

2.3

ekinex
CONTROL YOUR LIVING SPACE

Automations

System Configuration Manual



DELÉGO SERVER

Visualisation webserver

EK-DEL-SRV01 / EK-DEL-SRV01-M

Index

1 Introduction.....	4
1.1 Scope of this manual.....	4
1.2 Conventions.....	4
1.3 Safety information.....	4
1.4 Warnings.....	5
1.5 Copyright.....	5
1.6 Open Source Licenses.....	5
1.7 More information.....	6
2 Connections.....	7
2.1 Introduction.....	7
2.2 Creating a new connection.....	7
2.3 Relationships between nodes.....	8
2.4 Event-based logics.....	9
3 Triggers and actions.....	11
3.1 Introduction.....	11
3.2 Triggers.....	11
3.3 Actions.....	13
4 Logics.....	14
4.1 Introduction.....	14
4.2 Logic programs.....	14
4.3 Toolbar.....	14
4.4 Blocks.....	15
4.5 Execution order.....	15
4.6 TRG.....	16
4.7 Simulation.....	16
4.8 Execution of logic programs.....	17
5 Library.....	18
5.1 Introduction.....	18
5.2 Combinatorial logics.....	18
5.3 Sequences.....	21
5.4 Gates.....	24
5.5 Comparisons.....	33
5.6 Math Operations.....	34
5.7 Counters.....	36
5.8 Timers & schedules.....	37
5.9 Variables.....	41
5.10 Constants.....	42
6 Drawing Tools.....	43
6.1 Introduction.....	43
6.2 Labels.....	43
6.3 Rectangular areas.....	43
7 Revision history.....	44

1 Introduction

1.1 Scope of this manual

This document contains all the information for configuring the different types of AUTOMATIONS offered by DELÉGO SERVER:

- Connections
- Triggers and actions
- Logics

The present user manual is part of the suite of application manuals dedicated to the DELÉGO SERVER, available on the section dedicated to this product in the EKINEX website. Before reading this document, it is recommended to consult the following application manuals:

REF	DOCUMENT	DESCRIPTION
[D1]	DOCUMENTATION PLAN	Overview of the application manuals of DELÉGO SERVER
[D2]	GENERAL FUNCTIONS - APPLICATION USER MANUAL	General installation manual of the webserver
[D3]	KNX - APPLICATION USER MANUAL	Configuration and visualisation of a KNX automation system in DELÉGO SERVER

These documents are available for download in the area of the EKINEX website dedicated to the webserver.

1.2 Conventions

The following conventions are used in this manual:

[KEY]	The keyboard keys that the installer must press are shown in square brackets
Courier	Software messages generated by the DELÉGO SERVER configuration software are reported in the "courier" font
Function name	Names of buttons, table fields or other graphic elements of the user interface of the DELÉGO SERVER configuration software are shown in italics
	Information note
	Important note that requires particular attention

1.3 Safety information

DELÉGO SERVER is a product intended for trained and specialized personnel only; anyone who interacts with the device must first read this documentation, especially this safety information. This document supplements and does not replace any legislation or legal directives on safety.

The device has been developed following the safety regulations currently in force; however, it is not possible to exclude with certainty possible damage or interactions with other devices during its operation. The device complies with EMC guidelines and harmonized European standards; any modifications to the device may affect EMC compatibility.

The supply voltage must be strictly within the range indicated in the GENERAL FUNCTIONS application manual and on the device; risk of fire or explosion if powered outside this range. The CE declaration of conformity of the device can be requested from EKINEX®, at the references on the site www.ekinex-delego.com.



In compliance with Directive 2002/96/EC the DELÉGO SERVER electronic device must be disposed of in the appropriate facilities and not in the collection of solid urban waste.

1.4 Warnings

- This application manual is aimed at installers, system integrators and designers.
- The assembly, electrical connection, configuration and commissioning of the appliance can only be carried out by specialized personnel in compliance with the applicable technical standards and the laws in force in the respective countries
- Opening the appliance case causes the immediate interruption of the warranty period
- In the event of tampering, compliance with the essential requirements of the applicable directives for which the appliance has been certified is no longer guaranteed.
- EKINEX defective appliances must be returned to the manufacturer at the following address:
Ekinex S.p.A. Via Novara 37, 28010 Vaprio d'Agogna (NO)

1.5 Copyright

© Ekinex S.p.A. 2025

This application manual, the DELÉGO SERVER software and the DELÉGO iOS and Android apps are subject to copyright; all rights are reserved. Copying, reproduction, translation and/or modification, even partial, are expressly prohibited unless approved in writing by EKINEX®.

1.6 Open Source Licenses

DELÉGO SERVER contains open source software, such as the Linux operating system and its kernel. These software components are subject to various open-source licenses, including:

- GNU General Public License (GPL), <https://www.gnu.org/licenses/gpl-2.0.html>
- GNU Lesser General Public License (LGPL), <https://www.gnu.org/licenses/lgpl-2.0.en.html>

If you own a product, for a period of 3 years from the last production date, you can request from EKINEX the source code of the software components licensed under the GNU General Public License (GPL) or the GNU Lesser General Public License (LGPL), and use, distribute and modify them in accordance with the respective licenses.

EKINEX cannot be considered responsible in any way for the source code thus distributed, which is provided without guarantees. EKINEX cannot also be held responsible for any damage or consequence resulting from modifications (additions / removals) made to this software by third parties, unless specifically authorized.

1.7 More information

- For further information on the product, you can contact EKINEX technical support® to the e-mail address: support@EKINEX.com or consult the website www.EKINEX.com
- EKINEX® is a registered trademark of Ekinex S.p.A.
- KNX® ed ETS® are registered trademarks of KNX Association cvba, Brussels

The company reserves the right to make changes to this documentation without notice.

2 Connections

2.1 Introduction

Starting from the release 2.6.0, it is possible to establish relations among widgets / objects with a visual editor, by creating one or more CONNECTIONS.

CONNECTIONS are a visual representation of *relationships* among the objects of the project, which determine an automatic action on an object (the *destination*) when the status of another object changes (the *source*).

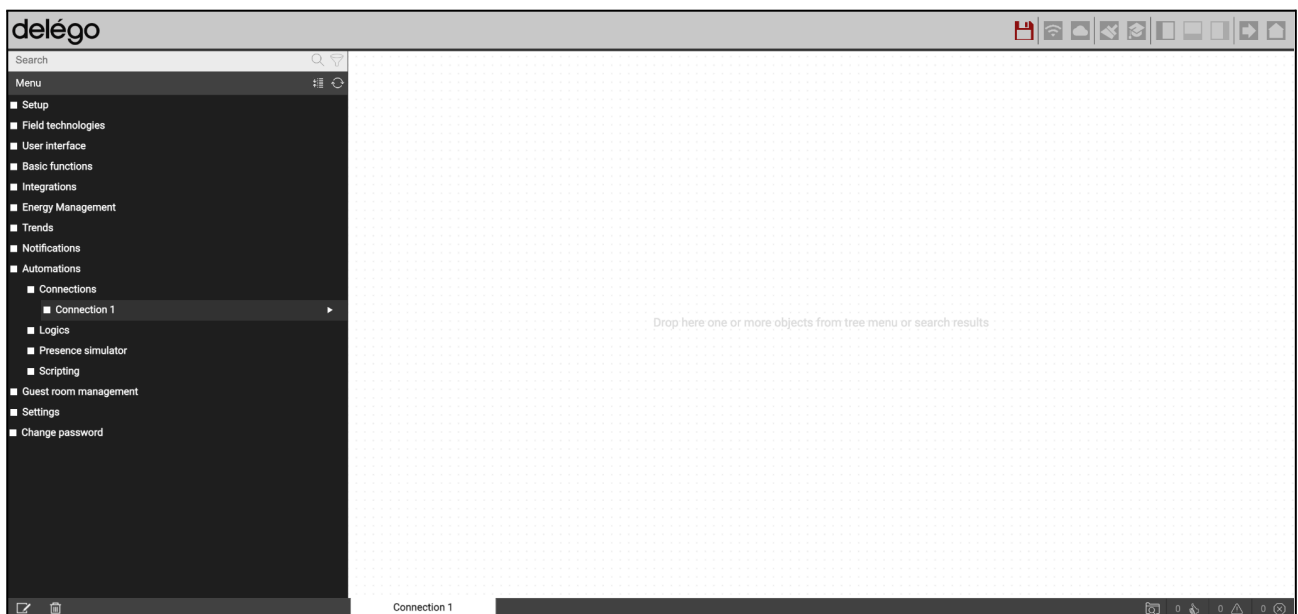
Differently from the LOGICS (see [section 4](#)) these *relationships* are not permanently in execution in the background of the webserver, but they are activated only “on event” when the involved objects receive a new state / incoming telegram from the corresponding field technology.

2.2 Creating a new connection

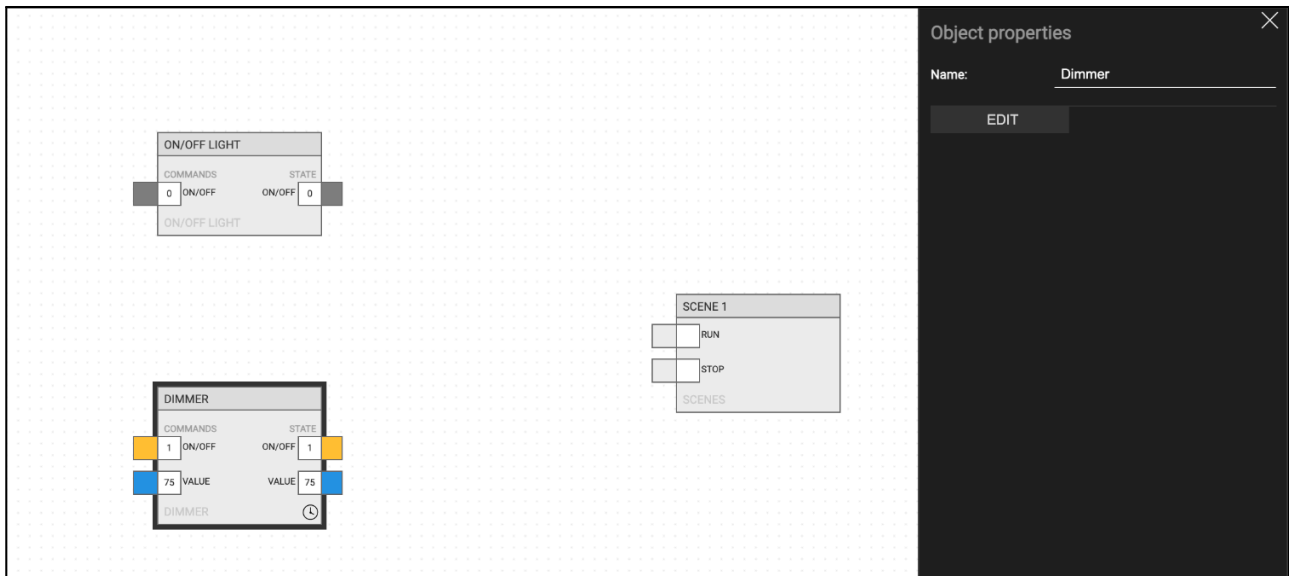
In the administration view navigate to the section

AUTOMATIONS > CONNECTIONS

and add a new connection by clicking the [+] button. After selecting the new item, an empty editor like in the following figure is displayed:



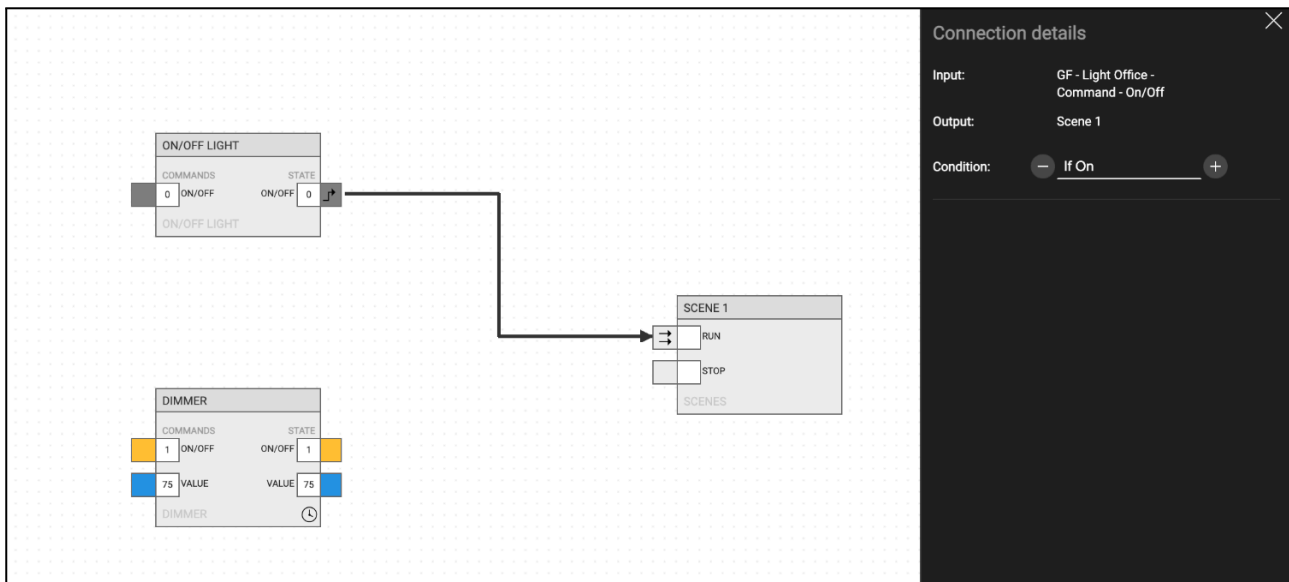
Any “functional” object of the project (widgets, scenes, etc) can be Drag&Dropped inside this editor, in order to display it as a “block” with input and output nodes. By **right clicking** on the selected block, it will be opened a right panel with some information of the object and a *shortcut* button to its details page:



The number and typology of nodes depend on the type of element dropped into the editor. Only objects that execute actions can be added to a connection; elements like floors, rooms etc, are not possible to use in the connections. The color and numeric value for each node, gives a visual indication of its actual state.

2.3 Relationships between nodes

In order to connect two nodes (representing the “output” state of an item, on the right, and the “input” command to another item, on the left) start pressing the mouse button on the starting node, and draw an arrow by releasing the mouse over the destination node:



Boolean nodes (that can be only ON or OFF) can only be related with other boolean nodes, and not with *numerical* nodes. When drawing a new arrow, incompatible nodes are automatically disabled, to avoid a mismatching relation.

Each row represents an “event based” relationship between the two items, meaning that each time the *source* object receives a status change (a new telegram from the KNX bus, a change of value on a Modbus register, etc) the new value is evaluated and, if the new value matches the *conditions* of the relation, an *action* is performed on the *destination* object, by eventually passing a *value* (if supported by the action).

After drawing an arrow, it is possible to specify (in the right panel) the attributes of the underlying relation, by **right clicking** on it:

CONDITION	A filter on the value of the <i>source node</i> that must match in order to do the <i>action</i> on the <i>destination node</i>
ACTION	The action to be done on the <i>destination node</i>
ACTION VALUE	The value to be passed to the <i>destination node</i> if supported by the selected action

In the previous figure, when the status of the ON/OFF light is ON, the connected scene is executed: note that the symbol in the *source* node represents the condition of the relationship, while the symbol in the *destination* node indicates the combination of action and value to be sent.



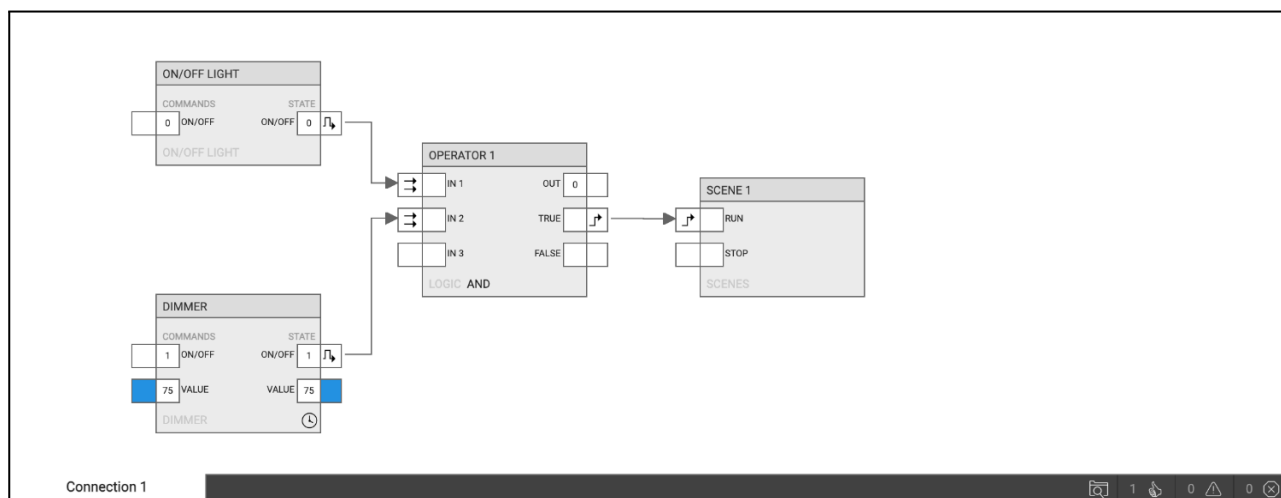
Possible symbols for the output and input nodes:

OUTPUT NODES		INPUT NODES	
↗	Rising edge (if ON)	↗	ON value (despite of the input value)
↘	Falling edge (if OFF)	↘	OFF value (despite of the input value)
↕	Any change of value (never)	↔	Value received in input (current value)
✕	In any condition (never)	✕	Opposite of the value received in input (inverted value)

2.4 Event-based logics

It is possible to create complex relationships between objects. In this way and using also the items available in the ADVANCED FUNCTIONS section of the administration menu (e.g., combinatorial operators,

mathematical operators, etc), it is possible to program logics inside a CONNECTION, like in the following example:



For this purpose, are particularly suitable the following types of elements:

COMBINATORIAL OPERATORS	Objects that perform an AND, OR or NOT operation among the connected inputs, and return in output the result of the operation
MATHEMATICAL OPERATORS	Objects that perform an arithmetical operation among the connected inputs, and return the result in output
CONDITIONS	Objects that perform a comparison between two values and return a true/false result in output
INTEGRATORS	Objects that calculate the <i>integral</i> along time of the input. Suitable, for example, as “time counters” when an ON/OFF input is connected.



Even if the graphical editor of the CONNECTIONS is very similar to the one used for the LOGICS, as better detailed in the next chapter, the two concepts have a big difference that must be clear to the installer / system integrator.

CONNECTIONS, on one side, visually represent *relationships* among objects, that instruct the software to *do something* when *some event occurs*.

On the other hand, LOGICS represent a *routine* continuously in execution in background, which for that reason, can also do actions along the time, even when the involved objects do not vary their state.

CONNECTIONS are suggested for simple automations, whereas LOGICS must be considered for more complex elaborations.

3 Triggers and actions

3.1 Introduction

The same *relationships* between two objects, that we have seen as *arrows* inside the CONNECTIONS, can also be realized outside the connections editor, directly in the details page of the involved elements.

3.2 Triggers

Each time in a detail page the TRIGGERS section is visible, like in the following example of a SCENE

Object properties

General information

Name:

Scene 1

Room:

Permissions

Visible:

☒

Execute at startup:

☐

Allow editing in visualisation:

☐

Cloud

Enable cloud sync:

☐

Voice control:

☐

CLOSE

PLAY

STOP

Actions

NAME	DETAILS	ACTION	VALUE
Drop here objects from search results or tree menu			

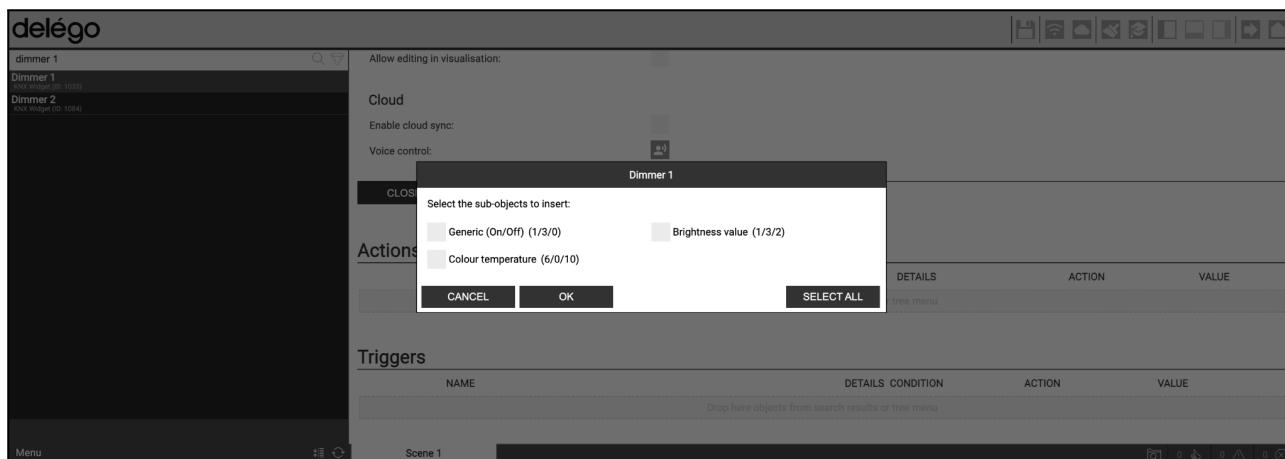
Triggers

NAME	DETAILS	CONDITION	ACTION	VALUE
Drop here objects from search results or tree menu				

Scene 1

it is possible to drop another object inside it, in order to create a *relationship* with that object. Each time that object will change its value, the new value will be evaluated and, if matching, the selected *action* will be performed on the current object.

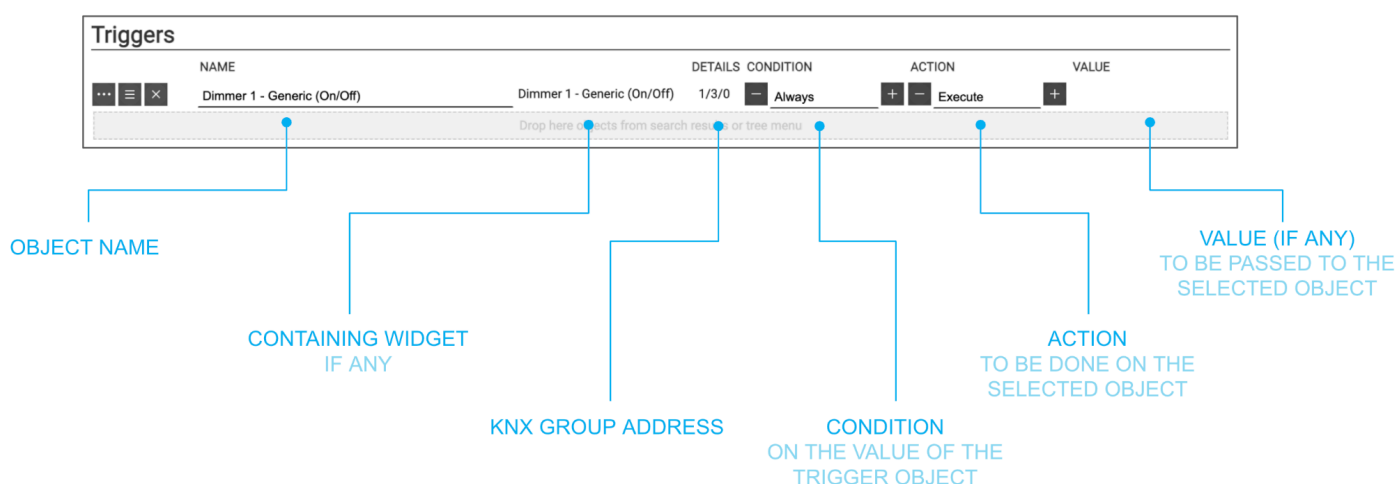
When dropping an element that has multiple sub-items, like a WIDGET, the following dialog is shown:



It is possible to select one or more sub-items of the dropped widgets that will be related to the active element (the SCENE in our example).

Once one or more objects are dropped in the TRIGGERS list, for each of those relations, it is possible to set, exactly in the same way already seen for the *arrows* of the CONNECTIONS, the following attributes:

CONDITION	A filter on the value of the <i>source node</i> that must match in order to do the <i>action</i> on the <i>destination node</i>
ACTION	The action to be done on the <i>destination node</i>
ACTION VALUE	The value to be passed to the <i>destination node</i> if supported by the selected action



3.3 Actions

If a details page also offers the ACTIONS section, it is possible to drop other objects (from the menu or the search results) in order to create *relationships* that will produce actions on those objects, when the active element changes.



TRIGGERS and ACTIONS can be seen as the two “points of view” of the same concept of a *relation*: the trigger for an object, is an action for the related items, and vice versa.



The functional effect of TRIGGERS and ACTIONS is exactly the same as the *arrows* in a CONNECTION. In fact, those relations are also visible in the TRIGGERS and ACTIONS lists, but they cannot be edited (outside the belonging connection).

It is suggested, when possible, to use the CONNECTIONS to create *relationships* among objects, not only because of the graphic editor, but also because it's more handy to find them in a second moment, instead of opening all the detail pages of the involved objects.

TRIGGERS remain a preferred approach, for example, in those situations where multiple objects must be related at a glance with a single object e.g. to bind a lot of objects (alarms) to the same NOTIFICATION message.

4 Logics

4.1 Introduction

DELÉGO SERVER allows to create logics among objects previously created for different technologies. DELÉGO SERVER's integrated logic module is based on a LUA script, automatically generated by the GUI running continuously in background. The script keeps the output objects in the state established by the logic, based on a series of rules that will be described in detail in this chapter.

4.2 Logic programs

It is possible to create one or more programs, also called "tasks", each characterized by a graphic page in which to create a logical network; to create a new task, in the Administration view navigate to

AUTOMATIONS > LOGICS > LOGIC PROGRAMS

and press the ADD (+) button in the bottom toolbar. Each task is represented by a graphic page, into which it is possible to drag technological objects (from the side menu or from the search results), and logical blocks available in the LIBRARY.

4.3 Toolbar

When you work on a task, an additional toolbar appears, containing the following buttons:

	RUN/PAUSE	Start or stop a program
	CONTINUOUS SIMULATION	Start the simulation in real-time mode
	STEP BY STEP SIMULATION	Start the simulation in step-by-step mode
	STOP SIMULATION	Stop the current simulation
	SORTING	Reorder blocks by automatic sorting following the position of the blocks
	ZOOM+	Enlarges the zoom factor of the work area
	ZOOM-	Reduces the zoom factor of the work area

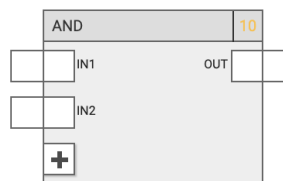
4.4 Blocks

Technology widgets, when dragged into a task, are represented by a block.

The input and output nodes, the number and type of which depend on the type of widget, respectively allow you to use the state and control that particular technological function. The same input node can be connected to multiple blocks, while the same output node cannot be connected to multiple blocks within the same task. Trigger nodes have a very specific role, which will be explained later.

The logical blocks, vice versa, must be dragged from the LIBRARY into the side menu.

In some cases the block includes a variable number of nodes (input or output); in this case, the “+” button allows you to add nodes to the block, up to the maximum number (e.g., and AND operator can have a max number of input of 10).



The logical function can only be executed correctly if the nodes are connected to other blocks. Not all nodes are strictly necessary for the correct execution of the logic; if an input node is not connected, its used the default value, which can be modified by selecting the node and opening the corresponding details panel (if it is not possible to set it, it means that the node must necessarily be connected).

Blocks can be moved in the work area, by single selection, or by selecting a rectangular area involving multiple blocks. To remove a block from the work area, select it and press the DELETE key. Connections can also be removed in the same way.

4.5 Execution order

During the simulation and compilation phases, the editor generates a Lua script which is executed cyclically. The tasks are concatenated in the script based on their ordering on the LOGIC PROGRAMS summary page; within each task, the logical blocks also have their own execution order, indicated at the top right for each block.

Under normal conditions, blocks are assigned an ascending order based on the order in which they are inserted into the program; however, it is possible to force a different execution order as follows:

- Select the affected block
- Open the details panel
- Choose “MANUAL” as the sort order
- Enter an order number, taking care to enter a number that has not yet been used

Manually sorted blocks are highlighted in yellow.

Finally, it is possible to reorder the blocks from top/left to bottom/right using the appropriate button in the toolbar.

4.6 TRG

In normal conditions, the logic module is set to the state of the output coming from the objects based on the state of the logic; as soon as these objects change state (for example, because they receive a command from the bus, external to the logic), their state is set to the default one.

This behavior is not always desirable; in some cases it is needed to control an object according to the change of the state of other objects, but if this object is controlled externally, it is needed to keep it to this last state. In other words, we want to create an “event based” logic.

In this case, it is necessary to use the input and output TRG nodes. Input triggers emit a pulse (that is, a value of 1 for a single cycle of script execution, then return to 0) each time a value is received on a given object. Exit triggers, on the other hand, allow you to establish when to issue a command on an object; every time an impulse (value 1) is sent on this trigger, the value of the object is sent on the bus, even if it has not changed since the previous execution cycle.

To ensure that the output object is commanded every time the inputs change, but can then change state in turn, if commanded externally, it is necessary:

- connect the two input TRG nodes of the first objects to the output TRG node of the third
- make sure that the command node has "trigger only" as an output strategy (therefore sending the command only when it receives the trigger for changing the state of one of the inputs)

When you connect the output trigger, the object's output strategy is automatically set to “trigger only” (and the symbol on the output nodes changes accordingly).

It is possible to set the strategy manually, to regulate when the command must be sent on the bus; in fact, the trigger can also be used to periodically to send the value of an output, in addition to its change of state, by connecting a periodic logic upstream, a bit like what happens with many KNX devices.

In the case of composite widgets, by opening the details (side panel) on the TRG node, it is possible to filter the nodes that determine the impulse (for input triggers) or that are influenced by the trigger (output), through the appropriate multiple selector.

4.7 Simulation

Once a logical task has been created, it is possible to simulate its operation within the editor, manually entering the status of the inputs and verifying the processing of the outputs in real time, also by means of the logic blocks which involve a variation of the outputs over time .

There are two types of simulation:

- *Continuous simulation*: task execution occurs in the background and is affected in real-time by changes in state of the nodes
- *Step-by-step simulation*: each task execution cycle must be launched manually, between one and the other it is possible to change the state of the nodes

The first type allows a more realistic evaluation of the logical networks created, the second allows an in-depth and timely verification of each single passage of values between blocks, and offers a higher level of diagnostics.

By pressing one of the simulation buttons (continuous or step-by-step), the editor window undergoes the following changes:

- The main menu is limited to the PROJECT view only, allowing only switching between logical programs. It is not possible to create or delete programs.
- The details panel is closed to provide maximum working space for the simulation
- Any drag&drop, connection, modification or deletion of the contents of the logic programs is blocked
- The nodes take on a color based on the state and allow you to force the value manually (as detailed below)

The color of the nodes follows the following convention:

Binary nodes	grey	Value 0 (OFF)
	yellow	Value 1 (ON)
Numeric nodes	blu	Any value

During the simulation, the editor reports in the message area information relating to the execution of the programs and manual or automatic changes of the logic state. During the step-by-step simulation some "debug" level messages are displayed

To manually set the state of a node during simulation:

- Double click on the node value (the label becomes editable)
- Delete the current value and enter the new value
- Press ENTER

The color of the node (if digital) changes according to the new value, and this is passed to the simulator, which propagates it instantly (in the case of continuous simulation) or to the next execution cycle (in step-by-step mode). You can stop the simulation at any time by pressing the stop simulation button in the toolbar (normally not accessible outside the simulation).

4.8 Execution of logic programs

Once the implementation of the tasks has been completed, with the help of the simulation, it is necessary to execute the programs, using the appropriate button on the toolbar. During execution, tasks cannot be modified.

If a task is not ready to be executed, it is possible to disable it on the LOGIC PROGRAMS detail page, before executing them.

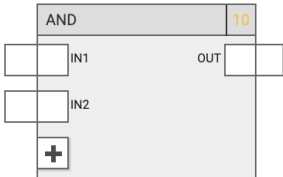
5 Library

5.1 Introduction

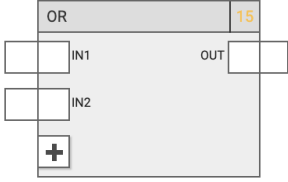
In this chapter are listed and described all the possible logics that are present in the LIBRARY and that can be used to program LOGICS.

5.2 Combinatorial logics

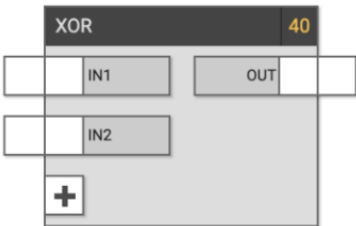
AND

DESCRIPTION:	Performs the logical AND function between two or more binary inputs (up to a maximum of 10)			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN 1 ... IN 10	Input 1 ... 10 Possible values: 0 → OFF 1 → ON	X	
	OUT	Output Possible values: 0 → OFF 1 → ON		X
	+	Add node	X	
OPTIONS:				

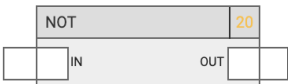
OR

DESCRIPTION:	Performs the logical OR function between two or more binary inputs (up to a maximum of 10)			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN 1 ... IN 10	Input 1 ... 10 Possible values: 0 → OFF 1 → ON	X	
	OUT	Exit Possible values: 0 → OFF 1 → ON		X
	+	Add node	X	
OPTIONS:				

XOR

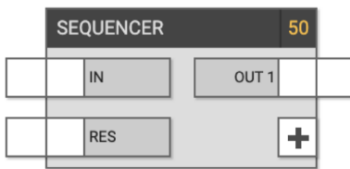
DESCRIPTION:	Performs the XOR logic function between two or more binary inputs (up to a maximum of 10)			
PREVIEW:				
NOTE:	TAG	DESCRIPTION		
	IN 1 ... IN 10	Input 1 ... 10 Possible values: 0 → OFF 1 → ON	X	
	OUT	Exit Possible values: 0 → OFF 1 → ON		X
	+	Add node	X	
OPTIONS:				

NOT

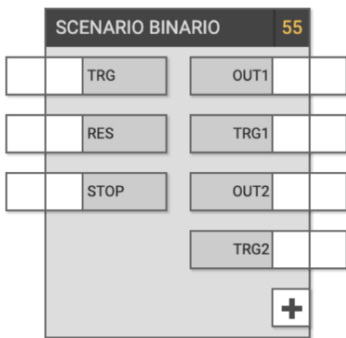
DESCRIPTION:	Performs the logical NOT function of the input			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Entrance Possible values: 0 → OFF 1 → ON	X	
	OUT	Exit Possible values: 0 → OFF 1 → ON		X
OPTIONS:				

5.3 Sequences

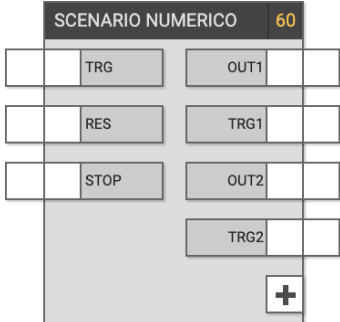
SEQUENCER

DESCRIPTION:	Upon receiving an impulse at the IN input, it activates and deactivates up to 10 Boolean outputs in sequence, keeping each of them active for a settable time			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Start of sequence Possible values: 0 → OFF 1 → ON	X	
	RES	Sequence reset Possible values: 0 → OFF 1 → ON	X	
	OUT 1 ... OUT 10	Output 1 ... 10 Possible values: 0 → OFF 1 → ON		X
	+	Add node		X
OPTIONS:	Sequence cyclical	Determines whether the sequence should be repeated once finished Possible values: TRUE/FALSE		
	Step duration 1 ... 10	Waiting time between step X and the next Possible values: from 1 second to 12 hours		

BINARY SCENARIO

DESCRIPTION:	Upon receiving an impulse in the TRG input, it executes a sequence of Boolean commands, each settable, possibly interspersing each command with a pre-set time, common to all outputs			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	TRG	Entrance Possible values: 0 → OFF 1 → ON	X	
	RES	Exit Possible values: 0 → OFF 1 → ON	X	
	STOP	Stop scenario Possible values: 0 → OFF 1 → ON	X	
	OUT 1 ... OUT 10	Output 1 ... 10 Possible values: 0 → OFF 1 → ON		X
	TRG 1... TRG 10	Trigger 1 ... 10 Possible values: 0 → OFF 1 → ON		X
	+	Add node (and related trigger)		X
OPTIONS:	Interval go out	Waiting time between output commands Possible values: 1 ... 60 (seconds)		
	Set output 1 ... 10	Value to set at output 1 ... 10 Possible values: 0 → False (OFF) 1 → True (ON)		

NUMERIC SCENARIO

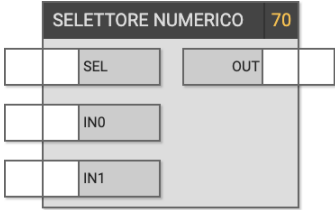
DESCRIPTION:	Upon receiving an impulse in the TRG input, it executes a sequence of numerical commands, each settable, possibly interspersing each command with a pre-set time, common to all outputs			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	TRG	Input trigger Possible values: 0 → OFF 1 → ON	X	
	RES	Reset scenario Possible values: 0 → OFF 1 → ON	X	
	STOP	Stop scenario Possible values: 0 → OFF 1 → ON	X	
	OUT 1 ... OUT 10	Output 1 ... 10 Possible values: any numeric value		X
	TRG 1... TRG 10	Trigger 1 ... 10 Possible values: 0 → OFF 1 → ON		X
	+	Add node + trigger		X
OPTIONS:	Interval go out	Waiting time between output commands Possible values: 1 ... 60 (seconds)		
	Set output 1 ... 10	Value to set at output 1 ... 10 Possible values: any numeric value		

5.4 Gates

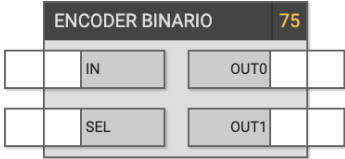
BINARY SELECTOR

DESCRIPTION:	Returns the value of one of the inputs based on the value of the SEL input which acts as a selector If SEL=False → OUT=IN0 If SEL=True → OUT=IN1			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	SEL	Input selector Possible values: 0 → OFF (in this case OUT = IN0) 1 → ON (in this case OUT = IN1)	X	
	IN 0 IN 1	Input 0, input 1 Possible values: 0 → OFF 1 → ON	X	
	OUT	Exit Possible values: 0 → OFF 1 → ON		X
OPTIONS:				

NUMERIC SELECTOR

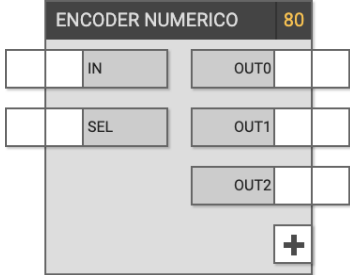
DESCRIPTION:	Returns the value of one of the inputs based on the value of the SEL input which acts as a selector If SEL=False → OUT=IN0 If SEL=True → OUT=IN1			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	SEL	Input selector Possible values: 0 → OFF (in this case OUT = IN0) 1 → ON (in this case OUT = IN1)	X	
	IN 0 IN 1	Input 0, input 1 Possible values: any numeric value	X	
	OUT	Exit Possible values: any numeric value		X
OPTIONS:				

BINARY ENCODER

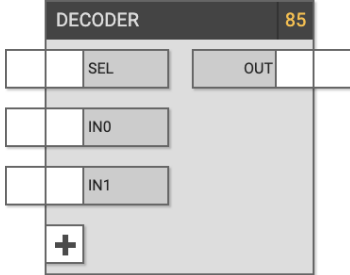
DESCRIPTION:	It places the value of IN (boolean) in one of its two outputs based on the input value SEL (boolean) which acts as a selector Number of exits: 2			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Entrance Possible values: 0 → OFF 1 → ON	X	
	SEL	Output selector Possible values: 0 → IN is set to OUT0 1 → IN is set to OUT1	X	

	OUT 0 OUT 1	Exit Possible values: 0 → OFF 1 → ON		X
OPTIONS:				

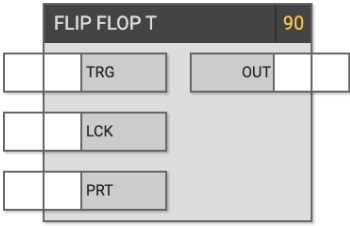
NUMERIC ENCODER

DESCRIPTION:	It places the value of IN (boolean) in one of its outputs based on the SEL (numeric) input value which acts as a selector. Number of exits: from 2 to 10			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Entrance Possible values: 0 → OFF 1 → ON	X	
	SEL	Output selector Possible values: 1 → IN is set to OUT0 2 → IN is placed on OUT1 ... 10 → IN is placed on OUT9	X	
	OUT 0 ... OUT 9	Output 0 ... 9 Possible values: 0 → OFF 1 → ON		X
	+	Add exit		X
OPTIONS:				

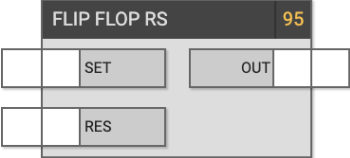
DECODER (OR NUMERIC SELECTOR)

DESCRIPTION:	Returns the value of one of the inputs (boolean) based on the input value SEL (numeric) which acts as a selector			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	SEL	Input selector Possible values: 0 → OUT is set with IN0 1 → OUT is set with IN1 ... 9 → OUT is set with IN9	X	
	IN0 ... IN9	Input 0 ... 9 Possible values: 0 → OFF 1 → ON	X	
	OUT	Exit Possible values: 0 → OFF 1 → ON		X
	+	Add input	X	
OPTIONS:				

FLIP-FLOP T

DESCRIPTION:	T-type flip-flops It works like a step-by-step relay. Every time a rising edge is proposed to its input (TRG) the output (OUT) changes state. If the LCK (Block) input is at 1 (True) the effect of the TRG is inhibited so the output never changes. If the PRT (Priority) input is at 1 the output takes on the value set in the VAL (Priority Value) parameter. It can be used for example to control the light in a corridor. It is possible to ensure that the light is normally controlled only if a brightness threshold is satisfied (this condition would end up inside the LCK) and is in any case always on during night time (Flag which should be connected to the PRT input)			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	TRG	Trigger Possible values: 0 → OFF 1 → ON	X	
	LCK	Freeze the current state Possible values: 0 → OFF 1 → ON	X	
	PRT	Priority flag Possible values: 0 → OFF 1 → ON	X	
	OUT	Output signal Possible values: 0 → OFF 1 → ON		X
OPTIONS	Priority value	Value to assign to the output in case of priority flag Possible values: TRUE / FALSE		

FLIP-FLOP RS

DESCRIPTION:	Flip-flop on type RS It is a block of elementary memory which is "Loaded" with the SET input and reset with the RES input (reset). If both inputs are at 1, the one specified by the "priority" parameter prevails. It can, for example, be used to manage an alarm signal. An alarm contact must be connected to the SET. Once it has gone to 1 the Flip Flop holds the output at 1 until it is reset by the RES. In this way, even if the alarm "normalizes" (goes to 0), the information is maintained.			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	SET	Set Possible values: 0 → OFF 1 → ON	X	
	RES	Reset Possible values: 0 → OFF 1 → ON	X	
	OUT	Output signal Possible values: 0 → OFF 1 → ON		X
OPTIONS:	Priority selection	Possible values: Set / Reset		

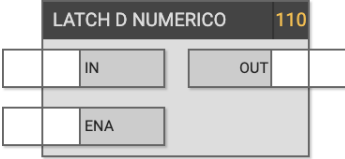
FLIP-FLOP D

DESCRIPTION:	D-type flip-flop The operation is similar to that of the Latch D with the difference that the Flip flop D operates on the variation of the CLK edge. The data in DAT is brought back to OUT only at the rising edge of the CLK signal and is maintained until the next rising edge of CLK (in fact, this block constitutes a memory cell)			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	DAT	Given Possible values: 0 → OFF 1 → ON	X	
	CLK	Clock Possible values: 0 → OFF 1 → ON	X	
	OUT	Output signal Possible values: 0 → OFF 1 → ON		X
OPTIONS	Priority selection	Possible values: DAT / CLK		

BINARY D LATCH

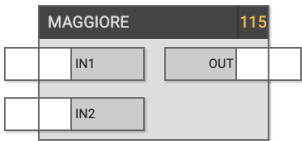
DESCRIPTION:	Binary D Latch In this block the input signal IN is propagated to the output OUT if the enable signal ENA is enabled (1). If the ENA enable signal is disabled, the last state remains at the OUT output. When enable ENA becomes active again (transition 0 → 1), the last value read in the input node IN is sent to the OUT output. Essentially, with ENA=0, the Latch block stores the last value read to send it out when the ENA becomes enabled again. The data format of IN and OUT is binary			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Entrance Possible values: 0 → OFF 1 → ON	X	
	ENA	Enable Possible values: 0 → OFF 1 → ON	X	
	OUT	Output signal Possible values: 0 → OFF 1 → ON		X
OPTIONS:	Priority selection	Possible values: In / Ena		

NUMERIC D LATCH

DESCRIPTION	NUMERIC D latch In this block the input signal IN is propagated to the output OUT if the enable signal ENA is enabled (1). If the ENA enable signal is disabled, the last state remains at the OUT output. When enable ENA becomes active again (transition 0 --> 1), the last value read in the input node IN is sent to the output OUT. Essentially, with ENA=0, the Latch block stores the last value read to send it out when the ENA becomes enabled again. The data format of IN and OUT is numeric			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Entrance Possible values: any numeric value	X	
	ENA	Enable Possible values: 0 → OFF 1 → ON	X	
	OUT	Output signal Possible values: 0 → OFF 1 → ON		X
OPTIONS:	Priority selection	Possible values: IN / ENA		

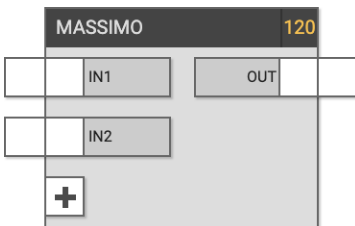
5.5 Comparisons

COMPARISON OPERATORS

DESCRIPTION:	Compares the value of the two inputs, and outputs a TRUE/FALSE value based on the specific operator Available operators: • Greater • Greater than or equal • Minor • Minor or equal • Equal • Different			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN 1 IN 2	Entrance 1, entrance 2 Possible values: any numeric value	X	
	OUT	Comparison result Possible values: 0 → OFF 1 → ON		X
OPTIONS				

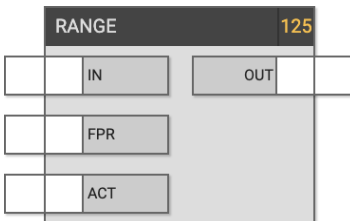
5.6 Math Operations

MATHEMATICAL OPERATORS

DESCRIPTION:	Performs a mathematical operation on the inputs, based on the type of operator Available operators: • Maximum • Minimum • Media • Sum • Subtraction • Multiplication • Division • Absolute value • Log10 • Integral			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN 1 IN 2(*) ...	Input 1, input 2... Possible values: any numeric value	X	
	OUT	Value (operation result) Possible values: any numeric value		X
OPTIONS:				

(*) The number of outputs can be limited based on the operation (e.g. division max 2, absolute value max 1)

RANGE

DESCRIPTION	Performs a linear interpolation of the IN input value based on an assigned mapping, also called "characteristic", defined by two pairs of values (X,Y). The IN value is rationed between X0 and X1, and this ratio is in turn calculated between the Y0 and Y1 values to determine the output value. If priority mode is set, a fixed value is returned. The typical field of application of this block is the conversion of values between different quantities.			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Entrance Possible values: any numeric value	X	
	FPR	Priority enablement Possible values: 0 → No priority 1 → Priority (the priority value is returned)	X	
	ACT	Direct/reverse operation Possible values: 0 → Direct operation 1 → Reverse operation	X	
	OUT	Exit Possible values: any numeric value		X
OPTIONS:	X0 Y0 X1 Y1	Characteristics of linear interpolation Possible values: any numeric value		
	Value priority	Value to be returned in case of priority enabling		

5.7 Counters

COUNTER UP, COUNTER DOWN, COUNTER UP/DOWN

DESCRIPTION	It counts the number of pulses received at the input (trigger), increasing or decreasing its value each time (depending on the type of counter). Counter types: up counter, down counter, up/down counter			
PREVIEW:				
NOTE	TAG	DESCRIPTION	IN	OUT
	IN	Enabling Possible values: 0 → Not enabled 1 → Enabled	X	
	TRG	Trigger Possible values: 0 → OFF 1 → ON (the counter is incremented)	X	
	RES	Reset Possible values: 0 → OFF 1 → ON (resets the counter)	X	
	OUT	Exit Possible values: 0 → OFF 1 → ON		X
	VAL	Current value Possible values: any numeric value		X
OPTIONS:	Preset	Default value, set when reset or when logic starts Possible values: any numeric value		

5.8 Timers & schedules

TIMER

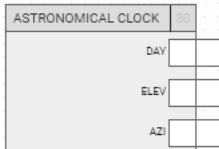
DESCRIPTION:	Delays the value received at the input by a preset time When a 1 is received at the IN input (rising edge), an internal counter starts until the time specified as "rising delay", after which the output is brought to 1; vice versa, upon receiving a 0 at the input (falling edge), the block waits the time specified as "falling delay" before setting the output to 0.			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Input ON/OFF signal Possible values: 0 → OFF 1 → ON	X	
OPTIONS:	OUT	Output ON/OFF signal, delayed by timer Possible values: 0 → OFF 1 → ON		X
	Delay in word	Delay in the propagation of the rising edge received at the input Possible values: from 1 second to 24 hours		
	Delay in descent	Delay in the propagation of the falling edge received at the input Possible values: from 1 second to 24 hours		

TRIGGER / NUMERICAL TRIGGER

DESCRIPTION:	Generates a trigger (pulse lasting one cycle) on an edge detected at the input When it receives an input 1, it sets the output to 1 for the duration of a single processing cycle, then the output is set to 0 again. In this way it is possible to generate a "pulse" for logic blocks that require it (e.g.: scenarios, sequencers etc...) on the rising edge of the input.			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	IN	Input edge	X	
OPTIONS:	OUT	Pulse of the duration of a cycle		X
	Edge	Rise or fall front to be detected at the input, in case of digital trigger		

	Condition	In the case of a numerical trigger, it allows you to set the numerical value, against which the block generates the impulse
--	-----------	---

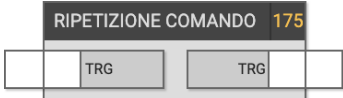
ASTRONOMICAL CLOCK

DESCRIPTION:	Generates the following information according to time and location: - day / night (twilight) - height of the sun - azimuth			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	DAY	Day / night (1 = night)		X
	ELEV	Sun elevation		X
	AZI	Azimuth		X
OPTIONS:	Latitude Longitude	Coordinates in numerical format (point as decimal separator)		
	Offset	Offset in minutes compared to day/night time		
	Time zone	Time zone of the place where the system is installed		
	Change hour	Disabled: Sunrise time/legal time not active Automatic: automatic solar/legal time Manual: manual daylight saving time setting		

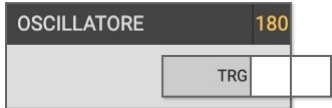
DATE/TIME

DESCRIPTION	Returns the current system time			
PREVIEW:	DATA/ORA 170 MIN HOUR MDAY MONTH YEAR WEEK WDAY ISDST			
NOTE:	TAG	DESCRIPTION	IN	OUT
	MIN HOUR	Hours and minutes that make up the current time		X
	MDAY MONTH YEAR	Date components (day, month, and year)		X
	WEEK	Week number		X
	WDAY	Day of the week		X
	ISDST	Specifies whether daylight saving time is active (1) or not (0)		X

REPEAT COMMAND

DESCRIPTION:	It is equipped with two trigger nodes; upon arrival of a trigger on the TRG input it generates N+1 triggers on the output TRG node spaced one from the other by a configurable parameter.			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	MARKET	Input trigger	X	
	MARKET	Outgoing trigger, repeated n times with interval set in the options		X
OPTIONS:	Interval (s)	Time in seconds that elapses between the generation of an output trigger and the next		
	Repetitions	Number of triggers to generate in addition to the first		

PULSE

DESCRIPTION:	Generates a trigger (pulse lasting one cycle) cyclically based on a time set in the options.			
	It actually acts as <i>clock</i> for operations that must be carried out cyclically on the basis of a pre-established time interval; it can, for example, be connected to the TRG output node of one or more objects, to force the periodic emission of the value on the field (even if it has not changed)			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	MARKET	Pulse lasting one cycle		X
OPTIONS:	Timer	Time between the emission of one trigger and the next		

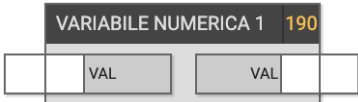
5.9 Variables

Variables allow you to pass values between different programs. The variables must be previously created using the “+” button in the appropriate section of the main menu, then they can be dragged into the programs that need to use them.

BINARY VARIABLES

DESCRIPTION:	They allow you to transfer a Boolean value between different programs.			
CATEGORY:	Binary variables			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	VAL	Value to assign to the variable Possible values: 0 → OFF 1 → ON	X	
	VAL	Current value of the variable Possible values: 0 → OFF 1 → ON		X
OPTIONS:				

NUMERIC VARIABLES

DESCRIPTION:	They allow you to transfer a Boolean value between different programs.			
CATEGORY:	Binary variables			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	VAL	Value to assign to the variable Possible values: any numeric value	X	
	VAL	Current value of the variable Possible values: any numeric value		X
OPTIONS:				

5.10 Constants

Constants can be used to set an input value to a block, with a value that cannot change during execution. Similarly to what was seen for variables, constants must be created by pressing the [+] button in the toolbar, before they can be dragged into the task.

CONSTANTS

DESCRIPTION:	They allow you to set a fixed numerical value by connecting it to other logic blocks			
CATEGORY:	Constants			
PREVIEW:				
NOTE:	TAG	DESCRIPTION	IN	OUT
	VAL	Constant value (numeric)		X
OPTIONS:				

6 Drawing Tools

6.1 Introduction

In order to increase the readability of LOGIC PROGRAMS, especially in the case of logical complex networks, the editor provides some drawing tools, in such way that the user can insert annotations or highlight areas of the program.

These tools are available in the LOGICS area of the main menu, inside of the "DRAWING TOOLS" section; they can be inserted in the logical program with a Drag&Drop like the other types of objects.

6.2 Labels

Labels allow to insert free text into programs; it is possible to enter an unlimited number of labels for each logical program. Once you drag a label into the program it can be customized by opening the details panel:

Text	Text displayed within the logic program
Color	Allows you to choose the color of the text
Font size	Allows you to choose the font size

Labels can be removed from logical programs using the corresponding "DELETE" button in the details panel, or by directly pressing the CANC key on the keyboard once selected.

6.3 Rectangular areas

It is possible to highlight one or more portions of the logic program by dragging as many colored rectangular areas from the main menu. Once a rectangular area is inserted into a program it is possible to:

- Resize it by dragging the appropriate slider in its lower right corner
- Change the color of the border, using the "color" picker in the details panel

Rectangular areas are always drawn under blocks and their connections; furthermore, they do not support multiple selection (like blocks or labels), so to customize them or remove them from the program it is necessary to click on them, one at a time, and use the tools in the details panel (change color and "DELETE" button " to remove them from the program).

7 Revision history

The following table contains the most relevant changes in the software releases, concerning the topics of this application manual:

Release	Changes	Date
2.8.0	Minor changes	15/04/2026
2.7.0	Minor changes Removed LUA code section	01/09/2025
2.6.0	First revision of the document	02/04/2025



Before the software release 2.6 the application manuals of DELÉGO SERVER were organized differently. An exact correspondence with older versions of the manuals and the changelog could not be found for this reason.