Programmable Events module manual

Version 1.0

REV00-20160630

GENERAL INFORMATION

DIVUS GmbH
Pillhof 51
I-39057 Eppan (BZ) - Italy

You are always welcome to inform us of errors or make suggestions for improving the program.

The present agreement also applies to special appendices to the manual.

This manual can contain terms and descriptions, which improper use by third can harm the copyrights of the author.

Please read the manual before beginning and keep the manual for later use.

The manual has been conceived and written for users who are experienced in the use of PCs and automation technology.

CONVENTIONS

[KEYS]	Keys that are to be pressed by the user are given in square brackets, e.g. [CTRL] or [DEL]
COURIER	On-screen messages are given in the Courier font, e.g. C:\>
COURIER BOLD	Keyboard input to be made by the user are given in Courier bold, e.g. C:\>DIR).
„…“	Names of buttons to be pressed, menus or other onscreen elements and product names are given within double quotes. (e.g. "Configuration").
PICTOGRAMS	In this manual the following symbolic are used to indicate particular text blocs.
⚠	*Caution!* A dangerous situation may arise that may cause damage to material.
ℹ	*Hint* Hints and additional notes
NEW	*New* New features

INDEX

1 Programmable Events - Introduction

NEW

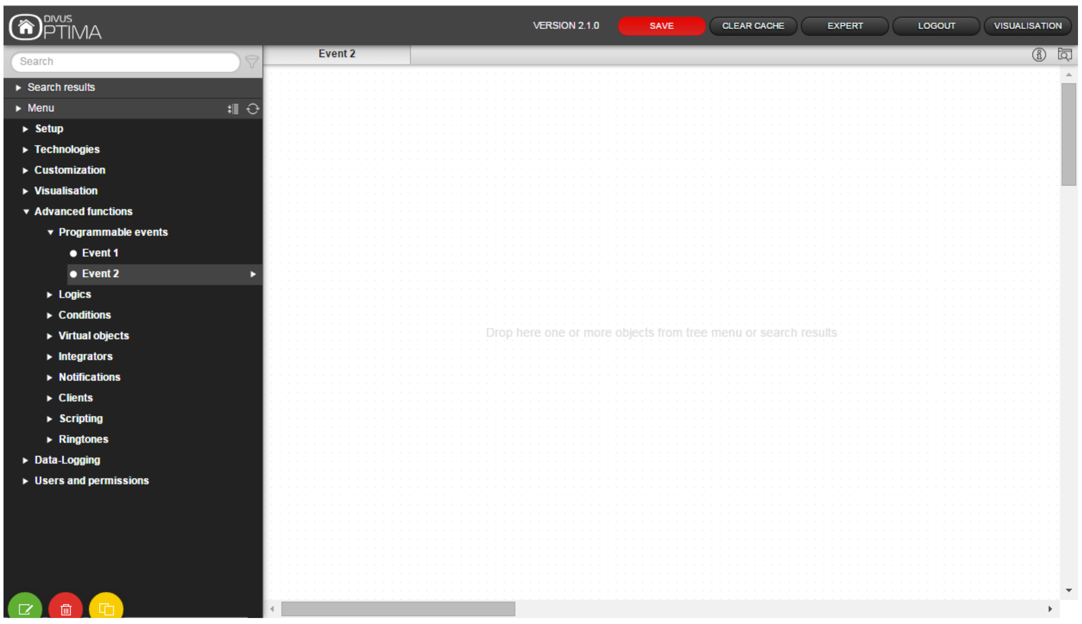
1.1 INTRODUCTION

Since version 2.1 of OPTIMA the possibility to plan and test parts of the system graphically has been introduced.

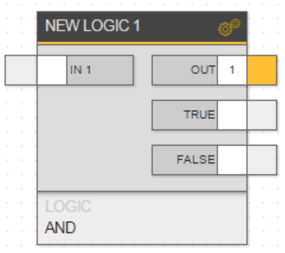
This has multiple advantages, mainly:

- The graphical representation helps to keep the overview for complex relations/dependencies between objects
- The changes of one value, which triggers other commands in a more or less long chain, can be visually checked and tested (more about this in chapter 2.5)
- The work on the project is more efficient because the need to switch between the single objects' detail views is strongly reduced.

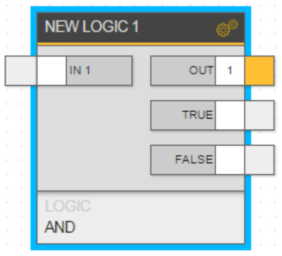
The PROGRAMMABLE EVENTS may be found in the administration area under Advanced Functions. After creating and opening a new Programmable Event, this is initially shown in this way:



Once you put e.g. a logic inside the white window using drag & drop, it will be shown in this way:



not selected



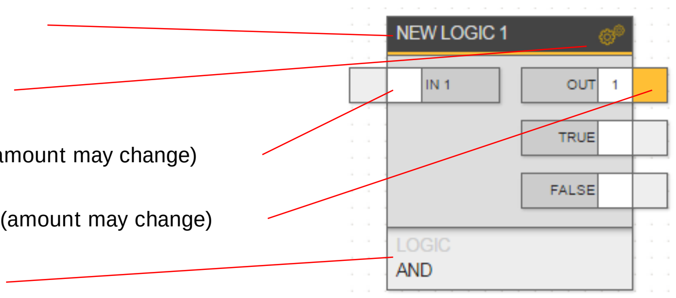
selected

1.2 OBJECTS IN THE PROGRAMMABLE EVENTS WINDOW

1.2.1 GRAPHICAL REPRESENTATION

In this chapter we will take a closer look at the peculiarities of the objects as shown in this window. Although there are differences between different object types, they all have some elements in common:

1. title bar
2. icon
3. inputs (amount may change)
4. outputs (amount may change)
5. footer



The title bar obviously is used to distinguish the object from others.

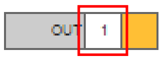
The icon not only shows the type of objects directly, it also shows the current status through its colour.

Inputs and outputs are shown as bars which partially protrude on the left and right side of the main rectangle.

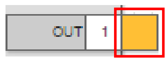
Inputs are on the left side, outputs on the right.

From there you create relations to other objects per drag & drop. More about that in chapter 2.2. There is a central section in the bar for the value which comes in or goes out as well as a section for a descriptive label (e.g. *FALSE*, *TRUE*, *IN 1*, *OUT* – see above).

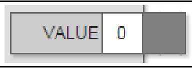
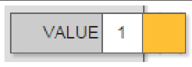
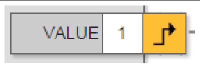
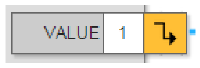
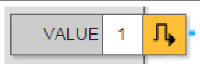
The amount of inputs and outputs is usually not limited. Whenever you create a relation connecting an input with an output, a new, unconnected input (or output) is automatically added below – ready for the next connection.



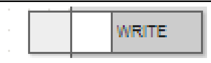
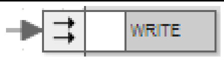
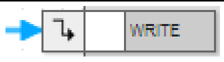
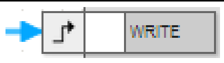
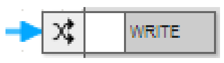
The small input field for the value of an output has also a useful function for testing purpose: by double-clicking you may switch the value directly (0/1) or enter the field value manually (i.e. per keyboard), thus testing how a value change reflects onto the connected objects. See chapter 2.5 for more details.



The last part of the output bar may hold different information depending on its current state:

	The output in not connected. Value is 0/OFF – grey.
	The output in not connected. Value is 1/ON – coloured. (Each function has its distinctive colour. See the OPTIMA User Manual chapter 3 for more details)
	The output is connected. The condition is set to "when 1/ON" meaning the value will be passed on to the other object's input only in that case.
	The output is connected. The condition is set to "when 0/OFF" meaning the value will be passed on to the other object's input only in that case.
	The output is connected. The condition is set to "every status change" meaning the value will be passed on to the other object's input whenever the value of the output object changes.

Also the first section of an input bar shows different symbols depending on the state and type of connection:

	Input is not connected, no value passed.
	Input is connected, the passed value is set to "current value"
	Input is connected, the passed value is set to "OFF (0)"
	Input is connected, the passed value is set to "ON (1)"
	Input is connected, the passed value is set to "inverted value"

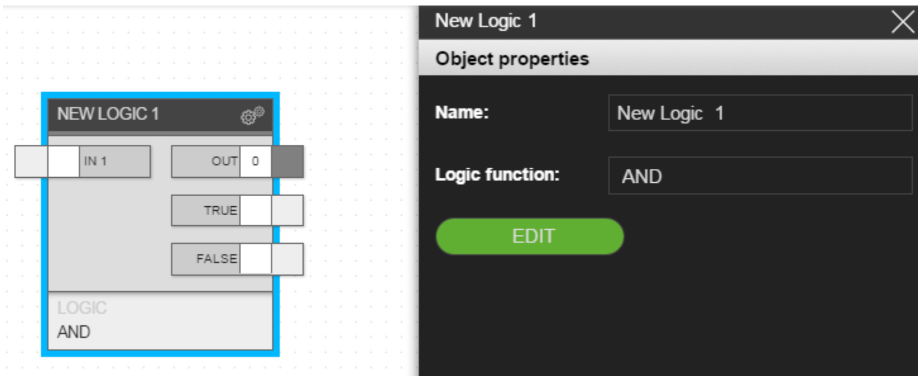
1. In the footer you'll find

   - The object type (grey)
   - The current setting of the main property – different for every object type (black)

In the example above the object type is "Logic" and the main property (logic function) is set to *AND* meaning that all the input objects will be logically connected with AND operators and the result will be made available as the object's output value.

1.2.2 LATERAL DETAIL WINDOW

When you click on an object with the right mouse button, a window will slide in from the right side, showing the main properties of the selected object and giving the possibility to change those values. If there is the need to change properties which are not shown, the object's detail window may be reached over the green EDIT button. More about the detail windows of OPTIMA's object types may be found in the OPTIMA Administrator Manual.

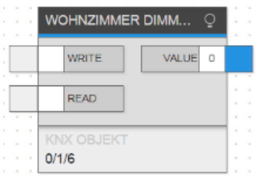
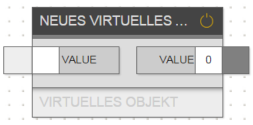
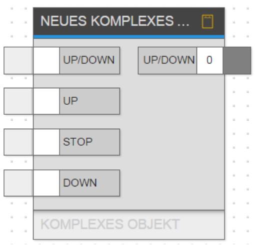


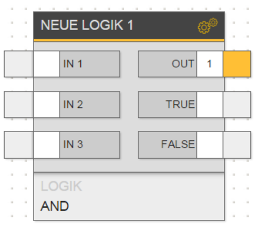
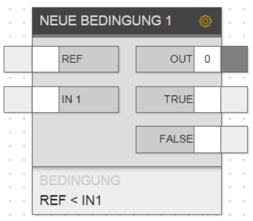
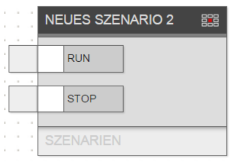
Name: New Logic 1

Logic function: AND

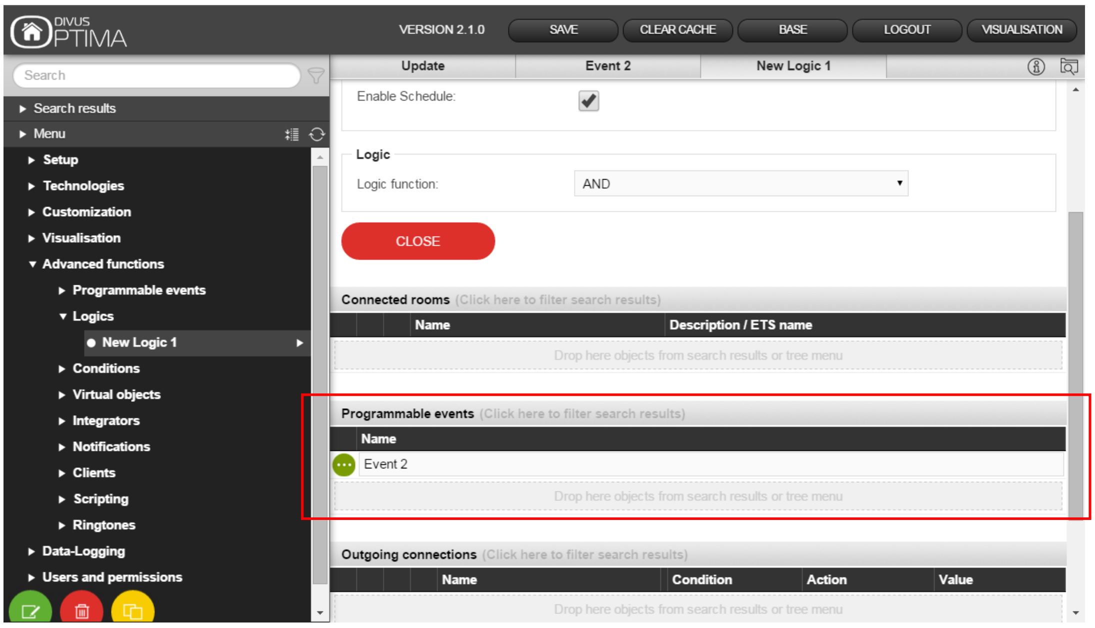
Like in the default detail windows in OPTIMA, also here the BASE/EXPERT setting in the upper menu bar changes the amount of shown properties: more advanced or less often used settings are shown only when the EXPERT mode is active.

1.3 UNTERSTÜTZTE OBJEKTE

Object	Representation (example)	Main property	Notes
KNX Object		Group address	• Object must be visible • Icon must be assigned
Virtual Object		---	• Object must be visible • A value must have been assigned to the object.
Complex Object		---	• Available inputs/outputs depend on the object's type

Logic	 NEUE LOGIK 1 IN 1 IN 2 IN 3 OUT 1 TRUE FALSE LOGIK AND	Logical operator (*AND*, *OR*, *NOT*)	• *NOT* is available only with one single input
Condition	 NEUE BEDINGUNG 1 REF IN 1 OUT 0 TRUE FALSE BEDINGUNG REF < IN1	Type of condition	• The amount of needed inputs (1 or 2) depends on the type of condition • *REF* is short for reference object – the one which is compared to the input(s).
Scenario		---	• Has only inputs. May be played or stopped (interrupted during play) from other objects.

Whether an object is available for the usage in *Programmable Events* is also recognizable from the object's detail window. There you may find a section in the bottom part of the window:



Enable Schedule: ✓

Logic

Logic function: AND

Connected rooms (Click here to filter search results)

Name	Description / ETS name
Drop here objects from search results or tree menu	

Programmable events (Click here to filter search results)

Name
Event 2
Drop here objects from search results or tree menu

Outgoing connections (Click here to filter search results)

Name	Condition	Action	Value
Drop here objects from search results or tree menu			

Like elsewhere in OPTIMA, these relations between objects can be created in both directions. You may drag an object to the Event's window, like explained before, or you may also drag the event to the section shown here.

The next time the Programmable Event is opened, the object to which it was connected here will show up. There it will be available to connections with other objects.

Single objects cannot be added to the same Programmable Event more than once. That leads to an error message!

2 Creating new Programmable Events

The procedure to create a new Programmable Event is the following:

1. Create a new Programmable Event: Advanced Functions – Programmable Events – ●
2. Create the needed objects, if not already available (see chapter 1.3)
3. Drag the needed objects onto the previously created Event's area.
4. Connect the objects as planned (see chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**)
5. If needed, test directly inside the event's window (see chapter 2.5)

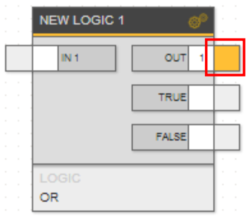
2.1 PREPARING THE NEEDED OBJECTS

If the process you planned to realise through the Programmable Event (one or more) needs its own logics, conditions or scenarios, you first have to create them. Go to the section of those objects in the left menu and add new objects using the ● button. More detailled information about the general usage of OPTIMA may be found in the OPTIMA Administrator Manual.
Once all objects were created, drag them onto the opened Event window and distribute them such that all of them are visible and in the correct order.

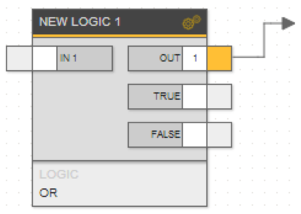
2.2 CONNECTING OUTPUTS WITH INPUTS

The action through which objects are connected, thus allowing one value's change to trigger other actions on connected objects following a certain logic, is really simple:

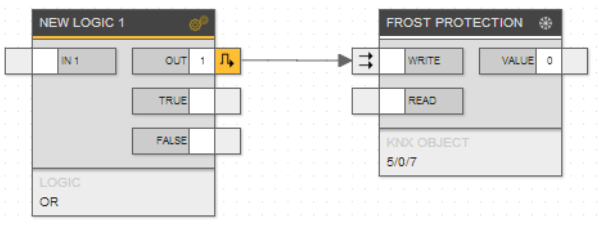
1. You move the two objects to be connected such that the triggering one is on the left and the one to be triggered is on the right (i.e. the outputs of the first are in front of the inputs of the second)
2. Click on the external part of the output of the left object and keep the mouse button pressed



3. From there you move to the input of the second object with your pointing device. When starting to move, an arrow will show up which represents the connection graphically.



4. When you are over the external part of the input of the second object, you can release the mouse button. The arrow will remain connected to the two objects, also when moving the objects.



If you decide to interrupt a connection operation, just release the mouse button over the free white area.

To remove an existing connection you may:

- Click on the external part of the input (or output)...
- ...or select the arrow (which will become light blue) and remove it using the keyboard's **[DEL]** button.

Warning: When using Logics or Conditions in Programmable Events, you should **carefully avoid to use the same Logics/Conditions in multiple Events**; that may lead to unpredictable results! A good way to avoid possible issues is to start by naming the objects with a reference to the Programmable Event they're used in, thus making the relations more easily recognizable. A naming scheme for an Event with e.g. two logics and one condition could look like this:

Programmable Event:	***PE01*** *light-sensor-logic*
Condition (1):	*Condition01* ***PE01***
Logic (1):	*Logic01* ***PE01***
Logic (2):	*Logic02* ***PE01***

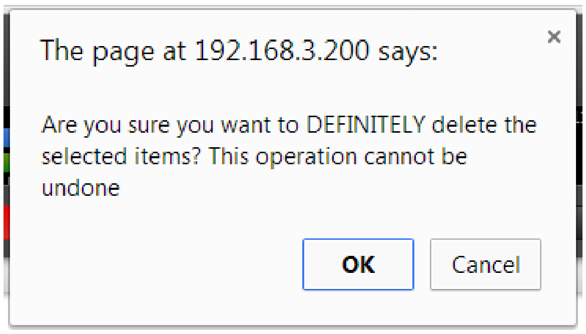
2.3 EDITING OBJECTS INSIDE A PROGRAMMABLE EVENT

Editing inside the Programmable Event window is possible using the lateral detail window. It appears when you click on an object with the right mouse button. Then you can also use the left button to click to another object: the properties shown will automatically switch to those of the new selection.
The properties which can be changed in the right detail window depend on the object type and on the BASE/EXPERT view setting in the upper menu bar. There are the name, the description and one or more other

settings which can be changed directly. For other settings there is the green EDIT button which opens the object's complete detail window.

2.4 REMOVING OBJECTS FROM A PROGRAMMABLE EVENT

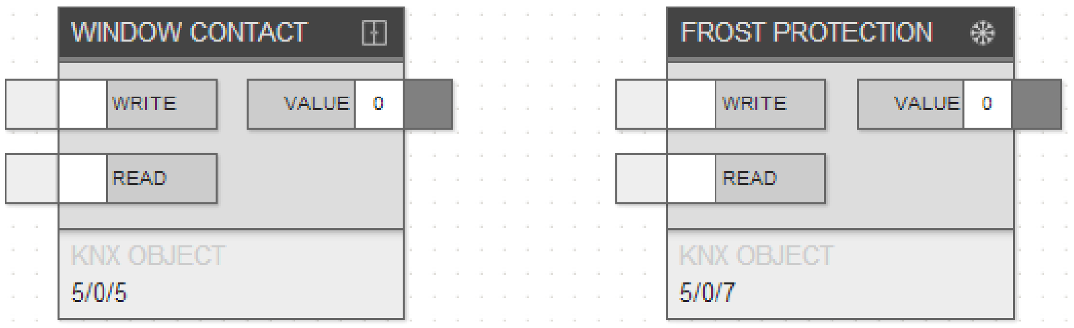
The removal is like the one you already know from your file system: you select an object (like you do with a file or folder) and push the [DEL] button on your keyboard. A confirmation dialog will remove the object definitely or cancel the removal.


The page at 192.168.3.200 says:

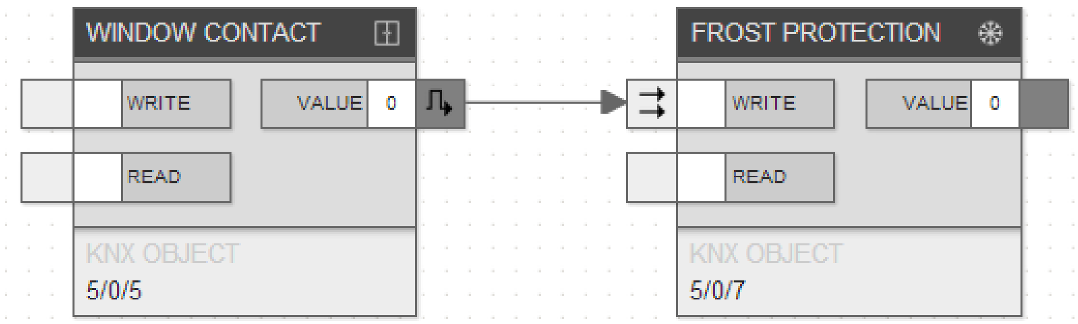
Are you sure you want to DEFINITELY delete the selected items? This operation cannot be undone

2.5 TESTING PROGRAMMABLE EVENTS

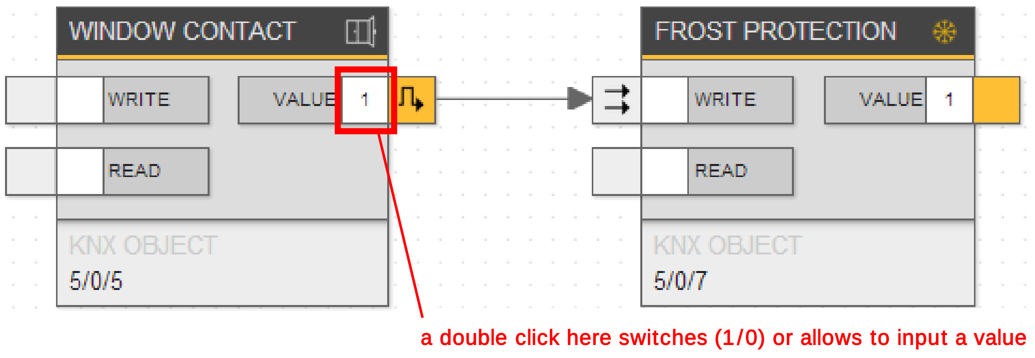
The testing of a Programmable Event will be shown using an example here. In this example a simple connection between two 1 bit KNX objects will be used: a window contact device and a *frost protection* device. The plan is to have the frost protection switch (part of a thermostat) to be directly triggered by the window contact device (in simple words "when the window is opened, the room's thermostat should switch to frost protection mode.")



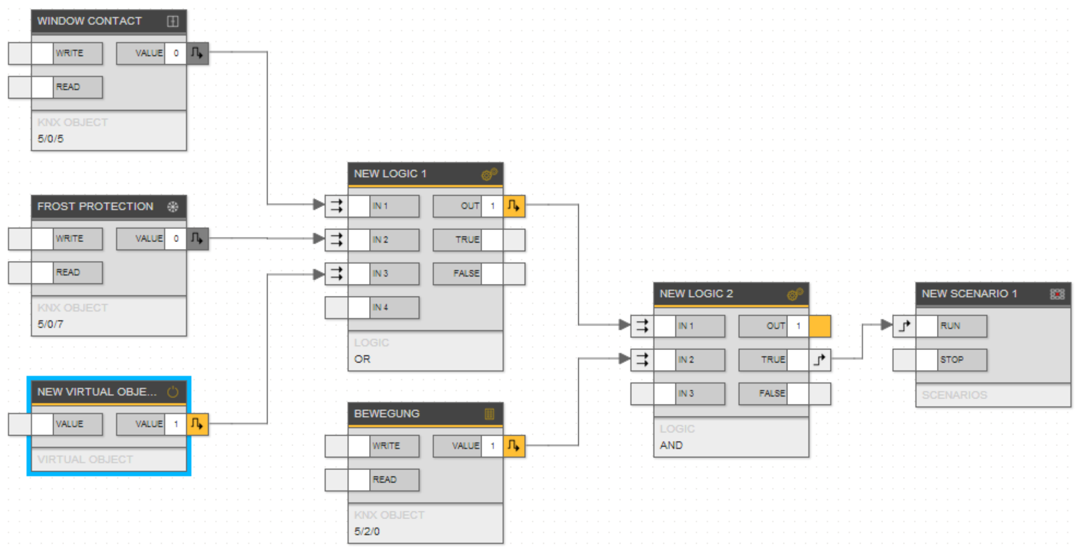
After the output was linked to the input, they will look like this::



Now you may start to test: what happens when the window contact is switched? You just need to doubleclick the small square with the "0" (near *VALUE* – see screenshots). This will cause the device to switch to "1" (being a 1 Bit object) and the external output section becomes yellow. The input of the frost protection object is set to "current value", therefore the window contact value will be passed on without any changes. So also frost protection gets value "1" and its output becomes yellow too.



The passing of a value/triggering may also go over multiple steps, like e.g. here:



It’s recommended to limit the chains/steps of each single Programmable Event to 3 or 4. In this way both the visual and the logical overview of the processes remain under control. You may still build more complex processes by creating a new Programmable Event and taking the output of the previous one as a starting point. For this task you may use a Virtual Object of type 1 BIT where the result of an Event is “stored” and which is then processed further in another Event. The risk of adding too many steps inside one Programmable Event is that of creating loops (procedures where the last step triggers the first one without a way to exit the repetitions); this risk grows with the complexity of the Programmable Event (i.e. the amount of involved objects and relations/triggers).

2.6 NOTES