

Lifedomus

Logic Module

31/01/2018

Version 1.2

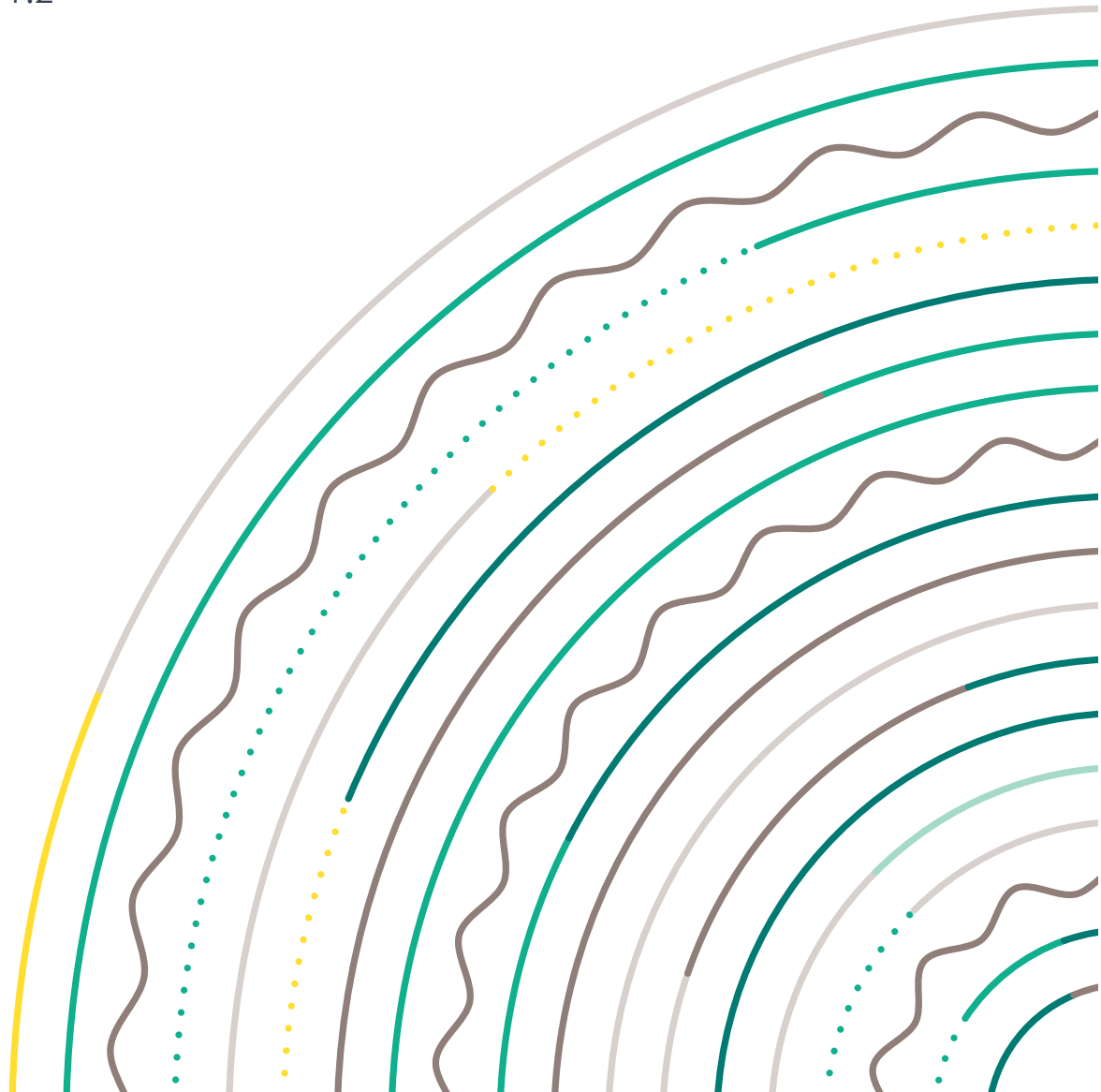


Table of contents

Table of contents.....	2
1. Presentation.....	3
1.1 Module architecture	3
1.2 Confirmation and backup.....	4
1.3 Execution check	4
1.4 Activation	5
2. Starting a controller.....	5
3. Creating a controller	6
3.1 'Action' item.....	7
3.2 'If' item	9
3.3 'While' item.....	12
3.4 'Variable' item	13
3.5 'Wait' item	15
3.5.1 Wait for a duration:	15
3.5.2 Active wait:	16
4. Trigger	16
5. Functions	18
6. Variables	19
Rights	20
Design Studio.....	21
Appendices.....	22
7. Operators	22
7.1 Character string operators	22
7.2 Numeric operators.....	22
7.3 Boolean operators	23
8. Variable type	24
8.1 Main types	24
8.2 Specific types	24

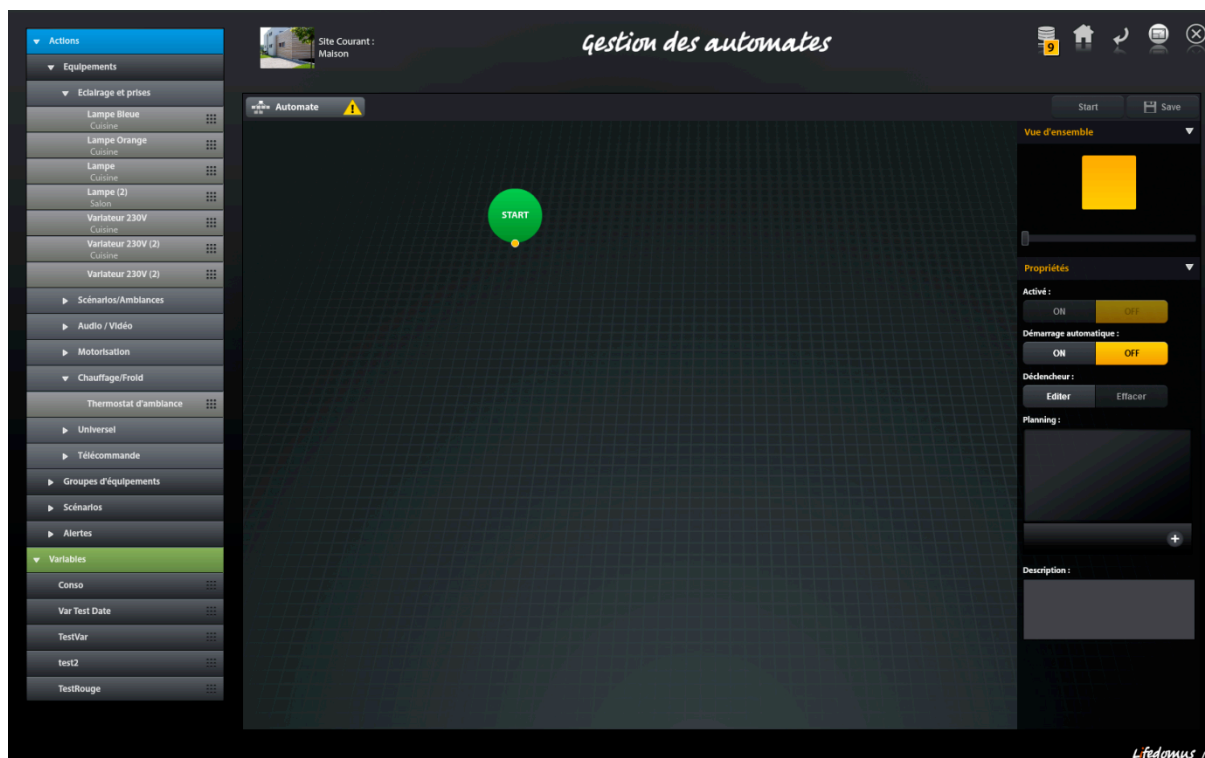
Logic module

1. Presentation

The logic module is a system that fully controls Lifemodus controllers. Controllers are the most advanced and powerful solution for optimally operating a home automation system. Lifemodus controllers are in fact made up of a sequence of operations that can be determined by environmental information or customised parameters. The logic module interface will easily and quickly meet your needs in terms of the most complex control systems.

1.1 Module architecture

The distinctive features of the logic module result in a special interface, made up of three panes.



The left pane includes the list of all usable objects in a controller such as devices, variables, functions, etc.

These objects can be dragged and dropped via the icon: 

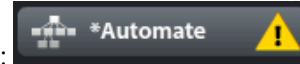
The middle pane is a limitless space where you will view and develop your controller's architecture. Each operation is represented by a graphic item. The tree structure of the links between each item will identify the sequence of operations.

The right pane shows the properties of the item selected for your controller. The 'START' item is the starting point of your controller. It is automatically positioned when a controller is created. To view your controller's properties, you can click on the item to select it.

1.2 Confirmation and backup

When creating a controller, the logic module checks for the presence of any errors in real time. You will thus be informed of any discrepancies, which will be clearly identified. In the tab on the top of the page, which identifies

the controller being edited, you will see the main controller state data:



The yellow warning icon will warn you that the controller is not properly configured and that it cannot be executed as is. This may be due to an incorrectly configured item, in which case there will be the same icon on the item and a message in the properties panel. There may also be a problem with the controller's trigger.

The star sign on the left of the controller's name indicates that the controller has been changed and that you can save it via the top right 'Save' button. This rule is derived from most publishing software, so you will easily recognise this symbol.

If you save this controller when there is an alert, it will be neither activated nor run.

1.3 Execution check

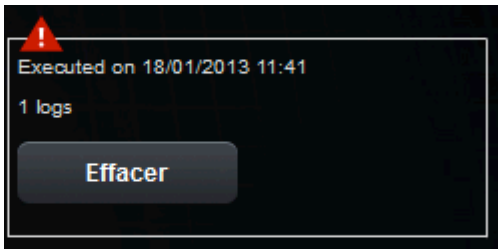
For optimised control of the controllers' operation, the logic module also checks each controller during execution.

If a controller was executed at least once, you can view the last execution date in the properties panel. This will indicate the controller's last execution, as well as the last execution of each item separately to the controller.

The execution dates are a first means of checking the process of a controller. The logic module will also inform you of any execution errors. If this is the case, a red warning icon will appear where the controller error occurred.



The icon is always placed on the controller's 'START' item and on the item where the execution error occurred. A log will show you the number of controller error occurrences and the time of occurrence. You can acknowledge the log for your failed item with the 'Delete' button.



The log can possibly help you to identify what is wrong with the controller. Errors can involve a deleted variable or function used by the controller, or a calculation for the state of a device that was incorrectly initialised (Red Widget in Lifemodus) or incompatible data types that the application did not check.

When a controller error occurs, the information is sent to the Design Studio applications via the alert system.

1.4 Activation

Each controller can be activated or left deactivated. This property is in the properties panel:



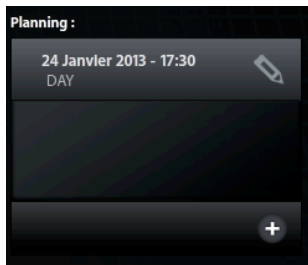
If your controller is deactivated, it will not start, regardless of the types of activation.

The controller is automatically deactivated when you change it and save the changes. The logic module considers that execution may be impaired by changes to one of its items. You will have to reactivate the controller. It is automatically activated if you use the 'Start' button if you run a test.

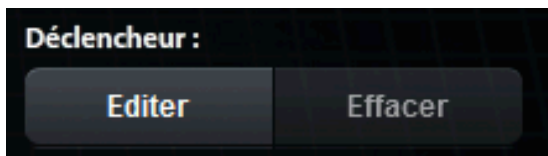
2. Starting a controller

Several activation modes are available to start a controller. These modes are found in the controller's property panel.

- Manual: The controller can be started via Design Studio by creating a widget and related action in What I Do.
- Automatic Start: The controller can be run automatically like a Windows service and will start when the Lifemodus server starts.
- Scheduling: To be done in the controller's property panel.



- Triggers: You can create a trigger by clicking on the Edit button:

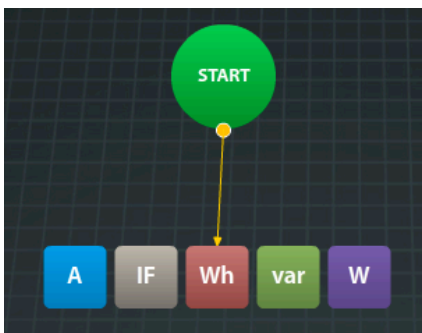


The 'Delete' button is deactivated (greyed out), meaning that your controller currently has no trigger.






3. Creating a controller

A controller is a sequence of operations materialised graphically by items.

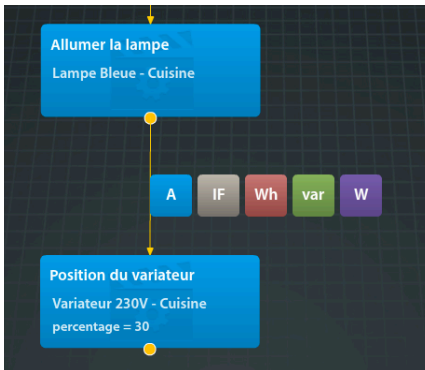
To create an item, use a yellow dot on the 'Start' item or any other item created subsequently. Draw a line by clicking and holding the mouse, then drag to a location of your controller and release the mouse:




There are 5 item types:

	'Action' item: executes an action as in the scenarios or in What I Do.
	'If' item: an item with two outputs for the controller depending on the result of a boolean condition.
	'While' item: executes a list of items until the 'While' condition is confirmed.
	'Variable' item: assigns the result of a function, entered value or state feedback, etc. to a variable.
	'Wait' item: enables you to wait either for a duration in milliseconds or an active wait period, i.e. a trigger.

You can add an item between two items already created by clicking on the link interconnecting them:

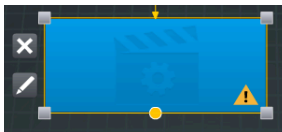


Each item can be deleted either with the 'Del' button on your keyboard or via the  icon on the left of the item.

Each item has a description field in its properties so that you can describe its usefulness. This description will be available in a tooltip when hovering over the item with the mouse.

3.1 'Action' item

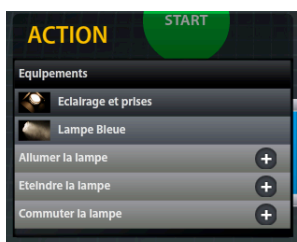
An 'Action' item is an item that enables the execution of an action on all the Lifemodus objects.



By default, your item is in a warning state as no action is selected.

An action can be selected in either of two ways:

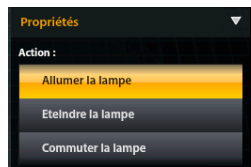
- Either by clicking the  icon and selecting the required action in the pop-up,



- or by using the left-hand side bar and dragging and dropping a device, group, scenario, etc. on the blue item,



Then select the action for this object in the item's property panel (by default the 1st action is selected):




Once configured, this generates an item like the one below, with the name of the action and object.



For a configured action, you can enter the parameter in the property panel. Each action parameter can be entered in a field or replaced by the content of a variable:

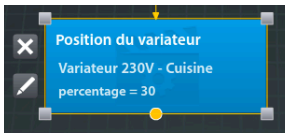


The replacement with this variable can for example enable you to send the information of a protocol to another.

 **Note:** the parameter type must be the same as the required parameter.

For example, to enter the dimmer position, the variable type must be numeric.

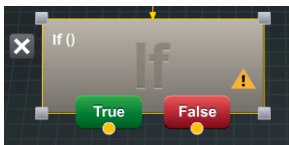
This parameter, if there is only one, will be visible directly on the item:



Once an object is associated with the item (e.g. a device), you will then be able to change only the action for the item's properties, or via the pencil icon. If it is the wrong device, you will have to either delete the item, or use Drag and Drop to overwrite the current item.

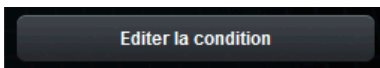
3.2 'If' item

An 'If' item executes one or more items only in a specific case that can be defined.

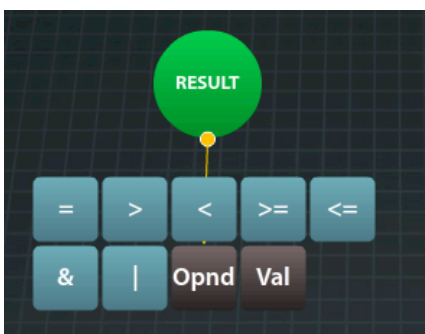


By default, your item is in a warning state as no condition is selected.

To define the condition, click on the 'Edit condition' button in the item properties.



This will open a different tab to the controller's. To generate your condition, proceed as for controller items.



There are no items to be added, but operators, operands or values. An operand is a calculation element for operators. This element can be the state of a device, a variable, the result of a system function or data (time, date, etc.).

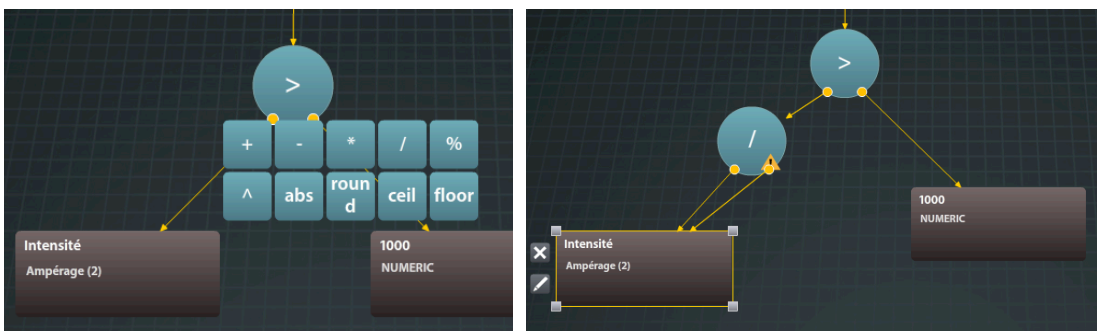
The list of operators is provided in an appendix.

In this case, only boolean operators are available, as the result of an 'if' condition always ends by true or false.

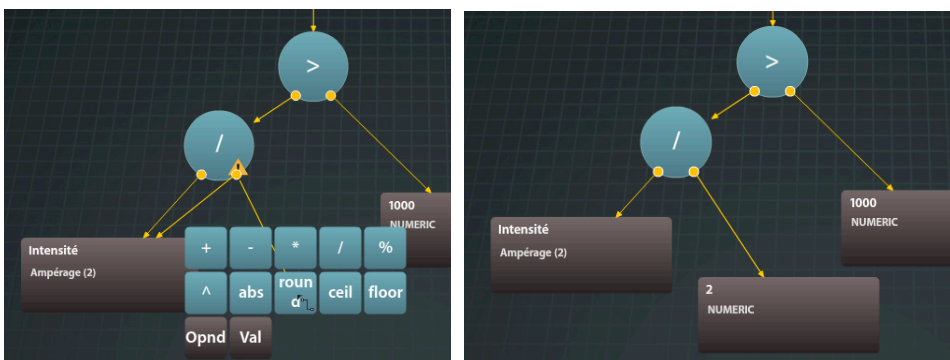
There is no level limit, you can apply as many comparisons or calculations as necessary by creating operators.

For operands, as for 'Action' items, you can either use the pencil icon, or the Drag and Drop function in the list on the left and select the relevant state.

You can add an operator to a calculation by clicking on the link between operators or operands:

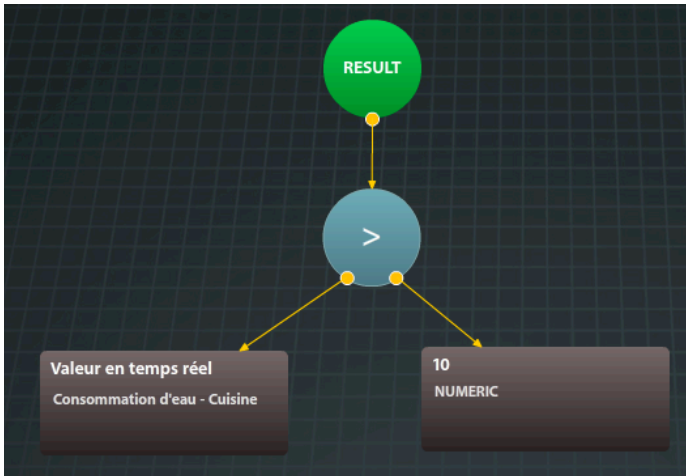


If some operators only have 2 inputs, the two inputs will point to the same operand by default. You can draw a line from one of the yellow dots to add an operator or operand.



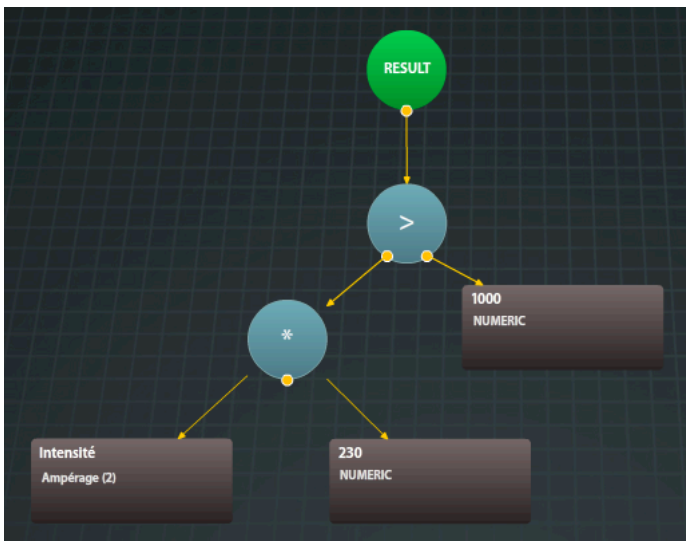
Example 1:

If the output is greater than 10m³/h.



Example 2:

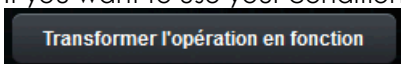
If my consumption is greater than 1kw (Power = Current * Voltage (≈230V)).



To return to the controller, there are two similar output points: the 'Save' button in the top right-hand corner of the screen or by clicking on the controller's tab in the top left hand corner.



If you want to use your condition for a different controller, you can convert it to a function by clicking on:

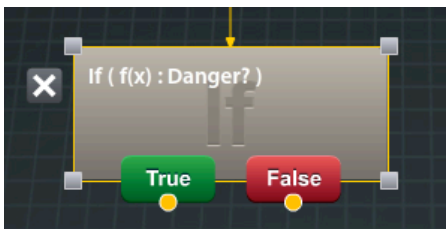


Then enter a name for your function and press on 'Confirm'.



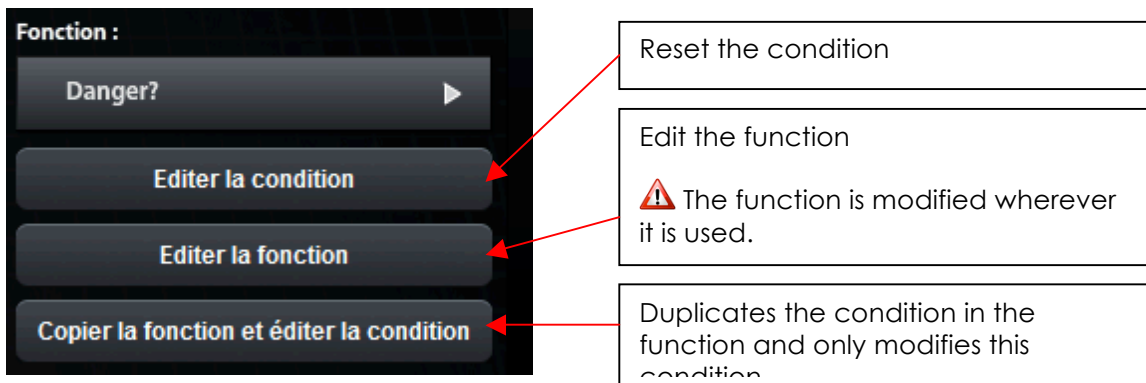
The function will now appear in the list on the left and will now be applicable everywhere.

If you want to add an 'If' item subsequently, you can use the Drag and Drop function for the function for this item:



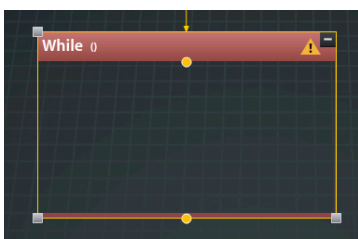
 Note that only the functions that return a boolean result can be dragged onto this item.

To edit the 'If' item, three options are available:



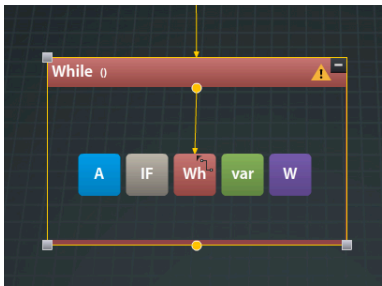
3.3 'While' item

The 'While' item is a loop which executes a group of items, as long as the condition is not verified. This item is very similar to 'If' items in terms of construction:



To edit the condition, it is the same principle as for an 'If' item: click on the 'Edit condition' button in the item's property panel.

To edit the list of items to execute in your loop, draw a line from the yellow dot inside the frame that represents the loop.



Pay particular attention with this item.

First of all, make sure that your condition is feasible, to avoid an infinite loop, otherwise your controller will never end.

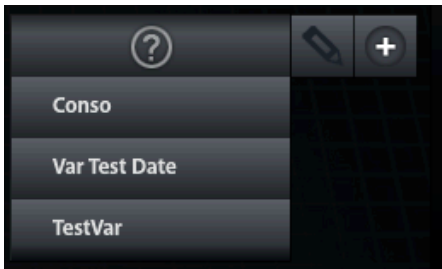
Then, we advise you to add at least one 'Wait' item to this loop, to avoid poorer performance. Either a duration or an active wait for a feedback state that will modify the 'while' condition.

3.4 'Variable' item

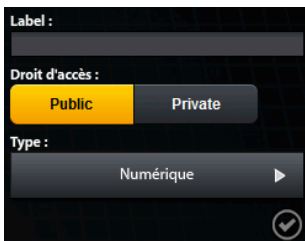
This item allocates a value to a variable.



First or all, select the variable that you want to modify, either from the list on the left then with a Drag and Drop, or by selecting it from the item's property panel.



To create a variable, press the '+' button.



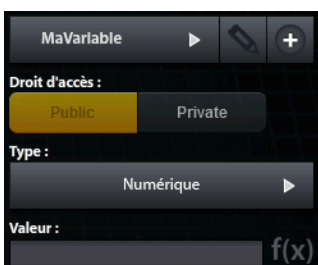
You can assign a name to it (The name must be unique for a Lifemodus server).

The selection of the access right will depend on the use of your variable. If it is a controller programming variable (e.g.: counter) that will not be used anywhere else (in an different controller or seen in Design Studio) then you can set it as private or leave it public.

You can then select a data type for the variable (list of types will be detailed in the appendix).

Then click on 'Confirm'.

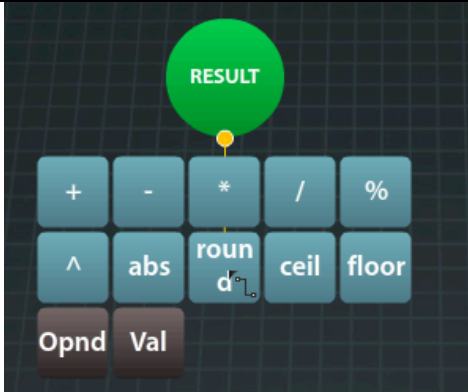
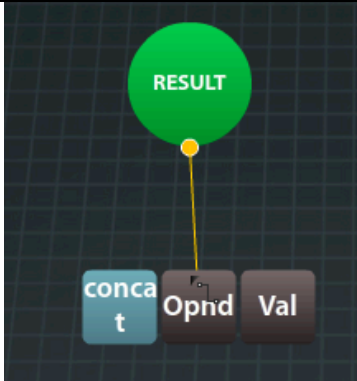
The variable is automatically selected, you can edit it by clicking on the pencil icon. Note: you can only edit the name and private/public status of the variable.



Once your variable is selected, you can either assign a value by writing in the dedicated field, or you can perform a calculation by clicking on the 'f(x)' button.



The procedure for editing this function is similar to editing the condition of an 'If' or 'While' item, except that the operators will change according to the type of variable.

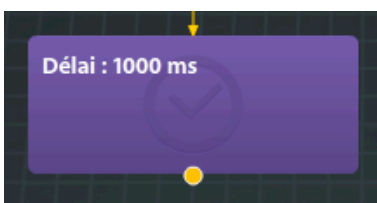
Numeric variable example:	Character string variable example:
	

Similarly to editing an 'If' item, you can convert your predicate according to the associated button in the item's property panel.

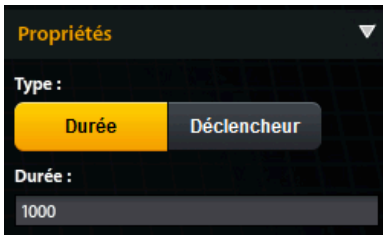
3.5 'Wait' item

The 'Wait' item is an item that waits while your controller operates. It is distinctive in that it is the only item with 2 types of operation. Either it waits for a duration in ms, or it waits for a trigger.

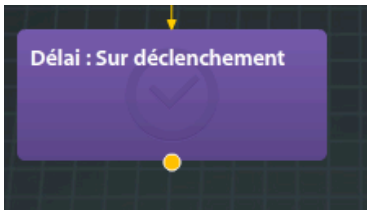
3.5.1 Wait for a duration:



The 'Wait' item is a wait for a duration of 0 by default. You can configure it or change the type from the item's property panel.



3.5.2 Active wait:



The active wait is configured like a trigger, as explained below. You can edit it via the 'Edit' button in the item's property panel.

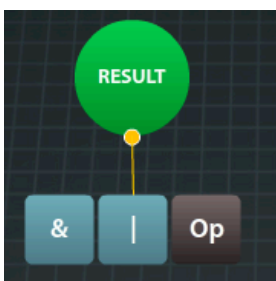
The 'Eval' button evaluates the condition as soon as the trigger is launched or not. If 'yes' is selected, your trigger will change to blocking 'While'.

4. Trigger

A trigger can be edited from two locations: the controller's trigger or the trigger in a 'Wait' item. In both cases, the edition procedure is the same.

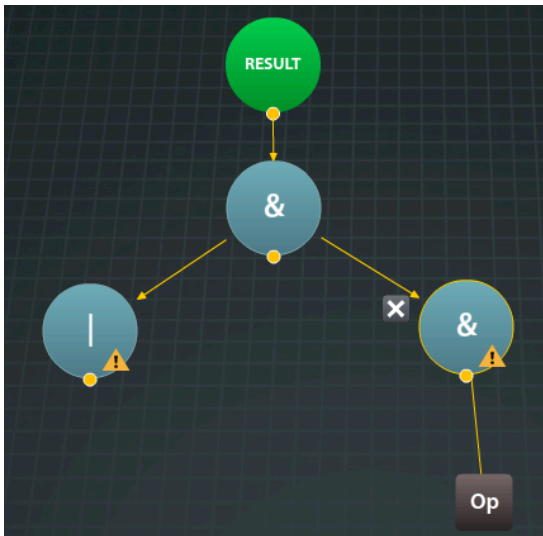
Editing a trigger is a more graphic and upgraded version of the current system in Lifemodus.

When you draw a line from the 'Result' dot, you have either a logical '&' (AND), a logical '|' (OR), or an operand.

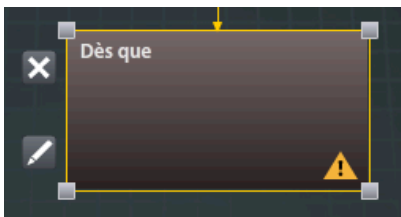


You are currently limited to a maximum of 2 '&' or '|' operator levels.

Example:



An operand is actually a condition set for a Lifemodus object (device state, variable or system data). To edit an operand, you can either drag and drop from the list on the left or use the pencil icon:



Then edit it from the item's property panel:

Condition :

Dès que ▶

Position en % ▶

> ▶

50

There are 3 types of validation for a 'As soon as', 'Whenever' and 'At each change' operand:

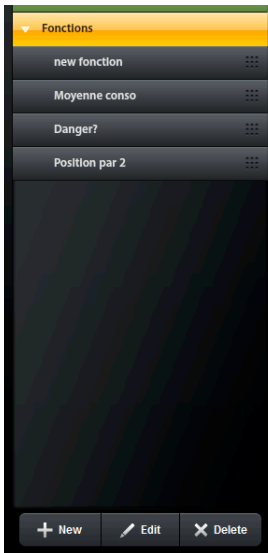
As soon as	Whenever	At each change
<p>Condition :</p> <p>Dès que ▶</p> <p>Position en % ▶</p> <p>> ▶</p> <p>50</p>	<p>Condition :</p> <p>A chaque fois que ▶</p> <p>Position en % ▶</p> <p>> ▶</p> <p>50</p>	<p>Condition :</p> <p>A chaque changement ▶</p> <p>Position en % ▶</p>
<p>In this case, for the operand to be enabled, the percentage of your dimmer has to change from a value below 50 to a value above 50.</p>	<p>In this case, for the operand to be enabled, the percentage of your dimmer has to change from a value above 50, even if your dimmer was already above 50. E.g. 70 → 75</p>	<p>Here, for an operand to be enabled, the percentage value of your dimmer must change, irrespective of the value.</p>
<p>Advantage:</p> <p>Trigger at threshold</p>	<p>Advantage:</p> <p>For KNX push buttons that always sends 1 to the bus, never 0.</p>	<p>Advantage:</p> <p>If you run a calculation for a value, irrespective of the value, e.g. whenever the intensity changes, I calculate the consumption after this waiting time.</p>
<p>↓</p> <p>Dès que Position en % - Variateur 230V (Cuisine) > 50</p>	<p>↓</p> <p>A chaque fois que Position en % - Variateur 230V (Cuisine) > 50</p>	<p>↓</p> <p>A chaque changement Position en % - Variateur 230V (Cuisine)</p>

5. Functions

As explained above, you can change the condition of 'If', 'While' and 'Variable' items into a function.

You can also create a blank function from the list on the left.

Select either a function, or the function node:



The 'New' button will edit a function in the same view as for editing a condition, however, when you draw a line, all operators will be available and the returned data type will depend on your calculation.

The 'Edit' button edits a function. Note that this will affect all the items to which it is applied. Changing the return type may affect the controller's operation.

The 'Delete' button deletes a function. Note that no check for use in a controller is performed. This could cause your controller to stop operating.

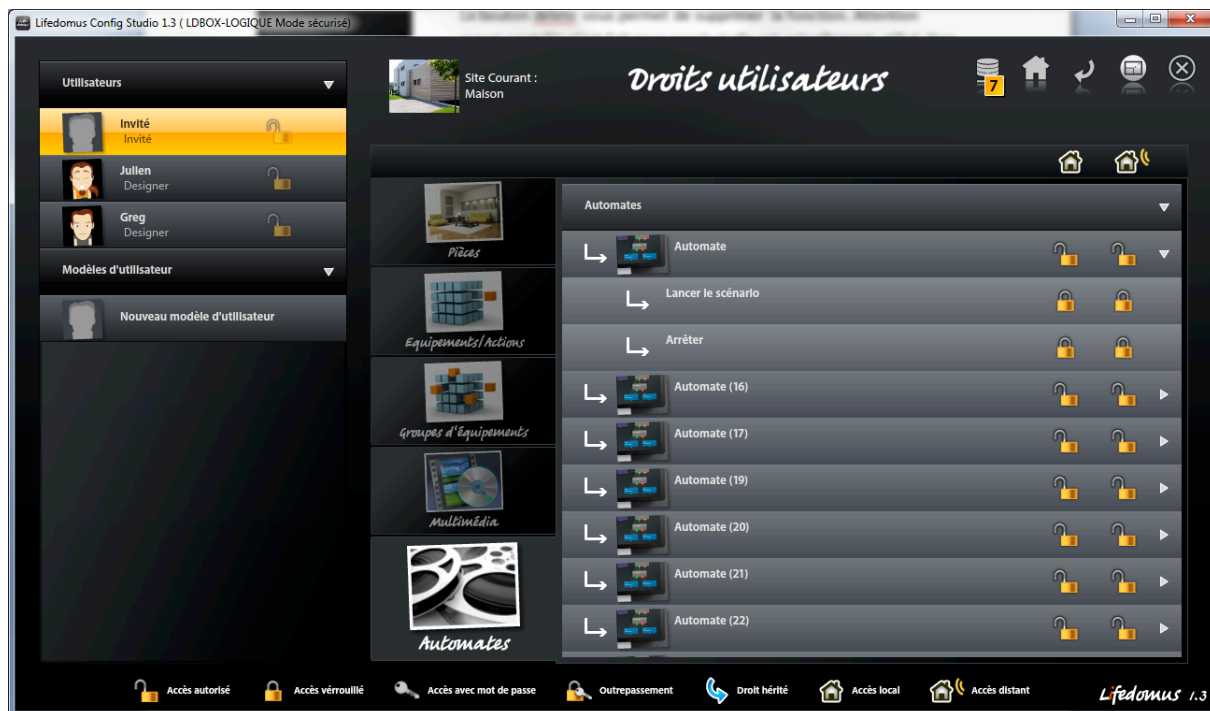
6. Variables

Like for functions, a variable can be created, edited or deleted via the list on the left by selecting a variable or the variable node.

Rights

As for the entire Lifedomus system, any variables and controllers operated in 'Design Studio', are subject to user rights. They are configurable in the Config Studio's rights page and by selecting the controller tab.

By default, controllers and variables are available (see their state) to all users. In addition, each user can change the variables. However, by default, users cannot start or stop a controller.




Design Studio

You can use the variables and the controllers in Design Studio (Only PC and Mac currently. An update on iPad is in progress).

In 'What I See', you can see the value of a variable and the state of a controller as a boolean value, to check whether it is active.

 Note that ONLY public variables will be displayed in 'What I See'.




In 'What I Do', you can change a variable or activate/deactivate a controller.

 Note that by default, users do not have the rights to 'start' and 'stop' a controller.





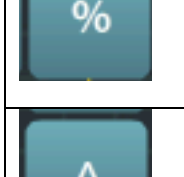

Appendices




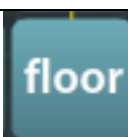
7. Operators

7.1 Character string operators








	<p>This operator places 2 character strings after one another.</p> <p>It uses all the data types that will be converted automatically in input.</p> <p>E.g. Value concat ('It does: ') and state feedback (outdoor temperature).</p>
	<p>This operator converts the input value into a character string.</p> <p>The input value can be all data types.</p>
	<p>This operator can convert a character string into a data type.</p> <p>The input value is a character string and the output depends on the associated context.</p> <p>String => Number ('22', '10.5')</p> <p>String => Date (Format 'YYYY-MM-DD')</p> <p>String => Time (Format 'HH:MM')</p>




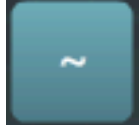
7.2 Numeric operators

	<p>This operator has at least two inputs. It adds these inputs and returns data with the same type as the input.</p> <p>Available data types are: Numeric and Time.</p>
	<p>This operator has two inputs. It subtracts the second input from the first one and returns data with the same type as the input.</p> <p>Available data types are: Numeric and Time.</p>
	<p>This operator has at least two numeric inputs. It multiplies the inputs and returns numeric data.</p>
	<p>This operator has two numeric inputs. It can divide the 1st input by the second and this generates numeric data.</p>
	<p>This operator has two numeric inputs. It can divide the 1st input by the second and generates the remainder of the division as numeric data.</p> <p>E.g. 10 % 3 => 1 / 25%7 => 4</p>
	<p>This operator has two numeric inputs. It can multiply by itself the 1st input of the number of times defined by the second and this generates numeric data.</p> <p>E.g. 5^3 => 5*5*5 => 125</p>

	<p>This operator has a numeric input. It can return the input's absolute value and generates numeric data.</p> <p>E.g.: <code>abs(22) => 22 / abs(-12.5) => 12.5</code></p>
	<p>This operator has a numeric input. It can return the input's rounded value and generates a numeric data.</p> <p>E.g.: <code>round(2.3) => 2 / round(4.6) => 5</code></p>
	<p>This operator has a numeric input. It can return the input's value rounded up to the nearest integer and generates numeric data.</p> <p>E.g.: <code>ceil(2.3) => 3 / ceil(4.6) => 5</code></p>
	<p>This operator has a numeric input. It can return the input's value rounded down to the nearest integer and generates a numeric data.</p> <p>E.g.: <code>floor(2.3) => 2 / floor(4.6) => 4</code></p>

7.3 Boolean operators

	<p>This operator has two inputs. It can compare 2 inputs of the same type. It returns a boolean value: TRUE if all inputs are similar, FALSE otherwise</p> <p>All types are available as input.</p>
	<p>This operator has two inputs. It can compare 2 inputs of the same type. It returns a boolean value: TRUE if input 1 is greater than input 2, FALSE otherwise.</p> <p>Comparable data types are: Numeric, Date, time, day of the week and day of the month.</p>
	<p>This operator has two inputs. It can compare 2 inputs of the same type. It returns a boolean value: TRUE if input 1 is below input 2, FALSE otherwise.</p> <p>Comparable data types are: Numeric, Date, time, day of the week and day of the month.</p>
	<p>This operator has two inputs. It can compare 2 inputs of the same type. It returns a boolean value: TRUE if input 1 is similar to or greater than input 2, FALSE otherwise.</p> <p>Comparable data types are: Numeric, Date, time, day of the week and day of the month.</p>
	<p>This operator has two inputs. It can compare 2 inputs of the same type. It returns a boolean value: TRUE if input 1 is similar to or below input 2, FALSE otherwise.</p> <p>Comparable data types are: Numeric, Date, time, day of the week and day of the month.</p>
	<p>This operator has three inputs. It checks if the value of input 1 falls within input 2 and input 3 inclusive. It returns a boolean value: TRUE if input 1 falls within the others, FALSE otherwise.</p> <p>Only the numeric type can be used.</p>
	<p>This operator has three inputs. It checks if the value of input 1 is within input 2 and input 3 not inclusive. It returns a boolean value: TRUE if input 1 falls within the others, FALSE otherwise.</p> <p>Only the numeric type can be used.</p>

	This operator is not limited in terms of input number. It can be used for a logical AND on the inputs (boolean only). It returns a boolean value: TRUE if all inputs are TRUE, FALSE otherwise
	This operator is not limited in terms of input number, it can be used for a logical OR on the inputs (boolean only). It returns a boolean value: TRUE if at least one input is TRUE, FALSE otherwise
	This operator is not limited in terms of input number. It can be used for a logical Exclusive OR on the inputs (boolean only). It returns a boolean value: TRUE if one input is TRUE, FALSE otherwise
	This operator has one input. It can be used for a logical NO, i.e. invert the value. If the input is TRUE, the result will be FALSE, and conversely.

8. Variable type

8.1 Main types

Numeric (23, 2.5, etc.)	Character string ('Hello')
Boolean (True, False)	Date (26/03/2013)
Time (12:33)	Day of the week (Monday, Tuesday, etc.)
Day of the month (Number from 1 to 31)	Date and Time

8.2 Specific types

Specific types are device state feedbacks that can use certain values only.

You are not likely to be using them.

Heat mode	Thermostat device heat mode: Comfort, Eco, Reduced and Frost Protection
Position state	Values returned by some KNX engines (Up, Down, Intermediate)
KNX presetting state	Override On, Override Off, Unforce
Cumulus state	Cumulus state controlled by the varuna (Off, On, Load Shed, Overridden)
Energy mode	Varuna's energy mode (Winter, Summer, Frost Protection)
Regulation state	State of the regulation of one of the Varuna's heating zones (Absence, Presence, Comfort)
Zone override	Possible override of a Varuna's heating zone (Not overridden, override in Comfort mode, override in Absence mode, override in Presence mode, override on daily cycle from 1 to 8).
Photovoltaic cell threshold	Threshold reached by the Varuna's photovoltaic cell (None, Threshold 1, Threshold 2, both)
Temperature control	Mode of operation of the thermostat device and the temperature control unit: Heat or

unit's heating mode	cool
Temperature control unit's mode	Function that can be activated on the temperature control unit: OFF, manual temperature setting, frost protection temperature setting, heat protection temperature setting, temperature setting according to one or more programmed scenarios.
Local sensor value	Position of the Legrand thermostat/sensor in the room to set the temperature locally: comfort temperature +0, comfort temperature +1, +2, +3, -1, -2, -3, OFF, protection (frost protection or heat protection [35°C])
Battery state	Legrand alarm control unit battery charge level: battery OK, Empty or error (no feedback on the level).
Alarm control unit state	Legrand alarm control unit state/mode: Alarm on, alarm off or alarm control unit in maintenance mode.