

3.2

ekinex
CONTROL YOUR LIVING SPACE

MODBUS

Fieldbus Configuration Manual



DELÉGO SERVER

Visualisation webserver

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

Index

1 Introduction.....	3
1.1 Scope of this manual.....	3
1.2 Conventions.....	3
1.3 Safety information.....	3
1.4 Warnings.....	4
1.5 Copyright.....	4
1.6 Open Source Licenses.....	4
1.7 More information.....	5
2 MODBUS Master.....	6
2.1 MODBUS lines.....	6
2.1.1 New MODBUS line.....	7
2.2 Registers.....	10
2.2.1 Creation of registers.....	11
2.2.2 Multiple reads and writes.....	14
2.3 Widgets.....	15
2.3.1 Creating a new widget.....	15
2.3.2 Duplication.....	17
2.3.3 Coolmaster.....	18
3 MODBUS Slave.....	19
3.1 Introduction.....	19
3.2 General settings.....	19
3.3 Virtual Slaves.....	20
4 Revision history.....	22

1 Introduction

1.1 Scope of this manual

This document contains all the information for configuring and visualising one or more MODBUS devices inside DELÉGO SERVER, the multiprotocol webserver for the connected buildings by EKINEX.

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

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
<i>Courier</i>	Software messages generated by the DELÉGO SERVER configuration software are reported in the “courier” font
<i>Function name</i>	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 MODBUS Master

DELÉGO SERVER is able to use the MODBUS RTU and MODBUS TCP/IP protocols to control home automation systems in which there are one or more devices that use this communication technology.

DELÉGO SERVER can simultaneously manage multiple MODBUS SLAVE devices, operating both on RTU technology and RS485 cabling, and via LAN network.

Depending on the type of communication technology, a different software installation and configuration must be foreseen:

MODBUS RTU

In this case, it is possible to use the integrated RS485 port of DELÉGO SERVER; simply connect the terminals of the slave devices to the integrated RS485 port of the webserver via 2 wires, respecting the polarity.

Alternatively, for example because you want to manage multiple RS485 lines, you can use a USB/RS485 converter.

MODBUS TCP

Devices that use the MODBUS TCP/IP protocol are put into communication with DELÉGO SERVER via the system's data network. Just like DELÉGO SERVER, each peripheral with MODBUS TCP interface will have a unique IP address and a software communication port, both of which can be set on the devices. The physical connection will take place, as in a classic LAN, through network cables or wi-fi networks, switches and routers.

2.1 MODBUS lines

DELÉGO SERVER potentially supports more than one MODBUS interface, through the creation of as many MODBUS lines on different communication channels; these lines can operate simultaneously, as long as they are appropriately configured on the communication interfaces available in DELÉGO SERVER:

- Integrated RS485
- USB
- LAN network

For each line it is possible to create a certain number of objects (identifiable as MODBUS registers), relating to the various devices connected to it, which can subsequently be inserted into the supervision graphic pages. It is also possible to manage more than one slave device if connected "in cascade" to the same RS485 line, in this case creating a single line in DELÉGO SERVER and discriminating the devices through the slave address, as detailed further below.

It is possible to directly create one or more graphic components within a MODBUS line, which are in turn composed of a list of registers, in order to simplify, speed up and make the management of numerous slave devices within the same line more orderly.

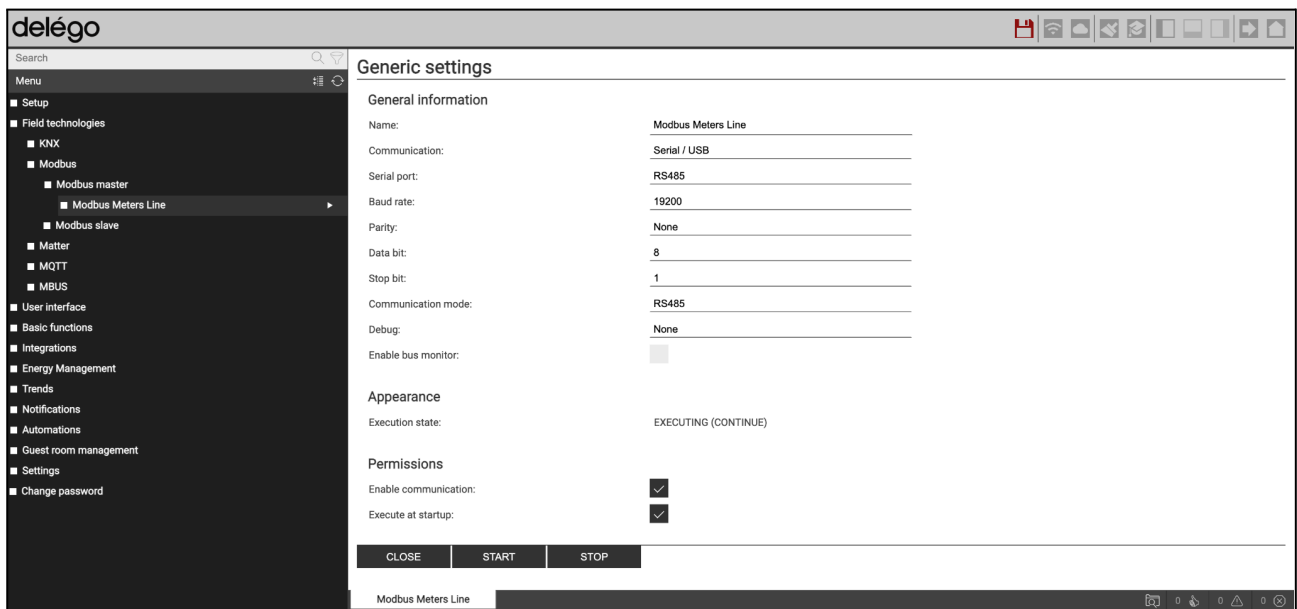
It is essential to create a single bus line for each DELÉGO SERVER communication port; otherwise, communication conflicts occur.

2.1.1 New MODBUS line

To create a new MODBUS line, proceed as follows:

- Access the configuration area
- Open the FIELD TECHNOLOGIES section of the side menu
- Select the “MODBUS” item
- Press the “ADD” button in the toolbar at the bottom (“+” symbol)

A new bus line is created and added to the list; at this point, by clicking on it and selecting the "three dots" on the right (or, alternatively, after selecting it, press the "EDIT" button on the toolbar), you access the following page:



The following parameters are available in the “GENERAL DATA” section:

NAME	MODBUS line identification label
COMMUNICATION	Configure the MODBUS communication protocol, with the following options: • Serial/USB • Net
DEBUG	In the DEBUG field there are two entries: • None • Log files

	By selecting "Log file", all the events that will affect the MODBUS line, both in transmission and in reception, will be written to a file, which can be downloaded with the "Download Log" button, with related errors, line settings and other information. This tool can prove useful during the commissioning of MODBUS devices with DELÉGO SERVER but, once the system is functioning, it is recommended to activate it only in case of necessity (e.g. maintenance or software changes).
ACTIVATE BUS MONITOR	By selecting this option, all MODBUS traffic is stored in a temporary file, which can be consulted via the BUS MONITOR button (available only if the option is active).

In case of serial communication, the following options are also available:

SERIAL PORT	Establishes which communication channel will be used by DELÉGO SERVER to establish the connection with the connected MODBUS devices. It is possible to use any of the serial ports of the webserver: • RS485 port • RS232 port (*) • USB port 1 <i>(*) Only available on the 4 DIN-rail module hardware version</i>
BAUD RATE	Sets the communication speed of the MODBUS channel in bits/s. The speeds must be the same as those of the devices connected to the server: • 1200 • 2400 • 4800 • 9600 • 19200 • 38400 • 57600 • 115200
PARITY	MODBUS line communication parameter to be selected based on that set on the devices with which communication must take place: • EVEN • ODD • NONE
DATA BIT	MODBUS line communication parameter to be selected based on that set on the devices with which communication must take place: • 5 • 6

	• 7 • 8
STOP BIT	MODBUS line communication parameter to be selected based on that set on the devices with which communication must take place: • 1 • 2
COMMUNICATION MODE	Establishes which serial protocol to use: • RS485 • RS232

For IP-based communication, the following options are available instead:

IP ADDRESS	IP address of the MODBUS device.
PORT	Communication port used by the MODBUS device.

The "Execution status" item informs the user of the status of the MODBUS line: "EXECUTING (CONTINUE)" if there is communication, "STOPPED" if there is no transmission and reception of packets.

To act on line communication, two buttons are available: "START" and "STOP" which respectively allow you to start or stop communication with the MODBUS line. Furthermore there is the "CLOSE" button, which simply closes the page of the open MODBUS line.



After each change to the configuration, it is necessary to stop and start the communication again or, alternatively, press the "REFRESH" button at the beginning of the sections in the lower part of the page,, as illustrated below.

By activating the EXPERT mode () , the following advanced settings are available:

QUERY TIME [ms]	Waiting time between querying one register and the next; time expressed in milliseconds.
COMMAND TIME [ms]	Waiting time, after sending a command on the bus, before resuming polling, or executing another command in the queue.
WAITING TIME AFTER QUESTION CYCLE [ms]	Waiting time, at the end of the polling scan of the registers configured in the line, before restarting from the beginning
CONNECTION TIMEOUT [ms]	Waiting time for response from the device, before declaring <i>timeout</i> and move to the next register

KEEP THE CONNECTION ACTIVE	By enabling this flag, the server does not disconnect communication with the slave device at the end of a query cycle, but keeps it perpetually busy.
----------------------------	---

It is possible to optimise the reading of contiguous registers by automatically reading them in groups instead of one-by-one; this is done automatically by the Modbus master process despite of the configuration of the registers, according to the following parameters:

OPTIMIZE READING OF CONTIGUOUS ReGISTERS	By enabling this flag, contiguous registers are automatically read together, up to a maximum number set with the next parameter
MAXIMUM NUMBER OF REGISTERS FOR A SINGLE READ	Enter the maximum number of registers that can be read simultaneously

2.2 Registers

Once communication has been configured, it is possible to create the registers of the Slave devices connected to the DELÉGO MODBUS line. The table of read and/or write registers of each peripheral connected to the MODBUS line can be found in the technical reference manual of the product.

Inside you will be able to find useful information for DELÉGO SERVER to communicate with the device, among which the fundamental ones to know before proceeding with the integration are:

COMMUNICATION PARAMETERS	They vary depending on which type of protocol is used by the MODBUS device. In the case of <u>MODBUS RTU</u> the entries are as follows: ▪ Baud rate ▪ Equality ▪ Bits of data ▪ The bit is stopped ▪ With the <u>MODBUS TCP</u> you need to know: ▪ IP address ▪ Port
SLAVE ADDRESS	Each MODBUS peripheral, whether it communicates via RTU or TCP protocol, has a unique address on the network. Unless the manufacturer specifies it, it can be changed. The procedure to follow is specified in the manual and is usually performed through peripheral pre-configuration software or via hardware with dedicated switches. The admissible addresses in a MODBUS network range from 1 to 247. The Master device of the line, which in our case will always be DELÉGO, it does not require an address for communication while address 0 is reserved for "broadcast" messages.
FUNCTIONS	In MODBUS the "Function" item indicates the second byte of a message. It is sent by the Master and indicates which Slave register

	table it must enter to access the data and whether the operation to be carried out is a read or a write. <u>Reading functions:</u> FC 01: Read coil status FC 02: Read input status FC 03: Read holding register FC 04: Read input registers <u>Writing functions:</u> FC 05: Force single coil FC 06: Preset single register FC 15: Force multiple coils FC 16: Preset multiple registers It is essential to know each register of the network devices with which MODBUS function it must be written and read in order to access the correct data; as with other information, this is also contained in the manual for the device.
REGISTERS	A register contains the data to be read and/or written with DELÉGO. Each device has a mapping that identifies each register and declares its content: for example, a register can contain a temperature value, an on/off command, an alarm signal. As will be seen later, the registers will be the "objects" available to the user on the supervision page to be able to interact with the device via MODBUS, whether reading, writing or both. Being objects of DELÉGO they can be used for each of the server functions: planning, logic, conditions, composite objects etc.
DATA TYPES	Indicates for each register the set of values that the variable can take on e operations that can be carried out on it. For example, on an integer type register (INT) it is only possible to carry out simple arithmetic operations on integer type numbers. The device register table most of the time also reports the type of encoding needed to interpret the value.

Once you have obtained all the information listed above about the MODBUS device, we recommend carrying out the writing and reading tests with MODBUS software from your PC. This step allows you to delve deeper into the functionality of the device by being able to test it "on the bench" before proceeding with the integration with DELÉGO SERVER.

It is sufficient to install MODBUS Master software on your PC and connect it with a serial cable or network cable to the peripheral, depending on whether the protocol is MODBUS RTU or TCP, to be able to access the device and its registers. It will thus be possible to test all the key aspects that will allow integration: communication parameters, addresses, functions, registers and data types. Once you have managed to interact with the MODBUS device, solving any problems encountered, integration on DELÉGO SERVER will be simpler and faster.

Obviously this procedure must be followed for each peripheral model that you want to include in a DELÉGO SERVER line.

On the Web you can find many free MODBUS Master software, among which we recommend the following:

http://www.modbustools.com/modbus_poll.html (30 day free trial)

<https://oceancontrols.com.au/OCS-011.html>

<http://www.qmodbus.sourceforge.net/>

After having obtained all the useful information for integrating one or more MODBUS devices with DELÉGO and having carried out the writing and reading communication tests, you can proceed with creating the registers in the webserver.

2.2.1 Creation of registers

To create one or more registers for each slave connected to the MODBUS line, proceed as follows:

- Access the MODBUS Line
- Type the number of registers to create in the text field next to the "ADD" button within the "MODBUS REGISTERS" section title
- Press the ADD button

As many new registers as required are automatically created.

Once created, the registers can be renamed simply by typing the new name in the appropriate text box; It is also possible to access the register detail sheet through the corresponding edit button; the "DELETE" button, on the other hand, removes (permanently) the register from the project.

For each register you can set the following properties:

NAME	Label used within the supervision to identify the MODBUS Slave register. E.g.: "external temperature", "living room light" etc...
SLAVE	Address of the slave device to which the register belongs.
REGISTER	Register address.
TYPE	Establishes the type of data contained within the register: • Booleano (ON/OFF) • Numeric – rational (with comma) • Numeric – integer (no decimal values) Note: It is recommended to use integer encoding if you do not expect comma values to be handled.
R (READABLE)	Flag to enable if the register can be read.
READ	If the register is readable, establish which function to perform the query with: • FC1 (Read Coil Status) • FC2 (Read Input Register) • FC3 (Read Holding Registers) • FC4 (Read Input Registers)
W (WRITABLE)	Flag to enable if the register is writable.

WRITE	If the register is being written, choose the appropriate function to execute it: • FC05 (Force Single Coil) • FC06 (Preset Single Register) • FC15 (Force Multiple Coils) • FC16 (Preset Multiple Registers)																												
R/W	Allows you to establish the number of registers to read and/or write simultaneously. For further details, please refer to the appropriate section below																												
ENCODE	Establishes the type of coding to be used for interpreting the data within the register. The following encodings are supported: <table> <tr> <td>Unsigned integer <i>Unsigned integer</i></td><td>1 register (2 byte)</td></tr> <tr> <td>Whole with sign <i>Signed integer</i></td><td>1 register (2 byte)</td></tr> <tr> <td>Long unsigned <i>Unsigned Long</i></td><td>2 registers (2 bytes) *</td></tr> <tr> <td>Long with sign <i>Signed long</i></td><td>2 registers (4byte) *</td></tr> <tr> <td>Long with inverted sign <i>Signed long inverted</i> <i>Note: the order of the two registers is reversed compared to the "long" coding</i></td><td>2 registers (4 bytes) *</td></tr> <tr> <td>Floating point</td><td>2 registers (4 bytes) *</td></tr> <tr> <td>Inverted floating point <i>Note: the order of the two registers is reversed compared to the "floating point" coding</i></td><td>2 registers (4 bytes) *</td></tr> <tr> <td>Bit mask <i>Bit mask</i></td><td>1 register (2 byte)</td></tr> <tr> <td>Double with sign <i>Signed double</i></td><td>4 registers (8 bytes) *</td></tr> <tr> <td>Unsigned double <i>Unsigned double</i></td><td>4 registers (8 bytes) *</td></tr> <tr> <td>Integer with sign bits</td><td>1 register (2 byte)</td></tr> <tr> <td>Along with sign BIT</td><td>2 registers (4 bytes) *</td></tr> <tr> <td>Long inverted with sign BIT</td><td>2 registers (4 bytes) *</td></tr> <tr> <td>Double with BIT sign</td><td>4 registers (8 bytes) *</td></tr> </table> (*) In order to be selected, 4-byte encodings require the R/W field to be previously set to "2", as two registers must be read/written together. Similarly, the field should be set to 4 for 8-byte encodings.	Unsigned integer <i>Unsigned integer</i>	1 register (2 byte)	Whole with sign <i>Signed integer</i>	1 register (2 byte)	Long unsigned <i>Unsigned Long</i>	2 registers (2 bytes) *	Long with sign <i>Signed long</i>	2 registers (4byte) *	Long with inverted sign <i>Signed long inverted</i> <i>Note: the order of the two registers is reversed compared to the "long" coding</i>	2 registers (4 bytes) *	Floating point	2 registers (4 bytes) *	Inverted floating point <i>Note: the order of the two registers is reversed compared to the "floating point" coding</i>	2 registers (4 bytes) *	Bit mask <i>Bit mask</i>	1 register (2 byte)	Double with sign <i>Signed double</i>	4 registers (8 bytes) *	Unsigned double <i>Unsigned double</i>	4 registers (8 bytes) *	Integer with sign bits	1 register (2 byte)	Along with sign BIT	2 registers (4 bytes) *	Long inverted with sign BIT	2 registers (4 bytes) *	Double with BIT sign	4 registers (8 bytes) *
Unsigned integer <i>Unsigned integer</i>	1 register (2 byte)																												
Whole with sign <i>Signed integer</i>	1 register (2 byte)																												
Long unsigned <i>Unsigned Long</i>	2 registers (2 bytes) *																												
Long with sign <i>Signed long</i>	2 registers (4byte) *																												
Long with inverted sign <i>Signed long inverted</i> <i>Note: the order of the two registers is reversed compared to the "long" coding</i>	2 registers (4 bytes) *																												
Floating point	2 registers (4 bytes) *																												
Inverted floating point <i>Note: the order of the two registers is reversed compared to the "floating point" coding</i>	2 registers (4 bytes) *																												
Bit mask <i>Bit mask</i>	1 register (2 byte)																												
Double with sign <i>Signed double</i>	4 registers (8 bytes) *																												
Unsigned double <i>Unsigned double</i>	4 registers (8 bytes) *																												
Integer with sign bits	1 register (2 byte)																												
Along with sign BIT	2 registers (4 bytes) *																												
Long inverted with sign BIT	2 registers (4 bytes) *																												
Double with BIT sign	4 registers (8 bytes) *																												

BIT	Entry available only if the register on which the writing and reading operations are carried out is encoded as Bitmask. This coding is used where every single bit has a precise meaning (e.g. a one-byte register where every single bit is an on/off output of an 8-channel actuator). The Bit item therefore establishes on which bit within the mask the reading and/or writing operation will be carried out.
MULTIPLIER	The contents of the register are multiplied with the value set in this field. Es. record (temperature): 235; multiplier: 10; value in DELÉGO: $235/10 = 23.5$
OFFSET	Allows you to add a fixed value to the value read by the device. Es. record (temperature): 235; multiplier: 10; offset: 10; value in DELÉGO: $235/10 - 20 = 3.5$
COV	In the case of numerical coding, it allows you to set a threshold, below which value changes are ignored and not "propagated" within the supervision. <i>Note: For decimal values, use the period as a separator</i>
STATE	Current value of the register.
LIBRARY	Choice of graphic set for display in the registry viewing environment. Allows you to choose which icon to use in the visualization of DELÉGO to graphically represent the register. The library of available icons depends on the selection of the Register Type (Boolean or Numeric).

Press the "REFRESH" button on the left of the MODBUS Registers bar to confirm the creation, or modifications, of the registers and make them active.

2.2.2 Multiple reads and writes

It is possible to read and/or write simultaneously on more than one register; this option is necessary (by setting the value "2") for registers that adopt 4 byte encodings, but more generally it can be used (if the slave device supports it) to operate on contiguous registers with a single read/read operation writing, thus speeding up communication.

To activate multiple reading/writing, enter a value other than "1" in the R/W field; as many "sub-objects" are automatically created (with respect to the main register) as there are reads/writes set, automatically configured with the registers adjacent to the first:

Modbus registers															1	ADD	REFRESH
		NAME	COM SLAVE REGISTER TYPE		READABLE READ	WRITABLE WRITE	R/W	ENCODE	BIT	MULTIPL.	OFFSET	COV	STATE	LIBRARY			
...		Register		110	Numeric - Float	FC3 (Read)	FC16 (Pres)	5	Unsigned integer	1			0	123			
		Register		111	Numeric - Float	FC3 (Read)	FC16 (Pres)		Unsigned integer	1			0	123			
		Register		112	Numeric - Float	FC3 (Read)	FC16 (Pres)		Unsigned integer	1			0	123			
		Register		113	Numeric - Float	FC3 (Read)	FC16 (Pres)		Unsigned integer	1			0	123			
		Register		114	Numeric - Float	FC3 (Read)	FC16 (Pres)		Unsigned integer	1			0	123			

The sub-objects are constrained to share some attributes with the first, such as read/write flags and encoding. Conversely, it is possible to independently set any multiplication factor, the COV and the set of icons, given that they represent independent values in the graphic supervision (which could also represent different types of quantities).

2.3 Widgets

In the same way as KNX, it is possible to create widgets based on MODBUS technology, with the same available templates. Advantages instead of creating the single registers:

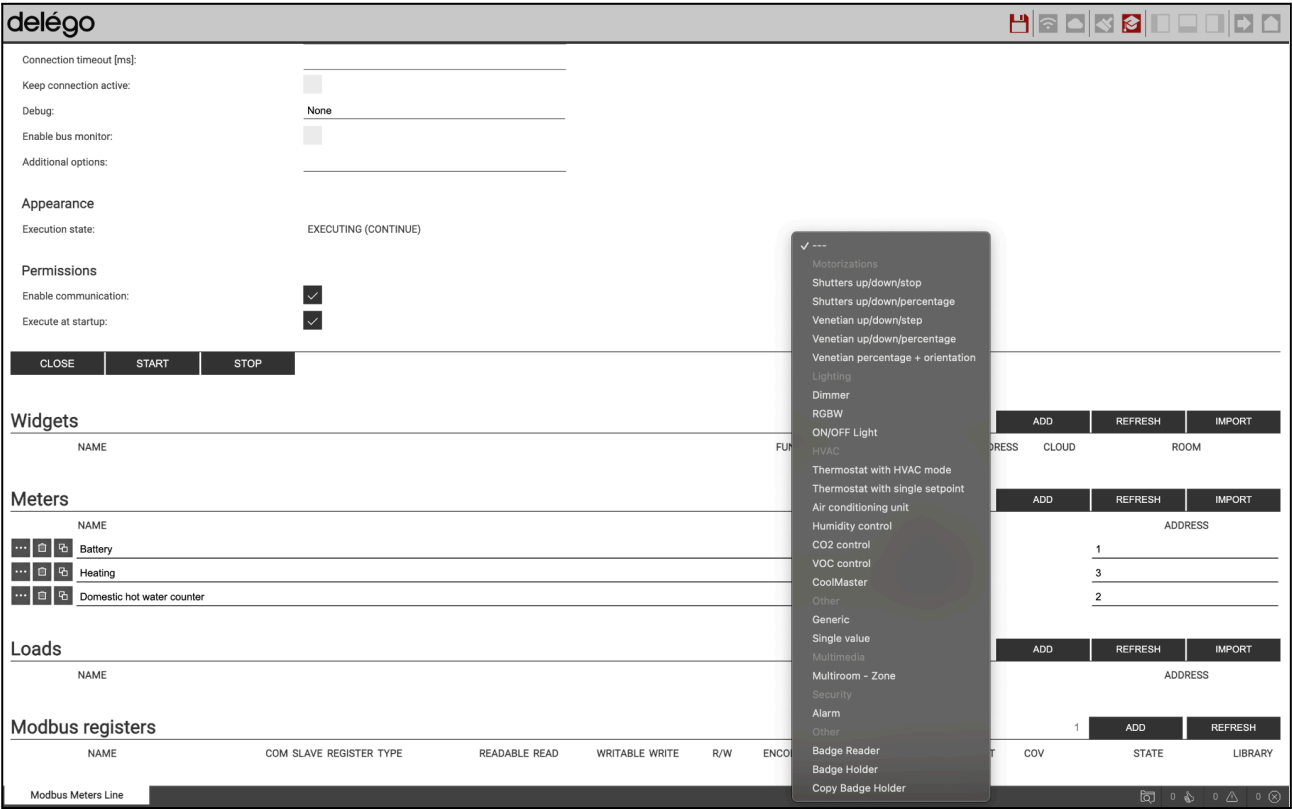
- the register list comes pre-filled based on the model chosen, and it is only necessary to configure the relevant communication mapping
- the widget object can be directly inserted into the rooms
- It is possible to duplicate a widget after configuring it, in order to speed up the construction of repetitive configurations
- the list of registers is more readable, as it is divided between the different slave devices, and not a single list

From the point of view of communication with the devices, and the configuration of the information required for each register, nothing changes compared to what was seen previously; In fact, WIDGETS are exclusively graphic structures and do not have an effective impact on protocol aspects.

2.3.1 Creating a new widget

To insert a new graphic component inside a MODBUS line, proceed as follows:

- Select the template you want to use, choosing from those available (the list is the same as the one that can be used to manually create a graphic component)
- Enter the number of graphic components to create, if different from 1
- Press the ADD button



Once the graphic component has been created, access its tab with the edit button; in the initial section of the page it is possible to specify the following (in addition to the parameters / flags common to all DELÉGO object types):

NAME	Identification label for the graphic component, also shown in supervision
ADDRESS	If specified, forces the slave number to all associated registers
ROOM	Allows you to choose the environment to which the graphic component belongs
FUNCTION	Allows you to choose the category to which the graphic component belongs

The screenshot shows the 'delégo' web interface. On the left is a dark sidebar menu with options like Setup, Field technologies, KNX, Modbus, Matter, MQTT, MBUS, User interface, Basic functions, Integrations, Energy Management, Trends, Notifications, Automations, Guest room management, Settings, and Change password. The main area is titled 'Generic settings' and contains 'General information' (Name: Widget 1, Address: 2, Room: ---), 'Appearance' (Function: None), and 'Cloud' (Enable cloud sync: ☐, Voice control:). Below these are buttons for CLOSE, REFRESH, CLONE, and EXPORT. The 'Modbus registers' section shows a table of registers with columns for FUNCTIONALITY, ENABLE LABEL, VOICE COM SLAVE REGISTER TYPE, R, READ, W, WRITE R/W, ENCODE, BIT, MULTIPL, OFFSET, COV, STATE, and LIBRARY. The table lists four registers: Movement (Up/Down), Percent value, Slats (step up/down), and Stop.

FUNCTIONALITY	ENABLE LABEL	VOICE COM	SLAVE	REGISTER	TYPE	R	READ	W	WRITE R/W	ENCODE	BIT	MULTIPL	OFFSET	COV	STATE	LIBRARY
Movement (Up/Down)	☒		2	10	Boolean (	☒	---	☒	FC0f	1	Bit mas	0			Up	
Percent value	☒		2	12	Numeric -	☒	---	☒	FC0f	1	Unsign	1			0%	
Slats (step up/down)	☒		2	13	Boolean (	☒	---	☒	FC0f	1	Bit mas	2			Open	
Stop	☒		2	16	Boolean (	☒	---	☒	FC1f	1	Bit mas	3			0	

In the MODBUS REGISTERS section there are all the registers created automatically, based on the template chosen initially.

If there are unnecessary registers in the list (because the template provides them, but they will not actually be used in the specific case) simply delete them with the appropriate trash can button.

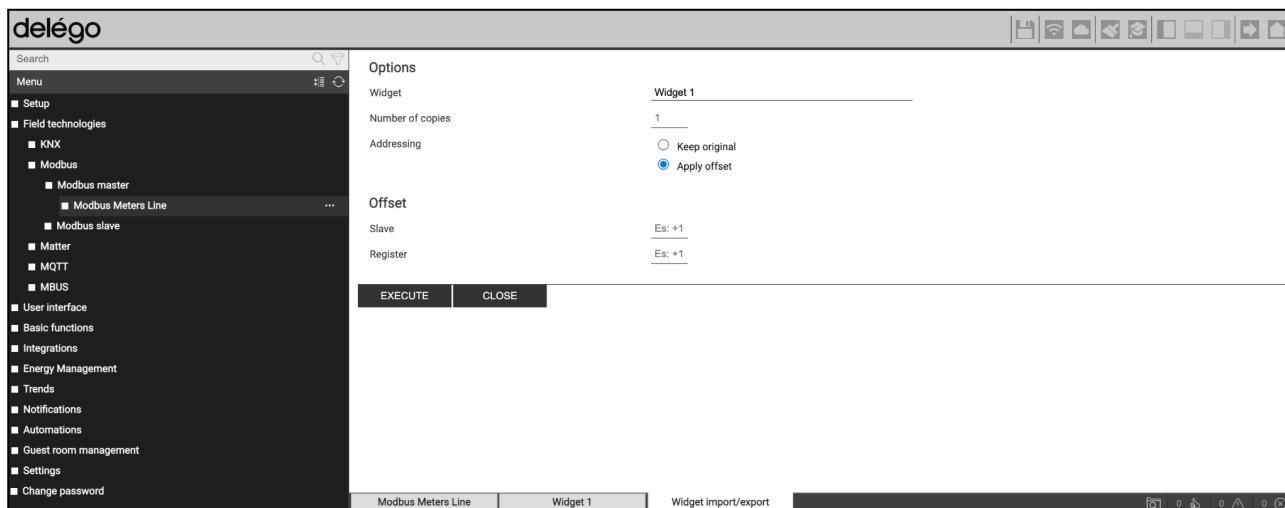


A graphics component does not necessarily have to coincide with a MODBUS slave device; for this reason, it is possible not to specify the slave address above, and fill in the SLAVE field for each of the addresses. Typically, however, graphic components are used precisely to automate the graphic creation of an element that represents a physical device.

After each change to the list of registers, it is necessary to press the UPDATE button to restart communication on the MODBUS line.

2.3.2 Duplication

Once a graphics component is successfully configured, you can duplicate it to speed up the configuration of repetitive elements. To do this, press the CLONE button in its detail sheet or, alternatively, the duplicate button directly from the list of graphic components in the line; in both cases, the following window is shown:



The WIDGET field is preselected with the starting element and cannot be modified; on the other hand, it is possible to establish how many copies must be made, and whether the corresponding registers must be created identical to the original, or vice versa by applying an offset.

The offset can be either on the slave address or on the register; indicate an incremental number (using for example "+1" to increase, "-1" to decrease) which will be applied, for each of the graphic components, compared to the original.

When finished, the new graphic components appear in the line list; by entering their tab, it is possible to verify that the creation of the registers has correctly implemented the offset, and possibly modify the configuration according to your needs.

2.3.3 Coolmaster

A particular type of widget is the template dedicated to devices [Coolmaster](#), which allow you to interface different types of air conditioners via MODBUS TCP protocol. Please refer to the dedicated APPLICATION MANUAL.

3 MODBUS Slave

3.1 Introduction

DELÉGO can act as *slave* making control of your objects available to a device *master* external, both to read the status of the connected systems and to be controlled.

In this case, no objects should be created to represent other devices *slave*, as seen so far, but it is necessary to operate in reverse, dragging objects already present in the project into the section reserved for MODBUS SLAVE functionality.

3.2 General settings

Before proceeding with the association with the supervision objects, it is necessary to establish the communication method. DELÉGO SERVER can also be connected to more than one *master* simultaneously (multi-connection) as long as they all adopt the same communication parameters.

By accessing the section

FIELD TECHNOLOGIES > MODBUS > MODBUS SLAVE

it is possible to set the following parameters:

SLAVE	Slave number that DELÉGO must use; important in case of communication on RS485 in the presence of other devices connected to the same line. If left blank, it takes on the value 1.
COMMUNICATION	Establish whether communication should take place via network - and in this case whether via TCP or UDP - or via serial
PROTOCOL	Based on the previous choice it is possible to set the communication protocol. In the case of network communication, the following options are available: TCP/UDP RTU over TCP/UDP ASCII over TCP/UDP In the case of serial communication vice versa: RTU ASCII
IP PORT	In case of network communication, set the listening port. Ports in the range 1024-49150 are available; default: 10502
WHITE LIST IP	Concatenation (via space) of individual IP addresses or classes of addresses from which the connection is accepted; for example: 192.168.0.0/24 192.168.1.0/24 192.168.0.14/32

	Optional field; if not specified, connections from all network addresses are accepted
SERIAL PORT BAUD RATE EQUALITY DATA BITS BIT IN STOP	In the case of serial mode, specify the connection port (default: integrated RS485) and the communication parameters

By activating the advanced mode (general toolbar at the top right), the following options are also available:

UPDATE MODE	Specify with which strategy the registers should be updated: COV: on change of value (Change Of Value) EVERY WRITING: every time a value is received, even if it is the same as the previous one AUTO: automatically determined by DELÉGO
DELAY IN RESPONSE	Delay between receiving a request and responding Default: 5 ms
TIMEOUT FOR VALID RESPONSE	Maximum time between one valid message and the next before considering the communication in alarm

Finally, the following flags are available in the PERMISSIONS section:

ENABLE COMMUNICATION	Select this flag to enable MODBUS functionality <i>slave</i> . Default: Not enabled
RUN AT STARTUP	Select this flag to start the process of listening for mode requests <i>slave</i> , if communication is enabled. Default: Enabled

Whenever the parameters indicated above are modified, it is necessary to stop and start the listening process, or alternatively press the UPDATE button in the lists described below. The first time, after enabling communication, you need to press the START button; the RUNNING STATUS must be "RUNNING (CONTINUOUS)".

3.3 Virtual Slaves

To expose one or more supervision objects to interrogation by a device *master* external, it is necessary to group them within a VIRTUAL SLAVE, which appears externally as if it were a MODBUS device *slave* to all intents and purposes.

DELÉGO SERVER may present one or more *virtual slaves* depending on your preferences; we recommend using more than one *virtual slave* in the event that multiple similar pieces of information need to be displayed but referring to multiple home automation devices, such as - for example - multiple thermostats, all

characterized by the same list of sub-objects (and therefore, as will be seen shortly, the same MODBUS register mapping).

To create a new *virtual slave* simply press the ADD button in the list of the same name, possibly specifying the number of devices to create if more than one.

Once a device is inserted *virtual slave*, it is necessary to specify its slave address in the appropriate ADDRESS field; it is also advisable to assign a name to identify it more easily within the project, even if it is not strictly essential (since this type of object is typically not included in the supervision, being aimed solely at MODBUS communication with the devices *master*).

By accessing the detail sheet (using the corresponding edit button, characterized by the "three dots") it is possible to specify the following:

NAME	Identification label, as seen above
ADDRESS	Slave number, as seen above
UPDATE MODE	If specified, allows you to override the general update mode specifically for the selected slave device. If left blank, the general settings seen previously are used
DELAY IN RESPONSE MAXIMUM TIMEOUT FOR VALID RESPONSE	If specified they allow you to overwrite the corresponding ones <i>timeout</i> specified at a general level

4 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	Introduced the optimization for the reading of contiguous registers	15/04/2026
2.7.0	Minor changes	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.