 135
6.8.11 TemperatureSensorsSwitch 142
6.8.12 WindowStatusSwitch 146
6.8.13 ValueHolder 147
6.9 Other 149
6.9.1 AnalogConstant 149
6.9.2 AnalogNode 150
6.9.3 BinaryConstant 151
6.9.4 BinaryNode 151
6.9.5 Ramp 152
6.10 Quick Start-up of Applications 153
6.10.1 Applications Manager 158
6.10.2 Data Point Manager 160
6.10.3 Equipment Manager 164
7 Networks Container 167

7.1 Networks Libraries ... 167
7.2 Basic Concept of Networks ... 168
7.3 Core (for Networks) ... 169
7.3.1 Interfaces ... 170
7.3.2 Ethernet ... 172
7.3.3 Serial ... 173
7.3.4 Network ... 175
7.3.5 Folder (Networks) ... 176
7.4 LocalIO ... 177
7.4.1 LocalIO Component ... 178
7.4.2 UniversalInput ... 180
7.4.3 DigitalInput ... 183
7.4.4 DigitalInputCounter ... 185
7.4.5 AnalogOutput ... 187
7.4.6 DigitalOutput ... 191
7.4.7 TriacOutput ... 193
7.4.8 DipSwitch ... 196
7.4.9 Quick Start-up of LocalIO ... 199
7.5 BACnet ... 203
7.5.1 BACnet Component ... 204
7.5.2 LocalDevice ... 206
7.5.3 Device ... 209
7.5.4 AnalogPoint ... 211
7.5.5 BinaryPoint ... 214
7.5.6 Quick Start-up of BACnet ... 216
7.6 Modbus ... 226
7.6.1 Modbus Component ... 227
7.6.2 Device ... 229
7.6.3 AnalogPoint ... 230
7.6.4 BinaryPoint ... 234
7.6.5 StringPoint ... 236
7.6.6 Quick Start-up for Modbus ... 243
8 Services Container ... 254
9 System Container ... 255
9.1 Basic Concept of System ... 255
9.2 License ... 256
9.3 iFnet ... 257

9.3.1 Statistics 259
9.4 Users 260
9.4.1 First Logging In 261
9.5 Logs 264
9.5.1 Extensions 266
9.5.2 Log Viewer 266
9.6 Backups 269
9.7 Platform 274
9.7.1 Performance 277
9.7.2 Time 277
9.7.3 Ethernet 278
10 RAC18-IP Quick Start-up 282

1 Introduction

The nano EDGE ENGINE is a software engine designed for the new generation of the iSMA CONTROLLI devices. It is the software engine for constructing multiple applications, and it provides a set of libraries and components tailored to create cycle-driven user applications. The nano EDGE ENGINE ensures connectivity with added devices, and allows to run dedicated services. It provides full management of the device–setting network properties, providing logs for diagnostics and performance data.

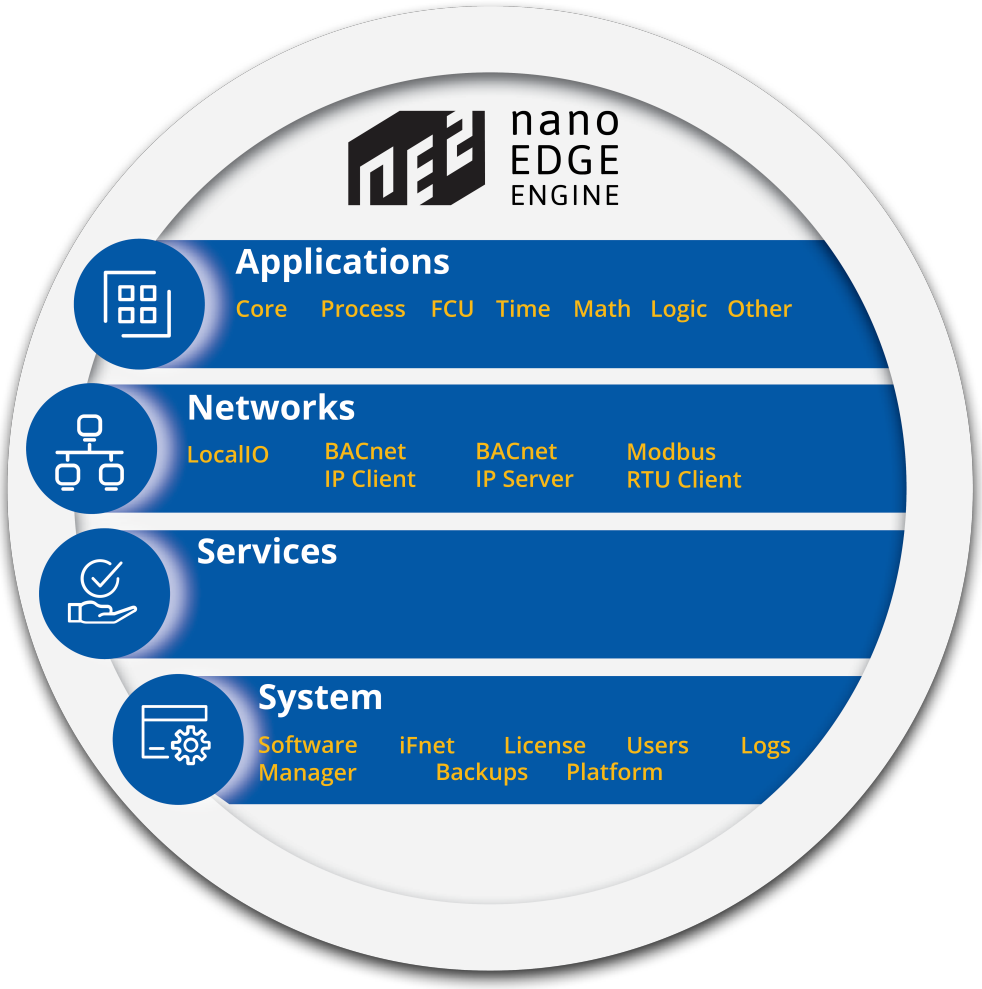


Figure 1. The nano EDGE ENGINE diagram

The nano EDGE ENGINE introduces a user-friendly interface that makes working with the device very efficient. It provides a predefined quadruple structure tree of the device, which guides the user through the whole process of setting and working with the device.

1.1 Revision History

Date	Rev.	Description
21 Apr 2022	1.0	First edition

Table 1. Revision history

2 Basic Concept of the nano EDGE ENGINE

The nano EDGE ENGINE is a software engine, which is based on a few innovative concepts:

- **clear and logical structure** (all elements grouped in four containers: Applications, Networks, Services, and System);
- **Data Points with automatic BACnet exposition** (universal components representing values in applications);
- **Reference linking method** (dedicated for linking Data Points with network points, transferring value along with component's status);
- **favorable licensing system** (Analog and Binary Data Points are the only licensed elements).

2.1 Structure of Containers and Their Functionalities

The structure tree of the nano EDGE ENGINE is based on **four containers**. Containers are components, which have no slots, and are designed to group other components within:

The first container is the Applications–a functional center of the device. Here are located Application components, which are designed to build user applications by adding components and linking them into a logical and mathematical sequences. Applications built with the nano EDGE ENGINE are cycle driven, meaning the included components are executed in repeated periods of time lasting as long as a cycle time is set by the user—thanks to the efficiency of the nano EDGE ENGINE, the cycle may be as short as 15 ms (depending on the complexity of application), allowing effectively real-time calculations. **The nano EDGE ENGINE allows constructing numerous independent, cycle-driven user applications that may work simultaneously**. With the nano EDGE ENGINE the user may freely add as many Application components to the Applications container as needed. The only considerations here are the number of available licensed Data Points (see the Licensing section), available memory, or cycle time. Data Points are main application components, representing values in applications and allowing to connect services from the Networks container with the application—while connecting components from the Applications and Networks containers, it is recommended to use the Reference linking method. Data Points may be exposed to external networks, which means that external exposition of data from the Applications container or any other space may be carried out only via the Data Point.

The second container is the Networks—a peripheral communication center of the device. This container includes **components that manage external communications of the device**, for example, the LocalIO, which manages physical inputs and outputs of the device, or the BACnet, which manages the exchange of data with the BACnet network (as well as the Modbus over a serial port). The Networks container includes components that allow the device to transfer the HVAC automation data needed for the device to communicate externally. These components cover the fundamental layer of HVAC data transmission and are crucial for the device to be able to properly exchange information. The network point class components are located in the Networks container; both data transmission from the network points and controlling them in the application is achieved by the Reference linking, which is a linking method recommended over linking input and output slots directly, as the standard linking method misses the advantage of transferring the status info.

The third container is the Services—a utility center of the device. This container incorporates all services that enhance the basic functionality of the device. As the crude automation data transmission in the nano EDGE ENGINE is covered by services in the Networks container, **the Services container provides the value added**: these services cover an exchange of information with systems or devices superior to the building automation level. They may feed the algorithms with data that are normally unavailable in the building automation system, for example, a Weather service, which delivers information (precipitation, humidity, cloudiness, etc.) to the algorithm in the application managing garden watering, or an Alarm service, which sends out alarm messages to external recipients, provided certain limit values are exceeded. These services provide the additional logical layer to algorithms working within applications. They introduce external factors to user applications, enhancing their functionalities outside the core building automation. They may also introduce limit values to evoke certain actions in applications.

The fourth container is the System–a configuration center of the device. The System container includes **components that provide hardware characteristics of the device and allow to configure its settings**. It is an area where the user may find information about the device model, version of operating system, free memory, etc. Most importantly though, it is where the user may set a unique IP address for the device to enable connecting it to the network. The System container is also a place to refer to for troubleshooting; it includes the Logs, which records an adjustable register of events in the device. If needed, it may be shared with the Technical Support team for further diagnostics.

These are the basic four pillars that encompass the central functionalities of the devices driven by the nano EDGE ENGINE—the space for application building, the space for enabling external communications of the device, the space for adding various services to the applications, and the space for device configuration. The nano EDGE ENGINE is a universal software engine, which provides a set of components and libraries destined for

building automation sector (i.e., network points and Data Points, Math components, Logic components, Process components, etc.).

3 Licensing

The license for the new generation of iSMA CONTROLLI devices driven by the nano EDGE ENGINE is constructed against the number of Data Points: each device based on the nano EDGE ENGINE is granted a specified number of license points (Data Points in this case), which can be used within applications. Therefore, the licensing system is only of quantitative, not functional, character–only the real number of Data Points in applications is taken into account, regardless of how many communication protocols are used to expose them, or how many network points are controlled. With the nano EDGE ENGINE-generation devices it is possible to create as big an application (or applications) as the number of licensed Data Points. No elements in the Networks, Services, or System containers are subject to license limitations, other than Data Points in the Applications container.

Note: In order to check the number of license points, please refer to the License in the device.

4 Linking

The Reference linking is a unique characteristic for the nano EDGE ENGINE, and it offers an exceptionally user friendly experience while creating applications.

Linking within applications, built with the nano EDGE ENGINE, may be performed twofold:

- using a special Reference link designed specifically to connect Data Point class components (in the Applications container) with network point class components (in the Networks container);
- using a standard linking method, which involves simple creating links between the input and output slots; a standard link transfers a value between the connected slots. Standard linking may be applied between all four containers of the nano EDGE ENGINE device structure.

4.1 Reference Linking

The nano EDGE ENGINE offers an innovative linking method called the Reference link.

The Reference linking method is unique for the nano EDGE ENGINE devices. Is it designed to link Data Points with network points, and it offers much more advantage than the standard linking method.

The **Reference link** is a special compound link designed to connect **Data Points with network points**. The Reference link is created between special Reference slots and **transfers values along with the component's status**. Alternatively, it may transfer values between Data Points and network points at the same time returning status from network points to Data Points, or it may return values from network points to Data Points. As network points are situated in the Networks container and Data Points are situated in the Applications container, Reference links are created using the Link Mark and Link From options from the context menu, and they are created between the tabs (or, for example, between the Application tab and the network points expanded in the Workspace Tree window) or within the Workspace Tree window between Data Points in the Applications container and network points in the Networks container. Either way the Reference link between tabs is displayed in the Wire Sheet as a bubble connected to the component's Reference slot.

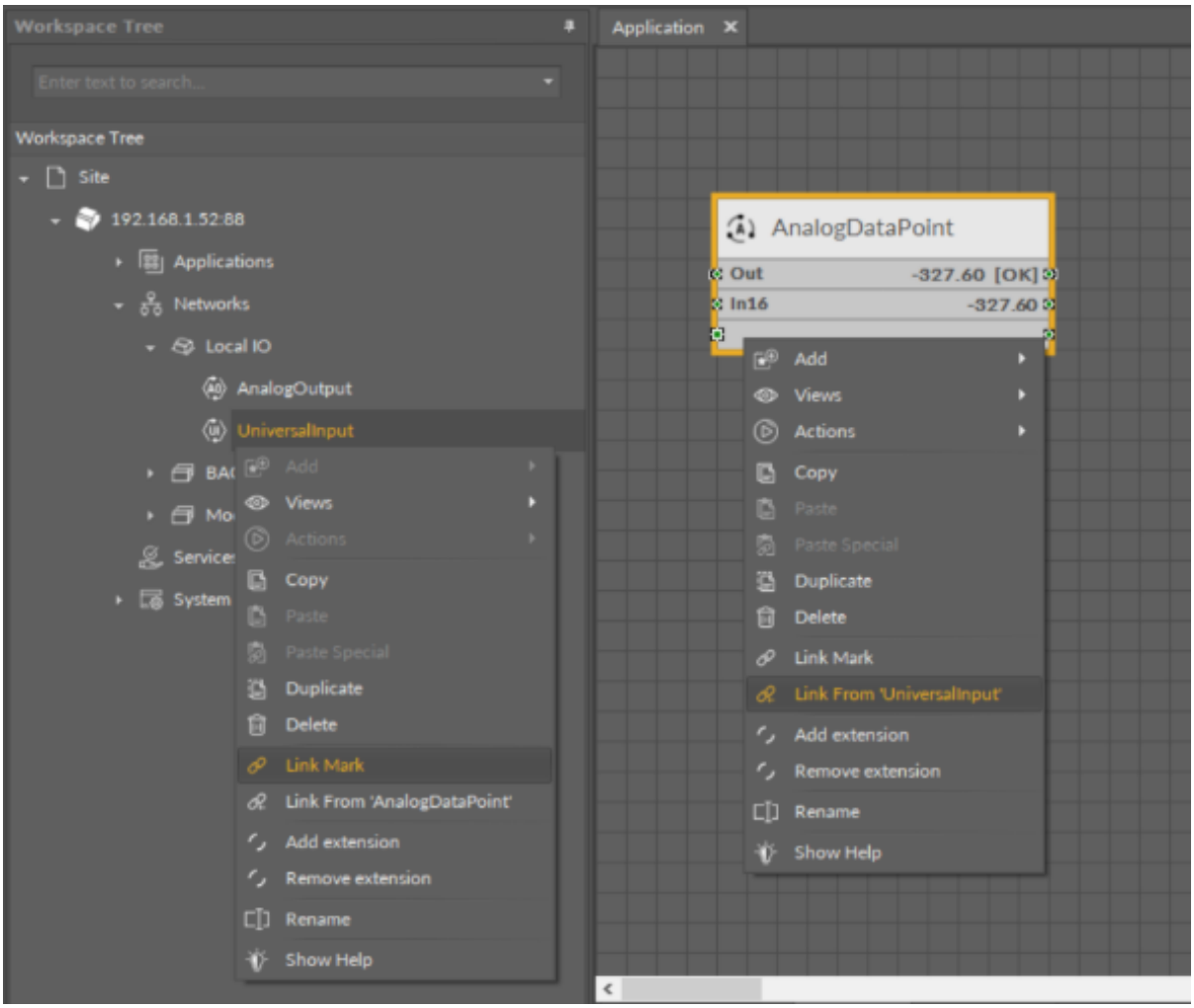


Figure 2. Link Mark-Link From options

The basic and exceptional feature of the Reference links is the fact that they are made to **transfer the value along with the component's status**. This feature gives a major advantage and translates to substantially enhanced functionality of linking. The fact of transferring the Status along with component's value is exceptionally important for the functionality of Data Points. Data Points are central elements in the nano EDGE ENGINE applications, and they represent values in applications on the Wire Sheet. Therefore, displaying network point's status makes the Data Point much more informative, and allows to display this important information directly in the Wire Sheet.

Data Point and the Input-type Network Point Links

If the Data Point is linked with the input-type network point, the Reference link transfers the network point's value and status to the Data Point (if the Data Point has a priorities array extension added, there is also the option to set the Input Priority slot in the network point, which defines the input priority in the Data Point receiving the value from the network point). In this variant, the Reference link is unidirectional, and provides the information about the change of value and the network point's status.

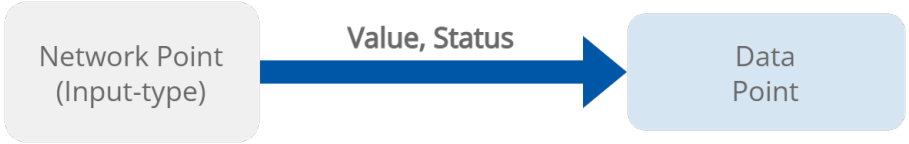


Figure 3. Data Point and the input-type network point links

If the Data Point is linked with the output-type network point (or network points), it offers even more advantages.

Data Point and the Output-type Network Point Links

- First of all, in such case the Reference link behaves **bidirectionally**. It transfers the value from the Data Point to the network point, and **in turn it informs whether the value has been correctly received by the network point by sending back the network point's status**. This hugely advantageous feature allows to instantly identify that at least one of the linked network points has gone into the fault status.

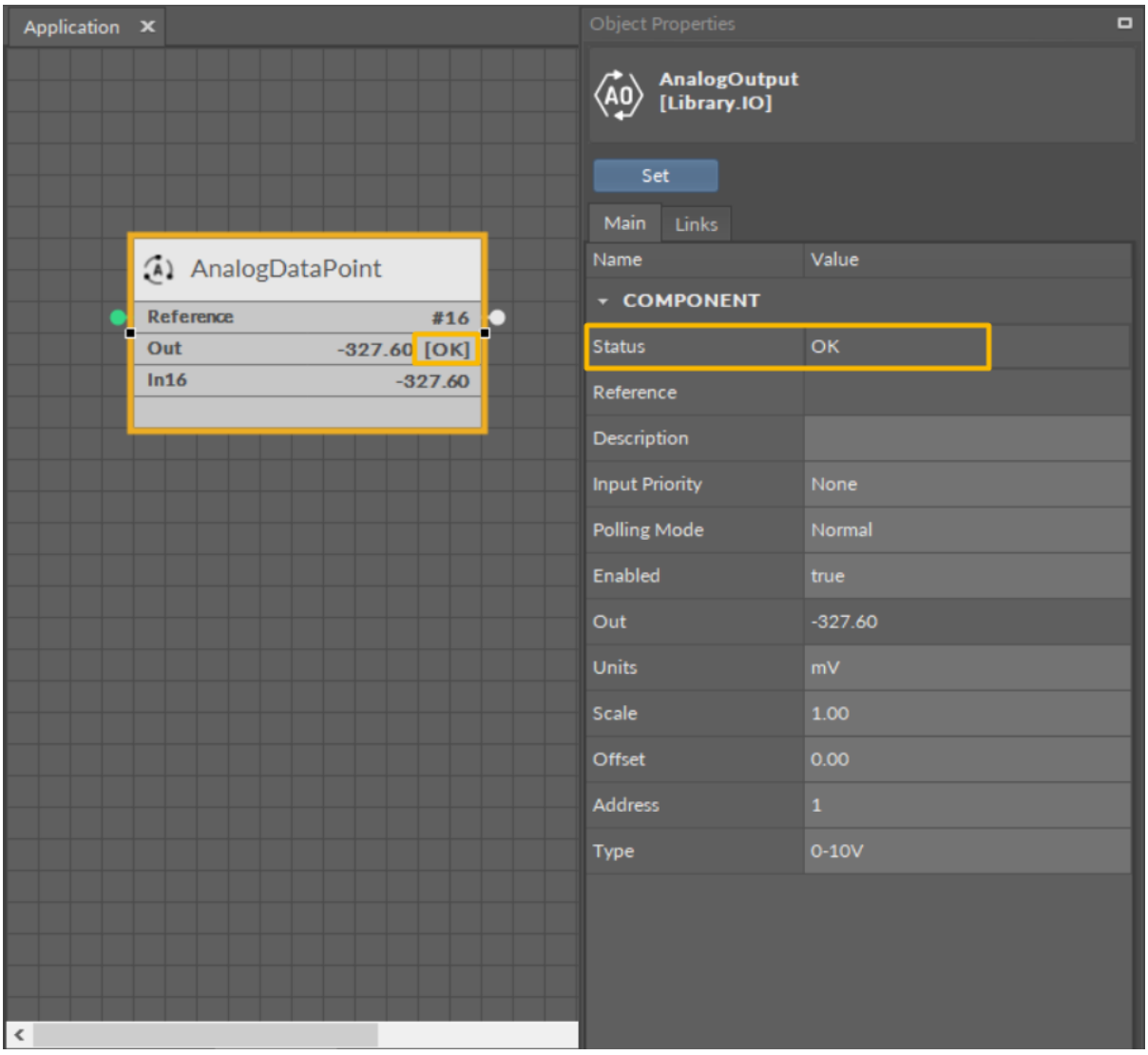


Figure 4. Output-type network point (status OK) referenced to the Data Point

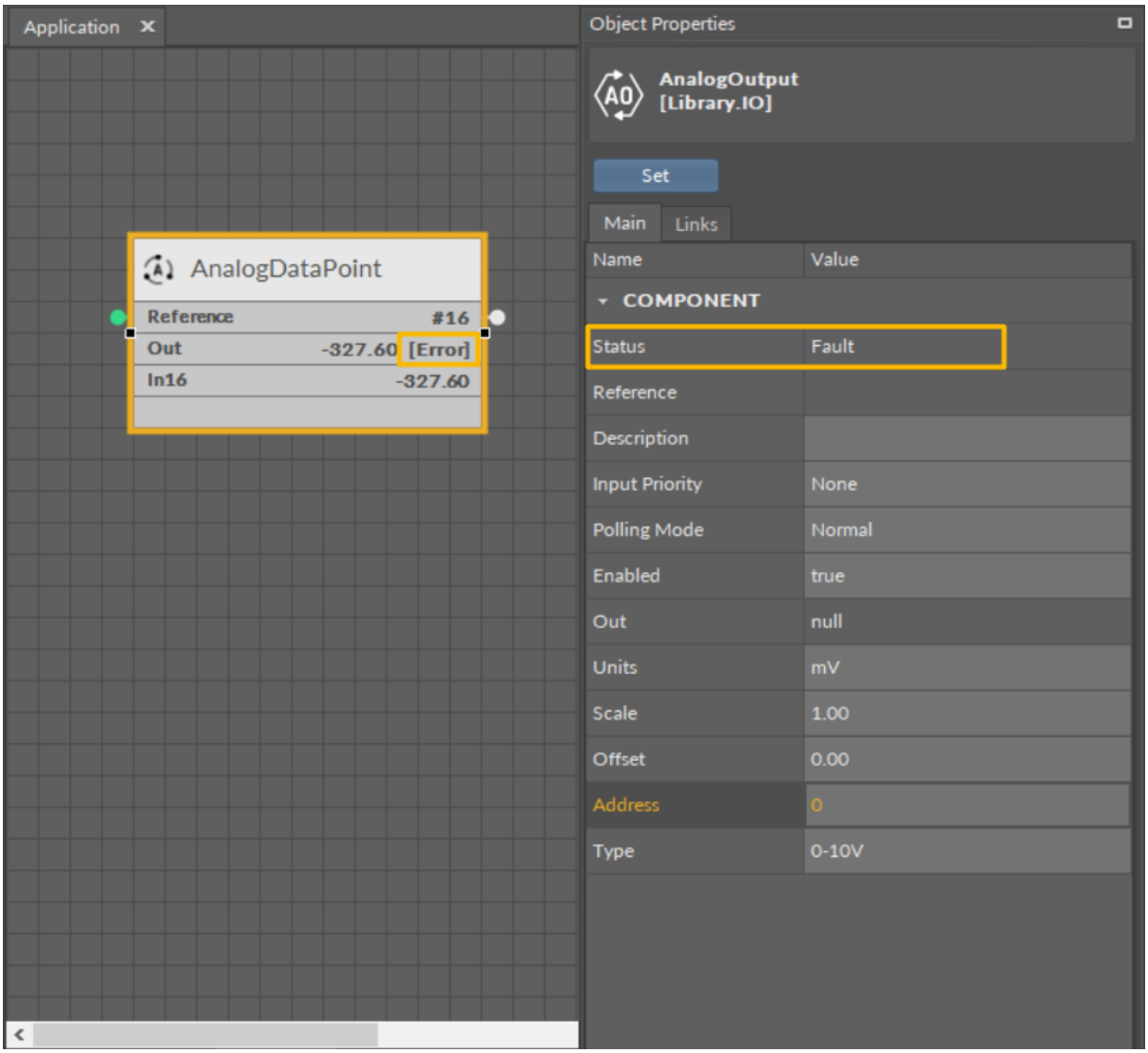


Figure 5. Output-type network point in the Fault status returning an Error status to the referenced Data Point

- Another innovative feature of the Reference linking is that the Reference link is able not only to return status from the network point but also to **return its value**. The network point's slot, Input Priority, is designed to identify the Data Point's priority, which the network point will transfer its value back to. For example, if the network point's Input Priority slot is set to In16, it will transfer its value back to the Data Point's 16th priority slot. In turn, if this value is the highest priority for the Data Point, it can distribute it to all network points linked with the Reference link. **This way, if there are more output-type network points linked with the Data Point, and one of them changes its value and sends it back to the Data Point, thanks to the bidirectional Reference link, the Data Point can synchronize values in all linked network points.**

Note: The difference between the two possible options for linking Data Points with output-type network points is derived from the network point's **Input Priority slot**. In the first scenario, the network point does not have the Input Priority set to any value, therefore, it cannot return a value back to the Data Point. In the second option, the network point has the Input Priority slot set and it reacts to the change of value–if the network point's value is changed, it is automatically sent back to the Data Point by the Reference link, and is updated on the defined input priority.

Priorities Array Extension

Setting the network point's Input Priority slot is effective providing that the Data Point has the priorities array extension added. The Data Point is available in its basic version with one input slot (In16), however, it can be expanded by another 15 writable input slots with the priorities array extension (available at the right-click on the Data Point). If the Data Point has 16 writable slots, setting the Input Priority slot in the network point defines the Data Point's input receiving the value from the output-type network point over the Reference link. If the Data Point remains in its basic version, setting the Input Priority slot in the network point has no actual effect, and the value is sent to the 16th input priority in the Data Point.

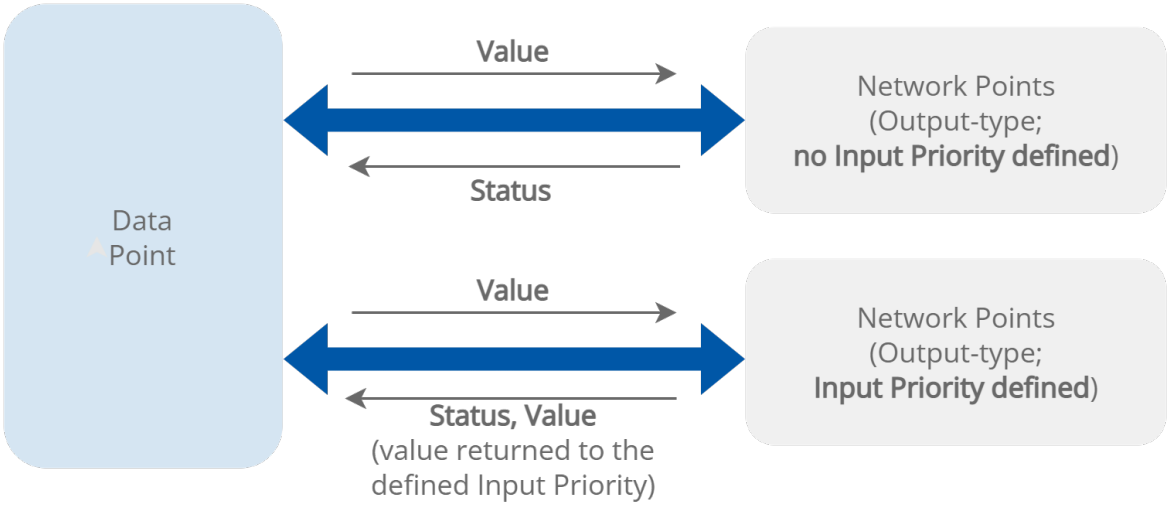


Figure 6. The Reference links between the Data Point and output-type network points

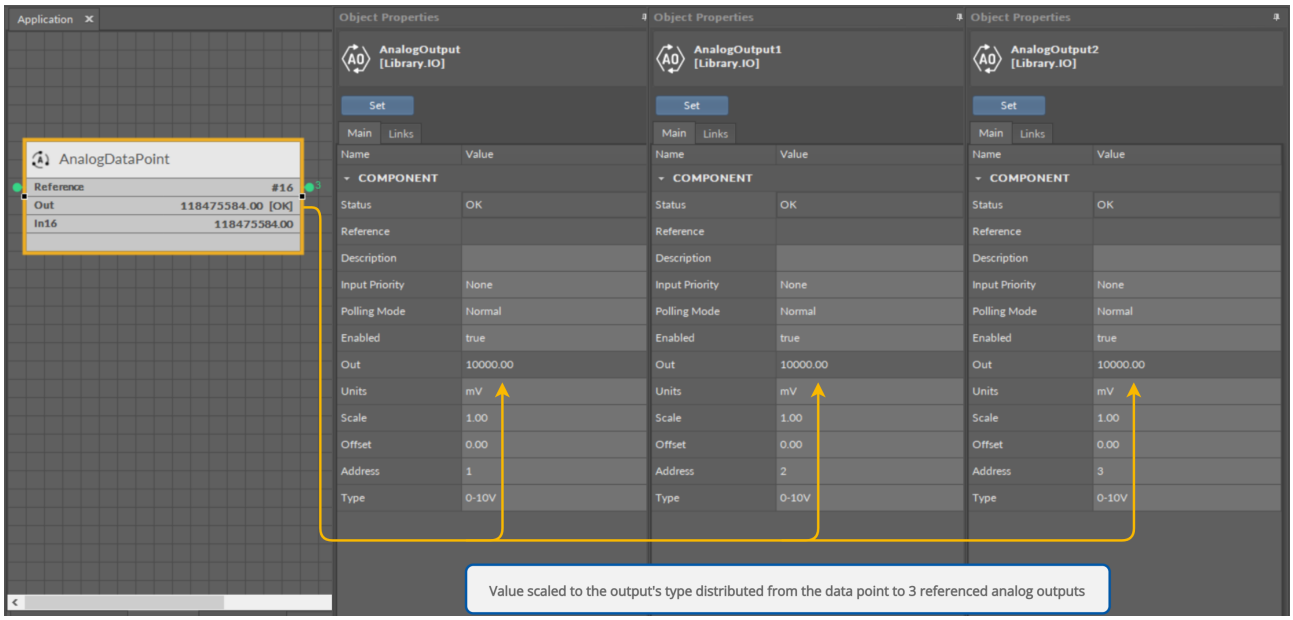


Figure 7. Values distributed from the Data Point to the referenced output-type network points

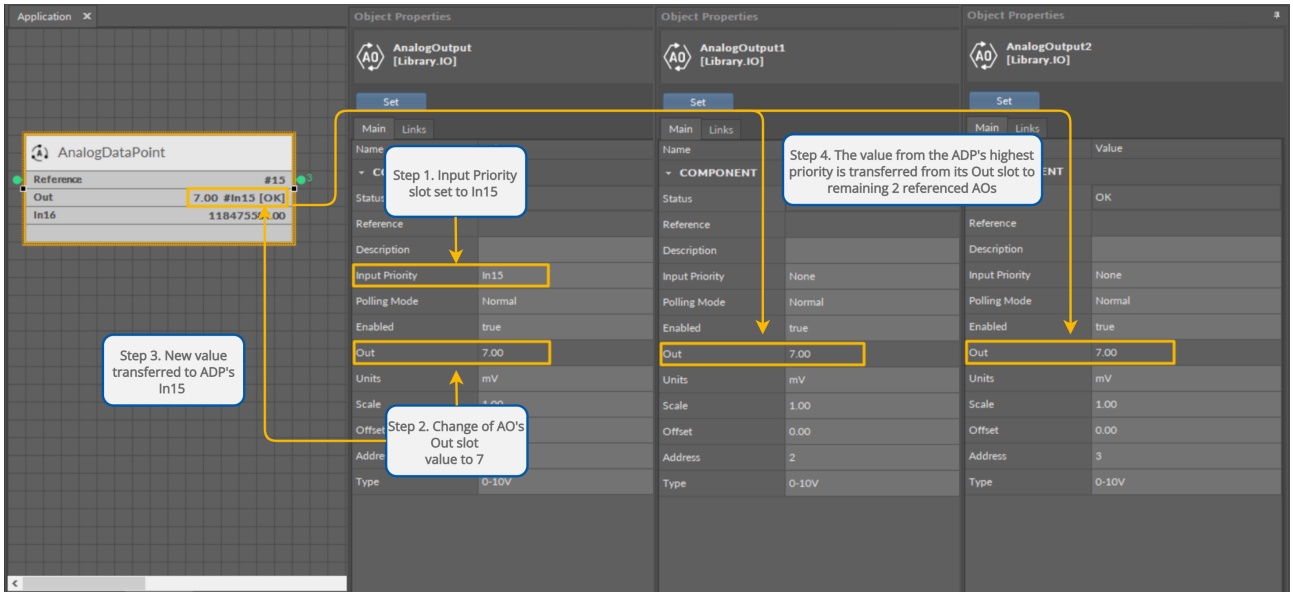


Figure 8. Value sent back to the specified input priority in the Data Point and distributed to remaining referenced output-type network points

The Reference linking is recommended for linking components in the Applications with components in the Networks container.

- Using the Link Mark and Link From options from the context menu of network points and Data Points that are to be linked, opens the dialog window, which allows to select Reference slots on both sides of the link:

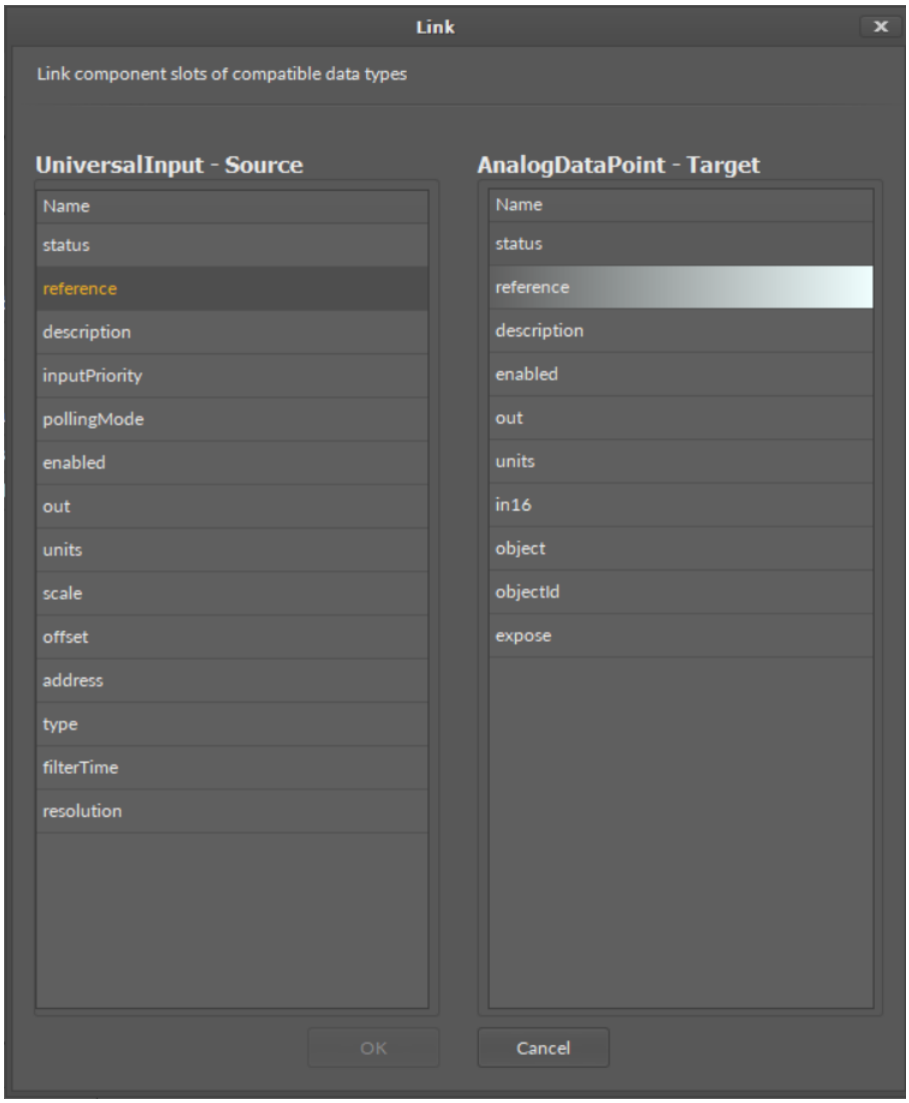


Figure 9. The linking dialog window

As the Reference links connect elements from the Applications container (Data Points) and the Networks container (network points), in the Wire sheet they are displayed as bubbles: on the left side of the Reference slot for the input-type linking, and on the right side of the Reference slot for the output-type linking:

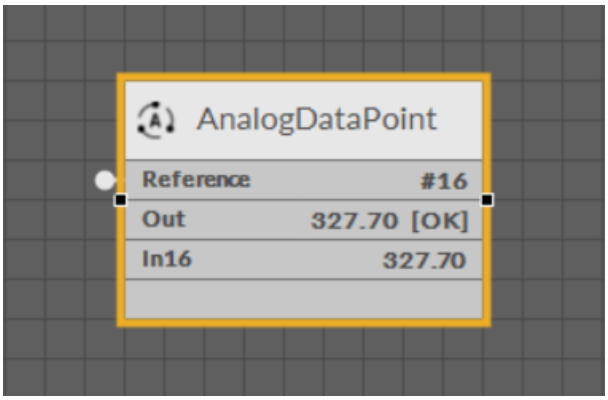


Figure 10. Analog Data Point linked from the Universal Input

The details of the Reference links are always visible in the Links tab in the Object Properties window of the linked element:

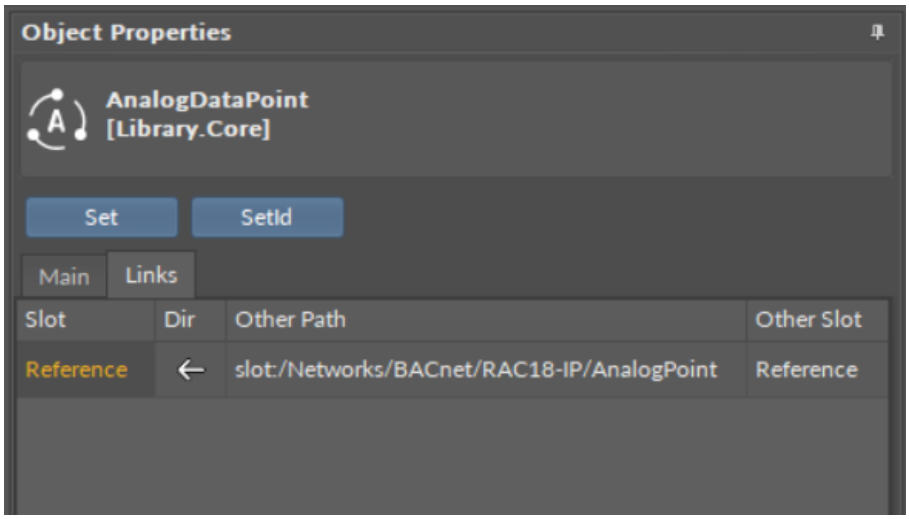


Figure 11. The Reference link path

5 Emergency Mode

The system and application(s) of the RAC18-IP controller are stored on an SD card. If the SD card is not detected in the device or the device detects frequent reboots (at least 5 times in 6 minutes), which prevent correct operation, the device enters an emergency mode.

What Causes the Emergency Mode?

- No SD card is detected in the device.
- The diagnostic process reveals error in I/Os.
- Storage limit is exceeded.
- Required files are missing during a start-up of the device.
- Libraries or files are corrupted.

5.1 Operation in Emergency Mode

In the emergency mode, the device operation is limited:

- libraries are not loaded;
- the SD card configuration is not loaded;
- only the System container with limited options (only Logs and Platform components) is displayed in the Workspace Tree;

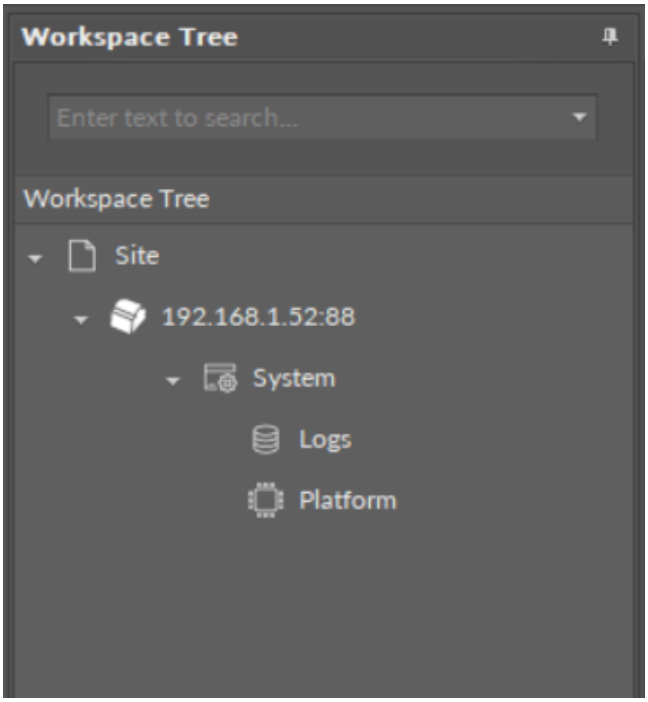


Figure 12. The device's Workspace Tree in the emergency mode

- the ALM LED is lit continuously;
- the iFnet runs with an IP/port taken from a flash storage;

Note: The flash storage must be synchronized to configuration slots when available.

- no authorization or credentials are taken from the flash storage (like IP/port).

When connecting to a device, which is in the emergency mode, a notification is displayed:

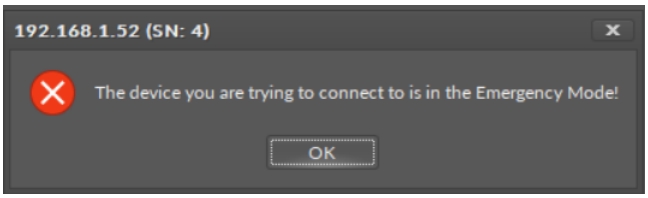


Figure 13. The emergency mode notification notice

5.2 Possible Actions

When the device enters the emergency mode, take one of a few possible actions:

- reboot;
- restore to factory defaults (restoring with S1 6th DIP switch): format the SD card (if available), restore default credentials, IP, mask, gateway, iFnet port;
- restore to defaults (restore in the iC Tool): remove files from the SD card (if available and formatted) excluding only files with IP, port, and credentials (libraries must be also removed);
- read logs from the SD card if available.

6 Applications Container

The first element in the Device structure is the Applications container. It is a non-removable element, which offers a space to create user applications.

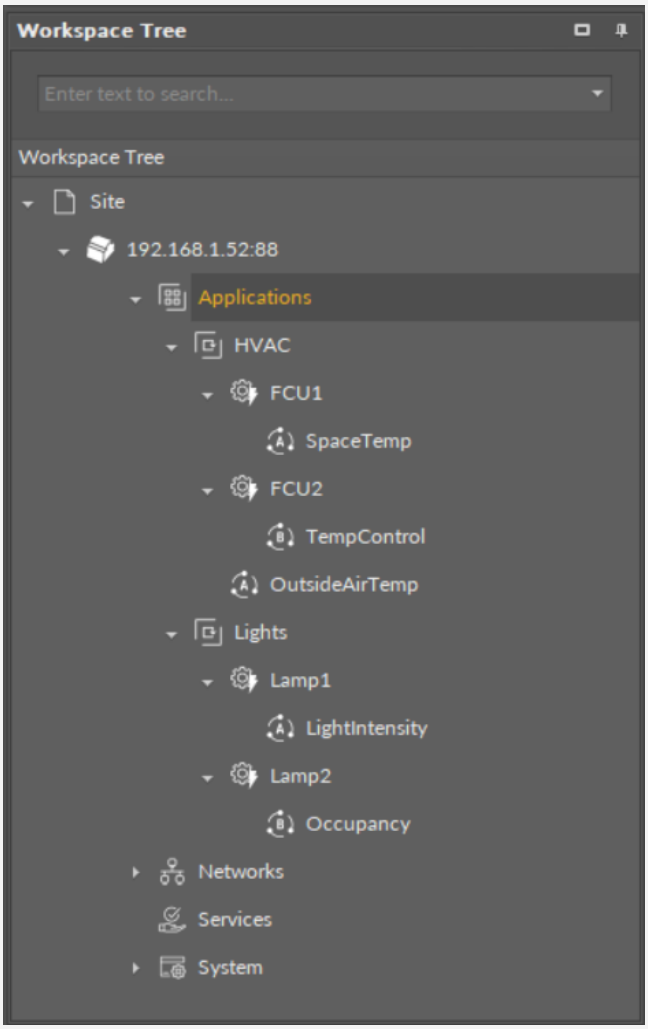
The Applications container allows to **add multiple Application components** for building independent user applications, which are **cycle driven and may work simultaneously**.

The user may define the application purpose (heating, lighting, etc.) and a cycle time of algorithms operation (cycles may differ between applications).

Tip

Multithreaded applications allow to differentiate cycles of applications (scan period) in order to adjust them according to a purpose of an application.

For example, a HVAC application may be set to 1000 ms scan period, and a Lights application to 200 ms scan period, as changes to lights operation have to be implemented immediately. Setting such scan periods would mean that the Lights application would be executed 5 times for each HVAC application cycle.



The user may create as many applications as needed—as long as the overall number of Data Points used within user applications does not exceed the device license.

Note: For top performance, it is recommended to use up to three different applications.

6.1 Applications Libraries

The nano EDGE ENGINE Applications libraries provide sets of components (service and basic) that allow to create HVAC applications. Libraries are grouped by specific functionalities, for example, by families of algorithms, etc. This section provides information about the nano EDGE ENGINE libraries designed for the Applications container, their functionalities and elements.

- Core library;
- FCU library;
- Logic library;
- Math library;
- Other library;
- Process library;
- Time library.

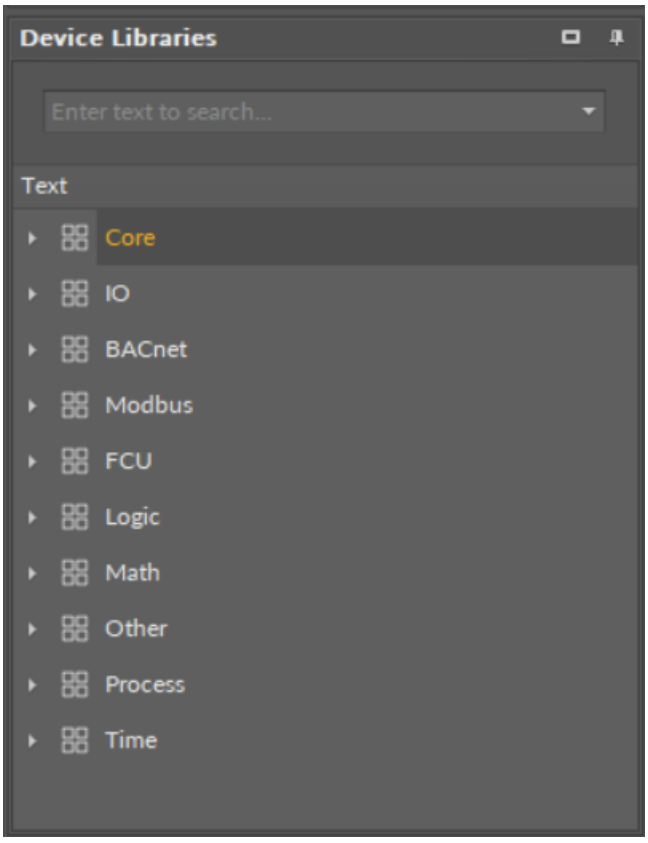


Figure 14. nano Edge Engine libraries

6.2 Basic Concept of Applications

The basic idea of the Applications container is to group the components meant to build applications.

Applications built with the nano EDGE ENGINE driven devices have unique characteristics:

- cycle-driven (the components included in the application are executed in repeated periods of time lasting as long as a cycle is set–**the duration of the application cycle can be set by the user, and may be even as short as 15 ms**)
- multithreaded (the user can have more than one application, all working simultaneously)
- **real-time operation** (the nano EDGE ENGINE is so fast that the applications are executed effectively real-time)
- **Data Points (with automatic BACnet exposition)** (basic components for applications, including priorities and allowing Reference linking)
- **Reference links** (unique linking method for Data Points and network points that allows to transfer the value along with the component's status)

6.2.1 Applications

In order to build universal applications, the place of reference is the Applications container. The first step is to place the Application component in the Applications container (or as many of them as necessary–with the nano EDGE ENGINE it is possible to create more than one user application on the device, and all of the applications are cycle-driven and can work simultaneously).

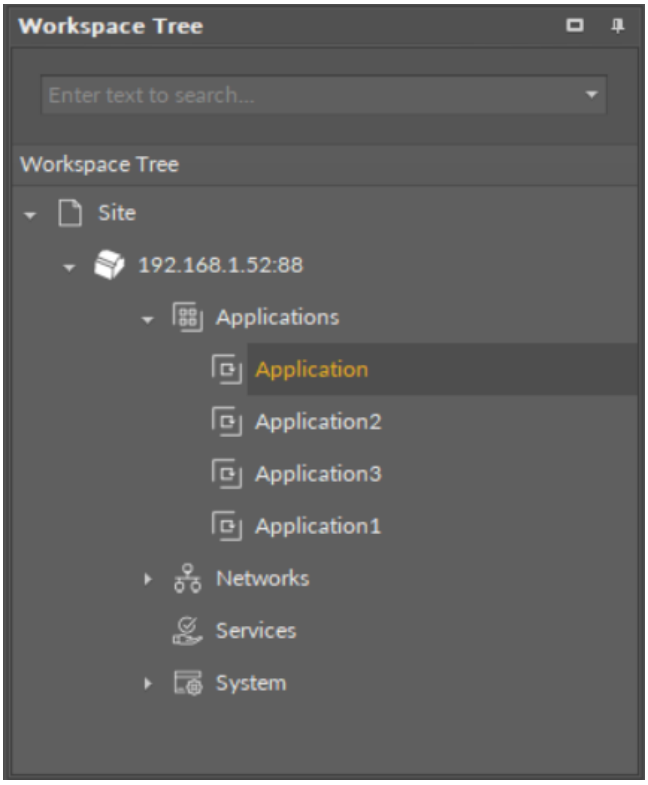


Figure 15. Multiple applications

Once the Application component is added to the Applications container, it is ready to have the Data Points added, and the user application with complementary components

can now be created. The components needed for this purpose are grouped in the Applications libraries, which are divided thematically:

- Core (for Applications) library (including Analog Data Point and Binary Data Point);
- Logic library;
- Math library;
- Process library;
- Time library;
- Other library;
- FCU library.

The distinctive thing about the Data Points used in the applications is that they are BACnet native–each Data Point contains a built-in BACnet extension, which allows to automatically expose it as a BACnet object to the network.

Using Equipment Folders

Good Practice Tip

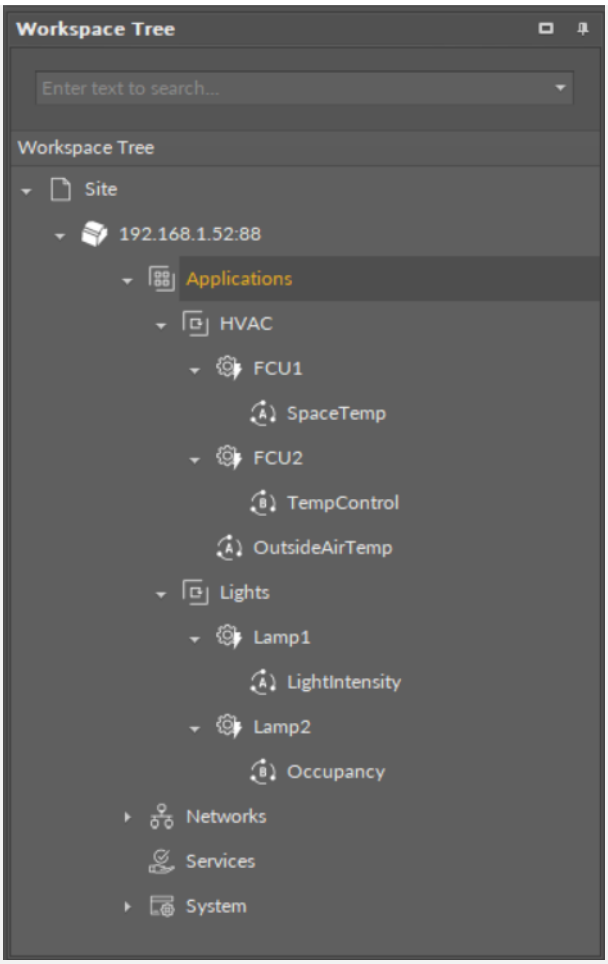
In future developments, the RAC18-IP controller will have a Haystack functionality implemented in its firmware, which calls for some good practice mechanisms that can be introduced now, with the current functionalities.

The Haystack functionality will offer multiple advantages of using tags–it will help identify parts of equipment controlled by the application and used Data Points, filter data by equipment, sensor, or value, any many more.

In order to facilitate a future use of the full Haystack functionality, the Equipment component has been included in the Core library. It is therefore recommended to use the following structure when creating applications:

- Applications container
  - Application component
    - Equipment component
      - Data Point(s)
      - other components
    - Equipment component
      - Data Point(s)
      - other components

Such structure will be fully recognized in the Haystack functionality and will require minimum effort to use its full potential once updated.



Difference Between Equipment and Folder Components

The Equipment component with Haystack functionality will allow to identify the types and other characteristics of controlled devices by tags, which will be a main difference between the Equipment and Folder components–the Folder component is merely an organizing component; in the future, it will not carry any Haystack tags functionalities. It is therefore very important to use the Equipment components in the applications structure.

Also refer to: Equipment Manager

6.2.2 Data Points

Data Points are universal components that represent a value in the application logic; they may serve as setpoints, sensors values, non-volatile variables, or any other data values. Data Points represent a layer of the application logic that is presented to an end user—this is where the end user is able to adjust desired setpoints (e.g., for air conditioning) or evoke other actions outlined in the application logic. Data Points also read values calculated in applications and control local or remote outputs.

Data Points in the application logic may work as regular writable variables with priorities, or—with Reference linking—they may be connected with network points, such as local I/O components.

Data Points are universal components that facilitate building application logics. They are adaptable by extensions, which expand their functionality by enabling them to be exposed to communication protocols. Among the fundamental extensions for Data Points are AnalogPriorities and BinaryPriorities—they add 16 priority input slots to Data Points, which allows to build logics based on using value significance priorities—a value given in the highest priority slot dismisses all values of lower priorities, which enables, for example, making sure that no emergency is omitted due to values of lower significance. Slots, which are not linked and have a value entered, save the value on a given priority.

The available Data Points:

- Analog Data Point with native BACnetAnalogPoint extension;
- Binary Data Point with native BACnetBinaryPoint extension.

In order to operate properly, Data Points must be placed in one of Application components in the Applications container. Only from this position they may work properly included in the application logic.

Data Points and Network Points

Simply put, in the nano EDGE ENGINE, Data Points are value-carriers, and network points are communication-carriers. Data Points are always located in the Applications container (in individual Application components), and network points are always located in the Networks container (in the LocalIO component or BACnet/Modbus devices). The below table lists nano EDGE ENGINE component referred to as Data Point– and network point–class components:

Data Point class components	Network point class components
AnalogDataPoint BinaryDataPoint	UniversalInput (LocalIO) DigitalInput (LocalIO) DigitalInputCounter (LocalIO) AnalogOutput (LocalIO) DigitalOutput (LocalIO) TriacOutput (LocalIO) DipSwitch (LocalIO) AnalogPoint (BACnet/Modbus Device) BinaryPoint (BACnet/Modbus Device) StringPoint (Modbus Device)

Table 2. Data Point– and network point– class components

6.3 Core (for Applications)

Applicable to library's version 1.0

WARNING!

The Core library is a library containing components featured for both Applications and Networks containers. In this section only components for the Applications container are described!

For the Applications container, the Core library includes components that are essential for building applications:

- the Application component itself,
- Data Points,
- and folders.

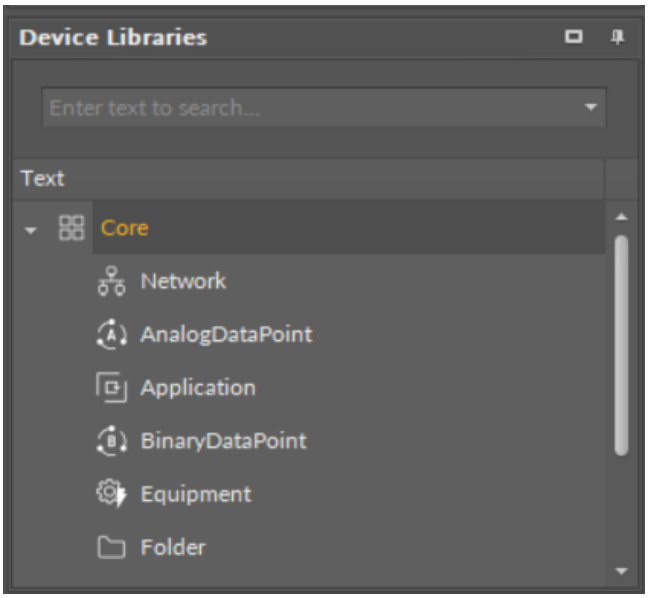


Figure 16. The Core library

6.3.1 Application

Applicable to library's version 1.0

The Application component is a component designed to create a user application. By default, the Applications container includes one Application component. However, the user may add as many Application components in the container as needed. The user application is constructed with Data Points and other components (though, it is not a requirement to include Data Points in each user application). The number of Data Points used across user applications must not exceed the license limit. Each Application component allows to create one independent and cycle driven user application.

The Application component can added to the Applications container in two ways:

- in the Applications Manager;
- dragging and dropping it directly from the Core library in the Device Libraries window.

The user application is created in the Wire Sheet view.

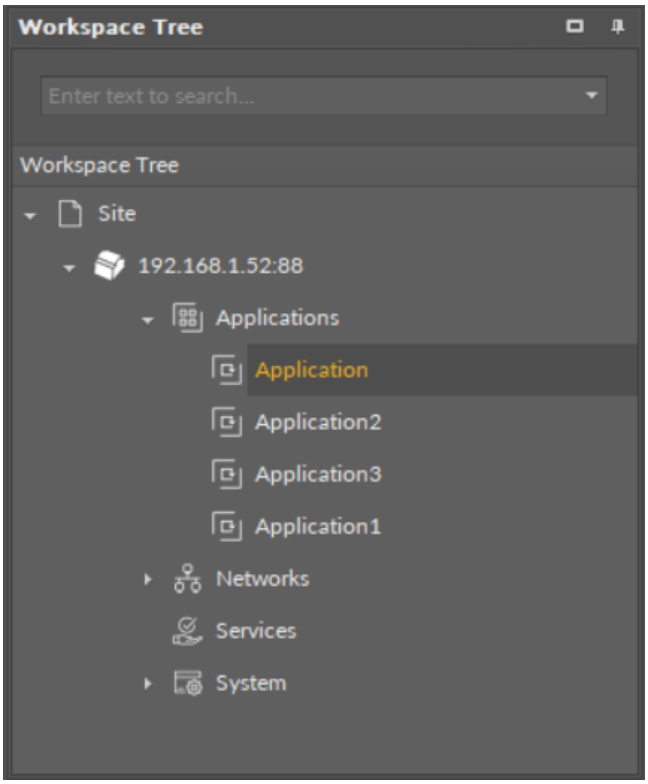


Figure 17. The Application components in the Applications container

The Application component operates in one special view—the Application Manager—and the standard iC Tool views—the Wire Sheet and Property Sheet. The Application Manager allows to add, remove, copy, or duplicate Data Points and the Folder or Equipment components and to add or remove Data Points' extensions. The Property Sheet view presents a list of configurable slots and read-only slots, which provide essential information about the work of the component. Application building is carried out in the Wire Sheet view.

The Wire Sheet view presents Data Points included in the application and other components that introduce dependencies between Data Points. Data Points may be defined for input or output network points and be connected to other components creating the application logics.

Linking within the applications in the iC Tool may be performed twofold: using a standard linking method or using a special Reference link designed specifically to connect Data Point class components with network point class components:

- The Reference link is a special compound link designed to connect Data Points with network points. The Reference link is created between special Reference slots and **transfers values along with the component's status**. Alternatively, it may transfer values between Data Points and network points **at the same time returning status from network points to Data Points**, or it may **return values from network points to Data Points**. As network points are situated in the Networks container and Data Points are situated in the Applications container, Reference links are created using the Link Mark and Link From options from the context menu, and they are created between the tabs (or, for example, between the Application tab and the network points expanded in the Workspace Tree window). Either way, the Reference link between tabs is displayed in the Wire Sheet as a bubble connected to the component's Reference slot.

- The standard linking method involves simple creating links between the input and output slots; a standard link will transfer a value between the connected slots. Such link may either be drawn from one component to another in the Wire Sheet or be created using the Link Mark and Link From option from the context menu. Standard linking may be applied between all four containers of the nano EDGE ENGINE device structure.

The Reference link is unique for the nano EDGE ENGINE solution, and it is a recommended method to be used in application building as it offers an advantage of transferring the status information between components (find out more about Reference links: Linking).

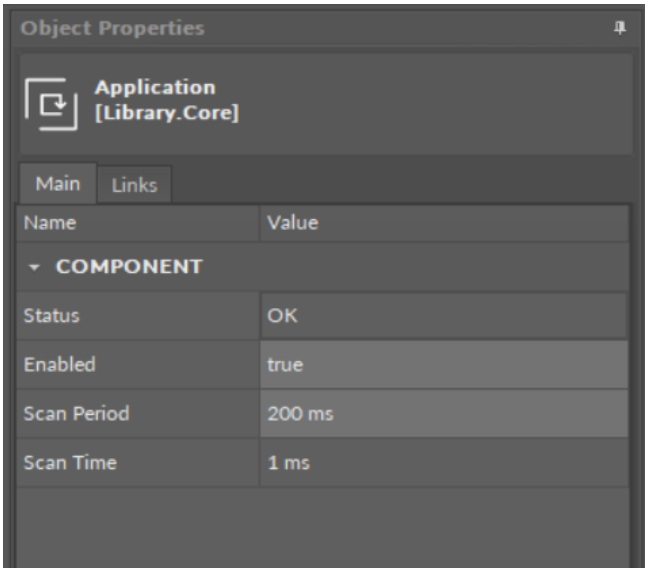


Figure 18. The Application component slots

Slots

The Application component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its Status is OK;
  - Available information: Disabled, Fault, OK;
- **Enabled:** starts and stops execution of the application contained in the component;
- **Scan Period:** defines a maximum intended duration of an application cycle; set in milliseconds (ms);
- **Scan Time:** an actual time used by the controller to perform a full cycle of an application, read from the CPU; it is desired that the Scan Time is lower than the Scan Period–it means that the application works efficiently and the CPU is not overloaded.

6.3.2 Analog Data Point

Applicable to library's version 1.0

The Analog Data Point component is a component that represents a numerical value in the application; it reads analog values calculated in applications and controls local or remote analog outputs.

The Analog Data Point allows to create standard links to all inputs and outputs; it may transfer a value via a link from the Out slot to other components. The other method to link the Analog Data Point is via a Reference link. The Reference link is a special compound link designed to connect Data Points with network points. The Reference link is created between special Reference slot in the network point and transfers value along with its status to Analog Data Point. Also, it may transfer values from Analog Data Points to network points at the same time returning status from network points to Analog Data Points, and it may return values from network points to Analog Data Points too. While linking Data Points and network points, it is recommended to use the Reference linking method.

In order to operate properly the component must be placed in the Application component in the Applications container.

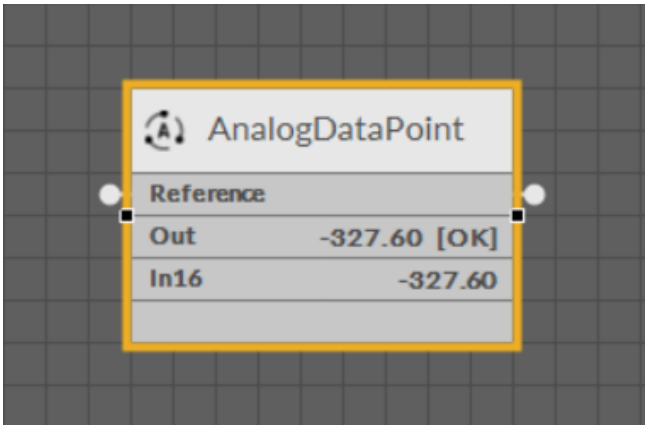


Figure 19. The Analog Data Point on the Wire Sheet view

The Analog Data Point can operate in two modes. First, it is a basic mode with one writable input slot (In16), one output slot (Out), Reference linking, and other basic slots: Units, Status, Enabled, Description. In the basic mode, the Analog Data Point has a native BACnet Analog Point extension.

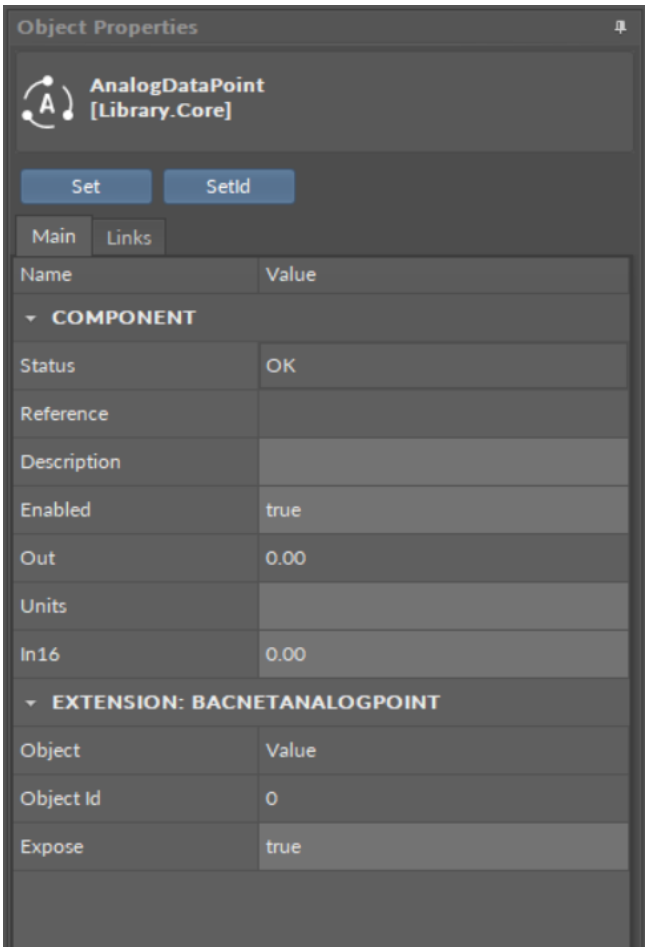


Figure 20. The Analog Data Point slots

Analog Data Point's Slots

The Analog Data Point has the following basic slots:

- **Status:** indicates the current status of the component. If the component works properly, its status is OK; however, it changes accordingly when values in other slots are adjusted.
    - Available information: Disabled, Unlicensed, Error, Overriden, OK;

Note: If the referenced network point goes into a Fault or Down status, the Data Point's status will show Error in order to communicate that the value in the Out slot may be invalid.

- **Reference:** a special slot allowing to transfer more than one value with one link (e.g., the Out slot value and the component's Status); connects Data Points with network point class components.

Note: The Data Point allows to create only one Reference connection leading to it and many Reference connections leading from it.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding defined in the user's system documentation or any other information the user finds applicable.

- **Enabled:** change of the slot's value enables or disables the component–if the component becomes disabled, it stops reading values from or transferring values to the linked network points. By default, the component is enabled.
    - Available settings: true (enabled), false (disabled).
- **Out:** shows a value transferred from the In16 slot in the basic mode, or from the first non-null input with highest priority in the extended mode; in case there are values on different priorities, only the value from the highest priority slot is transferred to the Out slot, the rest is dismissed.

Note: In the Wire Sheet the Out slot is visible as combined slots Out, Units, Priority (which is a source of the Out slot value; shown in the extended mode), and Status.

- **Units:** defines a unit of the Out slot value; units are consistent with BACnet requirements. The unit is entered manually from a drop-down list in accordance with a value type in the Out slot;
- **In16:** the basic input slot; receives numeric values; the In16 value may be set by a Set action;

The Analog Data Point has one basic action:

- **Set:** allows entering a numeric value to set the In16 slot;
- **SetId:** sets a BACnet object Id of the Analog Data Point.

The extended mode of the Analog Data Point is switched on by adding the AnalogPriorites extension to the Analog Data Point component. The extension adds 16 writable input slots to the Data Point. It is added by right-clicking on the Analog Data Point component (either in the Wire Sheet or Property Sheet view).

Extensions

Data Points can have their functionality modified by extensions. The Analog Data Point is originally equipped with the BACnet Analog Point extension (this one cannot be added or removed), but other extensions, which offer different functionalities, can be added or removed as necessary. Extensions are added by right-clicking the Analog Data Point, either in the Wire Sheet or Property Sheet view, Application Manager, or in the Workspace Tree.

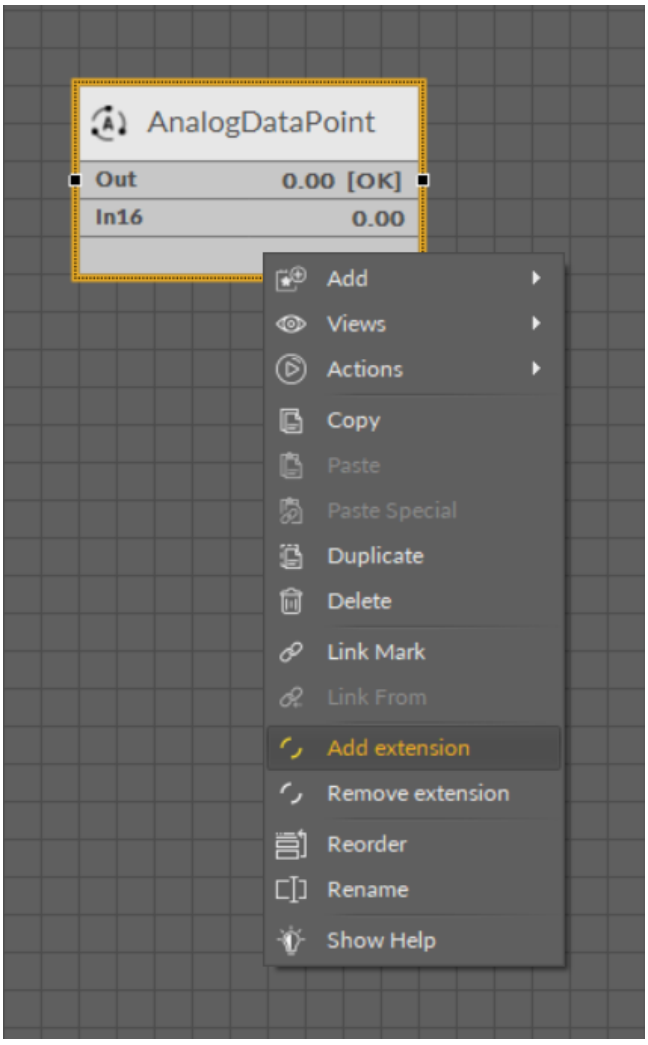


Figure 21. Adding extension in the Wire Sheet view

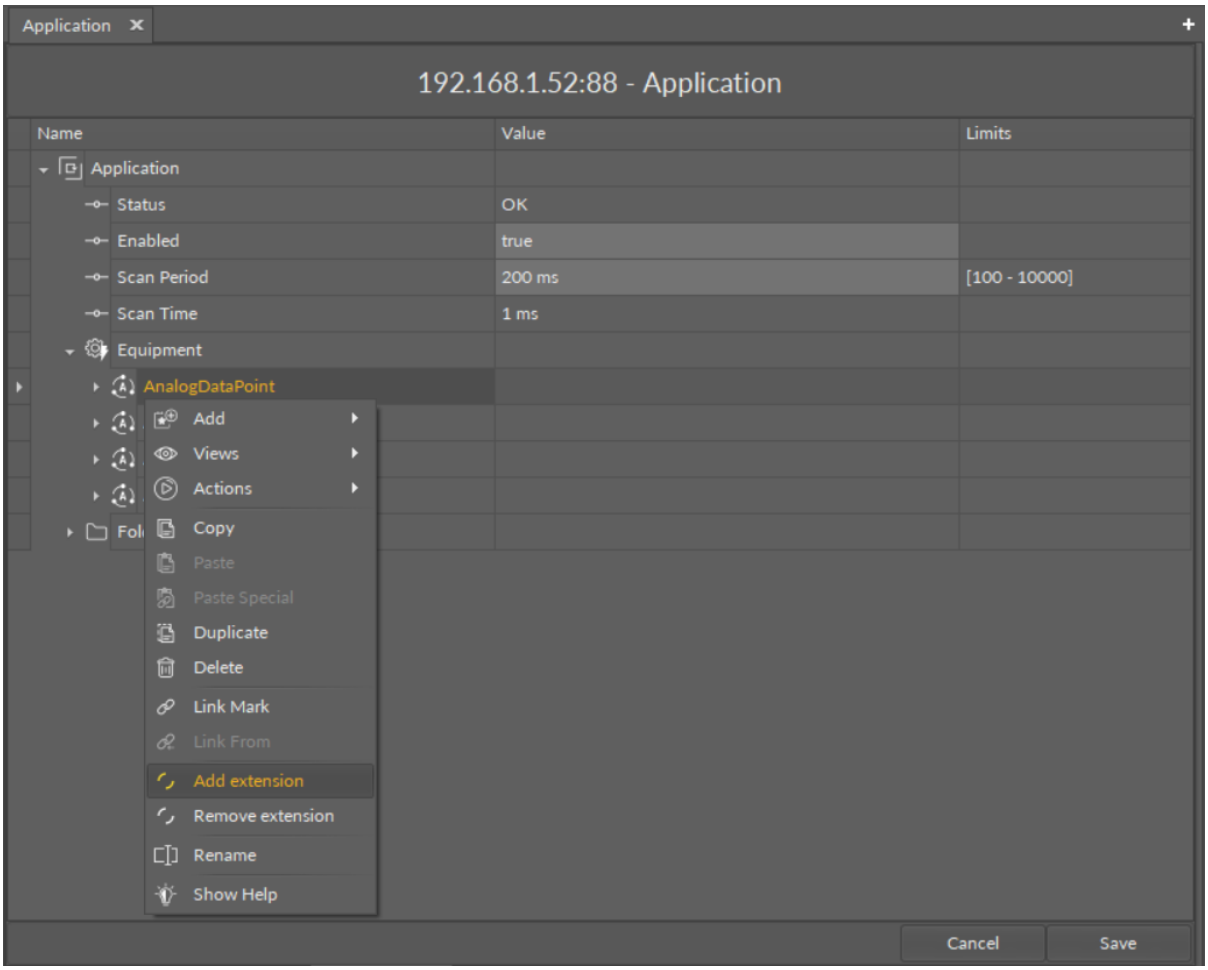


Figure 22. Adding extension in the Property Sheet view

From the context menu, select the Add Extension option. The pop-up window appears allowing to choose an extension to add.

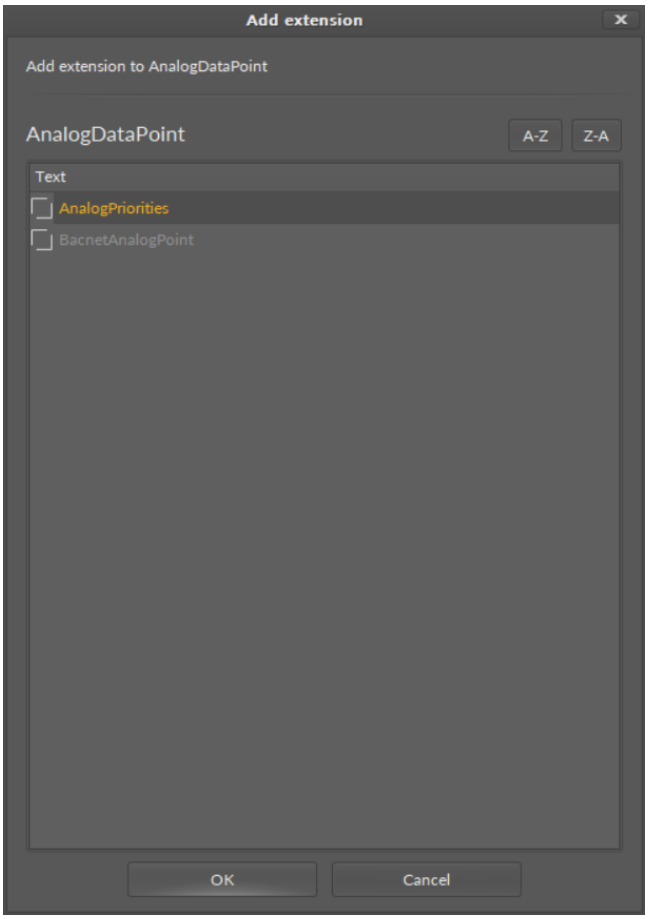


Figure 23. Adding extension

BACnetAnalogPoint Extension

The BACnetAnalogPointExtension expands the Analog Data Point's functionality giving it an option to expose it to the BACnet network as an Analog Value object, and otherwise, it allows to manually hide it from the network yet preserving its function in the application. It also transfers information to the BACnet network about the Analog Data Point's status. The extension is native (cannot be removed) and is visible along with the regular slots and actions of the Analog Data Point as a separate, integral part in the Object Properties view.

The extension has the following slots:

- **Object:** a read-only slot showing a type of BACnet object attributed to the extension;
- **ObjectID:** a BACnet object ID, which is automatically numbered from 0 up;
- **Expose:** enables the Data Point to be recognized within the BACnet network;
    - Available settings: true (exposed), false (hidden).

Apart from the BACnetAnalogPoint, it is possible to add the following extensions to the Analog Data Point:

AnalogPriorities Extension

The AnalogPriorities extension adds fifteen writable input slots and the default (lowest) priority slot to the Analog Data Point. The extension includes the Priority slot indicating, which slot is transferring value to the Out slot. The AnalogPriorites extension adds In1–In15 slots and the Default slot, which is the lowest, 17th priority. The extension also introduces new actions to the Data Point: EmergencyOverride, EmergencyAuto, Override, and OverrideAuto.

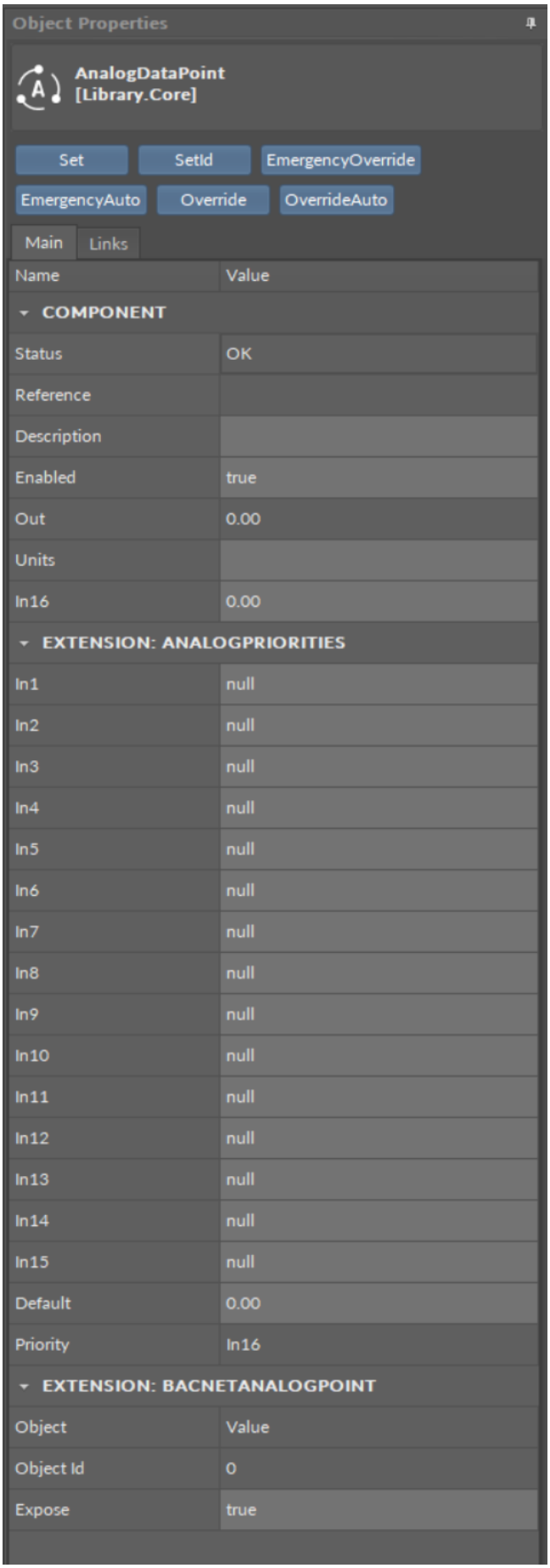


Figure 24. The AnalogPriorites extension

The Analog Data Point has the following slots available in the AnalogPriorities extension:

- In1–In15: input slots providing values to the Out slot (from 1 to 16, the highest priority is In1); only the highest priority value is provided to the Out slot, the rest is dismissed. All input slots are linkable. In the extended mode, the In1 and In8 slots have actions available for overriding their values.

Note: By default, only the In16 is displayed in the Wire Sheet. In case any other input slot receives a value via link, is it displayed in the Wire Sheet along with the In16. Only the null input, which is a lack of value, allows the higher priority input to be dismissed–zero (0) is still a value that will be provided to the Out slot.

- Default: the 17th, lowest priority input slot; allows to introduce a default value to the Data Point in case there are no links providing values from other components. If the value to the Data Point is provided by the Reference link, then the Default value is automatically dismissed (the Reference link cannot be directed to the 17th priority, only from the 16th up).

Note: According to BACnet requirements, the Default slot value can never be null; if no other value is set on the slot, it is zero (0).

- Priority: shows, which slot is currently providing the value to the Out slot.

The Analog Data Point has the following actions available in the AnalogPriorities extension:

- EmergencyOverride: enables entering a numeric value to the In1 slot;
- EmergencyAuto: sets the null value to the In1 slot (cancels the EmergencyOverride action);
- Override: enables entering a numeric value to the In8 slot;
- OverrideAuto: sets the null value to the In8 slot (cancels the Override action).

Note: If the link is connected to the slot that may be affected by an action, the value coming from the link connection has priority over the manually evoked action.

6.3.3 Binary Data Point

Applicable to library's version 1.0

The Binary Data Point component is a component that represents a Boolean value in the application; it reads binary values calculated in applications and controls local or remote binary outputs. The Binary Data Point allows to create standard links to all inputs and outputs; it may transfer a value via a link from the Out slot to other components. The other method to link the Binary Data Point is via a Reference link. The Reference link is a special compound link designed to connect Data Points with network points. The Reference link is created between special Reference slots in network points and transfers values along with the its status to Binary Data Points. Also, it may transfer values from Binary Data Points to network points at the same time returning status from network points to Binary Data Points, and it may return values from network points to Binary Data Points too. While linking Data Points and network points it is recommended to use the Reference linking method.

In order to operate properly, the component must be placed in the Application component in the Applications container.

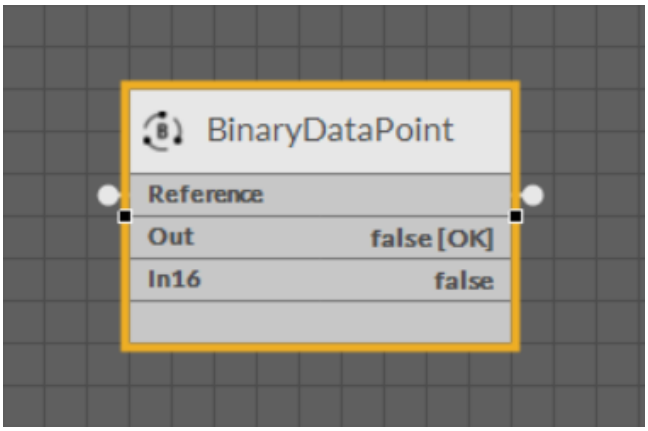


Figure 25. Binary Data Point

The Binary Data Point can operate in two modes. First, it is a basic mode with one writable input slot (In16), one output slot (Out), Reference linking, and other basic slots: Active Text, Inactive Text, Status, Enabled, Description. In the basic mode, the BACnet Data Point has a native BACnetBinaryPoint extension.

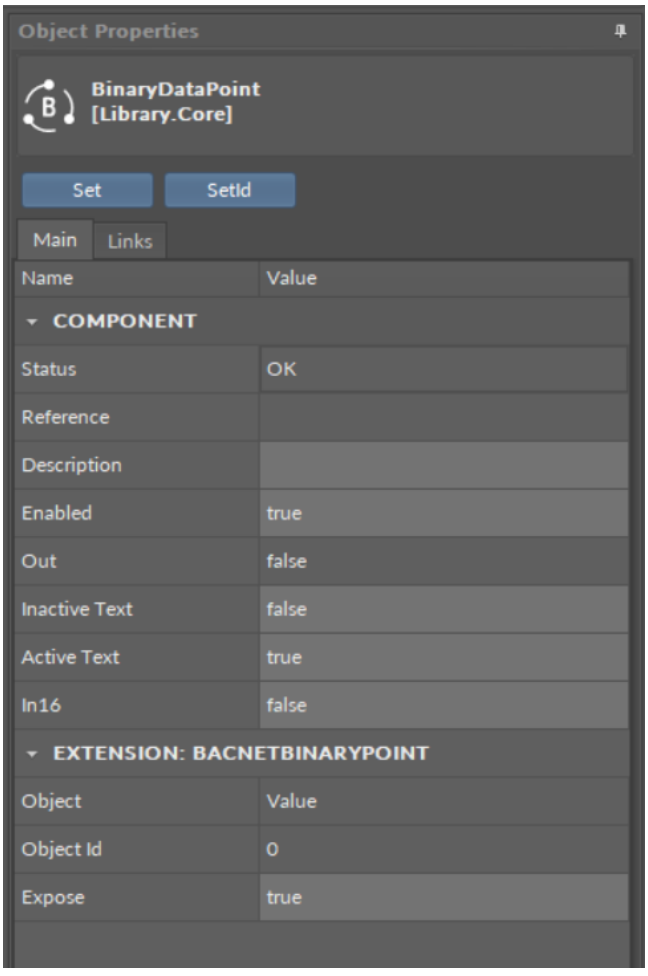


Figure 26. The Binary Data Point component in simple variant

In its basic mode, the Binary Data Point has one action: Set (which allows to set a value to the In16 slot).

The Binary Data Point has the following basic slots:

- **Status:** indicates the current status of the component. If the component works properly, its status is OK; however, it changes accordingly when values in other slots are adjusted;
    - Available information: Disabled, Unlicensed, Error, Overriden, OK;

Note: If the referenced network point goes into a high priority status (e.g., fault or down), the Data Point's status will show Error in order to communicate that the value in the Out slot may be invalid.

- **Reference:** a special slot allowing to transfer more than one value with one link (e.g., the Out slot value and the component's Status); connects Data Points with network point class components;

Note: The Data Point allows to create only one Reference connection leading to it and many Reference connections leading from it.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding defined in the user's system documentation or any other information the user finds applicable;
- **Enabled:** change of the slot's value enables or disables the component–if the component becomes disabled, it stops reading values from or transferring values to the linked Network Points. By default, the component is enabled;
    - Available settings: true (enabled), false (disabled);
- **Out:** shows a value of the first non-null input with highest priority; in case there are values on different priorities, only the value from the highest priority slot is transferred to the Out slot, the rest is dismissed;

Note: In the Wire Sheet, the Out slot is visible as combined slots Out, Priority, and Status. The Out slot displays the value in accordance with the settings in the ActiveText and InactiveText slots.

- **InactiveText:** a text value that is displayed in the Out slot if its value is false; the text is freely adjustable by the user. By default, this field is empty and "false" is displayed for a false value;
- **ActiveText:** a text value that is displayed in the Out slot if its value is true; the text is freely adjustable by the user. By default, this field is empty and "true" is displayed for a true value;
- **In16:** the basic input slot; receives Boolean values; the In16 value may be set by a Set action.

The Binary Data Point has one basic action:

- **Set:** allows entering a Boolean value to set the In16 slot;
- **SetId:** sets a BACnet object Id of the Binary Data Point.

The extended mode of the Binary Data Point is switched on by adding the BinaryPriorites extension to the Binary Data Point component. The extension adds 16 writable input slots to the Data Point. It is added by right-clicking on the Binary Data Point component (either in the Wire Sheet or Property Sheet view, Application Manager, or the Workspace Tree).

Extensions

Data Points can have their functionality modified by extensions. The Binary Data Point is originally equipped with the BACnet Binary Point extension (this one cannot be added or

removed), but other extensions, which offer different functionalities, can be added or removed as necessary. Extensions are added by right-clicking the Binary Data Point, either in the Wire Sheet or Property Sheet view, Application Manager, or the Workspace Tree.

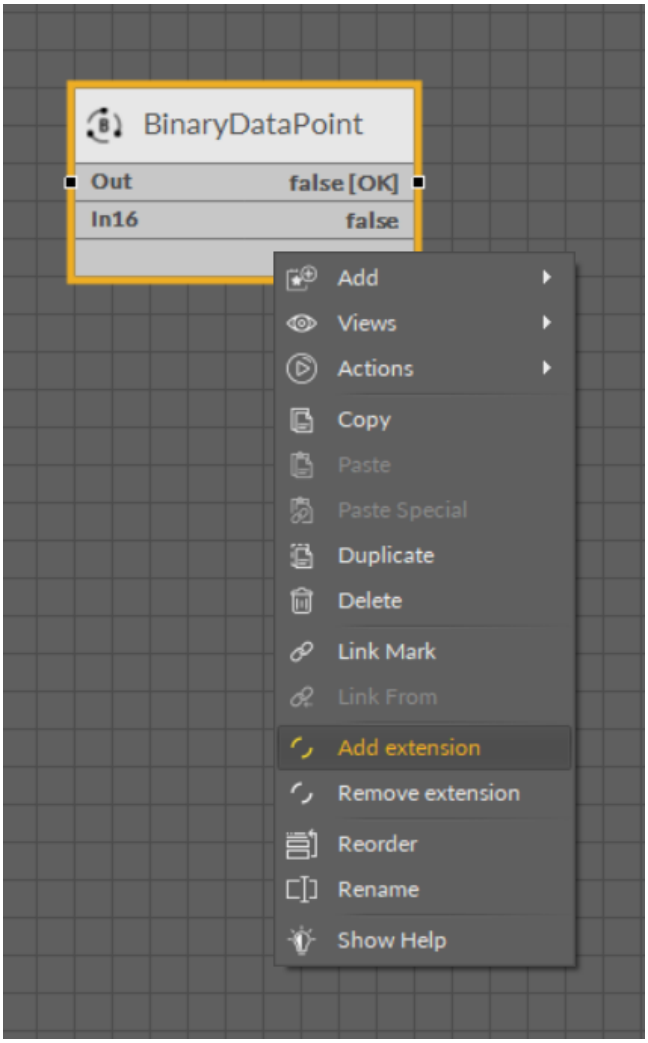


Figure 27. Adding extension in the Wire Sheet view

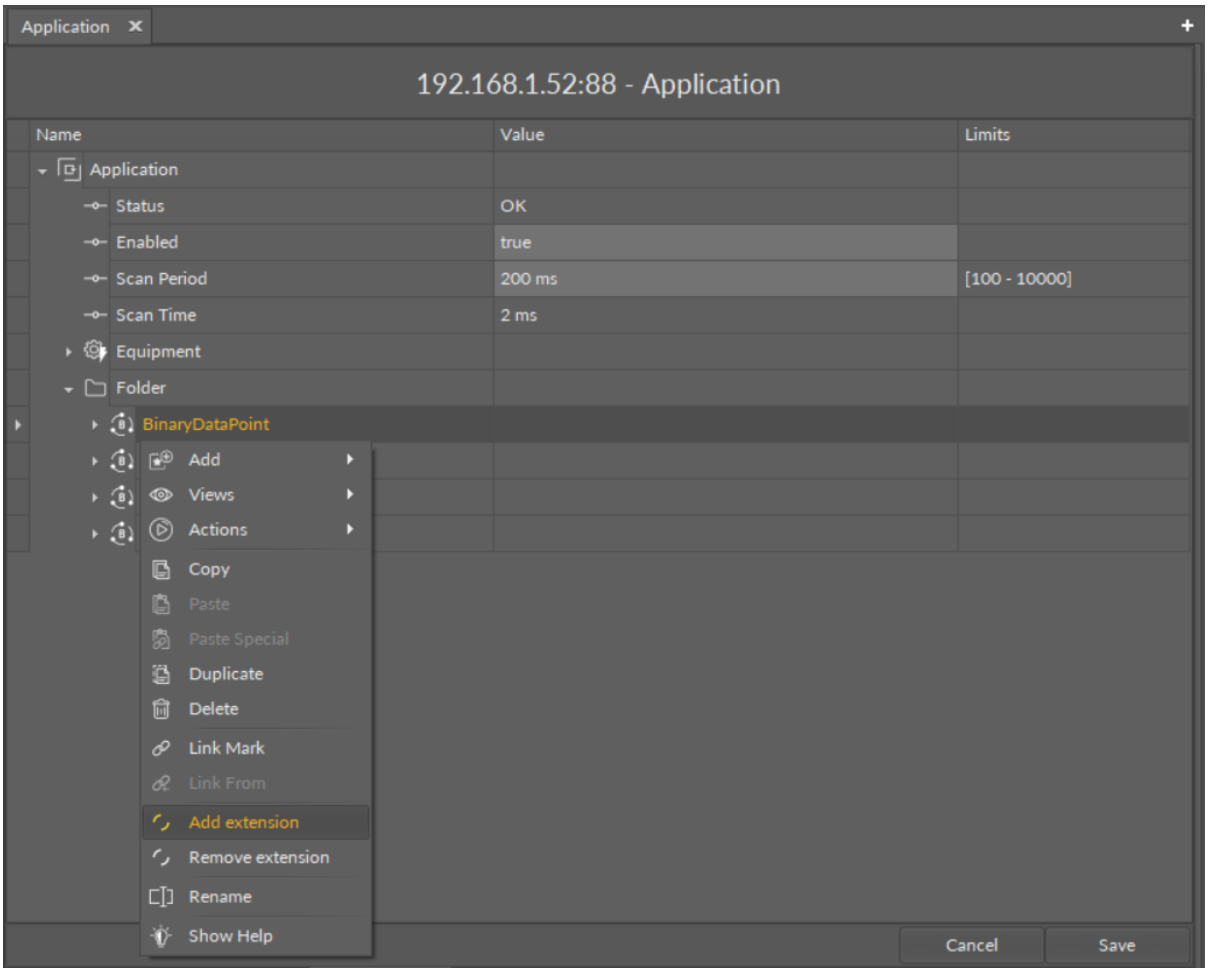


Figure 28. Adding extension in the Property Sheet view

From the context menu, select the Add extension option. The pop-up window appears allowing to choose an extension to add.

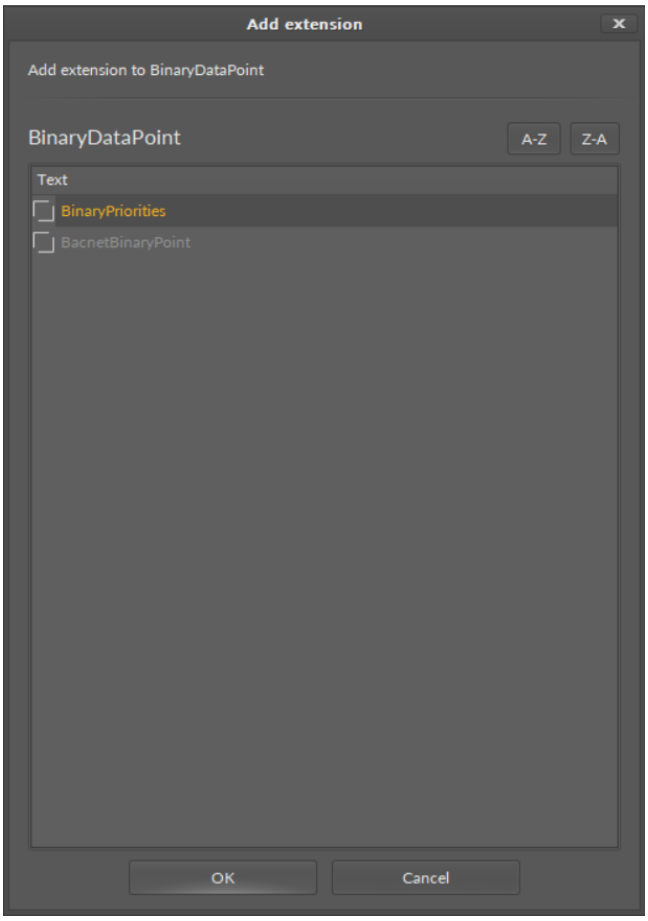


Figure 29. Adding extension dialog window

BACnetBinaryPoint Extension

The BACnetBinaryPoint extension expands the Binary Data Point's functionality giving it an option to expose it to the BACnet network as a Binary Value object, and otherwise, it allows to manually hide it from the network yet preserving its function in the application. It also transfers information to the BACnet network about the Binary Data Point's. The extension is native (cannot be removed), and is visible along with the regular slots and actions of the Binary Data Point as a separate, integral part in the Object Properties view.

The extension has the following slots:

- **Object:** a read-only slot showing a type of BACnet object attributed to the extension;
- **ObjectID:** a BACnet object ID, which is automatically numbered from 0 up;
- **Expose:** enables the Data Point to be recognized within the BACnet network;
    - Available settings: true (exposed), false (hidden).

BACnetPriorities Extension

The BinaryPriorites extension adds In1–In15 slots and the Default slot, which is the lowest, 17th priority. The extension also introduces new actions to the Data Point: EmergencyOverride, EmergencyAuto, Override, and OverrideAuto. The BinaryPriorities extension adds fifteen writable input slots and the default priority slot to the Binary Data Point. The extension includes also the Priority slot indicating, which slot is transferring value to the Out slot.

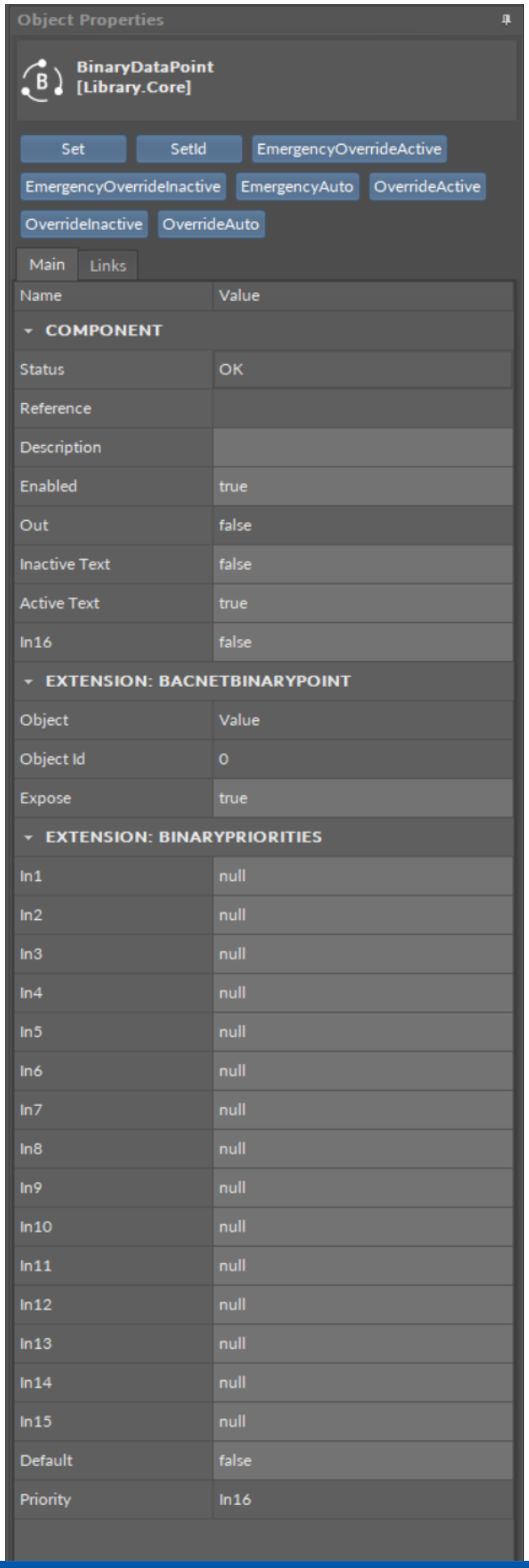

Object Properties
BinaryDataPoint
[Library.Core]
Set
SetId
EmergencyOverrideActive
EmergencyOverrideInactive
EmergencyAuto
OverrideActive
OverrideInactive
OverrideAuto
Main
Links
Name
Value
COMPONENT
Status
OK
Reference
Description
Enabled
true
Out
false
Inactive Text
false
Active Text
true
In16
false
EXTENSION: BACNETBINARYPOINT
Object
Value
Object Id
0
Expose
true
EXTENSION: BINARYPRIORITIES
In1
null
In2
null
In3
null
In4
null
In5
null
In6
null
In7
null
In8
null
In9
null
In10
null
In11
null
In12
null
In13
null
In14
null
In15
null
Default
false
Priority
In16

Figure 30. The extended variant of the Binary Data Point

The Binary Data Point has the following slots available in the BinaryPriorities extension:

- **In1–In15:** input slots providing values to the Out slot (from 1 to 16, the highest priority is In1); only the highest priority value is provided to the Out slot, the rest is dismissed. All input slots are linkable. The In1 and In8 slots have actions available for emergency overriding their values;

Note: By default, only In16 is displayed in the Wire Sheet. In case any other input slot receives a value via link, is it displayed in the Wire Sheet along with the In16. Only the null value allows the higher priority input to be dismissed.

- **Default:** the 17th, lowest priority input slot; allows to introduce a default value to the Data Point in case there are no links providing values from other components. If the value to the Data Point is provided by the Reference link, then the Default value is automatically dismissed (the Reference link cannot be directed to the 17th priority, only from the 16th up).

Note: According to BACnet requirements, the Default slot value can never be null; if no other value is set on the slot, it is false.

- **Priority:** shows which slot is currently providing the value to the Out slot.

The Binary Data Point has the following actions available in the BinaryPriorities extension:

- **EmergencyOverrideActive:** enables entering a true value to the In1 slot;
- **EmergencyOverrideInactive:** enables entering a false value to the In1 slot;
- **EmergencyAuto:** sets the null value to the In1 slot (cancels the EmergencyOverrideActive and EmergencyOverrideInactive action);
- **OverrideActive:** enables entering a true value to the In8 slot;
- **OverrideInactive:** enables entering a false value to the In8 slot;
- **OverrideAuto:** sets the null value to the In8 slot (cancels the OverrideActive and OverrideInactive actions).

6.3.4 Equipment

Applicable to library's version 1.0

The Equipment component is a grouping folder-type component, which can be used to gather other components, regarding specific equipment included in the Application, to help organize the Workspace Tree. The Equipment can be added to the device structure, however, it cannot be added directly to the container. The Equipment component can be freely renamed to facilitate categorization of components included within.

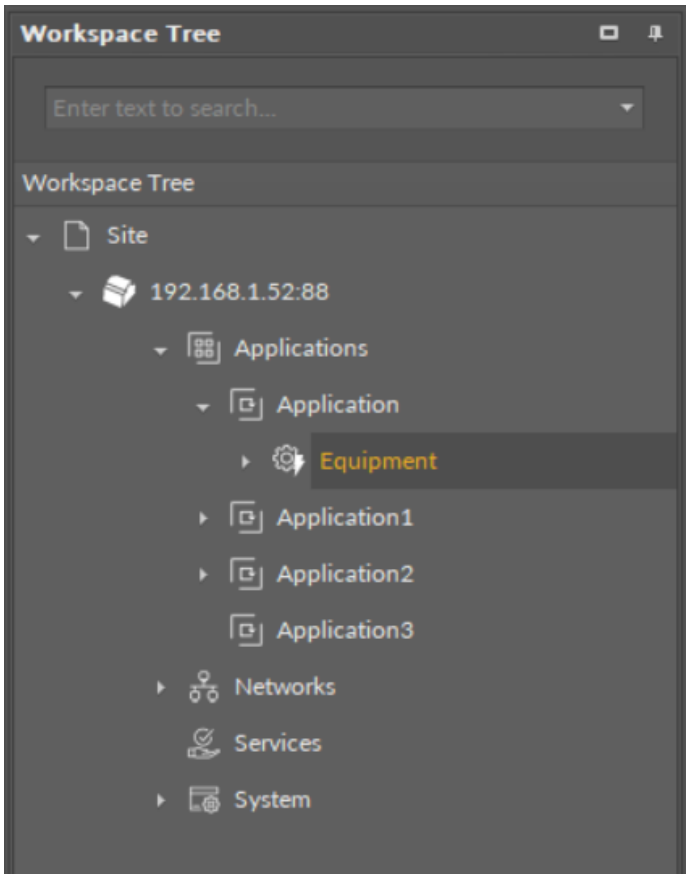


Figure 31. The Equipment component

The Equipment component has no slots nor actions.

6.3.5 Folder

Applicable to library's version 1.0

The Folder component is a grouping component, which can be used to gather other components to help organize the Workspace Tree. The Folder can be added to the device structure, however, it cannot be added directly to the container. The Folder component can be freely renamed to facilitate categorization of components included within.

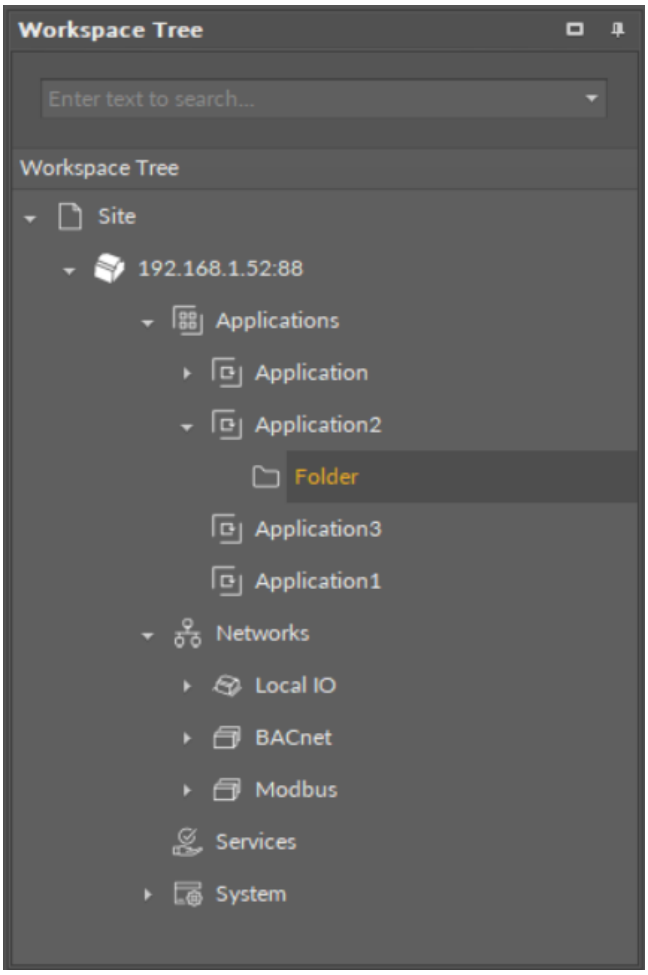


Figure 32. The Folder component

The Folder component has no slots nor actions.

Note: The Folder component can be used both in the Applications and Networks containers.

6.4 Logic

Applicable to library's version 1.0

The Logic library contains components representing logical operations essential for creating applications. The component represent logical operations such as conjunction, alternative denial, joint denial, disjunction, or simple comparisons and more.

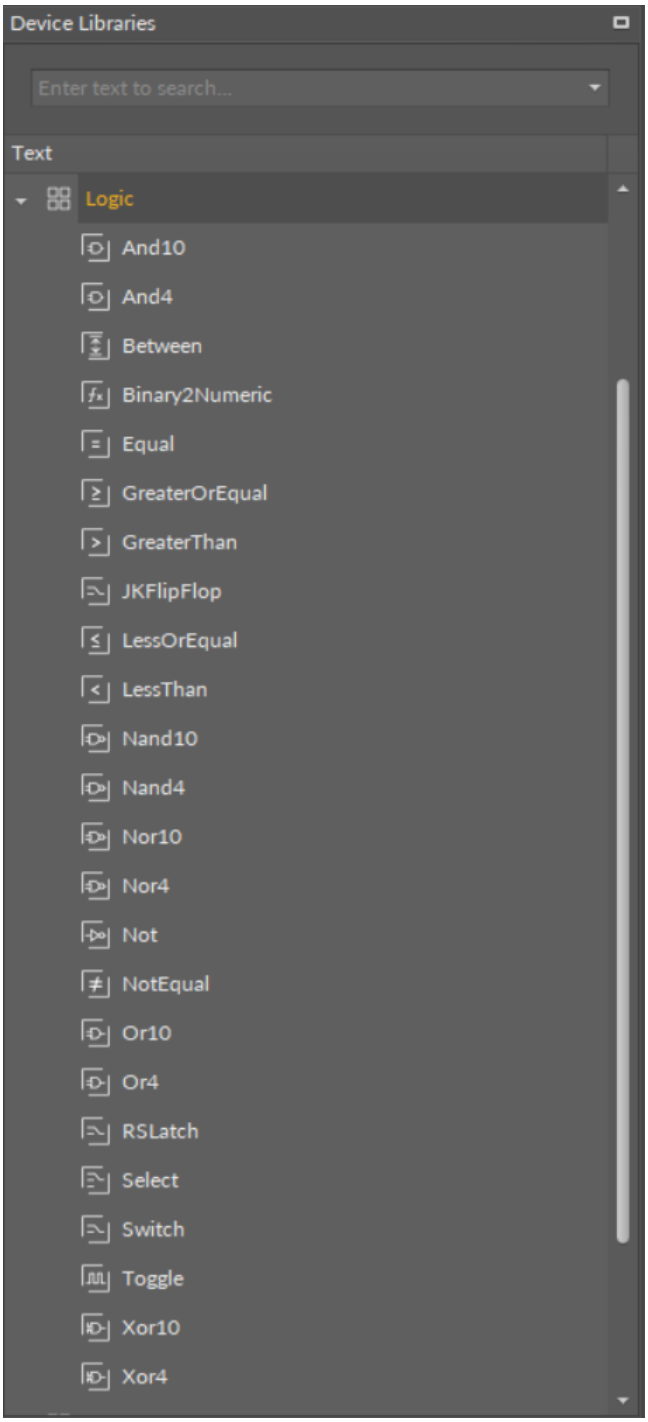


Figure 33. The Logic library

6.4.1 And

Applicable to library's version 1.0

The And component performs a logical conjunction—the Out slot value is true if and only if all active input values are true; in case any of the In slots is false, the Out value is false.

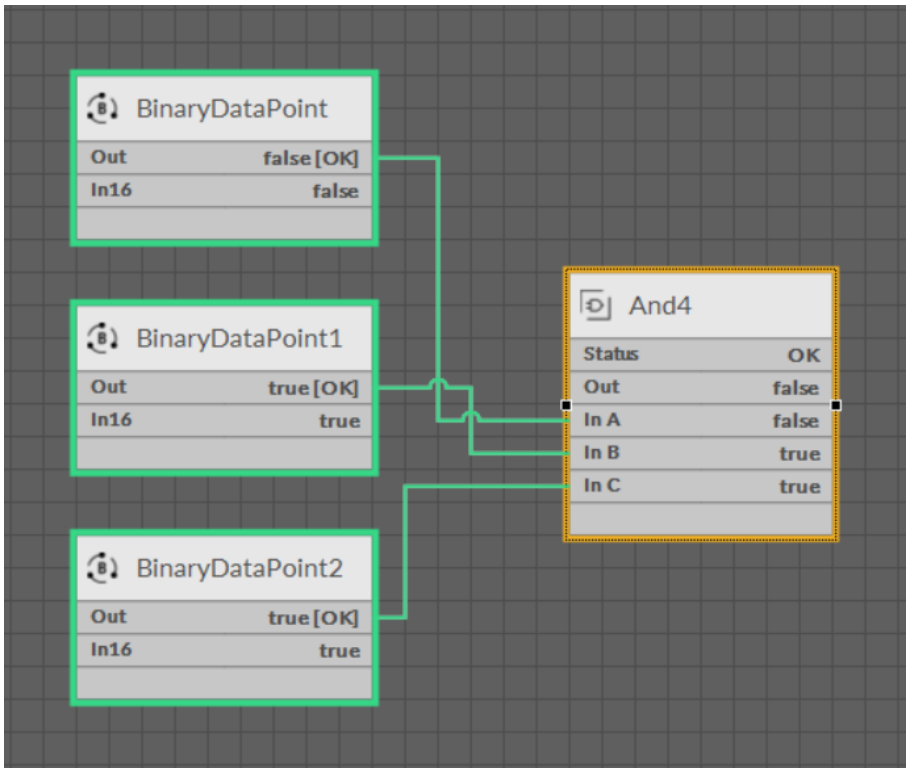


Figure 34. The And component

Slots

The And component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the logical conjunction of values of the In slots;
- **InA-InN:** 10 or 4 input slots (depending on the component's variant).

Note: By default, in the Wire Sheet the component shows four slots: Status, Out, 2 In slots.

Variants

The And component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **And10:** component with 10 In slots;
- **And4:** component with 4 In slots.

6.4.2 Between

Applicable to library's version 1.0

The Between component checks whether the In slot value is within limits set by the Low Limit and High Limit value. The Out slot returns a Boolean value—true if the In slot is within limits, false if the In slot is outside limits. The In, Low Limit, and High Limit slots receive numeric values.

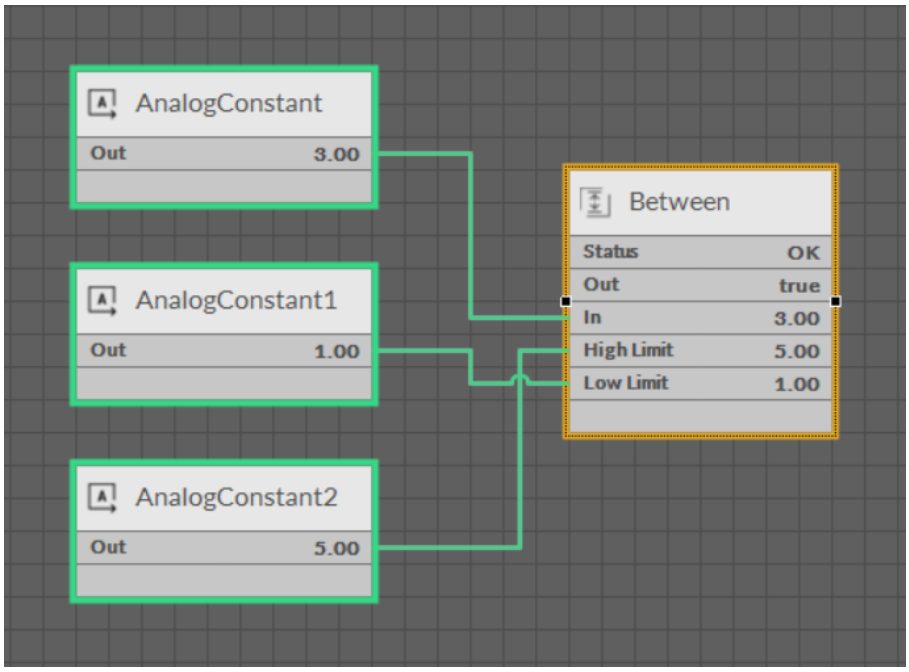


Figure 35. The Between component

Slots

The Between component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Boolean information whether the In slot holds with Low Limit and High Limit limits;
- **In:** the input value;
- **Low Limit:** the low limit value;
- **High Limit:** the high limit value.

6.4.3 Binary2Numeric

Applicable to library's version 1.0

The Binary2Numberic component returns a binary sum of Bit0 to Bit15 slots (2^0+2^1+... +2^15).

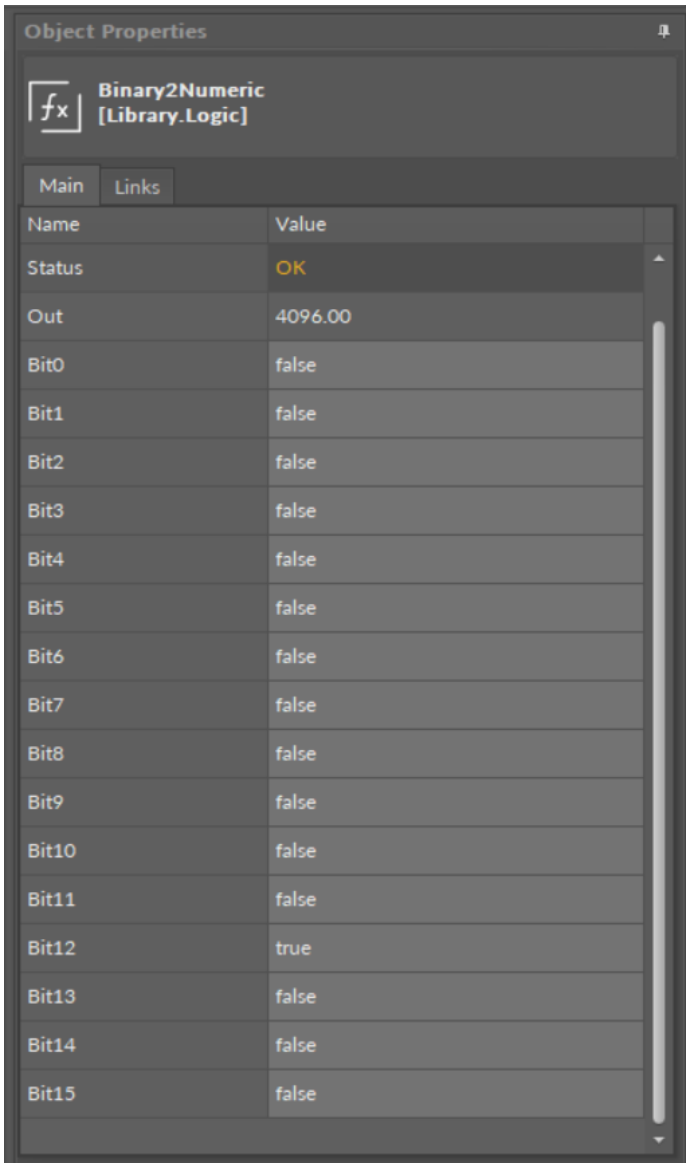


Figure 36. The Binary2Numeric component

Slots

The Binary2Numeric component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** calculates the binary sum of the Bit0–Bit15 slots values;
- **Bit0–Bit15:** the Boolean input slots.

6.4.4 Equal

Applicable to library's version 1.0

The Equal component checks whether the two In slot values are equal. The Out slot returns a Boolean value—true if the In slot values are equal, false if the In slot values are not equal. In slots receive numeric values.

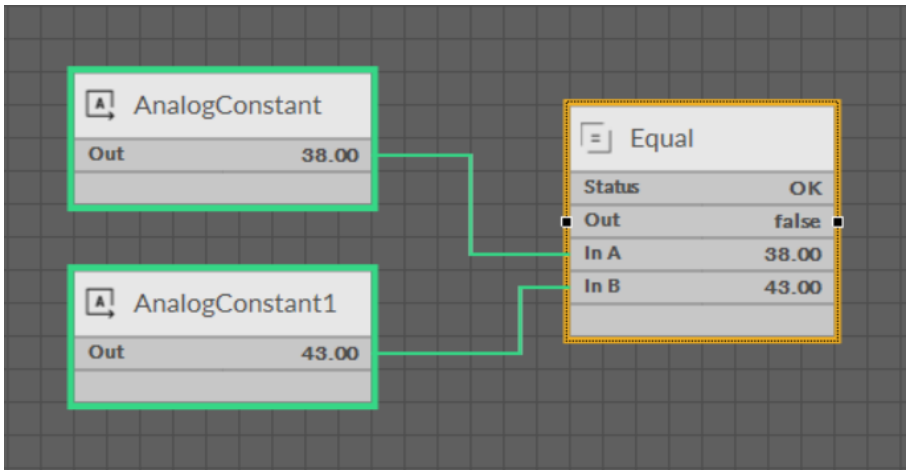


Figure 37. The Equal component

Slots

The Equal component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Boolean information whether the In slots have equal values;
- **InA-InB:** two input slots.

6.4.5 GreaterOrEqual

Applicable to library's version 1.0

The GreaterOrEqual component checks whether the InA slot value is greater than or equal to the InB slot value. The Out slot returns a Boolean value—true if the InA slot value is greater than or equal to the InB value, false if it is not. In slots receive numeric values.

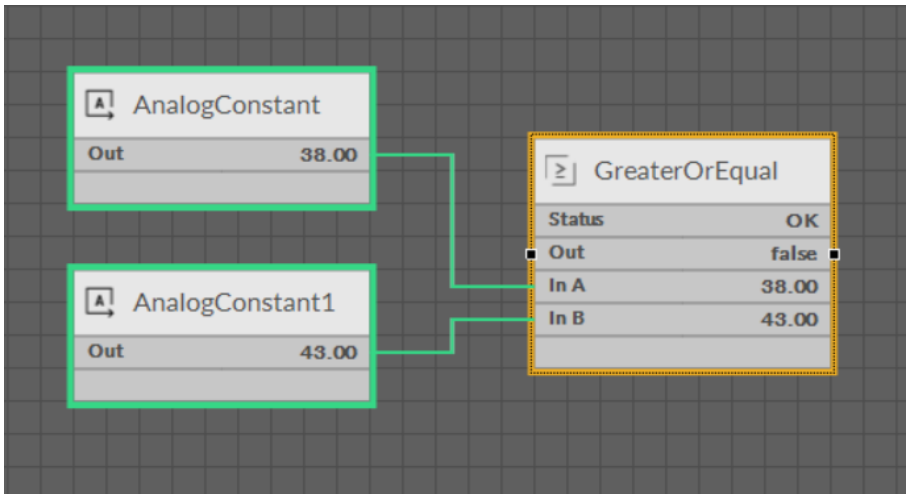


Figure 38. The GreaterOrEqual component

Slots

The GreaterOrEqual component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);

- **Out:** the Boolean information whether the InA slot has a value greater than or equal to the InB slot;
- **InA-InB:** two input slots.

6.4.6 GreaterThan

Applicable to library's version 1.0

The GreaterThan component checks whether the InA slot value is greater than the InB slot value. The Out slot returns a Boolean value—true if the InA slot value is greater than the InB value, false if it is not. In slots receive numeric values.

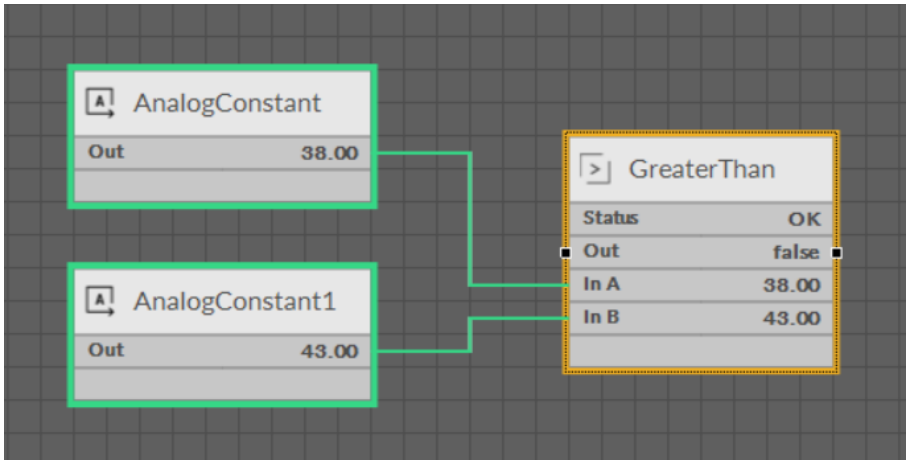


Figure 39. The GreaterThan component

Slots

The GreaterThan component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Boolean information whether the InA slot has a value greater than the InB slot;
- **InA-InB:** two input slots.

6.4.7 JKFlipFlop

Applicable to library's version 1.0

The JKFlipFlop component toggles the Out slot value, executing a mechanism based on three input values: Set, Clock, and Reset. The Out slot value is determined according to the following table:

Set	Reset	Clock	Out
False	False	(no action)	No change
True	True	Toggles between true and false	Toggles between true and false at every falling edge of the Clock value
True	False	(no action)	True

Set	Reset	Clock	Out
False	True	(no action)	False

Table 3. The JKFlipFlop component mechanism

Note: Apart from the state where Set and Rest slots are set to true, the JKFlipFlop component behaves in the same way as the RSLatch component.

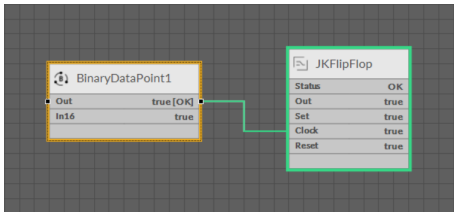


Figure 40. The JKFlipFlop component's mechanism (Set and Reset slots set to true)

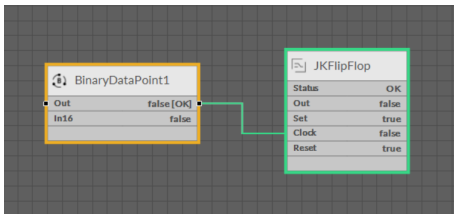


Figure 41. The JKFlipFlop component's mechanism (Set and Reset slots set to true)

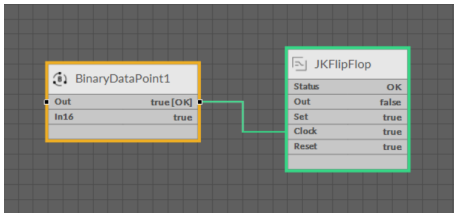


Figure 42. The JKFlipFlop component's mechanism (Set and Reset slots set to true)

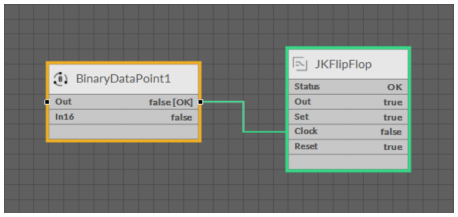


Figure 43. The JKFlipFlop component's mechanism (Set and Reset slots set to true)

Slots

The JKFlipFlop component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the output value based on the input slots values of Set, Clock, and Reset slots;
- **N Out:** the value always opposite to the Out slot value;
- **Set:** the input value transferred to the Out slot in accordance with the component's mechanism;

- **Clock:** the value synchronizing the component's mechanism if the Set and Reset slots are set to true;
- **Reset:** the input value, resetting the value latched in the Out slot.

6.4.8 LessOrEqual

Applicable to library's version 1.0

The LessOrEqual component checks whether the InA slot value is lesser than or equal to the InB slot value. The Out slot returns a Boolean value—true if the InA slot value is lesser than or equal to the InB value, false if it is not. In slots receive numeric values.

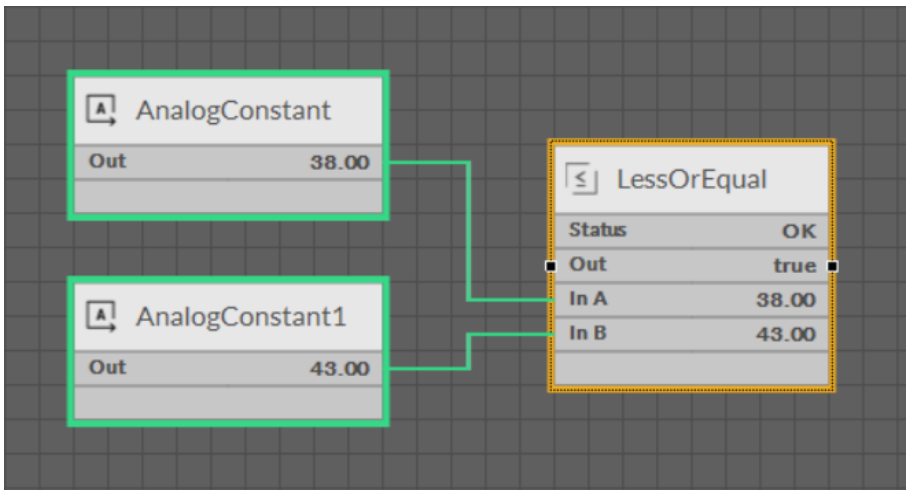


Figure 44. The LessOrEqual component

Slots

The LessOrEqual component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Boolean information whether the InA slot has a value lesser than or equal to the InB slot;
- **InA-InB:** two input slots.

6.4.9 LessThan

Applicable to library's version 1.0

The LessThan component checks whether the InA slot value is lesser than the InB slot value. The Out slot returns a Boolean value—true if the InA slot value is lesser than the InB value, false if it is not. In slots receive numeric values.

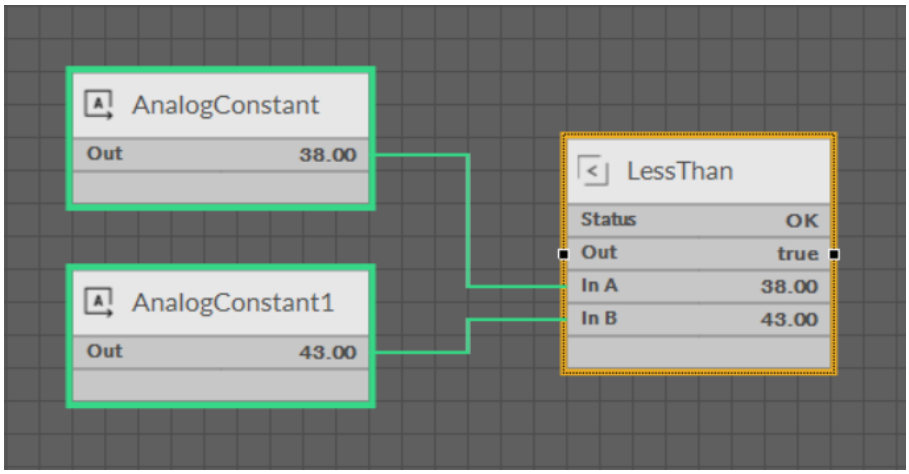


Figure 45. The LessThan component

Slots

The LessThan component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Boolean information whether the InA slot has a value lesser than the InB slot;
- **InA-InB:** two input slots.

6.4.10 Nand

Applicable to library's version 1.0

The Nand component performs an alternative denial (a reversed conjunction)—the Out slot value is false if and only if all active input values are true; in case any of the In slots is false, the Out value is true.

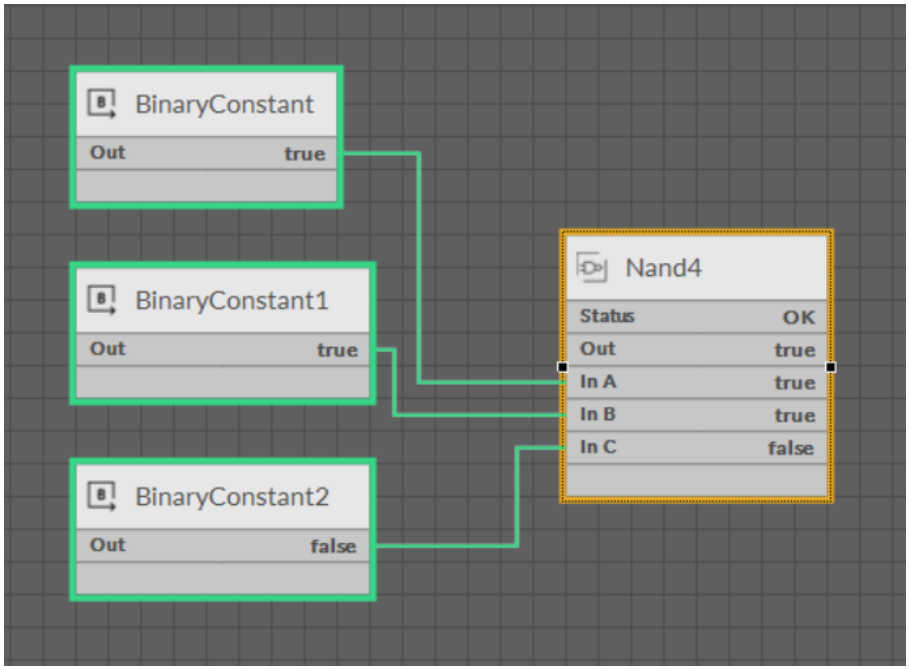


Figure 46. The Nand component

Slots

The Nand component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the alternative denial of values of the In slots;
- **InA -InN:** 10 or 4 input slots (depending on the component's variant).

Variants

The Nand component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Nand10:** component with 10 In slots;
- **Nand4:** component with 4 In slots.

6.4.11 Nor

Applicable to library's version 1.0

The Nor component performs a joint denial (a reversed disjunction)—the Out slot value is true if and only if all active input values are false; in case any of the In slots is true, the Out value is false.

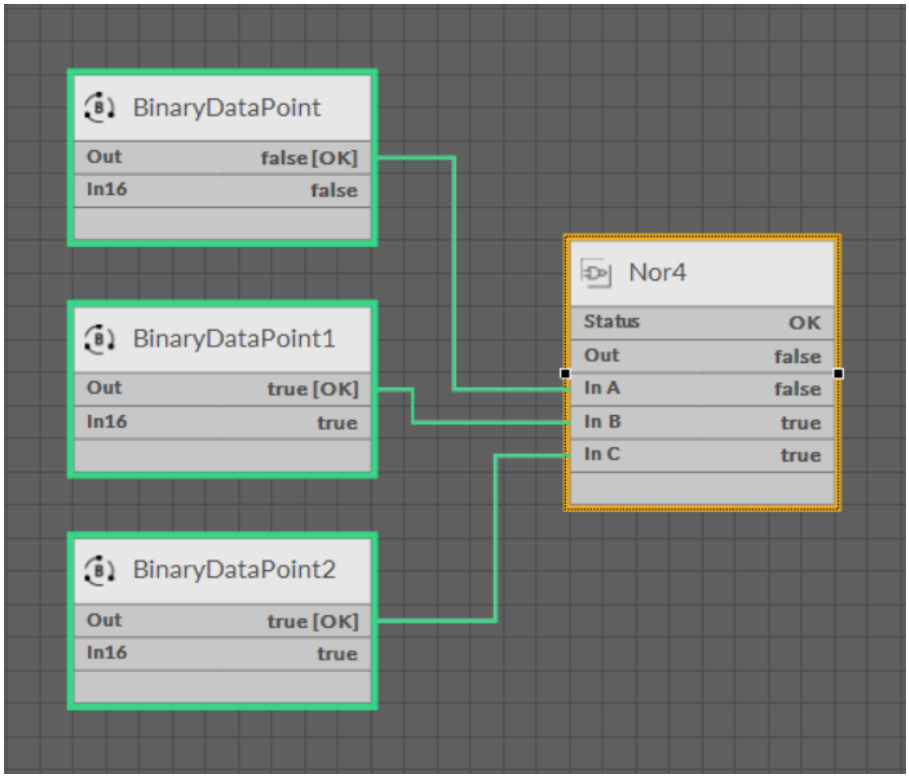


Figure 47. The Nor component

Slots

The Nor component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the joint denial of values of the In slots;

- InA -InN: 10 or 4 input slots (depending on the component's variant).

Variants

The Nor component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- Nor10: component with 10 In slots;
- Nor4: component with 4 In slots.

6.4.12 Not

Applicable to library's version 1.0

The Not component returns an inverted value of the In slot.

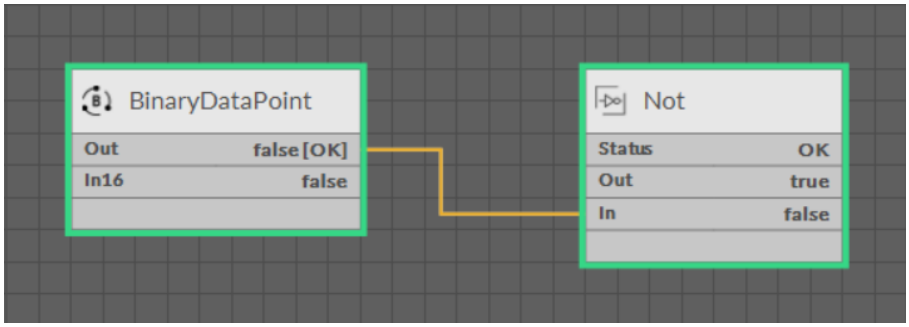


Figure 48. The Not component

Slots

The Not component has the following slots:

- Status: indicates the current status of the component (OK, Fault);
- Out: the inverted value of the In slot;
- In: the input value.

6.4.13 NotEqual

Applicable to library's version 1.0

The NotEqual component checks whether the two In slot values are not equal. The Out slot returns a Boolean value—true if the In slot values are not equal, false if the In slot values are equal.

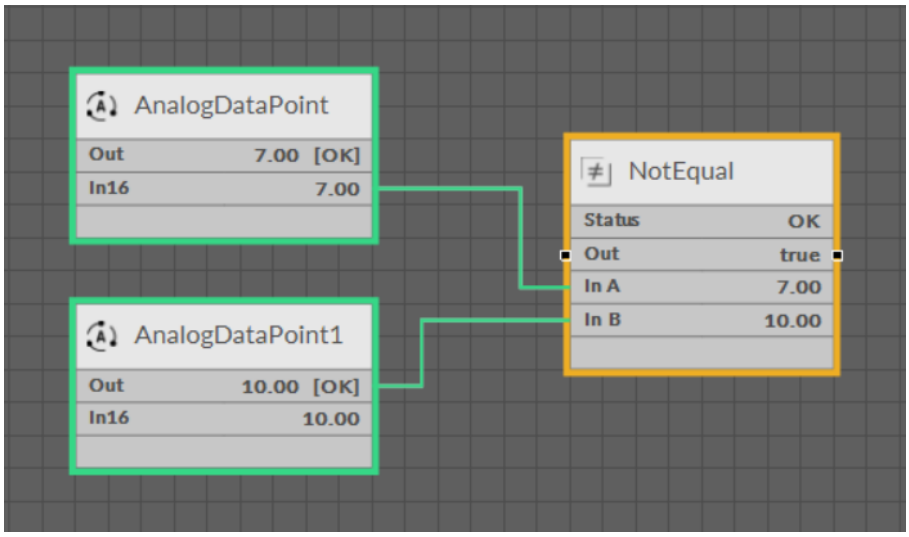


Figure 49. The NotEqual component

Slots

The NotEqual component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Boolean information whether the In slots have not equal values;
- **InA-InB:** two input slots.

6.4.14 Or

Applicable to library's version 1.0

The Or component performs a logical disjunction—the Out slot value is true if and only if at least one of the input values is true; only in case all the active In slots are false, the Out value is false.

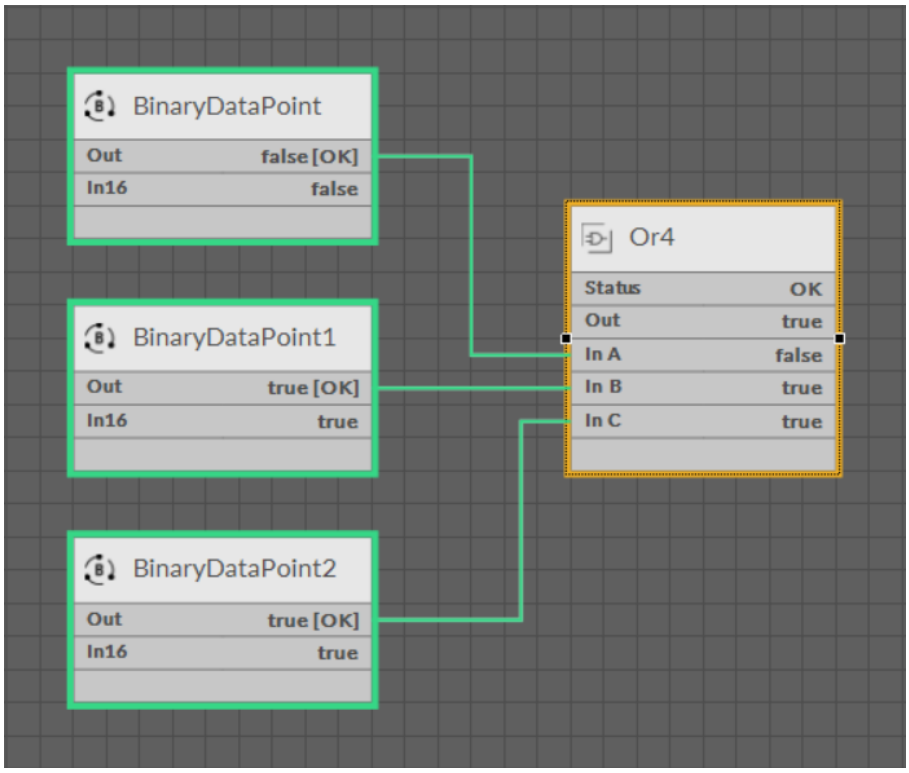


Figure 50. The Or component

Slots

The Or component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the logical disjunction of values of the In slots;
- **InA -InN:** 10 or 4 input slots (depending on the component's variant).

Variants

The Or component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Or10:** component with 10 In slots.
- **Or4:** component with 4 In slots.

6.4.15 RSLatch

Applicable to library's version 1.0

The RSLatch component latches the Out slot value, executing a mechanism based on two input values: Set and Reset. On the rising edge of the Set slot, the Out slot value is true; however, due to the latching mechanism, on the falling edge of the Set slot, the Out slot value remains true. The Reset slot value changes the Out slot value to false, regardless of the Set slot value–if the Reset value is true, the Out slot goes into false.

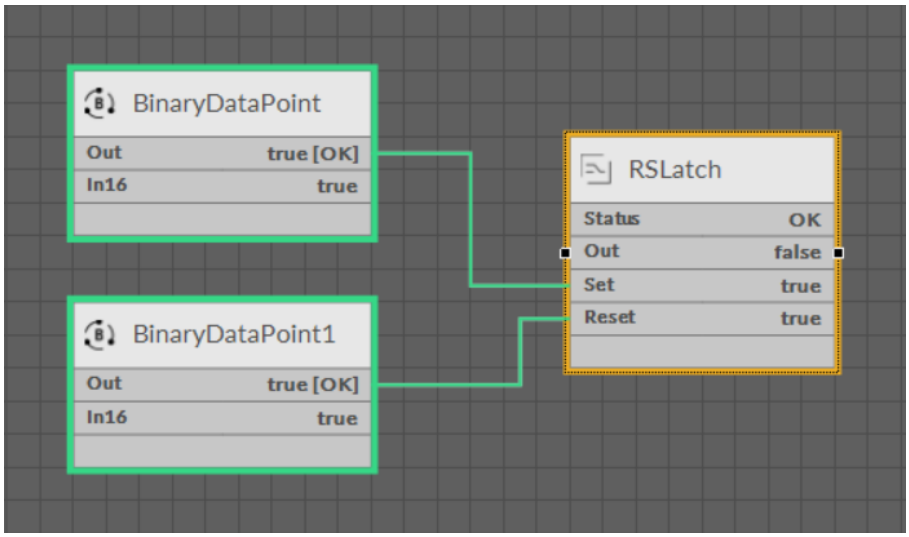


Figure 51. The RSLatch component

Slots

The RSLatch component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the output value based on the input slots values ruled by the latching mechanism;
- **N Out:** the value always opposite to the Out slot value;
- **Set:** the input value transferred to the Out slot in accordance with the latching mechanism;
- **Reset:** the input value, resetting the value latched in the Out slot.

6.4.16 Select

Applicable to library's version 1.0

The Select component allows to indicate one of the In slots to pass its value to the Out slot. In slots receive numeric values. The In slot to pass the value is indicated in the Select slot—the Select slot receives values from 0 to 9, 0 indicating the InA to pass the value to the Out slot, 1 indicating the InB, 2 indicating the InC, etc.

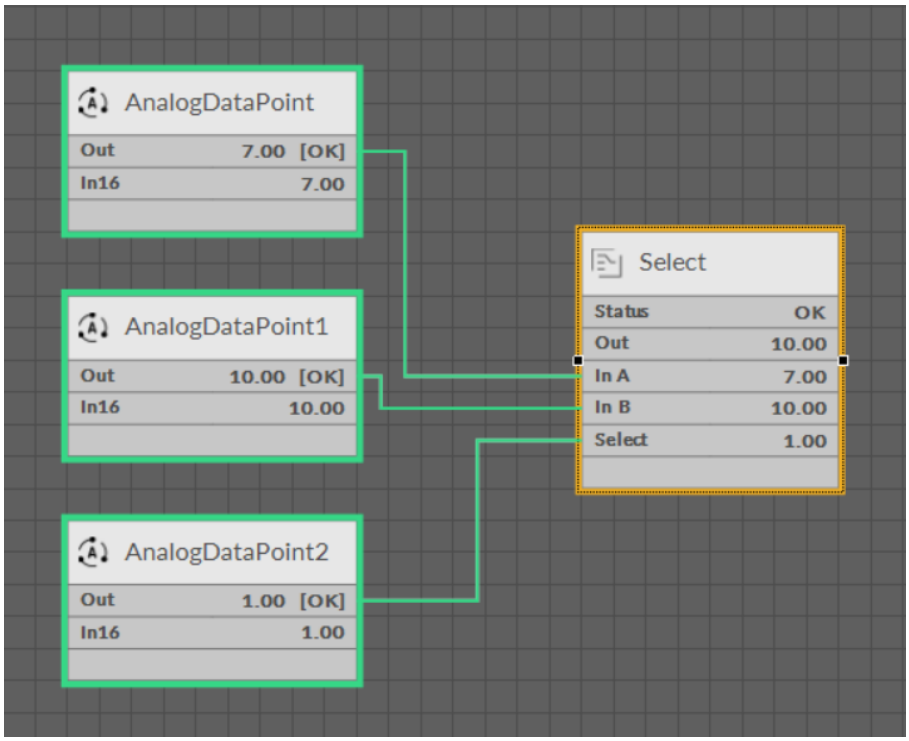


Figure 52. The Select component

Slots

The Select component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the value passed from the In slot indicated in the Select slot;
- InA -InJ: 10 input slots;
- **Select:** indicates the number of input, which passes the value to the Out slot.

6.4.17 Switch

Applicable to library's version 1.0

The Switch component selects between the two In slots based on a Boolean value in the Switch slot. In slots receive numeric values. The Switch slot receives Boolean values—if it is false, the Out slot value is passed from the InA slot; if it is true, the Out slot value is passed from the InB slot.

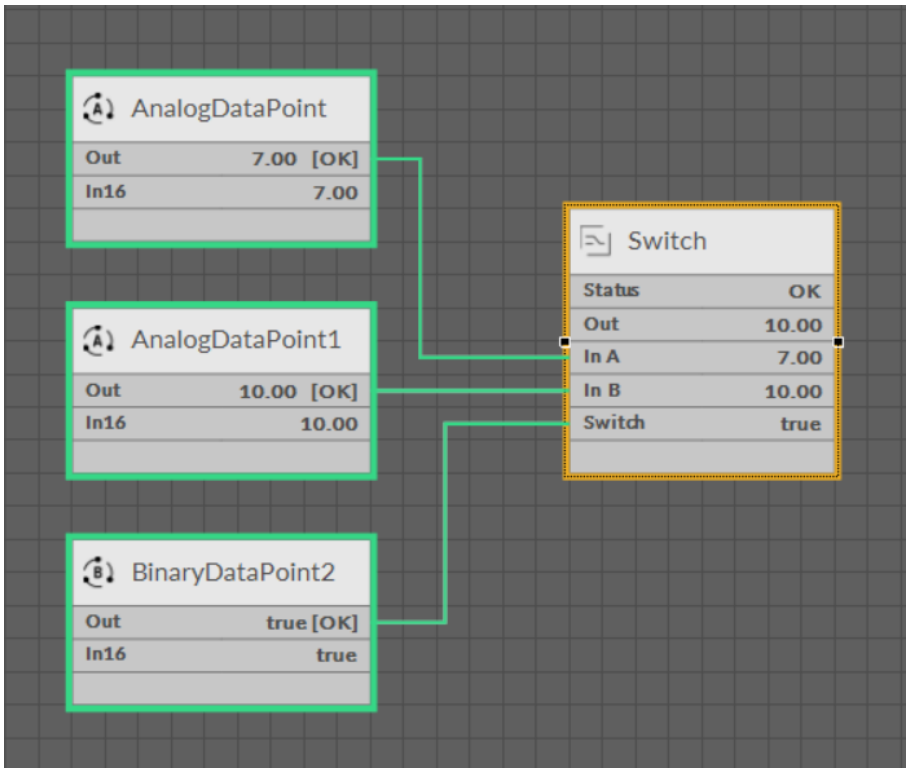


Figure 53. The Switch component

Slots

The Switch component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the value passed from one of the In slots based on the indication in the Switch slot;
- **InA-InB:** two input slots;
- **Switch:** a value determining from which In slot the value is passed to the Out slot.

6.4.18 Toggle

Applicable to library's version 1.0

The Toggle component changes its Out slot value on the rising edge of its input slot, Toggle. If the Out slot value is true, it changes to false if the Toggle slot changes its value to true (rising edge), and it is held so until the next rising edge of the Toggle slot (changing the Toggle slot value to false takes no effect on the Out slot). If the Out slot value is false, it changes to true if the Toggle slot changes to true (rising edge), and it is held so until the next rising edge of the Toggle slot (changing the Toggle slot value to false takes no effect on the Out slot). The change of the Reset slot from false to true (rising edge) takes effect on the Out slot only when its value is true. True state of the Reset slot keeps the Out slot false, even if the Toggle slot goes into the rising edge.

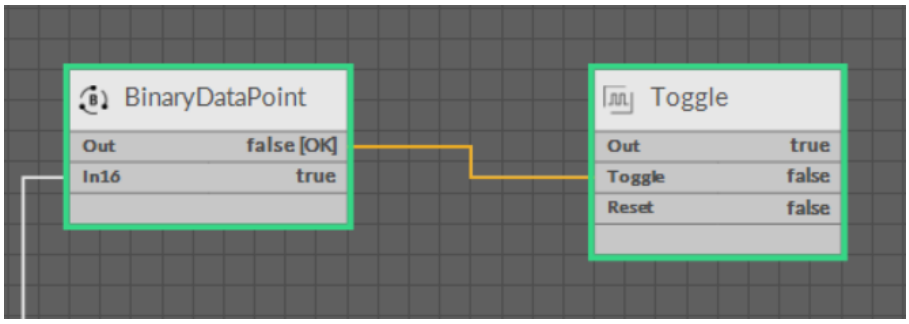


Figure 54. The Toggle component

Slots

The Toggle component has the following slots:

- **Out:** the Boolean value changed on the rising edge of either of input slots;
- **Toggle:** the input slot which changes the Out slot value on the rising edge—the first rising edge switches the Out slot to true, the next rising edge switches it to false, and the sequence recurs;
- **Reset:** the input slot which resets the Out slot value to false.

6.4.19 Xor

Applicable to library's version 1.0

The Xor component performs an exclusive disjunction—the Out slot value is true if and only if the odd number of slots is true. If the even number of slots is true, the Out slot value is false.

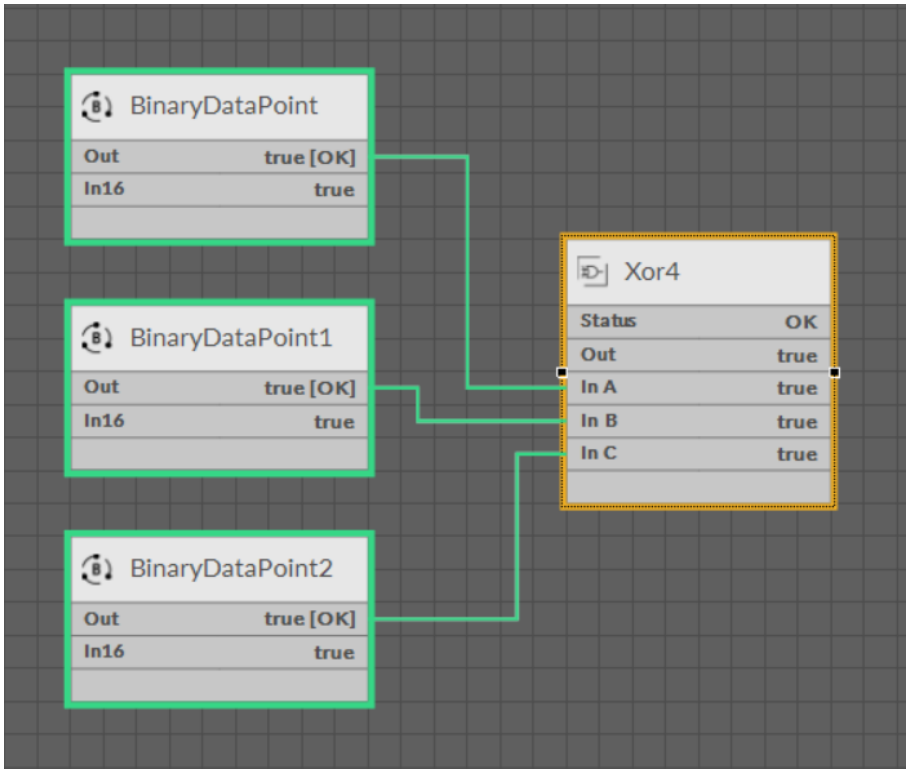


Figure 55. The Xor component

Slots

The Xor component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the exclusive disjunction of values of the In slots;
- **InA -InN:** 10 or 4 input slots (depending on the component's xorvariants).

Variants

The Xor component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Xor10:** component with 10 In slots;
- **Xor4:** component with 4 In slots.

6.5 Math

Applicable to library's version 1.0

The Math library includes components representing mathematical operations essential for creating logics.

In case of mathematical components, only the active input slots take part in mathematical operations. If all the input slots indicate a null value, it is also transmitted to the output slot.

Provided only one proper input value (other than null) is indicated in the component with many input slots, it is transmitted to the output slot (e.g., the Add component).

The results of mathematical operations are always presented to two decimal places.

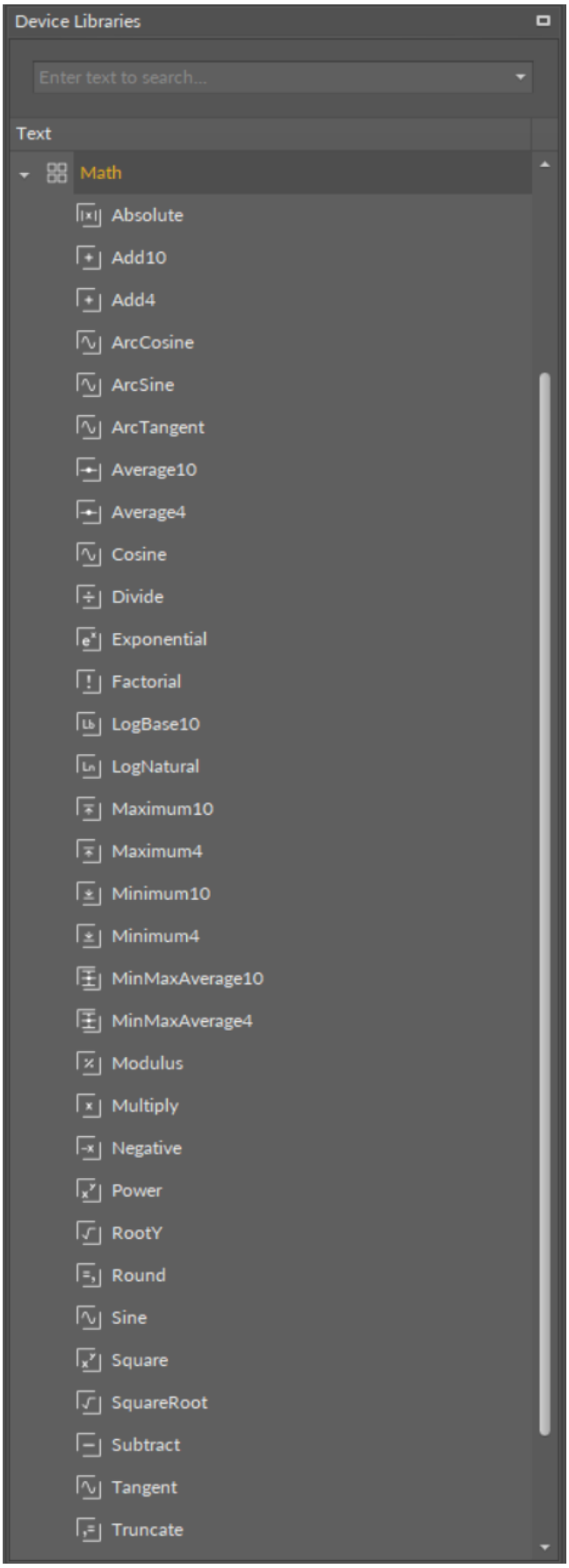


Figure 56. The Math library

6.5.1 Absolute

Applicable to library's version 1.0

The Absolute component returns an absolute value from the In slot.

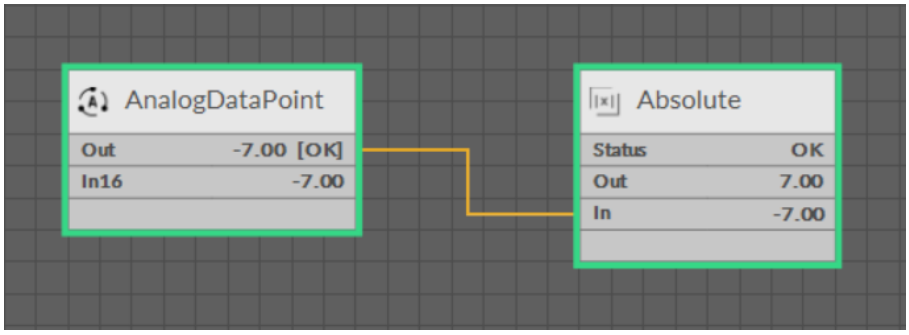


Figure 57. The Absolute component

Slots

The Absolute component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the absolute value of the In slot;
- **In:** the input value.

6.5.2 Add

Applicable to library's version 1.0

The Add component returns the sum of values of the In slots.

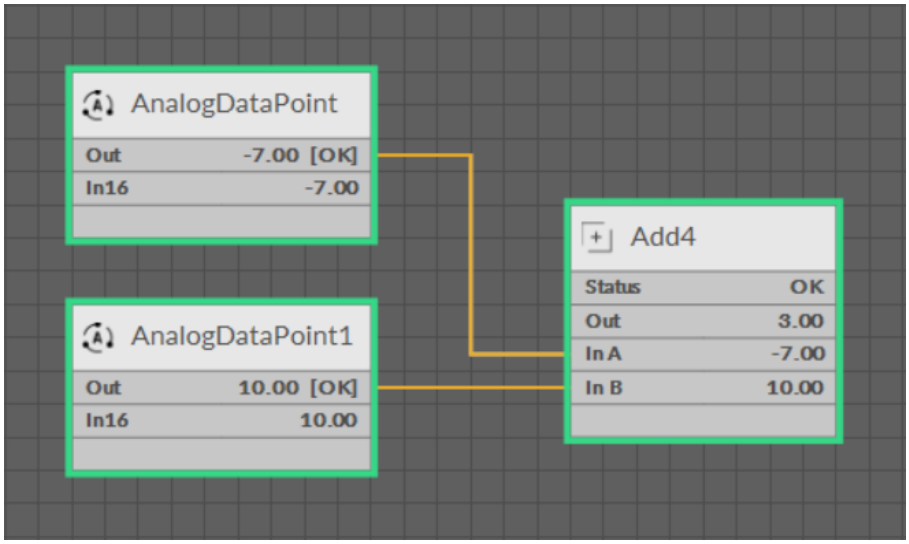


Figure 58. The Add component

Slots

The Add component as the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the sum of values of the In slots;
- **InA -InJ:** input slots (depending on the component's variant).

Variant

The Add component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Add10:** component with 10 In slots;
- **Add4:** component with 4 In slots.

6.5.3 ArcCosine

Applicable to library's version 1.0

The ArcCosine component returns an arccosine (an inverse cosine function) value of the In slot.

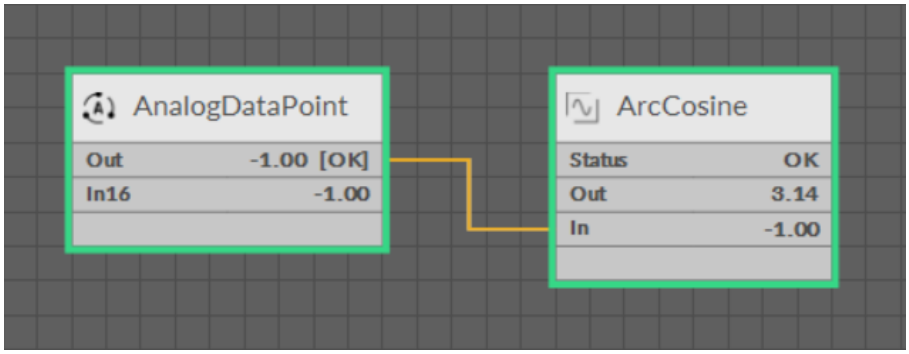


Figure 59. The ArcCosine component

Slots

The ArcCosine component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the arccosine value of the In slot expressed in radians to two decimal places;
- **In:** the input value; the possible range of input values is -1 to 1.

6.5.4 ArcSine

Applicable to library's version 1.0

The ArcSine component returns an arcsine (an inverse sine function) value of the In slot.

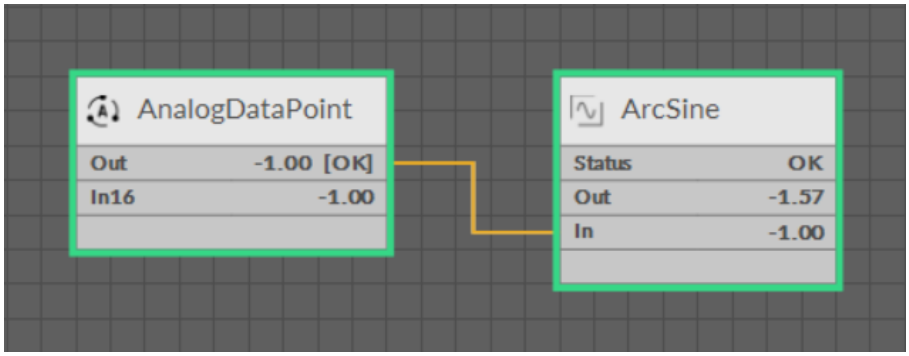


Figure 60. The ArcSine component

Slots

The ArcSine component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the arcsine value of the In slot expressed in radians to two decimal places;
- **In:** the input value; the possible range of input values is -1 to 1.

6.5.5 ArcTangent

Applicable to library's version 1.0

The ArcTangent component returns an arctangent (an inverse tangent function) value of the In slot.

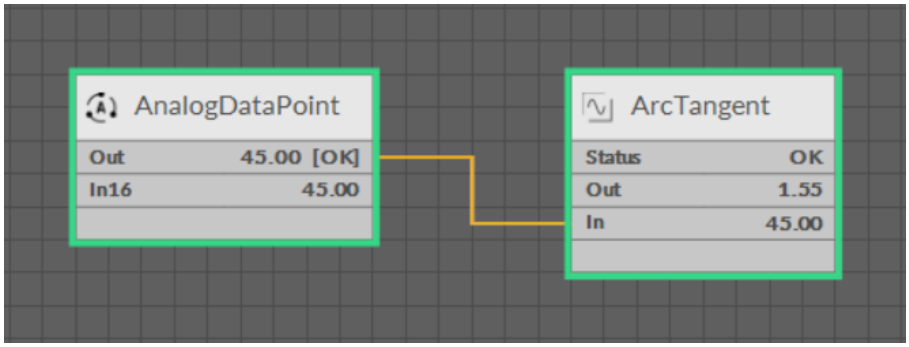


Figure 61. The ArcTangent component

Slots

The ArcTangent component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the arctangent value of the In slot expressed in radians to two decimal places;
- **In:** the input value; the possible range of input values require the input value to be a real number.

6.5.6 Average

Applicable to library's version 1.0

The Average component returns an average value of the active In slots values (input value other than null). The denominator is the number of active In slots.

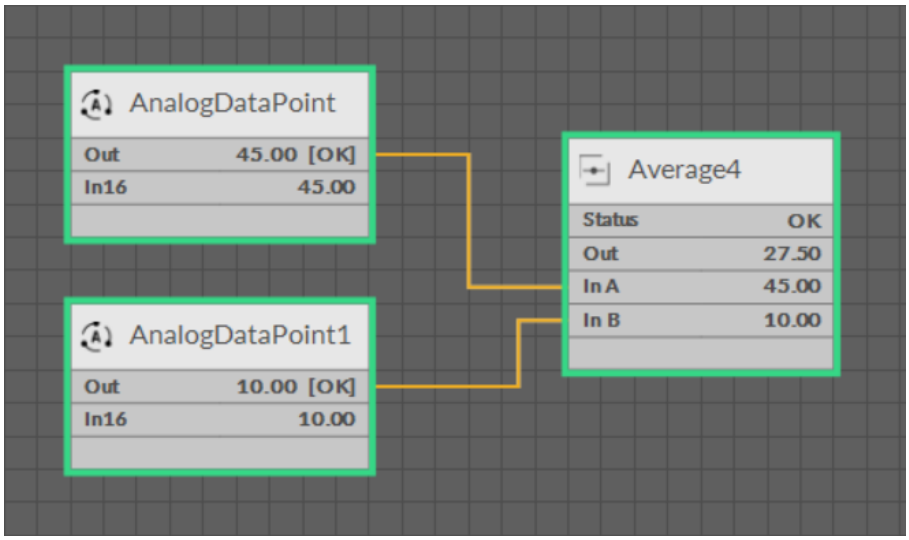


Figure 62. The Average component

Slots

The Average component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the average value of the active In slots values;
- **InA -InJ:** input slots.

Variants

The Average component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Average10:** component with 10 In slots;
- **Average4:** component with 4 In slots.

6.5.7 Cosine

Applicable to library's version 1.0

The Cosine component returns a cosine value of the In slot.

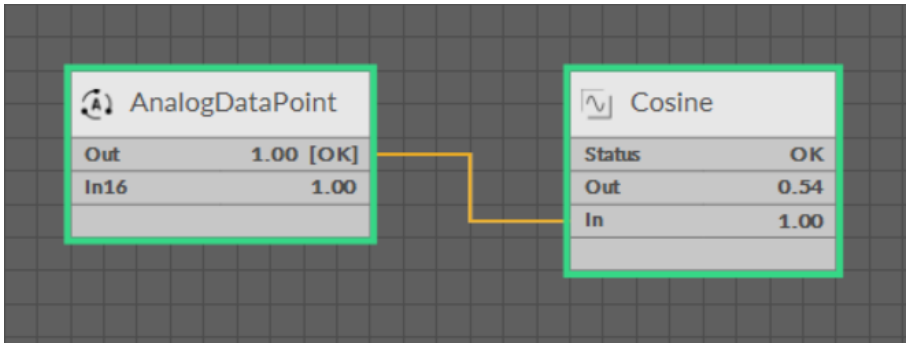


Figure 63. The Cosine component

Slots

The Cosine component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the cosine value of the In slot;
- **In:** the input value, taken into calculation in radians to two decimal places.

6.5.8 Divide

Applicable to library's version 1.0

The Divide component returns a quotient of the In slots values.

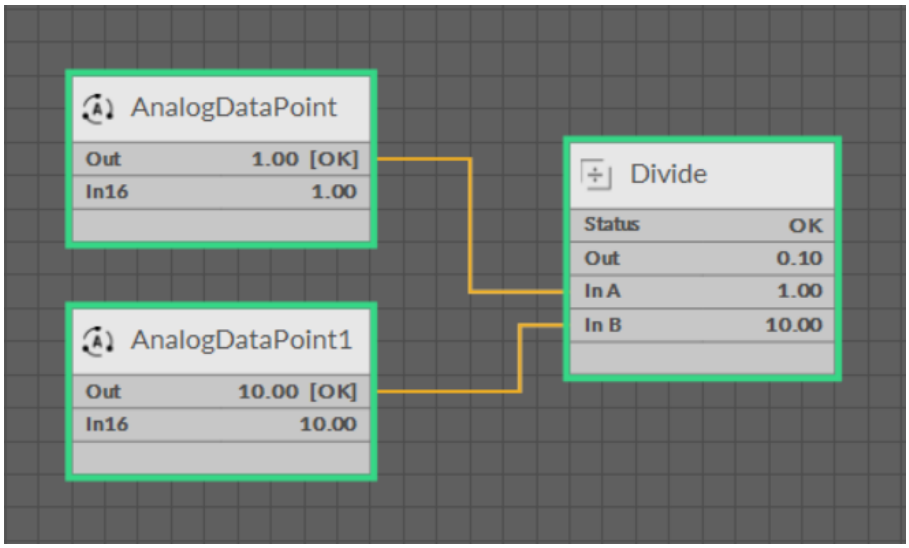


Figure 64. The Divide component

Slots

The Divide component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the quotient of the In slots values;
- **InA -InB:** 2 input slots.

Note: The InB slot value is the divider. In case the InB value equals 0, the Out value is null, and the Status is Fault.

6.5.9 Exponential

Applicable to library's version 1.0

The Exponential component returns the value of the e number (the exponential constant) raised to the power of the In slot value.

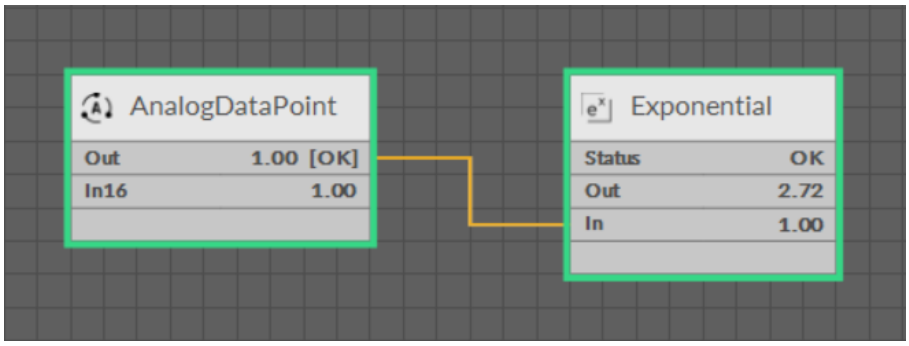


Figure 65. The Exponential component

Slots

The Exponential component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the value of the e raised to the power of the In slot value, the result expressed to two decimal places;
- **In:** the input value.

6.5.10 Factorial

Applicable to library's version 1.0

The Factorial component returns a factorial value of the In slot value.

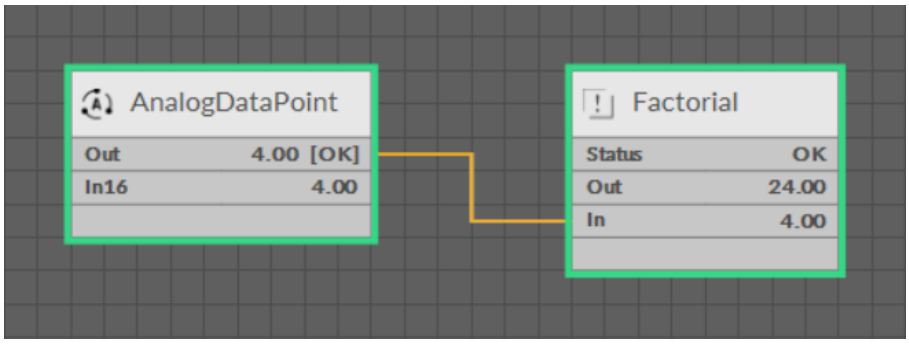


Figure 66. The Factorial component

Slots

The Factorial component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the factorial value of the In slot value;
- **In:** the input value; the possible range of input value requires integer values;

Note: In case the input slot value is with decimals, the result will be calculated from its integer value:

In = 3.67, Out = 6.00 (calculated 3.00! = 6.00).

6.5.11 LogBase10

Applicable to library's version 1.0

The LogBase10 component returns a common logarithm (a logarithm to the base 10) of the In slot value.

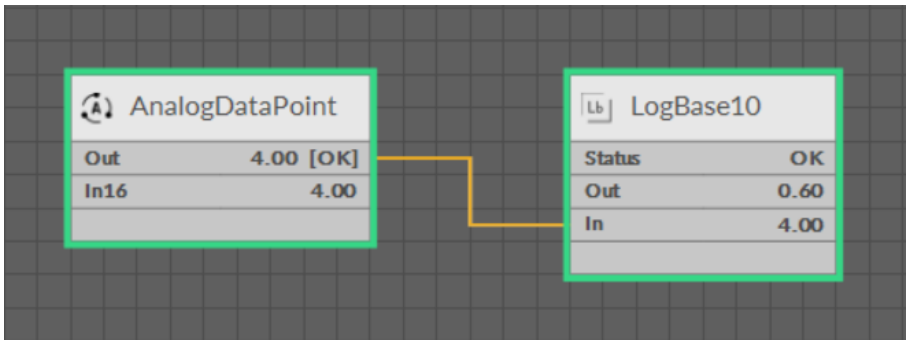


Figure 67. The LogBase10 component

Slots

The LogBase10 component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the common logarithm value of the In slot value;
- **In:** the input value.

6.5.12 LogNatural

Applicable to library's version 1.0

The LogNatural component returns a natural logarithm (a logarithm to the base of the mathematical constant e) of the In slot value. [out = ln(In)]

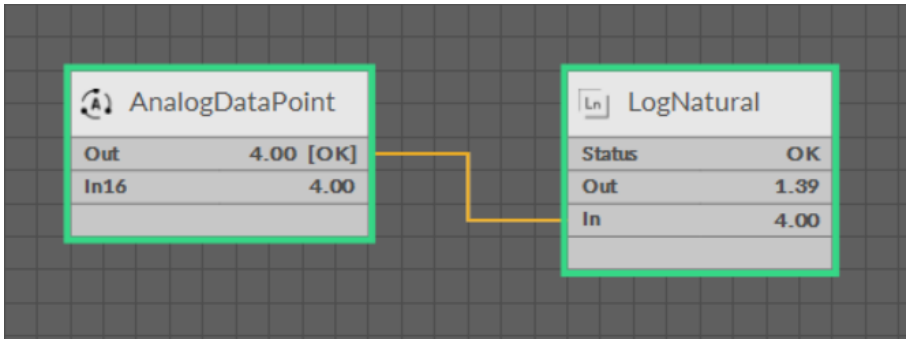


Figure 68. The LogNatural component

Slots

The LogNatural component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the natural logarithm value of the In slot value;
- **In:** the input value.

6.5.13 Maximum

Applicable to library's version 1.0

The Maximum component returns the greatest value of the In slots values.

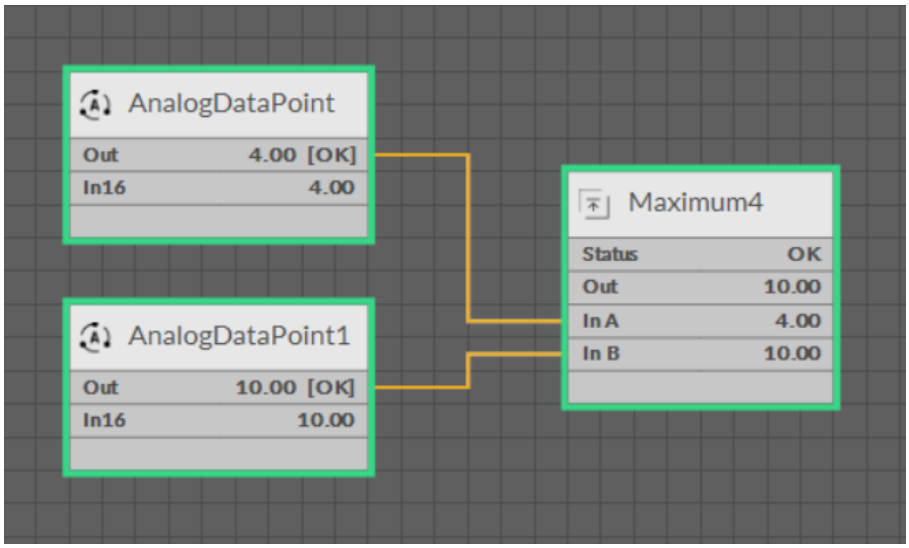


Figure 69. The Maximum component

Slots

The Maximum component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the maximum value of the In slots values;
- **InA -InJ:** input slots (depending on the component's variant).

Variants

The Maximum component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Maximum10:** component with 10 In slots;
- **Maximum4:** component with 4 In slots.

6.5.14 Minimum

Applicable to library's version 1.0

The Minimum component returns the least value of the In slots values.

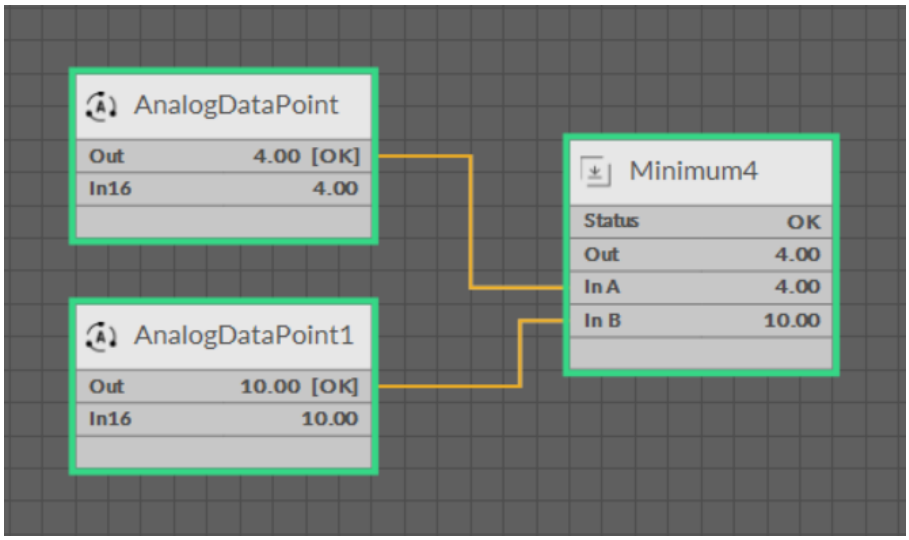


Figure 70. The Minimum component

Slots

The Minimum component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the minimum value of the In slots values;
- **InA-InJ:** input slots.

Variants

The Minimum component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **Minimum10:** component with 10 In slots;
- **Minimum4:** component with 4 In slots.

6.5.15 MinMaxAverage

Applicable to library's version 1.0

The MinMaxAverage component returns minimum, maximum, and average values of the In slots values.

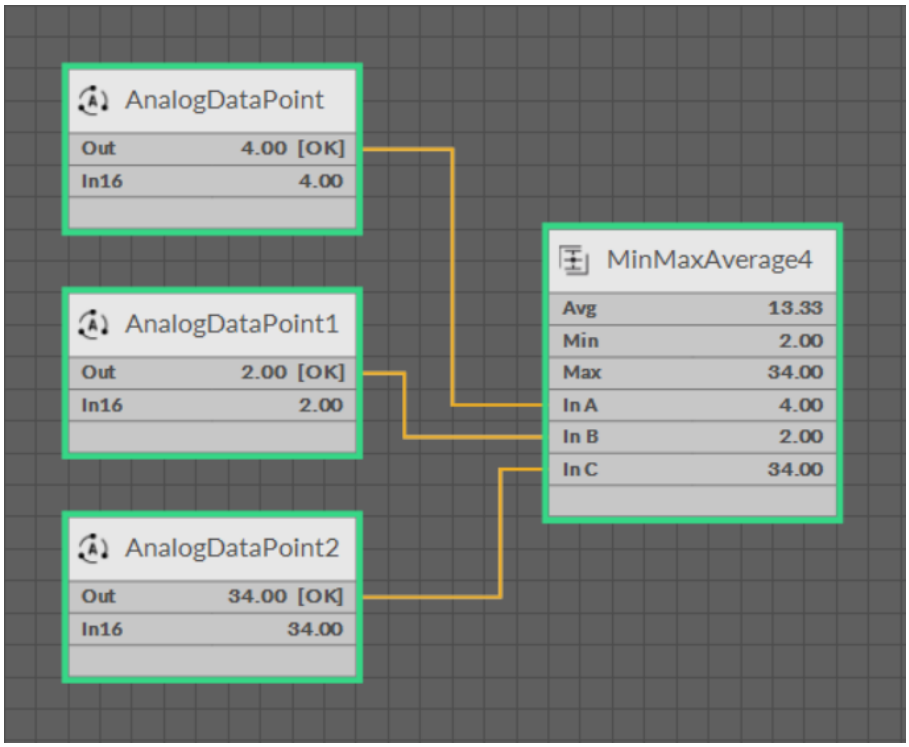


Figure 71. The MinMaxAverage component

Slots

The MinMaxAverage component has the following slots:

- **Avg:** the average value of the active In slots values;
- **Min:** the least value of the In slots values;
- **Max:** the greatest value of the In slots values;
- **InA -InJ:** input slots (depending on the component's variant).

Variants

The MinMaxAverage component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- **MinMaxAverage10:** component with 10 In slots;
- **MinMaxAverage4:** component with 4 In slots.

6.5.16 Modulus

Applicable to library's version 1.0

The Modulus component returns a modulus value (a remainder of the InA value divided by the InB value) of the In slots values.

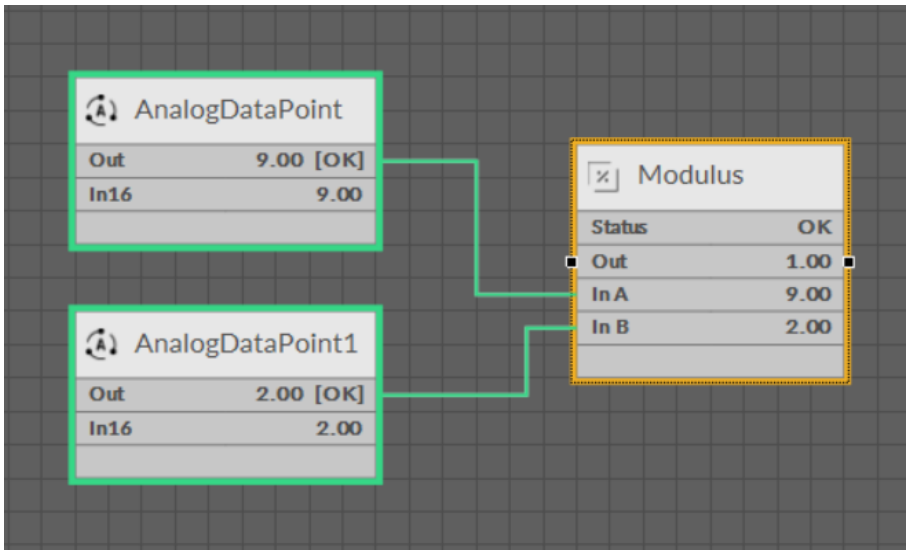


Figure 72. The Modulus component

Slots

The Modulus component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the modulus value of the In slots values;
- InA -InB: 2 input slots.

Note: The InB slot value is the divider. In case the InB value equals 0, the Out value is null, and the Status is Fault.

6.5.17 Multiply

Applicable to library's version 1.0

The Multiply component returns a product of the In slots values.

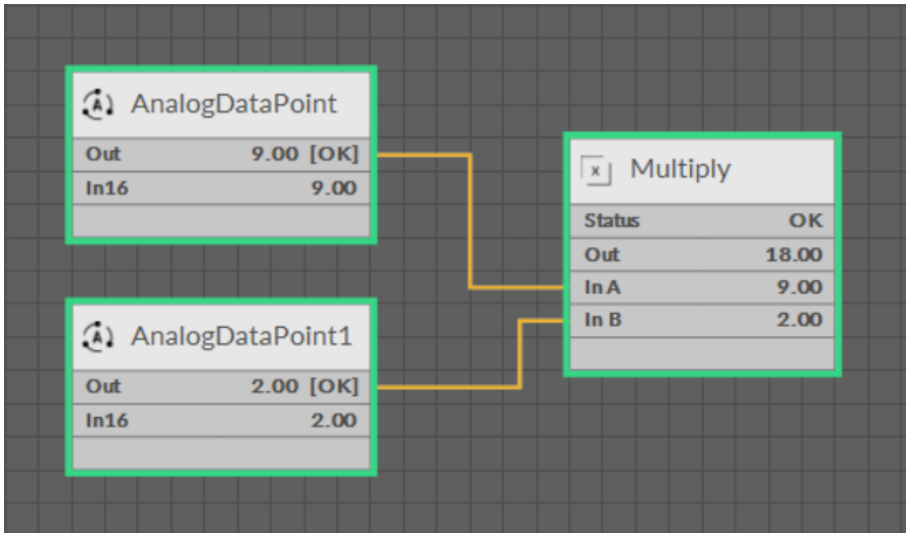


Figure 73. The Multiply component

Slots

The Multiply component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the product of the In slots values;
- **InA -InD:** 4 input slots.

Note: For the Multiply component to perform properly at least one In slot has to be active; otherwise the Out value is null, and the Status is Fault.

6.5.18 Negative

Applicable to library's version 1.0

The Negative component returns a negative value of the In slot value.

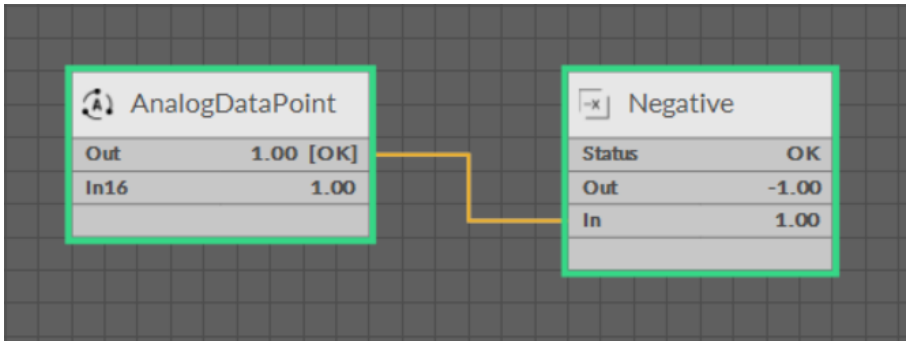


Figure 74. The Negative component

Slots

The Negative component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the negative value of the In slot value;
- **In:** the input value.

6.5.19 Power

Applicable to library's version 1.0

The Power component returns a power value of the In slots values.

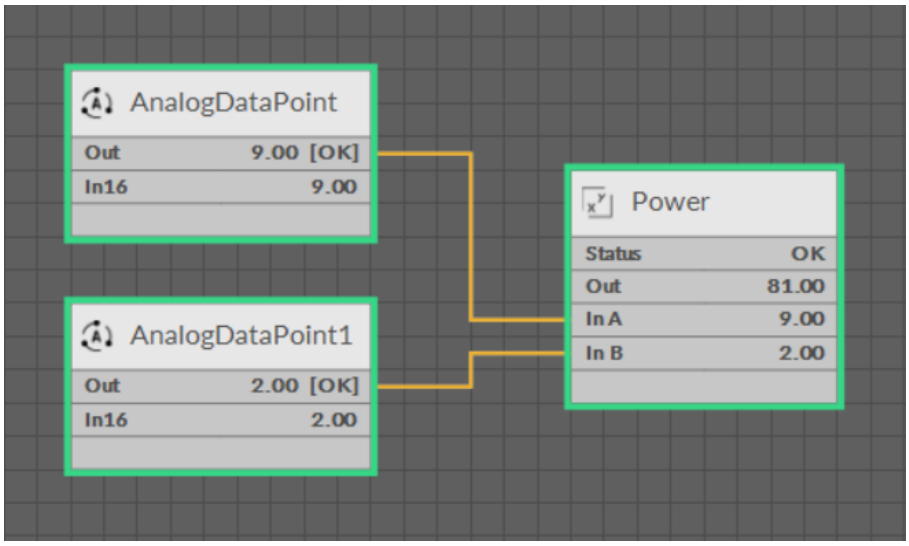


Figure 75. The Power component

Slots

The Power component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the power value of the In slots values;
- **InA -InB:** 2 input slots.

Note: The InB slot value is the exponent.

6.5.20 RootY

Applicable to library's version 1.0

The RootY component calculates the Y^{th} root of the In slot value. The Y slot represents the root's degree. [$\sqrt[Y]{In}$]

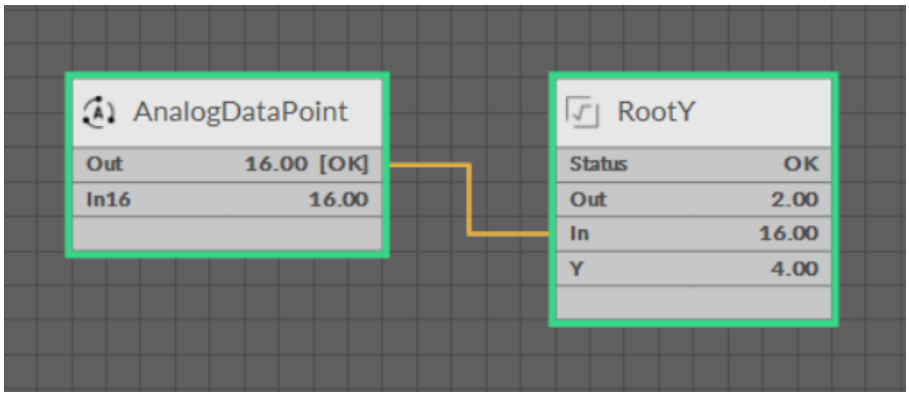


Figure 76. The RootY component

Slots

The RootY component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the Y^{th} root value of the In slot value;

- In: the input value;
- Y: the degree of the root.

6.5.21 Round

Applicable to library's version 1.0

The Round component returns the nearest integer value of the In slot.

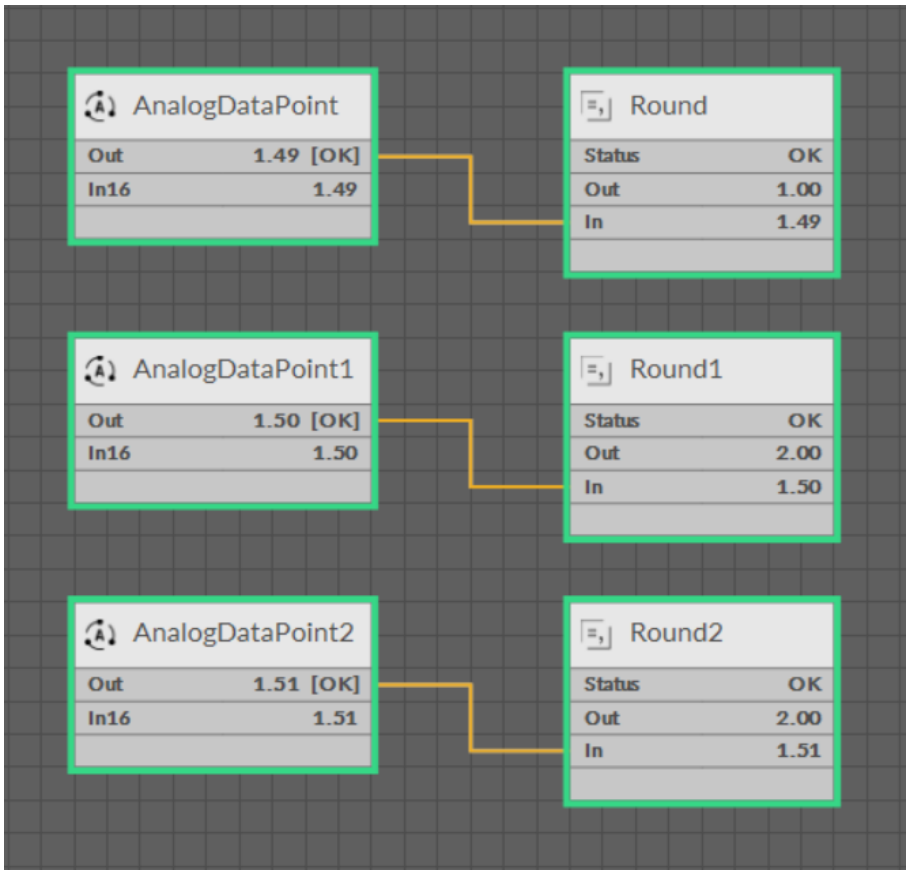


Figure 77. The Round component

Slots

The Round component has the following slots:

- Status: indicates the current status of the component (OK, Fault);
- Out: the rounded integer value of the In slot value;
- In: the input value, expressed to two decimal places.

6.5.22 Sine

Applicable to library's version 1.0

The Sine component returns a sine value of the In slot.

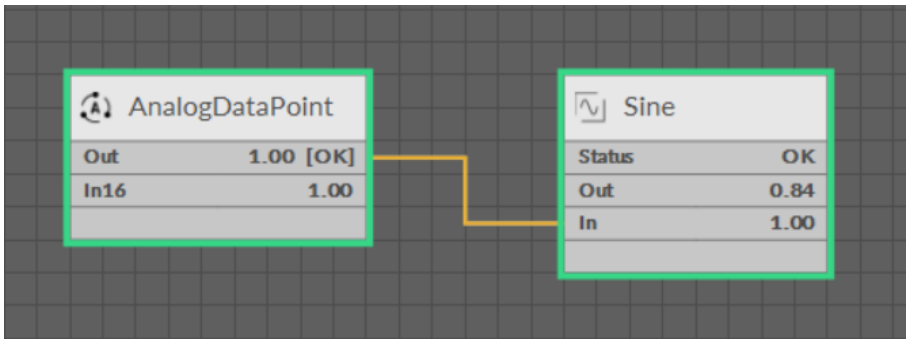


Figure 78. The Sine component

Slots

The Sine component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the sine value of the In slot;
- **In:** the input value, taken into calculation in radians to two decimal places.

6.5.23 Square

Applicable to library's version 1.0

The Square component calculates a square of the In slot value.

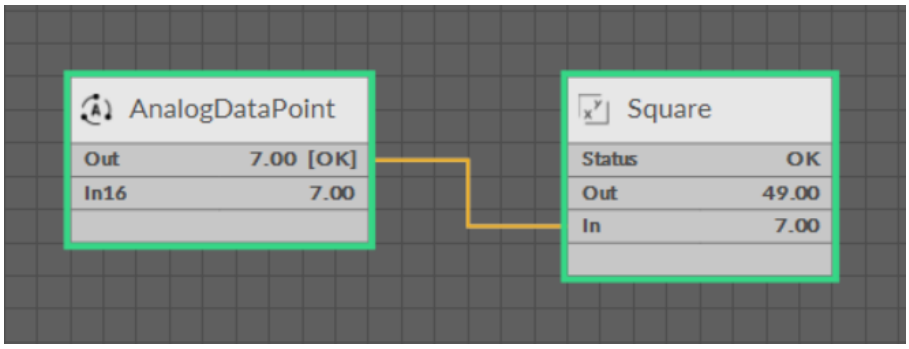


Figure 79. The Square component

Slots

The Square component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the square value of the In slot value;
- **In:** the input value.

6.5.24 SquareRoot

Applicable to library's version 1.0

The SquareRoot component returns a square root value of the In slot value.

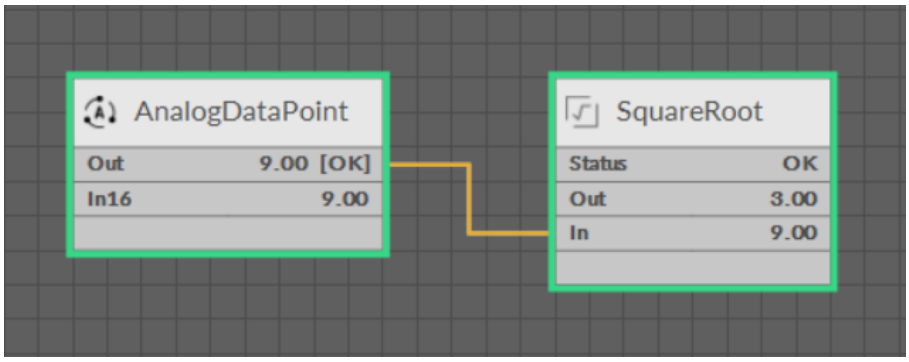


Figure 80. The SquareRoot component

Slots

The SquareRoot component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the square root value of the In slot value;
- **In:** the input value.

6.5.25 Subtract

Applicable to library's version 1.0

The Subtract component returns a subtract of the In slots values.

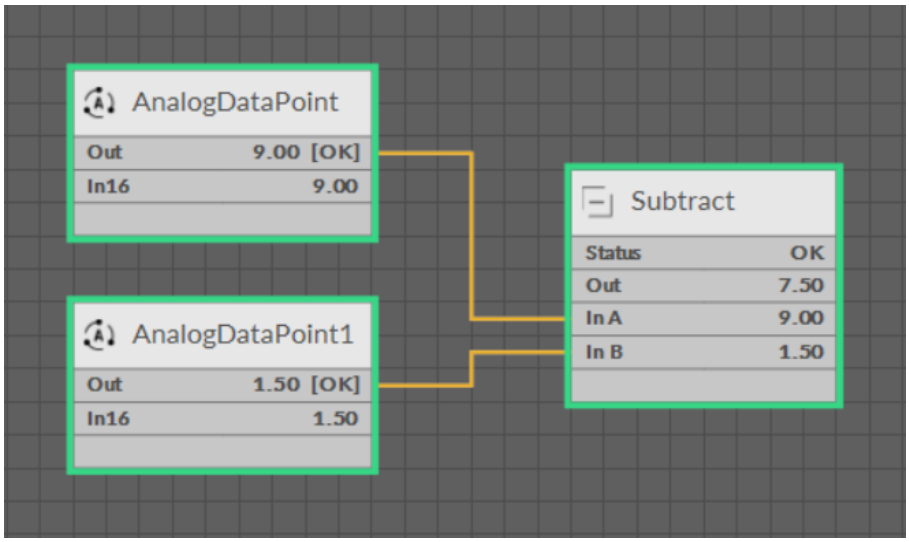


Figure 81. The Subtract component

Slots

The Subtract component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the subtract value of the In slots values;
- **InA -InB:** 2 input slots.

6.5.26 Tangent

Applicable to library's version 1.0

The Tangent component returns a tangent value of the In slot.

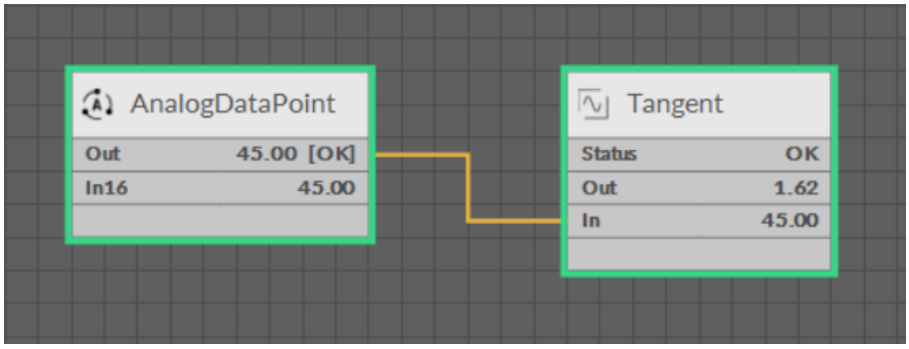


Figure 82. The Tangent component

Slots

The Tangent component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the tangent value of the In slot;
- **In:** the input value, taken into calculation in radians to two decimal places.

6.5.27 Truncate

Applicable to library's version 1.0

The Truncate component returns a truncated value of the In slot value. The In slot value is truncated to the integer.

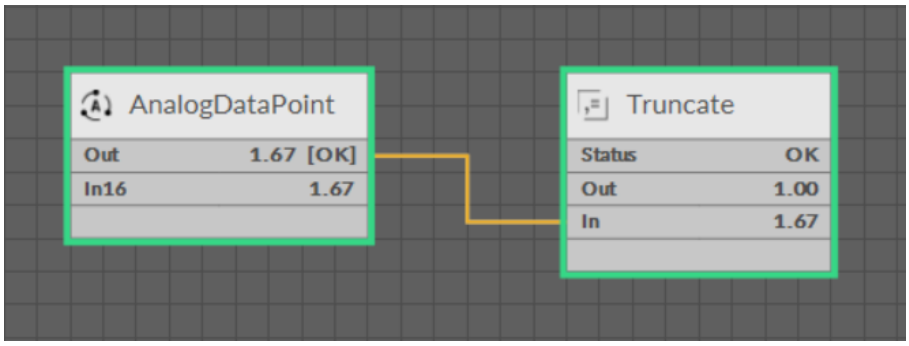


Figure 83. The Truncate component

Slots

The Truncate component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the truncated value of the In slot value;
- **In:** the input value.

6.6 Process

Applicable to library's version 1.0

The Process library includes components essential to represent processes, which can be applied to HVAC management or other uses.

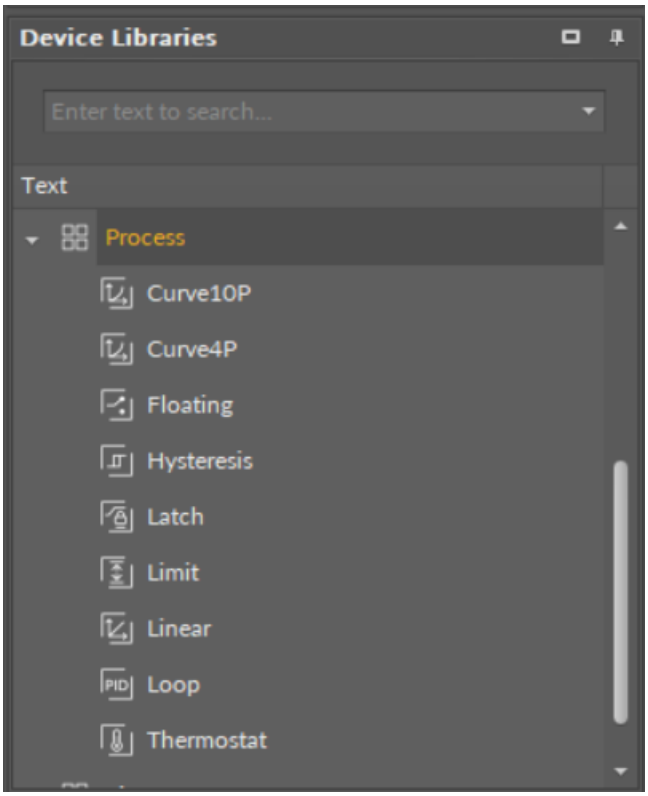


Figure 84. The Process library

6.6.1 Curve

Applicable to library's version 1.0

The Curve component performs a piecewise linearization according to intervals indicated by x1-xn values with y1-yn values assigned respectively. The Curve component reflects the mechanism of a heating curve. The In slot value is compared with x intervals, and the component estimates the Out slot value based upon the linear interpolation. The component has a Limit slot, which allows to limit the Out slot values to the interval between minimum and maximum y values.

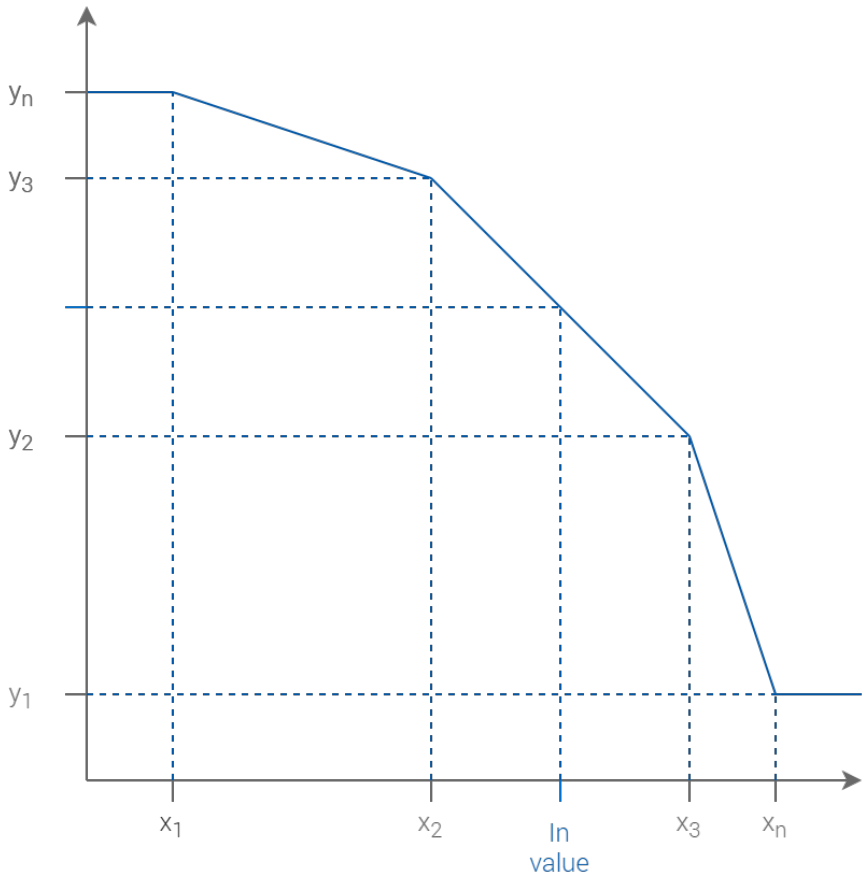


Figure 85. The Curve component mechanism

Slots

The Curve component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault).
- **Out:** the output value estimated based on the input value according to the curve algorithm;
- **In:** the input value;
- **X0-Xn:** values representing intervals for input values;
- **Y0-Yn:** values for corresponding X0-Xn values estimated according to the curve algorithm;
- **Limit:** allows to limit the Out slot value within y minimum and y maximum interval.

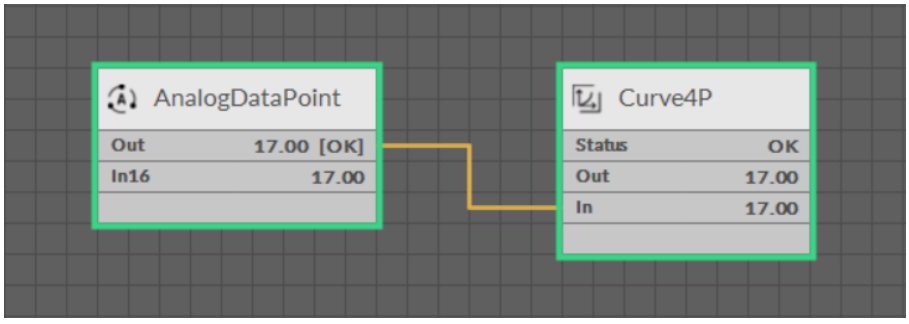


Figure 86. The Curve component

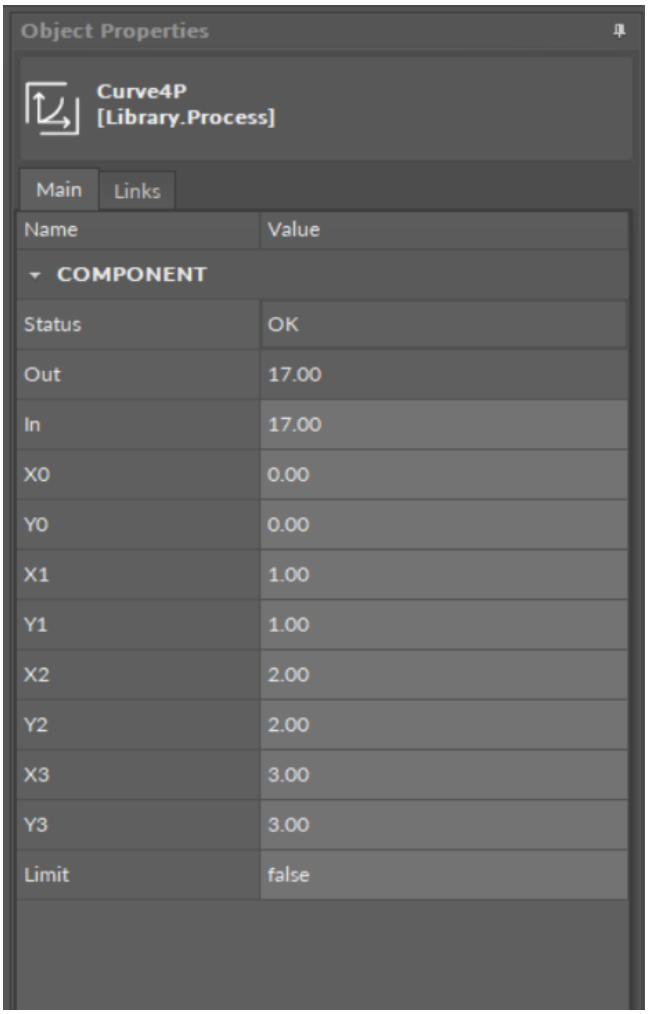


Figure 87. The Curve component slots

Variants

The Curve component comes in two variants—with 10 or 4 In slots. The 4 slots variant uses less memory within application.

- Curve10P: component with 10 In slots—represents a 10 point curve.
- Curve4P: component with 4 In slots—represents a 4 point curve.

6.6.2 Floating

Applicable to library's version 1.0

The Floating component simplifies the control of 3-point valve actuators. This component has the following functions:

- analog input, working with PID regulators;
- 2 digital outputs for dedicated valve actuators (the Up and Down slots);
- analog output for dedicated actuators by voltage level (the Out slot, additional device required);
- reset function to automatically adjust physical and virtual valve position.

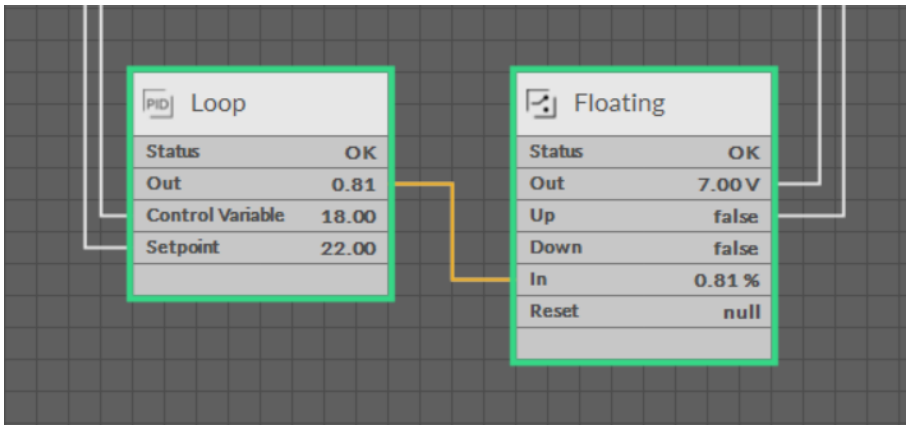


Figure 88. The Floating component

Slots

The Floating component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault).
- **Out:** the voltage value controlled by the Floating component mechanism and transferred to the dedicated actuator;
- **Up:** the digital output for the rising function of a 3-point direct control valve actuator;
- **Down:** the digital output for the lowing function of a 3-point direct control valve actuator;
- **In:** the input value representing the required percentage of the actuator openness;
- **Deadband:** the value limit within which the output deviation from the required value can be ignored;
- **Drive Time:** the time which the actuator requires to change from the fully closed position to the fully open position (expressed in seconds);
- **Reset:** resets physical and virtual valve position.

Out Value	Up	Down	Description
0 V	False	False	Off
4 V	False	True	Lowing
7 V	False	False	Static
10 V	True	False	Rising

Table 4. The Floating component values

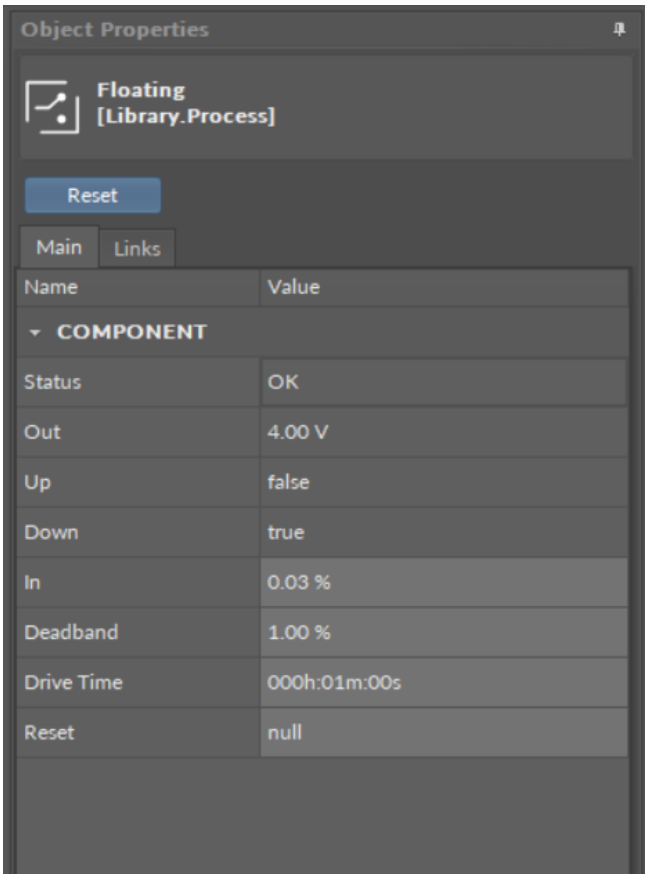


Figure 89. The Floating component slots

Action

The Floating component has the following action:

- **Reset:** allows to periodically close the valve in order to synchronize the position of full closure.

6.6.3 Hysteresis

Applicable to library's version 1.0

The Hysteresis component allows to set on and off trip points to an input value. Rising and Falling Edge slots define limit values for transferring On or Off value to the Out slot. The In slot receives numeric values.

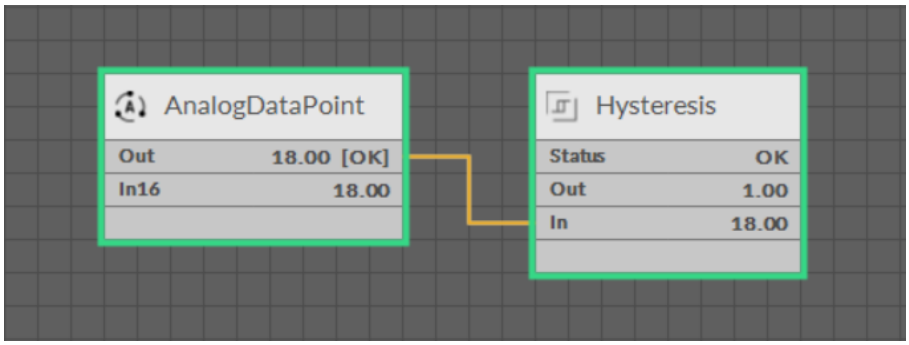


Figure 90. The Hysteresis component

Slots

The Hysteresis component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the value transferred from On or Off slot depending on the In slot value against the Rising and Falling Edge slots;
- **In:** the input value;
- **On:** the value transferred to the Out slot once the In slot value is higher than the set Rising Edge value;
- **Off:** the value transferred to the Out slot once the In slot value is lower than the set Falling Edge value;
- **Rising Edge:** a trip point for the On value to be transferred to the Out slot;
- **Falling Edge:** a trip point for the Off value to be transferred to the Out slot.

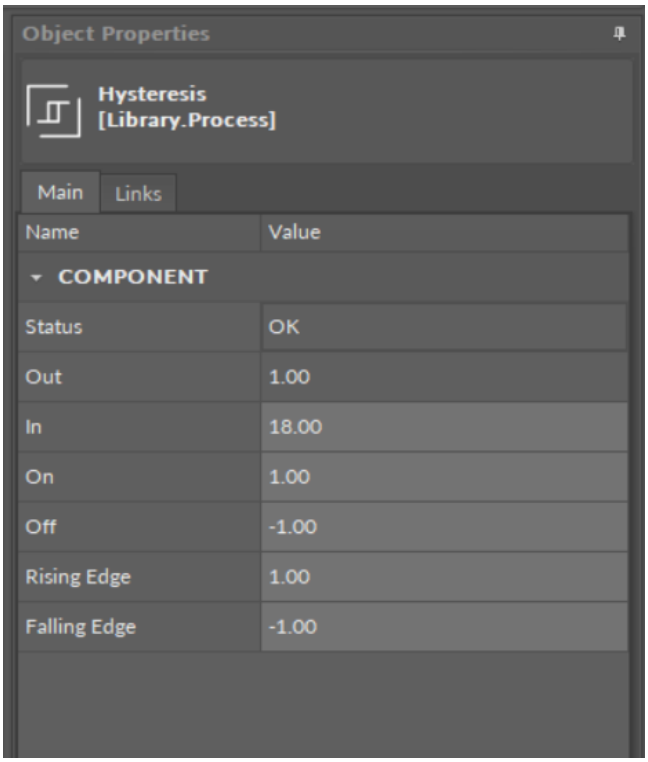


Figure 91. The Hysteresis component slots

6.6.4 Latch

Applicable to library's version 1.0

The Latch component temporarily latches the Out slot value based on the Clock slot value–if the Clock slot is set to true, the Out slot value is latched, regardless of changes to the In slot. The component has the Reset action, which brings the Clock slot value back to false.

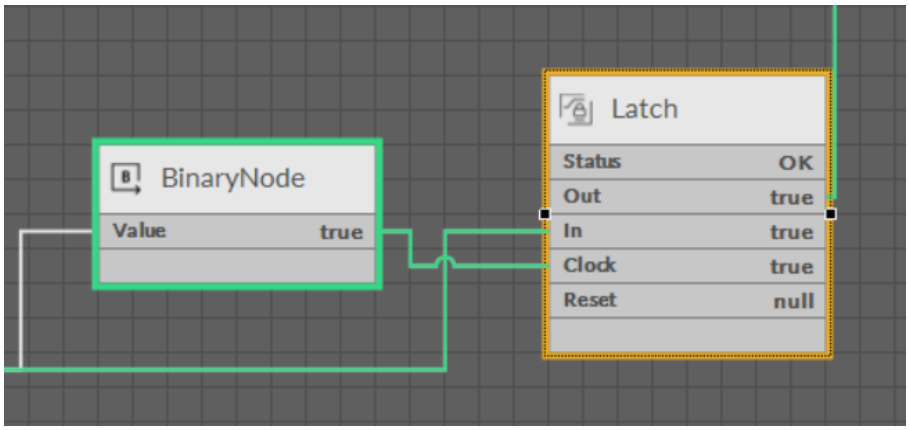


Figure 92. The Latch component

Slots

The Latch component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault).
- **Out:** the output value transferred from the In slot;
- **In:** the input value;
- **Clock:** latches the Out value if the Clk value changes from false to true;
- **Reset:** allows to reset the Clock slot value back to false, which releases the Out slot value from latching.

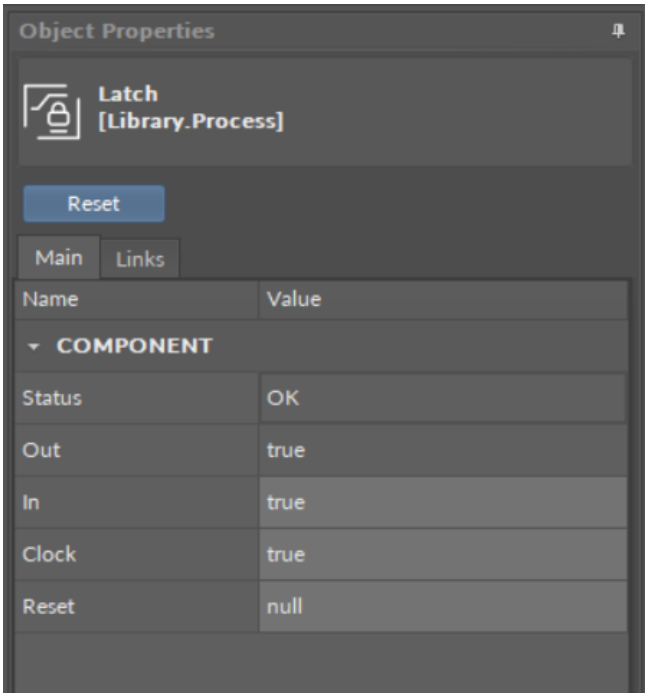


Figure 93. The Latch component slots

Action

The Latch component has the following action:

- **Reset:** allows to reset the Clock slot value back to false, which releases the Out slot value from latching.

6.6.5 Limit

Applicable to library's version 1.0

The Limit component allows to set the limits for the Out slot values. The In slot value is transferred directly to the Out slot only if it is within limit values. In case the In slot value exceeds limit values, so it is either lower than the Low Limit value, or higher than the High Limit value, the respective limit value is transferred to the Out slot.

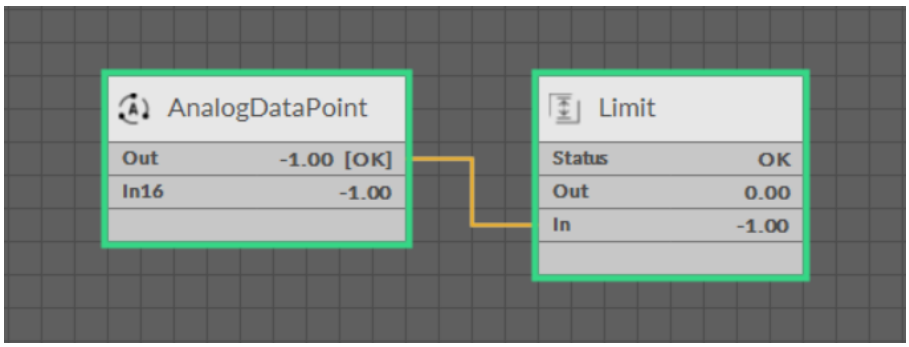


Figure 94. The Limit component

Slots

The Limit component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the value transferred from the In slot if it is within limits, if otherwise–the Low Limit value is transferred in case the In slot value is lower than the Low Limit value, and the High Limit value is transferred in case the In slot value is higher than the High Limit value;
- **In:** the input value;
- **Low Limit:** the low limit value;
- **High Limit:** the high limit value.

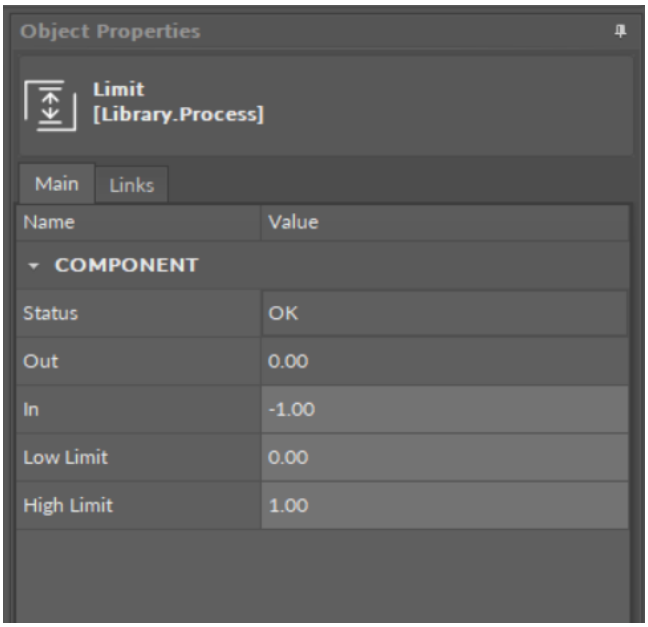


Figure 95. The Limit component slots

6.6.6 Linear

Applicable to library's version 1.0

The Linear component allows to scale the Out slot value depending on the set input values interval. The In slot value is compared with x intervals, and the component estimates the Out slot value based upon the linear function. The component has a Limit slot, which allows to limit the Out slot values to the interval between minimum and maximum y values.

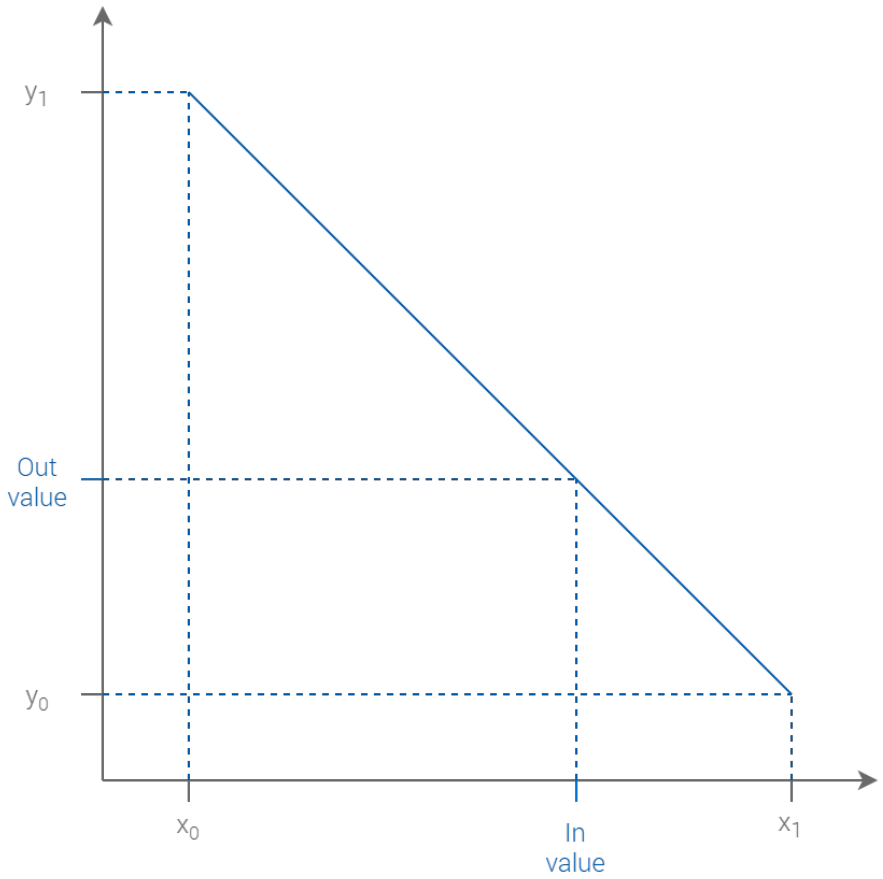


Figure 96. The Linear component mechanism

Slots

The Linear component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the output value estimated based on the linear function of the input value;
- **In:** the input value;
- **X0-X1:** values representing the intervals for the input value;
- **Y0-Y1:** values representing the internals for the output value;
- **Limit:** allows to limit the Out slot value within y minimum and y maximum interval.

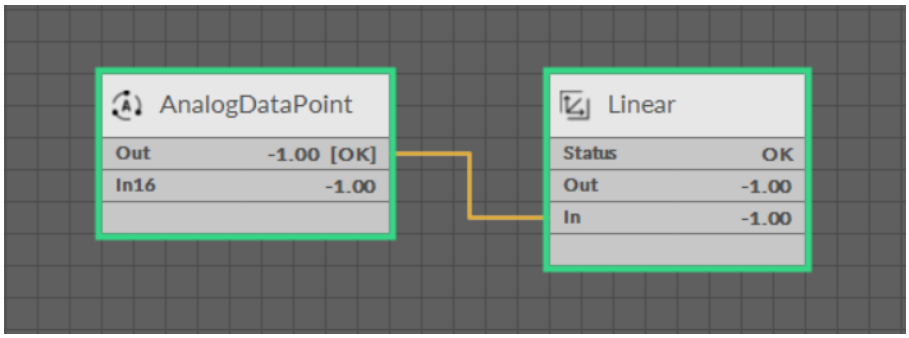


Figure 97. The Linear component

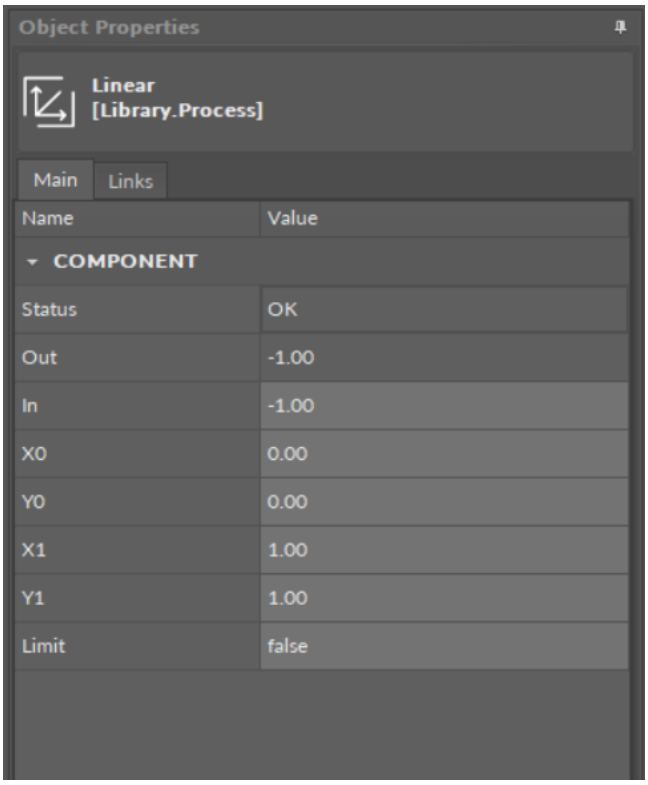


Figure 98. The Linear component slots

6.6.7 Loop

Applicable to library's version 1.0

The Loop component allows to configure a signal control loop–it allows to adjust the output value in order to reach the setpoint value comparing it with the current input value (measurement). The output value is estimated based on a proportional-integral-derivative mechanism. The Loop component may be configured as P-only, PI, or PID.

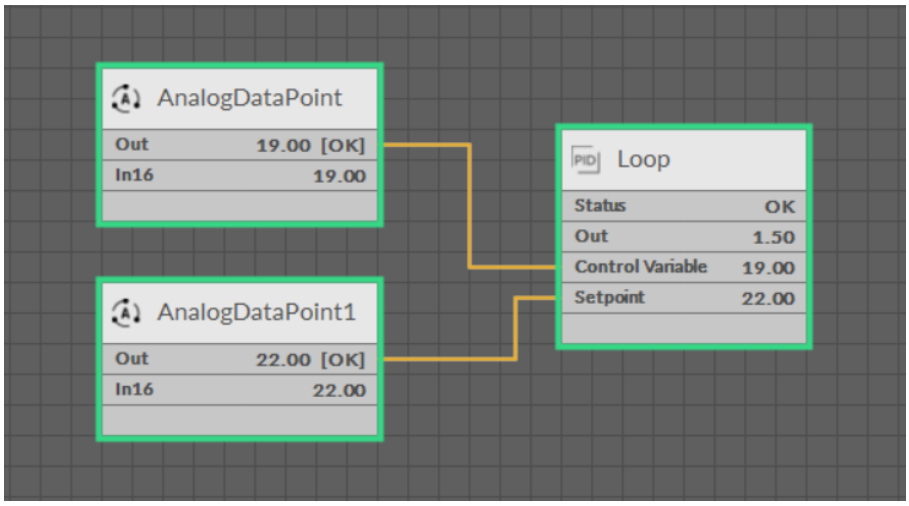


Figure 99. The Loop component

Slots

The Loop component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Enabled:** enables execution of the PID loop–if disabled, the Out slot is set to 0; if enabled, the component calculates the Out slot according to its regulation algorithm;
- **Out:** the result of the loop algorithm;
- **Control Variable:** the current, measured input value;
- **Setpoint:** the setpoint value;
- **K P:** defines a proportional gain of the loop algorithm–the value is a maximum difference between the setpoint and the current value, which results in the 100% output value;
- **K I:** defines an integral gain of the loop algorithm (expressed as 1/min);
- **K D:** defines a derivative gain of the loop algorithm (expressed as 1/min);
- **Bias:** defines the bias value added to the output;
- **Ramp Time:** defines the filtering time for the Out slot value;
- **Max Output:** allows to set a maximum limit on the Out slot value, by default set to 100;
- **Min Output:** allows to set a minimum limit on the Out slot value, by default set to 0;
- **Action:** allows to choose a Direct or Reverse operation mode of the component–the Direct mode reflects cooling/dehumidification process, and the Reverse mode reflects the heating/humidification process.

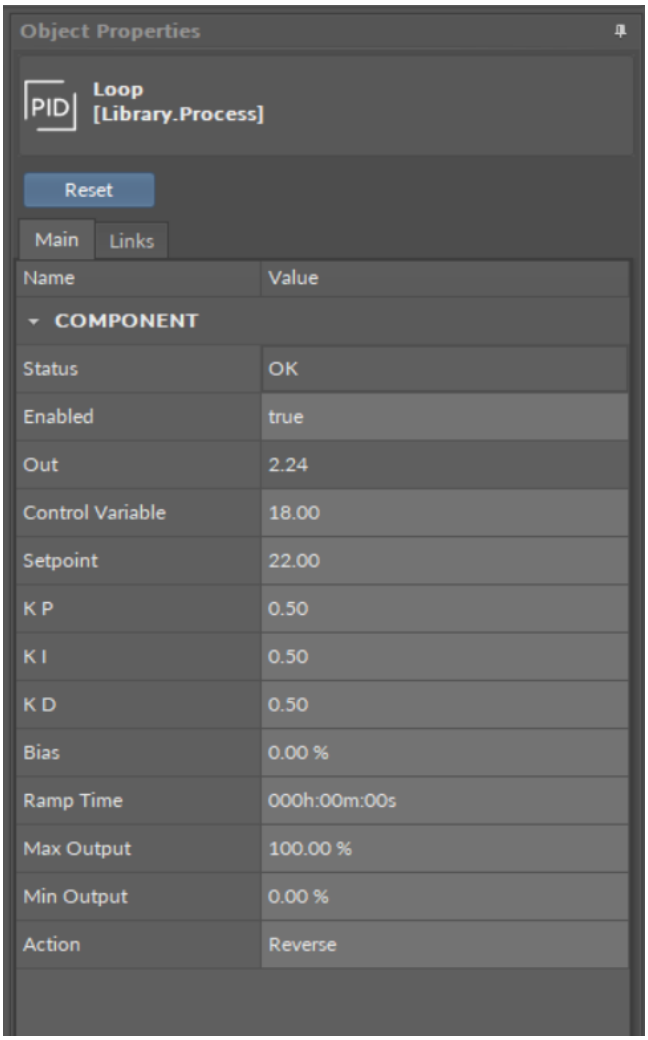


Figure 100. The Loop component slots

Action

The Loop component has the following action:

- **Reset:** brings the Out slot value back to Min Output value.

6.6.8 Thermostat

Applicable to library's version 1.0

The Thermostat component executes a basic thermostatic control. The In slot value is compared with the Setpoint value, and the thermostat is switched on or off accordingly. The Differential slot value provides a deadband for the current input value. The Thermostat component may be set in the Direct mode (reflecting a cooling process) or in the Reverse mode (reflecting a heating process).

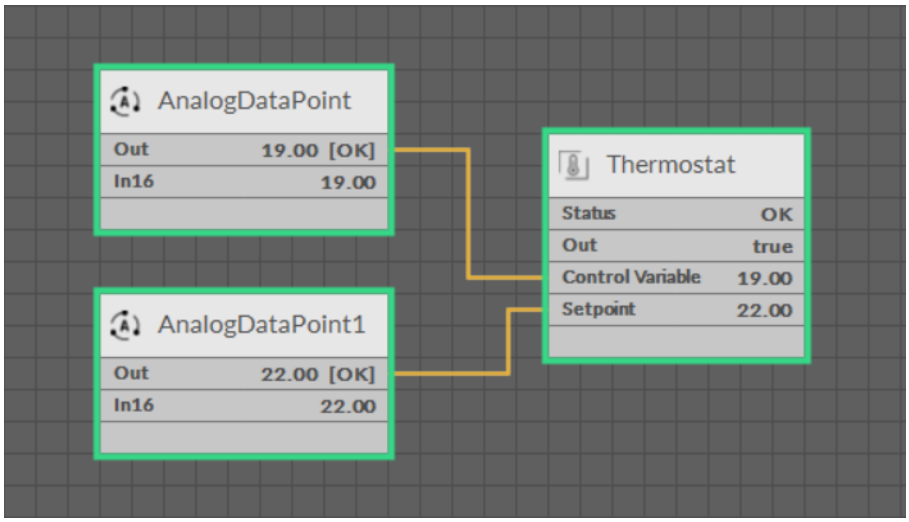


Figure 101. The Thermostat component

Slots

The Thermostat component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** indicates whether a thermostat is switched on or off;
- **Control Variable:** the current input value (measured temperature);
- **Setpoint:** the setpoint value (temperature setpoint);
- **Differential:** the deadband set for the current input value–if the Setpoint value is, for example, 25, the Diff value set to 5 means that for the Cv slot values from 22.5 to 27.5 no action is taken;
- **Mode:** allows to choose a Direct or Reverse operation mode of the component–the Direct mode reflects the cooling process, and the Reverse mode reflects the heating process.

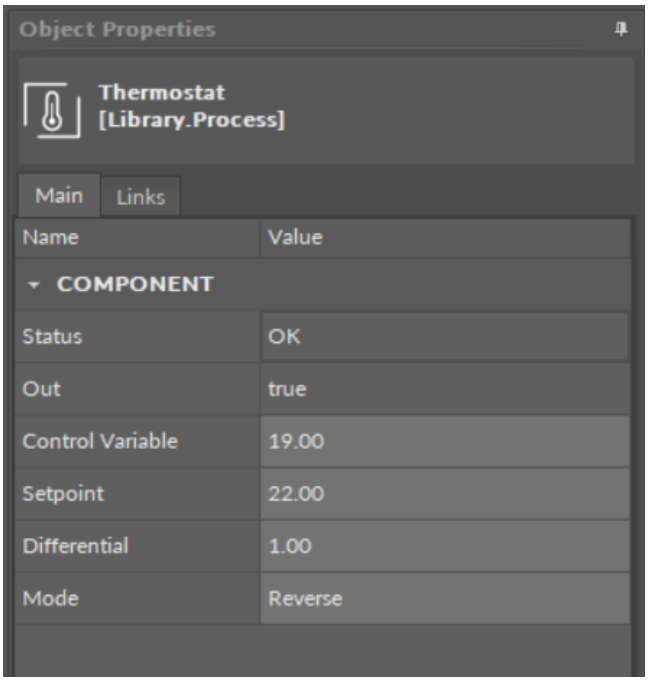


Figure 102. The Thermostat component slots

6.7 Time

Applicable to library's version 1.0

The Time library includes components representing time management operations essential for creating logics. The components in this library operate on Boolean input and output values.

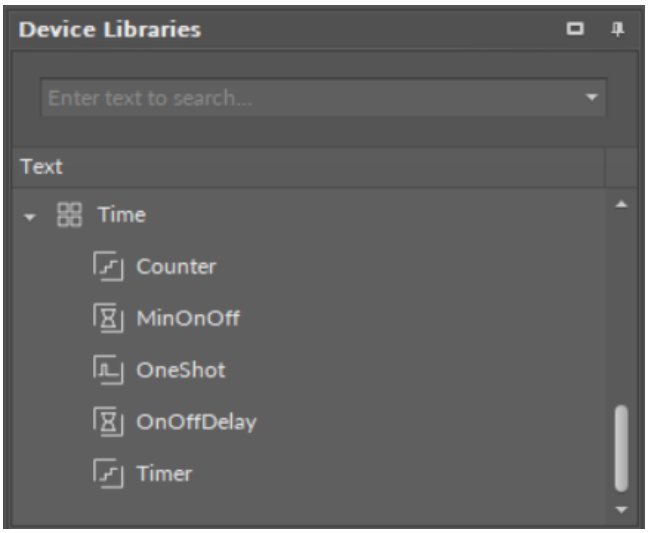


Figure 103. The Time library

6.7.1 Counter

Applicable to library's version 1.0

The Counter component adds up the number of rising edges received on the In slot. The counting direction in the component may be set upwards or downwards, and counting steps are set by integer values. If needed, the starting value may be set for the component (for example, counting starts at 100 and goes down every rising edge; 0 is not a threshold).

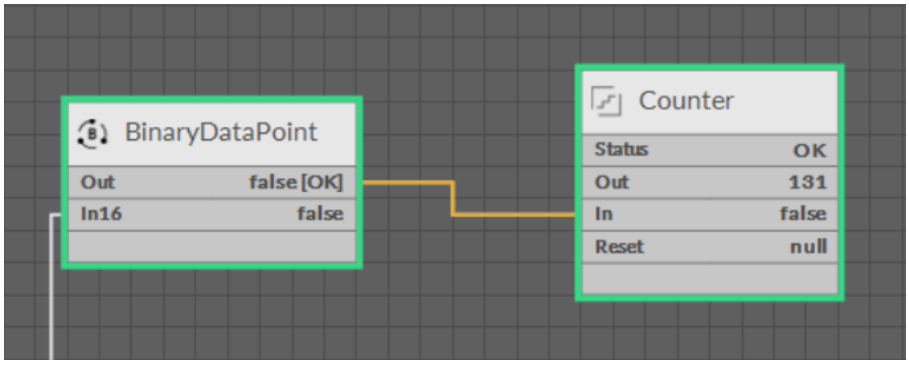


Figure 104. The Counter component

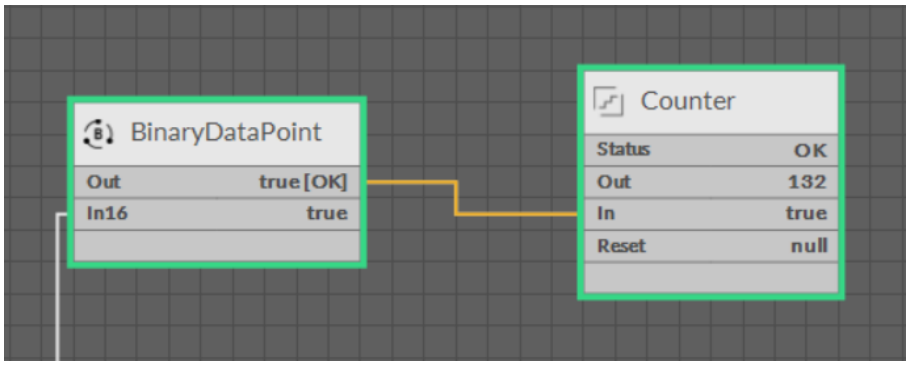


Figure 105. The Counter component

Slots

The Counter component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** the value representing the number of rising edges received on the In slot and taking into account the counting steps set in the Inc slot: if the Inc slot value is 2, and the In slot received 2 rising edges, the value transferred to the Out slot is 4;
- **In:** the input slot receiving Boolean values from linked components;
- **Reset:** sets the Out slot value to the value set in the Init Value slot;
- **Init Value:** the initial value of the Counter (set manually); by default, the initial value is set to 0;
- **Increment:** counting steps (set manually); by default, counting steps are set to 1;
- **Direction:** the counting direction (set manually); by default, the counting direction is set to Up.

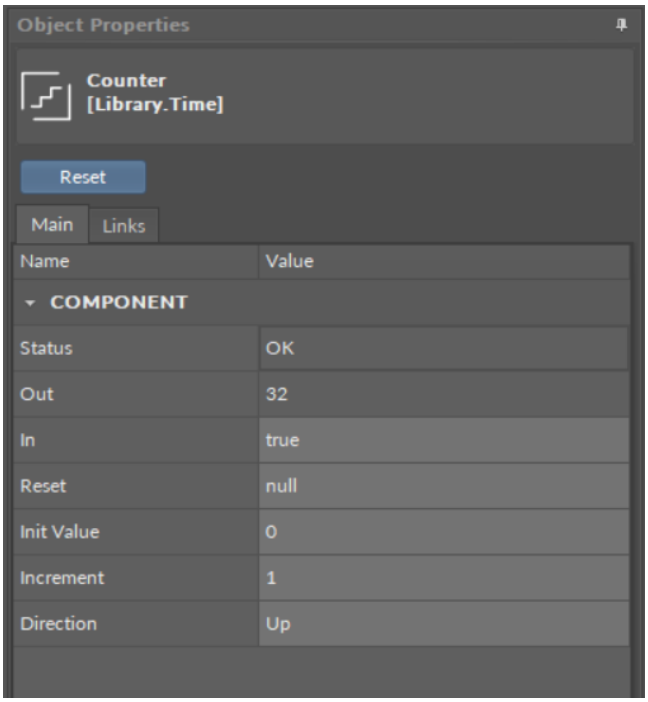


Figure 106. The Counter component's slots

Action

The Counter component has the following action:

- **Reset:** sets the Out slot value to the value set in the Init Value slot.

6.7.2 MinOnOff

Applicable to library's version 1.0

The MinOnOff component holds the value in the Out slot for the specified time. The value in the Time Base slot indicates the minimum time unit for how long the Out slot value will be held; the holding time is specified in the Min On slot for rising edges and Min Off slot for falling edges. Once a new value is received on the In slot, it is passed to the Out slot, and (supposing it is rising edge) it is held in the Out slot for the specified time. In case the In slot receives falling edge before the specified time passes, the Out value is not changed until this time ends —only then the Out value is changed to false. This mechanism allows to protect devices, which are sensitive to frequent value changes (for example, compressors).

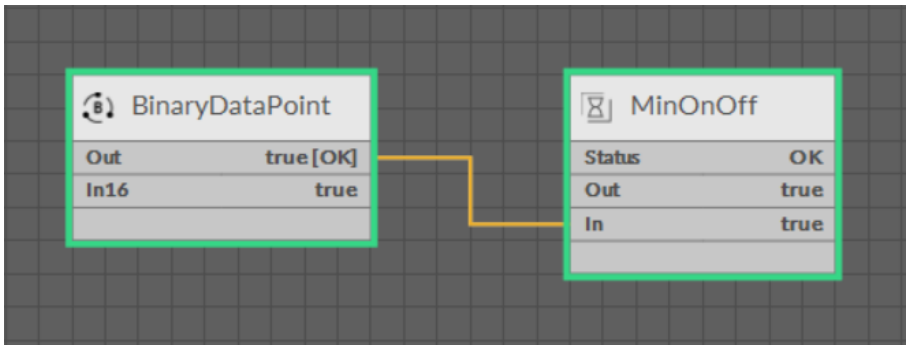


Figure 107. The MinOnOff component

Slots

The MinOnOff component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault, Disabled);
- **Enabled:** the change of the slot's value enables or disables the component's mechanism—if enabled, in case of change to the input value, the component holds the value in the Out slot for the specified time, and then transfers the changed value to the Out slot; if disabled, in case of change to the input value, the component transfers the input value directly to the Out slot;
- **Out:** receives the value passed from the In slot;
- **Min On Active:** indicates that, regardless of the rising edge in the In slot, the Out slot value is being withheld for specified time—if the slot indicates true, it means that the In slot value has changed to the rising edge, but the Min On slot is activated;
- **Min Off Active:** indicates that, regardless of the falling edge in the In slot, the Out slot value is being withheld for specified time—if the slot indicates true, it means that the In slot value has changed to the falling edge, but the Min Off slot is activated;
- **Left:** indicates how many rounds of the Time Base is left until the activated Min On or Min Off option allows the value to be passed to the Out slot;
- **In:** the input slot receiving Boolean values from linked components;

- **Min On:** indicates the holding time for the value on the rising edge, based on the time unit defined in the Time Base slot; the Min On value shall be set to an integer number, and by default it is set to 0, which means the value is passed immediately from the In slot to the Out slot;
- **Min Off:** indicates the holding time for the value on the falling edge, based on the time unit defined in the Time Base slot; the Min Off value shall be set to an integer number, and by default it is set to 0, which means the value is passed immediately from the In slot to the Out slot;
- **Time Base:** the time unit, which may be set manually to milliseconds, seconds, minutes, or hours.

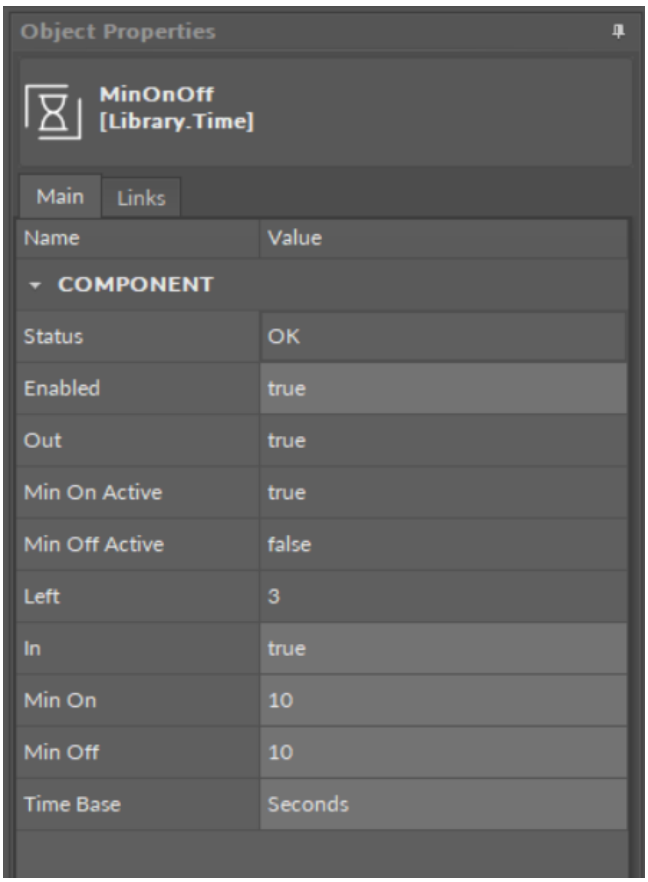


Figure 108. The MinOnOff component slots

6.7.3 OneShot

Applicable to library's version 1.0

The OneShot component creates a short pulse (from false to true and backwards) on the rising edge received in the In slot. The pulse lasts for the time specified in the Pulse slot. Additionally, the OneShot component has the N Out slot, which always presents an inverted value of the Out slot.

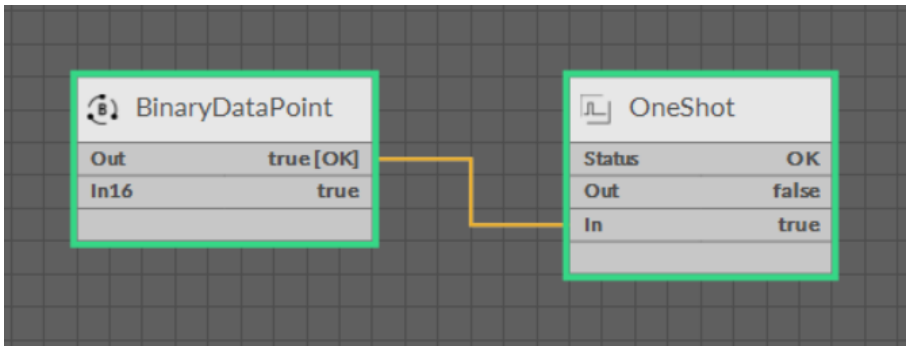


Figure 109. The OneShot component

Slots

The OneShot component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** creates the pulse (from false to true and backwards) on the rising edge received in the In slot;
- **N Out:** the value always opposite to the Out slot value;
- **In:** the input slot receiving Boolean values from linked components;
- **Pulse:** indicates the pulse duration, based on the time unit defined in the Time Base slot;
- **Time Base:** the time unit expressed in seconds [s]; the time unit may be set manually by the user (to milliseconds, minutes, hours, etc.).

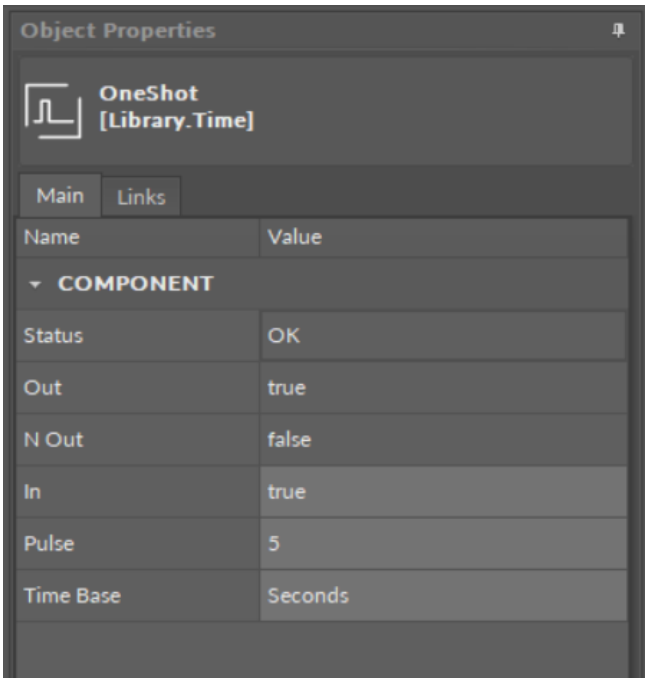


Figure 110. The OneShot component slots

6.7.4 OnOffDelay

Applicable to library's version 1.0

The OnOffDelay component stops a transfer of the value from the In slot to the Out slot for a time specified in the On Delay slot for rising edges and Off Delay slot for falling edges.

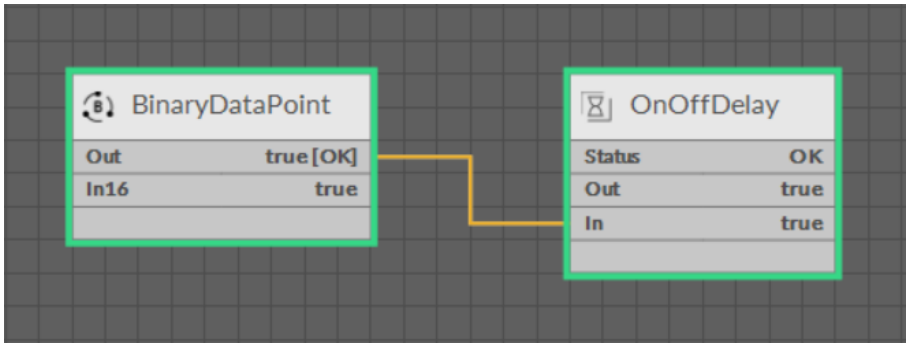


Figure 111. The OnOffDelay component

Slots

The OnOffDelay component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault, Disabled);
- **Enabled:** the change of the slot's value enables or disables the component's mechanism— if enabled, the component delays transferring the changed input value to the Out slot for the specified time; if disabled, the component directly transfers the input value to the Out slot;
- **Out:** receives the value passed from the In slot, after executing all set delays;
- **On Delay Active:** indicates whether the In slot value on the rising edge is being delayed from being passed to the Out slot—if the slot indicates true, it means that the In slot value has changed to the rising edge, but the On Delay slot is activated, so the value will not be passed to the Out slot until the specified time passes;
- **Off Delay Active:** indicates whether the In slot value on the falling edge is being delayed from being passed to the Out slot—if the slot indicates true, it means that the In slot value has changed to the falling edge, but the Off Delay slot is activated, so the value will not be passed to the Out slot until the specified time passes;
- **Left:** indicates how many rounds of the Time Base is left until the activated On Delay or Off Delay option allows the value to be passed to the Out slot;
- **In:** the input slot receiving Boolean values from linked components;
- **On Delay:** sets the delay time for the value on the rising edge, based on the time unit defined in the Time Base slot; the On Delay value shall be set to an integer number, and by default it is set to 0, which means the value is passed immediately from the In slot to the Out slot;
- **Off Delay:** sets the delay time for the value on the falling edge, based on the time unit defined in the Time Base slot; the Off Delay value shall be set to an integer number, and by default it is set to 0, which means the value is passed immediately from the In slot to the Out slot;
- **Time Base:** the time unit expressed in seconds [s]; the time unit may be set manually by the user (to milliseconds, minutes, hours, etc.).

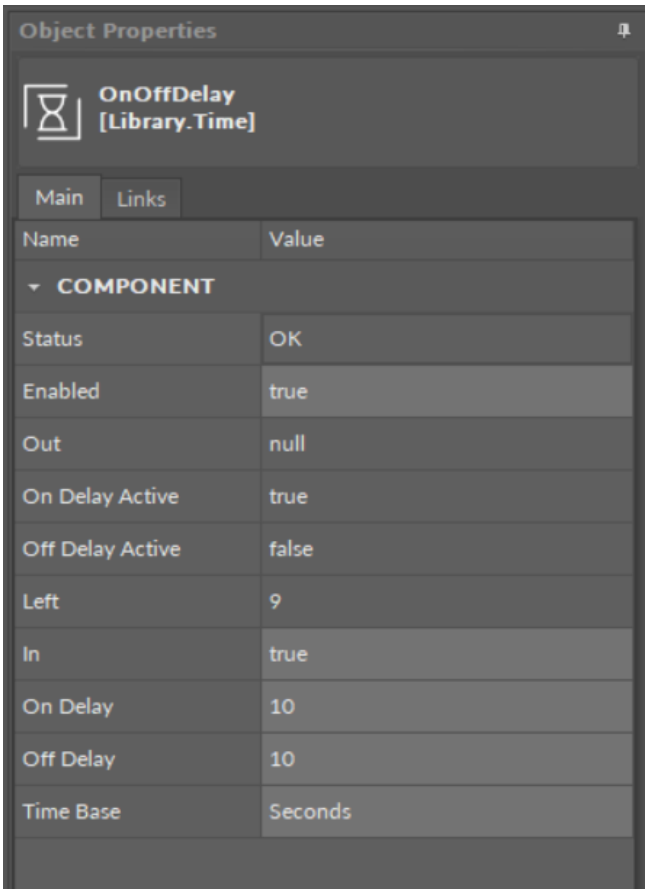


Figure 112. The OnOffDelay component's slots

6.7.5 Timer

Applicable to library's version 1.0

The Timer component counts the time up or down. The timer can be set with an initial value (the Init Value slot), which is a base for the timer to count up or down from. The component has the Reset action, which sets the timer back to the initial value. The timer is started with a rising edge of the Run slot and stopped on its falling edge. The component can be configured to count the time in milliseconds, seconds, minutes, and hours.

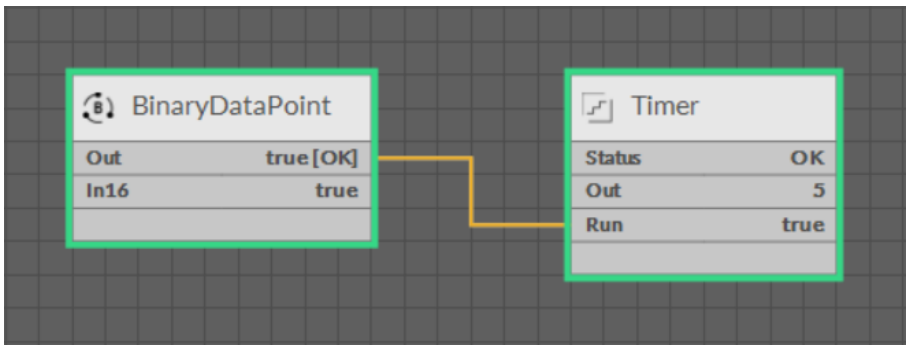


Figure 113. The Timer component

Slots

The Timer component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Out:** shows the time value counted up or down;
- **Run:** starts counting of the time;
- **Init Value:** sets the initial value, which is a base for the timer to count up or down from;
- **Direction:** set the counting direction (up or down);
- **Time Base:** defines the time unit of counting:
    - Available settings: milliseconds, seconds, minutes, hours.

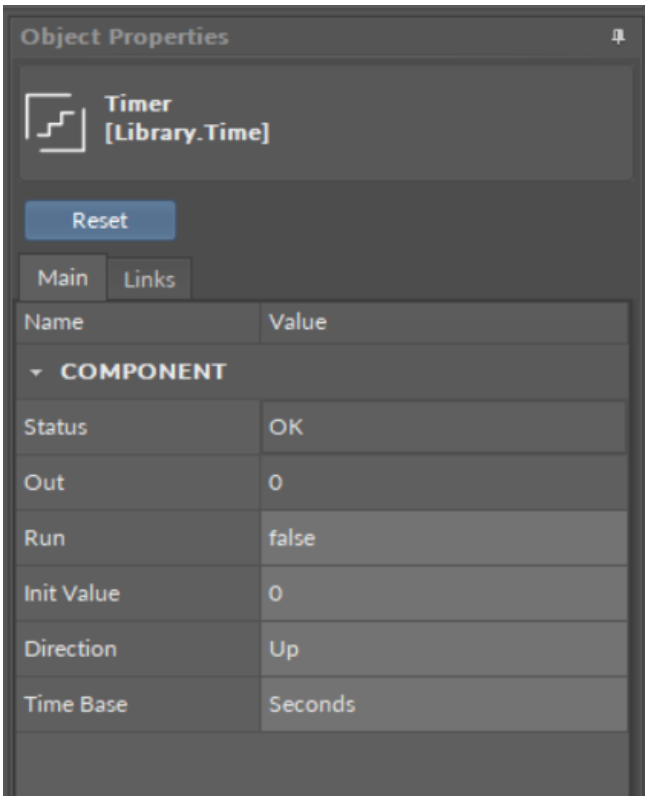


Figure 114. The Timer component slots

Action

The Timer component has the following action:

- **Reset:** sets the timer back to the initial value.

6.8 FCU

Applicable to library's version 1.0

The FCU library contains components to create FCU applications.

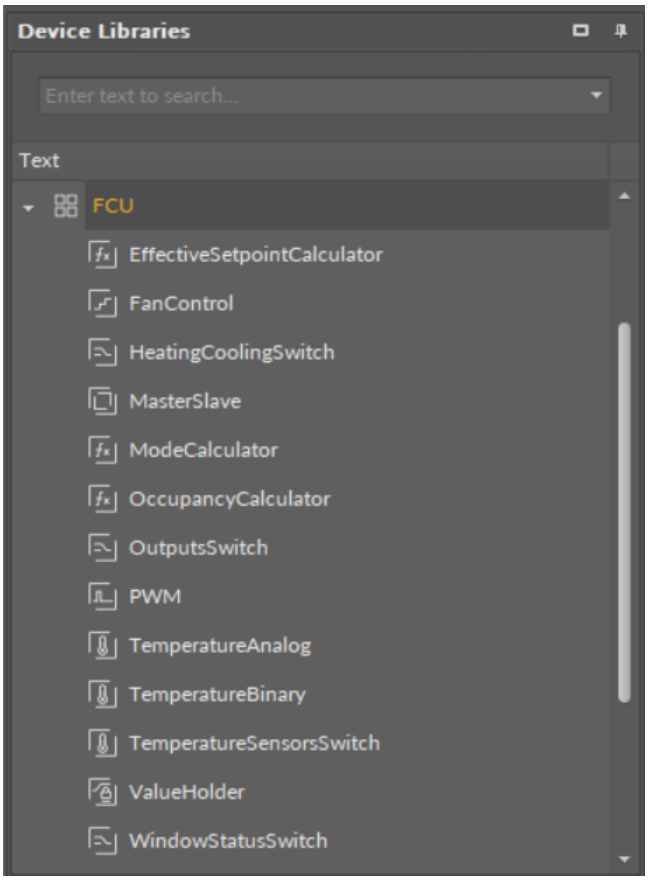


Figure 115. The FCU library

6.8.1 EffectiveSetpointCalculator

Applicable to library's version 1.0

The EffectiveSetpointCalculator is a component, which calculates the value of the effective temperature setpoint, according to the setpoint value from the upper-level system (for example, BMS), offset of this value, occupancy mode, and the heating or cooling modes.

Slots

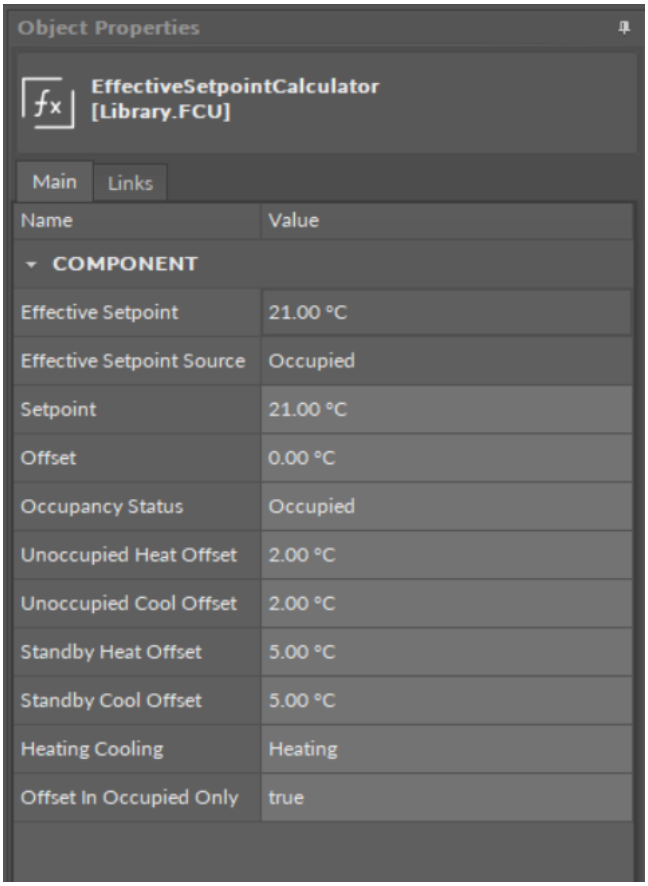


Figure 116. The EffectiveSetpointCalculator component slots

The EffectiveSetpointCalculator component has the following slots:

- **Effective Setpoint:** the main output slot of a component, which value is equal to the calculated effective temperature setpoint;
- **Effective Setpoint Source:** displays information about the way the setpoint is actually calculated, depending on the occupancy status and heating or cooling mode setting;
  - Available information: 1 (Occupied), 2 (Unoccupied_Heating), 3 (Unoccupied_Cooling), 4 (Standby_Heating), 5 (Standby_Cooling);
- **Setpoint:** the main input of component, which receives the value of the temperature setpoint from the upper-level system;
- **Offset:** the value of the setpoint offset;
- **Occupancy Status:** sets the occupancy status;
  - Available settings: 0 (Unoccupied), 1 (Occupied), 2 (Standby);
- **Unoccupied Heat Offset:** the offset value subtracted from the setpoint in the Unoccupied mode when the algorithm works in the heating mode;
- **Unoccupied Cool Offset:** the offset value added to the setpoint in the Unoccupied mode when the algorithm works in the cooling mode;
- **Standby Heat Offset:** the offset value subtracted from the setpoint in the Standby mode when the algorithm works in the heating mode;
- **Standby Cool Offset:** the offset value added to the setpoint in the Standby mode when the algorithm works in the cooling mode;
- **Heating Cooling:** sets the temperature mode;

    - Available settings: heating, cooling;
- Offset In Occupied Only: determines whether the calculation of the Effective Setpoint value is to be included in the process of calculating the value of the Offset slot if a component is not in the Occupied mode;
    - Available settings: true (means that for the Unoccupied and Standby modes, the value of the Offset slot is not to be included in calculating the Effective Setpoint), false (the value of the Offset slot is to be included in calculating the Effective Setpoint in all occupancy modes).

6.8.2 FanControl

Applicable to library's version 1.0

The FanControl is a component to control the fan. The component has been created for 1-, 2-, or 3-speed fans, and for fans with an analog input (the type of the fan can be selected by the user). The fan algorithm can be split into two modes:

- Standard: the demand for the fan is active, and the fan speed is calculated based on the temperature value. The standard mode conditions:
    - the FanDemand slot is false;
    - the Antifrost slot is false;
    - the HeatingOccupiedActive slot is false;
    - the CoolingOccupiedActive islot s false;
    - the TestMode slot is 0.
- Non-standard: the additional parameters override the fan speed. The non-standard mode must comprise at least one of the slots: FanDemand, Antifrost, HeatingOccupiedActive, CoolingOccupiedActive, is true, or the Test Mode is higher than 0.

In the standard mode, the fan is switched on when the internal variable, the Fan Control Value (calculated on the basis of the Control Variable and Setpoint slots), is higher than the Fan Speed 1 Threshold, and switched off when the Fan Control Value is lower than the Fan Off Threshold.

The non-standard operation is defined by the slots states combinations and is described below.

Slots

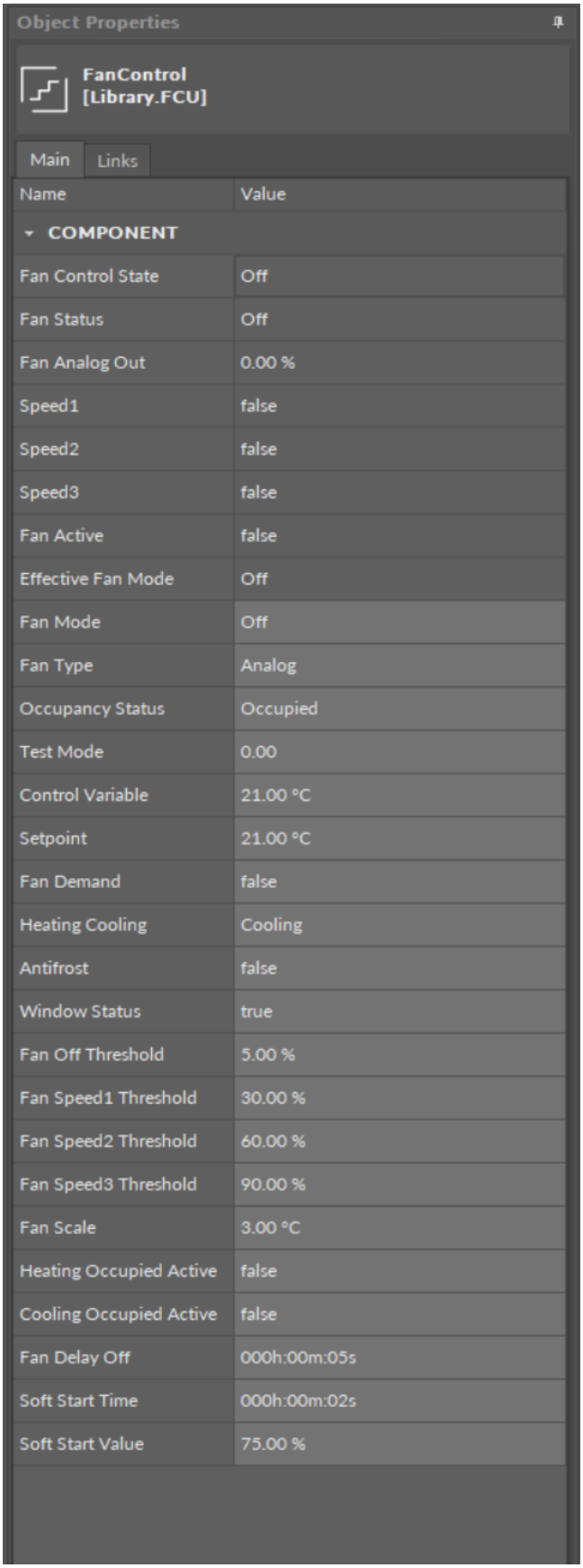


Figure 117. The FanControl component slots

The FanControl component has the following slots:

- **Fan Control State:** indicates the control state of the fan;
  - Available information: Running, Switching Speeds, DelayedOff, Off;
- **Fan Status:** indicates the current status of the fan;
  - Available information: Off, Speed 1 (Manual), Speed 2 (Manual), Speed 3 (Manual), Speed 1 (Auto), Speed 2 (Auto), Speed 3 (Auto), Analog Control;
- **Fan Analog Out:** the component's output for the fan with analog input, expressed as percentage; for fans with discrete inputs (the Fan Type slot is set to 1, 2 or 3), the value of the Fan Analog out is equal to 0%;
- **Speed 1, Speed 2, Speed 3:** the component's outputs for fans with binary inputs; for fans with analog inputs (the Fan Type slot is set to 0), states of the Speed 1, Speed 2, and Speed 3 slots are set to false, and cannot be changed by the algorithm of the component;

Note: The FanControl component has a built-in protection against enabling several speeds at the same time, which could cause physical damage to the fan. If the current fan speed has to be changed to another one, all binary outputs responsible for fan speeds are disabled for 1 second, and only then the new speed is enabled.

- **Fan Active:** allows to confirm the operation of the fan;
- **Effective Fan Mode:** controls the actual mode of the fan depending on the mode set in the Fan Mode slot and its overriding conditions;
- **Fan Mode:** the main input of the component;
  - Available settings: Off, Speed1(Manual), Speed2(Manual), Speed3(Manual), Auto

Fan Modes

- Speed1(Manual): the fan works with speed 1, regardless of temperature values. If the Fan Type slot is set to 0 (fan with analog input), the value of the Fan Analog Out is set to the value from the Fan Speed 1 Threshold slot.
- Speed2(Manual): the fan works with speed 2, regardless of temperature values. If the Fan Type slot is set to 0 (fan with analog input), the value of the Fan Analog Out is set to the value from the Fan Speed 2 Threshold slot.
- Speed3(Manual): the fan works with speed 3, regardless of temperature values. If the Fan Type slot is set to 0 (fan with analog input), the value of the Fan Analog Out is set to the value from the Fan Speed 3 Threshold slot.
- Auto: the fan works in the automatic mode, the current speed depends on the current space temperature and the setpoint.

Note: The value of Fan Mode slot (or the current speed, without changing the Fan Mode slot) can be overridden by the built-in algorithm of the component, disregarding the value that is set to the Fan Mode slot. It can occur in the following cases:

- The component works in the Unoccupied or Standby mode (the value of the Occupancy Status slot is set to Unoccupied or Standby)–the Fan Mode slot is overridden to the Auto mode always when the set value is different than Off. The overriding stops if the component works in the Occupancy mode (value of the Occupancy Status slot is set to Occupied).
- The window is open (the Window Status slot is set to false)–the Fan Mode slot is overridden to the Off mode. The overriding stops when the Window Status slot changes to true.
- The component works in the Antifrost mode (the Antifrost slot is set to true, even the Window Status slot is set to false)–the current speed is overridden by the maximum value available for the type of the fan (depending on the value of the Fan Type slot). The overriding stops when the Antifrost slot changes to false.
- The component works in the testing mode (the value of the Test Mode slot is not equal to 0)–the current speed is overridden by the maximum value available for the type of the fan (depending on the value of the Fan Type slot). The overriding stops when the Test Mode slot changes to 0.

- Fan Type: sets the type of the controlled fan;
  - Available settings: Analog, 1Speed, 2Speed, 3Speed;

Note: The FanControl component has a built-in protection against enabling speeds higher than these resulting from the value of the Fan Type slot. For example, if the Fan Type slot is set to 1Speed, it is not possible to enable speeds higher than 1. This protection pertains only to fans with binary outputs.

- Occupancy Status: sets the occupancy status.
  - Available settings: Unoccupied, Occupied, Standby;
- Test Mode: allows to enable or disable the testing mode. This mode is inactive if the value of the slot equals 0. In other cases, fan works in the testing mode–the current speed will be overridden by the maximum value available for the fan type (depending on the value of the Fan Type slot);

- **Control Variable:** the measured temperature, which is used for calculating the fan speed in the Auto mode;
- **Setpoint:** sets the temperature setpoint, which is used for calculating the fan speed in the Auto mode;
- **Fan Demand:** allows to force switch the fan on if it is off (the Fan Active slot is set to false);

Fan Demand

If the fan is off (the **Fan Mode** slot is set to Off), or it works in the Auto mode (the **Fan Mode** slot is set to Auto), but the speed calculated by the algorithm equals 0, the fan can be switched on by setting the **Fan Demand** slot to true. In this case, the fan works with speed 1 (for fans with binary inputs), or with the analog value set in the **Fan Speed 1 Threshold** slot (for the fan with analog input). If the speed (or analog output) calculated by the algorithm is higher than speed 1 (or the value from the **Fan Speed 1 Threshold** slot, for analog output), the speed switched on using the **Fan Demand** slot, is overridden by this value.

- **Heating Cooling:** allows to set the temperature mode, which the fan works in;
  - Available settings: Heating, Cooling;
- **Antifrost:** allows to switch on the Antifrost mode;
  - Available settings: true (Antifrost mode enabled, the current speed will be overridden by the maximum value available for the fan type, depending on the value of the Fan Type slot), false (Antifrost mode disabled);
- **Window Status:** allows to enable the Window Open mode;
  - Available settings: true (Window Open mode disabled), false (Window Open mode enabled, the current value of the Fan Mode slot will be overridden to Off);

Note: The Window Open mode can be overridden only by the Antifrost mode or the Test mode.

- **Fan Off Threshold, Fan Speed 1 Threshold, Fan Speed 2 Threshold, Fan Speed 3 Threshold:** set values of thresholds used for switching fan speeds (in Auto mode, for fans with binary inputs), calculating the value of the Fan Analog Out slot (in manual modes, for the fan with analog output);
- **Fan Scale:** sets the value, which is used for calculating the fan speed in the Auto mode;

Note: Calculating the fan speeds is based on the internal variable named Fan Control Value. The way of calculating this value is presented by the figure below:

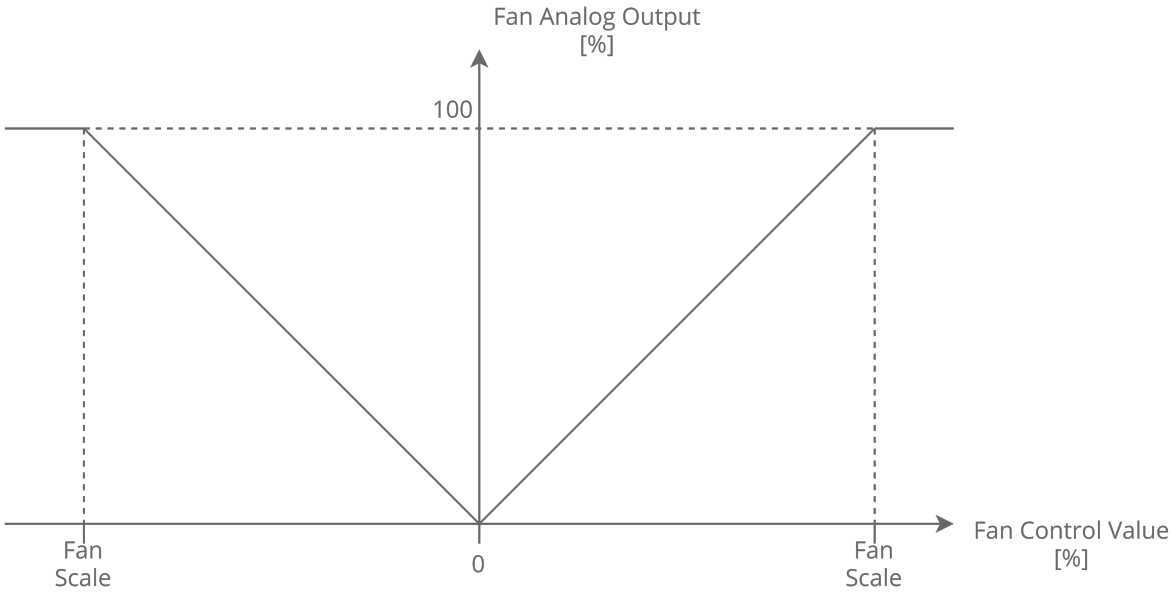


Figure 118. Way of calculation the Fan Control Value

The Fan Control Value is used to calculate the current speed of the fan (for fans with binary inputs), or the value of the Fan Analog Out slot (for fans with analog input). The way of calculating both values is presented by the below figures:

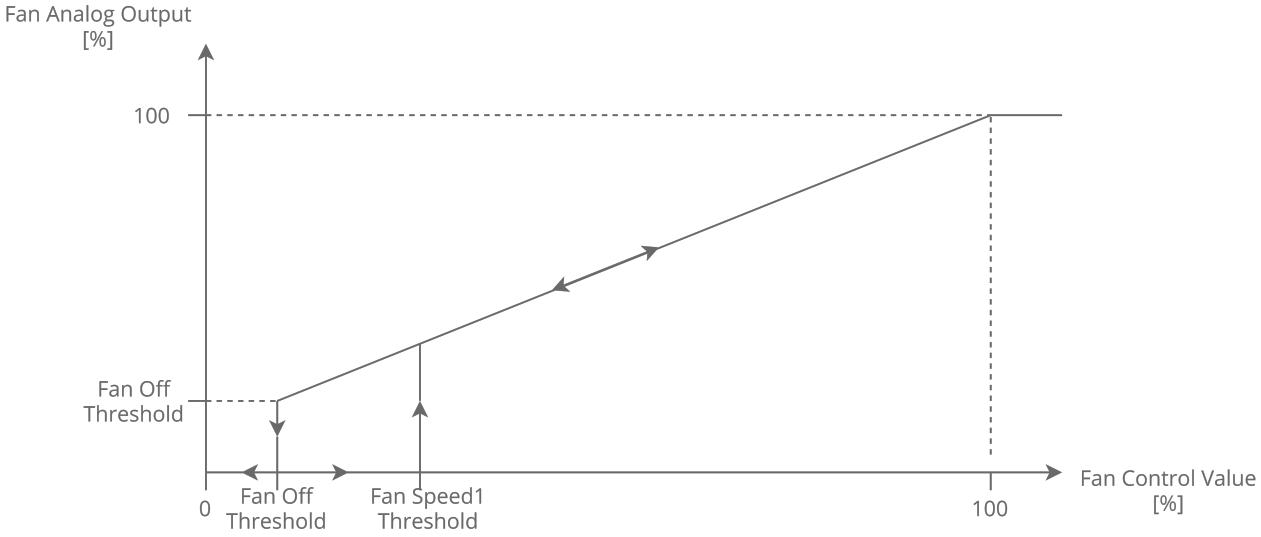


Figure 119. Controlling the fan with analog input in the Auto mode

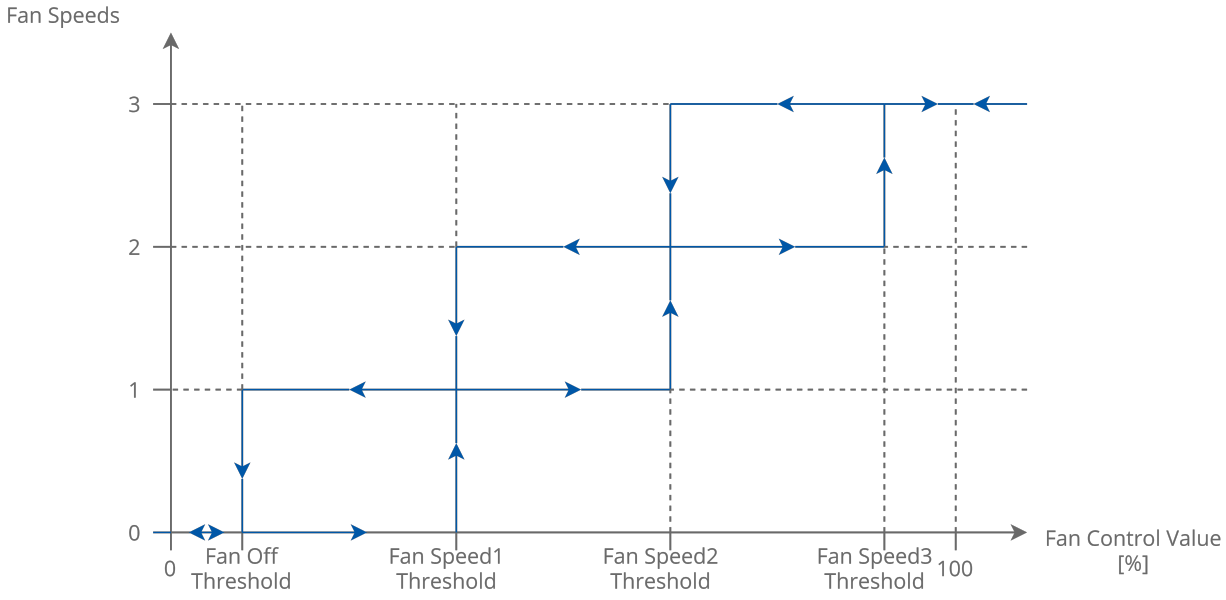


Figure 120. Controlling the fan with binary inputs in the Auto mode

- **Heating Occupied Active:** allows to enable or disable the function enforcing fan operation;
  - Available settings: true (enabled), false (disabled);

Heating Occupied Active Enabled

If the fan works in the Auto mode (the **Fan Mode** slot is set to Auto), the space is occupied (the **Occupancy Status** slot is set to Occupied), and the **FanControl** component works in the heating temperature mode (the **Heating Cooling** slot is set to Heating), the fan will always be switched on, even if the value of the **Setpoint** slot is lower than value of the **Control Variable** slot, in which case, according to the control algorithm, the fan should be switched off.

- **Cooling Occupied Active:** allows to enable or disable the function enforcing fan operation;
  - Available settings: true (enabled), false (disabled);

Cooling Occupied Active Enabled

If the fan works in the Auto mode (the **Fan Mode** slot is set to Auto), the space is occupied (the **Occupancy Status** slot is set to Occupied), and the **FanControl** component works in the cooling temperature mode (the **Heating Cooling** slot is set to Cooling), the fan will always be switched on, even if the value of the **Setpoint** slot is higher than the value of the **Control Variable** slot, in which case, according to the control algorithm, the fan should be switched off.

Note: The way of calculating the current speed of the fan (for fans with binary inputs) or value of Fan Analog Out slot (for the fan with analog input), when Cooling/Heating Occupied Active function is enabled, is presented in the figures below:

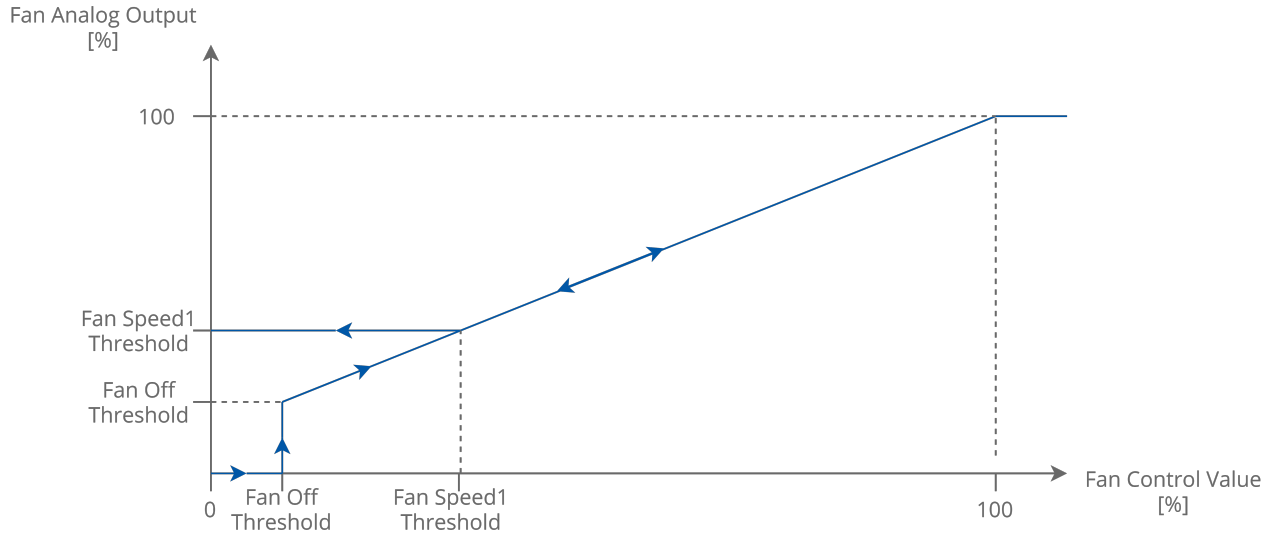


Figure 121. Heating/Cooling Occupied Active function for the fan with analog input

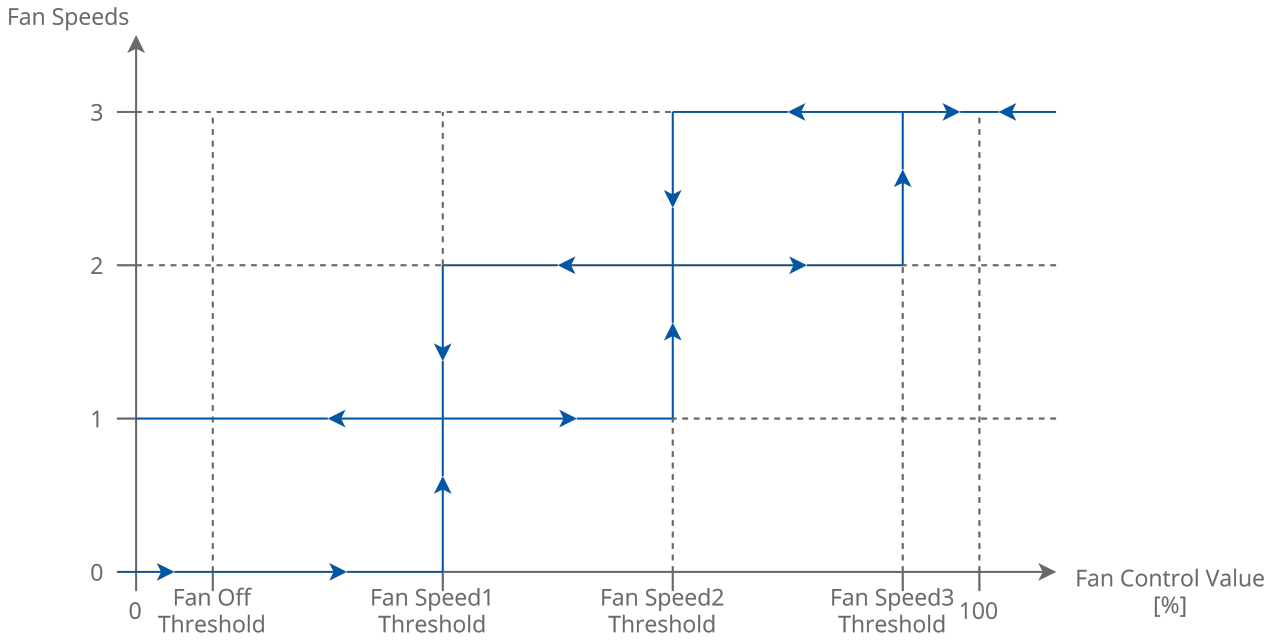


Figure 122. Heating/Cooling Occupied Active function for fans with binary inputs

- **Fan Delay Off:** sets the value of the delay-off time, expressed in seconds; each time, the fan should be switched off, it remains working for the time equal to the value of this slot; after this time, the fan will switch off. If the slot is set to 0, the function is disabled;
- **Soft Start Time:** sets the time, in which the fan is working in the Soft Start mode, expressed in seconds. If the slot is set to 0, the function is disabled.
- **Soft Start Value:** sets the value for the Soft Start mode, expressed as percentage.

Note: If the Soft Start Value is lower than the Fan Speed 1 Threshold, the value will be taken from the Fan Speed 1 Threshold slot.

Note: The Soft Start function is dedicated to fans with analog input. If the fan start with small control value ramp lasts too long or is impossible, overheating of the driver or motor can occur. In this function, the fan start output value will be increased to the Soft Star Value for the time defined in the Fan Soft Start Time. If the time of the soft start is

finished, the Fan Analog Out slot is set to the current value calculated by the algorithm of the component.

6.8.3 HeatingCoolingSwitch

Applicable to library's version 1.0

The HeatingCoolingSwitch component allows for switching between heating and cooling temperature modes, depending on the current temperature, the given setpoint, and the occupancy status.

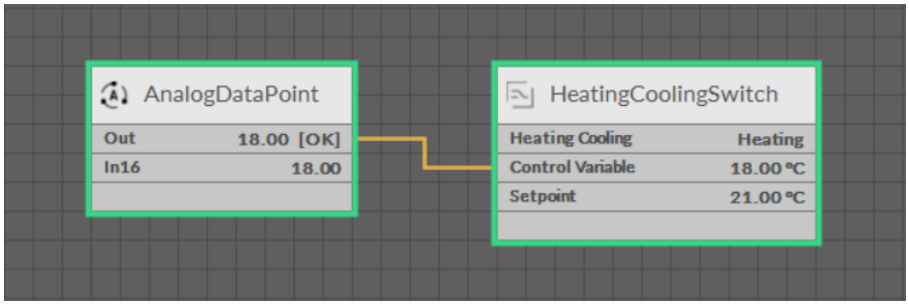


Figure 123. The HeatingCoolingSwitch component

Slots

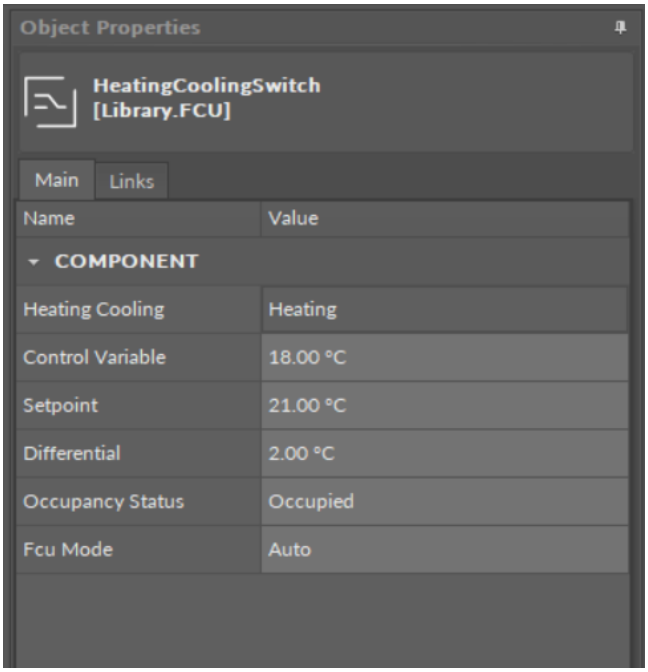


Figure 124. The HeatingCoolingSwitch component slots

The HeatingCoolingSwitch component has the following slots:

- **Heating Cooling:** the main output of the component; shows the operating status of the component–whether it works in cooling mode (false) or heating mode (true);
- **Control Variable:** the current input value (measured temperature),
- **Setpoint:** the setpoint value (temperature setpoint);
- **Differential:** the deadband set for the current input value–if the Setpoint value is, for example, 25, the Differential value set to 5 means that for the Control Variable slot values from 22.5 to 27.5 no action is taken;

- Occupancy Status: sets the occupancy status;
  - Available information: Unoccupied, Occupied, Standby;
- Fcu Mode: indicating the FCU mode, for example, from the higher-level system.
  - Available values: Off, Auto, Heating Only, Cooling Only, Fan Only.

Predefined modes for different values of the Fcu Mode slot

- Off: the Heating Cooling slot is permanently set to the heating mode (true), regardless of the measured temperature and setpoint values.
- Heating Only: the Heating Cooling slot is permanently set to the heating mode (true), regardless of the measured temperature and setpoint values.
- Cooling Only: the Heating Cooling slot is permanently set to the cooling mode (false), regardless of the measured temperature and setpoint values.
- Fan Only: the Heating Cooling slot is permanently set to the cooling mode (false), regardless of the measured temperature and setpoint values.
- Auto: the Heating Cooling slot switches between the heating and cooling modes, according to the figure below:

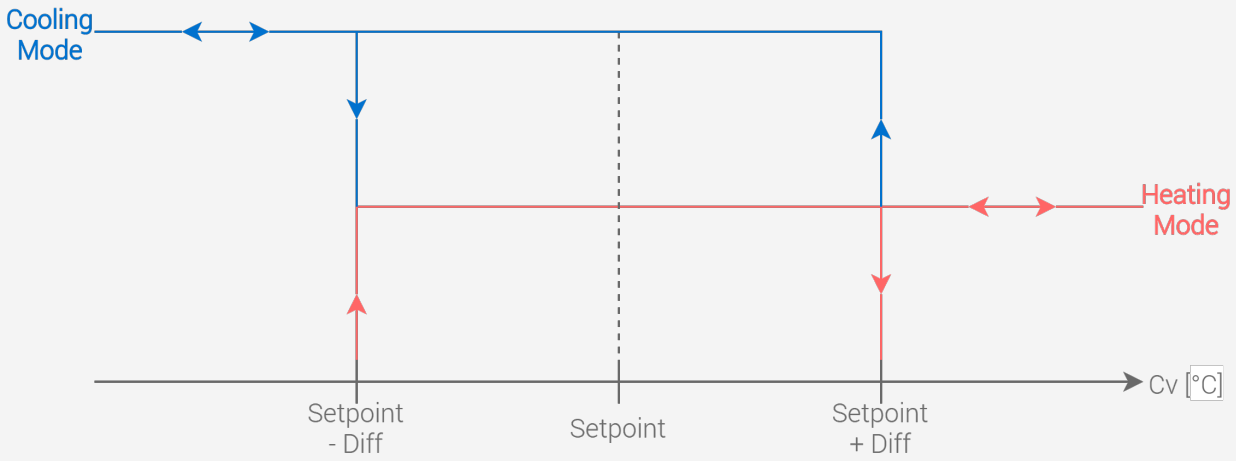


Figure 125. The Heating Cooling slot in the Auto Fcu mode

Note: The heating and cooling modes can only be switched, when the Occupancy Status slot is set to the Occupied mode. If the Occupancy Status slot is set to any other value, the Heating Cooling slot is set to the last mode, which has been calculated in the Occupied mode.

6.8.4 MasterSlave

Applicable to library's version 1.0

The MasterSlave component allows to automatically calculate the BACnet device ID of slave devices in the BACnet master-slave network, depending on the BACnet device ID of master devices. This function is called Auto Binding. The table below shows values of the master BACnet device ID and the corresponding BACnet device IDs of slave devices for the Auto Binding function:

Master Id	Slave 1 ID	Slave 2 ID	Slave 3 ID	Slave 4 ID	Slave 5 ID
826101	826001	826002	826003	826004	826005

Master Id	Slave 1 ID	Slave 2 ID	Slave 3 ID	Slave 4 ID	Slave 5 ID
826102	826006	826007	826008	826009	826010
826103	826011	826012	826013	826014	826015
826104	826016	826017	826018	826019	826020
826105	826021	826022	826023	826024	826025
826106	826026	826027	826028	826029	826030
826107	826031	826032	826033	826034	826035
826108	826036	826037	826038	826039	826040
826109	826041	826042	826043	826044	826045
826110	826046	826047	826048	826049	826050
826111	826051	826052	826053	826054	826055
826112	826056	826057	826058	826059	826060
826113	826061	826062	826063	826064	826065
826114	826066	826067	826068	826069	826070
826115	826071	826072	826073	826074	826075
826116	826076	826077	826078	826079	826080
826117	826081	826082	826083	826084	826085
826118	826086	826087	826088	826089	826090
826119	826091	826092	826093	826094	826095
826120	826096	826097	826098	826099	826100
Other	0	0	0	0	0

Table 5. Master-slave IDs - Auto Binding

The Auto Binding function can be disabled (by setting the Local Remote Auto Binding slot to true). In this case, the IDs of slave devices have to be set by the user (in the Remote Slave 1 Device Id-Remote Slave 5 Device Id slots).

Slots

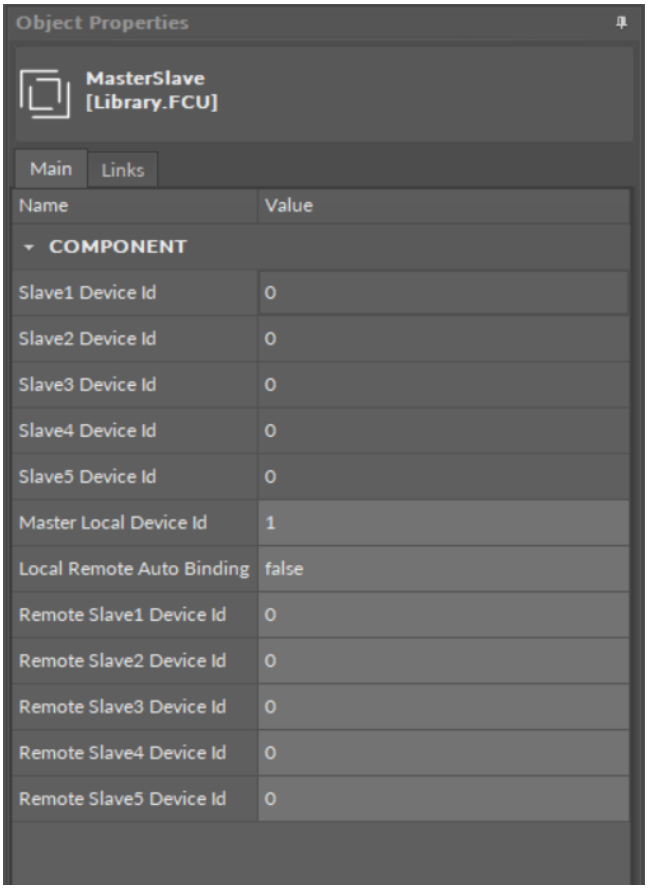


Figure 126. The MasterSlave slots

The MasterSlave component has the following slots:

- **Slave1 Device Id-Slave5 Device Id:** display the IDs calculated or set for five slave devices;
- **Master Local Device Id:** sets the ID of the master device.

Note: If the component uses the Auto Binding function, the value of the Master Local Device Id slot has to be set to the value ranged from 826101 to 826120. For other values, all output slots (Slave1 Device Id-Slave5 Device Id) will be set to 0.

- **Local Remote Auto Binding:** allows to switch between Auto Binding and Remote Binding functions;
  - Available settings: true (Remote Binding–IDs of each slave device are set to the corresponding values of the Remote Slave1 Device Id-Remote Slave5 Device Id slots), false (Auto Binding–IDs of each slave device are calculated according to the table above);
- **Remote Slave1 Device Id-Remote Slave5 Device Id:** allows to set remote IDs of slave devices.

6.8.5 ModeCalculator

Applicable to library's version 1.0

The ModeCalculator component allows to switch the main modes of application (for example, to override the fan mode according to others values, or to enable/disable temperature control), according to fan mode, Fcu Mode, and type of temperature control (analog or binary). The component can be used to protect against the incorrect operation of FCU device.

Slots

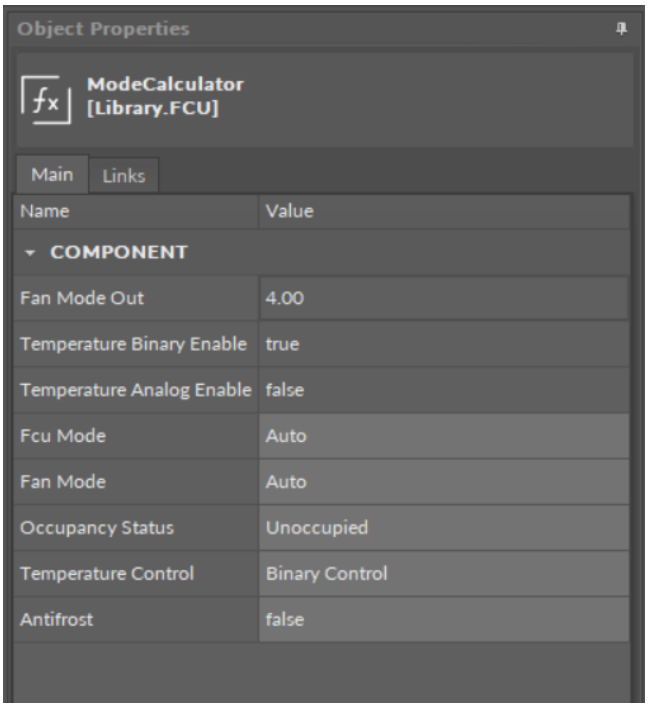


Figure 127. The ModeCalculator component slots

The ModeCalculator component has the following slots:

- **Fan Mode Out:** the output of the fan mode value, calculated according to the algorithm of component;

Modes Dependencies

If the value of the **Fcu Mode** slot is equal to 0 (FCU is switched off from the upper-level system), the **Fan Mode Out** slot is also set to 0 (the fan is also switched off).

If the value of the **Fan Mode** slot is lower than 4 (the fan is off, or works in one of the manual modes), and the value of the **Occupancy Status** slot is not equal to 1 (the component works in the unoccupied or standby mode), the **Fan Mode Out** slot is set to 4 (the fan works in the auto mode).

In other cases, the value of the **Fan Mode Out** slot is equal to the value of the **Fan Mode** slot.

- **Temperature Binary Enable:** enables or disables the binary temperature mode, according to the algorithm of component;

Temperature Binary Modes Dependencies

If the Temperature Control slot is set to false (binary control), and the value of the Fan Mode Out slot is higher than 0 (the fan is switched on), the Temperature Binary Enable slot is set to true.

If the Temperature Control slot is set to false (binary control), and the Antifrost slot is set to true (the component works in the antifrost mode), the Temperature Binary Enable slot is set to true.

In other cases, the Temperature Binary Enable slot is set to false.

- Temperature Analog Enable: enables or disables the analog temperature mode, according to the algorithm of the component;

Temperature Analog Modes Dependencies

If the Temperature Control slot is set to true (analog control), and the value of the Fan Mode Out slot is higher than 0 (the fan is switched on), the Temperature Analog Enable slot is set to true.

If the Temperature Control slot is set to true (analog control), and the Antifrost slot is set to true (the component works in the antifrost mode), the Temperature Analog Enable slot is set to true.

In other cases, the Temperature Analog Enable slot is set to false.

- FCU Mode: sets the mode of the FCU;
  - Available settings: Off, Auto, Heating Only, Cooling Only, Fan Only;

Note: If the FCU Mode slot is set to 4 (fan only), and the Antifrost slot is set to false (no antifrost mode), the Temperature Binary Enable and Temperature Analog Enable slots are overridden to false.

- Fan Mode: sets the fan mode;
  - Available settings: Off, Manual Speed 1, Manual Speed 2, Manual Speed 3, Auto;
- Occupancy Status: sets the occupancy status;
  - Available settings: Unoccupied, Occupied, Standby;
- Temperature Control: sets the mode of the temperature control;
  - Available settings: Analog Control, Binary Control;
- Antifrost: enables or disables the antifrost mode;
  - Available settings: true, false.

6.8.6 OccupancyCalculator

Applicable to library's version 1.0

The OccupancyCalculator component manages the occupancy status, depending on the occupancy data provided from the BMS system or remote devices such as room panels, presence sensors, or cardholders.

Slots

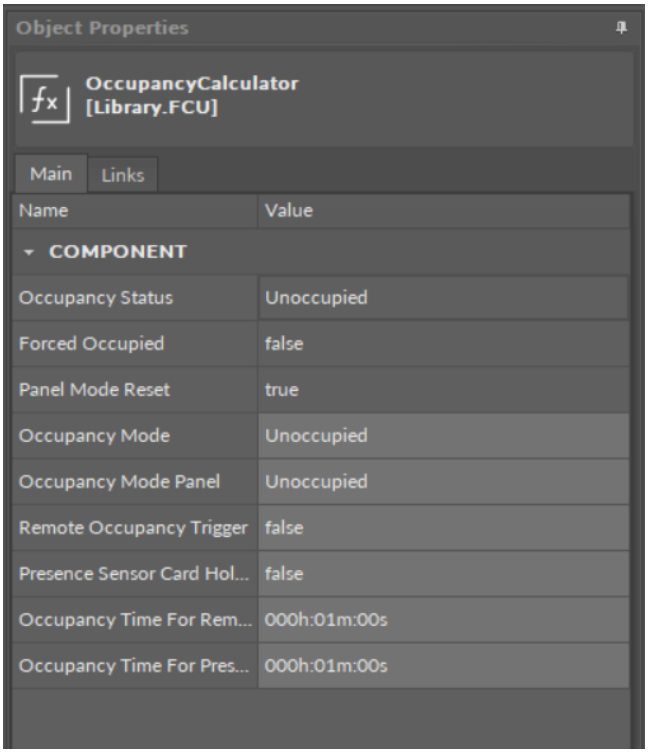


Figure 128. The OccupancyCalculator component slots

The OccupancyCalculator component has the following slots:

- **Occupancy Status:** shows the current occupancy status;
  - Available information: Unoccupied, Occupied, Standby;
- **Forced Occupied:** the binary output slot providing the information about the occupancy status source:
  - True: the occupied mode has been forced, which means, that the component works in the occupied mode, but this mode has not been calculated according to the Occupancy Mode slot, only forced by the room panel (the Occupancy Mode Panel slot set to Occupied), presence sensor cardholder (detected rising edge on the Presence Sensor Card Holder slot), or by the remote trigger (detected rising edge on the Remote Occupancy Trigger slot);
  - False: the occupied mode has not been forced;
- **Panel Mode Reset:** the binary output, resetting the value from a component connected to Occupancy Mode Panel slot after switching off the Occupancy Mode, which has been forced by the room panel. By default, the Panel Mode Reset slot is set to false. It is set to true for one application cycle if the time of the Occupancy mode forced by the room panel has passed;
- **Occupancy Mode:** the occupancy mode set from the higher-level system;
  - Available values: Unoccupied, Occupied, Standby;
- **Occupancy Mode Panel:** the occupancy mode set from an external source (for example, the iSMA-B-LP room panel);
  - Available values: Unoccupied, Occupied;

Note: The value of the Occupancy Mode Panel slot allows to force the occupied mode from the room panel. If the Occupancy Mode slot is set to Unoccupied or Standby values, setting the value of the Occupancy Mode Panel slot to Occupied forces the occupied

mode for the time defined in the Occupancy Time For Remote Trigger slot. During this time the Forced Occupied slot is set to true. After this time, the value of the Panel Mode Reset slot is set to true for one application cycle, and then component goes back to the previous occupancy mode. The occupied mode forced this way can be also cancelled by setting the Occupancy Mode Panel slot back to Unoccupied.

- Remote Occupancy Trigger: the binary input slot, recommended for remote occupancy triggers, allows to force the occupied mode. If the Occupancy Mode slot is set to Unoccupied or Standby, the rising edge detected on this slot will force the occupied mode for the time defined in the Occupancy Time For Remote Trigger slot. The time countdown starts when a falling edge is detected in the Remote Occupancy Trigger slot. During this time, the Forced Occupied slot remains set to true. After this time, the component goes back to the previous occupancy mode. The occupied mode forced this way cannot be cancelled.
- Presence Sensor Card Holder: the binary input slot, recommended for presence sensors or cardholders, allows to force the occupied mode. If the Occupancy Mode slot is set to Unoccupied or Standby, the rising edge detected on this slot will force the occupied mode for the time defined in the Occupancy Time For Presence Sensor slot. The time countdown starts when a falling edge is detected in the Presence Sensor Card Holder slot. During this time, the Forced Occupied slot remains set to true. After this time, the component goes back to the previous occupancy mode. The occupied mode forced this way cannot be cancelled.
- Occupancy Time For Remote Trigger: the time for which the occupancy mode, forced by the room panel or remote trigger, is held once the falling edge is detected on the Remote Occupancy Trigger slot;
- Occupancy Time For Presence Sensor: the time for which the occupancy mode, forced by the presence sensor or cardholder, is once the falling edge is detected on the Presence Sensor Card Holder slot.

6.8.7 OutputsSwitch

Applicable to library's version 1.0

The OutputSwitch component allows to manage outputs for the temperature and fan control, according to the FCU configuration (2- or 4-pipe system, analog or binary temperature control, etc.).

Slots

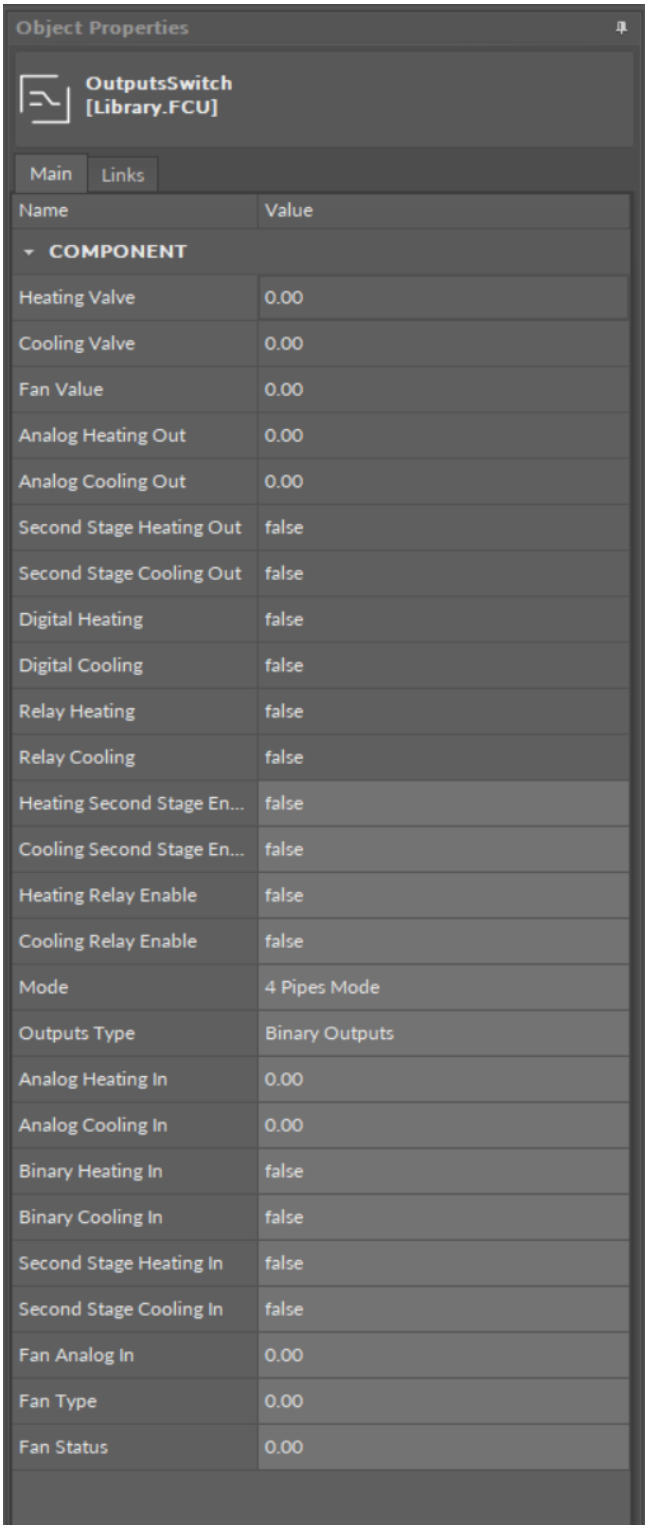


Figure 129. The OutputsSwitch component slots

The OutputSwitch component has the following slots:

- **Heating Valve:** displays the status of the heating valve; the status is displayed differently depending on the Analog Outputs Enable setting (binary or analog):
  - Available information for the temperature binary output: 0 (closed) or 1 (open);

    - Available information for the temperature analog output: displays the value of the Analog Heating In slot;
- **Cooling Valve:** displays the status of the the cooling valve;
    - Available information for the temperature binary output: 0 (closed) or 1 (open);
    - Available information for the temperature analog output: displays the value of the Analog Cooling In slot;
- **Fan Value:** displays the status of the fan; the status is displayed differently depending on the Fan Type setting (binary or analog):
    - Available information for the fan binary outputs: 0, 1, 2, or 3;
    - Available information for the fan analog outputs: displays the value of the Fan Analog In slot;

Binary Fan Values

The binary Fan Values are based on the values of the Fan Status slot:

- 0: Off (if the Fan Status slot is equal 0)
- 1: Speed 1 (if the Fan Status slot is equal 1 or 4);
- 2: Speed 2 (if the Fan Status slot is equal 2 or 5);
- 3: Speed 3 (if the Fan Status slot is equal 3 or 6).

- **Analog Heating Out:** the output slot for the analog heating valve;

Analog Heating Out Values

For the 2-pipe system (the **Mode** slot is set to 2 Pipes Mode), if the **Outputs Type** slot is set to Analog Outputs, the **Analog Heating Out** slot displays the value of the **Analog Heating In** slot or **Analog Cooling In** slot, depending on which slot has the value greater than 0.

For the 4-pipe system (the **Mode** slot is set to 4 Pipes Mode), the **Analog Heating Out** slot can only display the value of the **Analog Heating In** slot.

If the **Outputs Type** slot is set to Binary Outputs, the **Analog Heating Out** slot is set to 0.

- **Analog Cooling Out:** the output slot for the analog cooling valve;

Analog Cooling Out Values

For the 4-pipe system (the **Mode** slot is set to 4 Pipes Mode), the **Analog Cooling Out** slot displays the value of the **Analog Cooling In** slot.

If the **Outputs Type** slot is set to Binary Outputs, or the **Mode** slot is set to 2 Pipes Mode (for the 4-pipe system), the **Analog Cooling Out** slot is set to 0.

- **Second Stage Heating Out:** the output slot for the second stage heating–the slot displays the value from the Second Stage Heating In slot;

Note: If the Heating Second Stage Enable slot is set to false, or the Heating Relay Enable slot is set to false, the Second Stage Heating Out slot cannot be set to true.

- **Second Stage Cooling Out:** the output slot for the second stage cooling–the slot displays the value from the Second Stage Cooling In slot;

Note: If the Cooling Second Stage Enable slot is set to false, or the Cooling Relay Enable slot is set to false, the Second Stage Cooling Out slot cannot be set to true.

- **Digital Heating:** the output slot for the digital heating (recommended to service the heating valve switched on/off by triacs);

Digital Heating Values

For the 2-pipe system (the **Mode** slot is set to 2 Pipes Mode), if the **Outputs Type** slot is set to Binary Outputs, the **Digital Heating** slot displays the value of the **Binary Heating In** slot or the **Binary Cooling In** slot, depending on which slot has the true value.

For the 4-pipe system (the **Mode** slot is set to 4 Pipes Mode), the **Digital Heating** slot can only display the value of **Binary Heating In** slot.

If the **Outputs Type** slot is set to Analog Outputs, the **Digital Heating** slot is set to false.

- **Digital Cooling:** the output slot for the digital cooling (recommended to service the cooling valve switched on/off by triacs);

Digital Cooling Values

For the 4-pipe system (the **Mode** slot is set to 4 Pipes Mode), the **Digital Heating** slot displays the value of **Binary Cooling In** slot.

If the **Outputs Type** slot is set to Analog Outputs, or the **Mode** is set to 2 Pipes Mode (for the 4-pipe system), the **Digital Cooling** slot is set to false.

- **Relay Heating:** the output slot for digital heating in the first or second stage (recommended to service the heating valve switched by the relay output or electrical heaters);

Relay Heating Values

If the **Heating Relay Enable** slot is set to true, and the **Heating Second Stage Enable** slot is set to false (heating in the first stage only), the value from the **Binary Heating In** slot is set to the **Relay Heating** slot.

If the **Heating Relay Enable** slot is set to true, and the **Heating Second Stage Enable** slot is set to true (heating in the first and second stage), the value from the **Second Stage Heating In** slot is set to the **Relay Heating** slot.

If the **Heating Relay Enable** slot is set to false (the heating relay is disabled), the **Relay Heating** slot is set to false.

- **Relay Cooling:** the output slot for the digital cooling in the first or second stage (recommended to service the cooling valve switched by the relay output or electrical coolers);

Relay Cooling Values

If the **Cooling Relay Enable** slot is set to true, and the **Cooling Second Stage Enable** slot is set to false (cooling in the first stage only), the value from the **Binary Cooling** In slot is set to the **Relay Cooling** slot.

If the **Cooling Relay Enable** slot is set to true, and the **Cooling Second Stage Enable** slot is set to true (cooling in the first and second stage), the value from the **Second Stage Cooling In** slot is set to the **Relay Cooling** slot.

If the **Cooling Relay Enable** slot is set to false (the cooling relay is disabled), the **Relay Cooling** slot is set to false.

- **Heating Second Stage Enable:** allows to enable or disable the second stage heating;
  - Available settings: true (enabled,) false (disabled);
- **Cooling Second Stage Enable:** allows to enable or disable the second stage cooling;
  - Available settings: true (enabled,) false (disabled);
- **Heating Relay Enable:** allows to enable or disable the relay for heating;
  - Available settings: true (enabled,) false (disabled);
- **Cooling Relay Enable:** allows to enable or disable the relay for cooling;
  - Available settings: true (enabled,) false (disabled);
- **Mode:** allows to switch between the 2-pipe system and 4-pipe system;
  - Available settings: 2-pipe system, 4-pipe system;
- **Outputs Type:** allows to switch between the analog or binary control of the temperature outputs;
  - Available settings: Analog Outputs, Binary Outputs;
- **Analog Heating In:** sets the analog value for heating;
- **Analog Cooling In:** sets the analog value for cooling;
- **Binary Heating In:** sets the binary value for the first stage heating;
- **Binary Cooling In:** sets the binary value for the first stage cooling;
- **Second Stage Heating In:** sets the binary value for the second stage heating;
- **Second Stage Cooling In:** sets the binary value for the second stage cooling;
- **Fan Analog In:** sets the analog value for the fan;
- **Fan Type:** sets the type of fan;
  - Available settings: 0 (fan with analog output), other values (fan with binary outputs);
- **Fan Status:** sets the current fan speed;
  - Available settings: 0 (Off), 1 or 4 (Speed 1), 2 or 5 (Speed 2), 3 or 6 (Speed 3).

6.8.8 PWM

Applicable to library's version 1.0

The PWM component implements a pulse-width modulation (PWM) mechanism. The PWM is based on two slots: Period slot: the time period of the cycle, and Duty Cycle slot: the value of the pulse width, expressed as percentage. The Duty Cycle slot defines the

percentage of a time period (value from the Period slot), during which the Out slot is set to true. During the remaining period of time, Out slot will be set to "false".

For example, according to the values in the figure below:

- The Out slot will be set to true for 10 seconds: value from the Duty Cycle slot multiplied by the value from the Period slot (10% * 100s);
- After this, the Out slot will be set to false for 90 seconds (90% * 100s);
- The state of the Out slot is changed in the above way periodically (in periods defined in the Period slot).

Slots

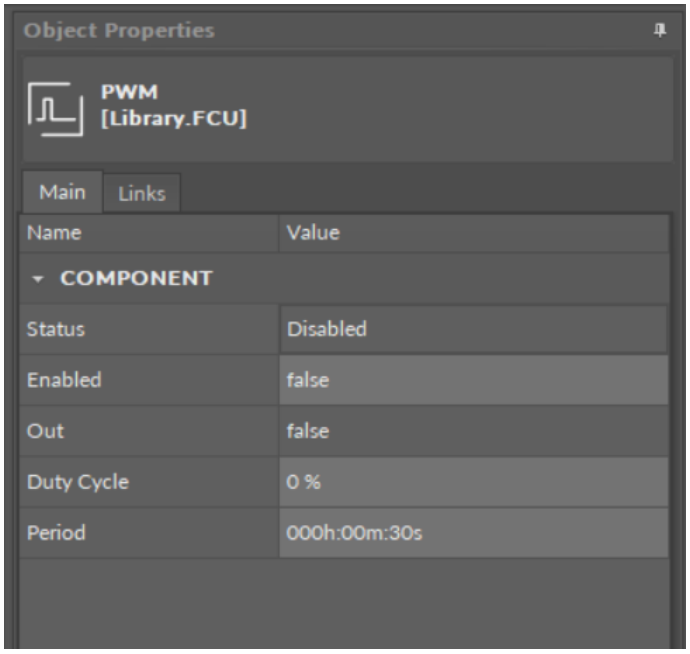


Figure 130. The PWM component slots

The PWM component has the following slots:

- **Status:** displays the status of the component;
- **Enabled:** enables/disables the component; true: enabled, false: disabled;
- **Out:** the binary output slot with the current state calculated by the component's algorithm;
- **Duty Cycle:** the numeric input slot corresponding to the pulse width;
- **Period:** the numeric input slot with the period of modulation.

6.8.9 TemperatureAnalog

Applicable to library's version 1.0

The TemperatureAnalog component allows to calculate numeric values for the Heating Analog Output and Cooling Analog Output slots, according to the external analog Control Value from range -100%-100% (for example, from the analog PID regulator, the Loop component). The component is dedicated to control valve actuators with analog inputs (or actuators, which can be controlled by the triac outputs with PWM). Negative values of the Control Value are used to calculate the Cooling Analog Output slot, and positive values are used to calculate the Heating Analog Output slot.

The TemperatureAnalog component can work in two temperature control modes:

- One-stage mode: only analog outputs (the Heating Analog Output and Cooling Analog Output slots) are calculated according to the Control Value;
- Two-stage mode: analog outputs are used for the first stage, and dedicated binary outputs (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output) are used for the second.

Note: Operation in two stages mode can be selected by setting the Heating Second Stage Enable and/or Cooling Second Stage Enable slots to true. If one of these slots is set to false, the component will operate in one stage mode for corresponding temperature mode (heating or cooling).

For proper operation, the component has to be enabled (the Temperature Analog Enable slot set to true), and the Fan Active slot has to be set to true. If the second condition is not met, the component is enabled, but the main outputs are blocked–values of the Heating Analog Output and Cooling Analog Output slots are set to 0, and the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots are set to false.

The values of the Heating Analog Output and Cooling Analog Output slots for one-stage mode control are calculated as shown in the figures below:

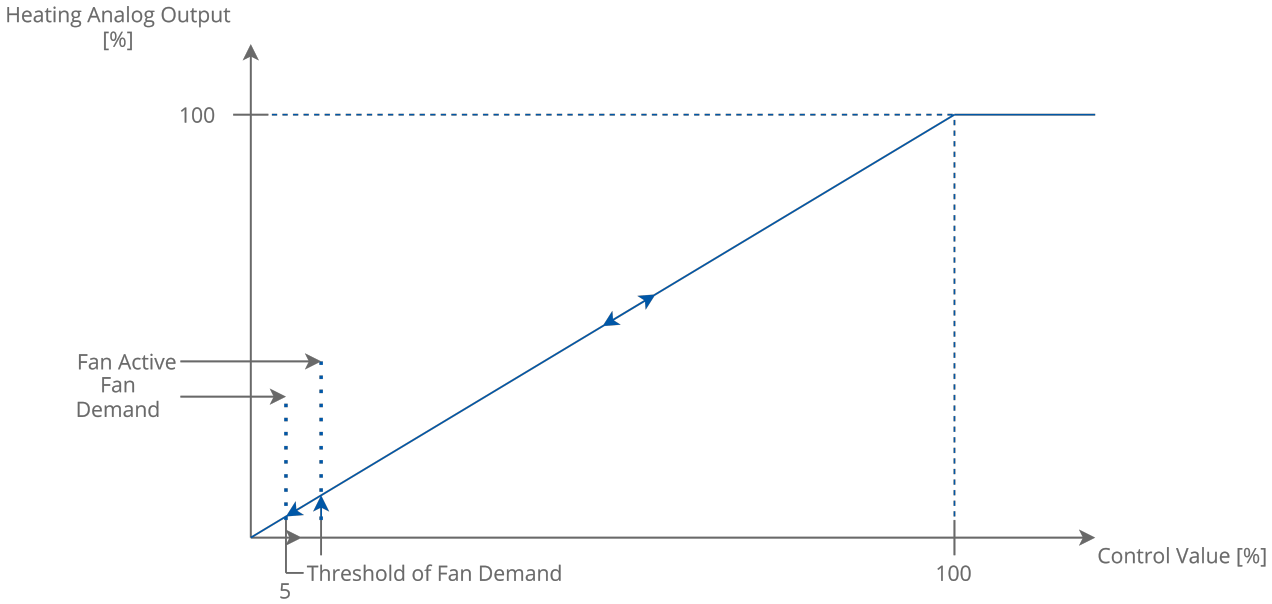


Figure 131. Analog control of the temperature for 1st stage only-Heating mode

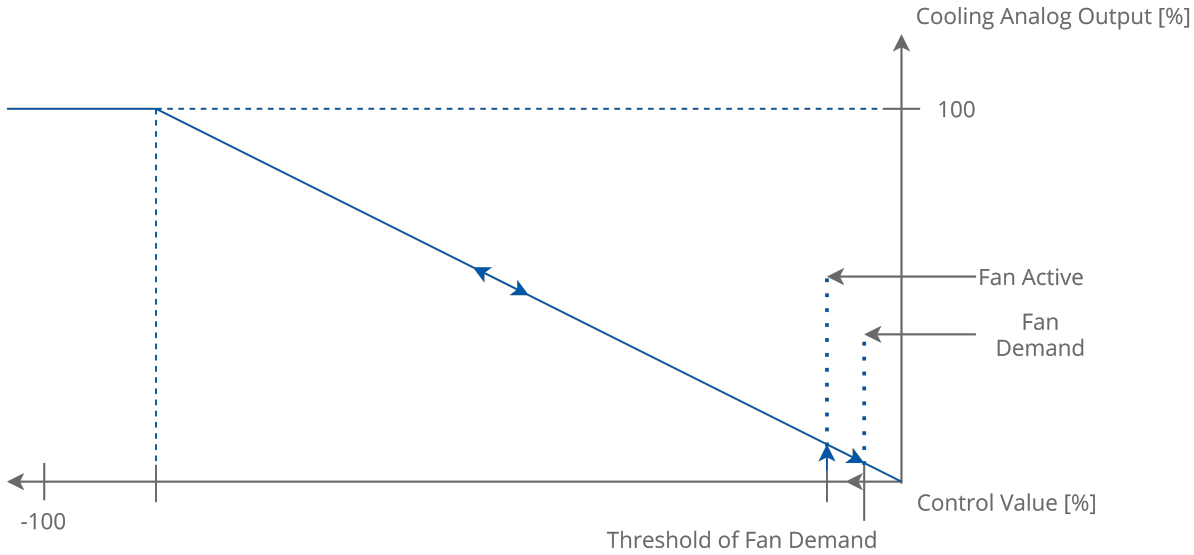


Figure 132. Analog control of the temperature for 1st stage only-Cooling mode

In this mode, the slots for the second stage are not used (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots are still set to false).

The values of the Heating Analog Output and Cooling Analog Output slots for the first stage, and the values of the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots (for the second stage) for two-stage mode are calculated as shown in the figures below:

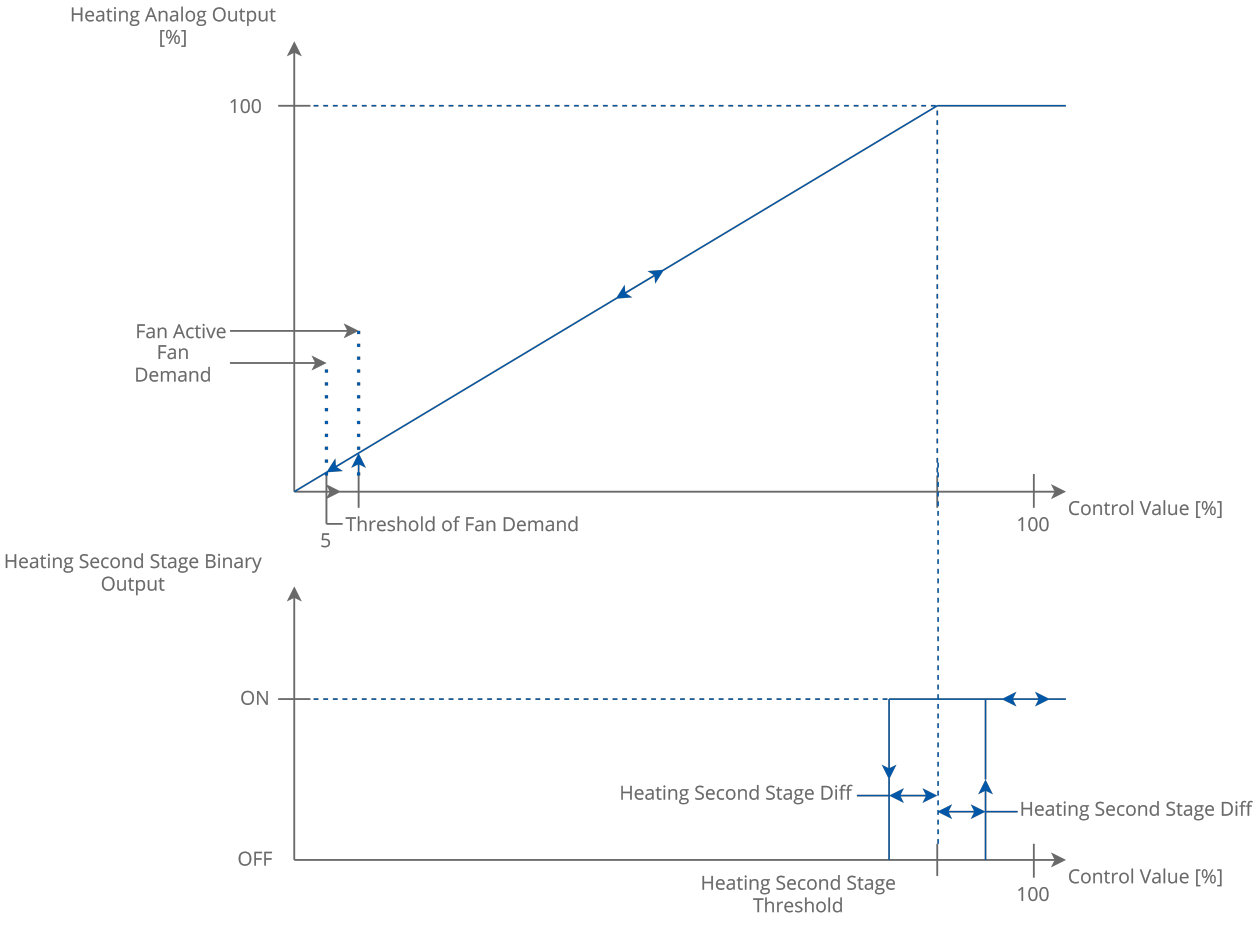


Figure 133. Analog control of the temperature for 1st and 2nd stage-Heating mode

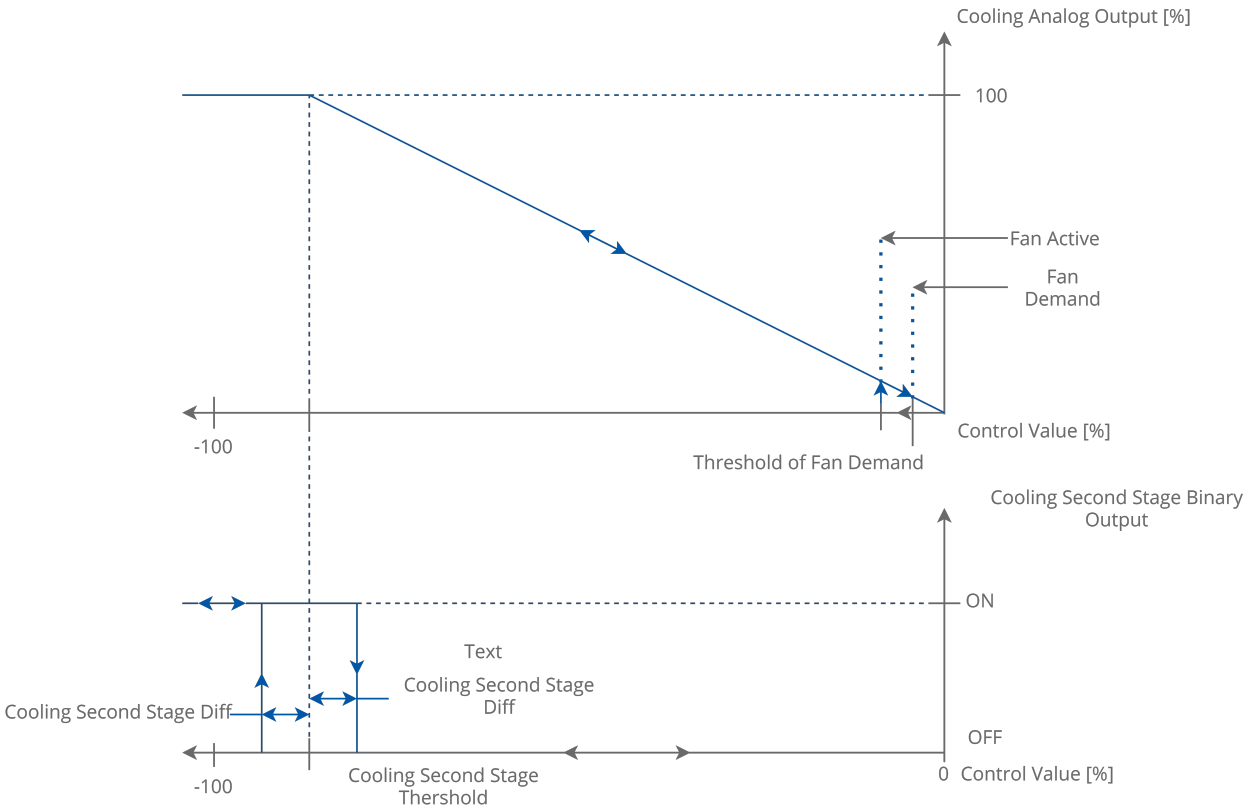


Figure 134. Analog control of the temperature for 1st and 2nd stage-Cooling mode

Slots

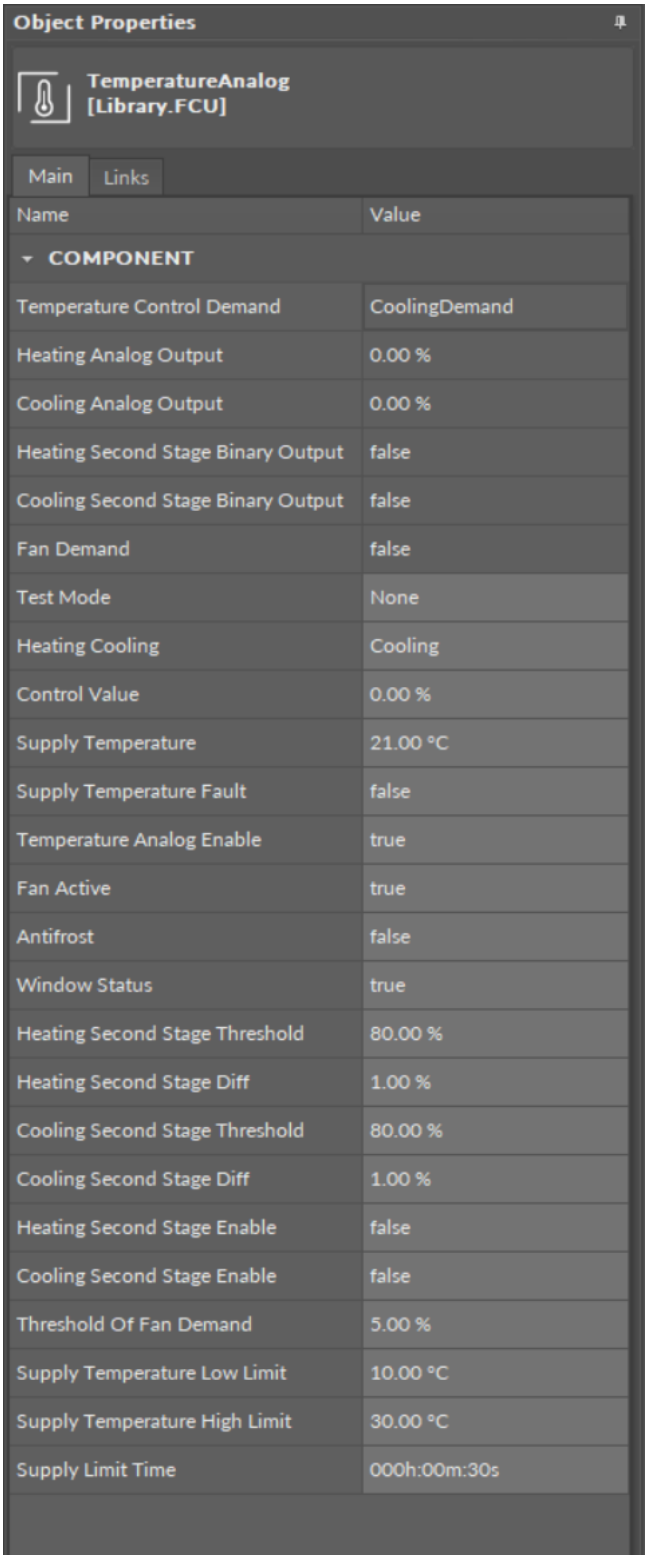


Figure 135. The TemperatureAnalog component slots

The TemperatureAnalog component has the following slots:

- **Temperature Control Demand:** shows the current component temperature demand;
  - Available information: HeatingDemand, CoolingDemand;
- **Heating Analog Output:** shows the heating demand level, expressed in percentage;

- **Cooling Analog Output:** shows the cooling demand level, expressed in percentage;
- **Heating Second Stage Binary Output:** shows the state of heating in the second stage;
- **Cooling Second Stage Binary Output:** shows the state of cooling in the second stage;
- **Fan Demand:** shows the fan demand;
    - Available information: true–if the absolute value from the Control Value slot exceeds the value set in the Fan Demand Threshold slot, false–in other cases;
- **Test Mode:** allows to set one of the predefined test modes;
    - Available settings: None, Full Heating, Full Cooling;

Full Heating Test Mode

In the Full Heating mode, the **Fan Demand** slot is set to true. The **Heating Analog Output** slot is set to 100%, and the **Heating Second Stage Binary Output** slot is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Full Cooling Test Mode

In the Full Cooling mode, the **Fan Demand** slot is set to true. The **Cooling Analog Output** slot is set to 100%, and the **Cooling Second Stage Binary Output** slot is set to true (only if the **Cooling Second Stage Enable** slot is set to true).

Note: Before starting the Full Heating or Full Cooling mode, the value of the **Fan Active** slot has to be set to true.

- **Heating Cooling:** sets the current temperature mode;
    - Available settings: heating, cooling;

Note: If the component operates in one of the above modes, outputs corresponding to the other mode are blocked. For example, if the component works in the Heating mode, the Cooling Analog Output is set to 0%, and the Cooling Second Stage Binary Output slot is set to false.

- **Control Value:** receives the value from an external component (for example, from the PID regulator, the Loop component); based on the received value, the values for output slots of the component are calculated; range: -100%-100%. Negative values of the Control Value are used for calculating the open level of the cooling valve and positive values for calculating the open level of the heating valve.
- **Supply Temperature:** allows to read the value of the supply temperature;
- **Supply Temperature Fault:** allows to read the information about the fault of the supply temperature;
- **Temperature Analog Enable:** allows to enable or disable the TemperatureAnalog component–if the component is disabled, analog outputs are set to 0, and binary outputs are set to false;
    - Available settings: true (enabled), false (disabled);
- **Fan Active:** informs the TemperatureAnalog component that the fan is switched on; if the slot is set to false, analog outputs (the Heating Analog Output and Cooling Analog Output slots) are set to 0, and binary outputs (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots) are set to false–these slots can be set to other values (calculated by the main algorithm) only when the Fan Active slot is set to true.
- **Antifrost:** allows to switch on the Antifrost mode;
    - Available settings: true (enabled), false (disabled);

Note: In the Antifrost mode enabled, the **Fan Demand** slot is set to true, the **Heating Analog Output** is set to 100%, and the **Heating Second Stage Binary Output** is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Note: The Antifrost mode has higher priority than the main algorithm, but it can be overridden by the Test mode.

- **Window Status:** allows to switch on the Window Open mode;
  - Available settings: true (Window Open mode disabled), false (Window Open mode enabled–the component operates in saving energy mode, analog outputs are set to 0, and binary outputs for the second stage are set to false);

Note: The Window Open mode can be overridden only by the Antifrost mode or the Test mode.

- **Heating Second Stage Threshold:** sets the threshold of the Control Value, above which (with the hysteresis) the heating in second stage is switched on;
- **Heating Second Stage Diff:** sets the differential for hysteresis of switching on/off the heating in second stage;
- **Cooling Second Stage Threshold:** sets the threshold of the Control Value, above which (with the hysteresis) the cooling in second stage is switched on;
- **Cooling Second Stage Diff:** sets the differential for hysteresis of switching on/off the cooling in second stage;
- **Heating Second Stage Enable:** allows to enable or disable the heating in second stage;
  - Available settings: true (enabled), false (disabled);
- **Cooling Second Stage Enable:** allows to enable or disable the cooling in second stage;
  - Available settings: true (enabled), false (disabled);
- **Threshold Of Fan Demand:** sets the threshold of the Control Value, above which the Fan Demand slot is set to true;
- **Supply Temperature Low Limit:** sets the minimum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function;
- **Supply Temperature High Limit:** sets the maximum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function;
- **Supply Limit Time:** sets the delay time for activation of the Supply Air Temperature Limitation function.

Supply Air Temperature Limitation

In order to maintain room conditions comfortable for the user, the supply air can have a temperature limitation. This function is available only if the supply air sensor is connected and works correctly. The supply air temperature can have a high limit defined by the Supply Temperature High Limit slot, and a low limit defined by the Supply Temperature Low Limit slot. The range between the Supply Temperature Low Limit and Supply Temperature High Limit values is called a comfort range.

- Supply Air Temperature limitation in the first stage analog control

In analog control, if the supply air temperature approaches the comfort range by 1°C, the TemperatureAnalog component starts the countdown set in the Supply Limit Time slot delay time . After this time, if the supply air temperature value is still approaching the comfort range limit by 1°C, the component starts a built-in algorithm, which reduces the air temperature (if the temperature value is close to or above the Supply Temperature High Limit), or increases the air temperature (if the temperature value is close to or below the Supply Temperature Low Limit). If the supply air temperature value returns to the comfort range ±1°C, the component will reset delay counter and return to normal operation.

- Supply Air Temperature limitation in the second stage analog control

In analog control, if the supply air temperature approaches the comfort range by 1°C, the TemperatureAnalog component disables the second stage, and starts counting the delay time set in the Supply Limit Time slot. After this time, if the supply air temperature value is still approaching the comfort range by 1°C, the component starts the built-in algorithm, which reduces the air temperature (if the temperature value is close to or above the Supply Temperature High Limit), or increases the air temperature (if the temperature value is close to or below the Supply Temperature Low Limit). If the supply air temperature value returns to the comfort range ±1°C, the component resets the delay counter, enables the second stage, and returns to normal operation.

6.8.10 TemperatureBinary

Applicable to library's version 1.0

The TemperatureBinary component allows calculating binary values for the Heating Binary Output and Cooling Binary Output slots, according to the difference between the Setpoint and Control Variable slots.

The TemperatureBinary component can work in two temperature control modes:

- One-stage mode: only the Heating Binary Output and Cooling Binary Output slots are switched on/off,
- Two-stage mode: the Heating Binary Output and Cooling Binary Output slots are used for the first stage, and dedicated binary outputs (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots) are used for the second stage.

Note: Operation in the two-stage mode can be selected by setting the Heating Second Stage Enable and/or Cooling Second Stage Enable slots to true. If one of these slots is set

to false, the component will operate in the one-stage mode for corresponding temperature mode (Heating or Cooling).

For proper operation, the component has to be enabled (the Temperature Binary Enable slot set to true), and the Fan Active slot has to be set to true. If the second condition is not met, the component is enabled, but main outputs are blocked–values of the Heating Binary Output, Cooling Binary Output, Heating Second Stage Binary Output, and Cooling Second Stage Binary Output slots are set to false.

The conditions for switching on/off heating binary output slots for one-stage mode (the Heating Binary Output and Cooling Binary Output slots) are presented in the figures below:

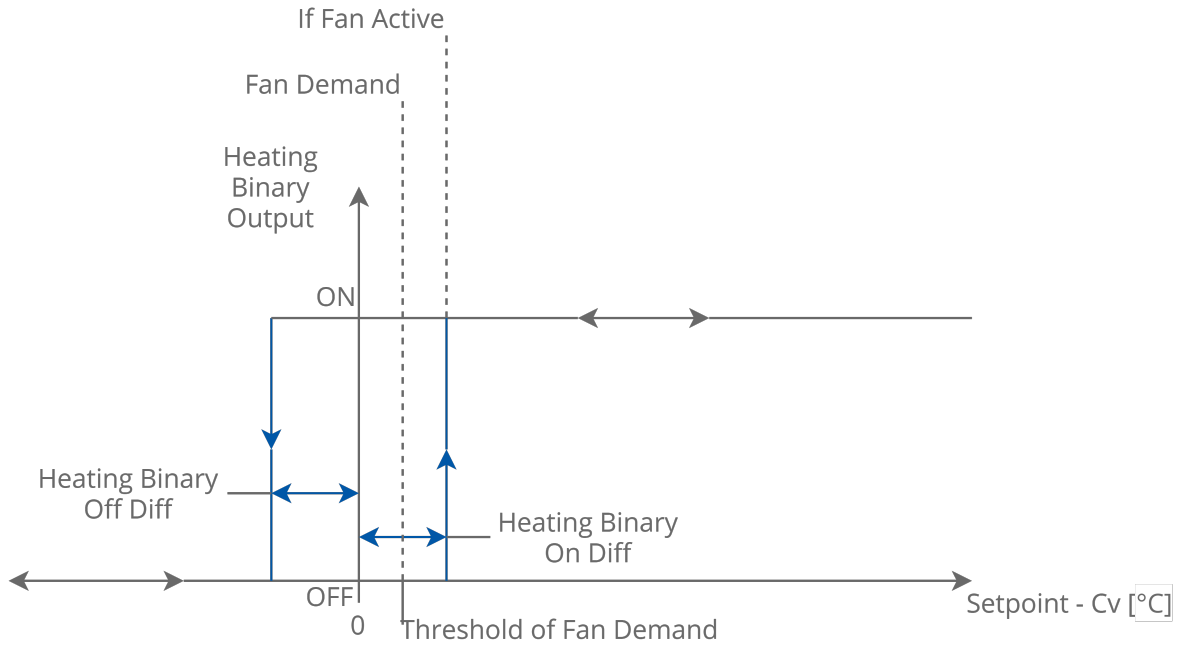


Figure 136. Binary control of the temperature for 1st stage only-Heating mode

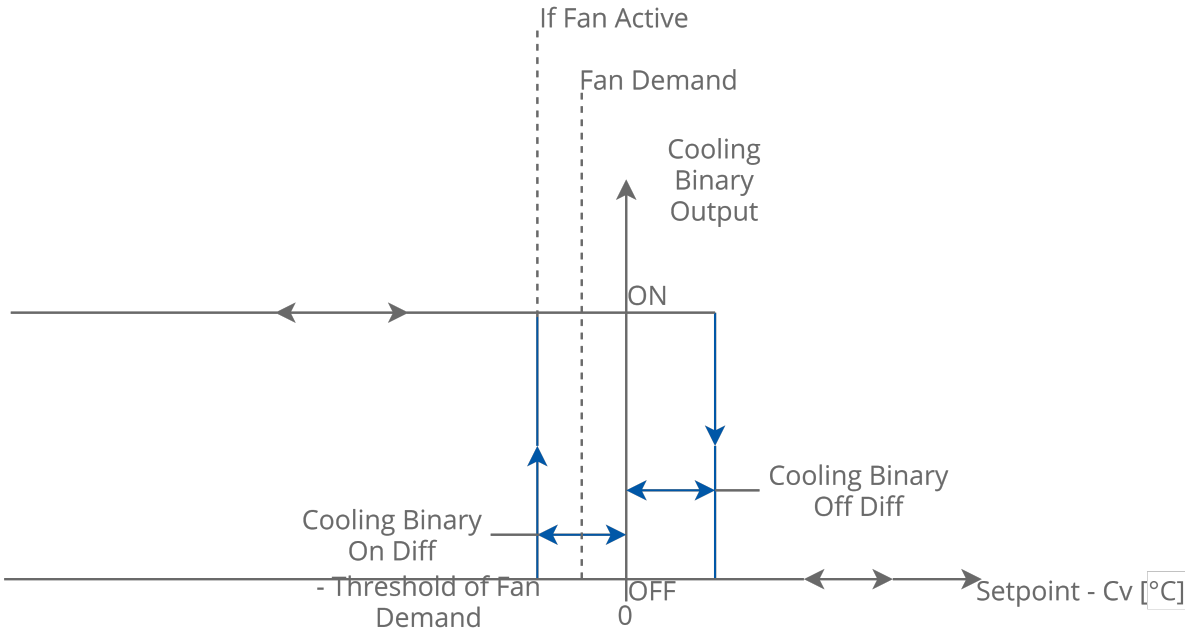


Figure 137. Binary control of the temperature for 1st stage only-Cooling mode

The values of the Heating Binary Output and Cooling Binary Output slots for the first stage, and the values of the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots (for the second stage) for the two-stage mode are calculated as shown in the figures below:

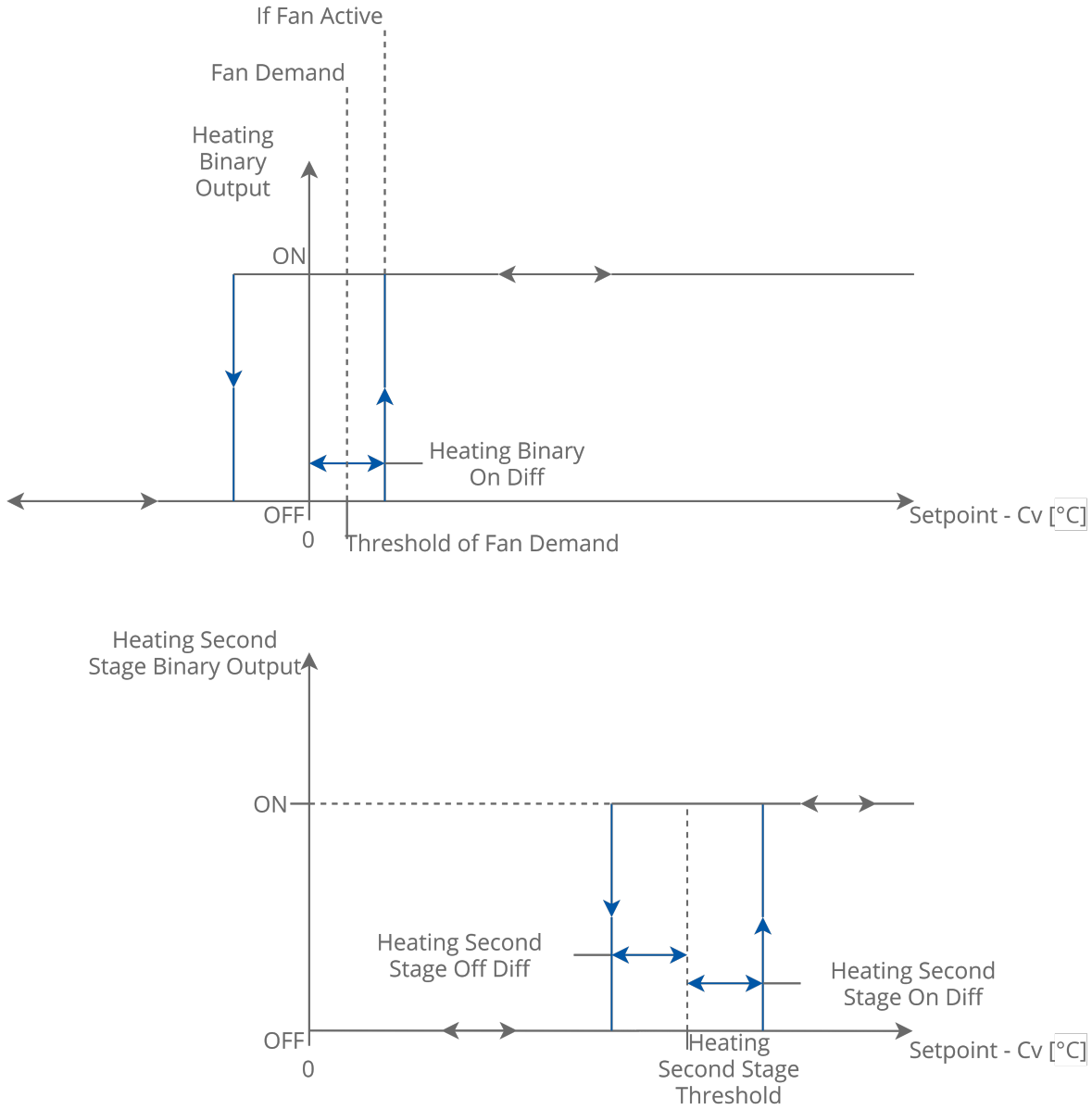


Figure 138. Binary control of the temperature for 1st and 2nd stage-Heating mode

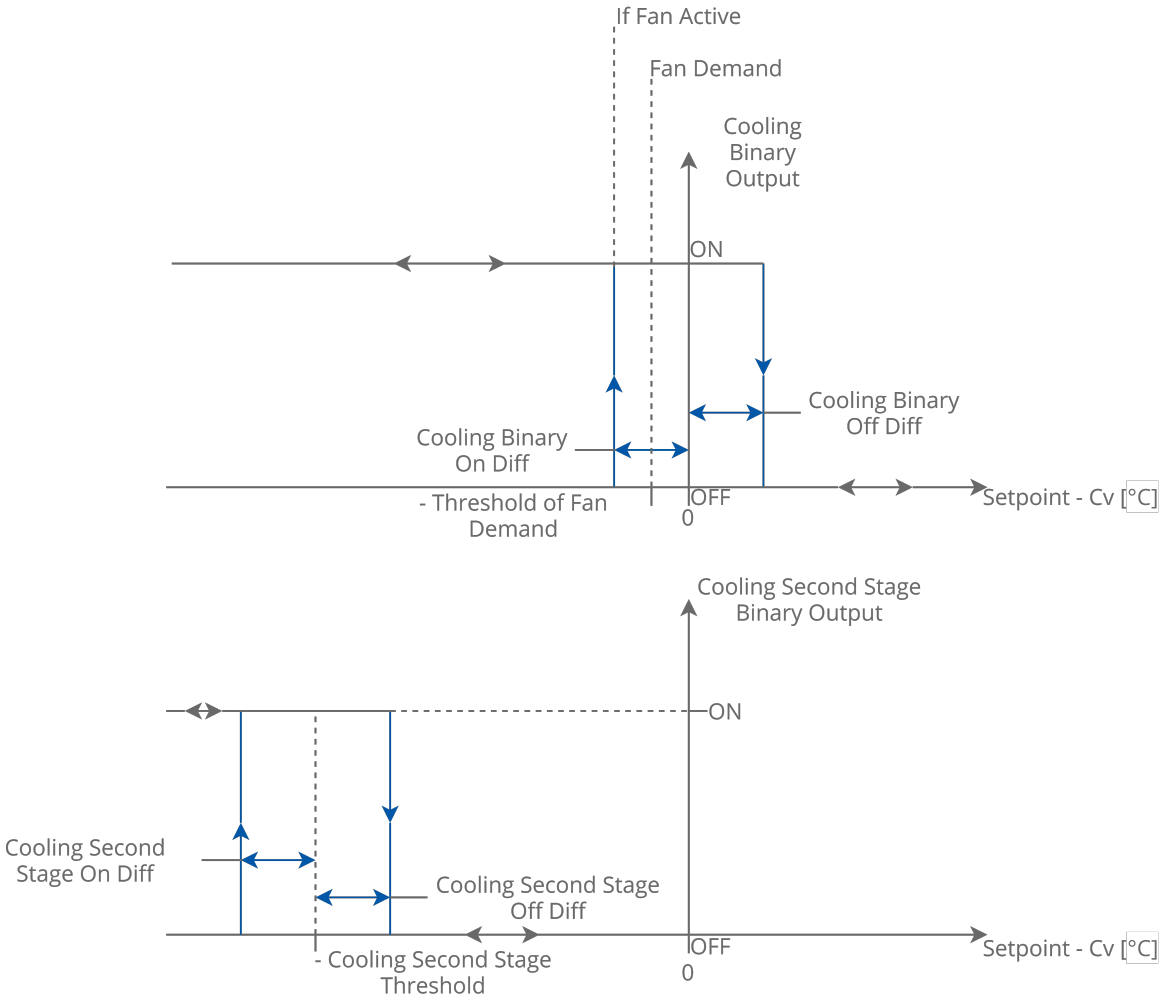


Figure 139. Binary control of the temperature for 1st stage and 2nd stage-Cooling mode

Slots

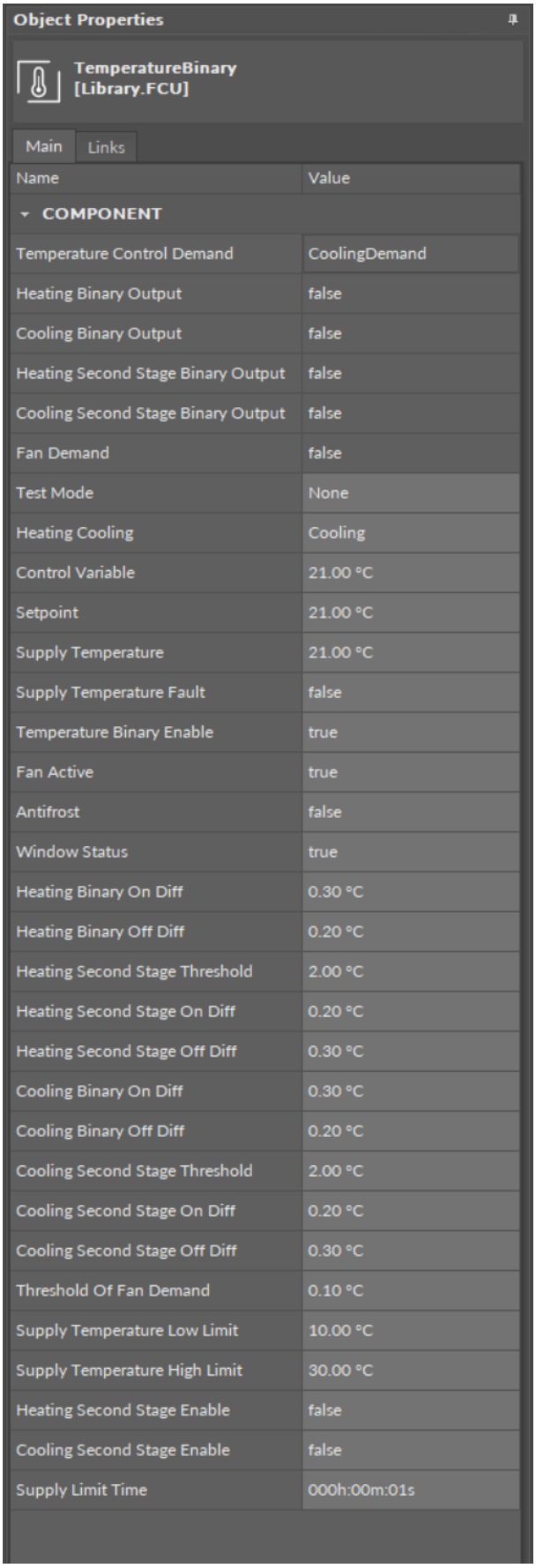

Object Properties
TemperatureBinary
[Library.FCU]
Main
Links
Name
Value
COMPONENT
Temperature Control Demand
CoolingDemand
Heating Binary Output
false
Cooling Binary Output
false
Heating Second Stage Binary Output
false
Cooling Second Stage Binary Output
false
Fan Demand
false
Test Mode
None
Heating Cooling
Cooling
Control Variable
21.00 °C
Setpoint
21.00 °C
Supply Temperature
21.00 °C
Supply Temperature Fault
false
Temperature Binary Enable
true
Fan Active
true
Antifrost
false
Window Status
true
Heating Binary On Diff
0.30 °C
Heating Binary Off Diff
0.20 °C
Heating Second Stage Threshold
2.00 °C
Heating Second Stage On Diff
0.20 °C
Heating Second Stage Off Diff
0.30 °C
Cooling Binary On Diff
0.30 °C
Cooling Binary Off Diff
0.20 °C
Cooling Second Stage Threshold
2.00 °C
Cooling Second Stage On Diff
0.20 °C
Cooling Second Stage Off Diff
0.30 °C
Threshold Of Fan Demand
0.10 °C
Supply Temperature Low Limit
10.00 °C
Supply Temperature High Limit
30.00 °C
Heating Second Stage Enable
false
Cooling Second Stage Enable
false
Supply Limit Time
000h:00m:01s

Figure 140. The TemperatureBinary component slots

The TemperatureBinary component has the following slots:

- **Temperature Control Demand:** shows the current component temperature demand;
  - Available information: HeatingDemand, CoolingDemand;
- **Heating Binary Output:** shows the state of the heating demand;
- **Cooling Binary Output:** shows the state of the cooling demand;
- **Heating Second Stage Binary Output:** shows the state of heating in the second stage;
- **Cooling Second Stage Binary Output:** shows the state of cooling in the second stage;
- **Fan Demand:** shows the fan demand.;
  - Available information: true–if the absolute value from the Control Variable slot exceeds the value set in the Fan Demand Threshold slot, the Fan Demand slot is set to true, false–in other cases;
- **Test Mode:** allows to set one of the predefined test modes;
  - Available settings: None, Full Heating, Full Cooling;

Full Heating Test Mode

In the Full Heating mode, the **Fan Demand** slot is set to true. The **Heating Binary Output** slot is set to true, and the **Heating Second Stage Binary Output** slot is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Full Cooling Test Mode

In the Full Cooling mode, the **Fan Demand** slot is set to true. The **Cooling Binary Output** slot is set to true, and the **Cooling Second Stage Binary Output** slot is set to true (only if the **Cooling Second Stage Enable** slot is set to true).

Note: Before starting the Full Heating or Full Cooling mode, the value of the **Fan Active** slot has to be set to true.

- **Heating Cooling:** sets the current temperature mode;
  - Available settings: heating, cooling;

Note: If the component operates in one of the above modes, outputs corresponding to the other mode are blocked. For example, if the component works in the Heating mode, the Cooling Binary Output and the Cooling Second Stage Binary Output slot are set to false.

- **Control Variable:** the current value of controlled temperature;
- **Setpoint:** sets the setpoint for the controlled temperature;
- **Supply Temperature:** allows to read the value of the supply temperature;
- **Supply Temperature Fault:** allows to read the information about the fault of the supply temperature;
- **Temperature Binary Enable:** allows to enable or disable the TemperatureBinary component–if the component is disabled, all outputs are set to false;
  - Available settings: true (enabled), false (disabled);
- **Fan Active:** informs the TemperatureBinary component that the fan is switched on; if fan is switched off, outputs for heating and cooling valves actuators (the Heating Binary Output and Cooling Binary Output slots) are set to false, and the binary output slots for heating and cooling in the second stage (the Heating Second Stage Binary Output

and Cooling Second Stage Binary Output slot) are set to false–these slots can be set to true only when the Fan Active slot is set to true;

- **Antifrost:** allows to switch on the Antifrost mode;
  - Available settings: true (enabled), false (disabled);

Note: In the Antifrost mode enabled, the **Fan Demand** slot is set to true, the **Heating Binary Output** is set to true, and the **Heating Second Stage Binary Output** is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Note: The Antifrost mode has higher priority than the main algorithm, but it can be overridden by the Test mode.

- **Window Status:** allows to switch on the Window Open mode;
  - Available settings: true (Window Open mode enabled), false (Window Open mode disabled–the component operates in saving energy mode, all outputs are set to false;

Note: The Window Open mode can be overridden only by the Antifrost mode or the Test mode.

- **Heating Binary On Diff:** sets the temperature (the difference between Setpoint and Control Variable) above which the Heating Binary Output is switched on;
- **Heating Binary Off Diff:** sets the temperature (the difference between Setpoint and Control Variable) above which the Heating Binary Output is switched off;
- **Heating Second Stage Threshold:** sets the threshold of the temperature (the difference between Setpoint and Control Variable) above which (with the hysteresis) the heating in the second stage is switched on;
- **Heating Second Stage On Diff:** sets the differential for hysteresis of switching on the heating in the second stage;
- **Heating Second Stage Off Diff:** sets the differential for hysteresis of switching off the heating in the second stage;
- **Cooling Binary On Diff:** sets the temperature (the difference between Setpoint and Control Variable) above which the Cooling Binary Output is switched on;
- **Cooling Binary Off Diff:** sets the temperature (the difference between Setpoint and Control Variable) above which the Cooling Binary Output is switched off;
- **Cooling Second Stage Threshold:** sets the threshold of the temperature (the difference between Setpoint and Control Variable) above which (with the hysteresis) the cooling in the second stage is switched on;
- **Cooling Second Stage On Diff:** sets the differential for hysteresis of switching on the cooling in the second stage;
- **Cooling Second Stage Off Diff:** sets the differential for hysteresis of switching off the cooling in the second stage;
- **Threshold Of Fan Demand:** sets the threshold of the temperature (the difference between Setpoint and Control Variable) above which the Fan Demand slot is set to true;
- **Supply Temperature Low Limit:** sets the minimum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function;
- **Supply Temperature High Limit:** sets the maximum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function,
- **Heating Second Stage Enable:** allows to enable or disable the heating in the second stage;
  - Available settings: true (enabled), false (disabled);

- Cooling Second Stage Enable: allows to enable or disable the cooling in the stage;
    - Available settings: true (enabled), false (disabled);
- Supply Limit Time: sets the delay time for the activation of the Supply Air Temperature Limitation function.

Supply Air Temperature Limitation

In order to maintain room conditions comfortable for the user, the supply air can have a temperature limitation. This function is available only if the supply air sensor is connected and works correctly. The supply air temperature can have a high limit defined by the Supply Temperature High Limit slot, and a low limit defined by the Supply Temperature Low Limit slot. The range between the Supply Temperature Low Limit and Supply Temperature High Limit values is called a comfort range.

- Supply Air Temperature limitation in the first stage binary control

In binary control, if the supply air temperature approaches the comfort range by 1°C, the TemperatureBinary component starts the countdown set in the Supply Limit Time slot delay time . After this time, if the supply air temperature value is still out of the comfort range, the component disables the heating (if the temperature value is above the Supply Temperature High Limit) or cooling (if temperature value is above the Supply Temperature Low Limit). If the supply air temperature value returns to the comfort range, the component resets the delay counter and returns to normal operation.

- Supply Air Temperature limitation in the second stage binary control

If the Supply Air Temperature value is out of the comfort range, the component disables the second stage, and starts counting the delay time set in the Supply Limit Time slot. After this time, if the supply air temperature value is still out of the comfort range, the TemperatureBinary component disables the heating (if the temperature value is above the Supply Temperature High Limit) or cooling (if the temperature value is above the Supply Temperature Low Limit). If the supply air temperature value returns to the comfort range, the component resets the delay counter, enables the second stage, and returns to normal operation.

6.8.11 TemperatureSensorsSwitch

Applicable to library's version 1.0

The TemperatureSensorsSwitch component allows for switching temperature sensors, according to the selected source.

Slots

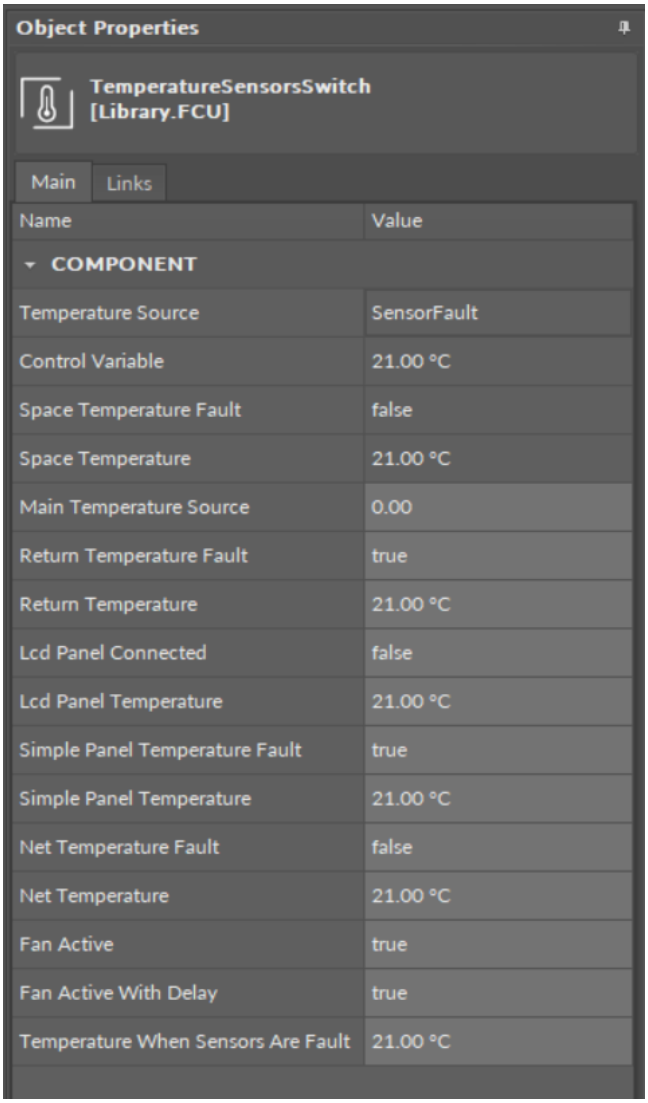


Figure 141. The TemperatureSensorsSwitch component slots

The TemperatureSensorsSwitch component has the following slots:

- **Temperature Source:** informs about the source of the temperature set to the Control Variable slot;
  - Available information: Sensor Fault (fault status of the sensor selected by the Main Temperature Source slot), LCD Panel, Simple Panel, Return temperature, Net temperature;
- **Control Variable:** the current input value (measured temperature), switched according to the value of the Main Temperature slot;

The Control Variable slot dependencies:

If the sensor selected in the Main Temperature Source slot has not gone into any fault or false state (slots corresponding to such fault–Return Temperature Fault, Lcd Panel Connected, Simple Panel Temperature Fault, Net Temperature Fault–are set to false), the temperature from the source selected in the Main Temperature Source is set to the Control Variable slot. If the Main Temperature Source slot is set to LCD Panel, and the panel is connected (the Lcd Panel Connected slot is set to true), the value from the Lcd Panel Temperature slot is set to the Control Variable slot. If the LCD panel is not connected (the Lcd Panel Connected slot is set to false), a value from the Temperature When Sensors Are Fault slot is set to the Control Variable slot.

If the sensor selected in the Main Temperature Source slot has gone into any fault or false state , the temperature from the Temperature When Sensors Are Fault slot is set to the Control Variable slot.

If the value from the Return Temperature slot is set to the Control Variable slot, it is possible to switch the temperature from Return to Space. The Space Temperature will be downloaded when the fan is off. There could also be a delay for switching to Return Temperature after the fan starts to blow the duct. The Fan Active and Fan Active With Delay slots are used for this purpose. If one or both slots have false states, then the value from the Space Temperature slot is set to the Control Variable slot (instead of the value from the Return Temperature slot). If both slots have the same true states, then the value from the Return Temperature slot is set to the Control Variable slot.

Note: For the proper operation of this Return to Space switching function, an external component is required, which delays the value informing about the fan status. The value without delay has to be connected to the Fan Active slot, and the delayed value has to be connected to the Fan Active With Delay slot.

Note: If the Return Temperature Fault slot is set to true (sensor has gone into fault), it transfers the value from the Temperature When Sensors Are Fault slot (instead of the Return Temperature) to the Control Variable slot. If the Space Temperature Fault slot is set to true (sensor has gone into fault), the function is inactive.

- Space Temperature Fault: indicates the status of the space temperature;
  - Available information: true (fault), false (no fault).
- Space Temperature: shows the output value of the space temperature;

The Space Temperature slot dependencies:

The Lcd Panel Temperature has the highest priority; if the panel is connected (the Lcd Panel Connected slot is set to true), the value from the Lcd Panel Temperature slot is transferred to the Space Temperature slot.

If the LCD panel is disconnected (the Lcd Panel Connected slot is set to false), and there is no fault of the simple panel temperature sensor (the Simple Panel Temperature Fault is set to false), the value from the Simple Panel Temperature slot is transferred to the Space Temperature slot.

If the LCD panel is disconnected, the Simple Panel sensor is in fault, and there is no fault of the Net Temperature, the value from the Net Temperature slot is transferred to the Space Temperature slot.

In case none of the conditions mentioned above is met, and there is no fault of the temperature sensor which is used to calculate the Control Variable value, the value from the Control Variable slot is set to the Space Temperature slot.

Note: In all cases described above, the Space Temperature Fault slot is set to false (there is no fault). In any other case, this slot is set to true.

- Main Temperature Source: allows to select the source of temperature; available values:
  - Available settings: 0 (LCD Panel), 1 (Simple Panel), 2 (return temperature), 3 (net temperature).
- Return Temperature Fault: allows to read the status of the return temperature sensor;
  - Available settings: true (sensor fault), false (sensor's operation is correct);
- Return Temperature: allows to read the temperature from the return temperature sensor;
- Lcd Panel Connected: allows to read the status of the LCD panel;
  - Available settings: true (panel connected), false (panel disconnected);
- Lcd Panel Temperature: allows to read the temperature from the LCD panel temperature sensor;
- Simple Panel Temperature Fault: allows to read the status of the Simple Panel temperature sensor;
  - Available settings: true (sensor fault), false (sensor's operation is correct);
- Simple Panel Temperature: allows to read the temperature from the Simple Panel temperature sensor;
- Net Temperature Fault: allows to read the status of the net temperature sensor;
  - Available settings: true (sensor fault), false (sensor's operation is correct);
- Net Temperature: allows to read the net temperature value;
- Fan Active: allows to read the status of the fan;
  - Available settings: true (fan switched on), false (fan switched off);
- Fan Active With Delay: allows to read the status of the fan (with external delay);
  - Available settings: true (fan switched on), false (fan switched off);
- Temperature When Sensors Are Fault: the temperature, which is to be set to the Control Variable slot if the sensor selected in the MainTemperature Source is faulty.

6.8.12 WindowStatusSwitch

Applicable to library's version 1.0

The WindowStatusSwitch component calculates the window status (open or closed), according to the information from up to 6 Window Status slots of one master and five slave devices. If the Master Window Status slot is set to true, and the Window Statuses of active slave devices are set to true, the Window Status Out slot is set to true (window closed). In other cases, the Window Status Out slot is set to false (window open).

Note: The slave device is considered inactive if its Slave Status slot is set to false. The window status from the inactive device is not used in the Window Status Out calculation.

Slots

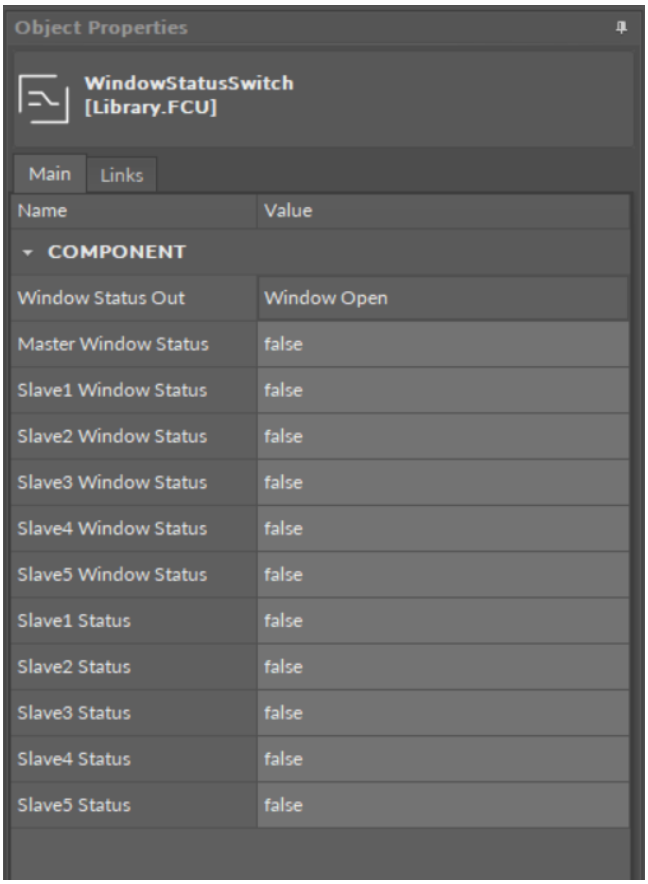


Figure 142. The WindowsStatusSwitch component slots

The WindowStatusSwitch component has the following slots:

- Window Status Out: informs about the current of window status (open or closed);
  - Available information: Window Closed, Window Open;
- Master Window Status: allows to read the window status from the master device;
  - Available settings: true (closed), false (open);
- Slave1 Window Status-Slave5 Window Status: allow to read the window statuses from slave devices.
  - Available settings: true (closed), false (open);
- Slave1 Status-Slave5 Status: allow to read the statuses of slave devices.
  - Available settings: true (slave device active), false (slave device inactive).

6.8.13 ValueHolder

Applicable to library's version 1.0

The ValueHolder component holds the value in the Out slot for a specified time. If the Input value changes, for example, from X to Y, the X value is shown in the Output slot and is upheld there for a specified time (Holding Time). After the Holding Time elapses, the Output slot is changed to the current value of the Input slot (Y, in this example). If, before the Holding Time expires, the Input value changes, these new value does not affect the value in the Output slot, which is maintained until the Holding Time ends. Only after the time elapses, the component writes the current value from the Input slot to the Output slot. This value is shown in the Output slot until the next change in the Input slot. If the Holding Time value is changed to a new one while the holding time is running (the value in the Output slot is being withheld), the newly set time will cause the elapsed Holding Time to be counted again.

Holding Time	3 s																2 s			4 s				
Time [s]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Input	0	1	2	3	4	5	6	7	8	8	8	9	9	10	10	11	11	12	12	13	14	15	16	17	18
Ouput	0	0	0	3	3	3	6	6	6	8	8	8	9	9	9	11	11	11	11	13	13	13	13	13	18
	Value of the Output slot stable for 3 s (the Holding Time)																	The Holding Time changed to 2 s (counting restarted in a moment of the change)			The Holding Time changed to 4 s (counting restarted in a moment of the change)				

Figure 143. The ValueHolder component mechanism

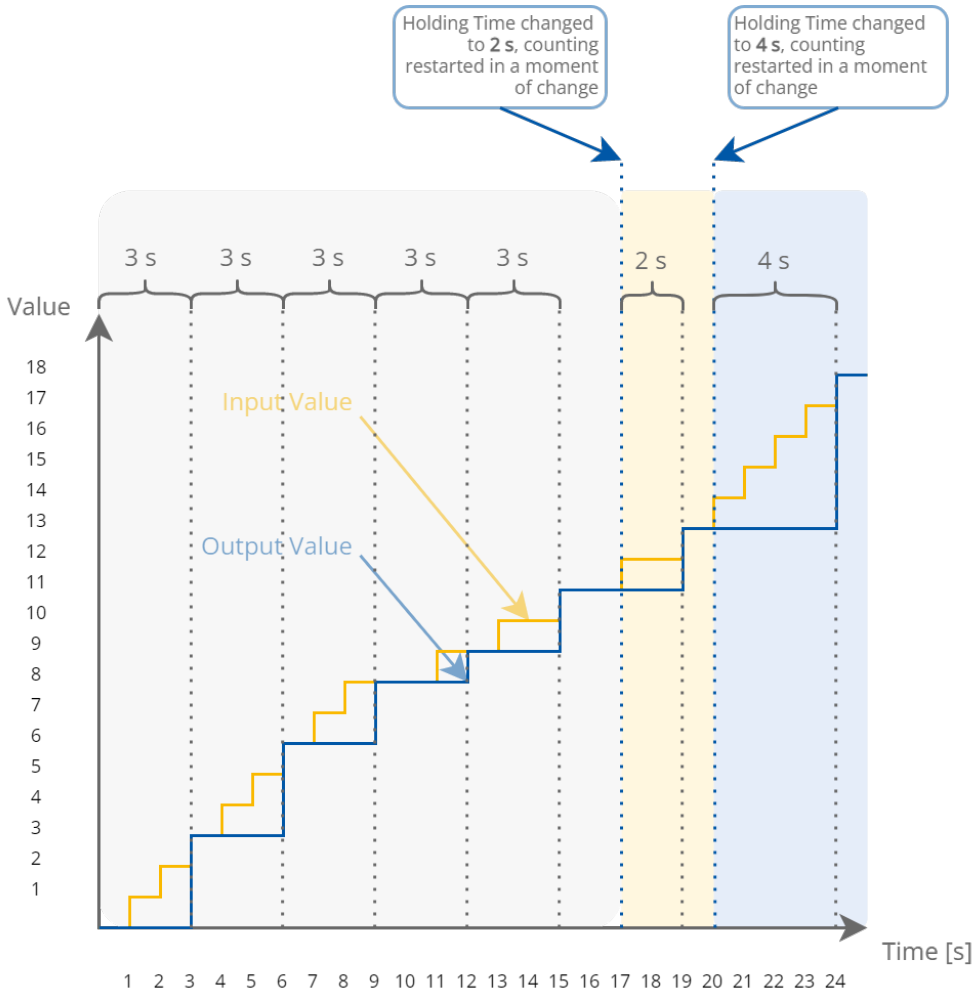


Figure 144. Chart of the ValueHolder component mechanism

Slots

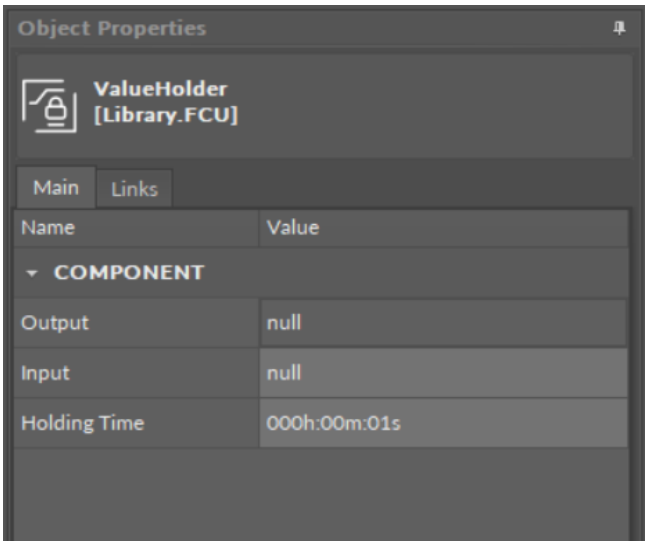


Figure 145. The ValueHolder component slots

The ValueHolder component has the following slots:

- **Output:** the output slot–value transferred from the Input slot;
- **Input:** the input slot;

- Holding Time: time, for which the value is held in the Output slot after a change in the In slot.

6.9 Other

Applicable to library's version 1.0

The Other library is designed for components that can facilitate operating of an application or that fulfill the roles, which are not met by any other thematic libraries.

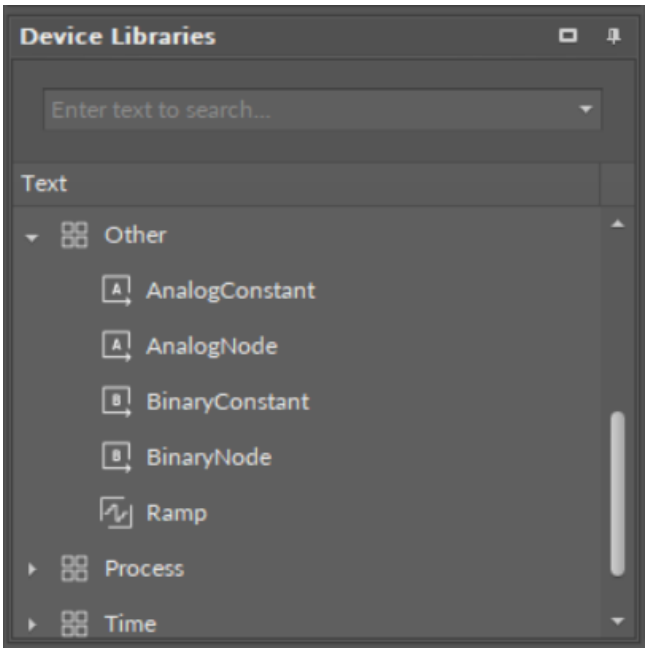


Figure 146. The Other library

6.9.1 AnalogConstant

Applicable to library's version 1.0

The AnalogConstant component is a simple component producing a constant analog value, which can be transferred to further components. The value in the AnalogConstant component can only be given by a Set action–it cannot be transferred to this component via link or entered manually in the slot.

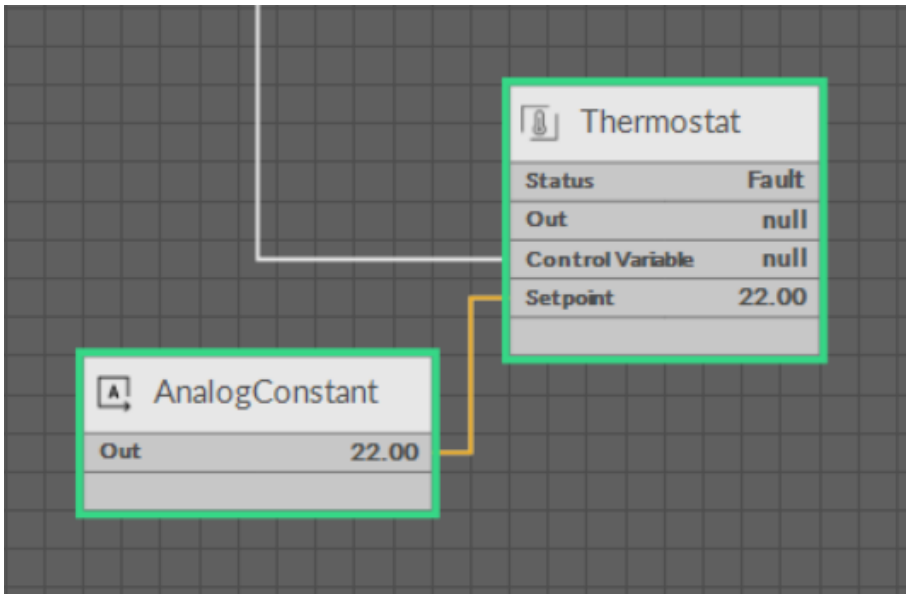


Figure 147. The AnalogConstant component

Slot

The AnalogConstant component has only one slot:

- **Out:** the output value set with the Set action.

Action

The AnalogConstant component has one action:

- **Set:** sets a constant analog value transferred to the Out slot.

6.9.2 AnalogNode

Applicable to library's version 1.0

The AnalogNode component is a simple analog node component, which transfers value received in the Value slot (either set manually or transferred via a link). The AnalogNode component transfers only values without any information about the component's status, therefore, the Reference linking is not possible in this case.

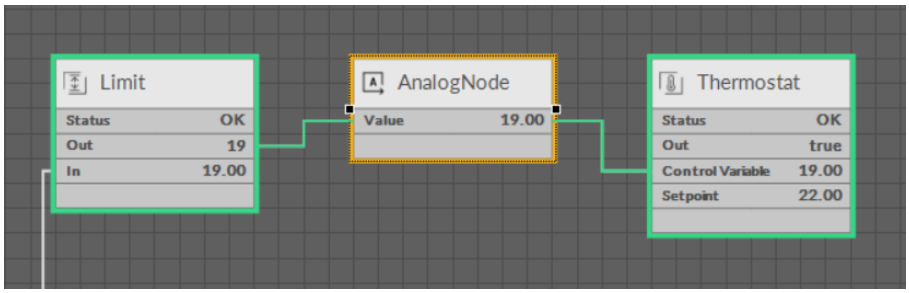


Figure 148. The AnalogNode component

Slot

The AnalogNode component has only one slot:

- **Value:** the value transferred via this analog node, either set manually, or given by a link.

6.9.3 BinaryConstant

Applicable to library's version 1.0

The BinaryConstant component is a simple component producing a constant binary value, which can be transferred to further components. The value in the BinaryConstant component can only be given by a Set action–it cannot be transferred to this component via link or entered manually in the slot.

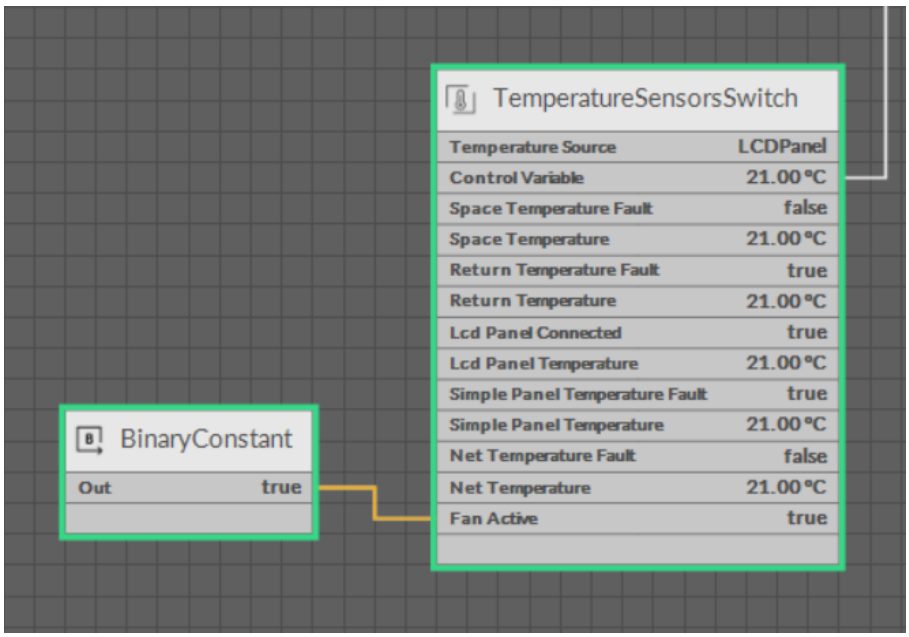


Figure 149. The BinaryConstant component

Slot

The BinaryConstant component has only one slot:

- **Out:** the output value set with the Set action.

Action

The BinaryConstant component has one action:

- **Set:** sets a constant binary value transferred to the Out slot.

6.9.4 BinaryNode

Applicable to library's version 1.0

The BinaryNode component is a simple binary node component, which transfers value given to the Value slot (either set manually or transferred via a link). The BinaryNode component transfers only values without any information about the component's status, therefore, the Reference linking is not possible in this case.

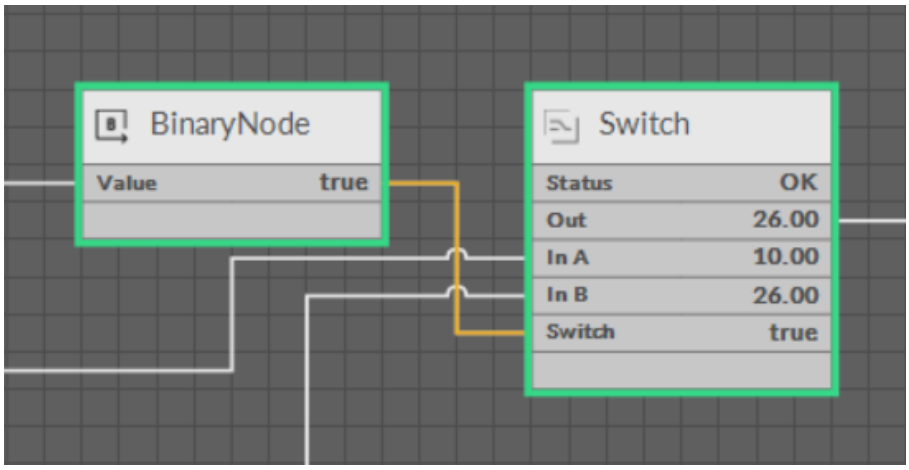


Figure 150. The BinaryNode component

Slot

The BinaryNode component has only one slot:

- **Value:** the value transferred via this binary node, either set manually, or given by a link.

6.9.5 Ramp

Applicable to library's version 1.0

The Ramp component is designed for testing applications. It creates a sinusoidal signal with set amplitude and period. It provides a numeric output value with a linear ramping out. The Out value continually changes, so the Ramp components is useful for simulation of changing conditions in application.

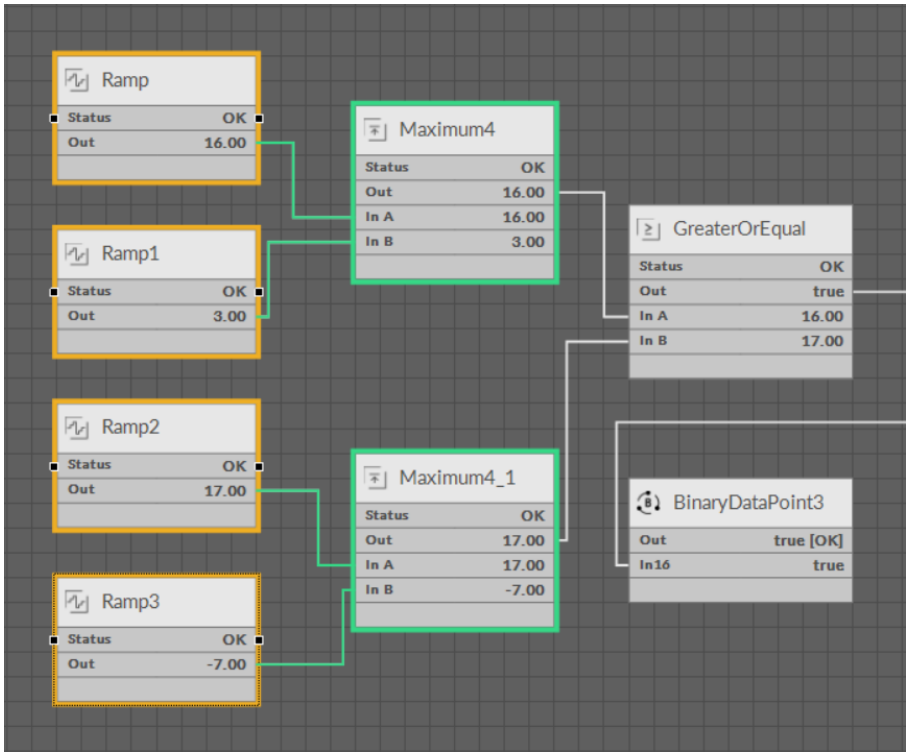


Figure 151. The Ramp component

Slots

The Ramp component has the following slots:

- **Status:** indicates the current status of the component (OK, Fault);
- **Enabled:** the change of the slot's value enables or disables the component—if enabled, the component calculates the Out slot value according to its algorithm; if disabled, the component latches the last value in the Out slot;
    - Available settings: enabled, disabled;
- **Out:** the output value calculated based on the set parameters;
- **Period:** allows to set the length of one cycle of the component's function;
- **Amplitude:** allows to set an amplitude for the output values function;
- **Offset:** allows to set an offset for the output values function;
- **Waveform:** allows to select a shape of the curve for the output values function (Triangle, SawTooth, InvertedSawTooth).

6.10 Quick Start-up of Applications

The RAC18-IP controller offers smart and easy way of creating applications.

The application created with the nano EDGE ENGINE device refers to containers:

- Applications;
- Networks;
- Services.

The Applications container is where the main logic is contained; the Networks container is where the external communication is configured, and the Services container will be the place to connect with additional functionalities.

In order to create a fully functioning application follow these steps:

Step 1: Make sure that the device is correctly connected and set up. Detailed instructions are available in the device's RAC18-IP Quick Start-up (Step1-7).

Step 2: Go to the Networks container to configure the external communication settings (Modbus or BACnet). Configure all the network-specific parameters there in order to enable proper communication in Modbus RTU as a master device, or in BACnet IP as a client or server device.

Step 3: Remain in the Network container. Go to the LocalIO component, and configure all input and output components that are going to be utilized in the main application logic. The input and output components are grouped in the IO library, available in the Device Libraries window. The first and foremost is to set their addresses (the Address slot in each input or output component)–unless the addresses are set properly, the components are in fault statuses.

Good Practice Tip

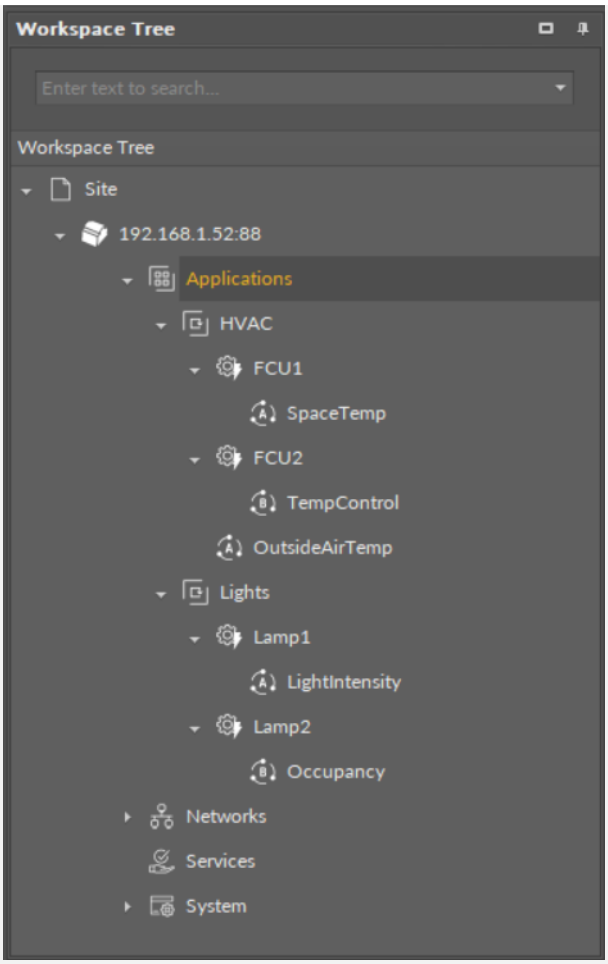
In future developments, the RAC18-IP controller will have a Haystack functionality implemented in its firmware, which calls for some good practice mechanisms that can be introduced now, with the current functionalities.

The Haystack functionality will offer multiple advantages of using tags–it will help identify parts of equipment controlled by the application and used Data Points, filter data by equipment, sensor, or value, any many more.

In order to facilitate a future use of the full Haystack functionality, the Equipment component has been included in the Core library. It is therefore recommended to use the following structure when creating applications:

- Applications container
  - Application component
    - Equipment component
      - Data Point(s)
      - other components
    - Equipment component
      - Data Point(s)
      - other components

Such structure will be fully recognized in the Haystack functionality and will require minimum effort to use its full potential once updated.



Difference Between Equipment and Folder Components

The Equipment component with Haystack functionality will allow to identify the types and other characteristics of controlled devices by tags, which will be a main difference between the Equipment and Folder components–the Folder component is merely an organizing component; in the future, it will not carry any Haystack tags functionalities. It is therefore very important to use the Equipment components in the applications structure.

Also refer to: Equipment Manager

Step 4: Go to the Applications container. This is the place to add Application components from the Core library–in fact, the nano EDGE ENGINE allows to add as many Application components to the container as necessary. Each application created this way is independent and cycle-driven. The Applications Manager is a view designed to manage Application components.

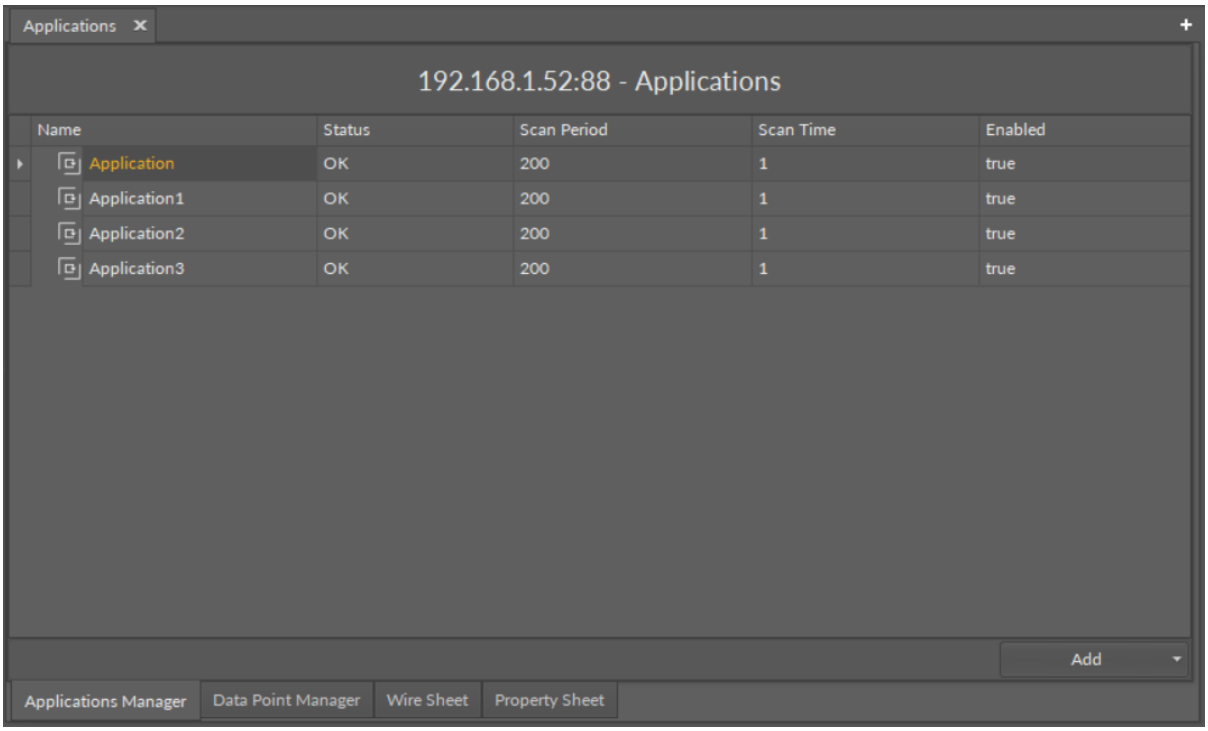


Figure 152. The Applications Manager

Step 5: **(Recommended)** Once all Application components are added to the Applications container, go to the Core library and add Equipment components for each item controlled by the application. Then add Data Points to the Equipment components. It is advised to rename components to fit the application characteristics.

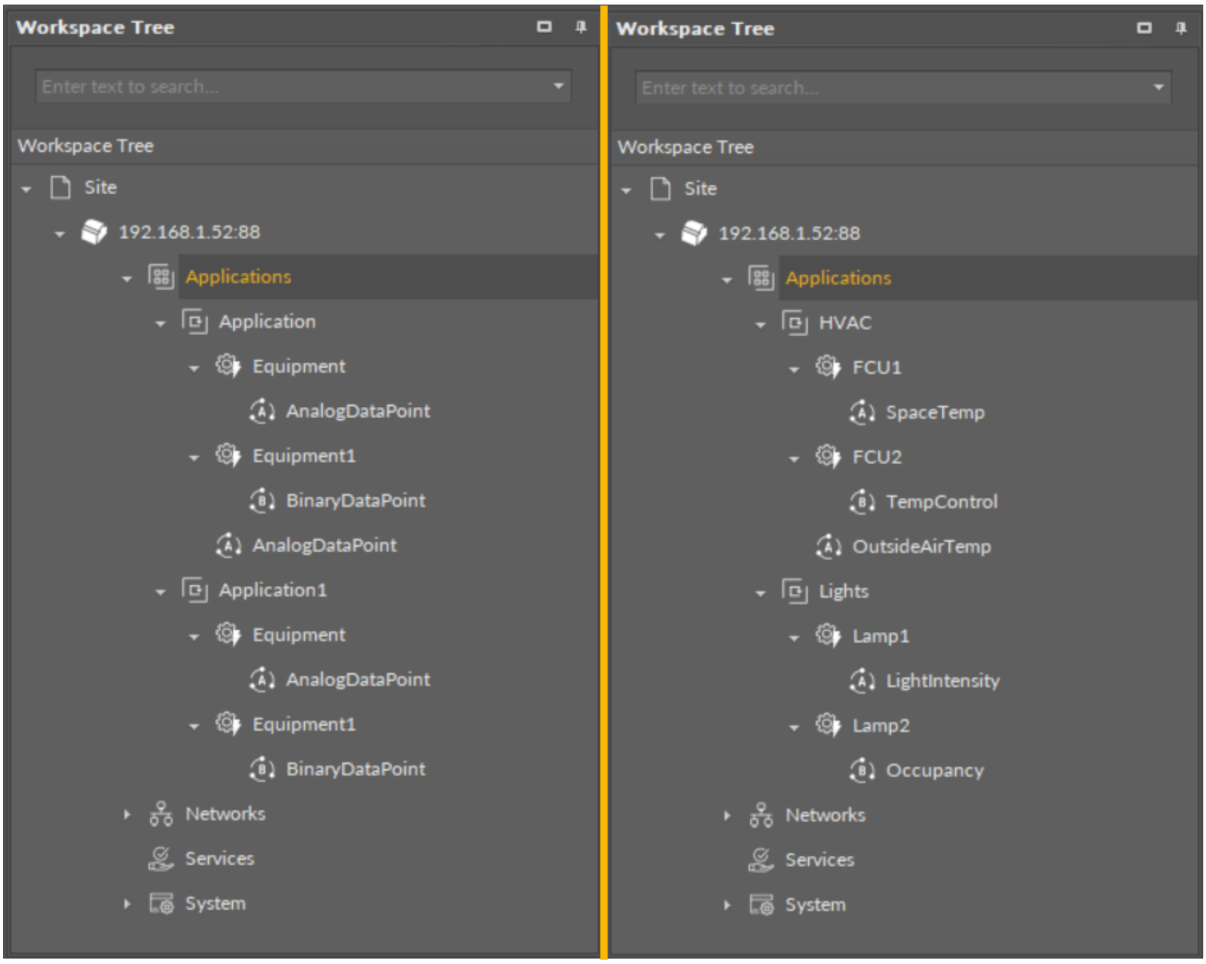


Figure 153. The recommended structure for applications

Step 6: (Recommended) Start adding other components to the Equipment component (or components) to create logics with added Data Points. All components available for creating applications are grouped in libraries in the Device Libraries window.

Tip

Data Points are universal components that represent a value in the application logic; they may serve as setpoints, sensors values, non-volatile variables, or any other data values. Data Points represent a layer of the application logic that is presented to an end user—this is where the end user is able to adjust desired setpoints (e.g., for air conditioning) or evoke other actions outlined in the application logic. Data Points also read values calculated in applications and control local or remote outputs.

Data Points in the application logic may work as regular writable variables with priorities, or—with Reference linking—they may be connected with network points, such as local IO components.

- Analog Data Point;
- Binary Data Point.

Step 7: Added component is ready to be linked. The nano EDGE ENGINE allows two methods of linking, standard and Reference linking. The Reference linking method is especially recommended to use to connect Data Points and network point class

components. Detailed information is available in the Linking section. Detailed technical information about linking in the iC Tool is available in the Linking Components section.

Tip

Here is the list of other quick start-ups that offer detailed instructions on how to configure external communication, network points, explain linking methods, etc.:

- Device quick start-up;
- BACnet quick start-up;
- Modbus quick start-up;
- LocalIO quick start-up;
- nano EDGE ENGINE linking methods.

6.10.1 Applications Manager

The Applications Manager is a special view that allows to manage the Application components added to the Applications container.

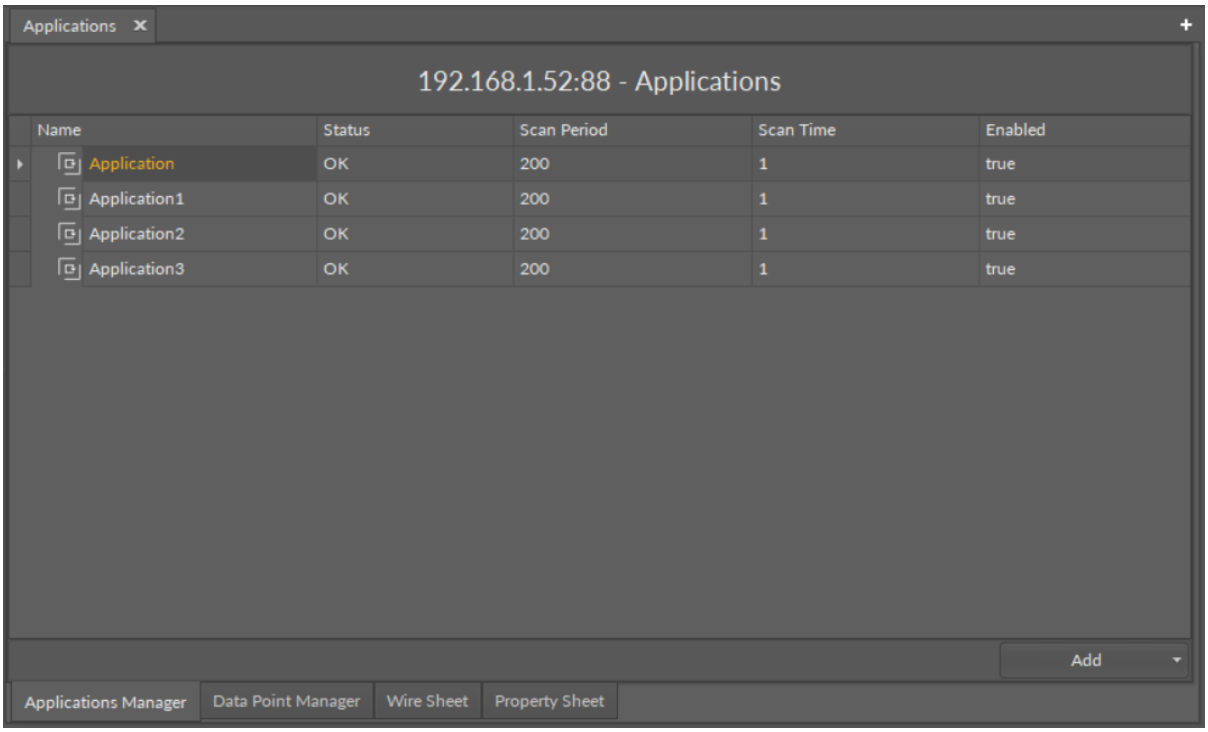


Figure 154. The Applications Manager

The Applications Manager lists all the Application components used on the device. The view shows the following fields:

- Name of the application;
- Status;
- Scan period;
- Scan time;
- Enabled or disabled status.

In the Applications Manager, it is possible to add, remove, copy, or duplicate Application components.

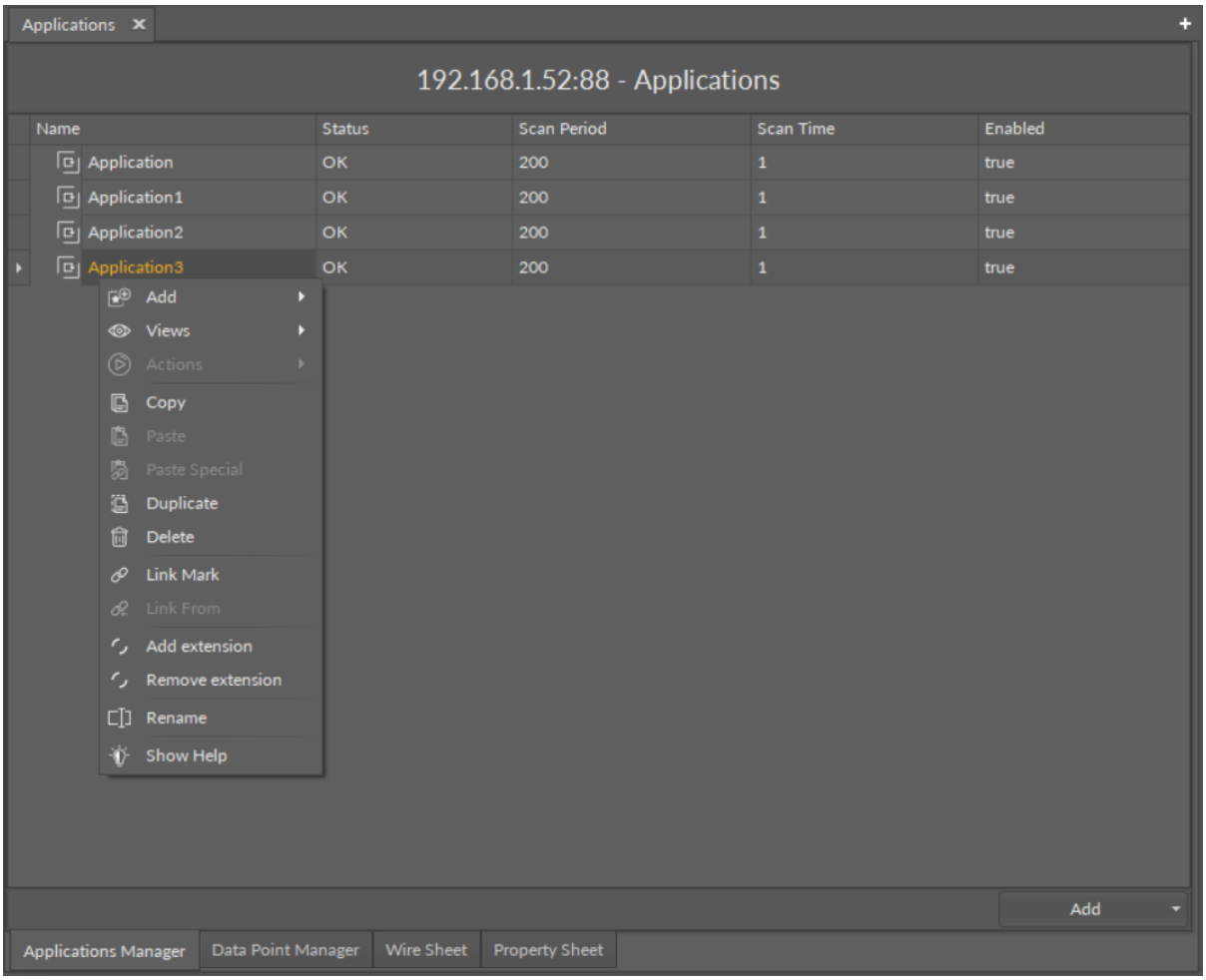


Figure 155. Options in the Applications Manager

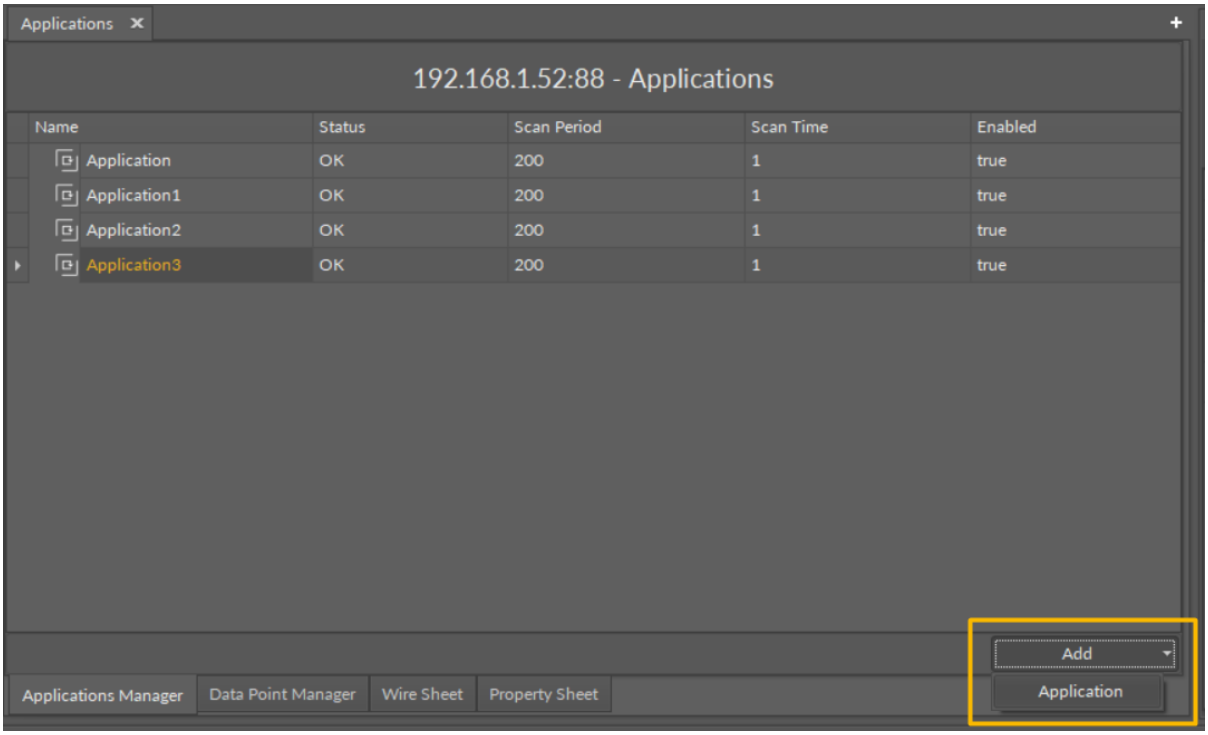


Figure 156. Options in the Applications Manager

Opening the Applications Manager

The Applications Manager view is accessible in the context menu of the Applications container.

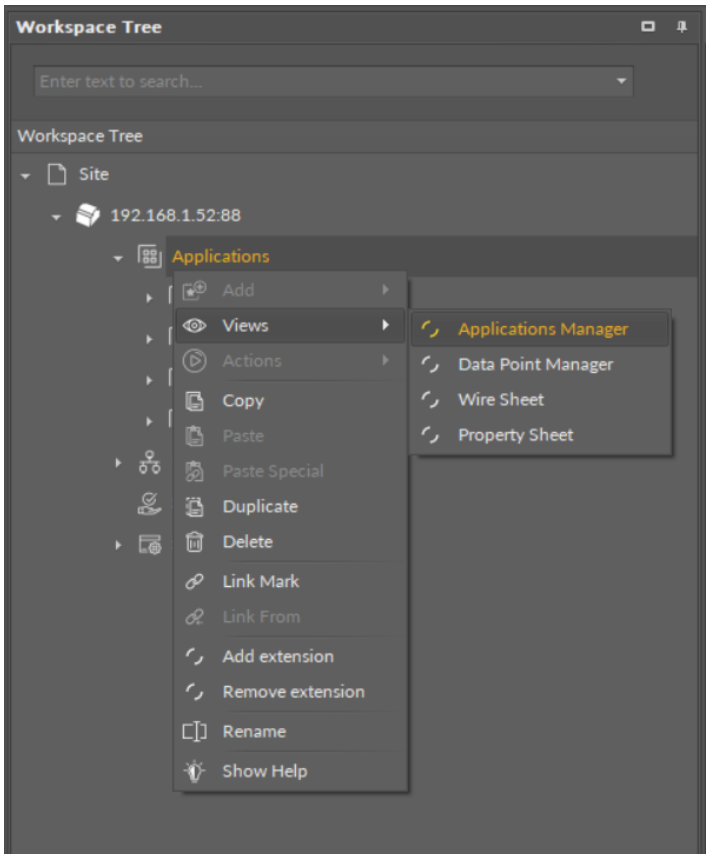


Figure 157. Accessing the Applications Manager from the Applications container context menu

The Applications Manager view is also automatically opened if the Applications container is double-clicked in the Workspace Tree window.

6.10.2 Data Point Manager

The Data Point Manager is a special view that allows to manage the Data Points available within the nano EDGE ENGINE license.

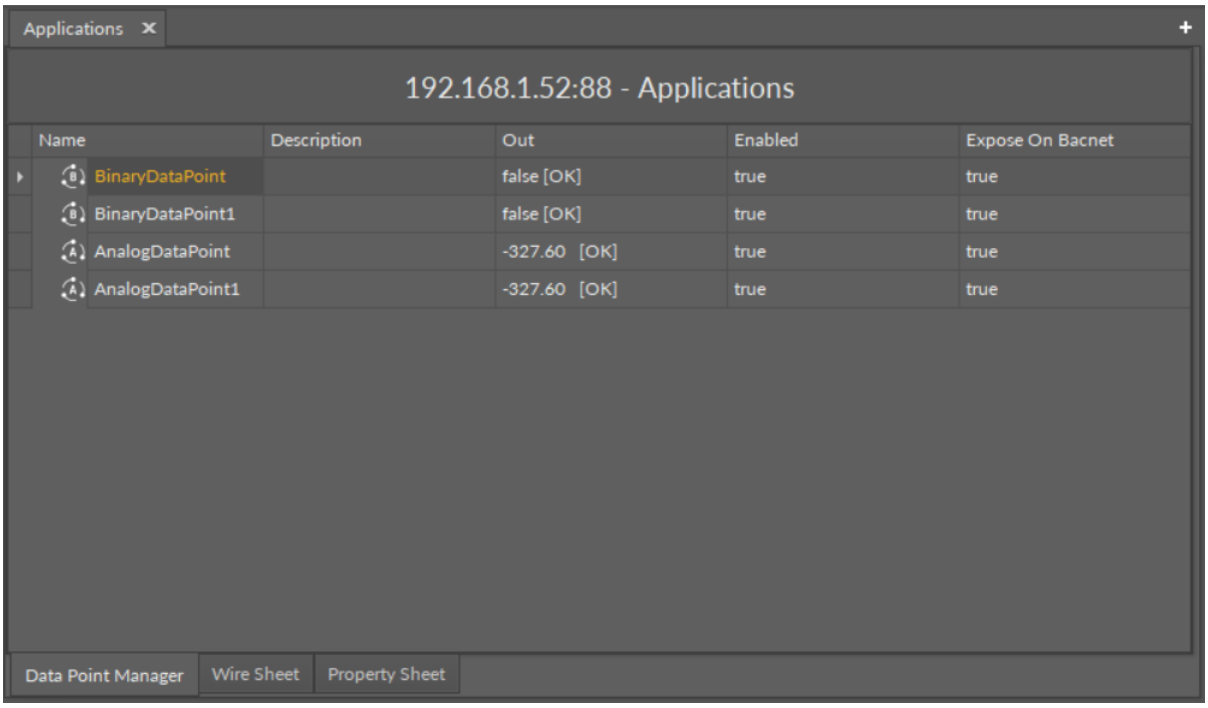


Figure 158. The Data Point Manager

The Data Point Manager lists all the Data Points used in applications saved on the device. The view shows the following fields:

- Equipment, which the Data Point belongs to;
- name of the Data Point;
- description;
- value on the Out slot;
- enabled or disabled status;
- exposition on BACnet status;
- BACnet object Id.

The Data Point Manager view is not editable; however, once a specific Data Point is clicked in the Manager view, it is displayed in the Object Properties window, where it can be freely edited. Also, the view allows for multiediting of compatible Data Points–if compatible Data Points are selected in the Data Point Manager, their common slots are available for multiedition in the Object Properties window.

Opening the Data Point Manager

The Data Point Manager view is accessible from two locations:

- in the context menu of the Applications container;
- in the context menu of the RAC18-IP (LocalDevice) component.

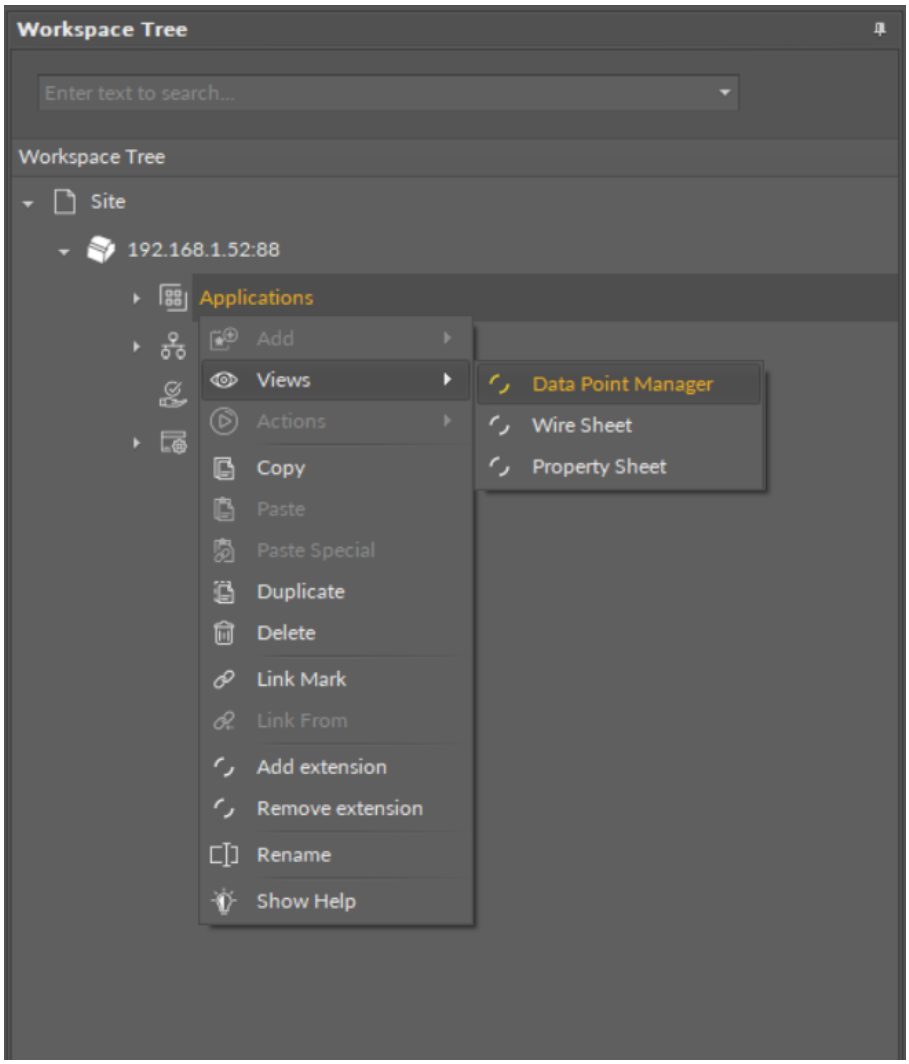


Figure 159. Accessing the Data Point Manager from the Applications container context menu

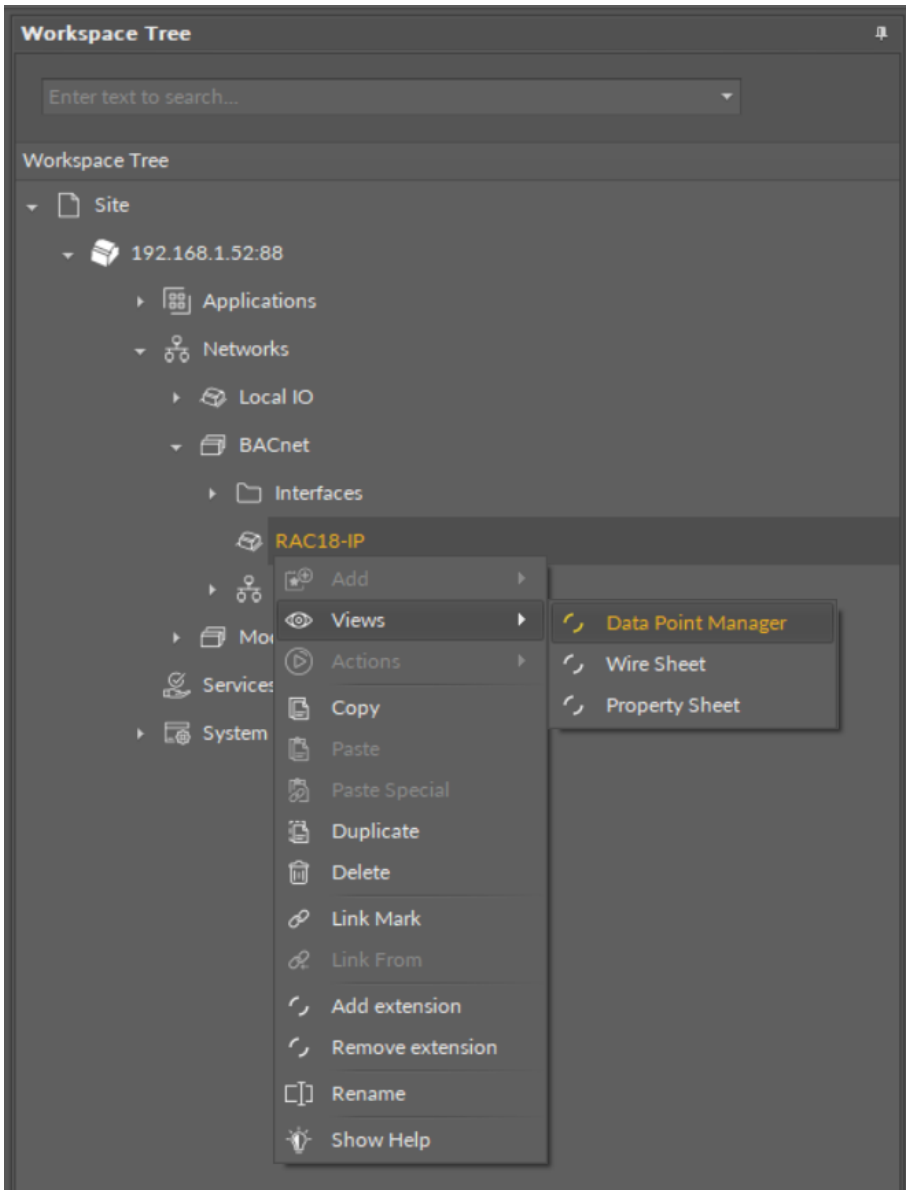


Figure 160. Accessing the Data Point Manager from the context menu of RAC18-IP (LocalDevice) component in the Networks container

The Data Point Manager view is also automatically opened if either the Applications container or the RAC18-IP (LocalDevice) component is double-clicked in the Workspace Tree window.

Licensing

The license for the new generation of iSMA CONTROLLI controllers driven by the nano EDGE ENGINE is constructed against the number of Data Points: each device based on the nano EDGE ENGINE is granted a specified number of license points (Data Points in this case), which can be used within applications. Therefore, the licensing system is only of quantitative, not functional, character–only the real number of Data Points in applications is taken into account, regardless of how many communication protocols are used to expose them, or how many network points are controlled. With the nano EDGE ENGINE-generation devices it is possible to create as big an application (or applications) as the number of licensed Data Points. No elements in the Networks, Services, or System containers are subject to license limitations, other than Data Points in the Applications container.

Note: In order to check the number of license points, please refer to the License in the device.

6.10.3 Equipment Manager

The Equipment Manager is a special view that allows to manage the Equipment components, Data Points, and Folder components added to a specific Application component.

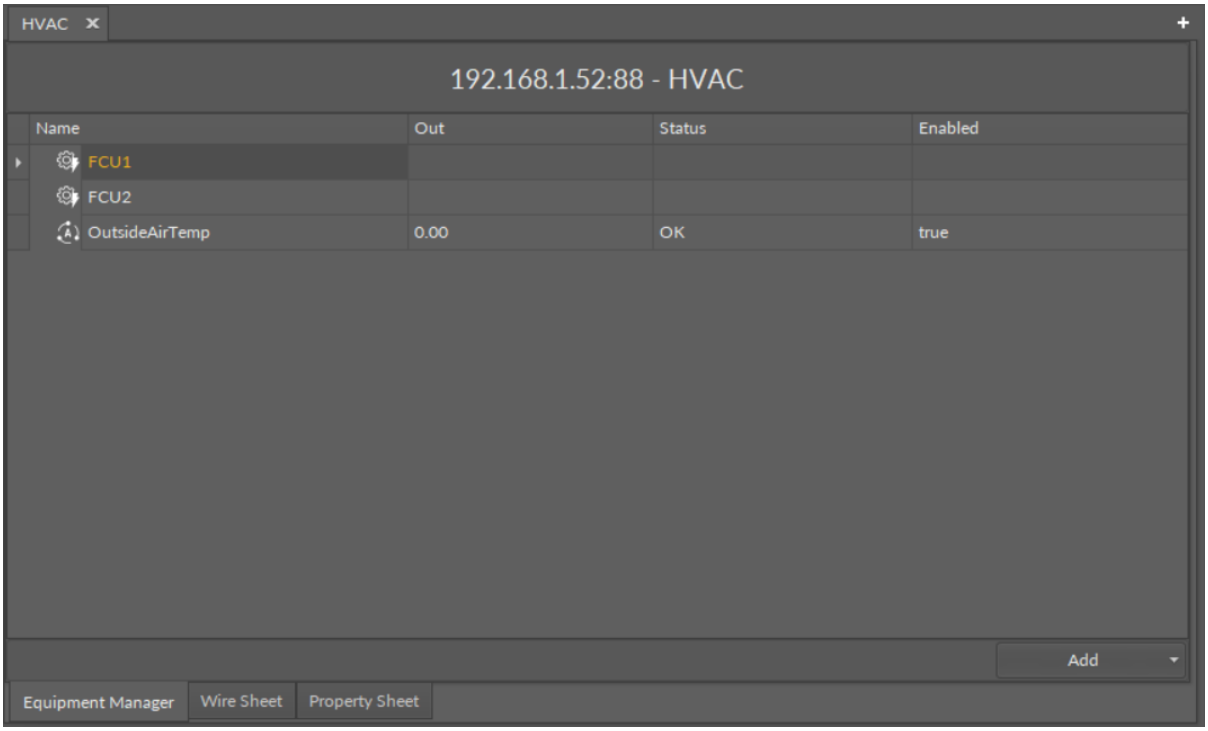


Figure 161. The Equipment Manager

The Equipment Manager lists all Equipment components and Data Points used in the specific Application component. The view shows the following fields:

- name of the Data Point;
- value on the Out slot;
- Data Point's status;
- enabled or disabled status.

The Equipment Manager view is not editable; however, it allows to add, remove, duplicate, and rename Equipment components, Data Points (withing the Equipment components and directly in the Application component), and Folder components, which help organize components in the application.

Once a specific Data Point is clicked in the Equipment Manager view, it is displayed in the Object Properties window, where it can be freely edited. Also, the view allows for multiediting of compatible Data Points–if compatible Data Points are selected in the Application Manager, their common slots are available for multiedition in the Object Properties window.

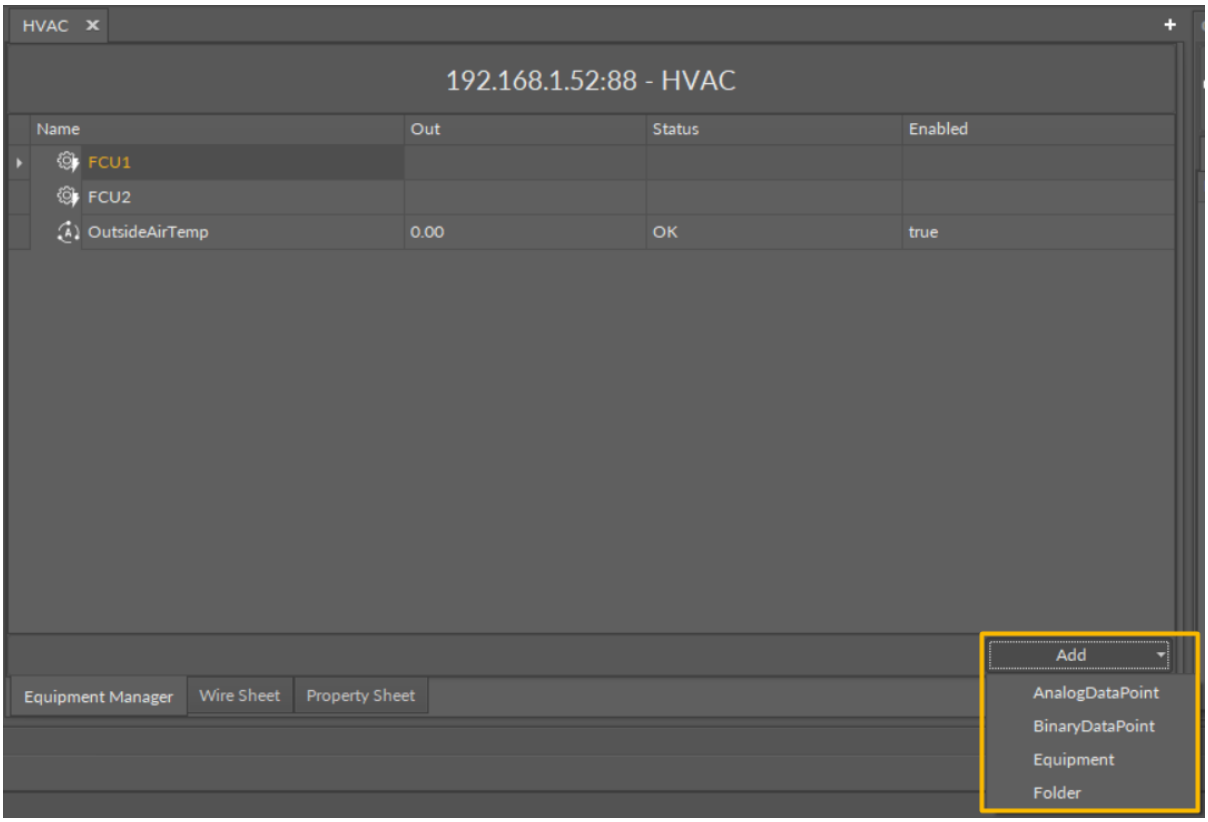


Figure 162. Options in the Equipment Manager

Opening the Equipment Manager

The Equipment Manager view is accessible in the context menu of the Application component.

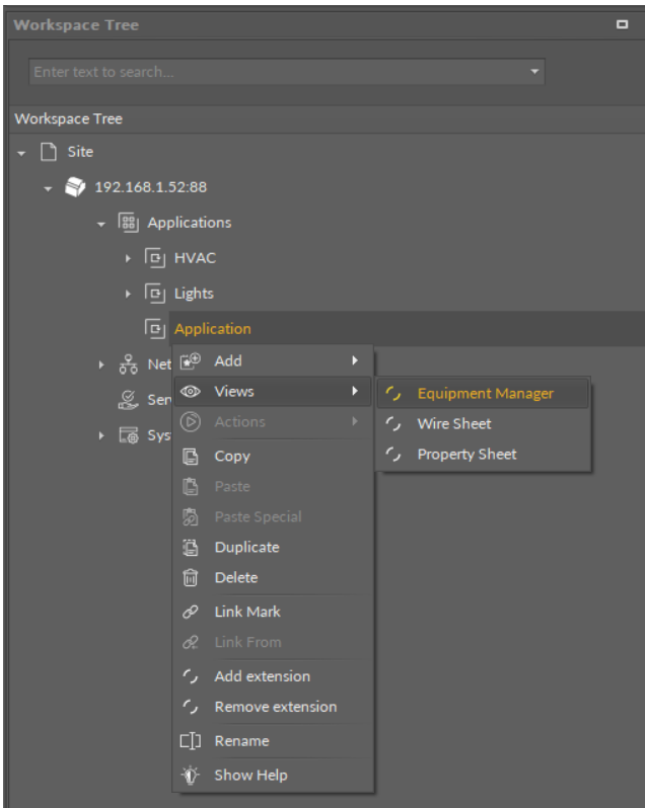


Figure 163. Accessing the Application Manager from the Applications container context menu

The Equipment Manager view is also automatically opened if either the Application component is double-clicked in the Workspace Tree window.

7 Networks Container

The Networks is a container that provides a space for components supporting external communications of the controller, for example, components managing physical inputs and outputs or the ones enabling communication protocols. Components within the Networks container provide for the controller's basic requirements to be able to communicate with external systems; these components allow to exchange fundamental automation data, which are used to create algorithms operating in user applications.

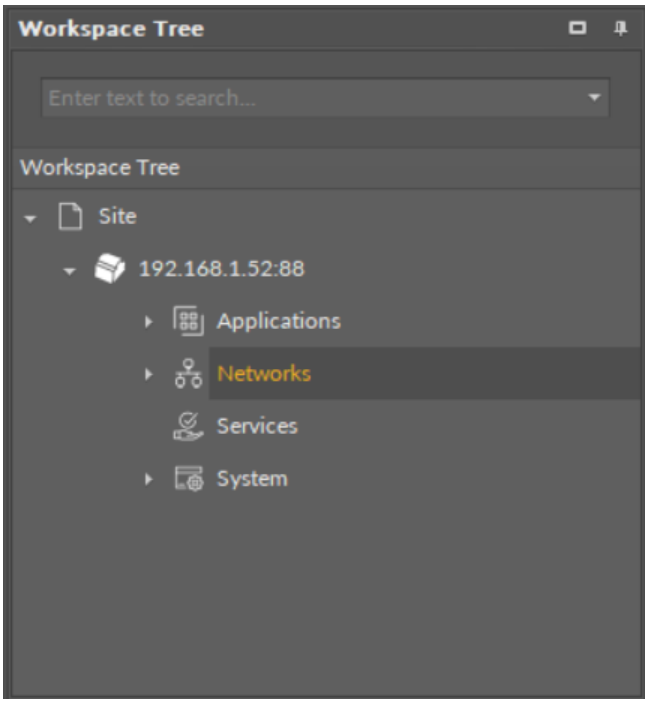


Figure 164. The Networks container

7.1 Networks Libraries

The nano EDGE ENGINE Networks libraries provide sets of components to manage external communications of the device. Libraries are grouped by specific functionalities, for example, managing local inputs and outputs and implementing a BACnet or Modbus protocol. Libraries designed for the Networks container are:

- Core library;
- IO library;
- BACnet library;
- Modbus library.

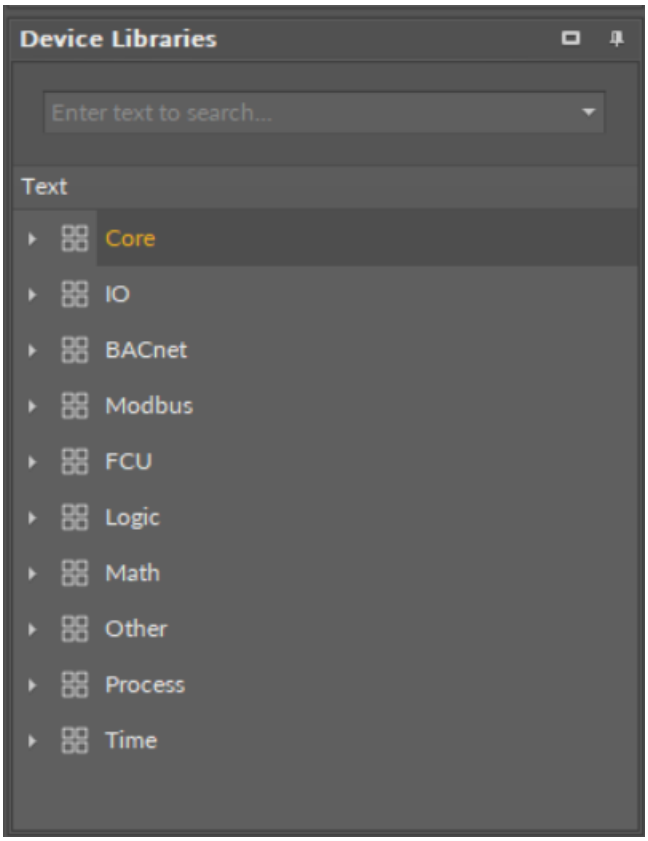


Figure 165. nano EDGE ENGINE libraries

7.2 Basic Concept of Networks

The basic idea for the Networks container is to group all components that manage external communications of the device. The Networks container includes components that allow the device to transfer the HVAC automation data needed for the device to communicate externally. These components cover the fundamental layer of HVAC data transmission and are crucial for the device to be able to properly exchange information. The network point class components are located in the Networks container; both data transmission from the network points and controlling them in the application is achieved by the Reference linking, which is a linking method recommended over linking input and output slots directly, as the standard linking method misses the advantage of transferring the status info.

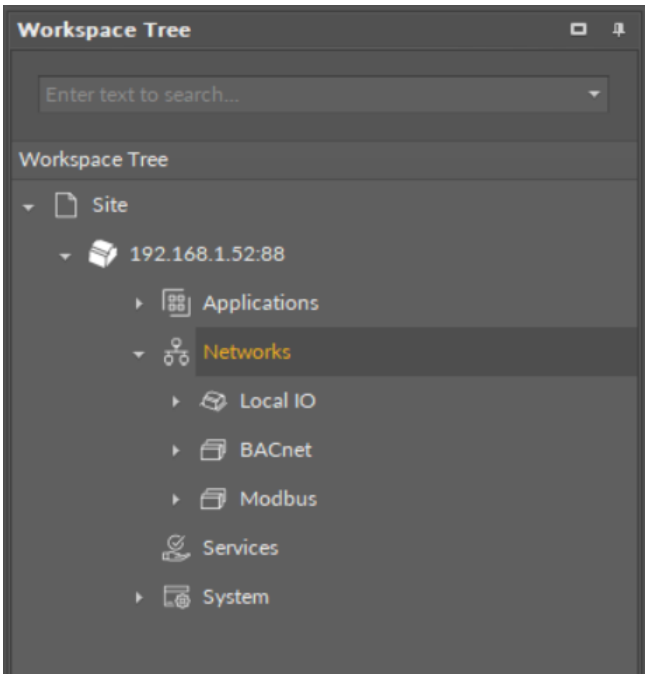


Figure 166. The Networks container

The components in the Networks container are meant to coordinate the external communications of the device: for example, the LocalIO manages the physical inputs and outputs, and the BACnet coordinates the exposition to the BACnet network.

7.3 Core (for Networks)

Applicable to OS version 1.0.0.4592

WARNING!

The Core library is a library containing components featured for both Applications and Networks containers. In this section only components for the Networks container are described!

For the Networks container, the Core library includes a component that is essential for setting up communication protocols:

- Network.

Note: The Folder component, described in the Core library for Applications container section, can be used in the Networks container.

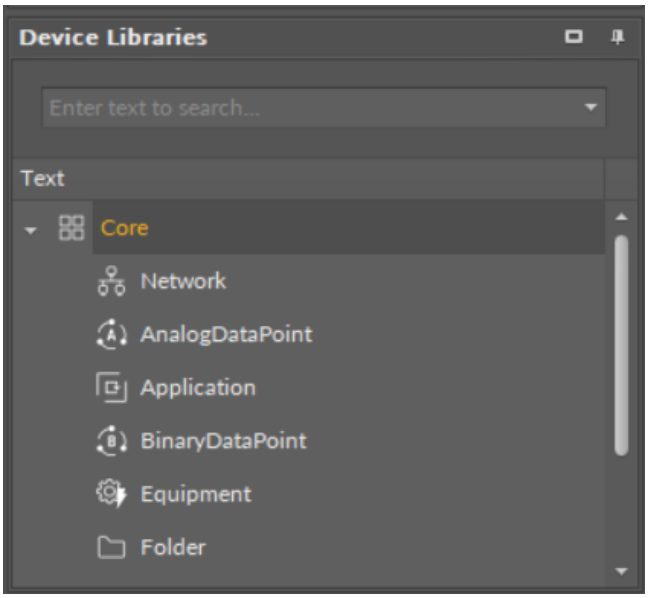


Figure 167. The Core library

This section also includes components for the Networks container available automatically in the structure:

- Interfaces:
- Ethernet;
- Serial.

The components in the Core library are universal for all implemented communication protocols.

7.3.1 Interfaces

Applicable to OS version 1.0.0.4592

The Interfaces is a folder-type component. It is designed to contain components configuring BACnet or Modbus communication ports. Similarly like the BACnet or Modbus component, the Interfaces folder is automatically included in the device structure, and, by default, it contains the Ethernet or Serial component.

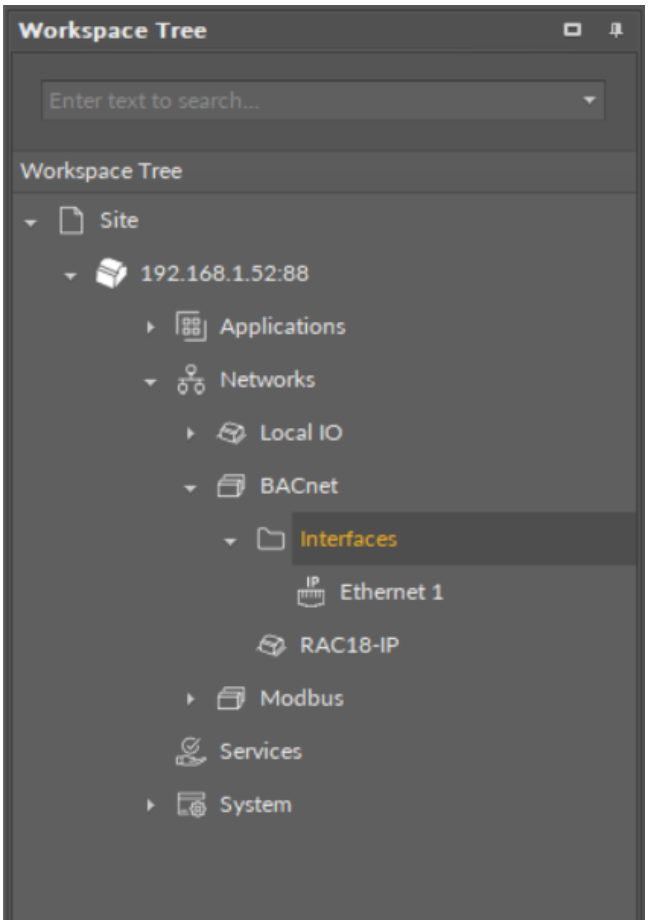


Figure 168. The Interfaces component for BACnet

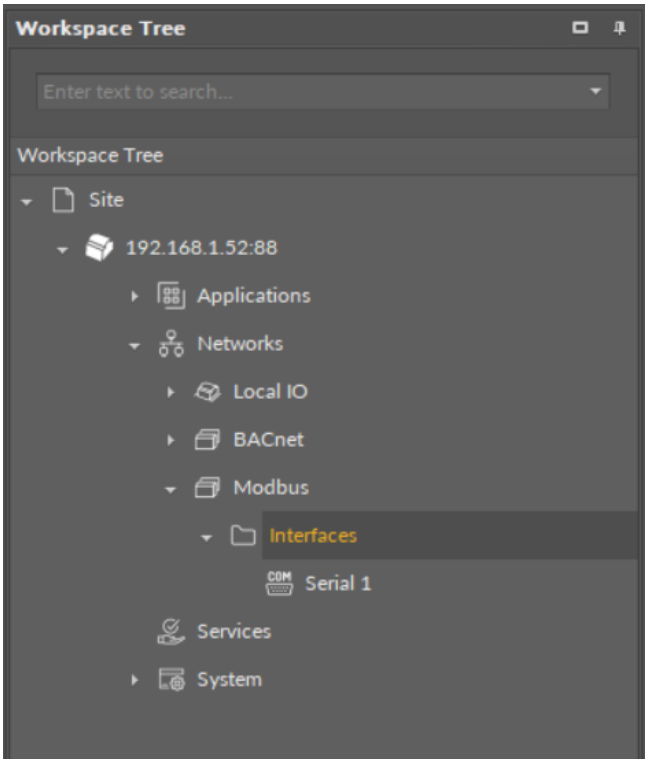


Figure 169. The Interfaces component for Modbus

The Interfaces component has no slots or actions.

7.3.2 Ethernet

Applicable to OS version 1.0.0.4592

The Ethernet component allows to activate the BACnet communication over IP and to identify the port adapter and the number for the BACnet network. The Ethernet component allows to identify the IP port number for BACnet communication (by default, it is 47808). It can be changed to any number from the slot's range (0-65535).

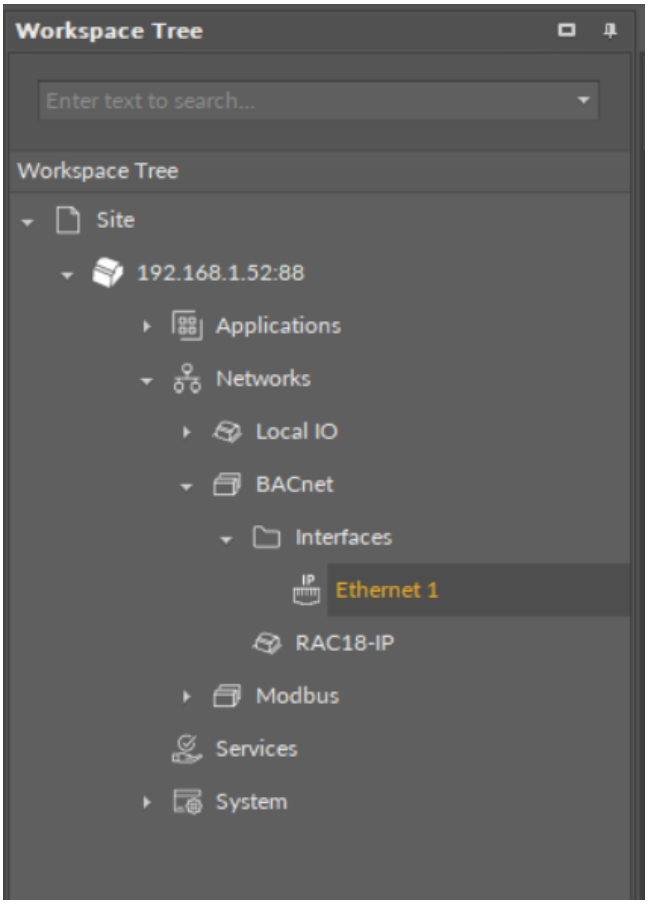


Figure 170. The Ethernet component

Slots

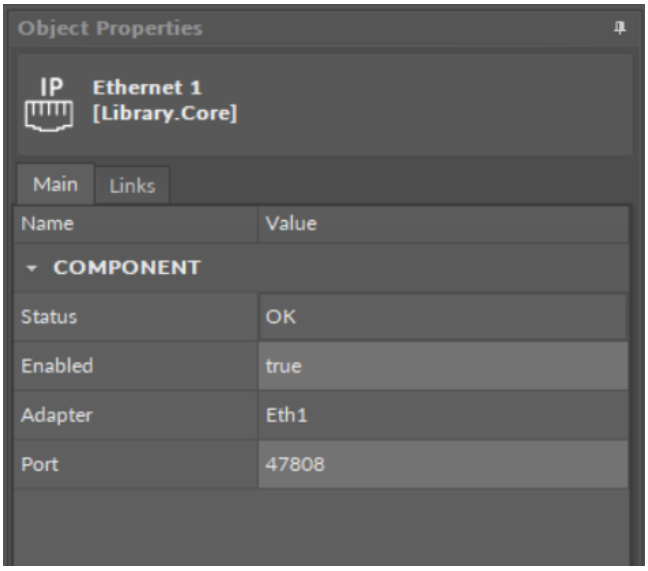


Figure 171. The Ethernet component slots

The Ethernet component has the following slots:

- Status: indicates the current status of the component. If the component works properly, its status is OK; the component's status becomes Disabled if its Enabled slot has been set to false.
  - Available information: Disabled, OK;
- Enabled: change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

Note: Disabling the component switches off both BACnet IP server and client functions, which means that the device is not visible as the BACnet device in the network, and, in case of the BACnet client configuration, the communication with server devices is switched off too.

- Adapter: informs, which communication port is used to connect to the network.
- Port: allows to set the number of the port used to communicate in the BACnet IP protocol.

7.3.3 Serial

Applicable to OS version 1.0.0.4592

The Serial component allows to configure the Modbus communication over the RS485 port.

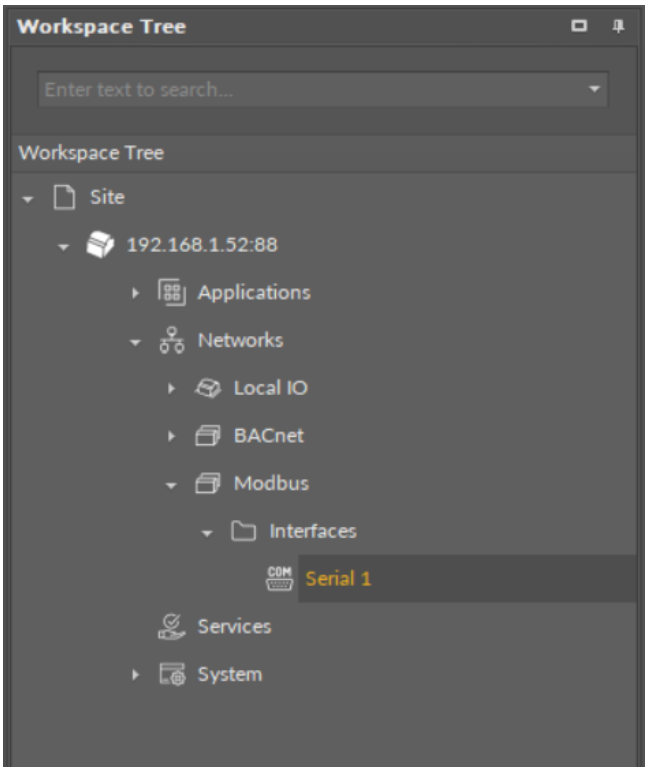


Figure 172. The Serial component

Slots

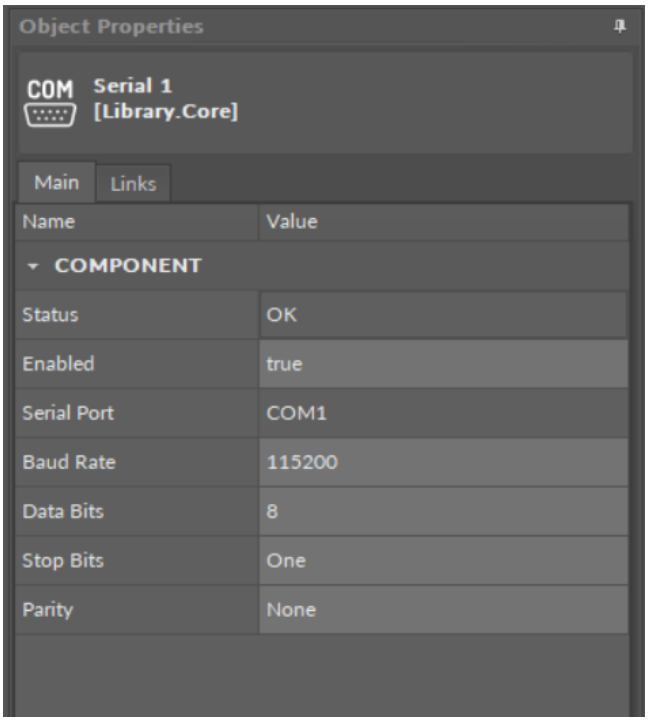


Figure 173. The Serial component slots

The Serial component has the following slots:

- **Status:** indicates the current status of the component. If the component works properly, its status is OK; the component's status becomes Disabled if its Enabled slot has been set to false.

- Available information: Disabled, OK.

- **Enabled:** change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

Note: By default, the component is enabled.

> **WARNING!**
>
> If the Serial component is disabled, the network configured to operate on the serial port defined in this component is also disabled.

- **Serial Port:** shows a serial communication port available in the device;
- **Baud Rate:** allows to set the baud rate;
  - Available settings: 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps;
- **Data Bits:** allows to set the data bits configuration of the Modbus data frame;
  - Available settings: 7-bits (Modbus ASCII, for future use) or 8-bits (Modbus RTU);
- **Stop Bits:** allows to set the stop bits configuration of the Modbus data frame;
  - Available settings: One, OnePointFive, Two;
- **Parity:** allows to set the parity bit configuration of the Modbus data frame;
  - Available settings: None, Even, Mark, Odd, Space.

7.3.4 Network

Applicable to OS version 1.0.0.4592

The Network component allows to configure the BACnet or Modbus client network. It allows to add remote devices (using the Device component) and configure basic polling schedule for such devices and their points.

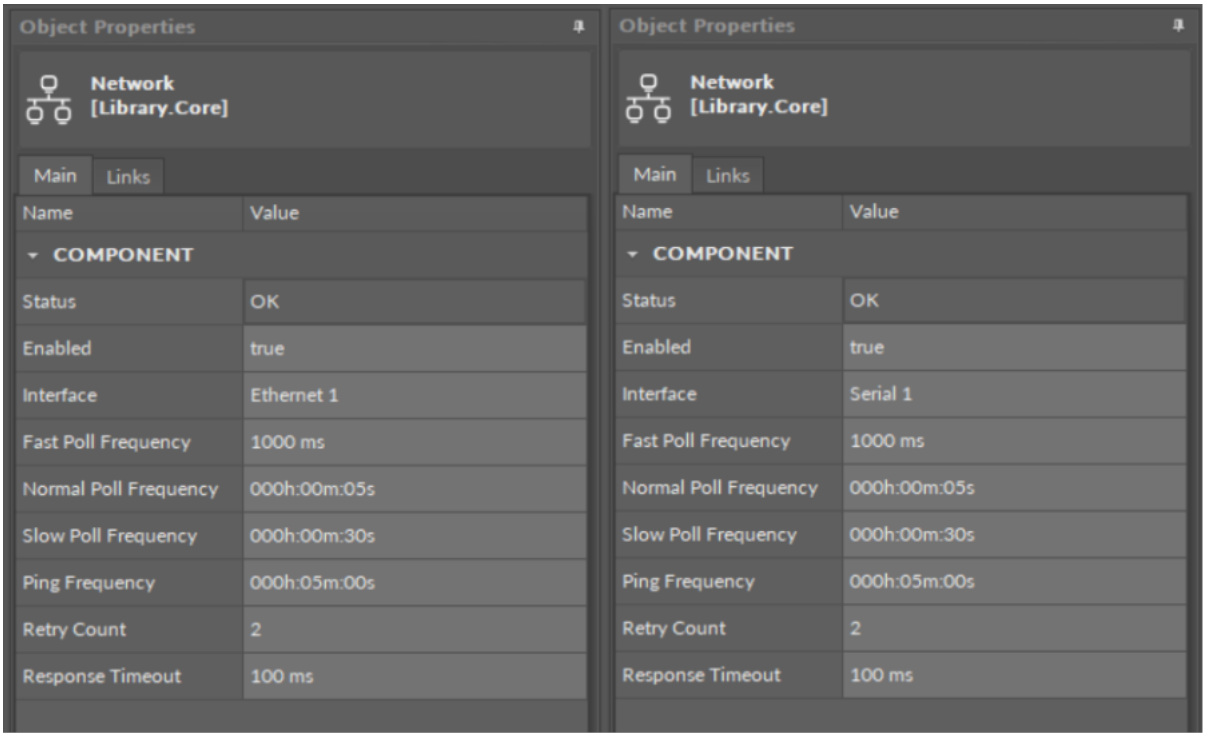


Figure 174. The Network component for BACnet (left) and Modbus (right)

The Network component has the following slots:

- **Status:** indicates the current status of the component. If the component works, its status is OK; the component's status becomes Disabled if its Enabled slot has been set to false.
    - Available information: Disabled, OK.
- **Enabled:** change of the slot's value enables or disables the component.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Interface:** allows to set the Ethernet interface for BACnet client network and the Serial port for Modbus;
- **Fast Poll Frequency:** sets the time between requests for the point's value sent in the fast mode;
- **Normal Poll Frequency:** sets the time between requests for the point's value sent in the normal mode;
- **Slow Poll Frequency:** sets the time between requests for the point's value sent in the slow mode;
- **Ping Frequency:** sets the time between testing requests to check the device's connection.
- **Retry Count:** shows a number of repeated requests;
- **Response Timeout:** time set to wait for the device's response.

Worth to Notice

In order for the network to be enabled, the Ethernet component (for BACnet) or the Serial component (for Modbus), configured for the port defined in the Adapter slot, has to be enabled too.

7.3.5 Folder (Networks)

Applicable to OS version 1.0.0.4592

The Folder component is a grouping component, which can be used to gather other components to help organize the Workspace Tree. The Folder can be added to the device structure, however, it cannot be added directly to the container. The Folder component can be freely renamed to facilitate categorization of components included within.

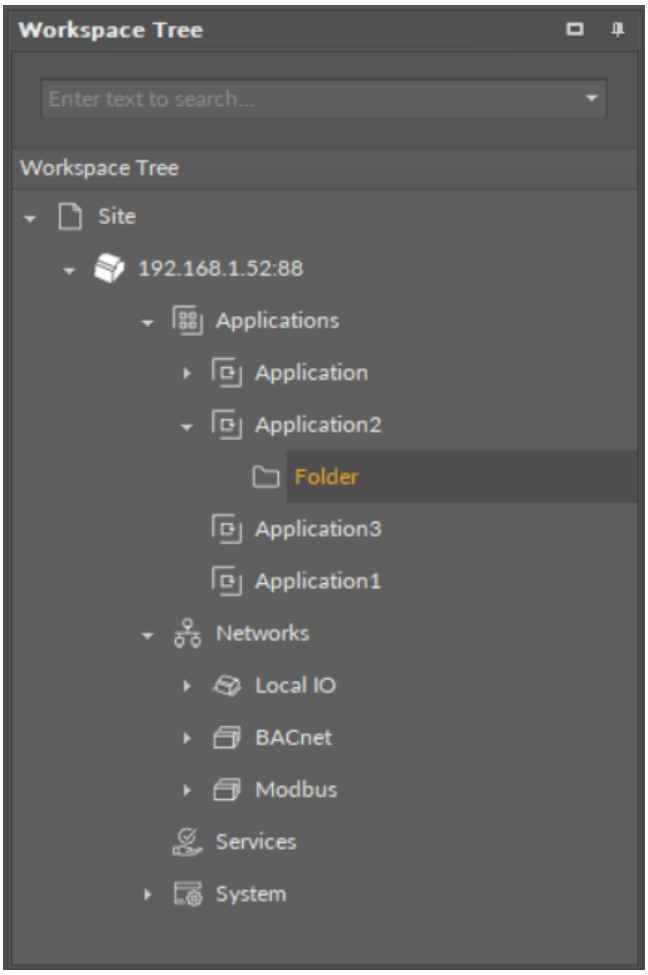


Figure 175. The Folder component

The Folder component has no slots nor actions.

Note: The Folder component can be used both in the Applications and Networks containers.

7.4 LocalIO

Applicable to OS version 1.0.0.4592

The IO library consists of seven network point components, dedicated to read and control the inputs and outputs of the actual device. The IO components are defined for various type of physical inputs and outputs.

In order to operate properly, the IO components must be placed in the Network container, under their superior component, the LocalIO.

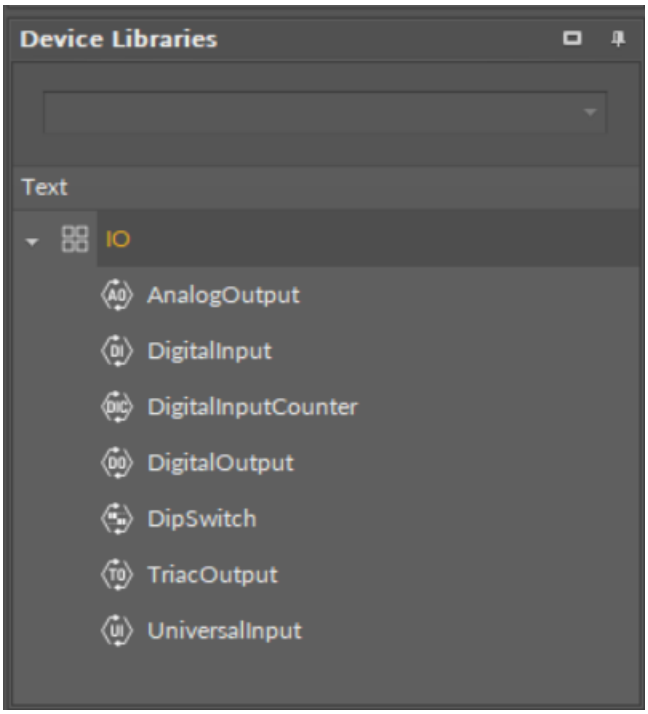


Figure 176. The LocalIO component

7.4.1 LocalIO Component

The LocalIO component is a component that manages the communication with physical inputs and outputs of the device. It allows to configure communication for seven network point class components, each dedicated to service different type of input or output.

In the Property Sheet the LocalIO component shows its status and its inferior components. It is possible to expand each of these components and control them from the LocalIO's Property Sheet view.

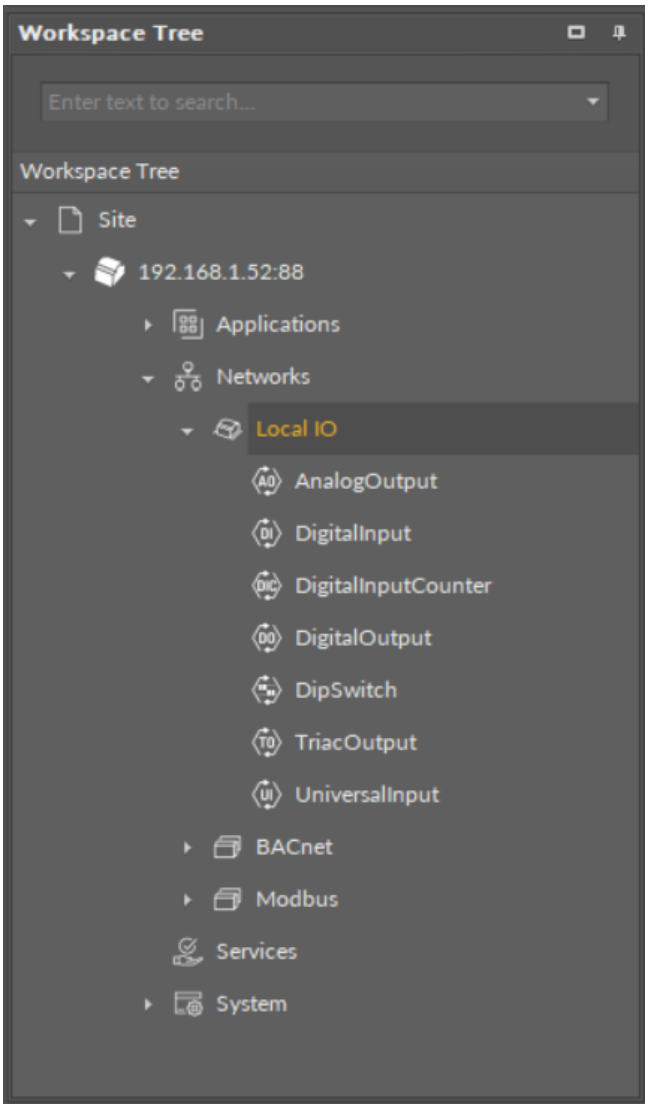


Figure 177. The LocalIO component

Slots

The LocalIO component has the following slots:

- Status: indicates the current status of the component. If the component works, its status is OK; the component's status becomes Disabled if it has been stopped by a manual action.
    - Available information: Disabled, OK.
- Enabled: allows to enable or disable the component and its inferior components (even if their Enabled slots are set to true):
    - Available settings: true (enabled), false (disabled);
- Digital Inputs: indicates the number of physical digital inputs in the device;
- Digital Outputs: indicates the number of physical digital outputs in the device;
- Universal Inputs: indicates the number of physical universal inputs in the device;
- Analog Outputs: indicates the number of physical analog outputs in the device;
- Triac Outputs: indicates the number of physical triac outputs in the device;
- Fast Poll Frequency: sets the time between requests for the point's value sent in the fast mode;

- Normal Poll Frequency: sets the time between requests for the point's value sent in the normal mode;
- Normal Poll Frequency: sets the time between requests for the point's value sent in the normal mode.

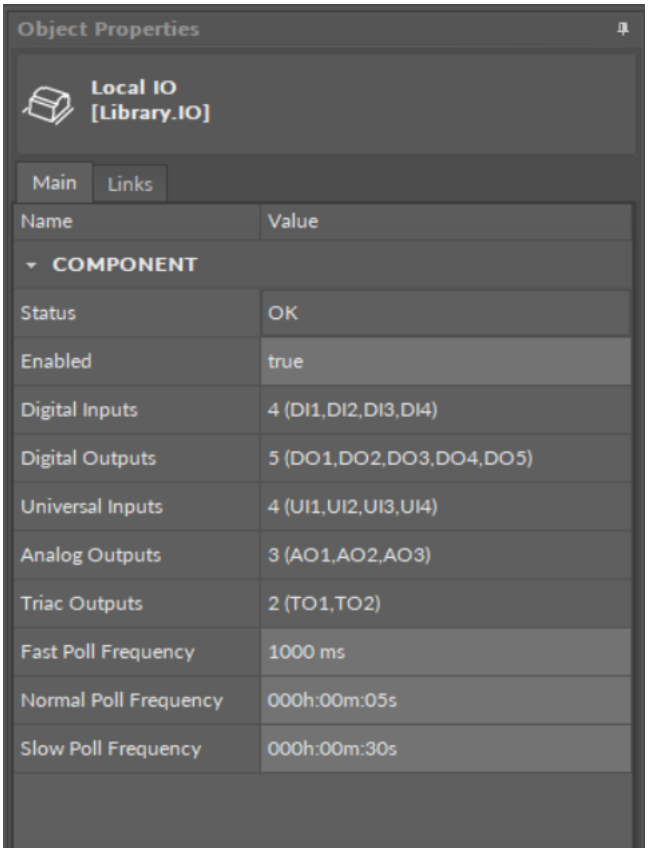


Figure 178. The LocalIO component's slots

7.4.2 UniversalInput

Applicable to OS version 1.0.0.4592

The UniversalInput component is an IO point (network point class) component that retrieves data from the physical universal input of a device. The component allows to configure the universal input according to its purpose—the universal input may serve as a voltage, current, digital, resistance, or temperature input—in order to communicate properly with the linked Data Point. The component allows to change the type of sensor, its resolution, and to apply an input signal filtering.

The component can pass data to the Data Point class component by linking Reference slots; the UniversalInput component may be linked to an Analog Data Point. In order to operate properly, the UniversalInput component must be located under the LocalIO component in the Networks container and have its unique number assigned in the Address slot.

Note: Before using the component, make sure that its individual address is assigned and its Status is OK.

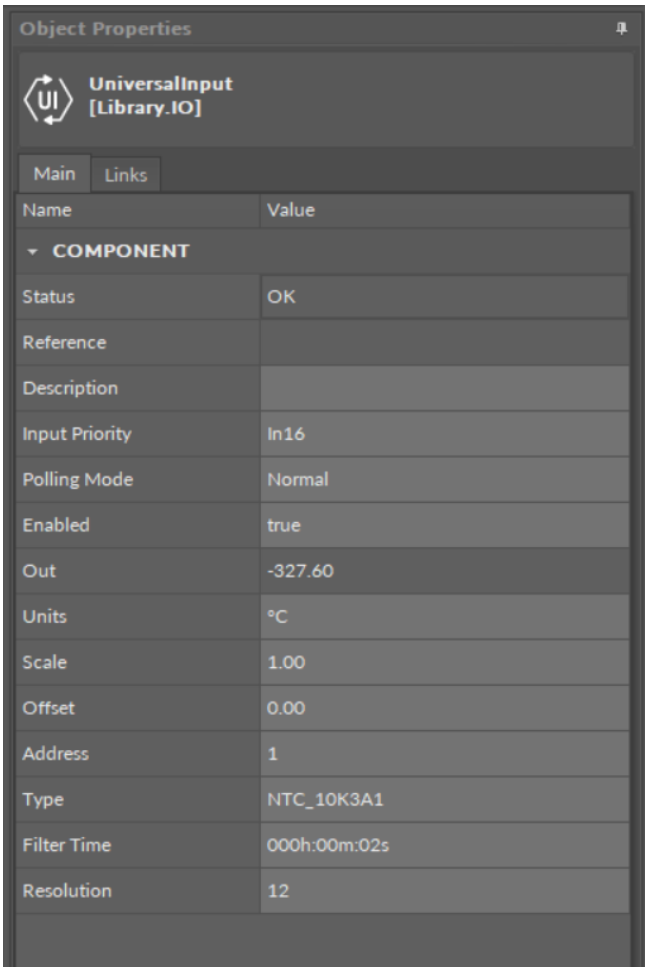


Figure 179. The UniversalInput component

Slots

The UniversalInput component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null, 0, or exceeding an available range;
  - Available information: Disabled, Fault, OK;
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

By default, once the Reference link is created from the network point to the Data Point it sets the input priority to 16, which later can be changed manually.

- **Description:** an additional detailed information about a service that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, or any other information the user finds applicable.
- **InputPriority:** allows to select the input number in the Data Point, which the value from the network point class component's output is sent to; by default, the priority is None and sets to 16 after linking with a Data Point (can be changed manually).
  - Available settings: none, 1-16.

Note: The Reference link from the network point to the Data Point cannot be changed to a 17th, default, priority.

- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
  - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component—if the component becomes disabled, it stops to read values from the physical input; by default, the component is enabled.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** a real value read from the physical input of the address set in the Address slot.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

Note: The Out slot value depends on the input type defined in the Type slot.

- **Units:** defines a unit of the Out slot value, depending on the sensor type set in the Type slot—the unit is automatically set once the sensor type is selected in the Type slot; however, it can be manually adjusted by the user;
  - Available units: according to BACnet units;
- **Scale:** sets a fixed scaling factor for output linearization; the Out value is calculated according to the linear function formula (y=ax+b), and the Scale slot set the a value of the formula;
- **Offset:** sets a fixed offset value to the output value; the Out value is calculated according to the linear function formula (y=ax+b), and the Offset slot set the b value of the formula;
- **Address:** allows setting an address of a physical input of the device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.
  - Available settings: 1-n, where "n" stands for the number of actual inputs in the device.
- **Type:** defines a sensor type for proper reading of values.
  - Available settings: Voltage, Current, Digital, Resistance, Temperature (supported temperature sensors:

Celsius degrees: NTC 10K3A1, NTC 10K4A1, NTC 10K Carel, NTC 20K6A1, NTC 2,2K3A1, NTC 3K3A1, NTC 30K6A1, NTC SIE1, NTC TAC1, NTC SAT1, PT1000, NI1000, NI1000 21C, NI1000 LG,

Fahrenheit degrees: NTC 10K Type2, NTC 10K Type3, NTC 20K NTC, NTC 3K, PT1000, NI1000 32F, NI1000 70F).

Note: Choosing one input type disables the rest—the universal input works only as a type defined in the InputType slot, and it will not adjust automatically if the incoming value changes its type. By default, the NTC 10K3A1 temperature sensor is set as an input type; if any other input type is relevant for the particular universal input, it needs to be adjusted manually.

- **Filter Time:** sets the filtering period in order to avoid the read values peaks.

- Available settings: 0-60 s (by default, the filtering time is set to 2 s).

- **Resolution:** defines a precision of data conversion from analog data to bits for further data processing in the controller's software.
  - Available settings: 12 or 16-bit (12-bit resolution provides fast data processing, whereas 16-bit resolution provides more precise data; however, it may hamper the speed of data processing).

Note: As the default value of the Resolution slot is 12-bit, it needs to be emphasized that PT1000 and NI1000 temperature sensors work correctly only with the 16-bit resolution; therefore, if these types of sensors are set in the Type slot, the Resolution slot is automatically adjusted to a 16-bit value.

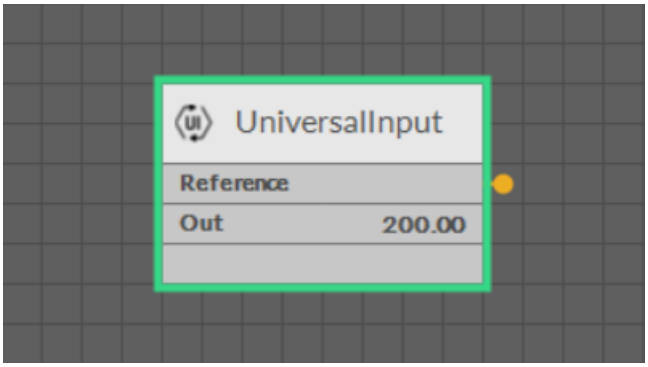


Figure 180. The UniversalInput component linked

7.4.3 DigitalInput

Applicable to OS version 1.0.0.4592

The DigitalInput component is an I/O point (network point class) component that retrieves data from a physical digital input of a device. The component allows to configure the digital input in order to communicate properly with the linked Data Point. The component can pass data to the Data Point class component by linking Reference slots; the DigitalInput component may be linked to a Binary Data Point. In order to operate properly, the DigitalInput component must be located under the LocalIO component in the Networks container and it have a unique number assigned in the Address slot.

Note: Before using the component, make sure that its individual address is assigned and its Status is OK.

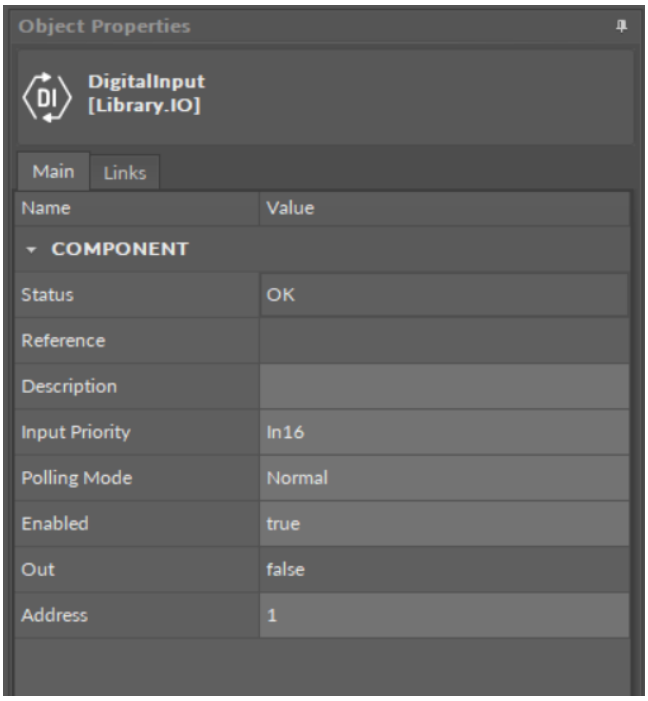


Figure 181. The DigitalInput component

Slots

The DigitalInput component has the following slots:

- Status: indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null, 0, or exceeding an available range;
  - Available information: Disabled, Fault, OK;
- Reference: a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

By default, once the Reference link is created from the network point to the Data Point it sets the input priority to 16, which later can be changed manually.

- Description: an additional detailed information about a service that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, or any other information the user finds applicable.
- InputPriority: allows to select the input number in the Data Point, which the value from the network point class component's output is sent to; by default, the priority is None and sets to 16 after linking with a Data Point (can be changed manually).
  - Available settings: none, 1-16.

Note: The Reference link from the network point to the Data Point cannot be changed to a 17th, default, priority.

- Polling Mode: allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
  - Available settings: fast, normal, slow;

- **Enabled:** change of the slot's value enables or disables the component—if the component becomes disabled, it stops to read values from the physical input; by default, the component is enabled.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** a real value read from the physical input of the address set in the Address slot.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- **Address:** allows setting an address of a physical input of the device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.
    - Available settings: 1-n, where "n" stands for the number of actual inputs in the device.

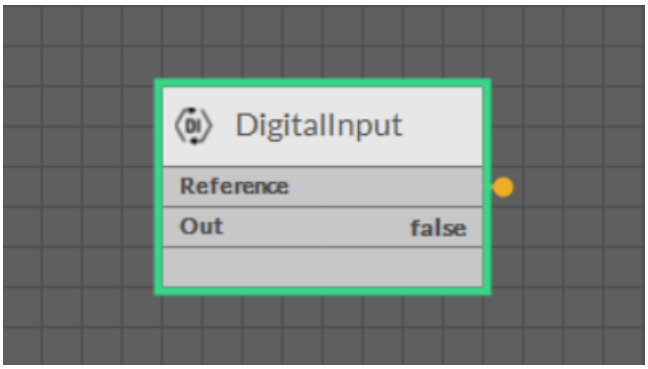


Figure 182. The DigitalInput component linked

7.4.4 DigitalInputCounter

Applicable to OS version 1.0.0.4592

The DigitalInputCounter component counts the pulses on rising edge in a physical digital input of the device. The counter may be employed, for example, to add up a number of pulses for used water meters or similar purposes. Before the DigitalInputCounter component is addressed to a physical input, the counter's default value is 0 and it changes once the slot starts reading values from a connected meter or sensor.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the counter's value is null.

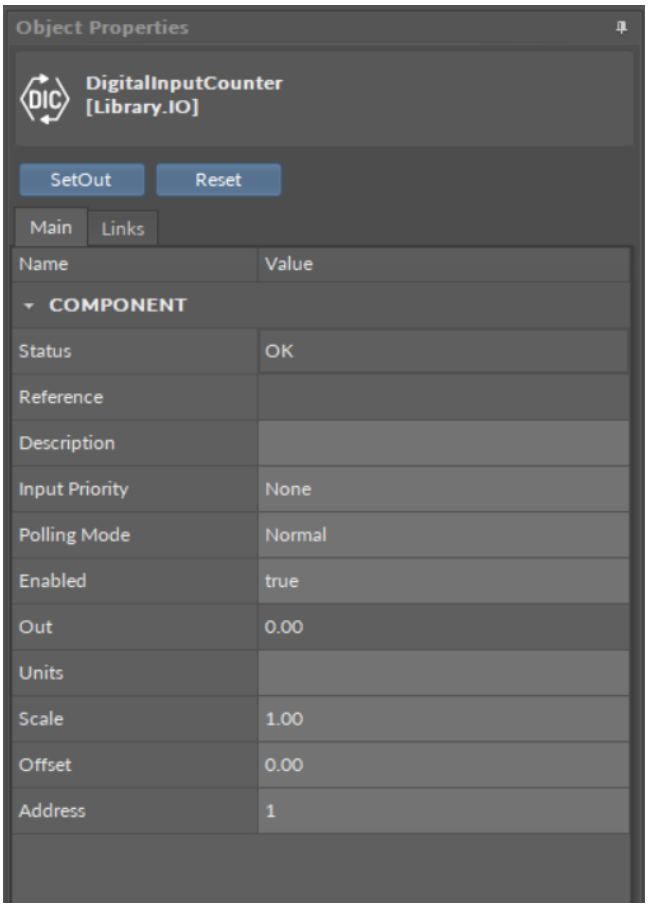


Figure 183. The DigitalInputCounter component

Slots

The DigitalInputCounter component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null, 0, or exceeding an available range;
  - Available information: Disabled, Fault, OK;
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

By default, once the Reference link is created from the network point to the Data Point it sets the input priority to 16, which later can be changed manually.

- **Description:** an additional detailed information about a service that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, or any other information the user finds applicable.
- **InputPriority:** allows to select the input number in the Data Point, which the value from the network point class component's output is sent to; by default, the priority is None and sets to 16 after linking with a Data Point (can be changed manually).
  - Available settings: none, 1-16.

Note: The Reference link from the network point to the Data Point cannot be changed to a 17th, default, priority.

- Polling Mode: allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
  - Available settings: fast, normal, slow;
- Enabled: change of the slot's value enables or disables the component—if the component becomes disabled, it stops to read values from the physical input; by default, the component is enabled.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- Out: shows a number of rising edges from the physical digital input of the address set in the Address slot.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- Units: defines a unit of the Out slot value; no unit is set by default;
- Scale: sets a fixed scaling factor for output linearization; the Out value is calculated according to the linear function formula (y=ax+b), and the Scale slot set the a value of the formula;
- Offset: sets a fixed offset value to the output value; the Out value is calculated according to the linear function formula (y=ax+b), and the Offset slot set the b value of the formula;
- Address: allows setting an address of a physical input of the device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.
  - Available settings: 1-n, where "n" stands for the number of actual inputs in the device.

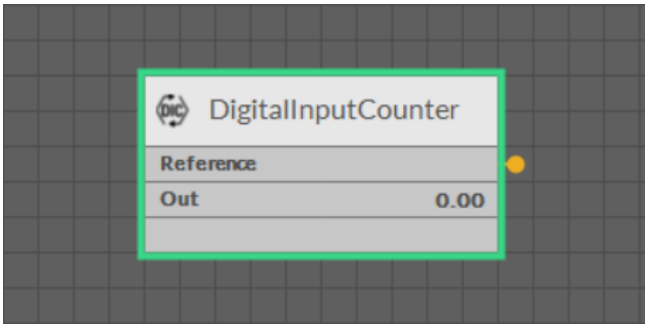


Figure 184. The DigitalInputCounter component linked

Actions

- SetOut: sets the Out slot to a specific value;
- Reset: sets the Out slot to the 0 value.

7.4.5 AnalogOutput

Applicable to OS version 1.0.0.4592

The AnalogOutput component is an IO point (network point class) component that transfers the value to a physical analog output of a device. The component allows to transfer data received from the linked Data Point in order to control the physical analog output of the device, and allows to set the polling mode of the point. The component can pass data from the Data Point class component by linking Reference slots; the AnalogOutput component may be linked from an Analog Data Point.

In order to operate properly, the AnalogOutput component must be located under the LocalIO component in the Networks container.

Note: Before using the component, make sure that its individual address is assigned and its Status is OK.

The AnalogOutput component includes an action that allows to set its Out value in case no Data Point class component is linked to it.

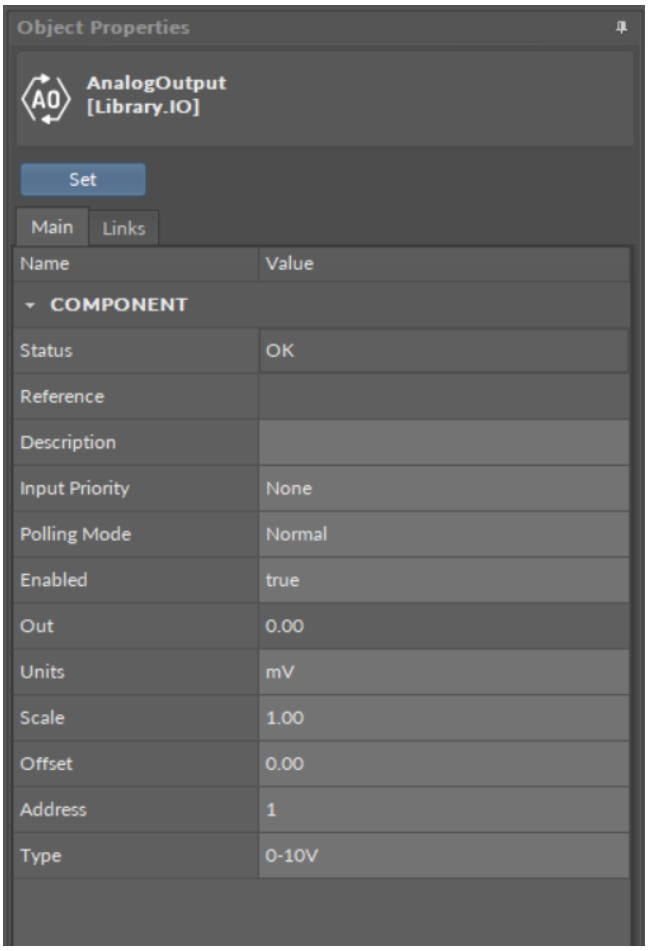


Figure 185. The AnalogOutput component

Slots

The Analog Output component has the following slots:

- Status: indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null, 0, or exceeding an available range;
    - Available information: Disabled, Fault, OK;

- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

Note: The Reference links work on a change-of-value. If a value in the Data Point's Out slot changes, it is immediately transferred to a network point linked by the Reference link– such change is not dependent on an application cycle.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.
- **InputPriority:** allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; by default, the priority is none.
  - Available settings: none, 1-16.
- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
  - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component—if the component becomes disabled, it stops to transfer values to the physical output; by default, the component is enabled.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** displays a value transferred to the output of the address set in the Address slot.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- **Units:** defines a unit of the Out slot value, depending on the selected mode of operation—the unit is automatically set once the mode type is selected in the Type slot, however, it can be manually adjusted by the user;
  - Available units: according to BACnet units;
- **Scale:** sets a fixed scaling factor for output linearization; the value transferred to a physical output is calculated according to the inversed linear function y=(x-b)/a, and recalculated according to the linear function y=ax+b to the Out slot; the Scale slot sets the a value of the formula;
- **Offset:** sets a fixed offset value to the output value; the value transferred to a physical output is calculated according to the inversed linear function y=(x-b)/a, and recalculated according to the linear function y=ax+b to the Out slot; the Offset slot sets the b value of the formula;

Example

The output linearization allows to adjust an output value to the requirements of a controlled device/equipment. For example:

- if an analog output controls an actuator, which operates within a 2-10 V range, the AnalogOutput component can be scaled to produce output fitting the required range:
  - the Type slot set to 0-10 V;
  - the Scale slot value set to **1.25** (10000 mV/8000 mV);
  - the Offset slot value scaled to **-2500** (redefining the output's range to **2-10 V**: 2000 mV/0.8 range).

Note: Though the Type slot is set to 0-10 V, the actual value is controlled in mV. Accordingly, the offset must be set in mV.

- to scale the output to control a device, which operates within a 0-5 V range, the AnalogOutput parameters have to be set as follows:
  - the Type slot set to 0-10 V;
  - the Scale slot value set to **2**;
  - the Offset slot value scaled to 0.
- to translate 0-100% to 0-10000 mV, the AnalogOutput component can be scaled to produce output fitting the required range:
  - the Scale slot value set to 0.01 (100%/10000 mV);
  - the Offset slot value scaled to 0.

Note: Incorrect scaling of the results in the calculated values (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input from Reference link and on the Out slot. If the scaling is correct, the values are identical.

- **Address:** allows setting an address of a physical output of the device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.
  - Available settings: 1-n, where "n" stands for the number of actual outputs in the device.
- **Type:** defines the mode of operation—the component may operate as a voltage or digital output, or in pulse width modulation mode; once the component has been added, the component's mode is set to Voltage;
  - Available modes: Voltage, Digital, PWM001Hz, PWM01Hz, PWM1Hz, PWM10Hz, PWM100Hz.

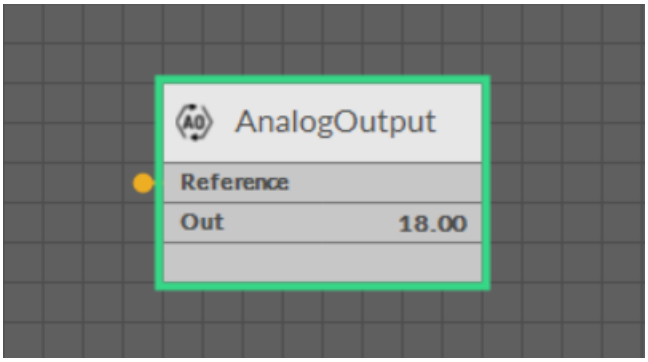


Figure 186. The AnalogOutput component linked

Action

- Set: sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

7.4.6 DigitalOutput

Applicable to OS version 1.0.0.4592

The DigitalOutput component is an I/O point (network point class) component that transfers the value to a physical digital output of a device. The component allows to transfer data received from the linked Data Point in order to control the physical digital output of the device. The component can pass data from the Data Point class component by linking Reference slots; the DigitalOutput component may be linked from a Binary Data Point. In order to operate properly, the DigitalOutput component must be located under the LocalIO component in the Networks container.

Note: Before using the component, make sure that its individual address is assigned and its Status is OK.

The DigitalOutput component includes an action that allows to set its Out value in case no Data Point class component is linked to it.

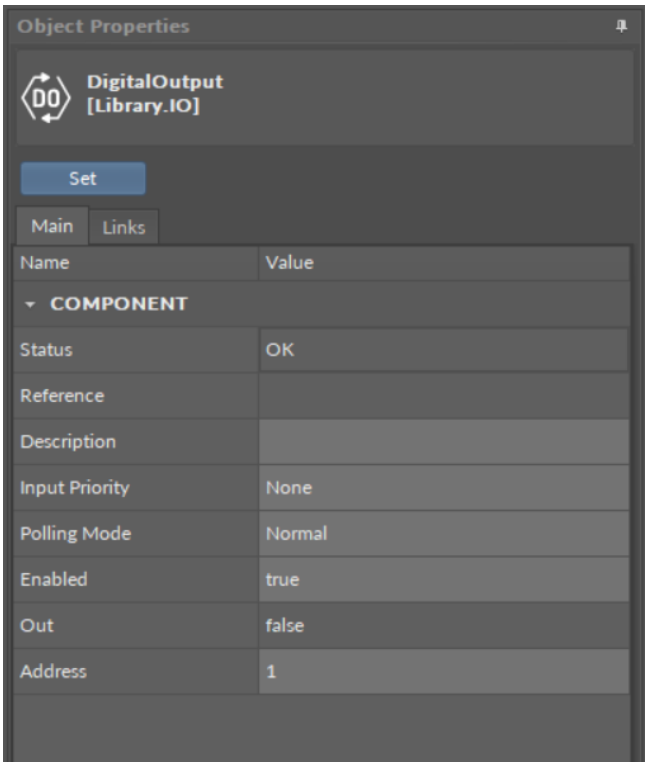


Figure 187. The DigitalOutput component

Slots

The DigitalOutput component has the following slots:

- Status: indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is

in false. The component's status is Fault, once the Address slot is null, 0, or exceeding an available range;
    - Available information: Disabled, Fault, OK;
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

Note: The Reference links work on a change-of-value. If a value in the Data Point's Out slot changes, it is immediately transferred to a network point linked by the Reference link– such change is not dependent on an application cycle.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.
- **InputPriority:** allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; by default, the priority is set to none.
    - Available settings: none, 1-16.
- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
    - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component—if the component becomes disabled, it stops to transfer values to the physical output; by default, the component is enabled.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** displays a value transferred to the output of the address set in the Address slot.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- **Address:** allows setting an address of a physical output of the device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.
    - Available settings: 1-n, where "n" stands for the number of actual outputs in the device.

Action

The DigitalOutput component has the following action:

- **Set:** sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

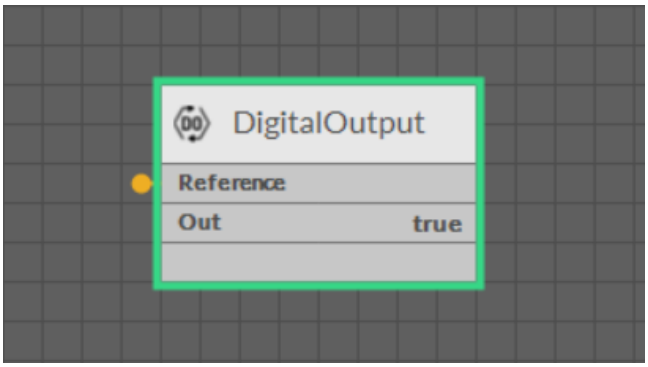


Figure 188. The DigitalOutput component linked

7.4.7 TriacOutput

Applicable to OS version 1.0.0.4592

The TriacOutput component is an I/O point (network point class) component that transfers the value to a physical triac output of a device. The component allows to transfer data received from the linked Data Point in order to control the physical triac output of the device. The component can pass data from the Data Point class component by linking Reference slots; the TriacOutput component may be linked from an Analog Data Point. In order to operate properly, the TriacOutput component must be located under the LocalIO component in the Networks container.

Note: Before using the component, make sure that its individual address is assigned and its Status is OK.

The TriacOutput component includes an action that allows to set its Out value in case no Data Point class component is linked to it.

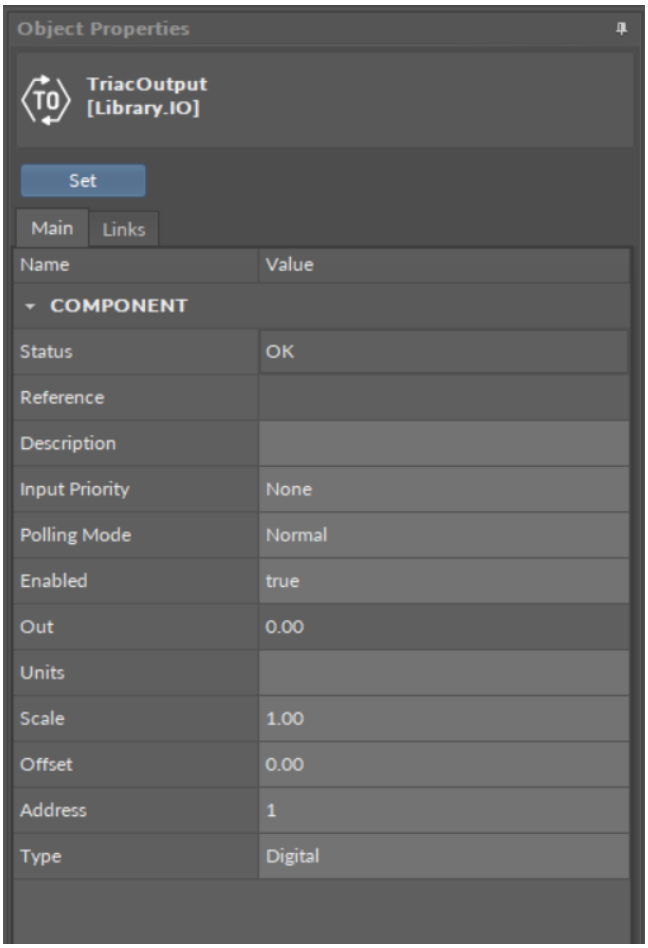


Figure 189. The TriacOutput component

Slots

The TriacOutput component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null, 0, or exceeding an available range;
  - Available information: Disabled, Fault, OK;
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

Note: The Reference links work on a change-of-value. If a value in the Data Point's Out slot changes, it is immediately transferred to a network point linked by the Reference link– such change is not dependent on an application cycle.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the

user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

- **InputPriority:** allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; by default, the priority is set to none.
  - Available settings: none, 1-16.
- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
  - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component—if the component becomes disabled, it stops to transfer values to the physical output; by default, the component is enabled.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** displays a value transferred to the output of the address set in the Address slot.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- **Units:** defines a unit of the Out slot value, depending on the selected mode of operation—the unit is automatically set once the mode type is selected in the Type slot, however, it can be manually adjusted by the user;
  - Available units: according to BACnet units;
- **Scale:** sets a fixed scaling factor for output linearization; the value transferred to a physical output is calculated according to the inversed linear function y=(x-b)/a, and recalculated according to the linear function y=ax+b to the Out slot; the Scale slot sets the a value of the formula;
- **Offset:** sets a fixed offset value to the output value; the value transferred to a physical output is calculated according to the inversed linear function y=(x-b)/a, and recalculated according to the linear function y=ax+b to the Out slot; the Offset slot sets the b value of the formula;

Note: Incorrect scaling of the results in the calculated values (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input from Reference link and on the Out slot. If the scaling is correct, the values are identical.

- **Address:** allows setting an address of a physical output of the device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.
  - Available settings: 1-n, where "n" stands for the number of actual outputs in the device.
- **Type:** defines the mode of operation—the component may operate as a digital output, or in pulse width modulation mode; once the component has been added, the component's mode is set to Digital;
  - Available modes: Digital, PWM001Hz, PWM01Hz, PWM1Hz, PWM10Hz.

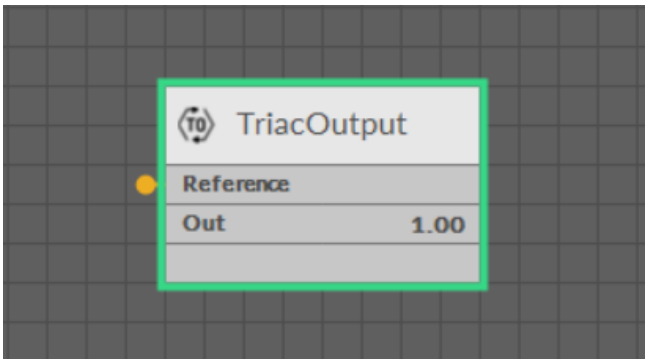


Figure 190. The TriacOutput component linked

Action

- Set: sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

7.4.8 DipSwitch

Applicable to OS version 1.0.0.4592

The DipSwitch component is an I/O point (network point class) component, which allows to read the bit states of DIP switches installed in the front panel of the RAC18-IP device. There are three DIP switches installed in the front panel: one 6-position (S1) and two 8-position (S2, S3). The DipSwitch component values are updated in every execution cycle of the LocalIO component.

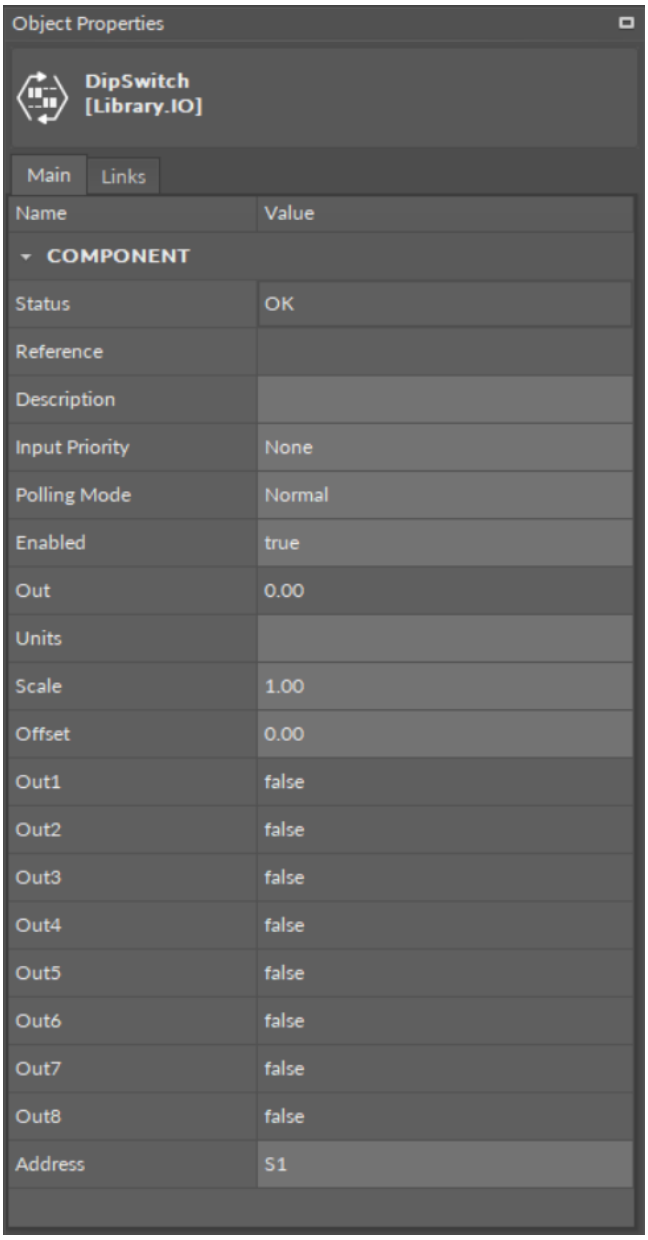


Figure 191. The DipSwitch component slots

Slots

The DipSwitch component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false;
  - Available information: Disabled, OK;
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

By default, once the Reference link is created from the network point to the Data Point it sets the input priority to 16, which later can be changed manually.

- **Description:** an additional detailed information about a service that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, or any other information the user finds applicable.
- **InputPriority:** allows to select the input number in the Data Point, which the value from the network point class component's output is sent to; by default, the priority is none and sets to 16 after linking with a Data Point;
    - Available settings: none, 1-16.

Note: The Reference link from the network point to the Data Point cannot be changed to a 17th, default, priority.

- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value—by default, the polling mode is set to normal;
    - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component—if the component becomes disabled, it stops to read values from the physical input; by default, the component is enabled.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** displays the binary sum of bits (bit no. 1 = 2^0, + 2^1 for bit no. 2, etc.);
- **Units:** defines a unit of the Out slot value; no unit is used here by default;
- **Scale:** sets a fixed scaling factor for output linearization; the Out value is calculated according to the linear function formula (y=ax+b), and the Scale slot set the a value of the formula;
- **Offset:** sets a fixed offset value to the output value; the Out value is calculated according to the linear function formula (y=ax+b), and the Offset slot set the b value of the formula;
- **Out1-Out8:** shows the binary value of the respective bit on the addressed DIP switch;
    - Available states: true or false;

Worth to Notice

In case of the 6-position DIP switch, the Out7 and Out8 slots are permanently false. The 6th switch in the true state restores factory settings in the controller.

- **Address:** allows setting the address of the DIP switch, which bits are read in slots Out1-Out8;
    - Available settings: S1 (6-position DIP switch), S2 (8-position DIP switch), S3 (8-position DIP switch).

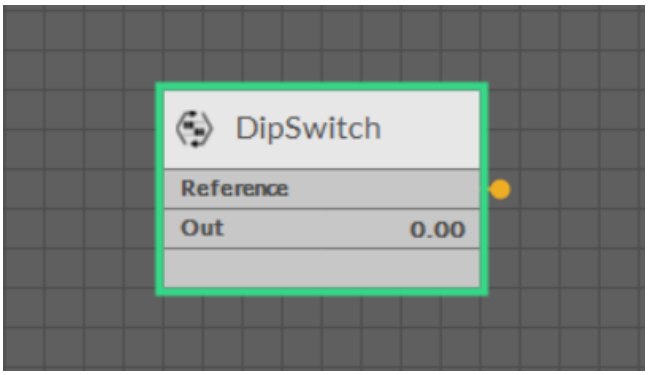


Figure 192. The DipSwitch component linked

Find out more about the Reference linking.

7.4.9 Quick Start-up of LocalIO

In order to have the components available in the IO library working properly, it is necessary to follow these steps:

Step 1: Having the device added correctly in the iC Tool, expand the Networks container, and go to the LocalIO component.

Step 2: Go to the Device Libraries and expand the IO library. Choose components to be added—components may be added one by one or grouped in one selection. Drag the selected component(s) and drop it(them) under the LocalIO component in the Network container.

Step 3: Go to each added component in the LocalIO, open its Property Sheet (or go the the LocalIO Property Sheet and expand each component), and set a proper number in the Address slot (a number representing the address of the physical input or output of the device). Configure all the other slots (Units, Type, etc.) according to the purpose of the component in the application. Make sure to save the changes.

Worth to Notice:

In order to facilitate working with LocalIO component, a special Point Manager view has be developed.

Ready to Use: Configured components are ready to be included in the application.

Point Manager for LocalIO

The LocalIO Point Manager view is available for the LocalIO component. It lists all I/O points added to the LocalIO component, and shows their:

- Out slot value;
- unit (for numeric values);
- status;
- number;

- enabled or disabled state.

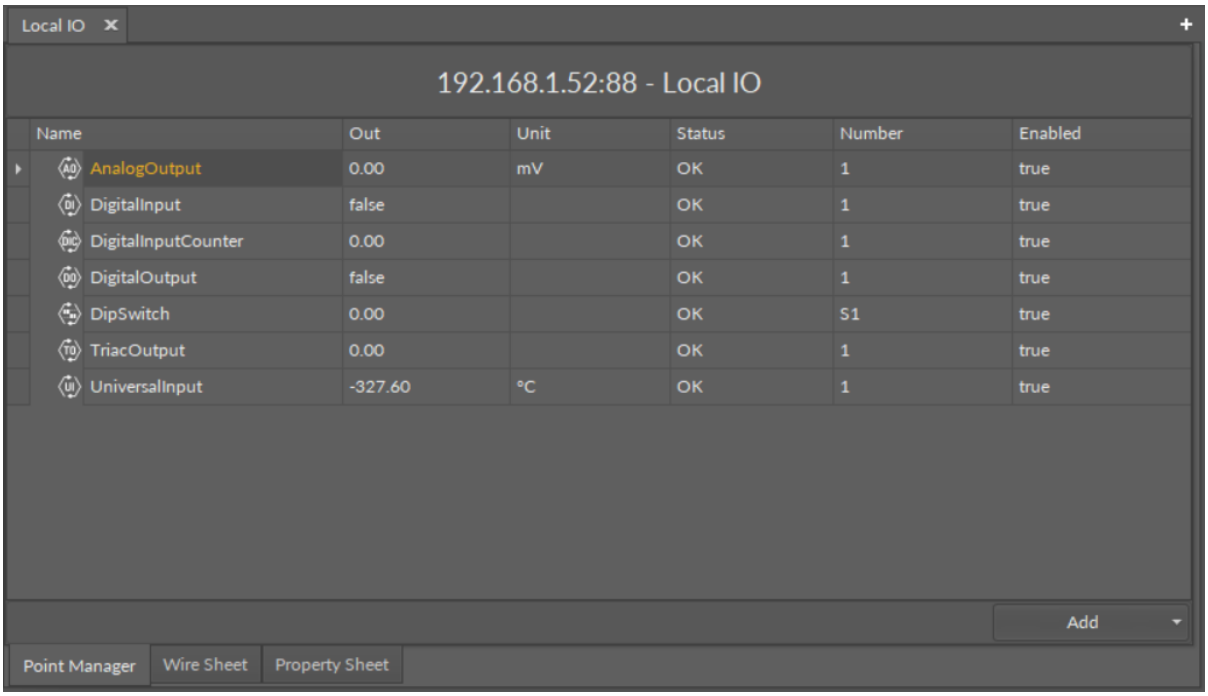


Figure 193. The Point Manager view

Opening Point Manager

The Point Manager view is accessible from the context menu of the LocalIO component. It is also automatically opened if the LocalIO component is double-clicked in the Workspace Tree window.

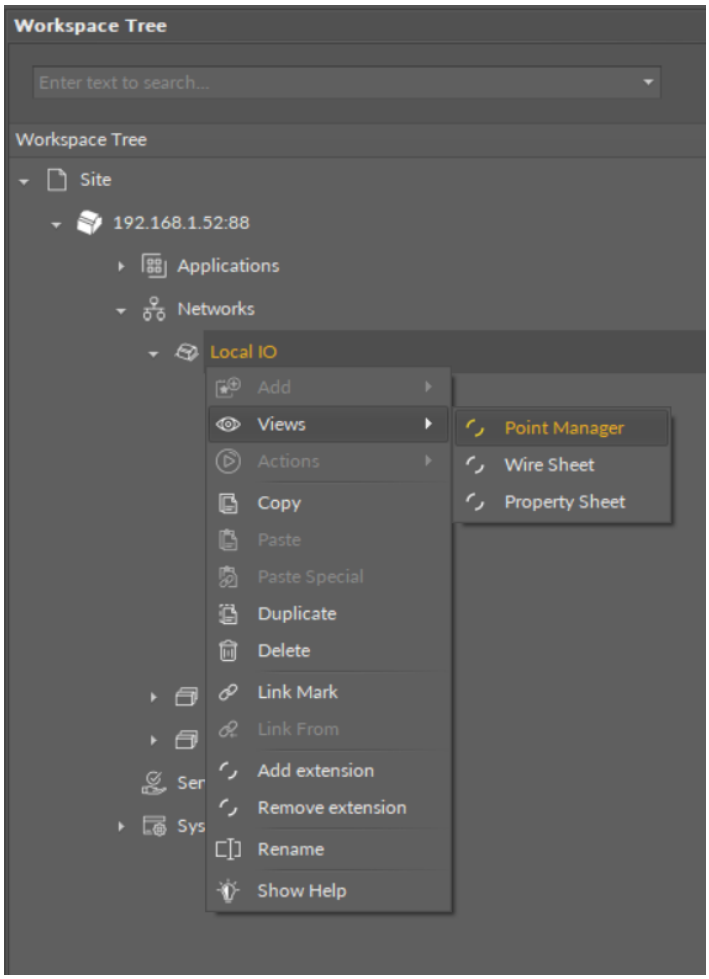


Figure 194. Opening the Point Manager view

Adding I/O Points

The I/O points may be added twofold: dragging and dropping the I/O points to the LocalIO component from the I/O library (in the Device Libraries window), or using a special Add function in the LocalIO Point Manager view available in the bottom right corner. The Add function allows to add any of the I/O points available in the IO library.

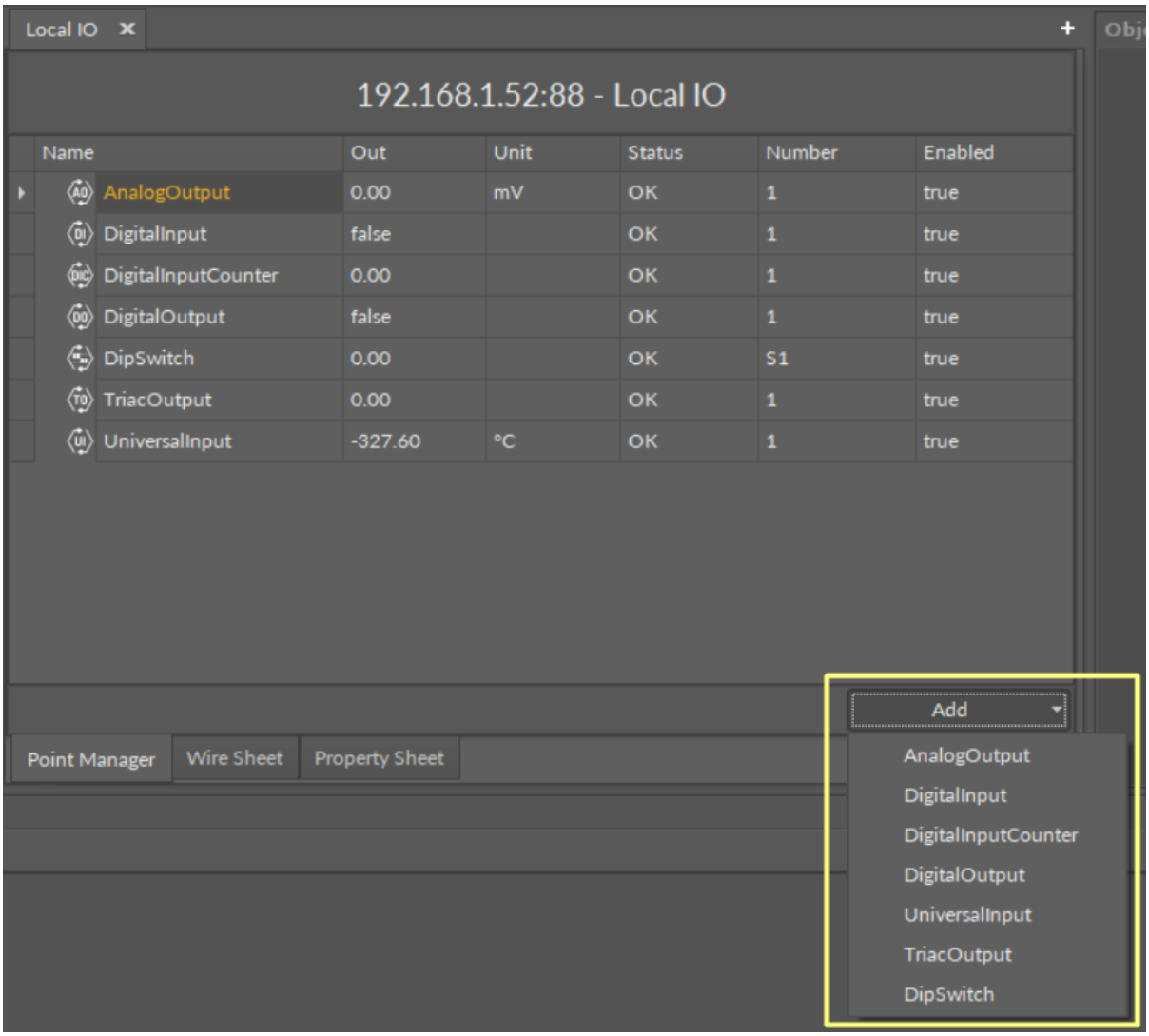


Figure 195. The Add button in the Point Manager

Using this Add button opens the dialog window, which allows to adjust the quantity of I/O points to be added.

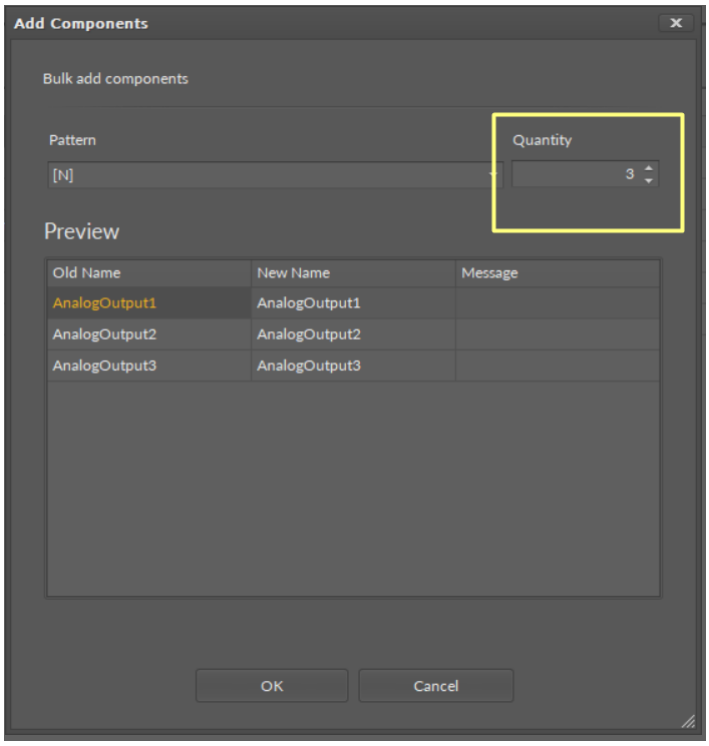


Figure 196. Adjusting quantity of added IO points

Multiediting of Common Slots

The Point Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is available in the Object Properties window, upon selecting one-type components in the Point Manager with Ctrl or Shift keys.

7.5 BACnet

Applicable to OS version 1.0.0.4592

The BACnet library includes components that service the BACnet communication protocol implemented in the device. It allows the device to be recognized and identified in the BACnet network as a BACnet device, and to handle an incoming BACnet communication as client (sending BACnet queries) and server (answering for BACnet queries) device. Both client and server communication is handled via IP. In order to operate properly, the BACnet library components have to be placed in the Networks container (the BACnet component being the superior, service-type component).

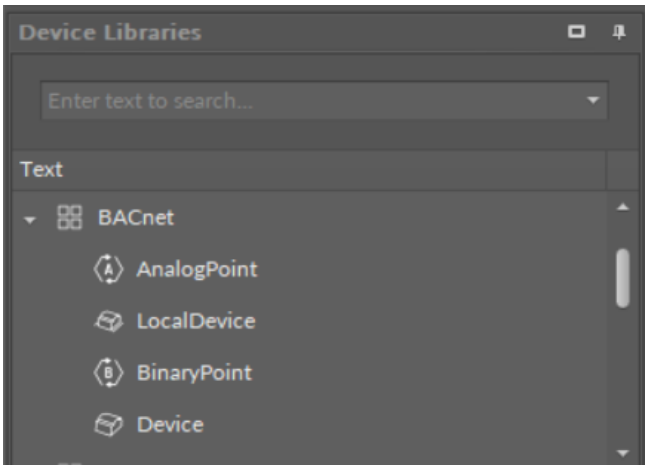


Figure 197. The BACnet library

7.5.1 BACnet Component

The essential feature of the BACnet management in the nano EDGE ENGINE devices is the Autoexposition option–allowing all Data Points added in the project to be exposed to the BACnet network by default. If the Autoexposition slot in the BACnet component is set to true, all Data Points added in the project are automatically exposed on the BACnet network as BACnet objects; however, Data Points may then be individually hidden from the network by manually changing the OnBacnet slot value in a Data Point's BACnet extension (i.e., native extensions, BACnetAnalogPoint and BACnetBinaryPoint, in Analog and Binary Data Points). In case the Autoexposition slot is set to false, all new added Data Points are automatically hidden from the network. They may be exposed to the network manually, changing the Expose slot value in Data Point's BACnet extensions to true.

The BACnet component can be either enabled, or disabled, using the Enabled slot that manually starts or stops the whole BACnet component. Stopping it makes the device and its BACnet objects invisible in the BACnet network.

The BACnet component is automatically added and cannot be removed from the device.

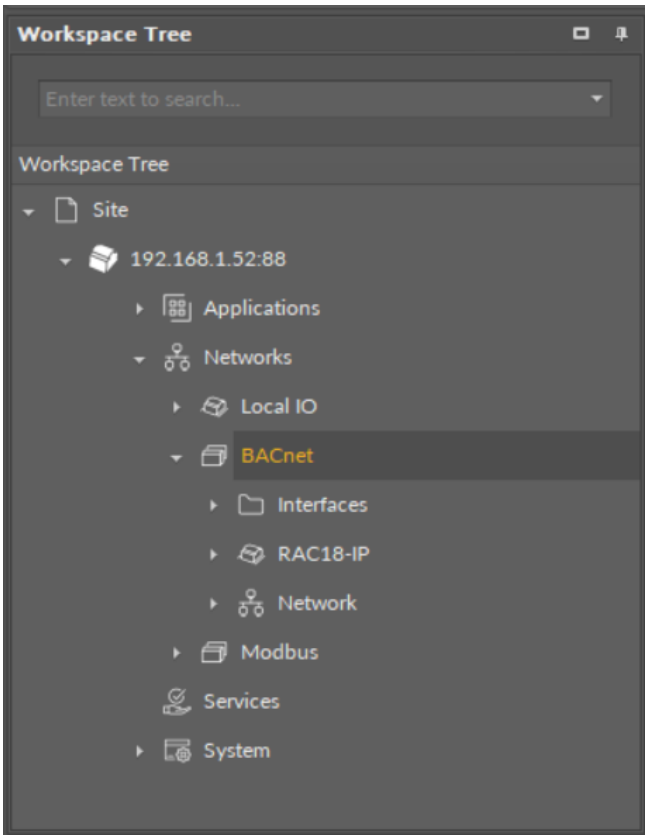


Figure 198. The BACnet component

Slots

The BACnet component has the following slots:

- Status: indicates the current status of the component. If the component works properly, its status is OK; if the Enabled slot has been set to false, the component's status becomes Disabled. The component goes into the Fault status if it cannot be started.
  - Available information: Disabled, Fault, OK.
- Enabled: change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- Send Requests: the number of sent requests;
- Received Responses: the number of received responses;
- Received Error Responses: the number of received error responses;
- Autoexposition: allows to automatically define the exposition of all Data Points added in the project to the BACnet network.
  - Available settings: true (exposed), false (hidden).

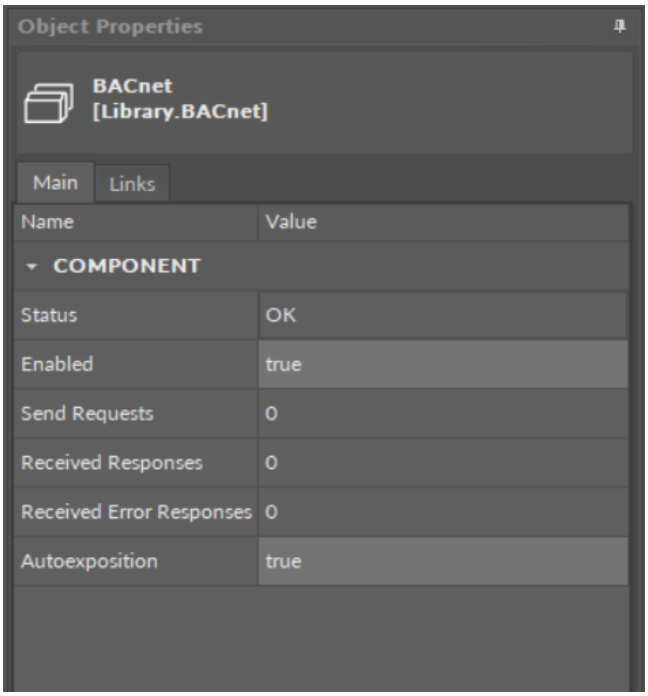


Figure 199. The BACnet component slots

7.5.2 LocalDevice

Applicable to OS version 1.0.0.4592

In order for the Data Point components to be visible in the BACnet network as BACnet objects, a LocalDevice component is essential. It is the component that allows to configure the device as a BACnet Device object that may be exposed to the BACnet network. The component defines the BACnet properties that are necessary for the BACnet network to recognize the BACnet device and communicate with it.

The LocalDevice component is automatically added and placed under its superior component, the BACnet, and it cannot be removed from the device.

In the Device structure, the LocalDevice component is displayed named according to a device's name saved in the Platform component (generated there automatically based on the device's model).

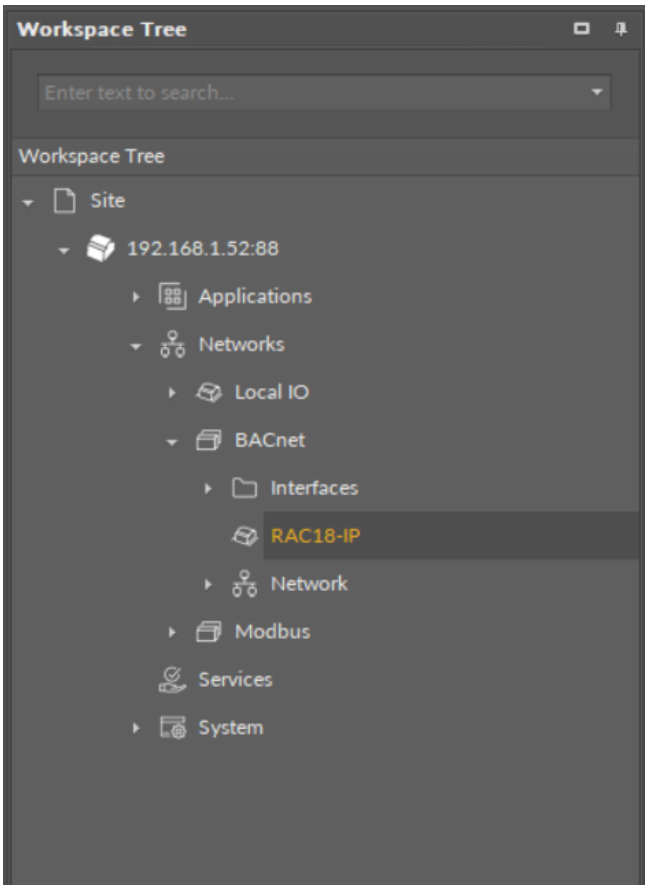


Figure 200. The LocalDevice component

Slots

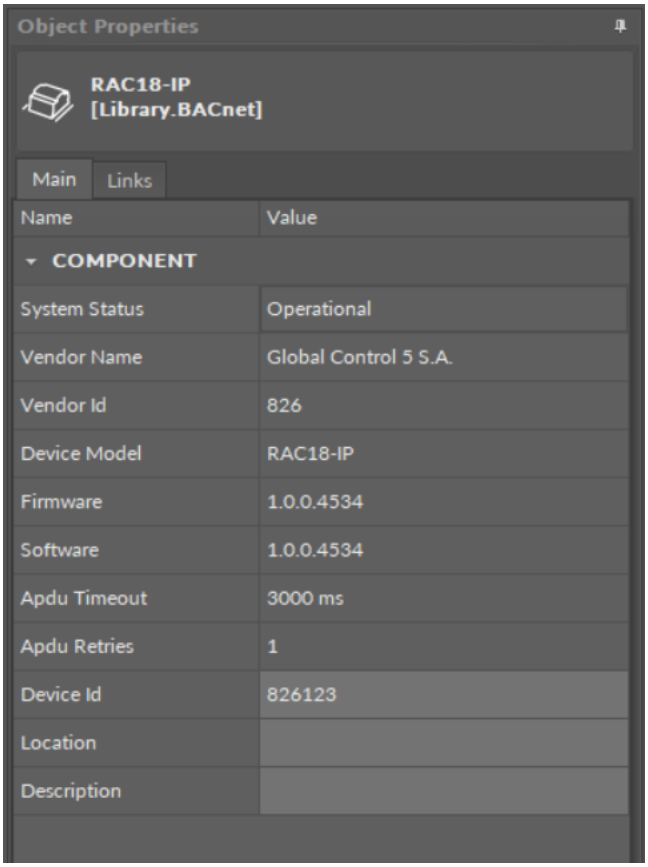


Figure 201. The LocalDevice component's slots

The LocalDevice component has the following slots:

- **System Status:** shows a current status of the BACnet device. If the BACnet device works properly, its status is Operational.
  - Available information: Operational, Operational ReadOnly, Download Required, Download in Progress, Not Operational, Backup in Progress.
- **Vendor Name:** shows a name of the device manufacturer.
- **Vendor Id:** shows an identification number of the device manufacturer that is provided by the BACnet parent organization, in order to enable transferring non-standard messages via the BACnet network. The RAC18-IP manufacturer's vendor id is 826.
- **Device Model:** shows the device model read from the device itself.
- **Firmware:** shows the firmware version currently installed on the device.
- **Software:** shows the OS software version currently installed on the device.
- **Apdu Timeout:** defines the time the system waits before retransmitting an APDU (application protocol data unit) that requires acknowledgment, for which no acknowledgement has been received.
- **Apdu Retries:** Defines the maximum number of APDU retransmissions.
- **Device Id:** represents the device identification within the BACnet network; by default, the value is set to 826123.
- **Location:** allows to enter the device's specific location, in order to facilitate its identification in the user's application.
- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the

user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

7.5.3 Device

Applicable to OS version 1.0.0.4592

The Device component represents a remote BACnet device, which the local BACnet device exchanges data with, as a client device. It allows to configure data for communication with remote server devices, add them to the network, connected on the IP port, and set their unique device IDs. If there is no response from the server device, the Device component goes into the Down status.

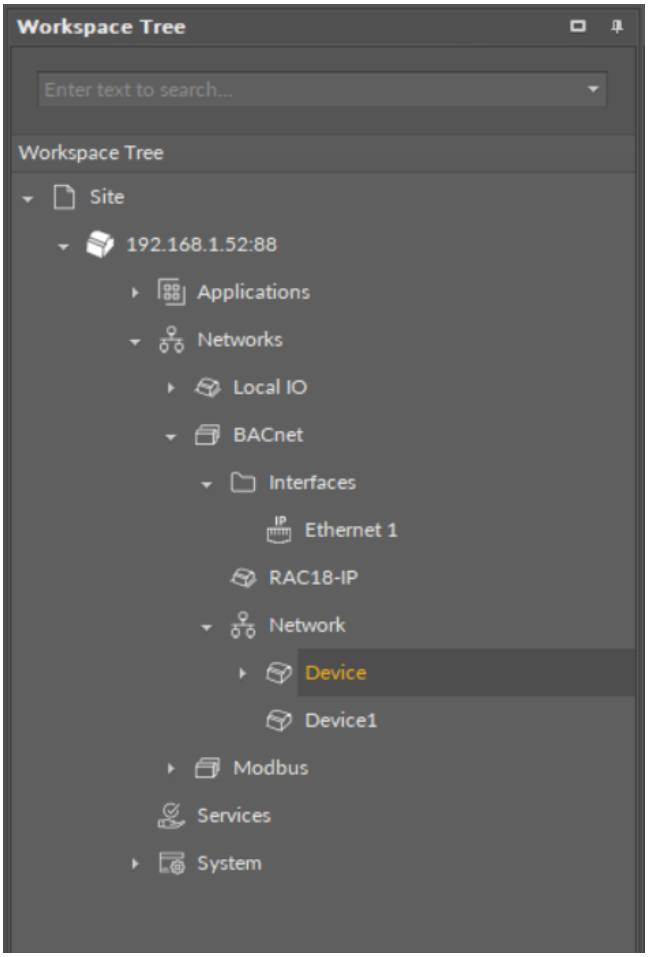


Figure 202. The Device component

Slots

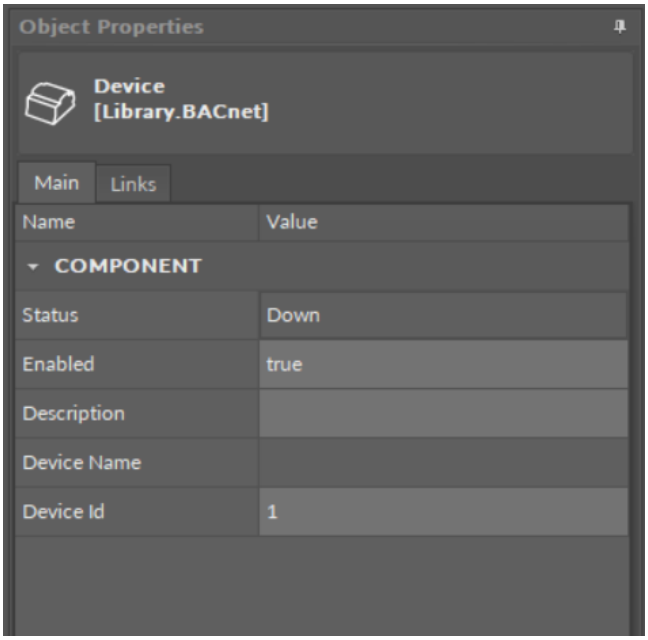


Figure 203. The Device components slots

The Device component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false or null. The component's status is Fault, once the Device Id slot is null. If there is no response from the addressed device, the component goes into the Down status.
  - Available information: Disabled, Fault, Down, OK.
- **Enabled:** change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

Note: The description is effectively added only if the remote device allows it–the description is not added internally in the RAC18-IP for the remote device, but it is sent directly to the remote device.

- **Device Name:** shows the name from the remote BACnet device identified in the Device Id slot;
- **Device Id:** allows to set the unique ID of the remote device in the BACnet network.

> **Worth to Notice**
>
> If the Network component is disabled, the Device component is disabled too.

7.5.4 AnalogPoint

Applicable to OS version 1.0.0.4592

The AnalogPoint component is a network point class component, which allows to read and write numeric values to a defined BACnet object in the remote device (the BACnet Analog Value object).

Note: Before using the component, make sure that its object ID is assigned and its Status is OK.

The AnalogPoint component includes an action that allows to set its Out value in case no Data Point class component is linked to it; the value is sent to the BACnet object with no priority.

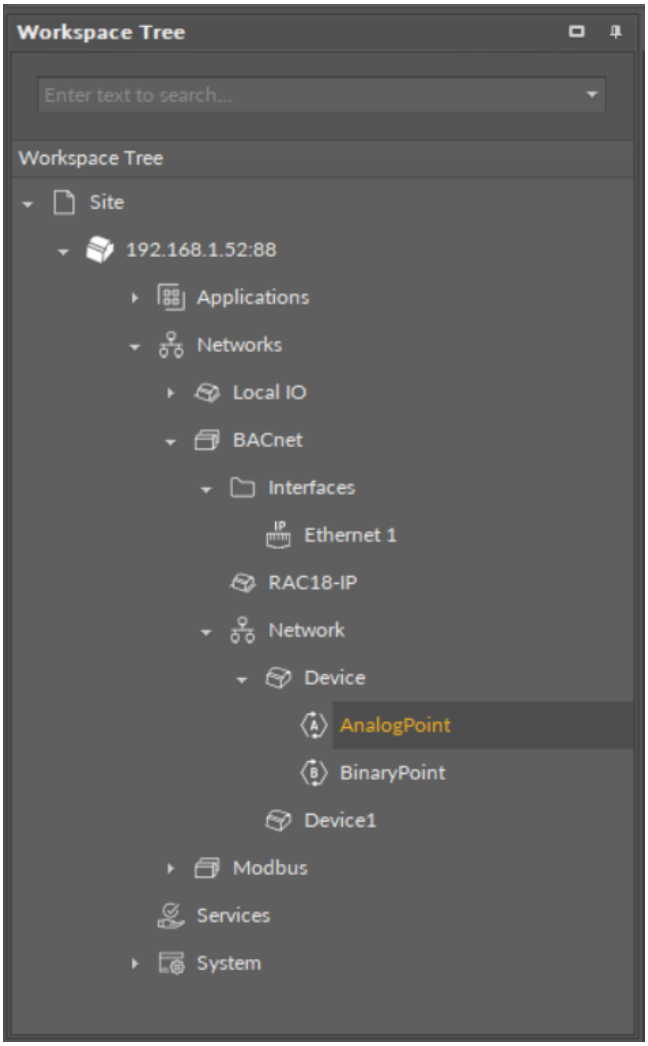


Figure 204. The AnalogPoint component

Slots

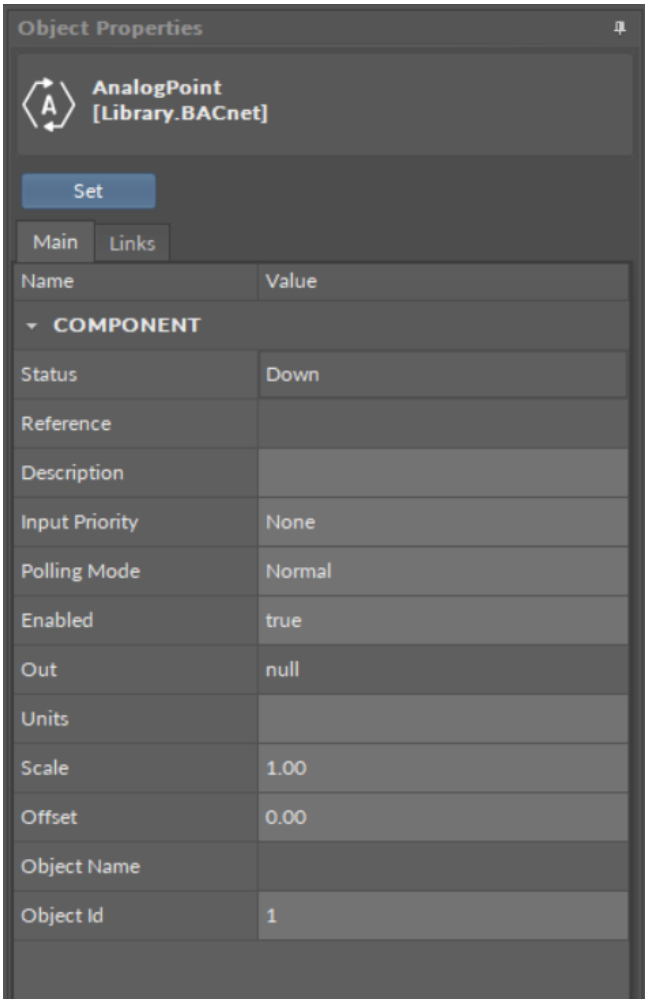


Figure 205. The AnalogPoint component slots

The AnalogPoint component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Object Id slot is null. If there is no response from the addressed point, the component goes into the Down status.
  - Available information: Disabled, Fault, Down, OK.
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

Note: The description is effectively added only if the point allows it–the description is not added internally in the RAC18-IP for the remote point, but it is sent directly to the point.

- InputPriority: allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; none priority is set by default.
    - Available settings: none, 1-16.
- Polling Mode: allows to set the frequency of sending polling requests for the point's value to remote devices—the polling mode is automatically set to normal;
    - Available settings: fast, normal, slow;
- Enabled: change of the slot's value enables or disables the component.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- Out: a real value read from or written to the object of the remote device.

Note: If the component's Status is fault (e.g., an invalid value in the Object Id slot), the Out value is null.

- Units: defines a unit of the Out slot value; no unit is set by default;
- Scale: sets a fixed scaling factor for output linearization; the written value is calculated according to the inverse linear function formula (x=(y-b)/a), and the calculated value is read to the Out slot according to the linear function (y=ax+b); the Scale slot sets the a value of the formula; incorrect scaling of the results in the calculated value (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input and on the Out slot. If the scaling is correct, the values are identical;
- Offset: sets a fixed offset value to the output value; the value is calculated according to the inverse linear function formula (x=(y-b)/a) and read to the Out slot according to the linear function (y=ax+b); the Offset slot sets the b value of the formula; incorrect scaling of the results in the calculated value (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input and on the Out slot. If the scaling is correct, the values are identical;
- Object Name: displays the BACnet remote device's point name;
- Object Id: allows setting an address of the BACnet object in the remote device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.

Note: Here, the ObjectID regards the BACnet Analog Value object.

Find out more about the Reference linking.

Action

- Set: sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

Note: The value is sent to the BACnet object with no priority.

7.5.5 BinaryPoint

Applicable to OS version 1.0.0.4592

The BinaryPoint component is a network point class component, which allows to read and write binary values to a defined BACnet Binary Value object in the remote device.

Note: Before using the component, make sure that its object ID is assigned and its Status is OK.

The BinaryPoint component includes an action that allows to set its Out value in case no Data Point class component is linked to it; the value is sent to the BACnet object with no priority.

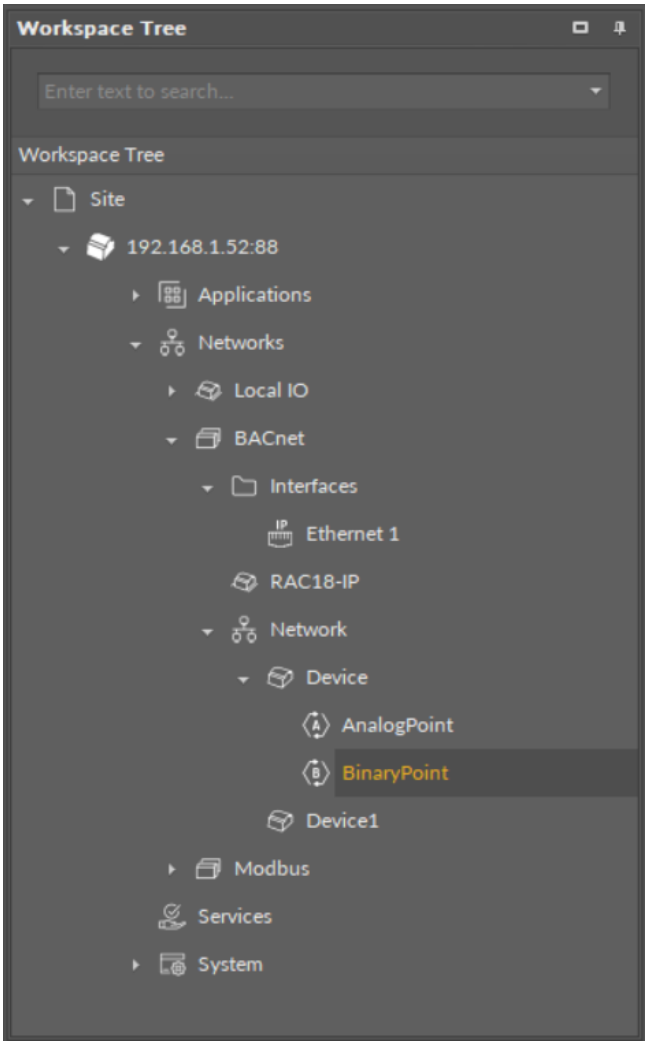


Figure 206. The BinaryPoint component

Slots

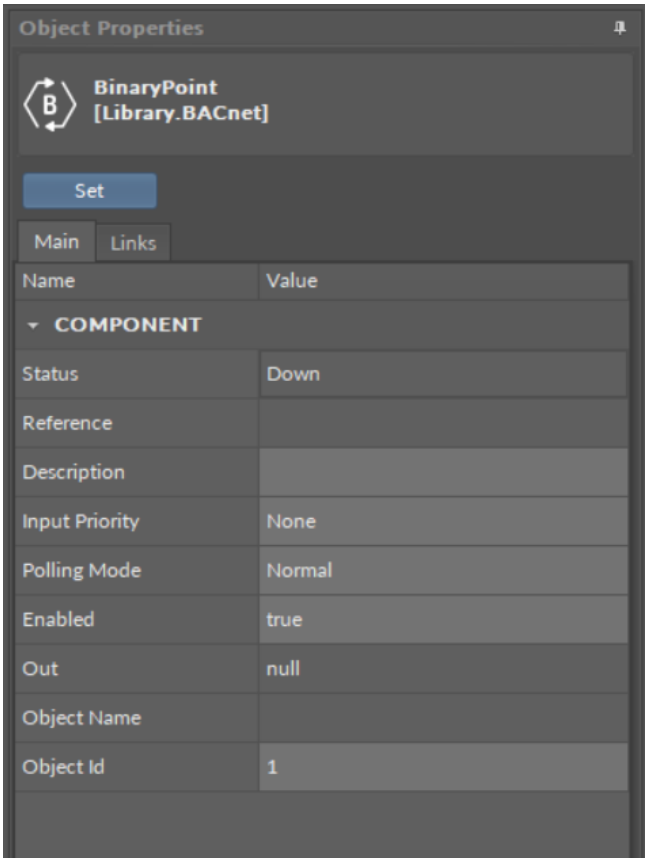


Figure 207. The BinaryPoint component slots

The BinaryPoint component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Object Id slot is null. If there is no response from the addressed point, the component goes into the Down status.
    - Available information: Disabled, Fault, Down, OK.
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

Note: The description is effectively added only if the point allows it–the description is not added internally in the RACIP-18 for the remote point, but it is sent directly to the point.

- **InputPriority:** allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; none priority is set by default.
  - Available settings: none, 1-16.
- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value to remote devices—the polling mode is automatically set to normal;
  - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** a real value read from or written to the object of the remote device.

Note: If the component's Status is fault (e.g., an invalid value in the Object Id slot), the Out value is null.

- **Object Name:** displays the BACnet remote device's point name;
- **Object Id:** allows setting an address of the BACnet object in the remote device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.

Note: Here, the Object Id regards the BACnet Binary Value object.

Find out more about the Reference linking.

Action

- **Set:** sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

Note: The value is sent to the BACnet object with no priority.

7.5.6 Quick Start-up of BACnet

In order to launch the BACnet communication properly, it is necessary to follow these steps:

Step 1: Having the device added correctly in the iC Tool, expand the Networks container, and go to the BACnet component.

Step 2: The Interfaces component with the Ethernet component are added by default. Go to the Ethernet component if changing the default port number is necessary.

Step 3: The RAC18-IP component (LocalDevice) is also added by default. In order to configure the BACnet server device, go the the component.

Step 4: In order to configure the BACnet client, the Network and Device components need to be added, along with AnalogPoint or BinaryPoint components, as necessary. Go to the Device Libraries and expand the Core library for the Network component and BACnet library for other components. Choose components to be added—components may be added one by one or grouped in one selection. Drag the selected component(s) and drop it(them) under the BACnet component in the Network container.

Worth to Notice:

Please remember that the components' hierarchy needs to be preserved here: the Network component has to be located under the BACnet component, the Device component under the Network component, and the AnalogPoint/BinaryPoint components have to be placed under the Device component.

If the superior component is selected in the Workspace Tree window and its special view (Network Manager/Device Manager/Point Manager) is opened in the main screen, the Device Libraries shows only the components that can be added directly under it. For example, if the BACnet component is selected in the Workspace Tree window, the Device Libraries shows only the Network component in the Core library.

Step 5: Go to each added component, open its Property Sheet (or go the the BACnet Property Sheet and expand each component), and set their parameters (enable, device Id, etc.).

Worth to Notice:

In order to facilitate working with BACnet component, special views have been developed:

- Network Manager, available in the BACnet component;
- Data Point Manager, available in the LocalDevice component;
- Device Manager, available in the Network component;
- Point Manager, available in the Device component.

Ready to Use: Configured components are ready for proper BACnet communication.

Network Manager for BACnet

The Network Manager view is available for the BACnet component. It lists all BACnet networks configured on the device's ports. The Network Manager view shows the statuses, ports (which the network is configured on), and enabled or disabled states of the the network. Once the network, listed in the Network Manager, is double-clicked, the respective Network component is opened.

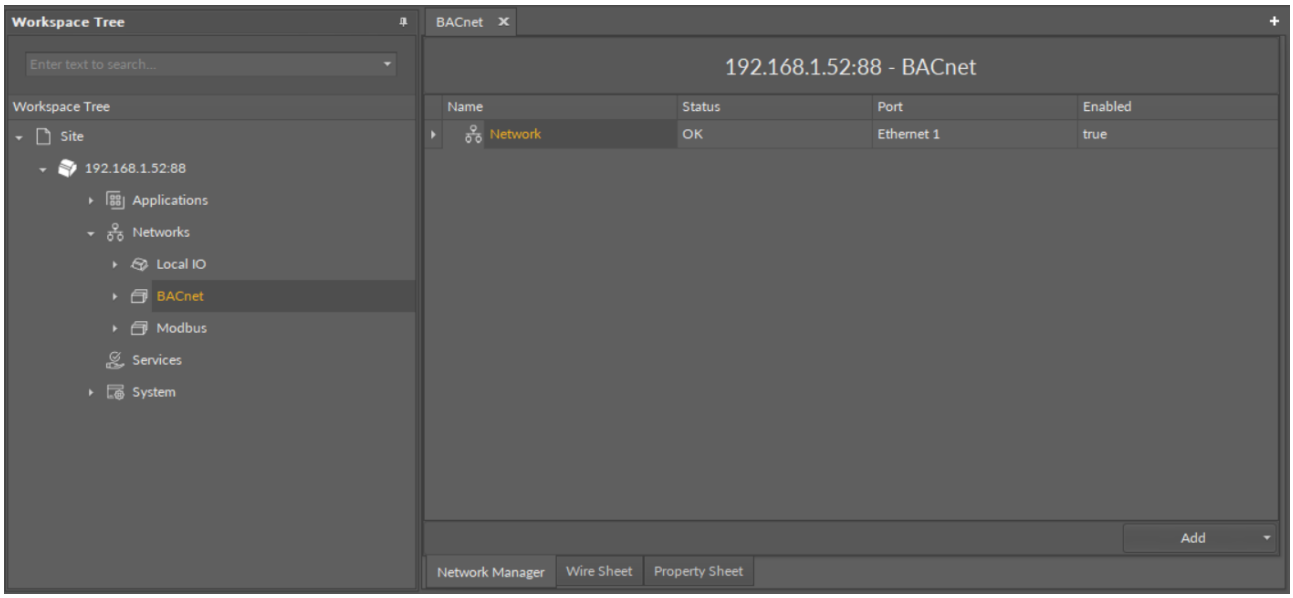


Figure 208. The Network Manager for BACnet

Opening Network Manager

The Network Manager view is accessible from the context menu of the BACnet component. It is also automatically opened if the BACnet component is double-clicked in the Workspace Tree window.

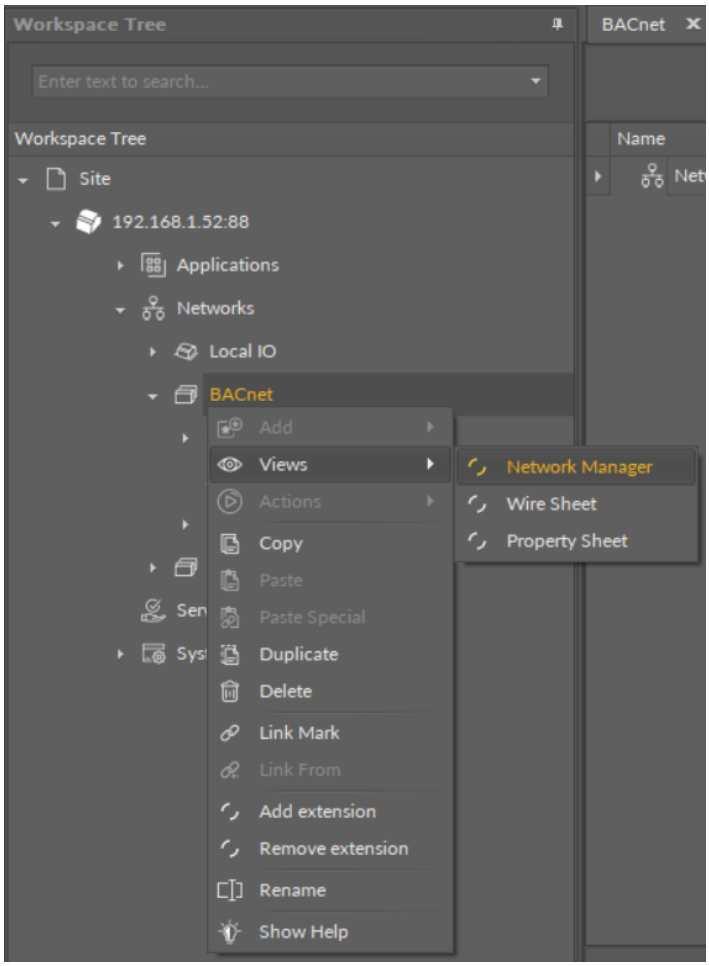


Figure 209. Opening the Network Manager

Adding BACnet Networks

The networks may be added to the Network Manager twofold: dragging and dropping the Network component to the BACnet component from the Core library (in the Device Libraries window), or using a special Add function in the Network Manager view available in the bottom right corner.

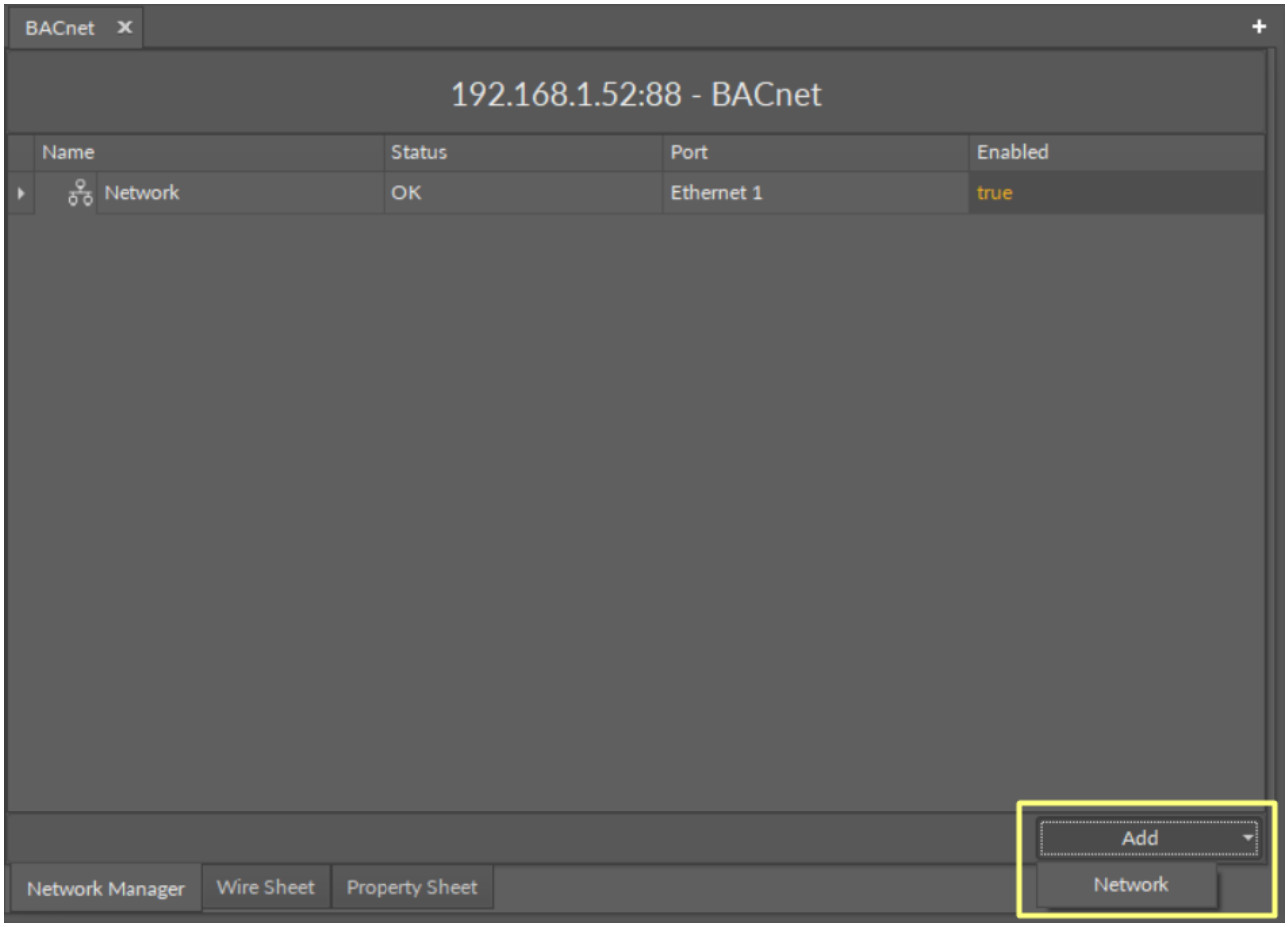


Figure 210. The Add button

Using this Add button opens the dialog window, which allows to adjust the quantity of networks to be added.

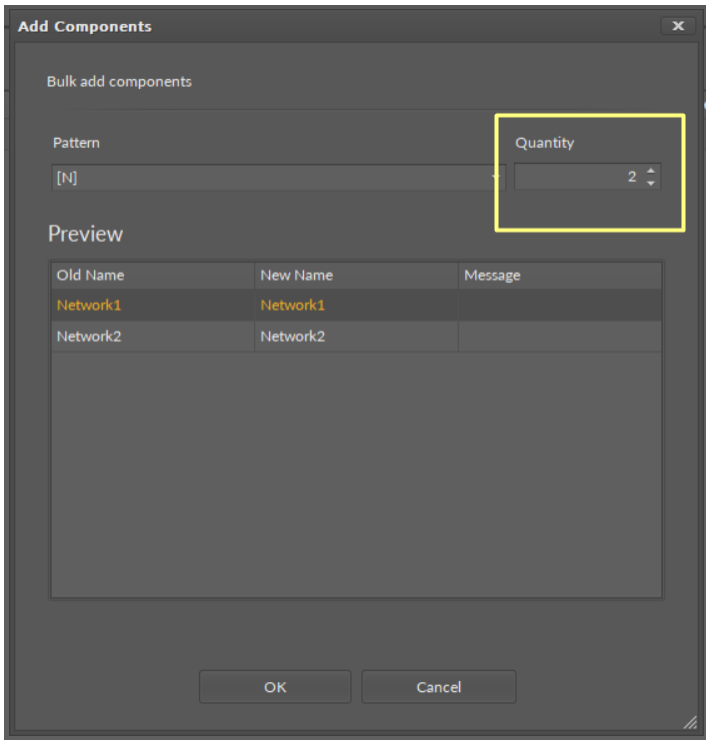


Figure 211. The dialog window

Multiediting of Common Slots

The Network Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is available in the Object Properties window, upon selecting one-type components in the Network Manager with Ctrl or Shift keys.

Device Manager for BACnet

The Device Manager view is available for the BACnet Network component. It lists all devices added to the configured BACnet network. The Device Manager view shows the statuses, BACnet device names and IDs, and enabled or disabled states of the devices in the network. Once the device listed in the Device Manager is double clicked, the respective Device component is opened.

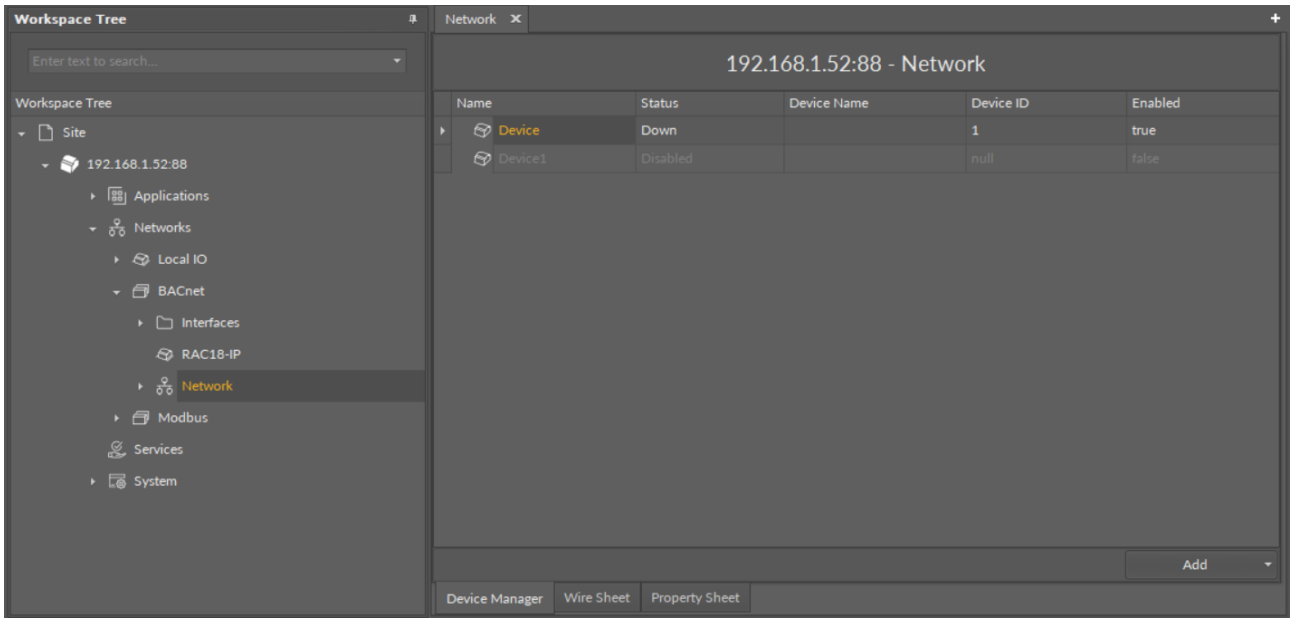


Figure 212. The Device Manager

Opening Device Manager

The Device Manager view is accessible from the context menu of the Network component. It is also automatically opened if the Network component is double-clicked in the Workspace Tree window.

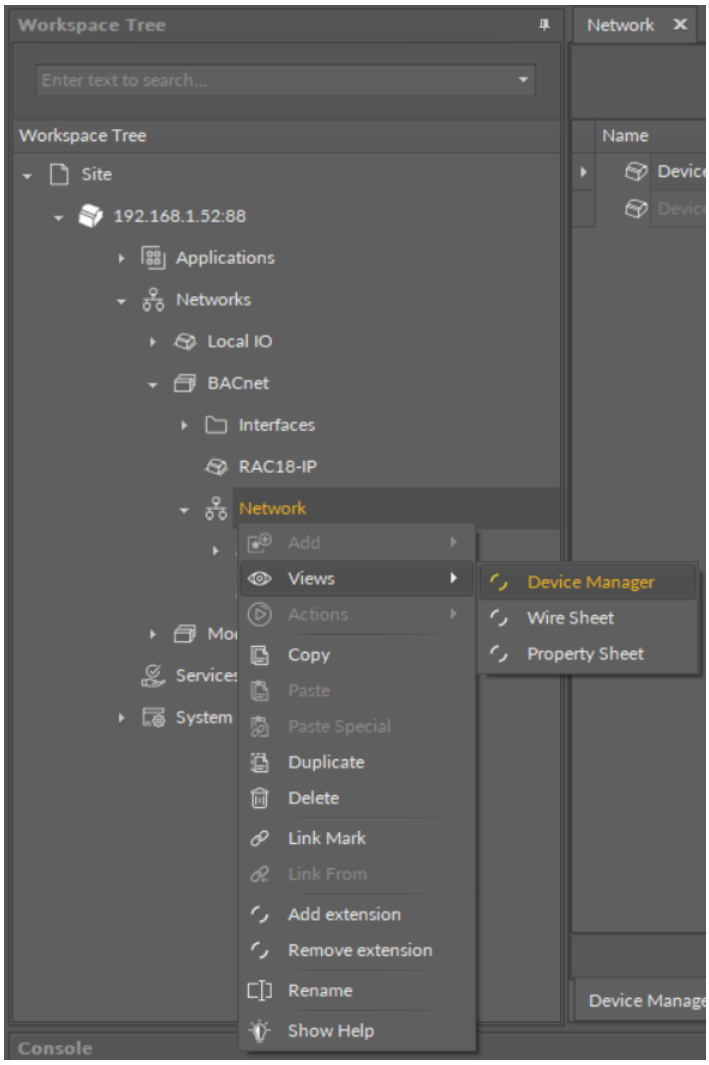


Figure 213. Opening the Device Manager

Adding BACnet Devices

The devices may be added to the BACnet network twofold: dragging and dropping the Device component to the Network component from the BACnet library (in the Device Libraries window), or using a special Add function in the Device Manager view available in the bottom right corner.

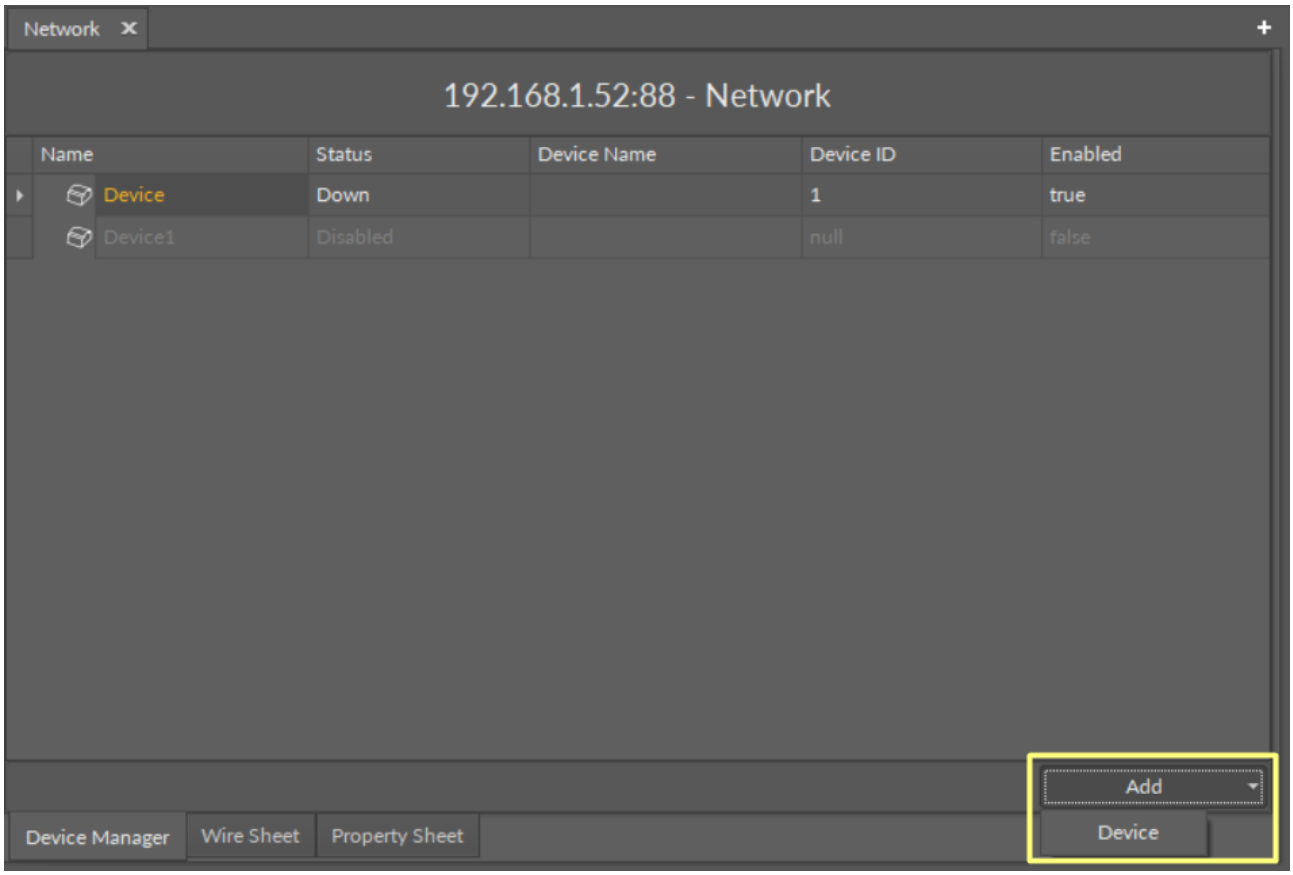


Figure 214. The Add button in the Device Manager

Using this Add button opens the dialog window, which allows to adjust the quantity of devices to be added.

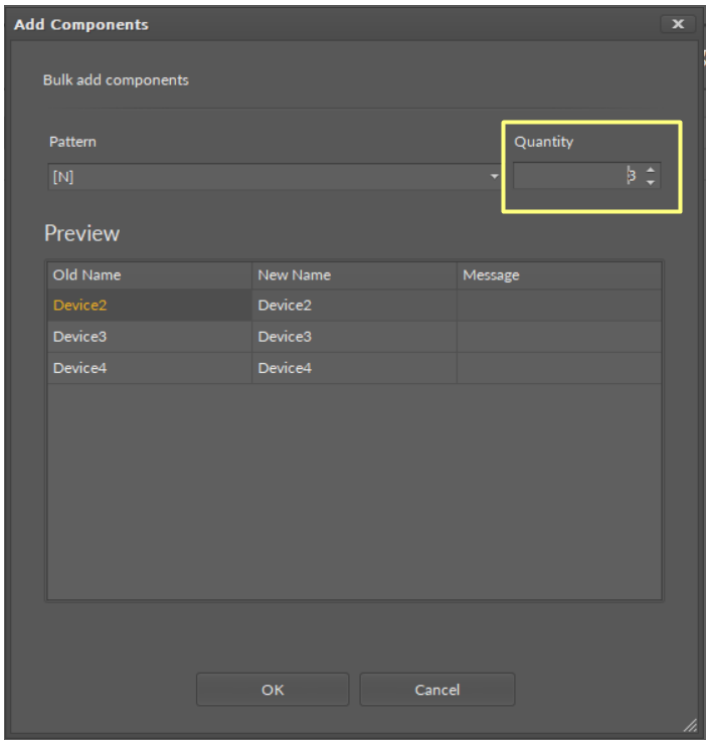


Figure 215. The dialog window

Multiediting of Common Slots

The Device Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is available in the Object Properties window, upon selecting one-type components in the Device Manager with Ctrl or Shift keys.

Point Manager for BACnet

The Point Manager view is available for each device added to the BACnet network. It lists all BACnet Points added to the Device component, and shows their Out slot value, status, object name and ID, polling mode, and enabled or disabled state.

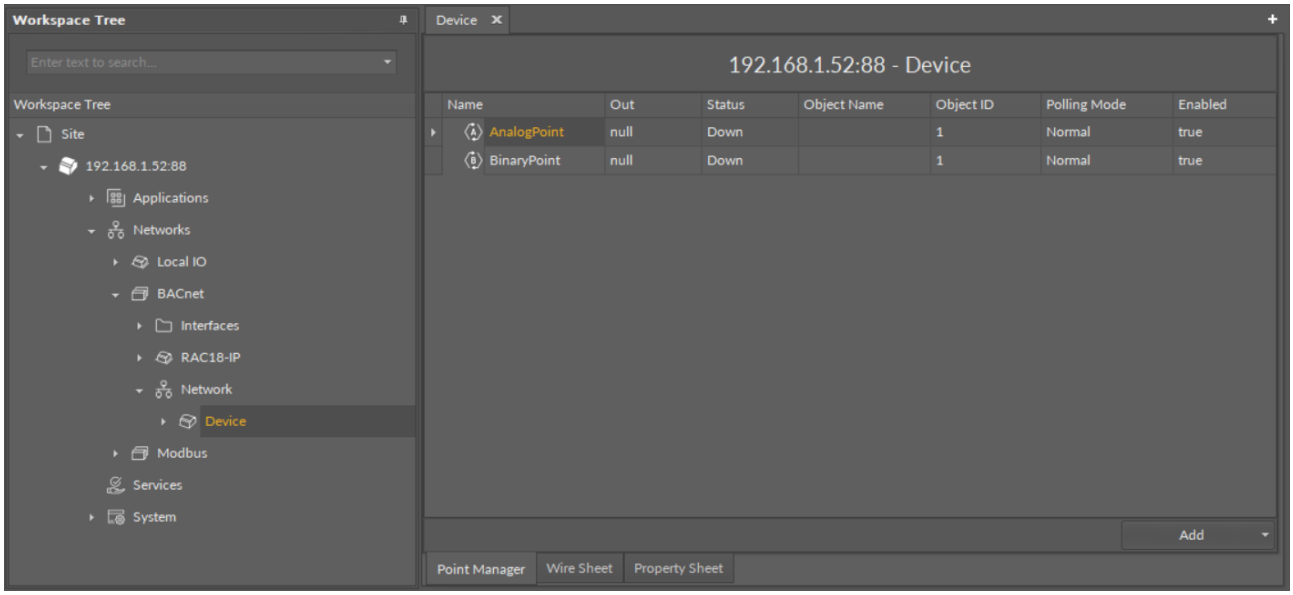


Figure 216. The Point Manager

Opening the Point Manager

The Point Manager view is accessible from the context menu of the Device component. It is also automatically opened if the Device component is double-clicked in the Workspace Tree window.

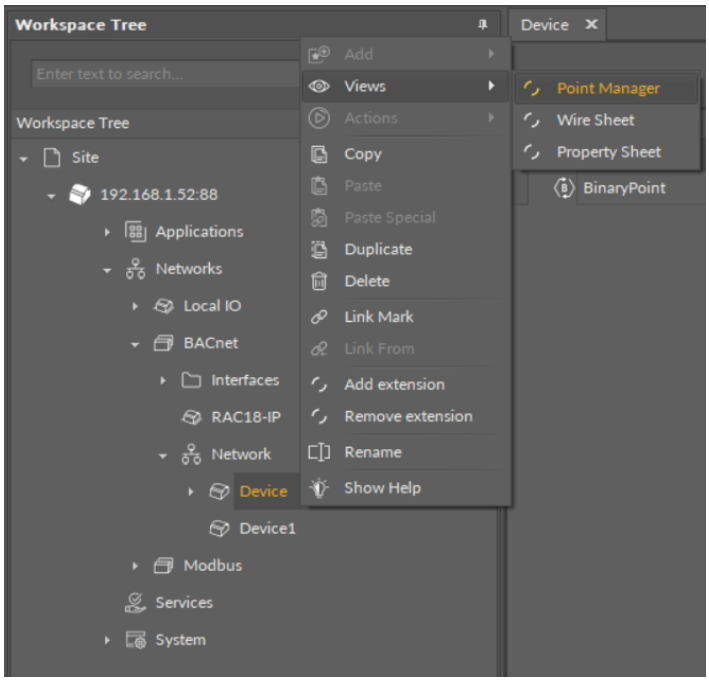


Figure 217. Opening the Point Manager

Adding BACnet Points

The BACnet points may be added to the device twofold: dragging and dropping the BACnet points to the Device component from the BACnet library (in the Device Libraries window), or using a special Add function in the Point Manager view available in the bottom right corner. The Add function allows to add any of the BACnet points available in the BACnet library.

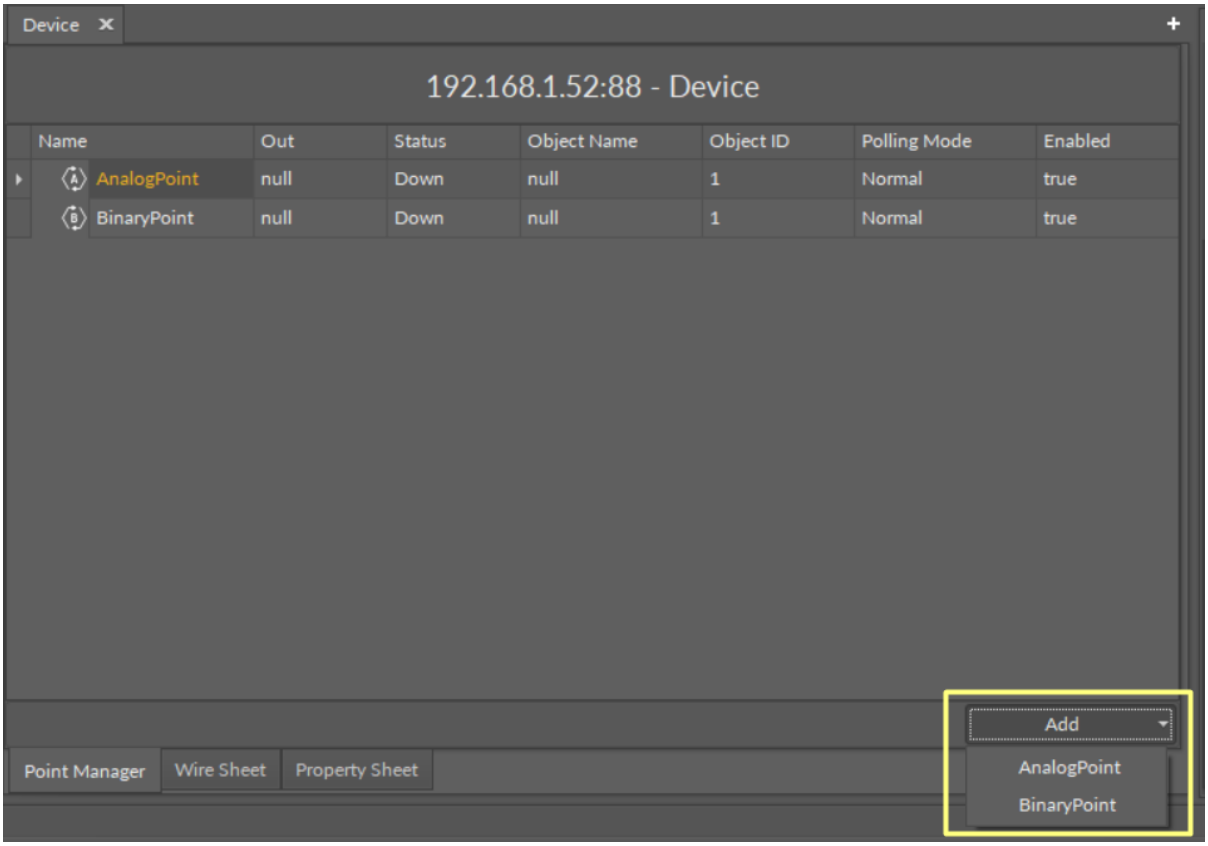


Figure 218. The Add button

Using this Add button opens the dialog window, which allows to adjust the quantity of BACnet Points to be added.

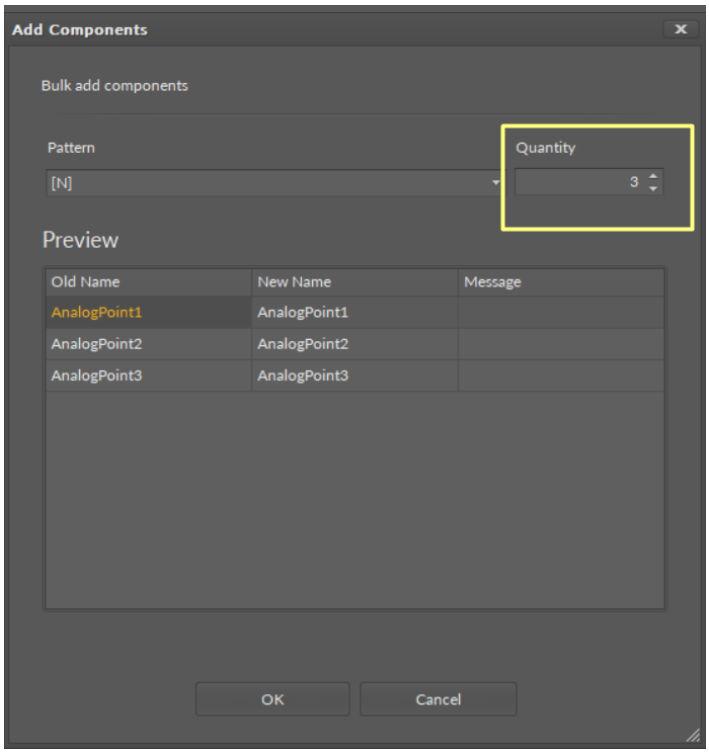


Figure 219. The dialog window

Multiediting of Common Slots

The Point Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is available in the Object Properties window, upon selecting one-type components in the Point Manager with Ctrl or Shift keys.

7.6 Modbus

Applicable to OS version 1.0.0.4592

The Modbus library includes components that service the Modbus RTU communication protocol implemented in the device. It allows the device to be recognized and identified in the Modbus network, and to work as a Modbus client device (sending requests to Modbus server devices for their registers values). The Modbus RTU communication is handled via the RS485 port. In order to operate properly, the Modbus library components have to be placed in the Networks container (the Modbus component being the superior, service-type component).

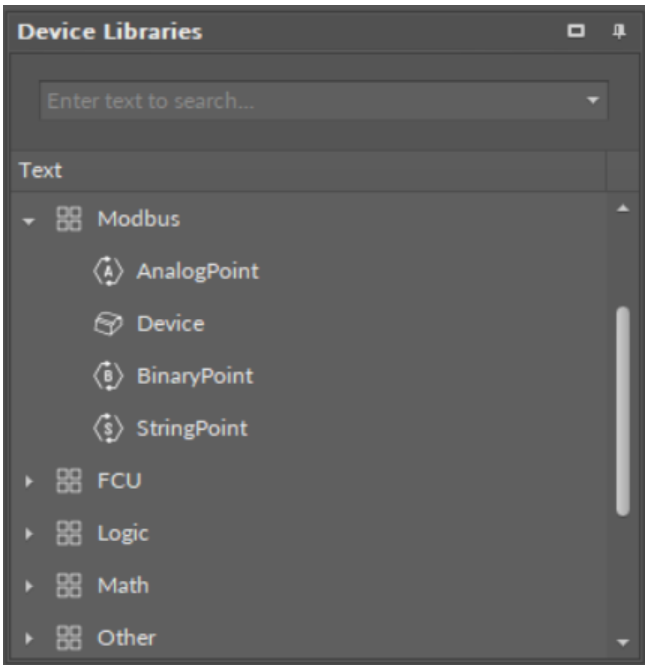


Figure 220. The Modbus library

7.6.1 Modbus Component

The Modbus component is the superior component allowing to manage the device's Modbus communication. It allows the device to operate as a Modbus client device. The Modbus communication in the RAC18-IP device is the Modbus RTU protocol over the RS485 communication port.

The Modbus component can be either enabled, or disabled, using the Enabled that manually starts or stops the whole Modbus component.

The Modbus component is automatically added and cannot be removed from the device.

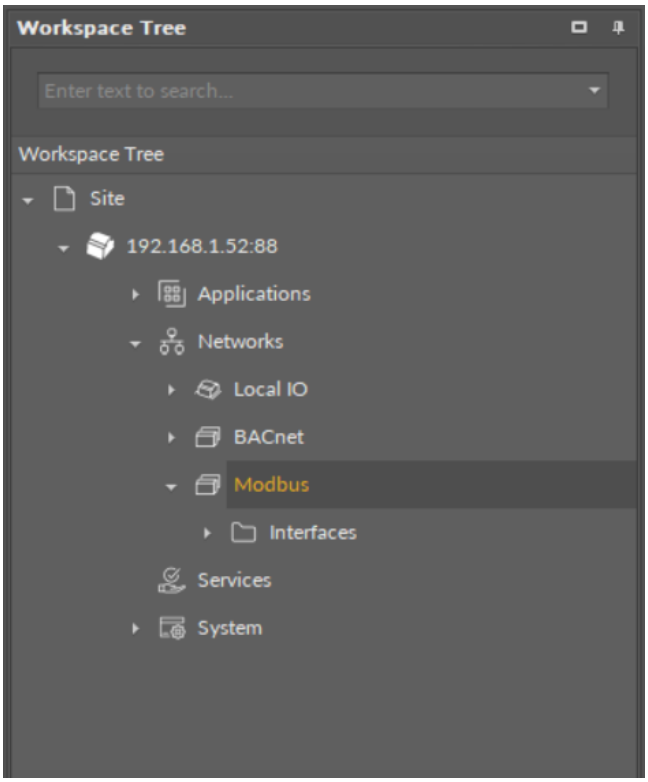


Figure 221. The Modbus component

Slots

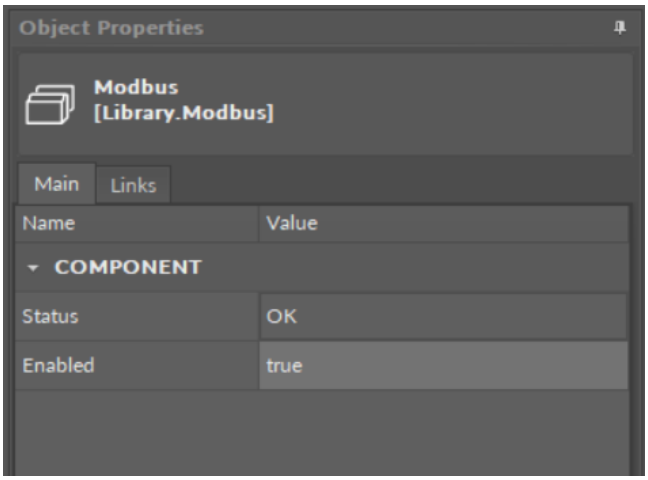


Figure 222. The Modbus component slots

The Modbus component has the following slots:

- Status: indicates the current status of the component. If the component works properly, its status is OK; if the Enabled slot has been set to false, the components's status becomes Disabled. The component goes into the Fault status if it cannot be started.
  - Available information: Disabled, Fault, OK.
- Enabled: change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

7.6.2 Device

Applicable to OS version 1.0.0.4592

The Device component is a component to configure data for communication with a server device in the Modbus network. The RAC18-IP controller acts as a client device in the Modbus network, and the Device component allows to add remote devices connected on the RS485 port, set their unique addresses and ping registers. The ping register number defines the remote device register, which answers the master request–if there is no response from the slave device, the Device component goes into the Down status.

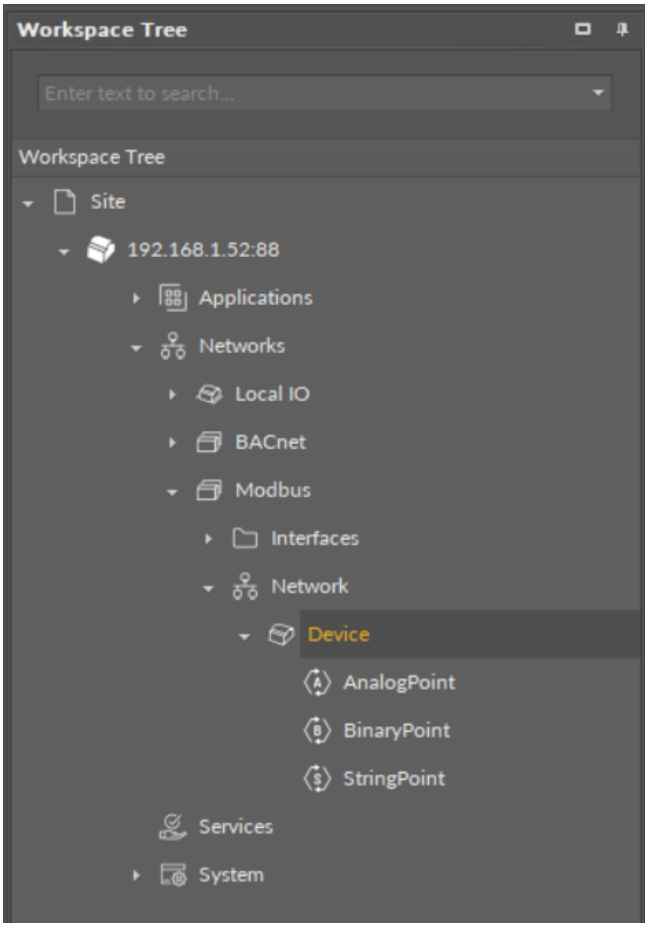


Figure 223. The Device component

Slots

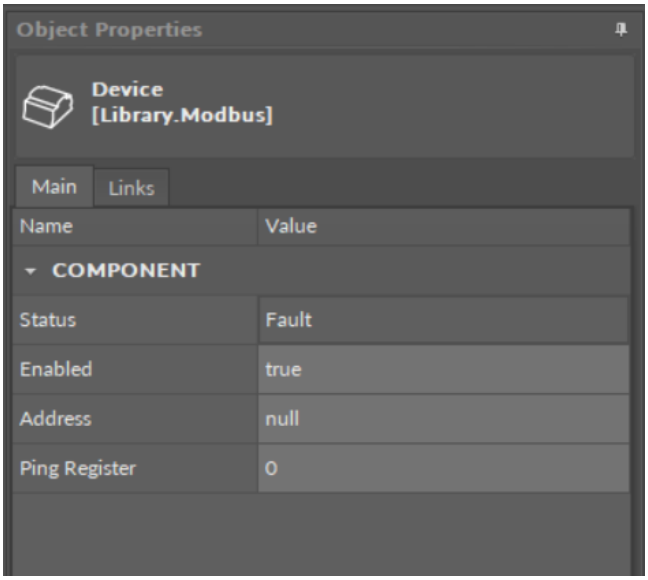


Figure 224. The Device component slots

The Device component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null. If there is no response from the addressed device, the component goes into the Down status.
  - Available information: Disabled, Fault, Down, OK.
- **Enabled:** change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Address:** allows to set the unique address of the remote device in the Modbus network;
- **Ping Register:** allows to set the number of the remote device's register (in decimal format), which will provide answers for the master device's Modbus requests.

> **Worth to Notice**
>
> If the Network component is disabled, the Device component is disabled too.

7.6.3 AnalogPoint

Applicable to OS version 1.0.0.4592

The AnalogPoint component is a network point class component, which allows to read and write numeric values to a defined Modbus holding register in the remote device.

Note: Before using the component, make sure that its register address is assigned and its Status is OK.

The AnalogPoint component includes an action that allows to set its Out value in case no Data Point class component is linked to it.

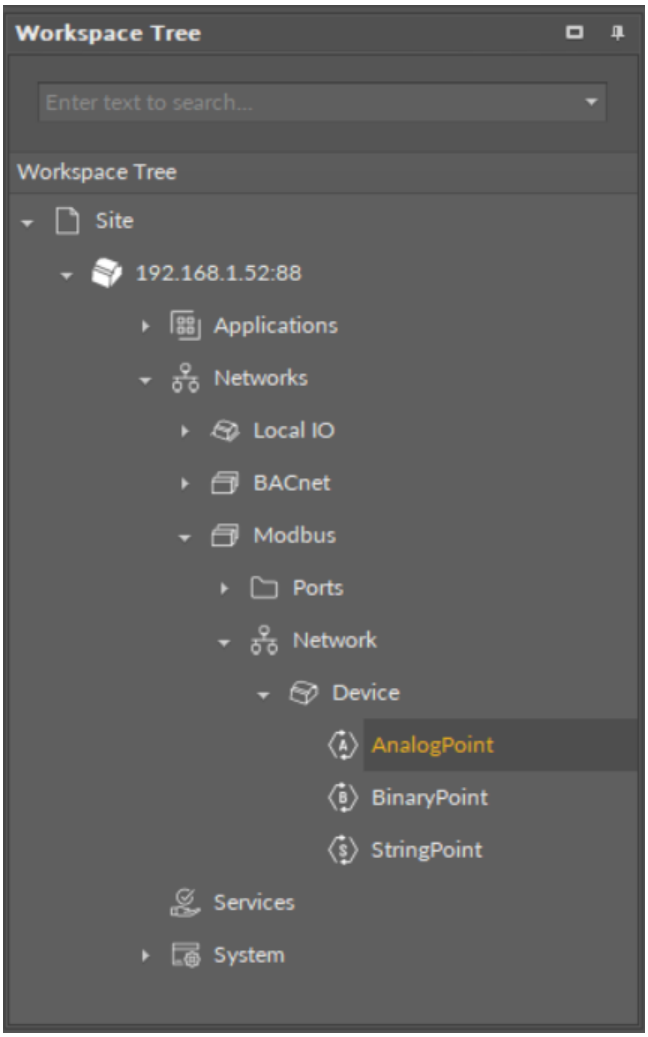


Figure 225. The AnalogPoint component

Slots

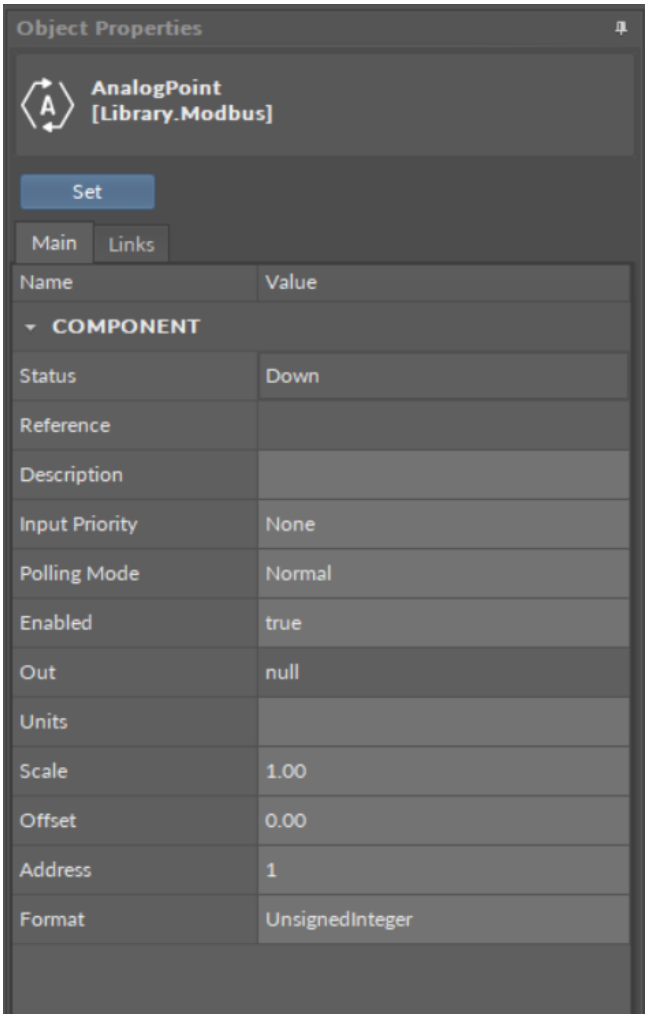


Figure 226. The AnalogPoint component slots

The AnalogPoint component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null. If there is no response from the addressed point, the component goes into the Down status.
  - Available information: Disabled, Fault, Down, OK.
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

Note: The description is effectively added only if the point allows it–the description is not added internally in the RAC18-IP for the remote point, but it is sent directly to the point.

- InputPriority: allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; none priority is set by default.
  - Available settings: none, 1-16.
- Polling Mode: allows to set the frequency of sending polling requests for the point's value to remote devices—by default, the polling mode is set to normal;
  - Available settings: fast, normal, slow;
- Enabled: change of the slot's value enables or disables the component.
  - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- Out: a real value read from or written to the register of the remote device.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- Units: defines a unit of the Out slot value; no unit is set by default;
- Scale: sets a fixed scaling factor for output linearization; the written value is calculated according to the inverse linear function formula (x=(y-b)/a), and the calculated value is read to the Out slot according to the linear function (y=ax+b); the Scale slot sets the a value of the formula; incorrect scaling of the results in the calculated value (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input and on the Out slot. If the scaling is correct, the values are identical;
- Offset: sets a fixed offset value to the output value; the value is calculated according to the inverse linear function formula (x=(y-b)/a) and read to the Out slot according to the linear function (y=ax+b); the Offset slot sets the b value of the formula; incorrect scaling of the results in the calculated value (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input and on the Out slot. If the scaling is correct, the values are identical;
- Address: allows setting an address of the Modbus register in the remote device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.

Note: The Modbus register address is set in decimal format.

- Format: allows to set the format of the numeric value read or written to the remote device's register;
  - Available settings: UnsignedInteger, SignedInteger (including values less than 0), Long (for 32-bit values, reads two subsequent registers).

Find out more about the Reference linking.

Action

- Set: sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

7.6.4 BinaryPoint

Applicable to OS version 1.0.0.4592

The BinaryPoint component is a network point class component, which allows to read and write Boolean values to a defined Modbus coil in the remote device.

Note: Before using the component, make sure that its register address is assigned and its Status is OK.

The BinaryPoint component includes an action that allows to set its Out value in case no Data Point class component is linked to it.

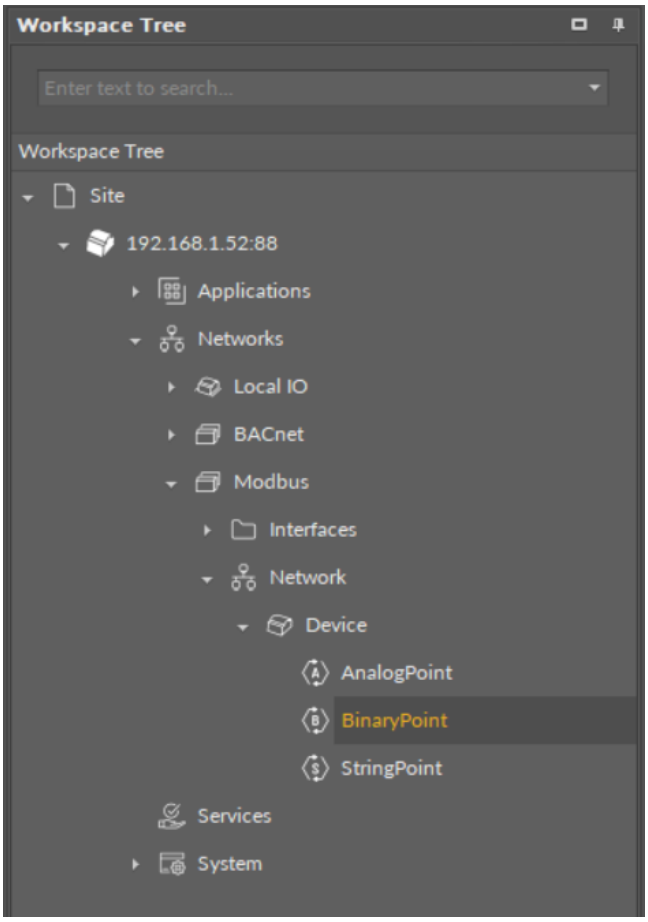


Figure 227. The BinaryPoint component

Slots

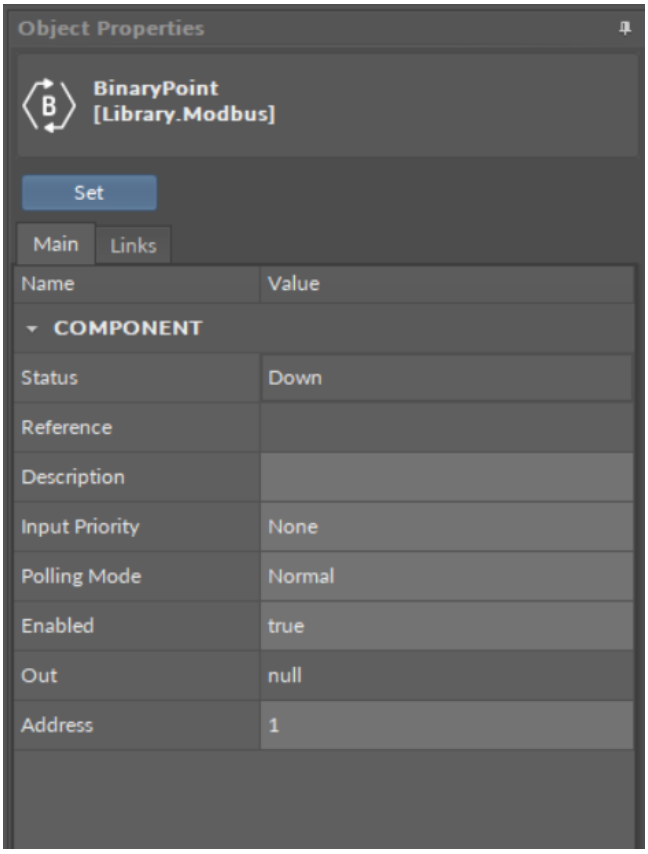


Figure 228. The BinaryPoint component slots

The BinaryPoint component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null. If there is no response from the addressed point, the component goes into the Down status.
  - Available information: Disabled, Fault, Down, OK.
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

Note: The description is effectively added only if the point allows it–the description is not added internally in the RACIP-18 for the remote point, but it is sent directly to the point.

- **InputPriority:** allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; none priority is set by default.
    - Available settings: none, 1-16.
- **Polling Mode:** allows to set the frequency of sending polling requests for the point's value to remote devices—by default, the polling mode is set to normal;
    - Available settings: fast, normal, slow;
- **Enabled:** change of the slot's value enables or disables the component.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- **Out:** a real value read from or written to the register of the remote device.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- **Address:** allows setting an address of the Modbus register in the remote device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.

Note: The Modbus register address is set in decimal format.

Find out more about the Reference linking.

Action

- **Set:** sets a value to the Out slot—in case no Data Point is linked to the output network point, it is possible to set its Out value with this action.

7.6.5 StringPoint

Applicable to OS version 1.0.0.4592

The StringPoint component is a network point class component, which is designed to work with the iSMA-B-LP room panel, and it allows to write values to registers displaying texts on the room panel's LCD display. The text sent to the room panel is defined in the StringPoint with the Set action, and it is always limited to 4 ASCII characters (any character entered over the 4th character is ignored and not displayed).

Note: Before using the component, make sure that its register address is assigned and its Status is OK.

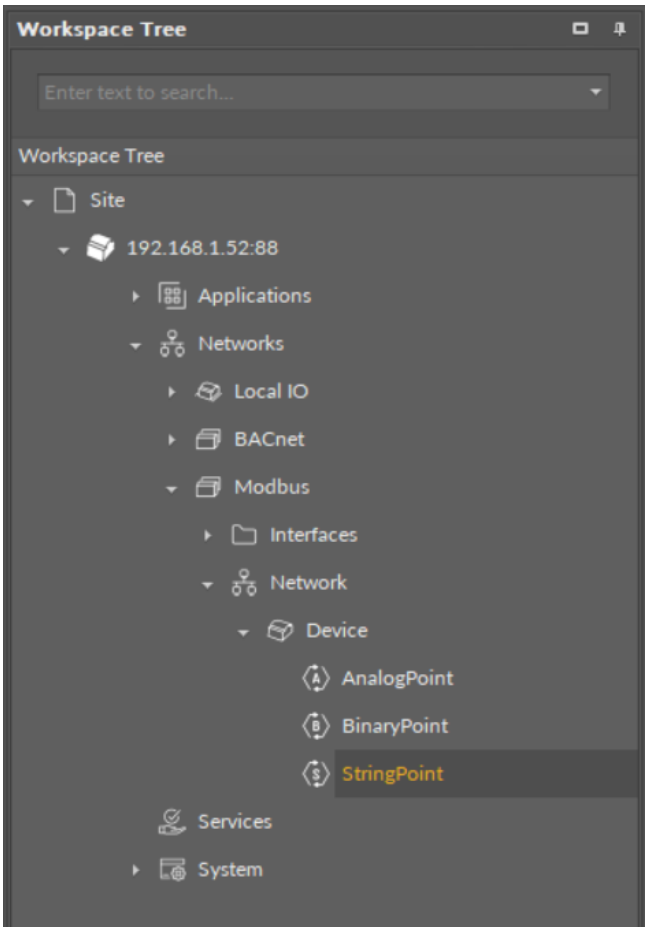


Figure 229. The StringPoint component

Slots

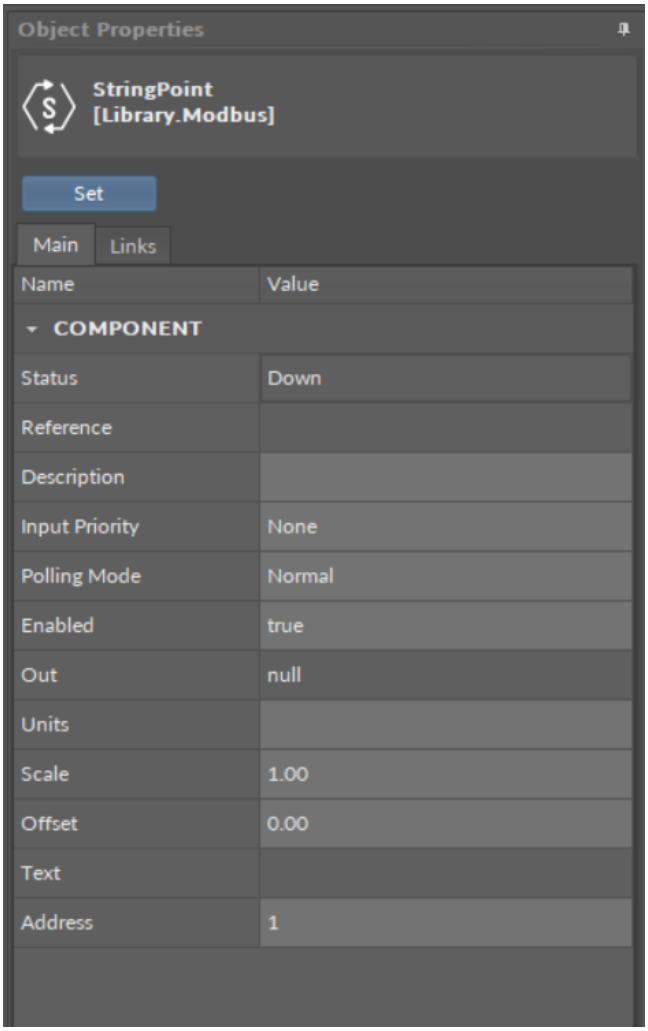


Figure 230. The StringPoint component slots

The StringPoint component has the following slots:

- **Status:** indicates the current status of the component; if the component works properly, its status is OK. The component becomes Disabled, once the Enabled slot is in false. The component's status is Fault, once the Address slot is null. If there is no response from the addressed point, the component goes into the Down status.
    - Available information: Disabled, Fault, Down, OK.
- **Reference:** a special slot allowing to connect network point class components with Data Point class components. It allows to transfer the Out slot value along with the component's status.

Note: Reference links from Data Points to network points also transfer values in the opposite direction, in a link-back-from process: having received a value by the Reference link, the network point transfers it back to the Data Point to whichever input priority from 1 to 16 is set in the network point.

- **Description:** an additional detailed information about a component that may be freely described by the user; the description may contain individual coding, defined in the user's system documentation, meter's or sensor's location, or any other information the user finds applicable.

Note: The description is effectively added only if the point allows it–the description is not added internally in the RAC18-IP for the remote point, but it is sent directly to the point.

- InputPriority: allows to indicate the input number in the Data Point, which the network point class component's output value is sent to, in case the network point detects the change on its Out slot; none priority is set by default.
    - Available settings: none, 1-16.
- Polling Mode: allows to set the frequency of sending polling requests for the point's value to remote devices—by default, the polling mode is set to normal;
    - Available settings: fast, normal, slow;
- Enabled: change of the slot's value enables or disables the component.
    - Available settings: true (enabled), false (disabled).

Note: If the Enabled slot is in false (meaning the component is disabled), the Status slot becomes Disabled.

- Out: a real value read from or written to the register of the remote device.

Note: If the component's Status is fault (e.g., an invalid value in the Address slot), the Out value is null.

- Units: defines a unit of the Out slot value; no unit is set by default;
- Scale: sets a fixed scaling factor for output linearization; the written value is calculated according to the inverse linear function formula (x=(y-b)/a), and the calculated value is read to the Out slot according to the linear function (y=ax+b); the Scale slot sets the a value of the formula; incorrect scaling of the results in the calculated value (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input and on the Out slot. If the scaling is correct, the values are identical;
- Offset: sets a fixed offset value to the output value; the value is calculated according to the inverse linear function formula (x=(y-b)/a) and read to the Out slot according to the linear function (y=ax+b); the Offset slot sets the b value of the formula; incorrect scaling of the results in the calculated value (exceeding the available range for the device, e.g., 0-10 V), results in indicating a different value on the input and on the Out slot. If the scaling is correct, the values are identical;
- Address: allows setting an address of the Modbus register in the remote device; once the component has been added, the slot's default value is null—for the component to operate properly, the unique address value must be set in this slot.

Note: The Modbus register address is set in decimal format.

- Text: displays the text, which is sent to the room panel's Modbus register defined in the Address slot. The text value is entered with the Set action, and it is transformed to the numeric value according to the ASCII code for each character. The text value is always limited to 4 characters. The text can either be entered in lower case characters, or capital letters—the room panel adjusts characters in case sensitive registers.

Modbus Registers for LCD Display

The Address slot in the StringPoint defines the Modbus register, which receives the text value to display on the room panel's LCD display. The Address slot receives the decimal address of the LP Modbus register. The list of Modbus registers, which can be addressed in the StringPoint is available here: StringPoint Modbus Registers.

Find out more about the Reference linking.

Action

- Set: allows to manually enter the text, which is sent to the room panel; it allows to change the room panel's text registers directly from the RAC18-IP.

StringPoint Modbus Registers

Applicable to OS version 1.0.0.4592

The Address slot in the StringPoint defines the Modbus register, which receives the text value to display on the room panel's LCD display. The registers that can receive a text value are listed below:

List of StringPoint Modbus Registers		
309 TEMPERATURE_NAME;	625 TEMPERATURE_BOOLEAN6 NAME;	1068 BLINDBOOLEAN4 FALSETEXT;
311 HUMIDITY_NAME;	627 TEMPERATURE_BOOLEAN6 TRUETEXT;	1074 BLINDBOOLEAN5 NAME;
313 CO2_NAME;	629 TEMPERATURE_BOOLEAN6 FALSETEXT;	1076 BLINDBOOLEAN5 TRUETEXT;
1508 OFFSET_NAME;	635 TEMPERATURE_BOOLEAN7 NAME;	1078 BLINDBOOLEAN5 FALSETEXT;
1510 SETPOINT_NAME;	637 TEMPERATURE_BOOLEAN7 TRUETEXT;	1084 BLINDBOOLEAN6 NAME;
1603 FAN_MODE_0 NAME;	639 TEMPERATURE_BOOLEAN7 FALSETEXT;	1086 BLINDBOOLEAN6 TRUETEXT;
1605 FAN_MODE_1 NAME;	645 TEMPERATURE_BOOLEAN8 NAME;	1088 BLINDBOOLEAN6 FALSETEXT;
1607 FAN_MODE_2 NAME;	647 TEMPERATURE_BOOLEAN8 TRUETEXT;	1094 BLINDBOOLEAN7 NAME;
1609 FAN_MODE_3 NAME;	649 TEMPERATURE_BOOLEAN8 FALSETEXT;	1096 BLINDBOOLEAN7 TRUETEXT;
1611 FAN_MODE_4 NAME;	656 FANNUMERIC1 NAME;	1098 BLINDBOOLEAN7 FALSETEXT;
1702 OCCUPANCY_MODE_0 NAME;	665 FANNUMERIC2 NAME;	1104 BLINDBOOLEAN8 NAME;
1704 OCCUPANCY_MODE_1 NAME;	674 FANNUMERIC3 NAME;	1106 BLINDBOOLEAN8 TRUETEXT;
319 MAIN_MENU_NUMERIC1 NAME;	683 FANNUMERIC4 NAME;	1108 BLINDBOOLEAN8 FALSETEXT;
324 MAIN_MENU_NUMERIC2 NAME;	692 FANNUMERIC5 NAME;	1115 ALARMSNUMERIC1 NAME;
329 MAIN_MENU_NUMERIC3 NAME;	701 FANNUMERIC6 NAME;	1124 ALARMSNUMERIC2 NAME;
334 MAIN_MENU_NUMERIC4 NAME;	710 FANNUMERIC7 NAME;	1133 ALARMSNUMERIC3 NAME;
339 MAIN_MENU_NUMERIC5 NAME;	719 FANNUMERIC8 NAME;	1142 ALARMSNUMERIC4 NAME;
344 MAIN_MENU_NUMERIC6 NAME;	728 FANBOOLEAN1 NAME;	1151 ALARMSNUMERIC5 NAME;
349 MAIN_MENU_NUMERIC7 NAME;	730 FANBOOLEAN1 TRUETEXT;	1160 ALARMSNUMERIC6 NAME;
354 MAIN_MENU_NUMERIC8 NAME;	732 FANBOOLEAN1 FALSETEXT;	1169 ALARMSNUMERIC7 NAME;
359 MAIN_MENU_BOOLEAN1 NAME;	738 FANBOOLEAN2 NAME;	1178 ALARMSNUMERIC8 NAME;
361 MAIN_MENU_BOOLEAN1 TRUETEXT;	740 FANBOOLEAN2 TRUETEXT;	1187 ALARMSBOOLEAN1 NAME;
363 MAIN_MENU_BOOLEAN1 FALSETEXT;	742 FANBOOLEAN2 FALSETEXT;	1189 ALARMSBOOLEAN1 TRUETEXT;
368 MAIN_MENU_BOOLEAN2 NAME;	748 FANBOOLEAN3 NAME;	1191 ALARMSBOOLEAN1 FALSETEXT;
370 MAIN_MENU_BOOLEAN2 TRUETEXT;	750 FANBOOLEAN3 TRUETEXT;	1197 ALARMSBOOLEAN2 NAME;
372 MAIN_MENU_BOOLEAN2 FALSETEXT;	752 FANBOOLEAN3 FALSETEXT;	1199 ALARMSBOOLEAN2 TRUETEXT;
377 MAIN_MENU_BOOLEAN3 NAME;	758 FANBOOLEAN4 NAME;	1201 ALARMSBOOLEAN2 FALSETEXT;
379 MAIN_MENU_BOOLEAN3 TRUETEXT;	760 FANBOOLEAN4 TRUETEXT;	1207 ALARMSBOOLEAN3 NAME;
381 MAIN_MENU_BOOLEAN3 FALSETEXT;	762 FANBOOLEAN4 FALSETEXT;	1209 ALARMSBOOLEAN3 TRUETEXT;
386 MAIN_MENU_BOOLEAN4 NAME;	768 FANBOOLEAN5 NAME;	1211 ALARMSBOOLEAN3 FALSETEXT;
388 MAIN_MENU_BOOLEAN4 TRUETEXT;	770 FANBOOLEAN5 TRUETEXT;	1217 ALARMSBOOLEAN4 NAME;
390 MAIN_MENU_BOOLEAN4 FALSETEXT;	772 FANBOOLEAN5 FALSETEXT;	1219 ALARMSBOOLEAN4 TRUETEXT;
395 MAIN_MENU_BOOLEAN5 NAME;	778 FANBOOLEAN6 NAME;	1221 ALARMSBOOLEAN4 FALSETEXT;
		1227 ALARMSBOOLEAN5 NAME;
		1229 ALARMSBOOLEAN5 TRUETEXT;
		1231 ALARMSBOOLEAN5 FALSETEXT;

List of StringPoint Modbus Registers		
397 MAIN_MENU_BOOLEAN5 TRUETEXT;	780 FANBOOLEAN6 TRUETEXT;	1237 ALARMSBOOLEAN6 NAME;
399 MAIN_MENU_BOOLEAN5 FALSETEXT;	782 FANBOOLEAN6 FALSETEXT;	1239 ALARMSBOOLEAN6 TRUETEXT;
404 MAIN_MENU_BOOLEAN6 NAME;	788 FANBOOLEAN7 NAME;	1241 ALARMSBOOLEAN6 FALSETEXT;
406 MAIN_MENU_BOOLEAN6 TRUETEXT;	790 FANBOOLEAN7 TRUETEXT;	1247 ALARMSBOOLEAN7 NAME;
408 MAIN_MENU_BOOLEAN6 FALSETEXT;	792 FANBOOLEAN7 FALSETEXT;	1249 ALARMSBOOLEAN7 TRUETEXT;
413 MAIN_MENU_BOOLEAN7 NAME;	798 FANBOOLEAN8 NAME;	1251 ALARMSBOOLEAN7 FALSETEXT;
415 MAIN_MENU_BOOLEAN7 TRUETEXT;	800 FANBOOLEAN8 TRUETEXT;	1257 ALARMSBOOLEAN8 NAME;
417 MAIN_MENU_BOOLEAN7 FALSETEXT;	802 FANBOOLEAN8 FALSETEXT;	1259 ALARMSBOOLEAN8 TRUETEXT;
422 MAIN_MENU_BOOLEAN8 NAME;	809 LIGHTNUMERIC1 NAME;	1261 ALARMSBOOLEAN8 FALSETEXT;
424 MAIN_MENU_BOOLEAN8 TRUETEXT;	818 LIGHTNUMERIC2 NAME;	1268 SETTINGSNUMERIC1 NAME;
426 MAIN_MENU_BOOLEAN8 FALSETEXT;	827 LIGHTNUMERIC3 NAME;	1277 SETTINGSNUMERIC2 NAME;
503 TEMPERATURE_NUMERIC1 NAME;	836 LIGHTNUMERIC4 NAME;	1286 SETTINGSNUMERIC3 NAME;
512 TEMPERATURE_NUMERIC2 NAME;	845 LIGHTNUMERIC5 NAME;	1295 SETTINGSNUMERIC4 NAME;
521 TEMPERATURE_NUMERIC3 NAME;	854 LIGHTNUMERIC6 NAME;	1304 SETTINGSNUMERIC5 NAME;
530 TEMPERATURE_NUMERIC4 NAME;	863 LIGHTNUMERIC7 NAME;	1313 SETTINGSNUMERIC6 NAME;
539 TEMPERATURE_NUMERIC5 NAME;	872 LIGHTNUMERIC8 NAME;	1322 SETTINGSNUMERIC7 NAME;
548 TEMPERATURE_NUMERIC6 NAME;	881 LIGHTBOOLEAN1 NAME;	1331 SETTINGSNUMERIC8 NAME;
557 TEMPERATURE_NUMERIC7 NAME;	883 LIGHTBOOLEAN1 TRUETEXT;	1340 SETTINGSBOOLEAN1 NAME;
566 TEMPERATURE_NUMERIC8 NAME;	885 LIGHTBOOLEAN1 FALSETEXT;	1342 SETTINGSBOOLEAN1 TRUETEXT;
575 TEMPERATURE_BOOLEAN1 NAME;	891 LIGHTBOOLEAN2 NAME;	1344 SETTINGSBOOLEAN1 FALSETEXT;
577 TEMPERATURE_BOOLEAN1 TRUETEXT;	893 LIGHTBOOLEAN2 TRUETEXT;	1350 SETTINGSBOOLEAN2 NAME;
579 TEMPERATURE_BOOLEAN1 FALSETEXT;	895 LIGHTBOOLEAN2 FALSETEXT;	1352 SETTINGSBOOLEAN2 TRUETEXT;
585 TEMPERATURE_BOOLEAN2 NAME;	901 LIGHTBOOLEAN3 NAME;	1354 SETTINGSBOOLEAN2 FALSETEXT;
587 TEMPERATURE_BOOLEAN2 TRUETEXT;	903 LIGHTBOOLEAN3 TRUETEXT;	1360 SETTINGSBOOLEAN3 NAME;
589 TEMPERATURE_BOOLEAN2 FALSETEXT;	905 LIGHTBOOLEAN3 FALSETEXT;	1362 SETTINGSBOOLEAN3 TRUETEXT;
595 TEMPERATURE_BOOLEAN3 NAME;	911 LIGHTBOOLEAN4 NAME;	1364 SETTINGSBOOLEAN3 FALSETEXT;
597 TEMPERATURE_BOOLEAN3 TRUETEXT;	913 LIGHTBOOLEAN4 TRUETEXT;	1370 SETTINGSBOOLEAN4 NAME;
	915 LIGHTBOOLEAN4 FALSETEXT;	1372 SETTINGSBOOLEAN4 TRUETEXT;
	921 LIGHTBOOLEAN5 NAME;	1374 SETTINGSBOOLEAN4 FALSETEXT;
	923 LIGHTBOOLEAN5 TRUETEXT;	1380 SETTINGSBOOLEAN5 NAME;
	925 LIGHTBOOLEAN5 FALSETEXT;	1382 SETTINGSBOOLEAN5 TRUETEXT;
	931 LIGHTBOOLEAN6 NAME;	1384 SETTINGSBOOLEAN5 FALSETEXT;
	933 LIGHTBOOLEAN6 TRUETEXT;	1390 SETTINGSBOOLEAN6 NAME;
	935 LIGHTBOOLEAN6 FALSETEXT;	1392 SETTINGSBOOLEAN6 TRUETEXT;
	941 LIGHTBOOLEAN7 NAME;	
	943 LIGHTBOOLEAN7 TRUETEXT;	
	945 LIGHTBOOLEAN7 FALSETEXT;	
	951 LIGHTBOOLEAN8 NAME;	
	953 LIGHTBOOLEAN8 TRUETEXT;	
	955 LIGHTBOOLEAN8 FALSETEXT;	
	962 BLINDNUMERIC1 NAME;	
	971 BLINDNUMERIC2 NAME;	
	980 BLINDNUMERIC3 NAME;	
	989 BLINDNUMERIC4 NAME;	
	998 BLINDNUMERIC5 NAME;	
	1007 BLINDNUMERIC6 NAME;	

List of StringPoint Modbus Registers		
599 TEMPERATURE_BOOLEAN3 FALSETEXT; 605 TEMPERATURE_BOOLEAN4 NAME; 607 TEMPERATURE_BOOLEAN4 TRUETEXT; 609 TEMPERATURE_BOOLEAN4 FALSETEXT; 615 TEMPERATURE_BOOLEAN5 NAME; 617 TEMPERATURE_BOOLEAN5 TRUETEXT; 618 TEMPERATURE_BOOLEAN5 FALSETEXT;	1016 BLINDNUMERIC7 NAME; 1025 BLINDNUMERIC8 NAME; 1034 BLINDBOOLEAN1 NAME; 1036 BLINDBOOLEAN1 TRUETEXT; 1038 BLINDBOOLEAN1 FALSETEXT; 1044 BLINDBOOLEAN2 NAME; 1046 BLINDBOOLEAN2 TRUETEXT; 1048 BLINDBOOLEAN2 FALSETEXT; 1054 BLINDBOOLEAN3 NAME; 1056 BLINDBOOLEAN3 TRUETEXT; 1058 BLINDBOOLEAN3 FALSETEXT; 1064 BLINDBOOLEAN4 NAME; 1066 BLINDBOOLEAN4 TRUETEXT;	1394 SETTINGSBOOLEAN6 FALSETEXT; 1400 SETTINGSBOOLEAN7 NAME; 1402 SETTINGSBOOLEAN7 TRUETEXT; 1404 SETTINGSBOOLEAN7 FALSETEXT; 1410 SETTINGSBOOLEAN8 NAME; 1412 SETTINGSBOOLEAN8 TRUETEXT; 1414 SETTINGSBOOLEAN8 FALSETEXT.

Table 6. List of StringPoint Modbus registers

7.6.6 Quick Start-up for Modbus

In order to launch the Modbus communication properly, it is necessary to follow these steps:

Step 1: Having the device added correctly in the iC Tool, expand the Networks container, and go to the Modbus component.

Step 2: The Interfaces component with the Serial component are added by default. Go to the Serial component, and configure the RS485 port for Modbus communication.

Step 3: The RAC18-IP controller allows to communicate as the Modbus RTU master device. For this purpose, the Network and Device components need to be added, along with AnalogPoint, BinaryPoint, and StringPoint components, as necessary. Go to the Device Libraries and expand the Core library for the Network component and Modbus library for other components. Choose components to be added—components may be added one by one or grouped in one selection. Drag the selected component(s) and drop it(them) under the Modbus component in the Networks container.

Worth to Notice:

Please remember that the components' hierarchy needs to be preserved here: the Network component has to be located under the Modbus component, the Device component under the Network component, and the AnalogPoint/BinaryPoint/StringPoint components have to be placed under the Device component.

If the superior component is selected in the Workspace Tree window and its special view (Network Manager/Device Manager/Point Manager) is opened in the main screen, the Device Libraries shows only the components that can be added directly under it. For example, if the Modbus component is selected in the Workspace Tree window, the Device Libraries shows only the Network component in the Core library.

Step 5: Go to each added component, open its Property Sheet (or go the the Modbus component's Property Sheet and expand each component there), and set their parameters (enable, addresses, etc.). Make sure to save the changes.

Worth to Notice:

In order to facilitate working with Modbus component, special views have been developed:

- Network Manager, available in the Modbus component;
- Device Manager, available in the Network component;
- Point Manager, available in the Device component.

Ready to Use: Configured components are ready for proper Modbus communication.

Network Manager for Modbus

The Network Manager view is available for the Modbus component. It lists all Modbus networks configured on the device's ports. The Network Manager view shows the statuses, ports (which the network is configured on), and enabled or disabled states of the the network. Once the network, listed in the Network Manager, is double-clicked, the respective Network component is opened.

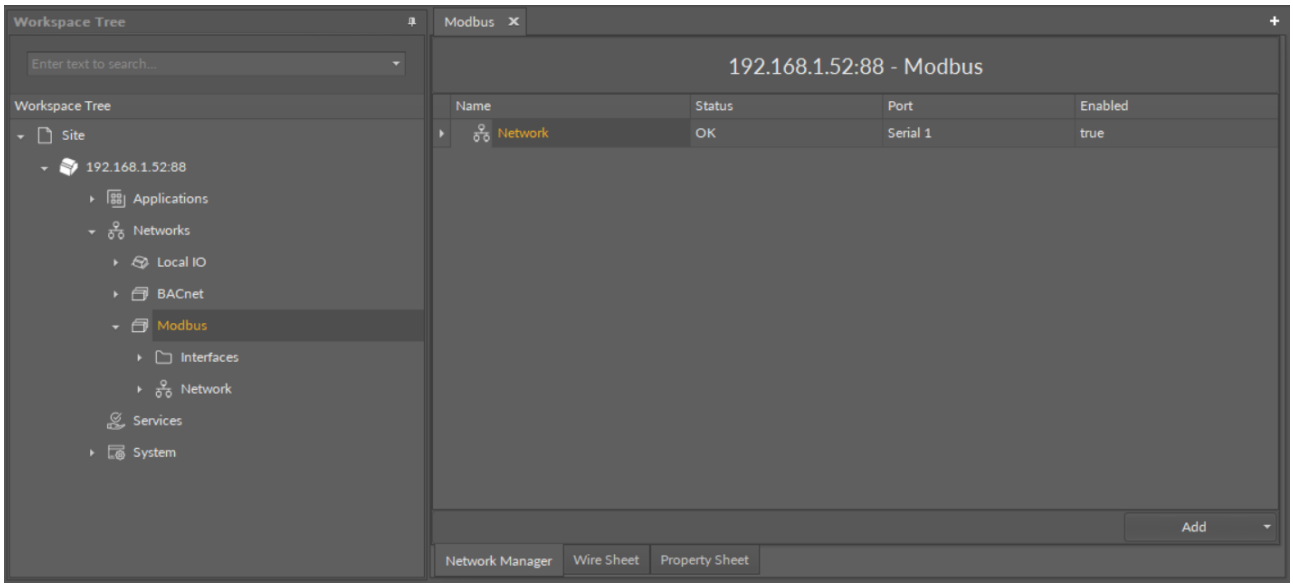


Figure 231. The Network Manager view

Opening Network Manager

The Network Manager view is accessible from the context menu of the Modbus component. It is also automatically opened if the Modbus component is double-clicked in the Workspace Tree window.

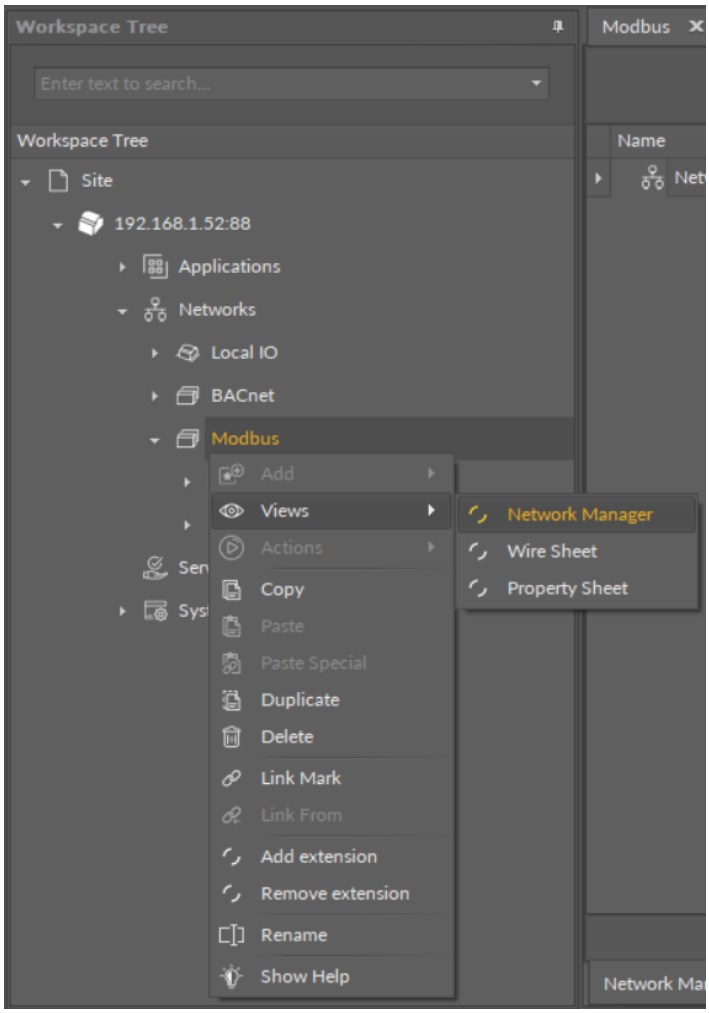


Figure 232. Opening the Network Manager

Adding Modbus Networks

The networks may be added to the Network Manager twofold: dragging and dropping the Network component to the Modbus component from the Core library (in the Device Libraries window), or using a special Add function in the Network Manager view available in the bottom right corner.

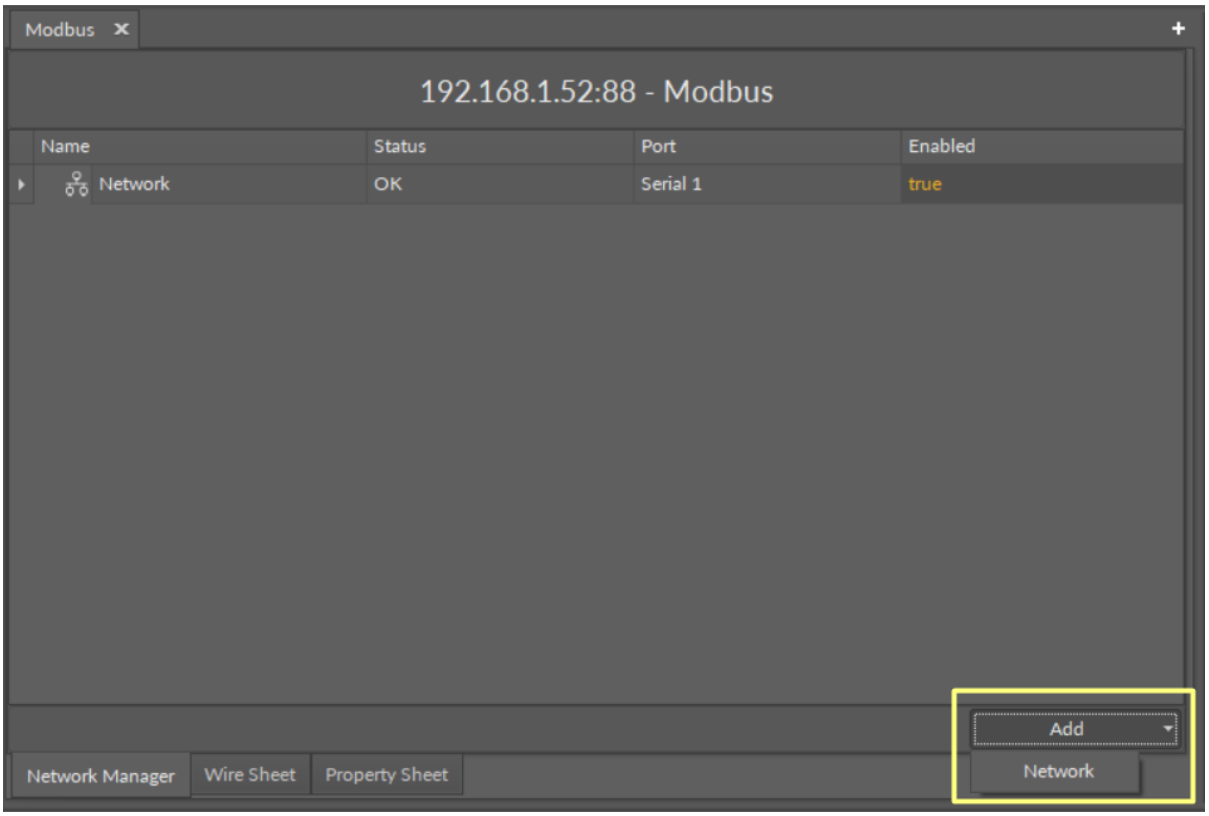


Figure 233. Adding the Modbus network

Using this Add button opens the dialog window, which allows to adjust the quantity of networks to be added.

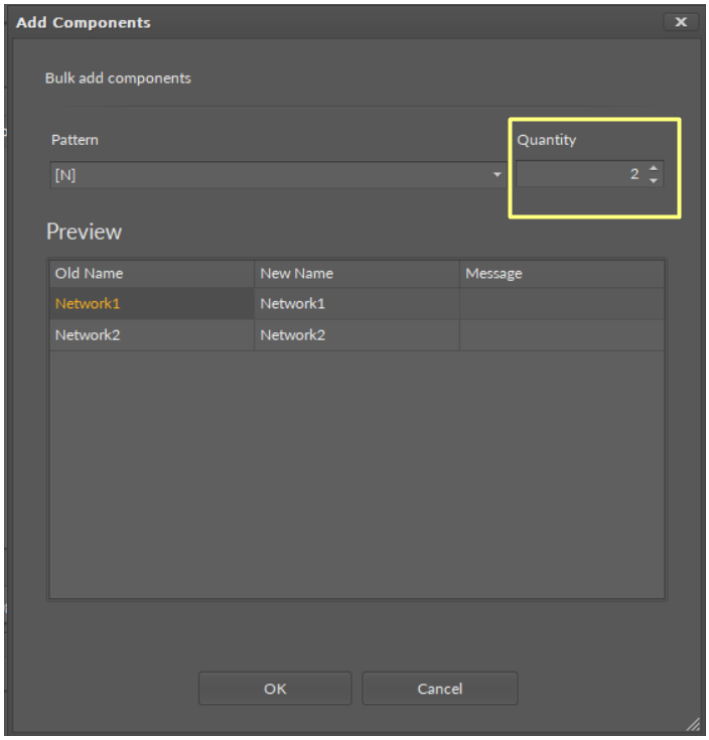


Figure 234. The dialog window

Multiediting of Common Slots

The Network Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is

available in the Object Properties window, upon selecting one-type components in the Network Manager with Ctrl or Shift keys.

Device Manager for Modbus

The Device Manager view is available for the Modbus Network component. It lists all devices added to the configured Modbus network. The Device Manager view shows the statuses, device addresses, and enabled or disabled states of the devices in the network. Once the device, listed in the Device Manager, is double clicked, the respective Device component is opened.

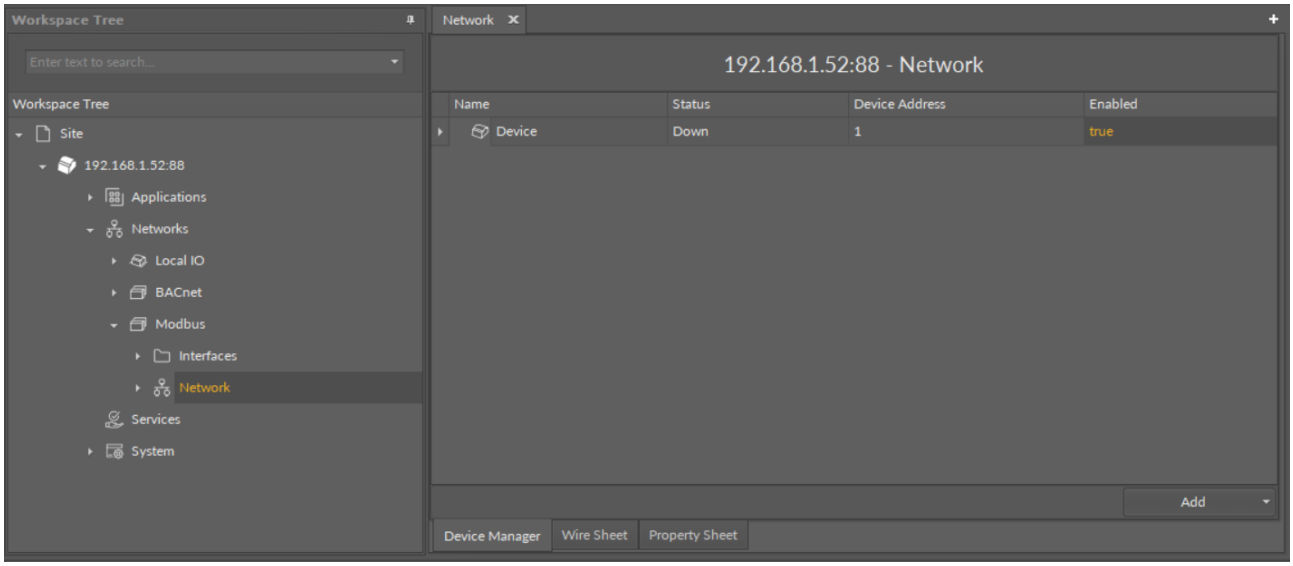


Figure 235. The Device Manager view

Opening Device Manager

The Device Manager view is accessible from the context menu of the Network component. It is also automatically opened if the Network component is double-clicked in the Workspace Tree window.

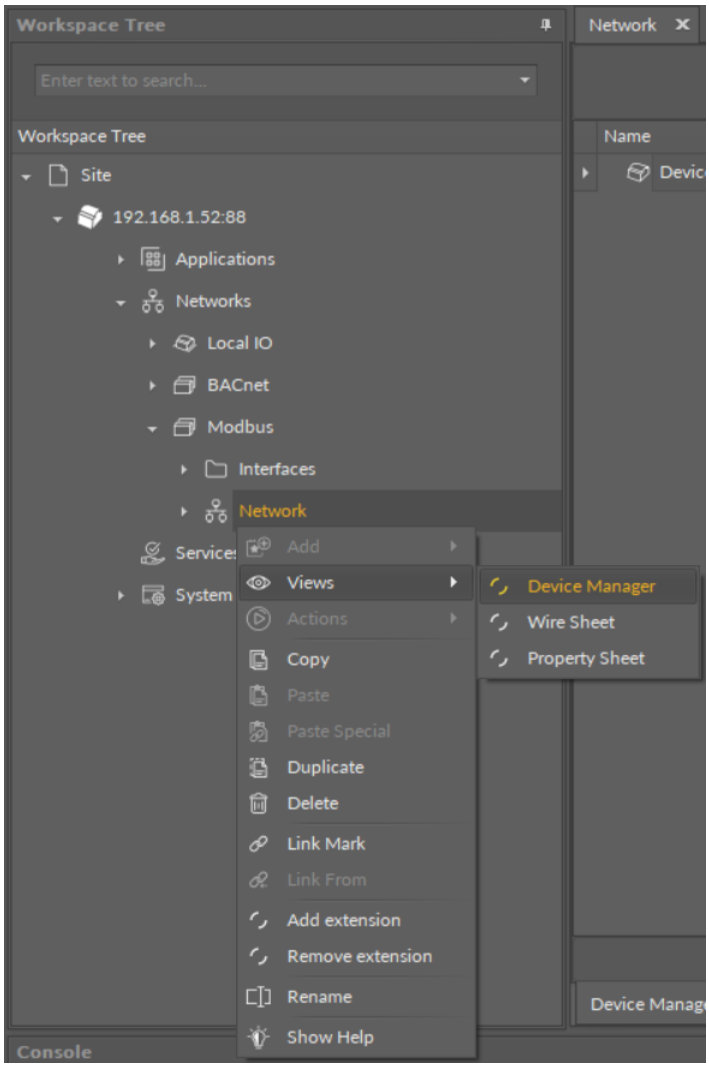


Figure 236. Opening the Device Manager

Adding Modbus Devices

The devices may be added to the Modbus network twofold: dragging and dropping the Device component to the Network component from the Modbus library (in the Device Libraries window), or using a special Add function in the Device Manager view available in the bottom right corner.

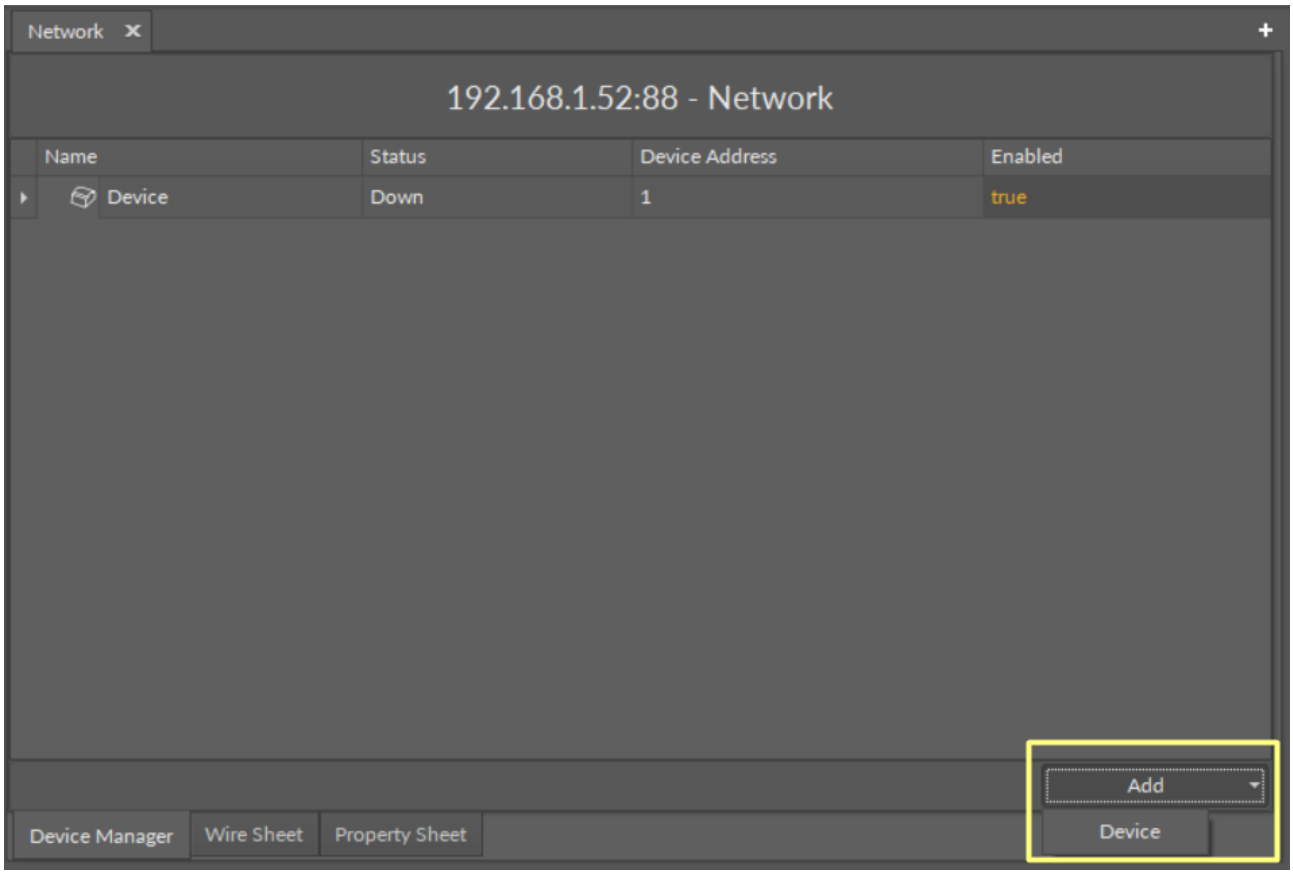


Figure 237. Adding Modbus devices

Using this Add button opens the dialog window, which allows to adjust the quantity of devices to be added.

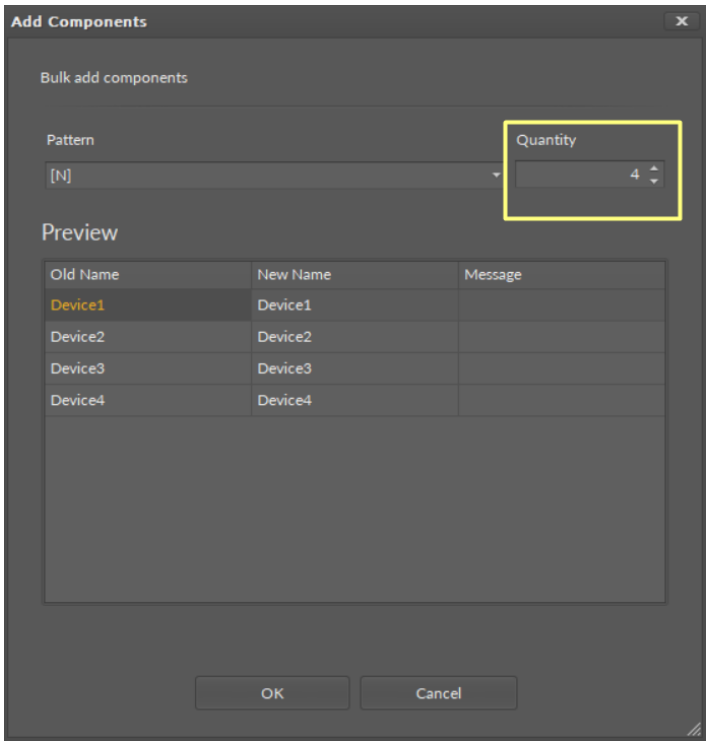


Figure 238. The dialog window

Multiediting of Common Slots

The Device Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is available in the Object Properties window, upon selecting one-type components in the Device Manager with Ctrl or Shift keys.

Point Manager for Modbus

The Point Manager view is available for each device added to the Modbus network. It lists all Modbus Points added to the Device component, and shows their Out slot value, status, address, polling mode, and enabled or disabled state.

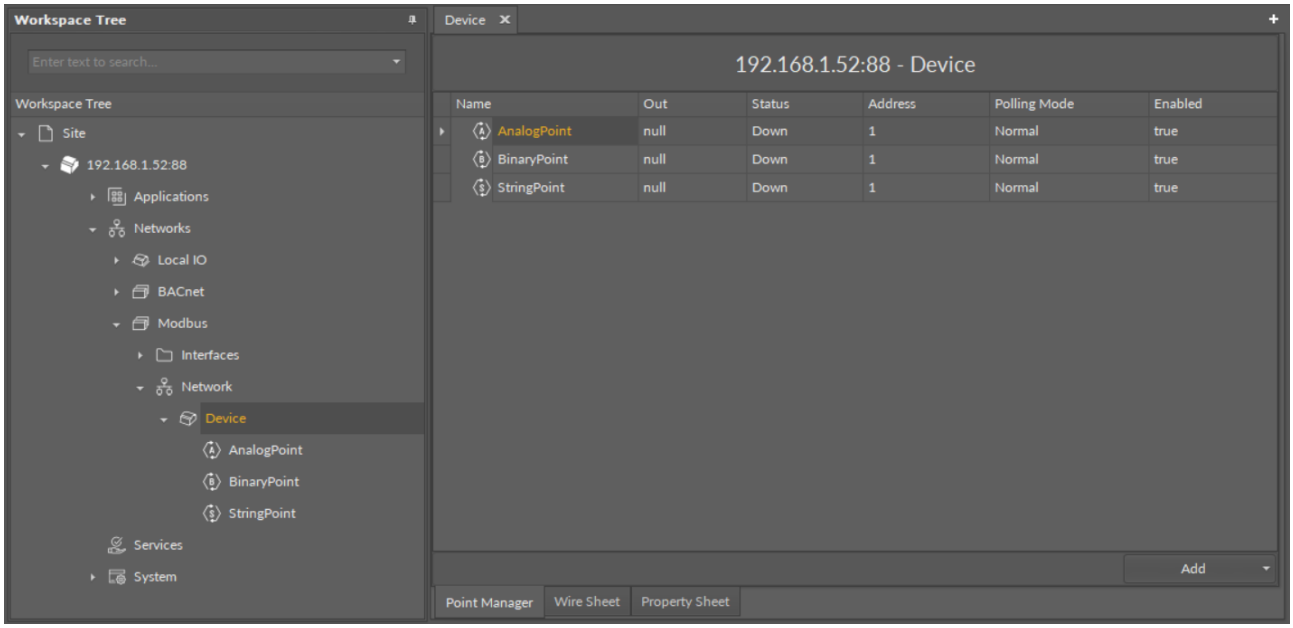


Figure 239. The Point Manager

Opening Point Manager

The Point Manager view is accessible from the context menu of the Device component. It is also automatically opened if the Device component is double-clicked in the Workspace Tree window.

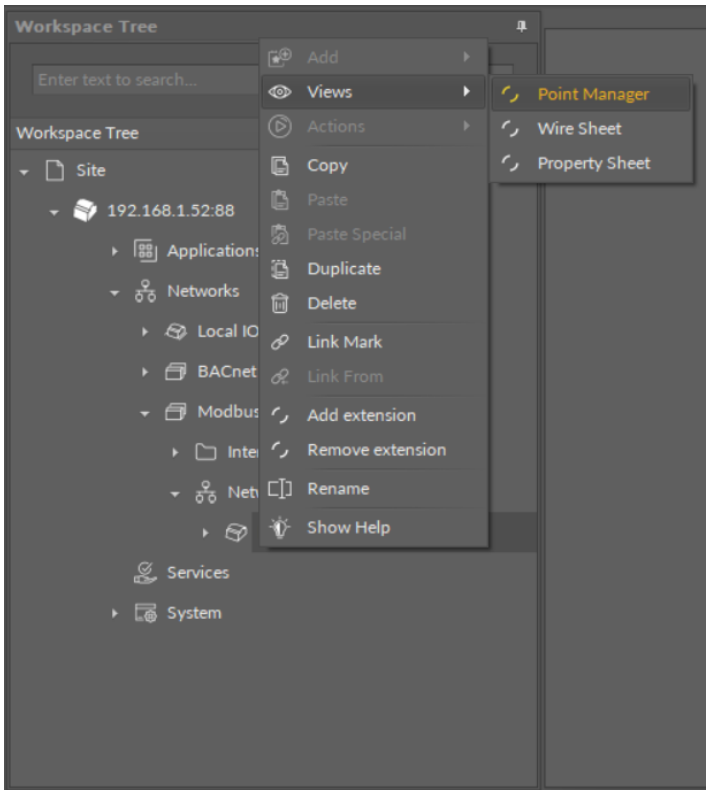


Figure 240. Opening the Point Manager

Adding Modbus Points

The Modbus points may be added to the device twofold: dragging and dropping the Modbus points to the Device component from the Modbus library (in the Device Libraries window), or using a special Add function in the Point Manager view available in the bottom right corner. The Add function allows to add any of the Modbus points available in the Modbus library.

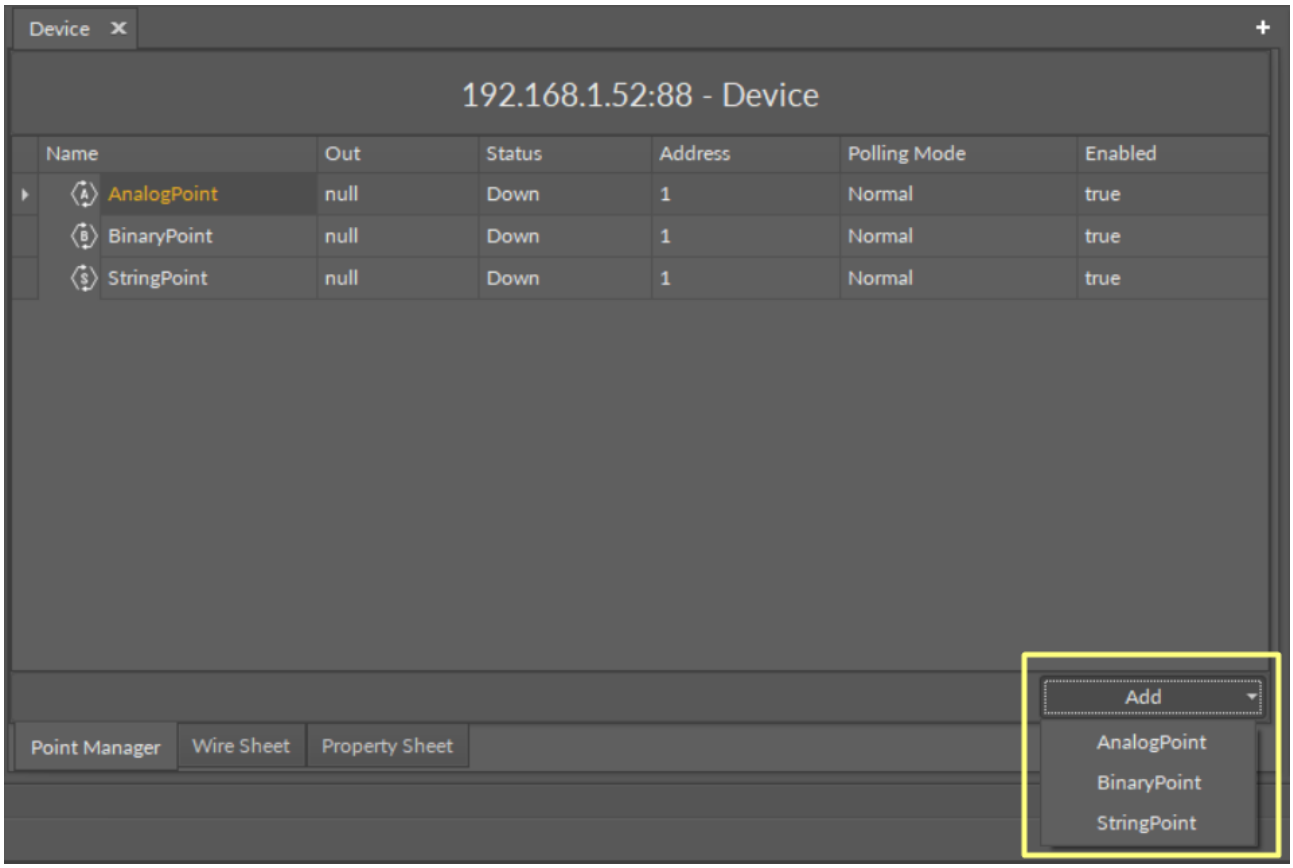


Figure 241. The Add button

Using this Add button opens the dialog window, which allows to adjust the quantity of Modbus Points to be added.

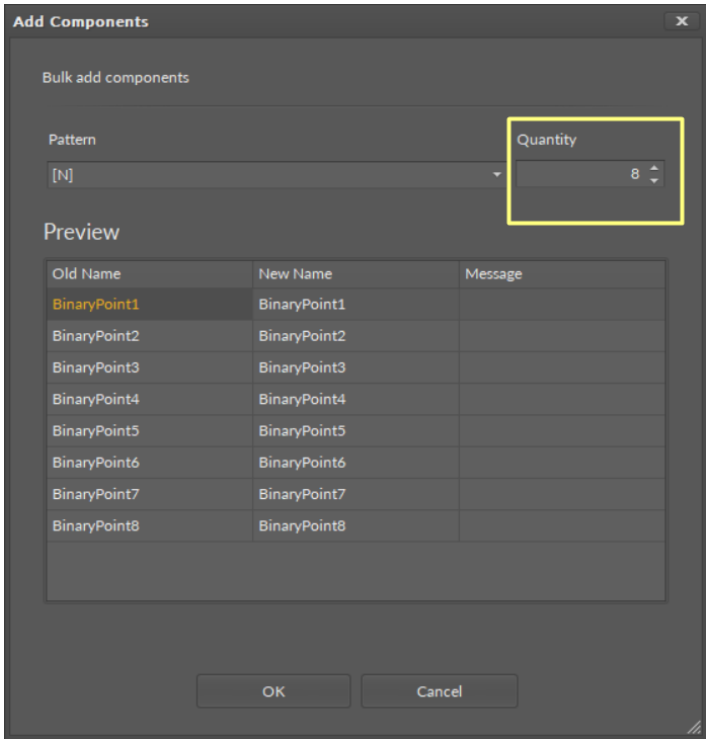


Figure 242. The dialog window

Multiediting of Common Slots

The Point Manager view allows multiediting of common slots in components of the same type, for example, to enable all one-type components at once. Multiediting is available in the Object Properties window, upon selecting one-type components in the Point Manager with Ctrl or Shift keys.

8 Services Container

The Services container provides a space for additional services developed to enhance the device's functionalities. Services may be added to the device and then used within applications. They are designed to provide additional functionalities to the basic algorithms included in applications, allowing the device to communicate with systems superior to building automation systems (including cloud data exchange) .

Services introduce an additional logical layer to the algorithm in the application–as the algorithm makes calculations in cycles, the service may correlate it with external factors, for example, setting limiting values that invoke specific actions in the application. The basic algorithm, therefore, retrieves some additional information from such service, and enhances its own functionality, for example, taking into account the weather while executing the application for watering the garden.

Unlike Networks, which manage strictly automation data essential for the operation of the device (management of physical inputs and outputs, communication protocols, etc.), Services enhance the algorithm's options, making it more resilient and adjustable for multiple application purposes.

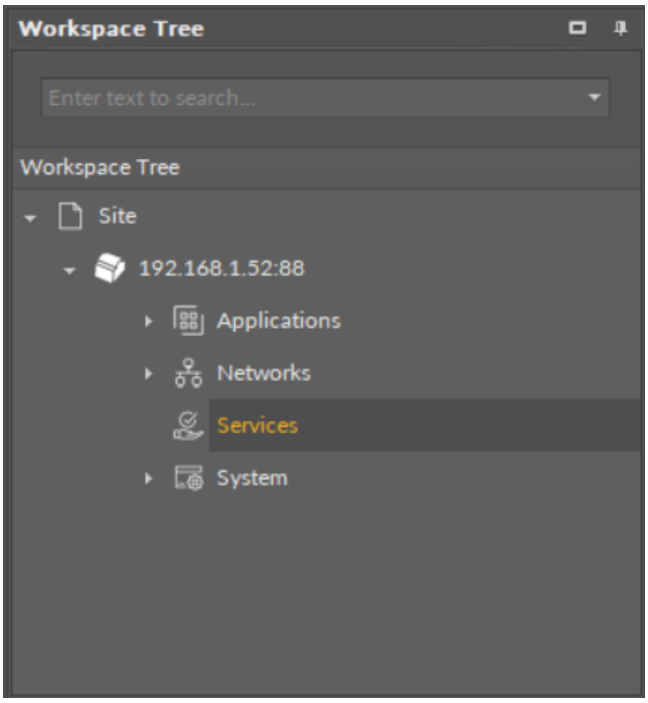


Figure 243. The Services container

9 System Container

The System container is a configuration center of a device; it includes information about the operating system and settings. It provides a hardware characteristic of the device, informing about the OS version, CPU performance, IP address, port number, or number of inputs and outputs.

The System container is a starting point to work with the device–it is a place to configure the device and connect it to the network. One of the most useful features of the System container is a possibility to set a unique IP address of the device.

In further operations it provides tools for monitoring and troubleshooting such as the Logs component, which provides an adjustable register of events taking place during the operation of the device. The System container includes a tree of components providing categorized information about the device and its settings.

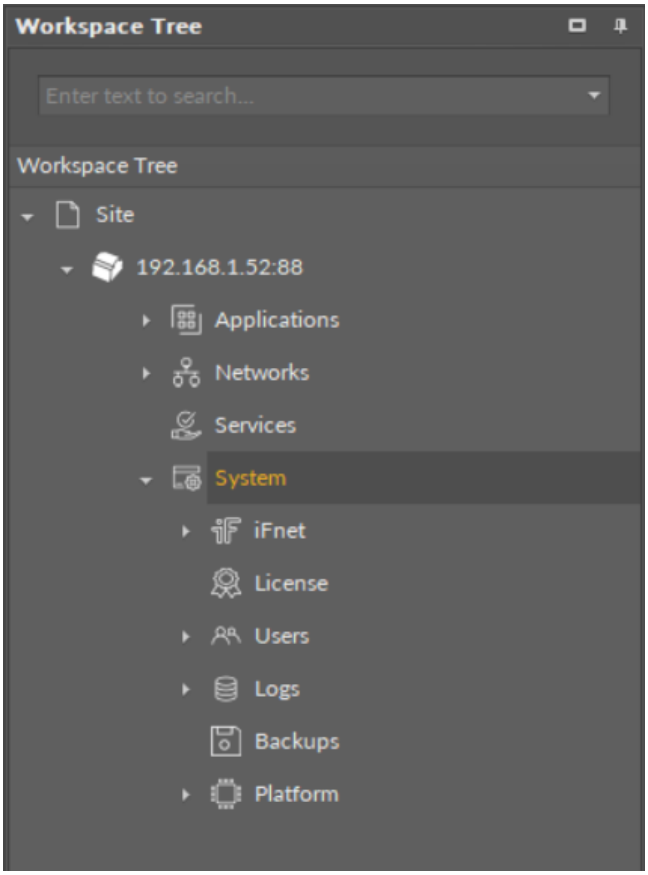


Figure 244. The System container

9.1 Basic Concept of System

The System container is a configuration center of a device; it includes information about the operating system and settings. It provides a hardware characteristic of the device, informing about the OS version, CPU performance, IP address, port number, or number of inputs and outputs.

The System container is a starting point to work with the device–it is a place to configure the device and connect it to the network. One of the most useful features of the System container is a possibility to set a unique IP address of the device.

In further operations it provides tools for monitoring and troubleshooting such as the Logs component, which provides an adjustable register of events taking place during the operation of the device. The System container includes a tree of components providing categorized information about the device and its settings.

9.2 License

Applicable to OS version 1.0.0.4592

The License component provides data about the license status in the device. It shows if the device is licensed, how many Data Points are included in the license, how many of them have already been used, and how many are still available. All slots in the License component are read-only type.

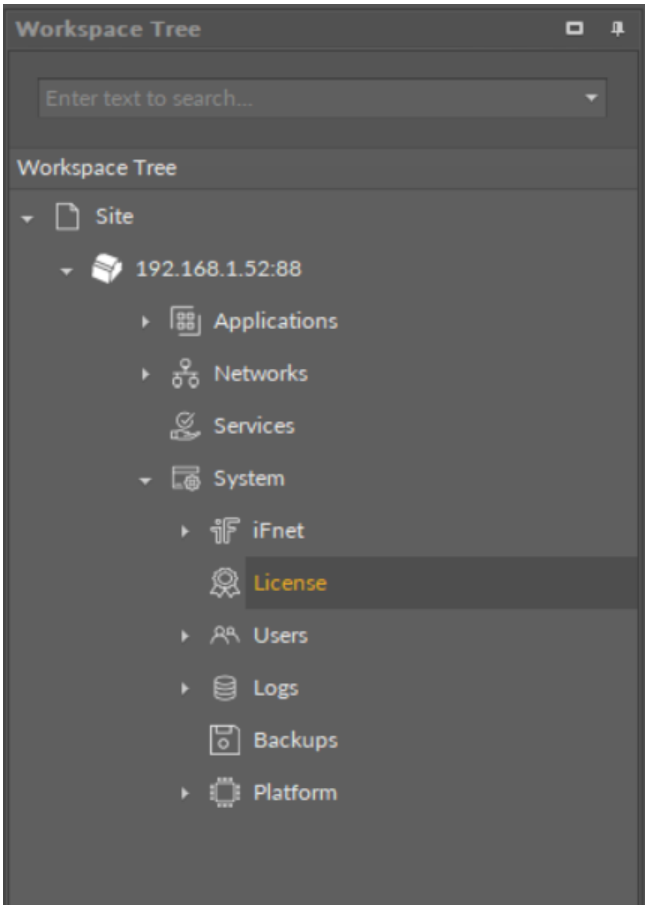


Figure 245. The License component

The License component has the following slots:

- **Status:** indicates the component's status:
  - Available information: OK, Fault, Exceeded;
- **Maximum Data Points:** shows the overall number of Data Points available withing the license;

- **Used Data Points:** shows the number of Data Points that have already been used in applications;
- **Available Data Points:** shows the number of Data Points remaining within the license.

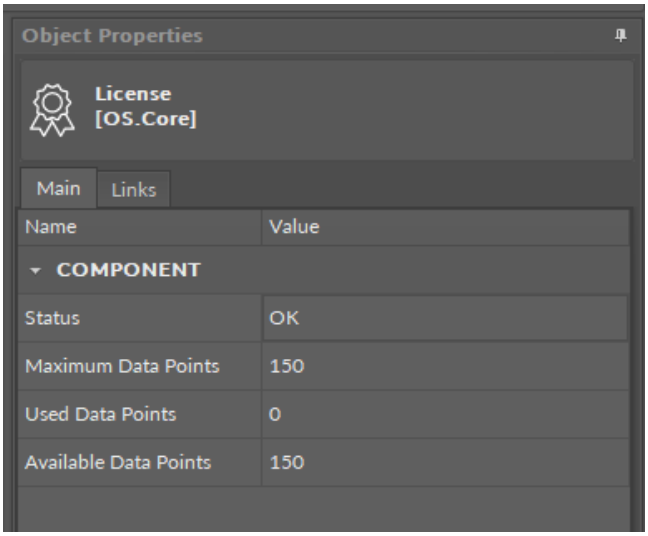


Figure 246. The License component slots

9.3 iFnet

Applicable to OS version 1.0.0.4592

The iFnet component provides information about the TCP port number that the user connects with the device from the iC Tool. The default port number is set to 88, and it can be changed after the device's restart.

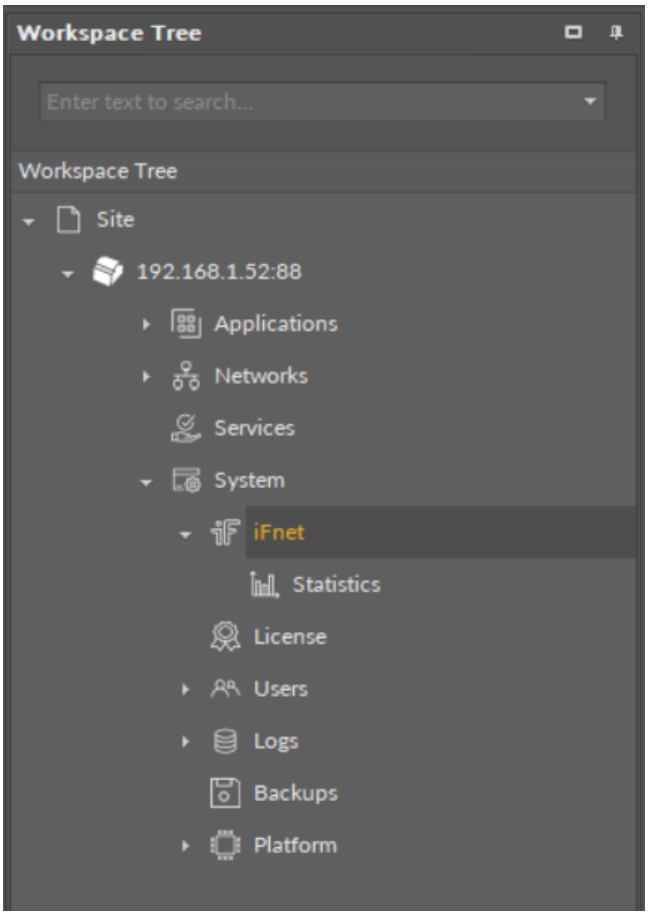


Figure 247. The iFnet component

The iFnet component has the following slots:

- **Info:** informs about a required device restart after changing the port's number;
- **Port:** shows the default communication port number and allows to change it manually.

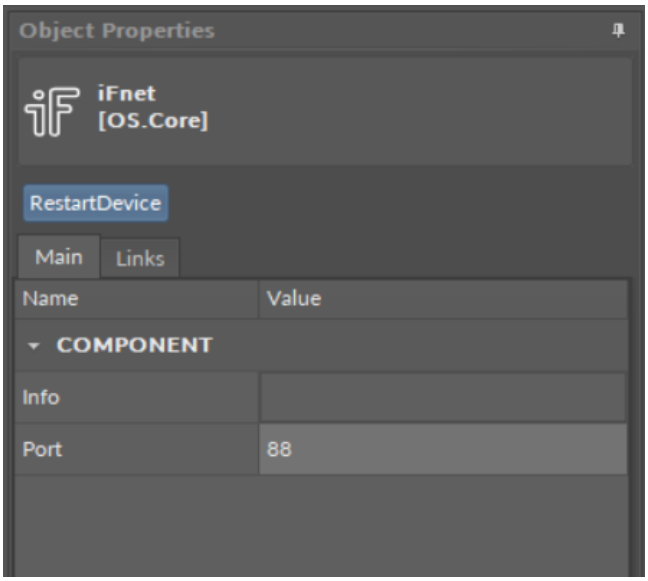


Figure 248. The iFnet component's slots

The iFnet port number can be changed by the user. Once the change is introduced, it requires saving and restarting the device. Before saving the component provides a notice:

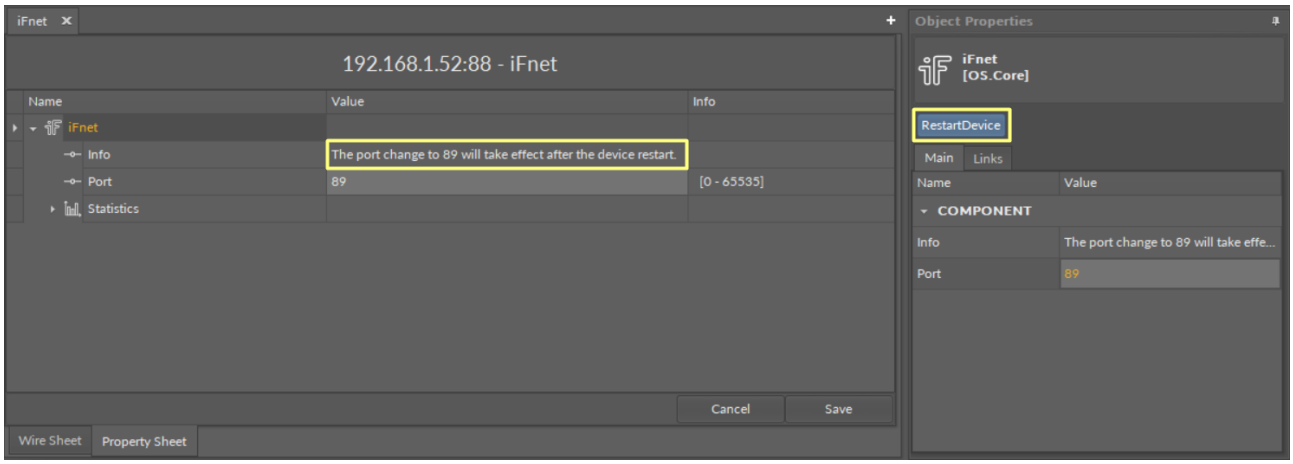


Figure 249. The restart notice and RestartDevice action

The iFnet component has a component's extension–the Statistics.

9.3.1 Statistics

Applicable to OS version 1.0.0.4592

The Statistics component provides information about the number of messages regarding the iFnet protocol (servicing the communication with the device from the iC Tool), transferred during one uptime of the device.

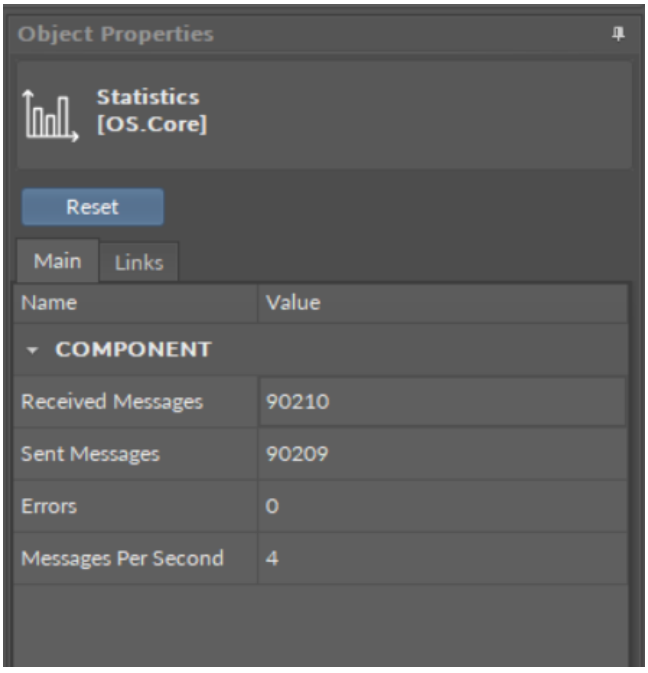


Figure 250. The Statistics component

Slots

The Statistics component has the following slots:

- **Received Messages:** shows the number of messages received by the device;
- **Sent Messages:** shows the number of messages sent by the device;
- **Errors:** shows the number of errors occurred while sending or receiving messages;

- **Messages per Second:** shows the number of messages sent and received by the device.

Action

The Statistics component has a Reset action available:

- **Reset:** erases all message statistics and starts counting from 0.

9.4 Users

Applicable to OS version 1.0.0.4592

The Users component lists the users available in the device. In the first iteration, the list contains an admin user with full permissions, and there is no option to add other users. The Users component allows to change the user's password.

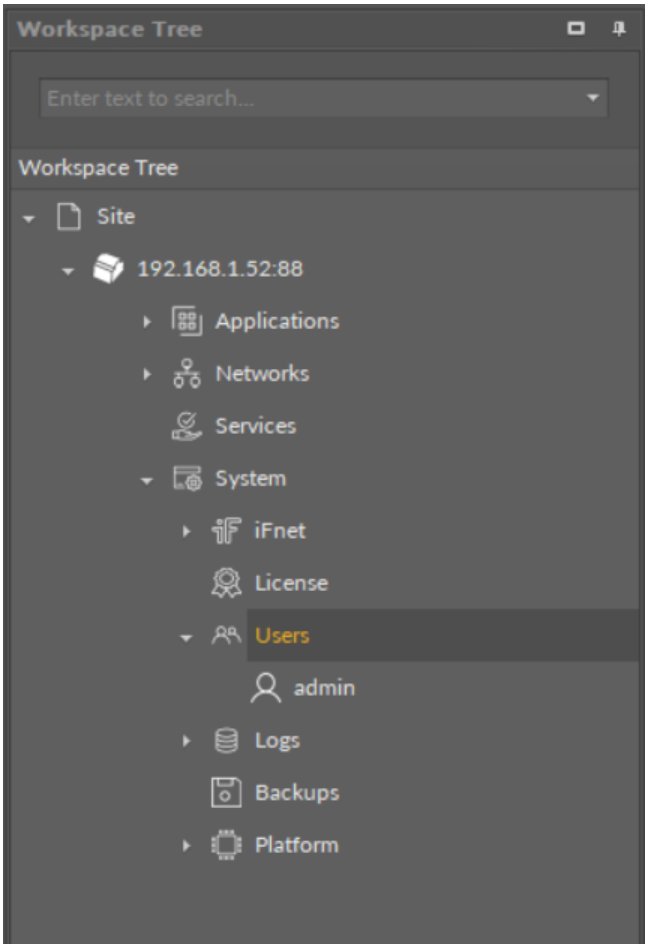


Figure 251. The Users component

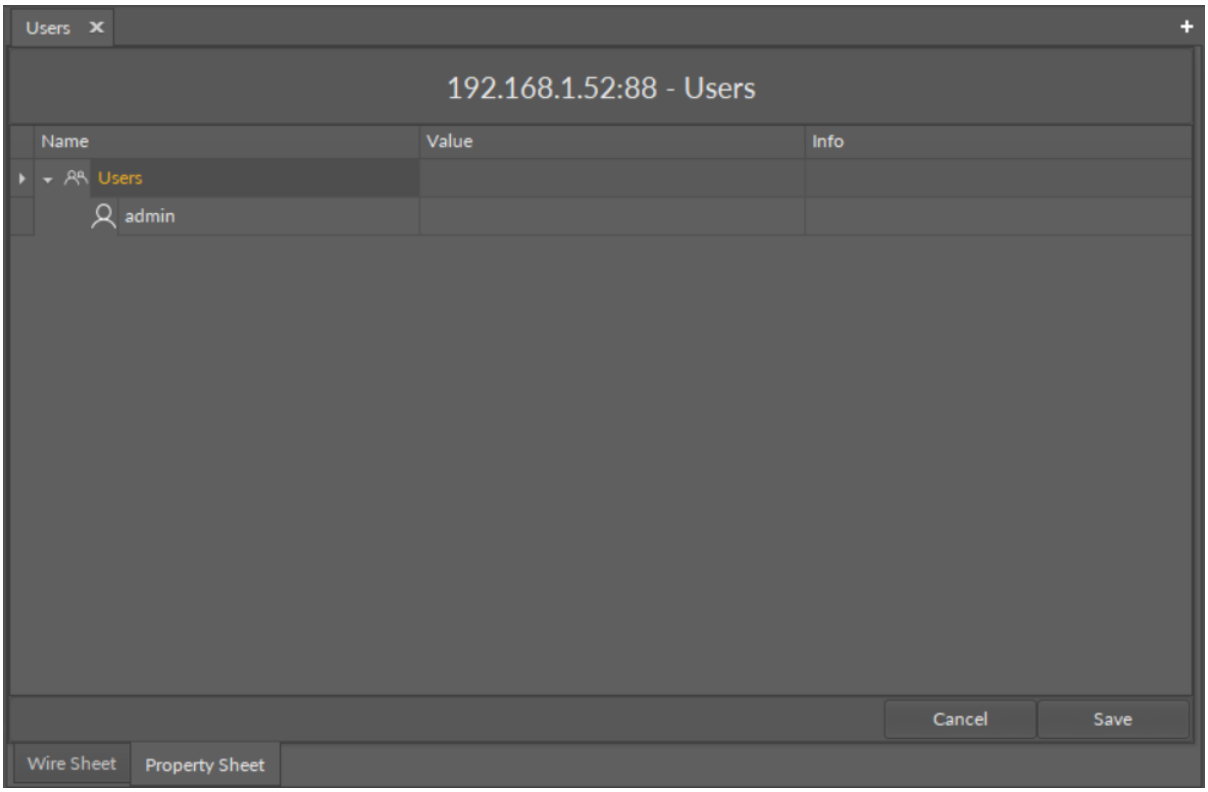


Figure 252. The Users Manager view

The user admin has one action available:

- Change Password: allows to change the user's password.

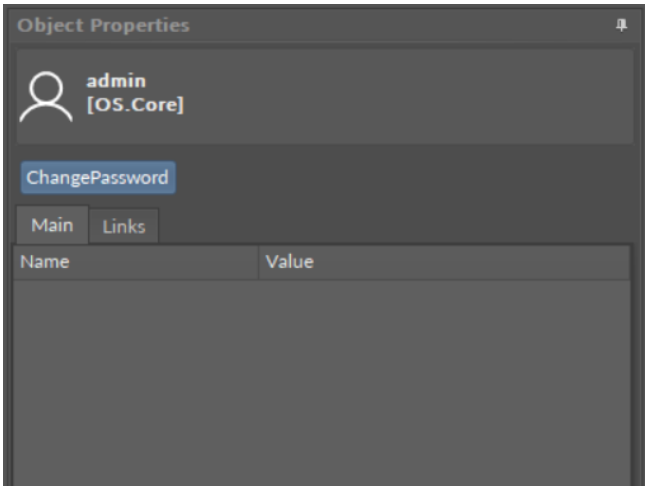


Figure 253. The Change Password action

WARNING!

Please remember that the new password set by the user has to be at least 8 characters long, have at least one digit and at least one capital letter.

9.4.1 First Logging In

Applicable to OS version 1.0.0.4592

The first time the user logs in to the device in the iC Tool, the software requires to change the default password.

Once the user initiates connecting the device, the iC Tool displays the authentication window:

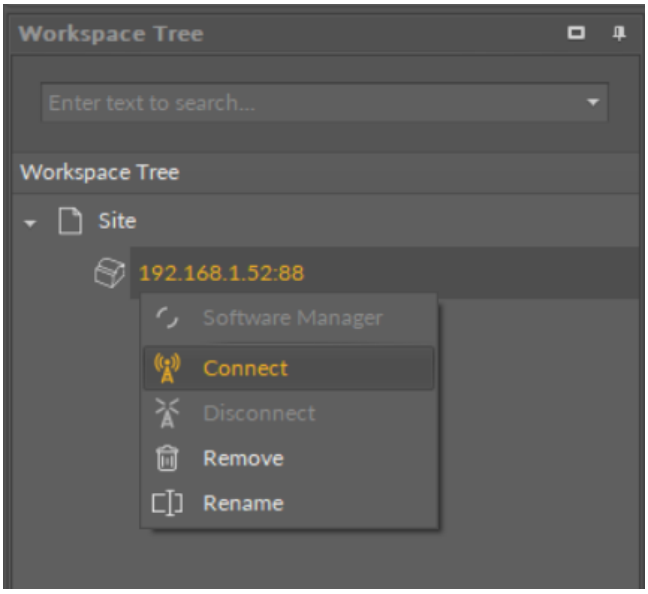


Figure 254. Connecting the device

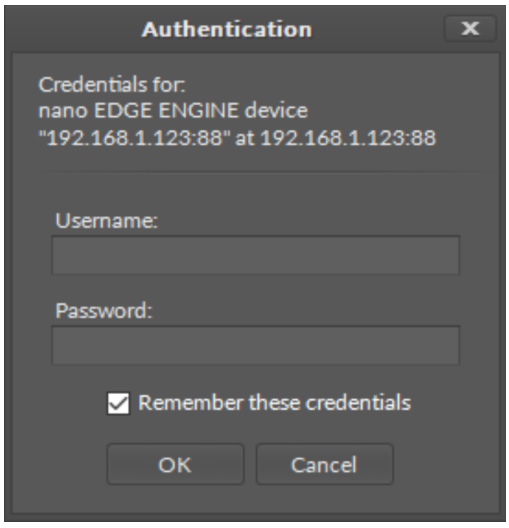


Figure 255. The authentication window

Enter default credentials:

- username: admin
- password: admin

Confirm with OK and proceed to the next authentication step, which is setting an individual user's password. After pressing OK, the following dialog window appears, leading the user to change the password:

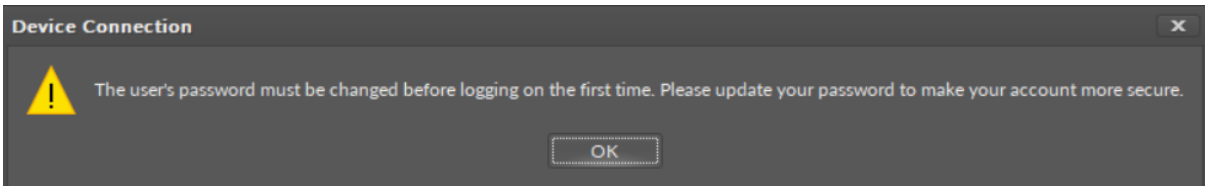


Figure 256. First change of password warning

Confirm the dialog window with OK, and proceed to a Change Password dialog window:

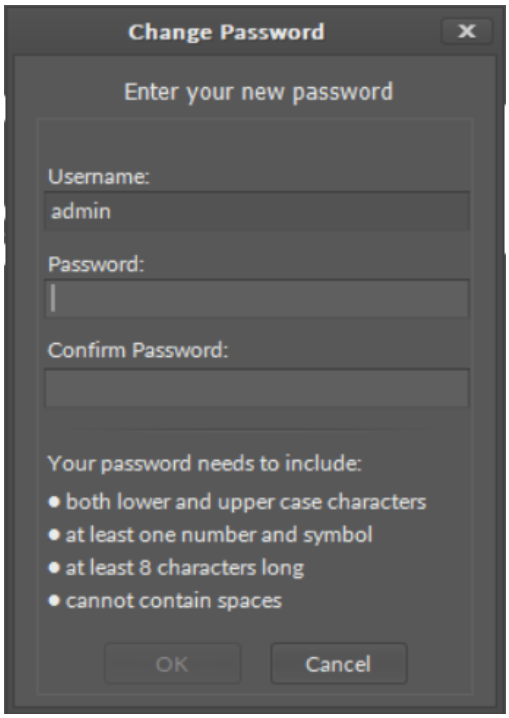


Figure 257. Change Password window

For the first logging in, the user is required to enter:

- Username: admin;
- Password: **new password set by the user, at least 8 characters long, at least one digit, at least one capital letter.**

While typing a new password, the dialog window verifies if the password is compliant with requirements (fulfilled requirement is marked in green):

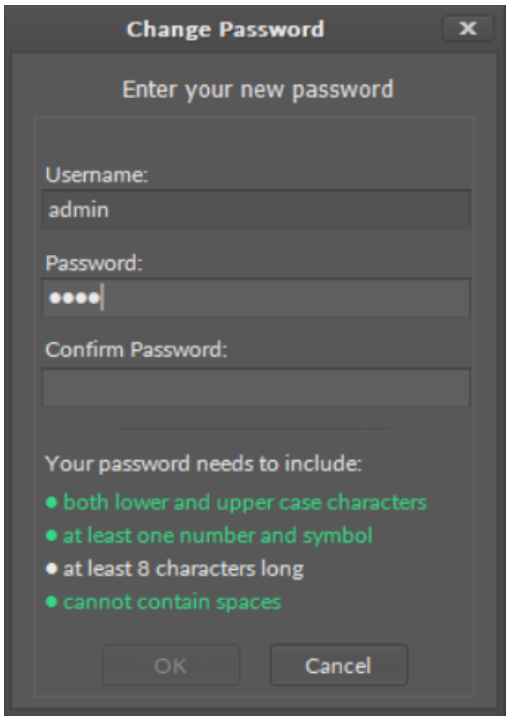


Figure 258. Fulfilled password requirements

Note: If a wrong username or password is inserted in the authentication window, the following notice pops up:

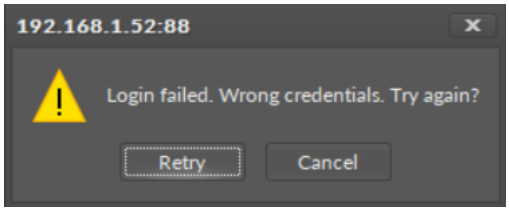


Figure 259. Wrong credentials notice

Click retry and enter correct credentials.

9.5 Logs

Applicable to OS version 1.0.0.4592

The Logs component runs an adjustable register of all events happening in the operating device. Such register becomes crucial when troubleshooting; it enables checking records, comparing recent ones with historical feeds, and, if needed, sharing them with the iSMA CONTROLLI Support Team for further diagnostics.

The records in the Logs register are grouped by their genre; they are categorized depending on the area they originated–firmware, core, BACnet, app, etc. Each group may have its individual log level defining priorities of data to be recorded, according to the users needs. These priorities are differentiated from Debug (each event happening is registered) to Critical (only few events that result in the system error are registered).

The Logs register is written to a file and saved on the SD card in the device. If needed, it may be copied from the SD card and shared with the iSMA CONTROLLI Support Team for troubleshooting.

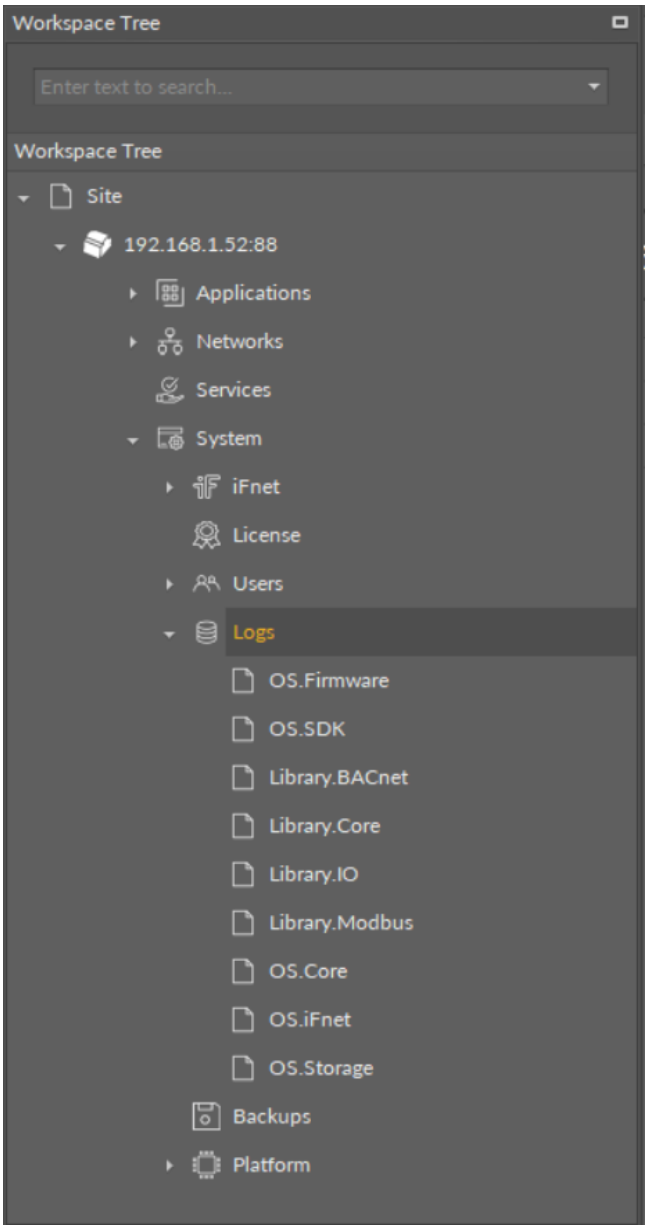


Figure 260. The Logs component

The Logs component has one slot:

- **Default Log Level:** assigns the importance of logs to be registered;
  - Available settings: Debug, Normal, Warning, Important, Error, Critical, Undefined.

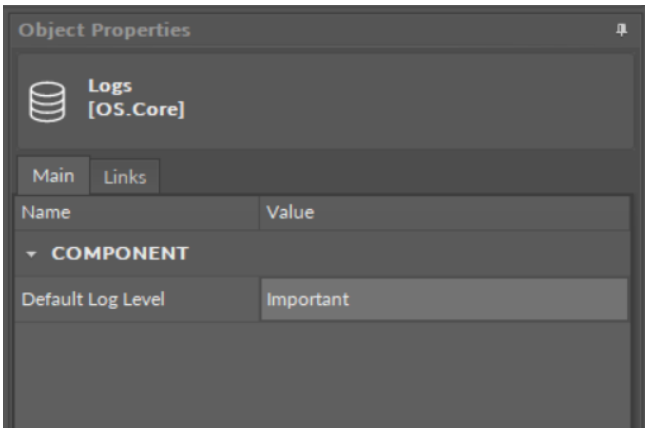


Figure 261. The Logs component slots

Worth to notice

Logs register information about network configuration (IP address, default gateway, mask) providing a convenient way to retrieve such data if lost.

9.5.1 Extensions

The Logs component has also available the component's extensions for each of groups defined for register:

- OS.Firmware;
- OS.SDK;
- Library.BACnet;
- Library.Core (library including Data Points, folders, etc.);
- Library.IO;
- Library.Modbus;
- OS.Core (system elements);
- OS.iFnet;
- OS.Storage.

Each of the above has a Log Level slot, which allows to individually set the importance of logs to be registered.

9.5.2 Log Viewer

Applicable to OS version 1.0.0.4592

The nano EDGE ENGINE Log Viewer provides logs for diagnosing and analyzing the work of the device driven by the nano EDGE ENGINE.

Logs are categorized by their source of occurrence:

- OS.Firmware;
- OS.SDK;
- Library.BACnet;
- Library.Core;
- Library.IO;
- Library.Modbus;
- OS.Core;
- OS.iFnet;
- OS.Storage.

The logs levels are defined in the Logs component. The Log Viewer is the default view of the Logs component.

Opening the Log Viewer

The Log Viewer is accessible by double-clicking the Logs component in the System container in the Workspace Tree window. Alternatively, it can be opened from the context menu of the Logs component, in the Views option.

Upon opening the Log Viewer shows the logs from the current log file, and updates it while running.

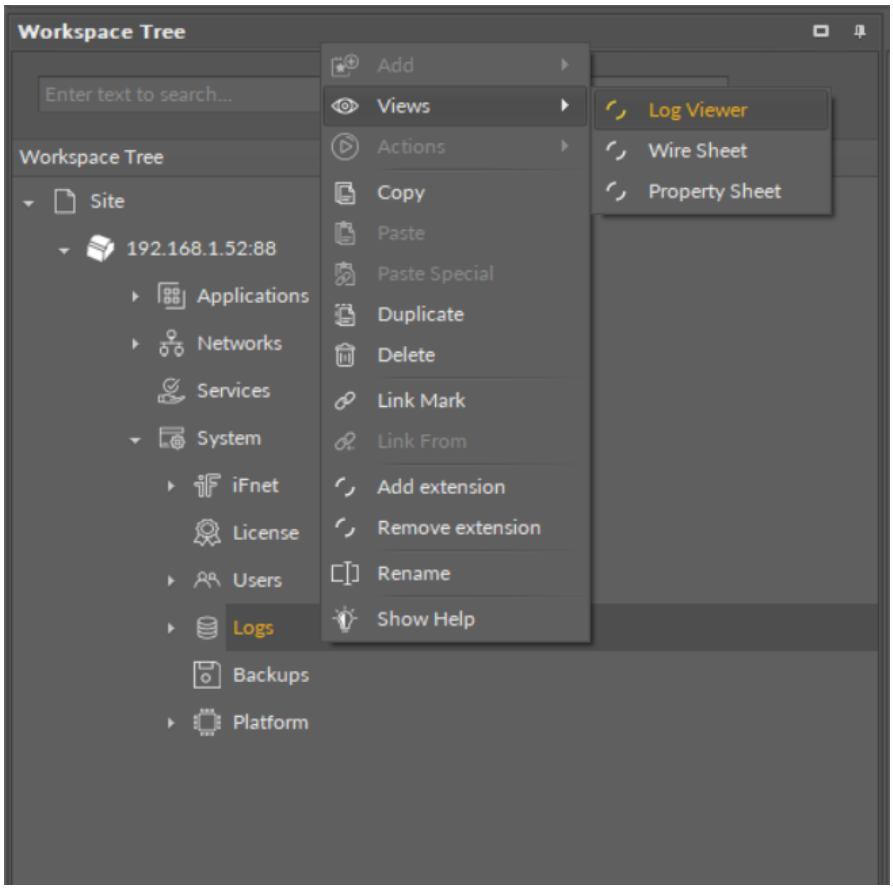


Figure 262. Accessing Log Viewer

Using the Log Viewer

Filtering

Once the Log Viewer is opened, it displays logs according to the configuration of the Logs component. The logs are categorized, and can be filtered by dates. The filtering option is available in the bottom left corner of the window.

Filtering can be time consuming, depending on the selected filtering option, however, it does not have any impact on the work of the iC Tool.

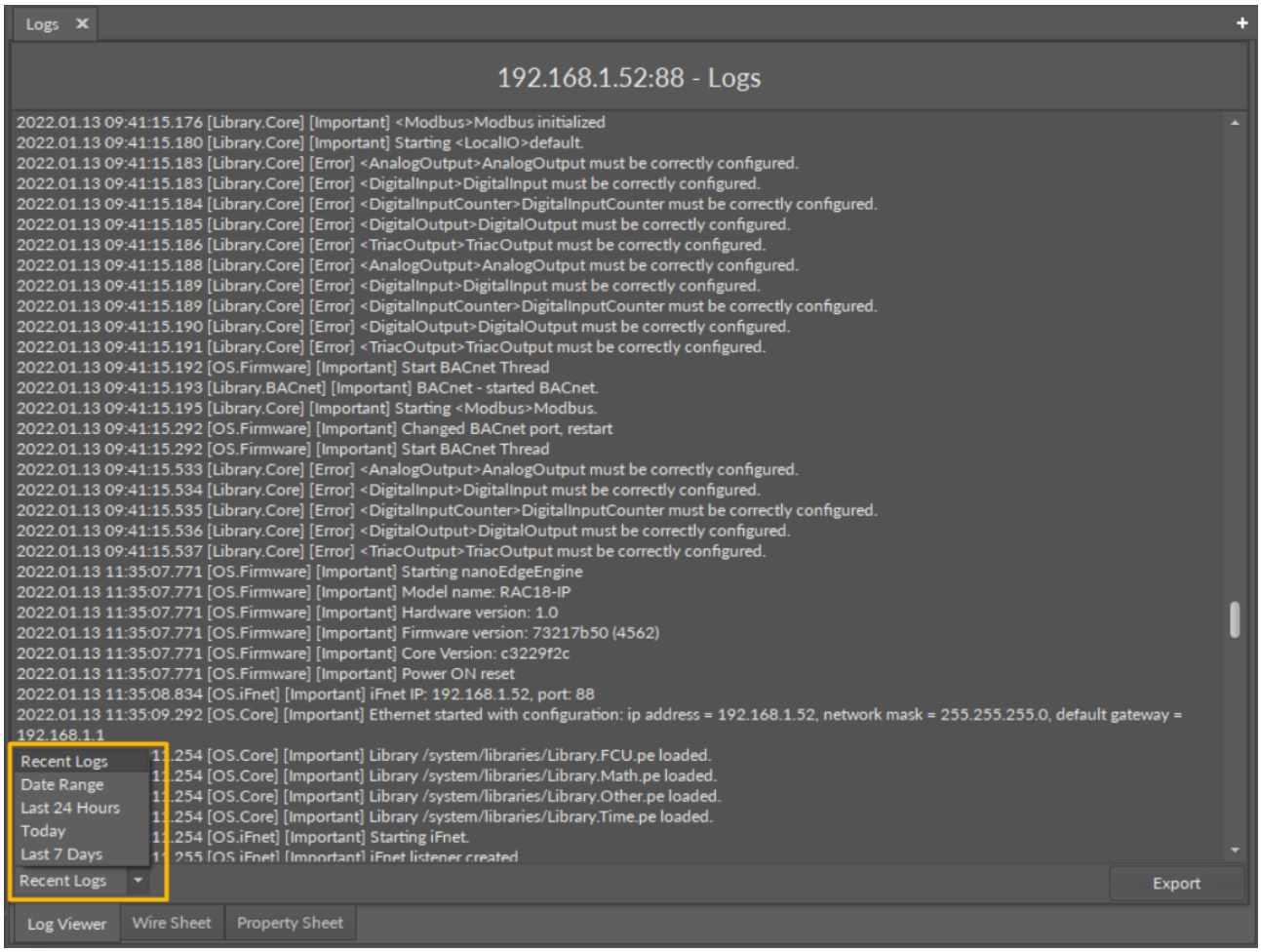


Figure 263. Filtering

Exporting

The contents of the Log Viewer can be exported to a chosen location on the hard drive. The Export options is available in the bottom right corner of the window. The logs file is exported as a .txt file, which name includes the date and hour of the export.

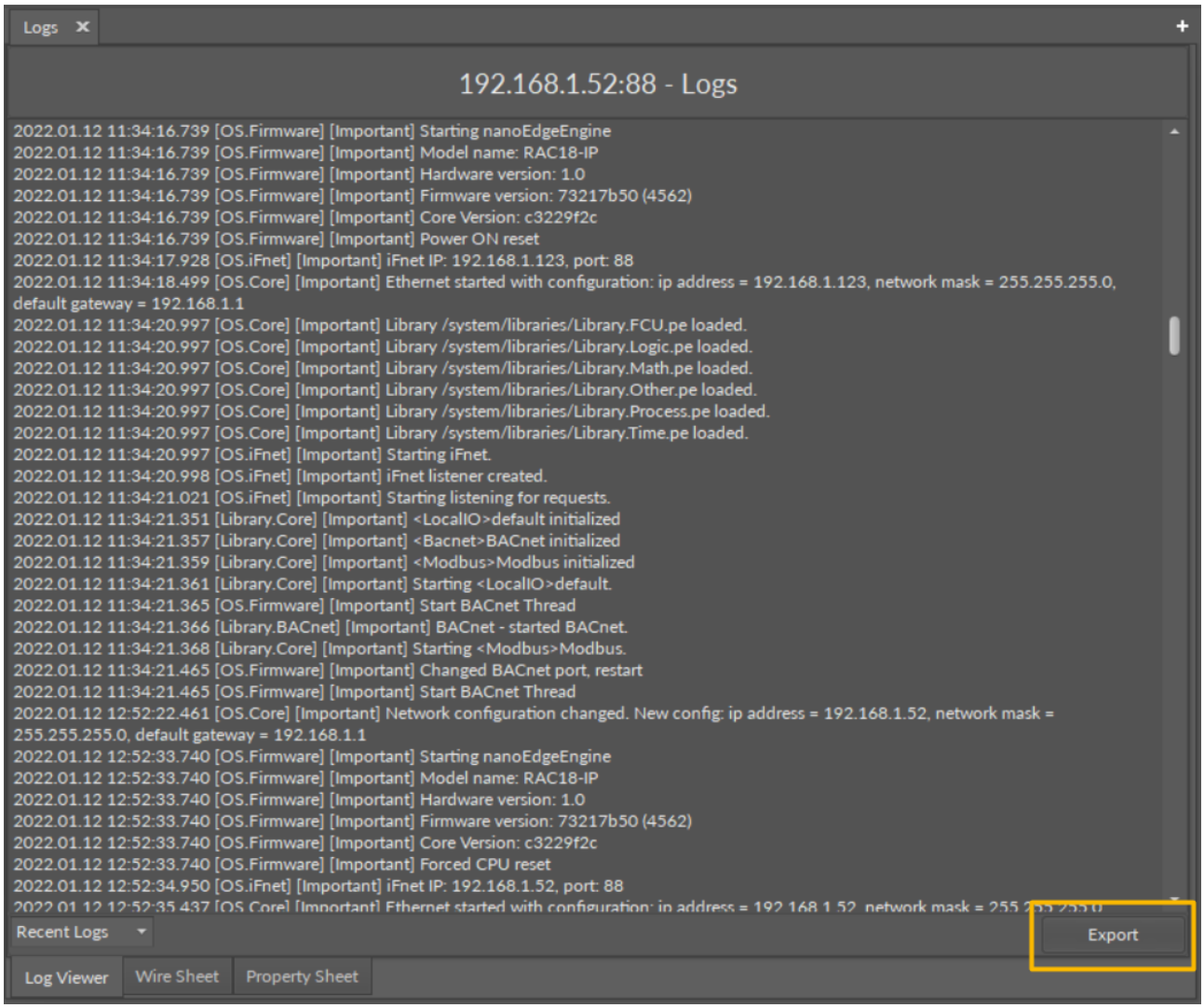


Figure 264. Exporting

9.6 Backups

Applicable to OS version 1.0.0.4592

The Backups component allows to perform and restore backups.

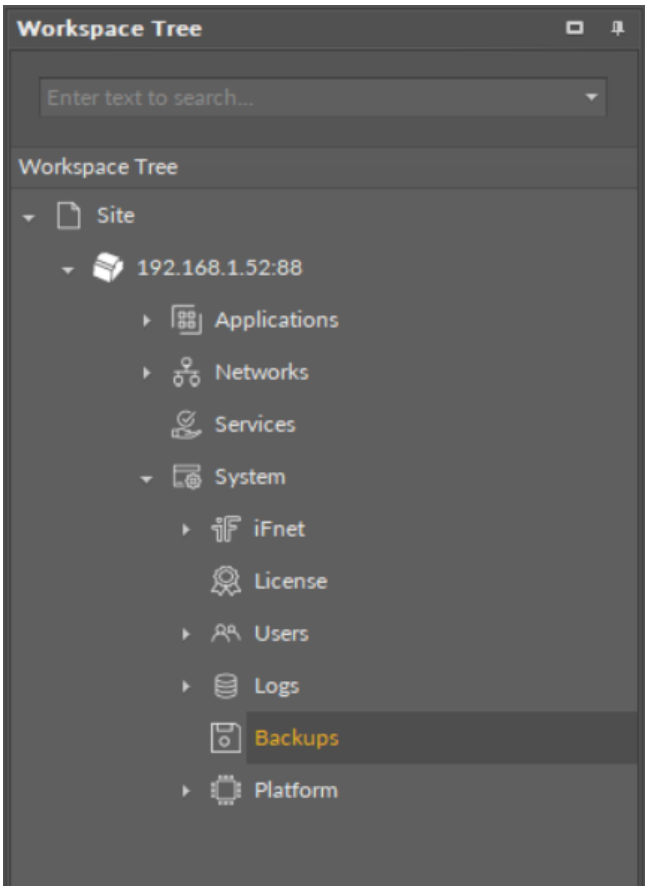


Figure 265. The Backups component

The Backups component has no configurable slots. It is operated in the Backup Manager view:

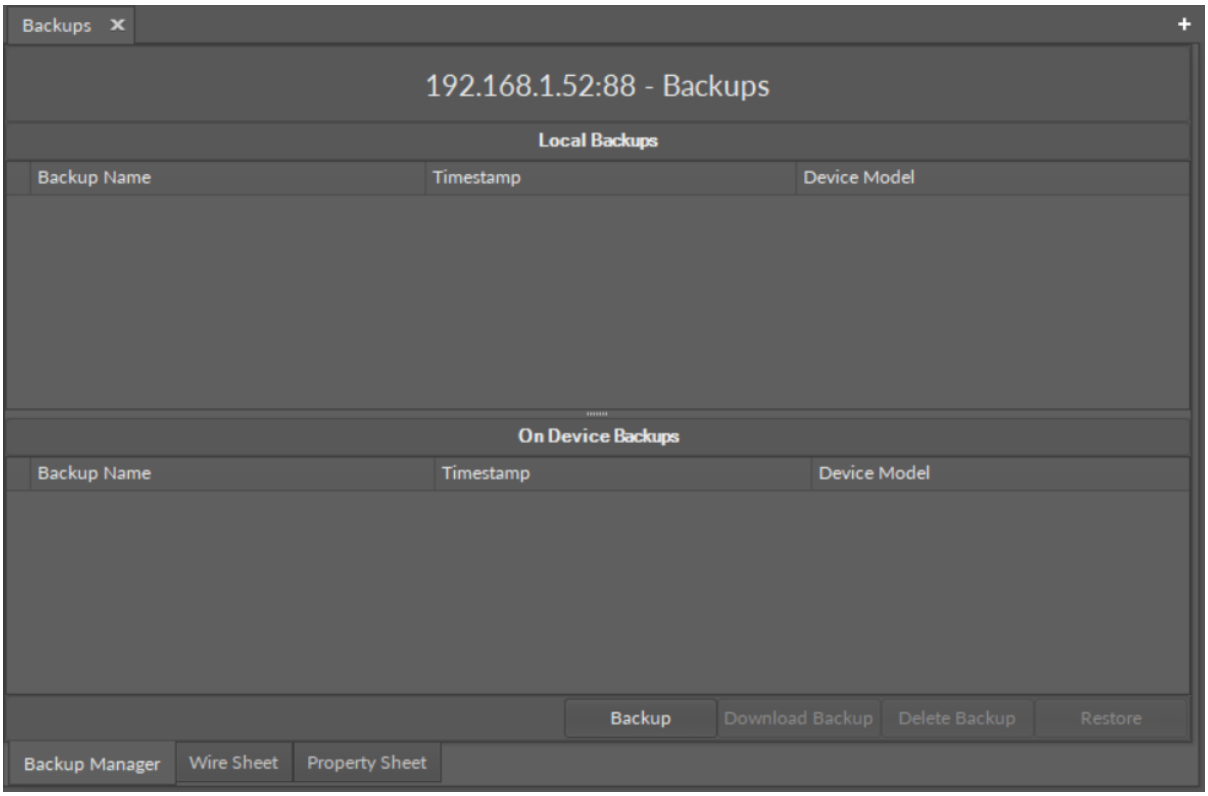


Figure 266. The Backup Manager view

The flowing options are available in the Backup Manager:

- Backup: creates a backup and saves it on the device; each next backup overwrites the last backup unless saved locally with the Download Backup option;

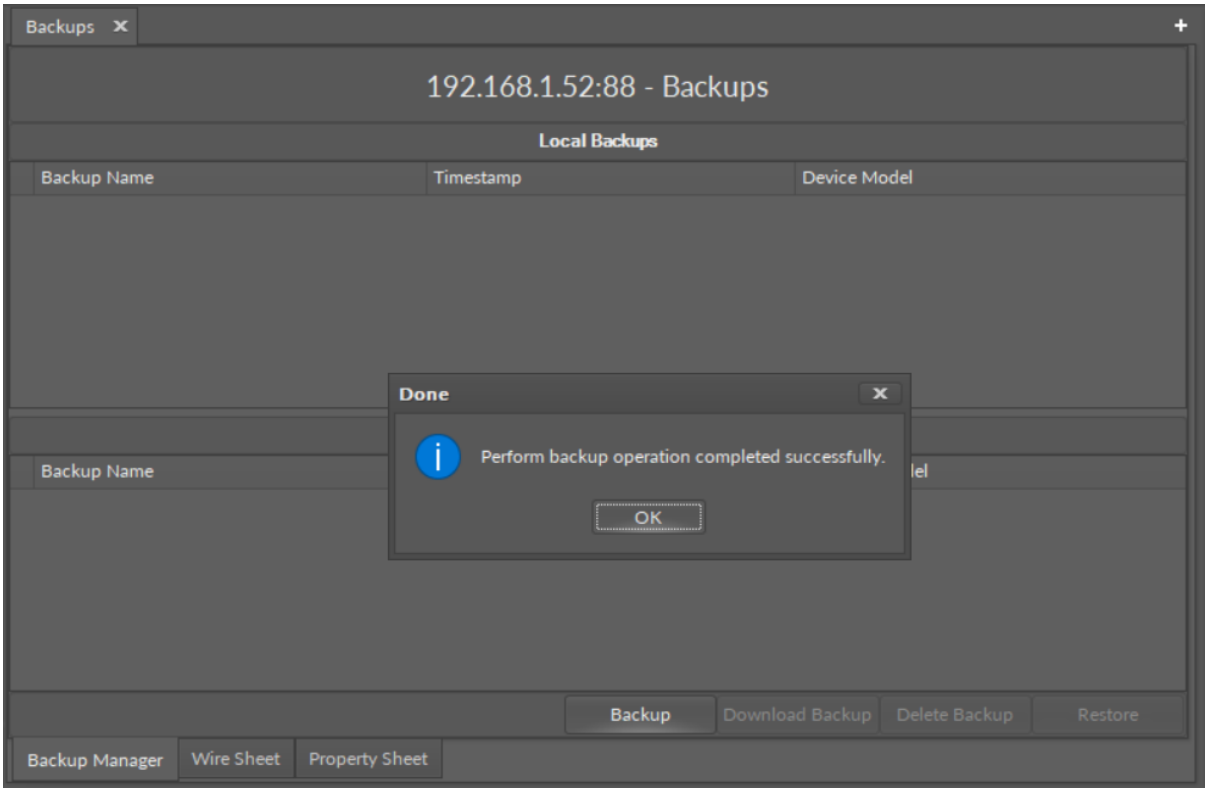


Figure 267. Backup saved on the device

- Download Backup: downloads a backup to the PC and saves it under a selected name in the iSMA Tool's default backup folder (home\backup\nano EDGE ENGINE); after downloading, the backup is displayed in the Local Backups section;

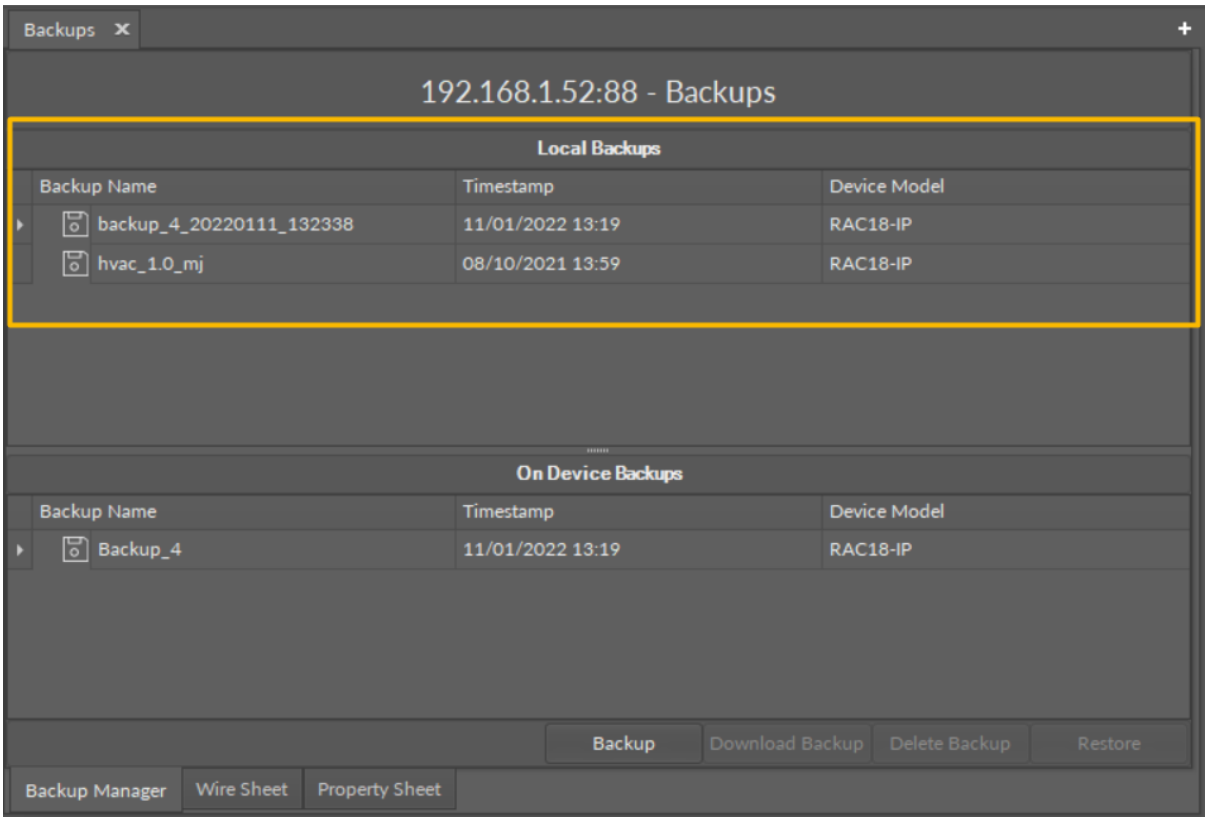


Figure 268. Backups downloaded locally

While downloading a backup, it is possible to set its individual name:

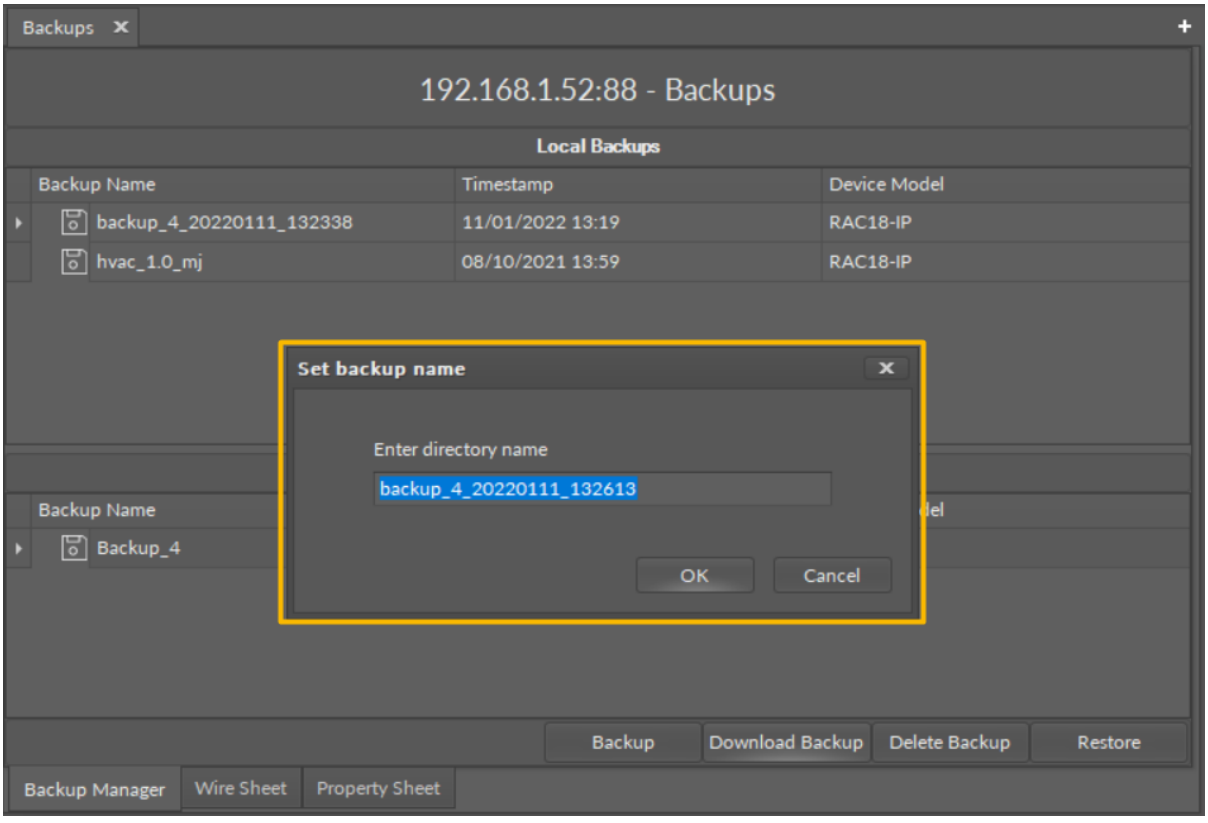


Figure 269. Setting backup's individual name

- Delete Backup: removes a backup from a selected location (PC or device);

- Restore: restores a backup to the device from a selected location (PC or most current backup saved on the device).

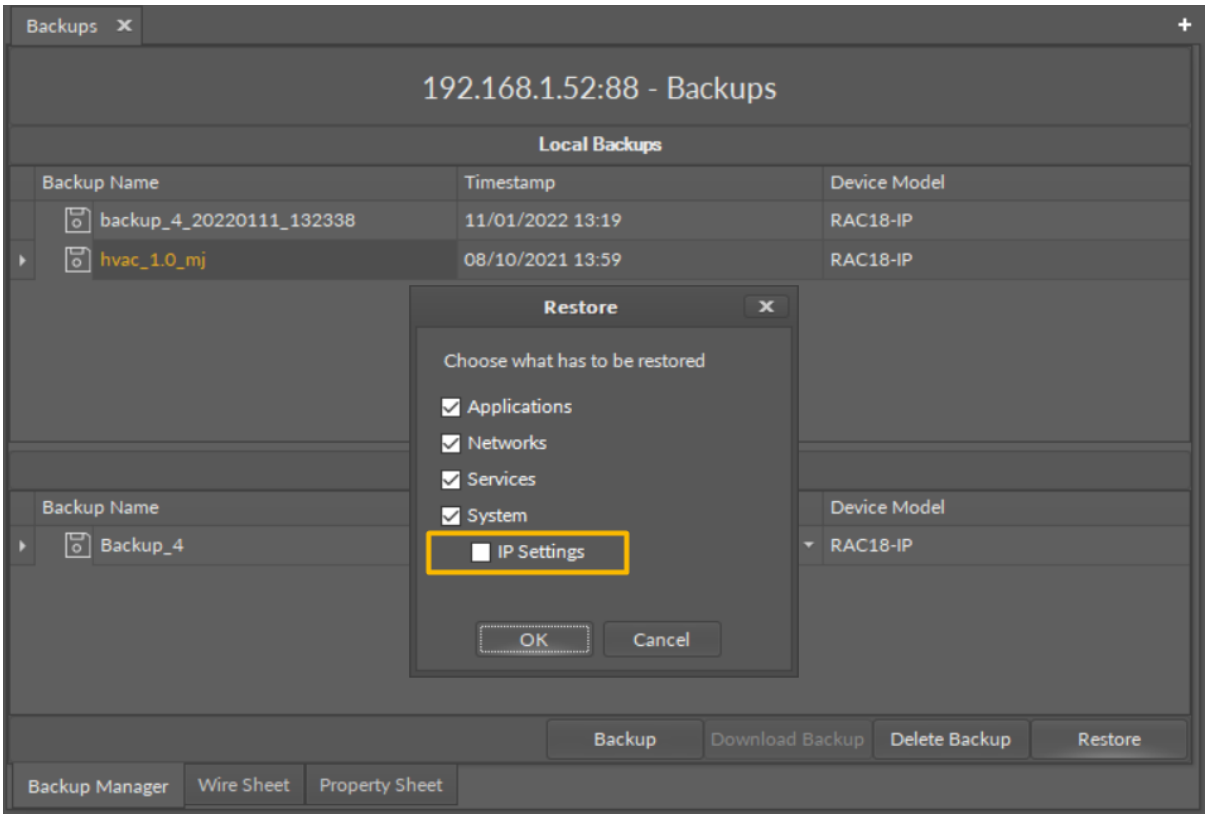


Figure 270. Restoring a backup

While restoring a backup, it is possible to selected whether the IP address is restored from the backup too, or lest as set on the device. By default, the option to restore IP address from a backup is unchecked.

Note: The only thing subject to selection in backup restoring is the IP address. The other items, including users and passwords are restored from the backup.

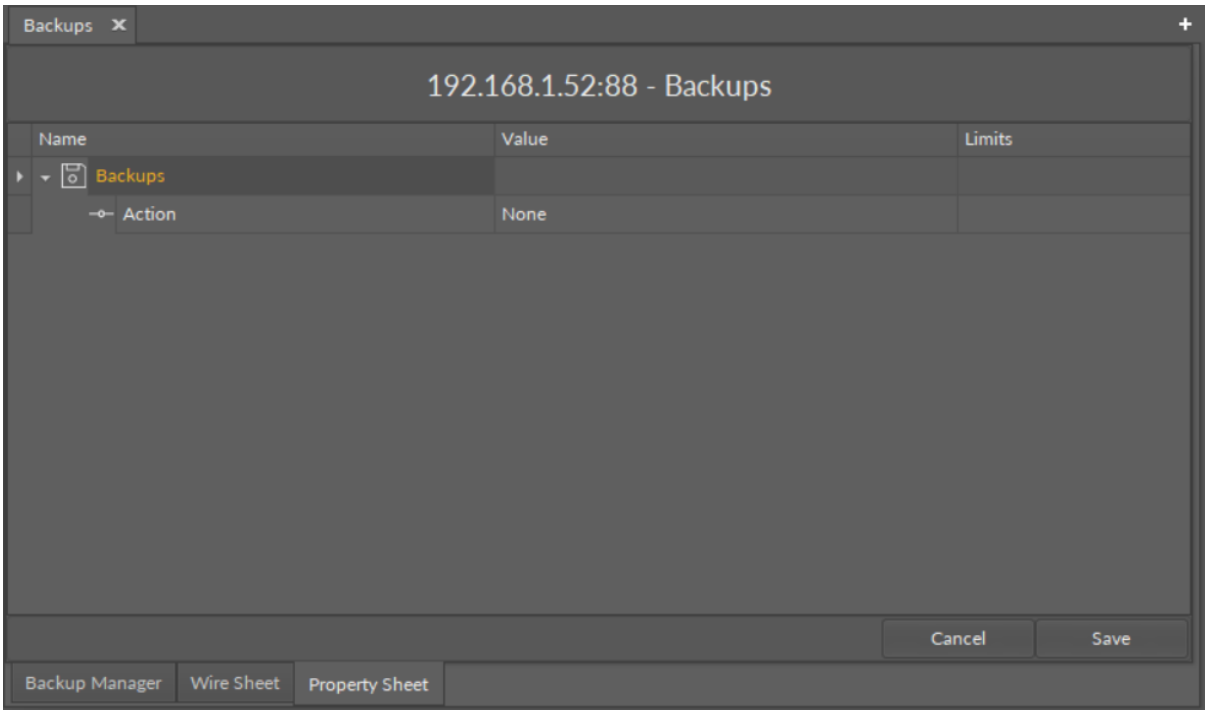


Figure 271. The Backups component's read-only Action slot

The Backups component has one read-only slot:

- Action: informs, which action is currently ongoing.

9.7 Platform

Applicable to OS version 1.0.0.4592

The Platform component includes all the basic information about the device and its operating system. It characterizes the hardware and other features essential for the device to operate correctly.

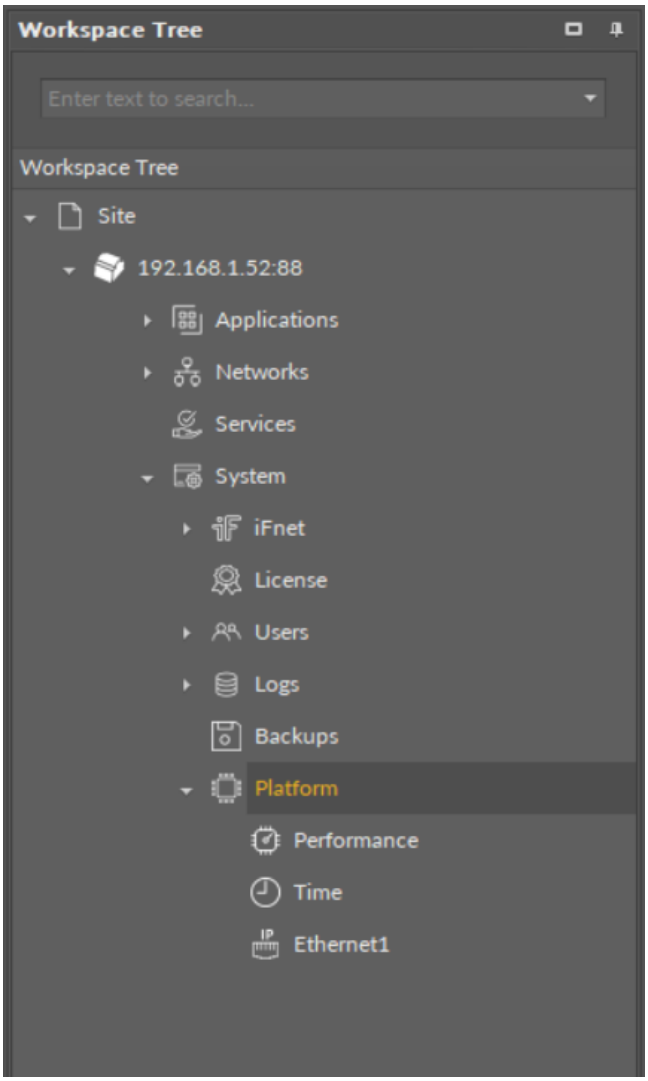


Figure 272. The Platform component

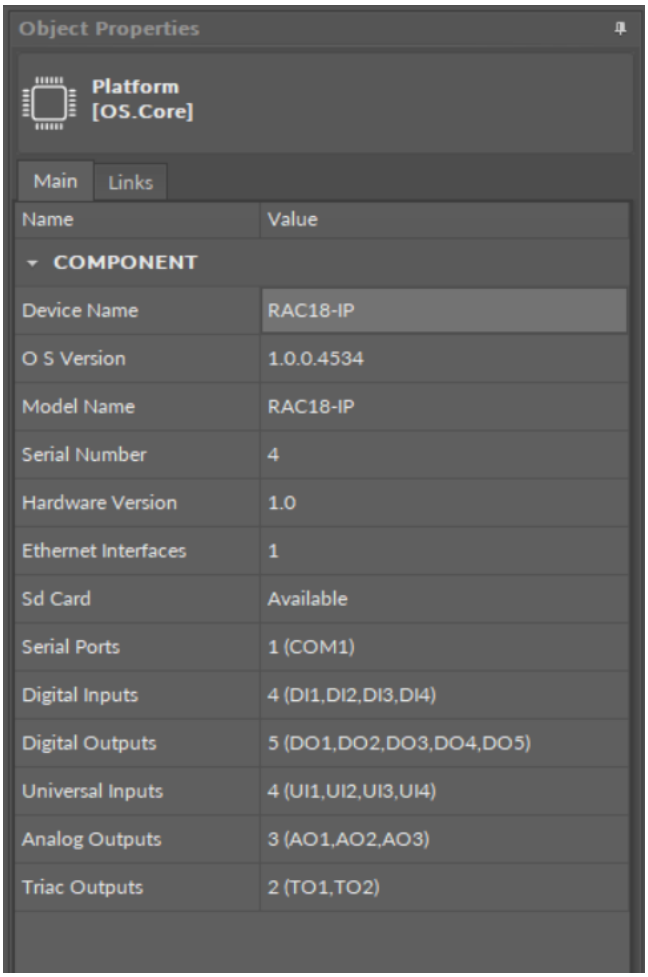


Figure 273. The Platform component slots

The Platform component has the following slots:

- **Device Name:** displays the name of the device, which can be changed manually;

Note: The name displayed by default is based on the Model Name slot. The manual change of name is associated with a corresponding change of the BACnet local device name (LocalDevice component).

- **OS Version:** shows the OS software version currently installed on the device;
- **Model Name:** shows the device model read from the device itself;
- **Serial Number:** shows the serial number of the device;
- **Hardware Version:** shows the hardware version of the device;
- **Ethernet Interfaces:** shows the number of Ethernet ports available in the device;
- **Sd Card:** informs if an SD card is available (available/none);
- **Serial Ports:** shows the number of serial ports available in the device;
- **Digital Inputs:** shows the number of digital inputs in the device;
- **Digital Outputs:** shows the number of digital outputs in the device;
- **Universal Inputs:** shows the number of universal inputs in the device;
- **Analog Outputs:** shows the number of analog outputs in the device;
- **Triac Outputs:** shows the number of triac outputs in the device.

Additionally, the Platform components has also three component's extensions: Performance, Time, and Ethernet.

9.7.1 Performance

Applicable to OS version 1.0.0.4592

The Performance component is a knowledge source about the operation of the device. It provides information about the CPU, memory in use, and transferred data on SD card.

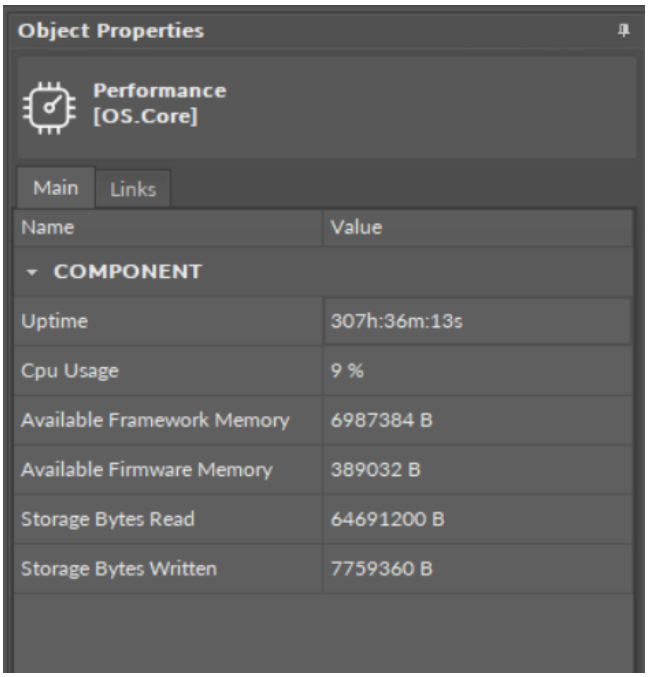


Figure 274. The Performance component

Slots

The Performace component has the following slots:

- **Uptime:** shows the time the device has been working since the last start-up;
- **CPU Usage:** shows the current use of the CPU; in %;
- **Available Framework Memory:** shows the available OS memory in bytes;
- **Available Firmware Memory:** shows the available firmware memory in bytes;
- **Storage Bytes Read:** shows the number of read bytes;
- **Storage Bytes Written:** shows the number of written bytes.

9.7.2 Time

Applicable to OS version 1.0.0.4592

The Time component sets time and date in the device. In case the device is not powered longer than for a week, the time in the device is reset. The Time component allows to manually set local time in the device (with an action button) according to the computer's local time.

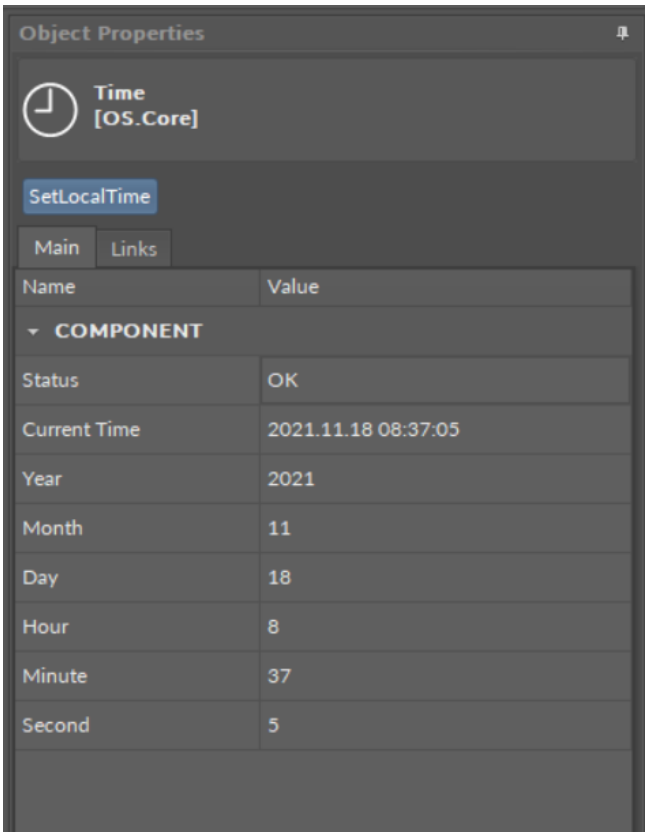


Figure 275. The Time component

Slots

The Time component has the following slots:

- **Status:** indicates the current status of the component;
- **Current Time:** shows the current time set in the device;
- **Year:** shows the current year;
- **Month:** shows the current month;
- **Day:** shows the current day;
- **Hour:** shows the current hour;
- **Minute:** shows the current minutes;
- **Second:** shows the current seconds.

Action

The Time component has one action:

- **SetLocalTime:** set the computer's (hosting the iC Tool) local time to the device.

9.7.3 Ethernet

Applicable to OS version 1.0.0.4592

Warning!

The Ethernet component is always associated with an Ethernet port it manages. It is, therefore, named consecutively according to the number of the physical Ethernet port it manages: Ethernet1, Ethernet2, etc.

The Ethernet component allows to configure three essential features to connect the controller to the network: IP address, mask, and default gateway; it is the most important component in terms of network configuration.

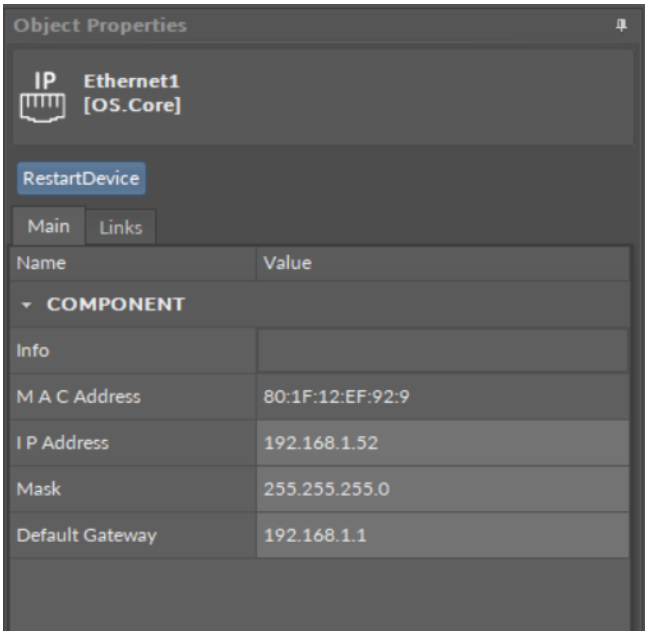


Figure 276. The Ethernet component

Slots

The Ethernet component has the following slots:

- **Info:** informs about a required device restart after changing one of the parameters (displayed after a new information is saved to the device);

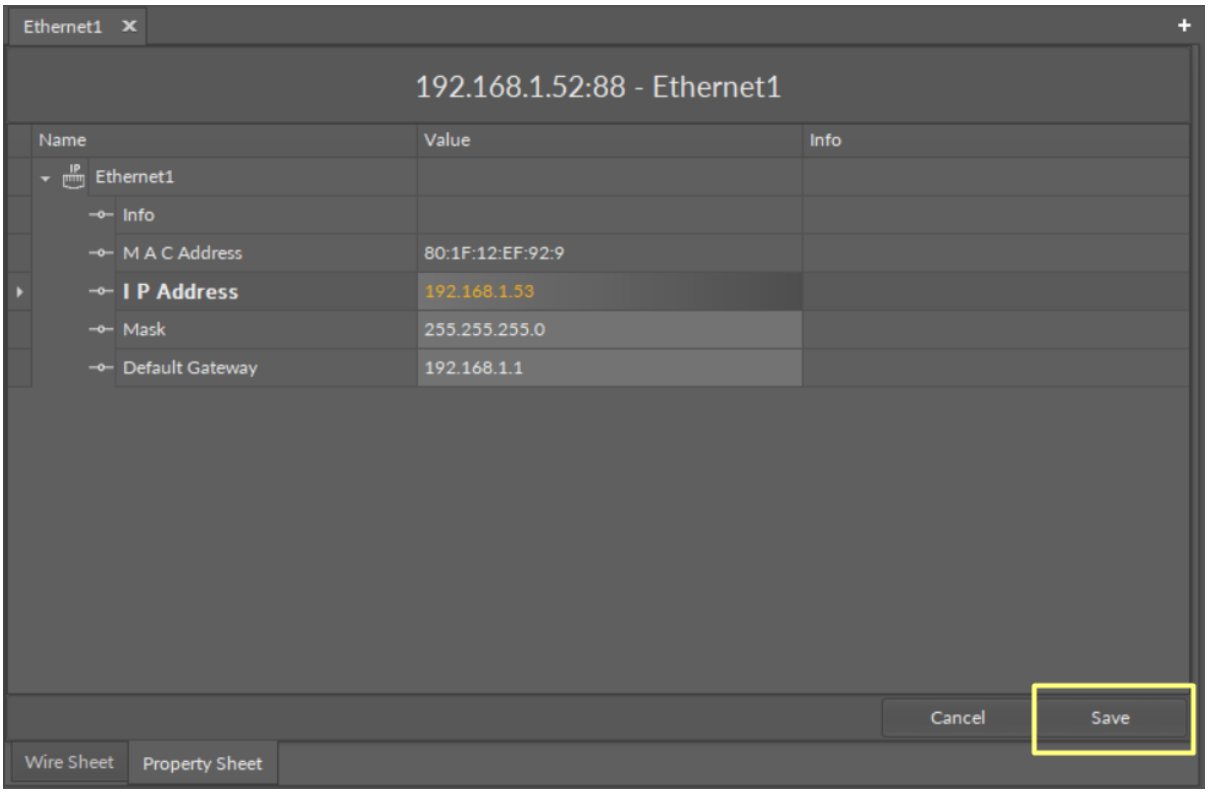


Figure 277. New information introduced to the Ethernet component (new IP address)

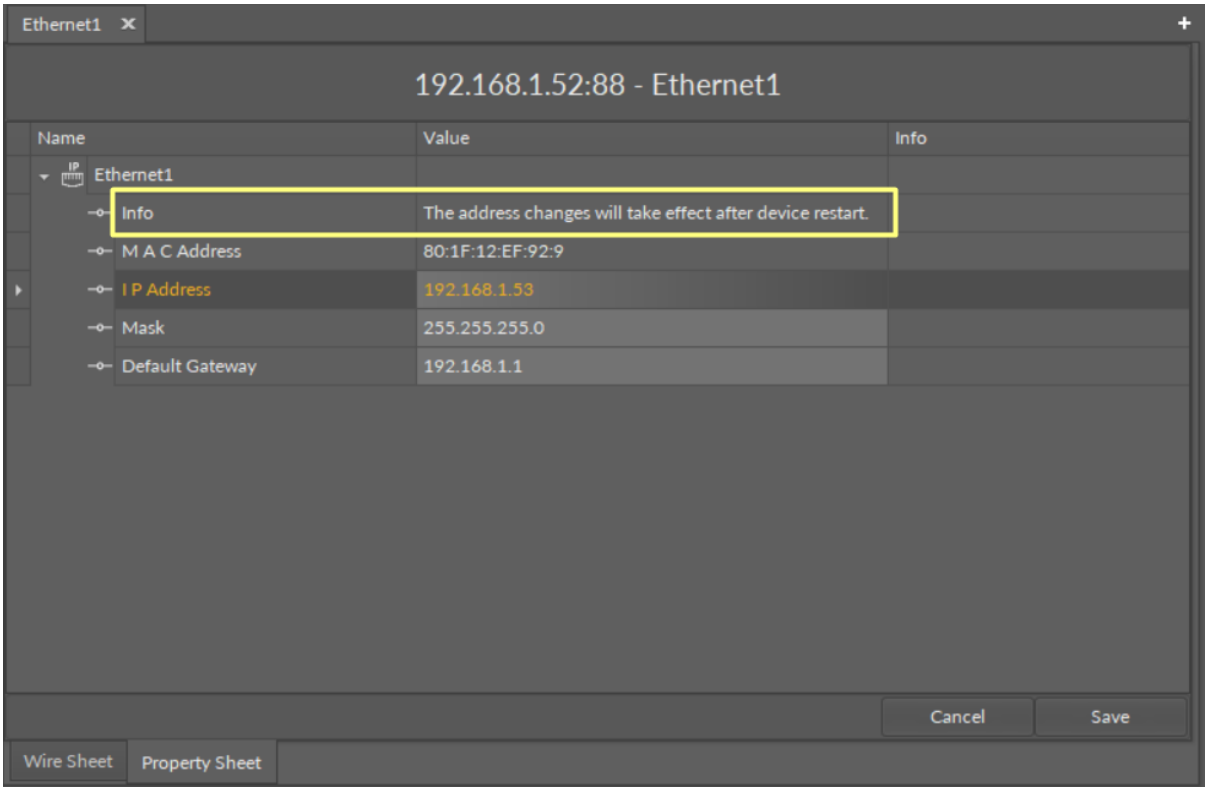


Figure 278. The notice in the Info slot

- **IP Address:** allows to set the IP address of the device; by default the address is set to 192.168.1.123;
- **Mask:** allows to set the mask of the device; by default the mask is set to 255.255.255.0;
- **Default Gateway:** allows to set the default gateway of the device; by default it is set to 192.168.1.1;

- **MAC Address:** shows the MAC address of the device.

> **Note:** Each device has a default, constant IP address: 198.162.1.123. In order to connect the device to the network it has to work with, the IP address may be set manually by the user according to the user's network requirements. It may also be left as is. The IP address setting is totally depending on the user's requirements.

10 RAC18-IP Quick Start-up

Start working with your RAC18-IP device in just 8 easy steps:

Step 1: Unbox the device.

Step 2: Connect power supply. The RAC18-IP device requires 24 V AC/DC power supply.

Step 3: If needed, connect the device to the network. The RAC18-IP device is equipped with the Ethernet IP fail-safe protected ports, RS485 to connect with other devices in the network, and RJ45 to connect with external panels.

Step 4: Make sure that the dowloaded software is the iC Tool.

Step 5: Open the iC Tool and add the device to the project: open the context menu in the Workspace Tree, add New Project, then open the context menu of the added project, and select the Add Device option. In the dialog window that pops up, set the device's port number (the default port number is 88). In this window, the type of device cannot be changed (it is iFnet type). The default IP address of the device is 192.168.1.123, and it can be changed in the Ethernet component.

Step 6: Connect with the device. Once the device is added, the Authentication dialog window pops up. The default credentials are:

- username: admin
- password: admin

Worth to Notice:

When logging in to a factory-set or reset device, it is required to change password. Please refer to First Logging In.

Step 7: Go to the System container in the Workspace Tree and configure essential properties in the Platform component. The Platform component provides essential data about the device, and it also includes the Ethernet component, where the IP address, mask, and default gateway can be changed. When setting the network parameters of the device, make sure that they are consistent with the computer network parameters that the device is connected to (more on this: Address Pool Consistency).

Step 8: The device is now ready to work. Start creating applications: go to the Applications container, and add as many Application components as there are user applications to be created. Then go to the Device Libraries, and drag the needed components to the Application components. The Core library is the one containing the Application component itself, licensed Data Points, and Folder component, which helps to organize the contents in the Applications container.