Inveo Sp. z o.o.

IND2

Instruction Manual

Overview

Warning

This manual applies only to the firmware version v0.18.5 onwards. Inveo does not guarantee that the information contained in this document applies to previous firmware revisions.



Purpose of the device

The IND2 is a modern RFID reader, designed for a wide range of applications such as industry, services, trade, and logistics. Thanks to comprehensive functionality, the device will perform well as a machine log-in device, access control to rooms (eg. warehouses), part of a customer loyalty system or a working time recorder.

Equipped with built-in memory, IND2 allows storing of saved logs, which allows it to function autonomously without need for an external server. IP65 ingress protected enclosure guarantees device reliability even in difficult external conditions, provided a cable gland is installed properly.

The device supports integration with different system thanks to Modbus TCP/RTU and HTTP (client/server) protocol support. It has numerous inputs and outputs that allow control over the device's elements such as the display, buzzer or additional inputs/outputs. Power can be supplied from various sources, optionally via PoE, which makes it flexible and easily adaptable to specific user requirements and installation conditions.

Changelog

Table of contents

- Overview
  - Purpose of the device
  - Changelog
- Device construction
  - Technical characteristics
  - Connectors
  - General characteristics
- Network configuration
  - LAN and Wi-Fi configuration
  - Checking the network status
- Device configuration
  - Device status
  - Administration
    - Module name
    - Changing the password
    - Time settings
    - SNTP time server
    - Diagnostics, events
    - Syslog
    - Special
  - I/O Settings
    - Inputs
    - Outputs
    - LCD display
    - Function keys (RCP2 only)
    - Predefined variables
  - RFID reader
  - Card Action
    - Action settings
    - Action known card
    - Action unknown card
    - Action hold card
    - Action remove card
- Managing cards and events

- Card database
  - Managing tags from a browser
  - CSV upload
  - Downloading the database
- Event logging
  - Events preview
  - Log download
  - Card log transmission

- Communication protocols
  - HTTP/HTTPS Server
    - HTTP GET – Status
    - HTTP GET – Control
    - HTTP GET – card database
  - HTTP Client
    - HTTP Client – Poll request
    - HTTP Client – Card action event request
    - HTTP Client – Control
  - MQTT Client
    - Publish topic
    - Subscribe topic
    - E-mail
  - Modbus TCP/RTU
    - Holding Registers addresses
    - Single Coil addresses
  - Lua scripts
    - Available system functions
    - Lua libraries

- Factory settings, backup
  - Restoring factory settings
  - Backup
  - Emergency firmware restore
- Warranty and manufacturer's liability
  - Storage, operation and transport conditions
  - Disposal and decommissioning

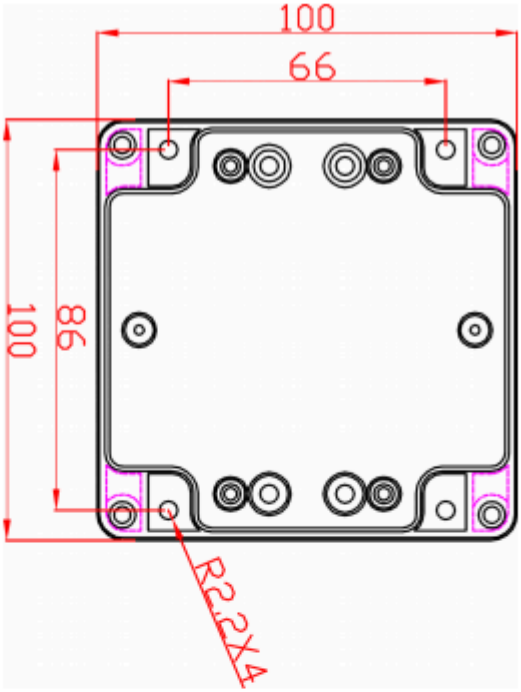
Device construction

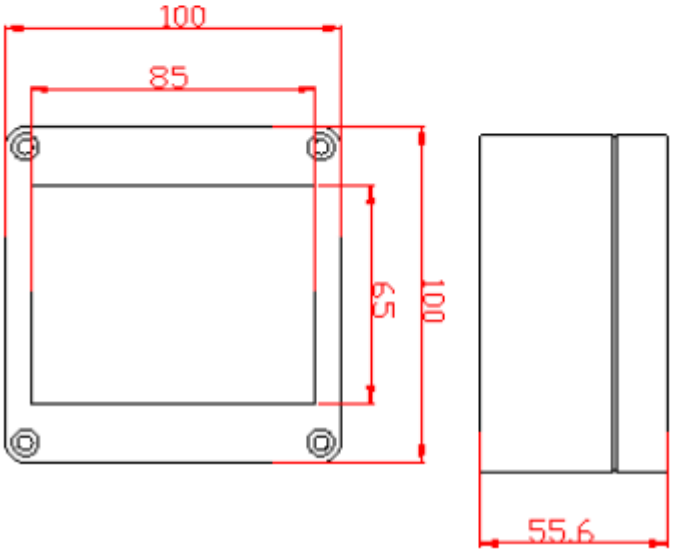
Technical characteristics

Parameter	Description
Power supply	**IND2** – **10-24V DC**, 3,5mm screw terminal
	IND2 PoE – **10-24V DC**, 3,5mm screw terminal; **PoE:** 33-57V PoE IEEE 802.3af
Power consumption	**Max. 2,5W** (~200mA @ 12V)
Transponders	**The following IND2 versions are available:** **Mif:** Version supporting Mifare Classic® transponders **Uni:** Unique EM4100, EM4102 transponders **HT2:** HITAG 2 transponders **Ico:** Version dedicated for ICODE® transponders **Read distance:** up to 8cm **Built-in memory:** 5000 tags, 100 000 events
Inputs	- **Amount of inputs:** 2 - **Input type:** Normally Open (NO)
Outputs	- **Amount of outputs:** 2 - **Output type:** NO (relay), - **Maximum load:** 1A @ 30V DC
Communication	- **Ethernet** 10/100Mbps Full Duplex, RJ45 port - **RS-485 port**, Modbus RTU **Wi-Fi** *(option)*
Enclosure ingress protection	**IP65**

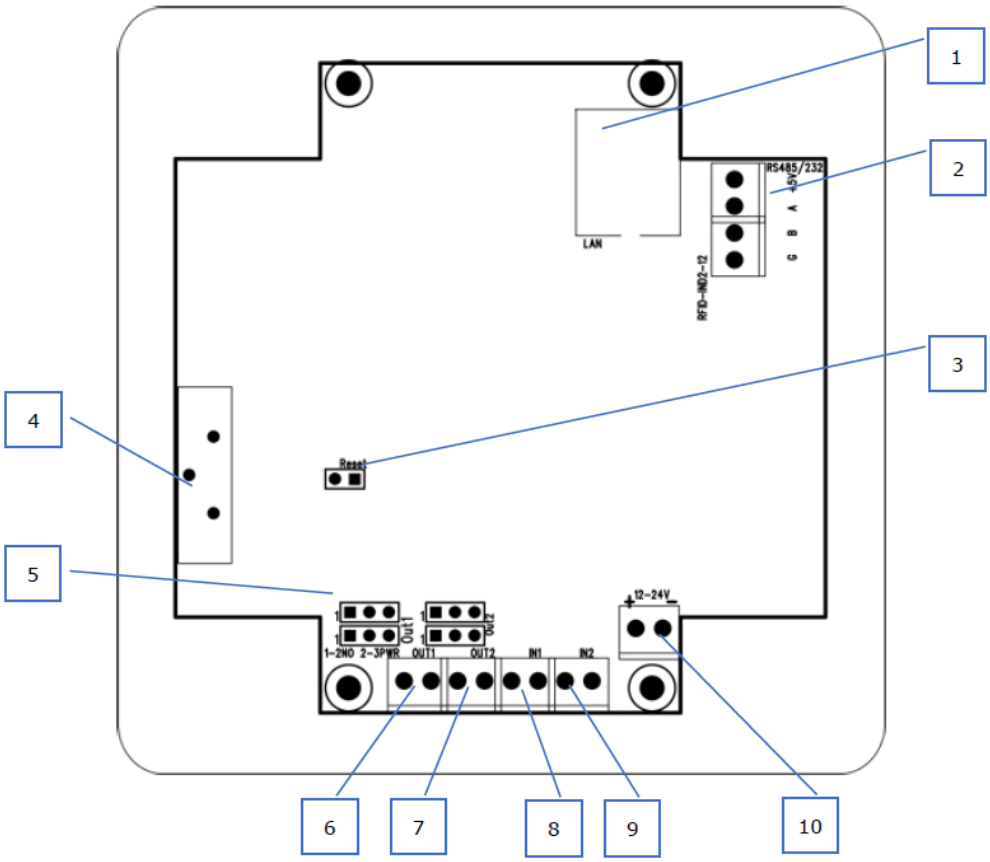
Because of different installation needs, the device does not have any mounting openings in the enclosure. Openings should be made by the customer anywhere in the rear section of the housing - this does not void warranty. Remember to use an appropriate cable gland in order to maintain the declared ingress protection.

Dimensions

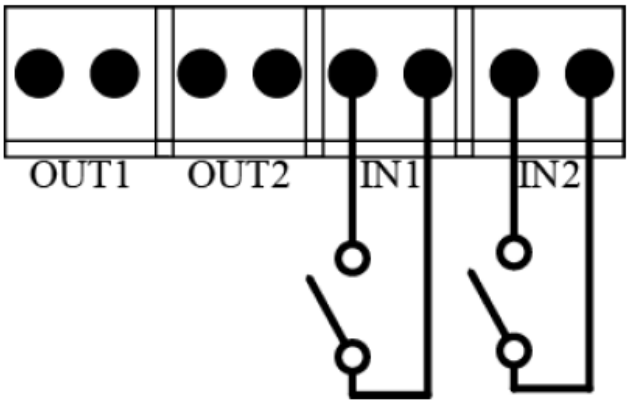




Connectors



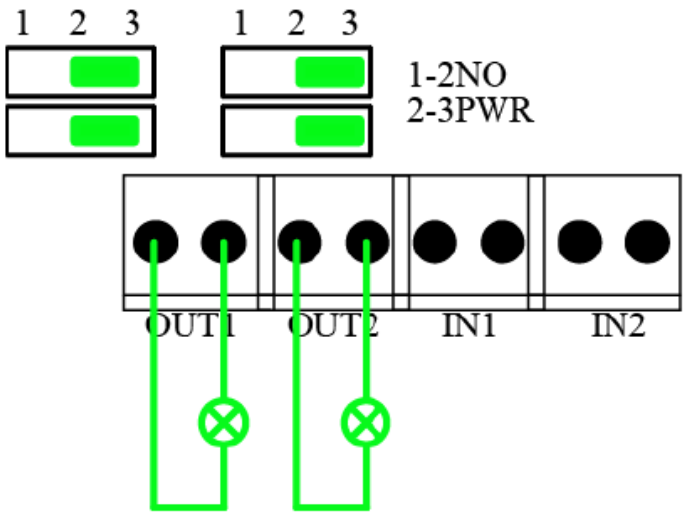
1. **LAN** – LAN connection socket,
2. **RS485** – RS485 Modbus RTU connector,
3. **RESET jumper** – Shorting the pins for 10-15 seconds will reset the module to factory settings,
4. **Battery holder**
5. **RelayMode** – Relay mode jumpers. In position 1-2 - NO relay output, position 2-3 - PWR output voltage is supplied from the power input,
6. **OUT1** – Relay 1 connector,
7. **OUT2** – Relay 2 connector,
8. **IN1** – General purpose input No. 1,
9. **IN2** – General purpose input No. 2,
10. **+12V-** – 10-24V DC module power supply input.


OUT1
OUT2
IN1
IN2

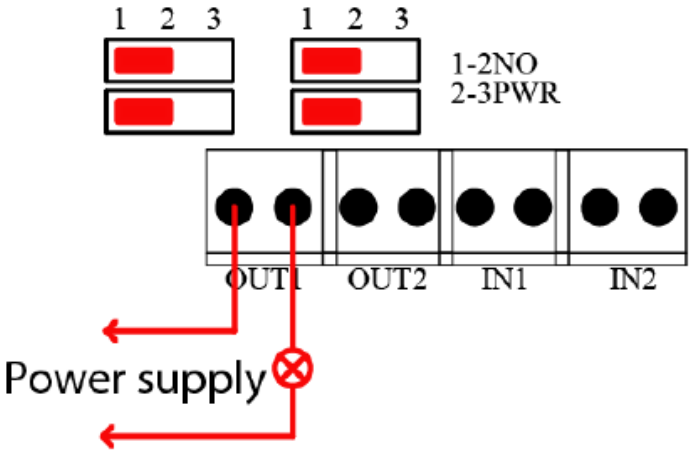
OUT1 and **OUT2** outputs can operate in two modes:

- **PWR mode** – Configuration jumpers set in position 2-3 (as shown in the image below). In this mode, an activated relay will provide the output with the power supply voltage eg. 12V DC. If the reader is powered with 24 VDC, 24V DC will be provided.

> **Warning**
>
> **If the reader is powered with PoE 802.3af and does not have a supply connected to the power input, outputs OUT1 and OUT2 won't have any voltage supplied to them.**



- **Normally Open mode** – Jumpers set in position 1-2 (as shown in the image below). In this mode an external power source is necessary.



General characteristics

IND2 LCD general view



IND2 LED general view



The module communicates via LAN or RS485.

The following methods can be used to access the read RFID tag:

- Built-in WWW server, available from a standard web browser,
- HTTP/HTTPS in server mode,
- HTTP/HTTPS in client mode,
- Modbus TCP,
- Modbus RTU (RS485 or optionally RS232),
- MQTT in client mode.

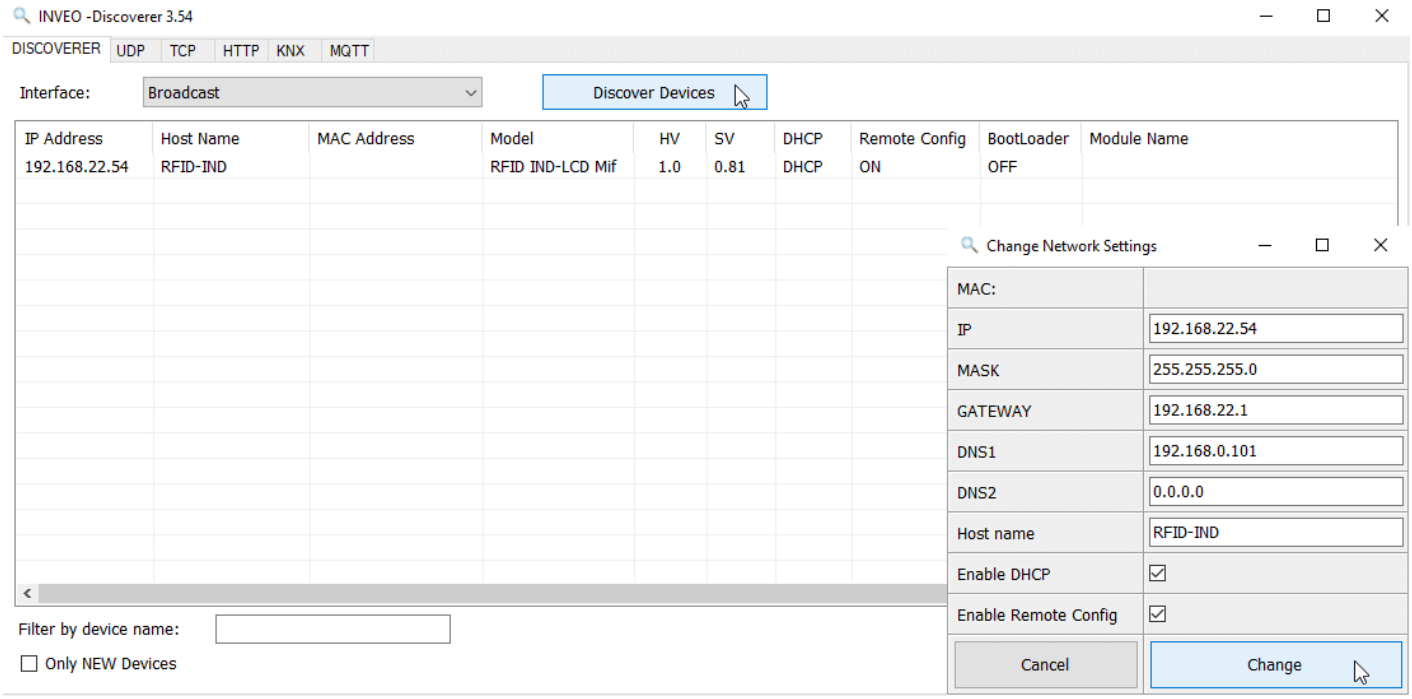
The module supports SSL encryption, HTTPS, certificate verification and e-mail messages.

The module is equipped with an LCD display or LED indicators, informing about power supply and current device state.

Network configuration

After the initial power-up, it's needed to configure the device. It can be done in two ways, the simplest method is to use the Inveo Discoverer programme.

Changing the device's IP address - Discoverer programme



After running the **Discoverer** programme and searching for the appropriate device:

1. Right-click the device,
2. Click the **Change settings** button.

After the dialogue window is opened you can:

- Set the IP address, mask, gateway, DNS1/DNS2,
- Change the Host name.

> **Note**
>
> **The device will be configured after the Change button is pressed.**

If **Remote Config** is disabled (enabled by default), the device needs to be configured by changing the computer's subnet:

To enable remote configuration:

1. Go to the **Administration** tab,
2. In the **Access configuration** window, select **Enable Remote Config**.

Enable Remote Network Config	☑	Enable Remote Network Config by Inveo Discover Software.

> **Tip**
>
> **After making changes, apply them using the Save button**

Changing the computer's subnet address for configuration

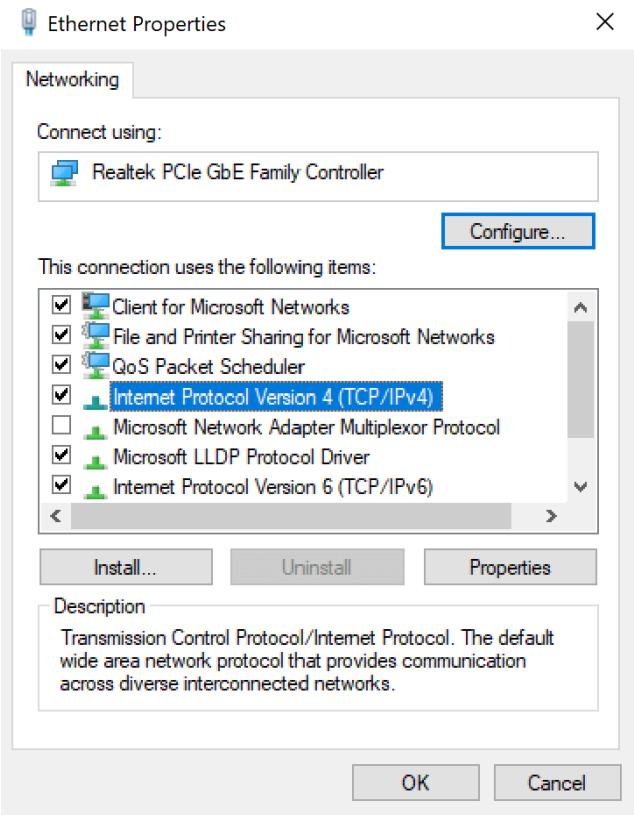
During configuration without **Discoverer** programme, it's needed to change the subnet address of the computer on the same network.

To access the computer's network configuration, do one of the following:

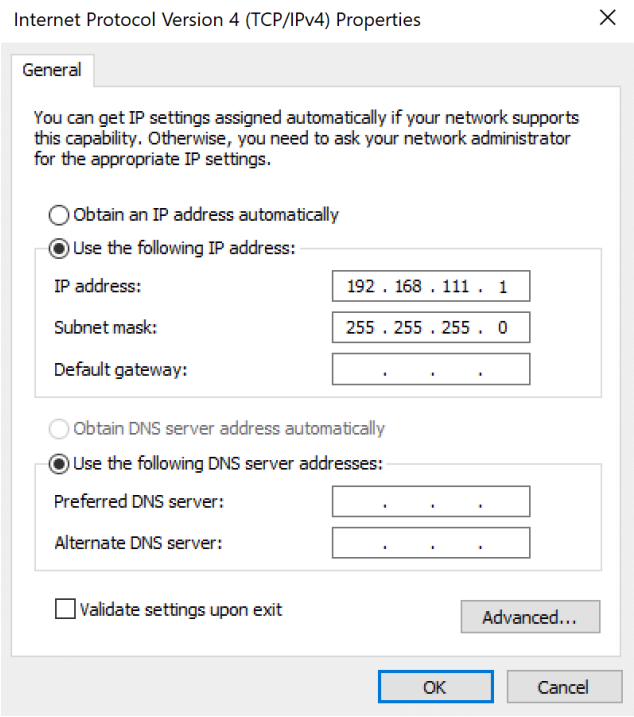
- Press **Win + R**, type in `ncpa.cpl`, and then press **Enter**,
- Alternatively, go to:
  Start → Control Panel → Network and Internet → Network and sharing center → Change adapter settings.

Then:

1. Select the network connection,
2. Right-click it and select **Properties**,
3. After selecting this option, network connection configuration screen will appear,



Select **Internet protocol version 4 (TCP/IPv4)** and type in the following parameters:

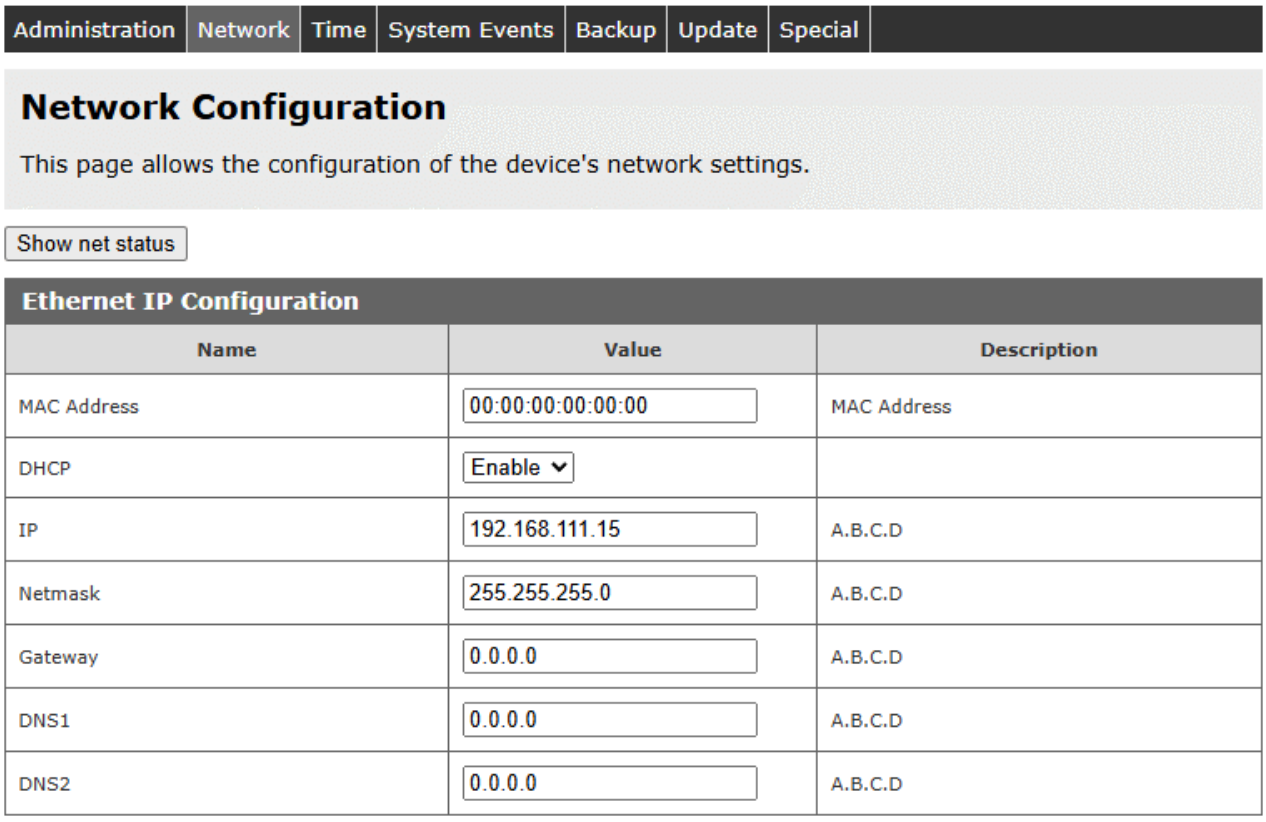


Note

After applying the configuration with OK, start a web browser and type in the following address: 192.168.111.15. (Default user and password: admin/admin)

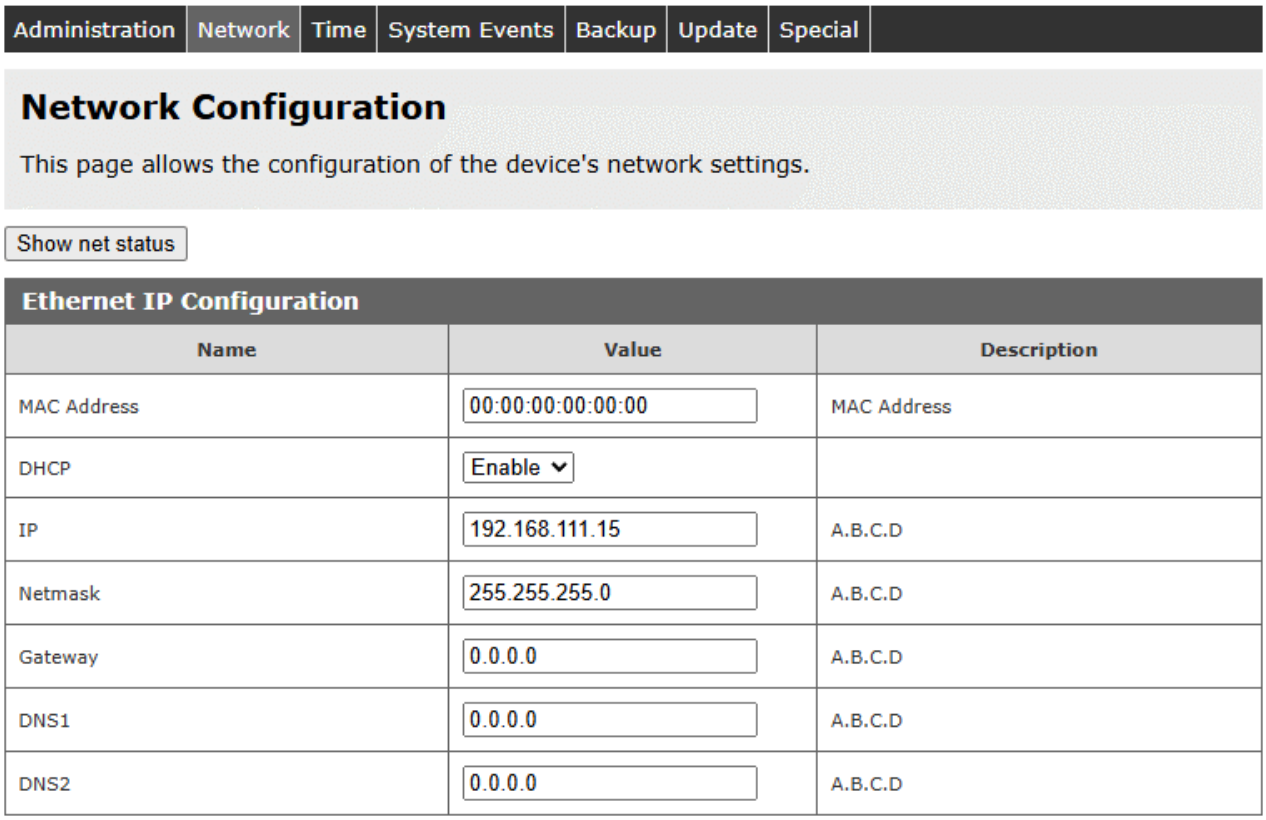
LAN and Wi-Fi configuration

In the **Administration / Network** tab, it's possible to change the LAN (and WLAN - if available) parameters.



- **MAC Address** – Device's MAC address,
- **DHCP** – Enable/disable the DHCP service,
- **IP** – Device's IP address,
- **Netmask** – IP Subnet mask,
- **Gateway** – Network gateway,
- **DNS1, DNS2** – DNS addresses.

The following additional wireless connection options are available in the device optionally equipped with Wi-Fi:



- **MAC Address** – Device's MAC address,
- **Enable** – Enable/disable Wi-Fi,
- **DHCP** – Enable/disable the DHCP service,
- **IP** – Device's IP address,
- **Netmask** – IP Subnet mask,
- **Gateway** – Network gateway,
- **DNS1, DNS2** – DNS addresses.
- **SSID** – Wireless network name,
- **Password** – Wireless network access password,
- **Encryption:**
  - **Open**,
  - **WEP**,
  - **WPA-PSK**,
  - **WPA2_PSK**,
  - **WPA_WPA2_PSK**,
  - **WPA3_PSK**.

After making changes, apply them using the **Save** button.

Checking the network status

The device allows for preview of current LAN and WLAN network parameters. In the **Administration / Network** tab, click **Show net status**

The **Interface eth** section contains the following information:

- **MAC** – Device's MAC address,
- **IP** – Current IP address,
- **Mask** – Network mask,
- **Gateway** – Network gateway,
- **DNS1, DNS2** – DNS server addresses,

The **Interface wlan** section contains the following WLAN (Wi-Fi) information:

- **MAC** – Device's MAC address,
- **IP** – Current IP address,
- **Mask** – Network mask,
- **Gateway** – Network gateway,
- **DNS1, DNS2** – DNS server addresses,
- **SSID** – Broadcast name of the WLAN, to which the device is connected,
- **RSSI** – Wi-Fi signal in dBm. Value of -110 indicates no signal.

Device configuration

Device status

The **Status** page allows the user to preview the current device parameters, such as the digital input, relay output status and RFID-related data: last read card UID and the amount of read tags since the last restart.

Status	
Name	**Value**
Input 1:	0 [cnt:0]
Input 2:	0 [cnt:0]
Relay 1:	0
Relay 2:	0
Last read ID:	
Number of read ID:	0

After the tab is selected, the **Status** table will appear:

- **Input 1** – Displays the current state of the first binary input,
- **Input 2** – Displays the current state of the second binary input,
- **Relay 1** – Informs about the current state of the relay output No. 1,
- **Relay 2** – Informs about the current state of the relay output No. 2,
- **Last read ID** – Displays the ID of last read RFID tag in hexadecimal format (HEX),
- **Number of read ID** – Displays the number of RFID tags read since last device restart.

Thanks to the built-in WWW server, the user can access the home page from anywhere within the local network, which simplifies management and device configuration.

Administration

The **Administration** tab in the device's user interface enables managing of key system settings, such as configuration of diagnostic services, time settings, network settings, firmware update, and changing the access password. The user also has the option to create a backup of current settings. The tab is divided into several sub-tabs, each offering specific functions.

Module name

In the **Settings** section, the user can assign a unique name to the device that can be used for identification purposes and easier network management.
The **Enable Remote Config** function enabling remote configuration has been described in the **Network configuration** chapter.

Settings

Name	Value	Description
Module name		
Enable remote config	Enable	Allow change configuration by Discoverer app

Changing the password

The user has the option to manage the safety of device access by changing the password. To do that:

1. In the **Current Password** field, input the current password,
2. In the **New Password** and **Re-type Password**, input the new password,
3. Click **Save** to save the changes.

Password

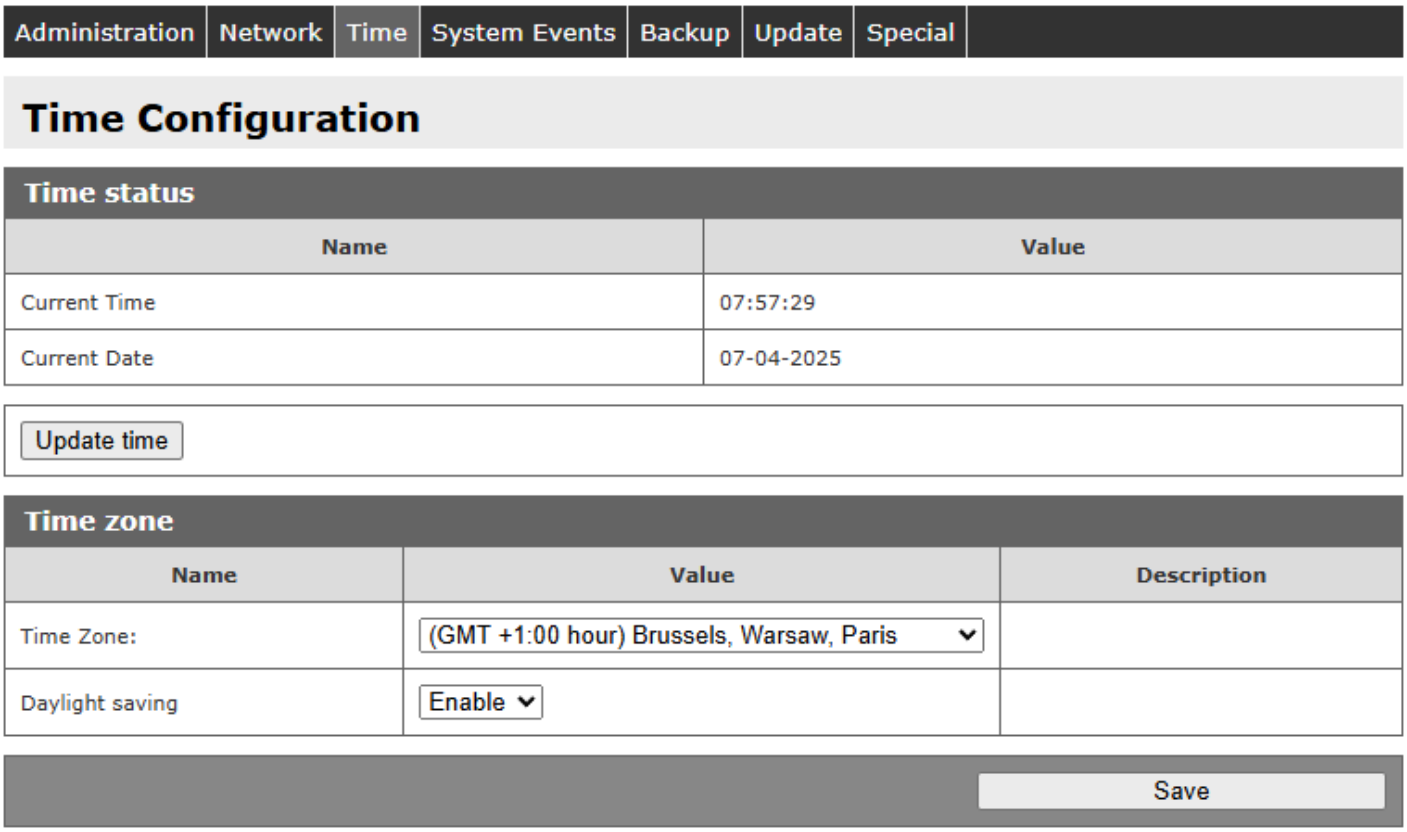
Name	Value	Description
Enable password	Enable	
Current password		
New password		
Retype new password		

Save

Time settings

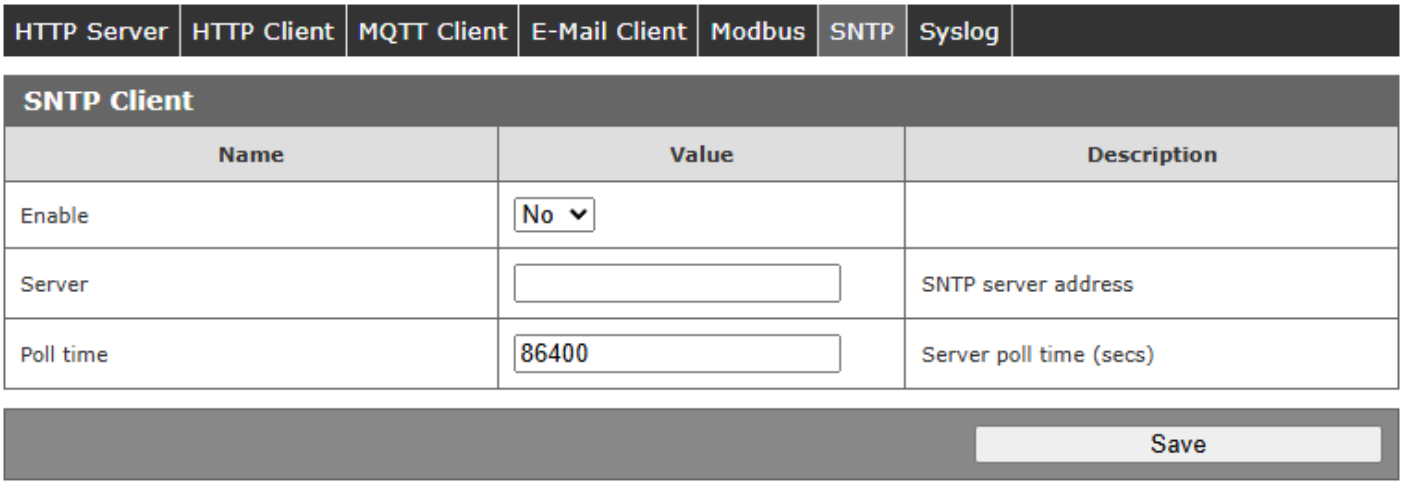
In the **Administration / Time** section, date and time settings are located. The device is fitted with a Real Time Clock with a battery backup, which allows for correct time measurement even during electricity loss.

The **Update Time** button synchronizes the device's internal clock with the current PC time. The time zone can be selected form an expandable list. Clicking **Daylight saving** will force the use of summer and winter time.



SNTP time server

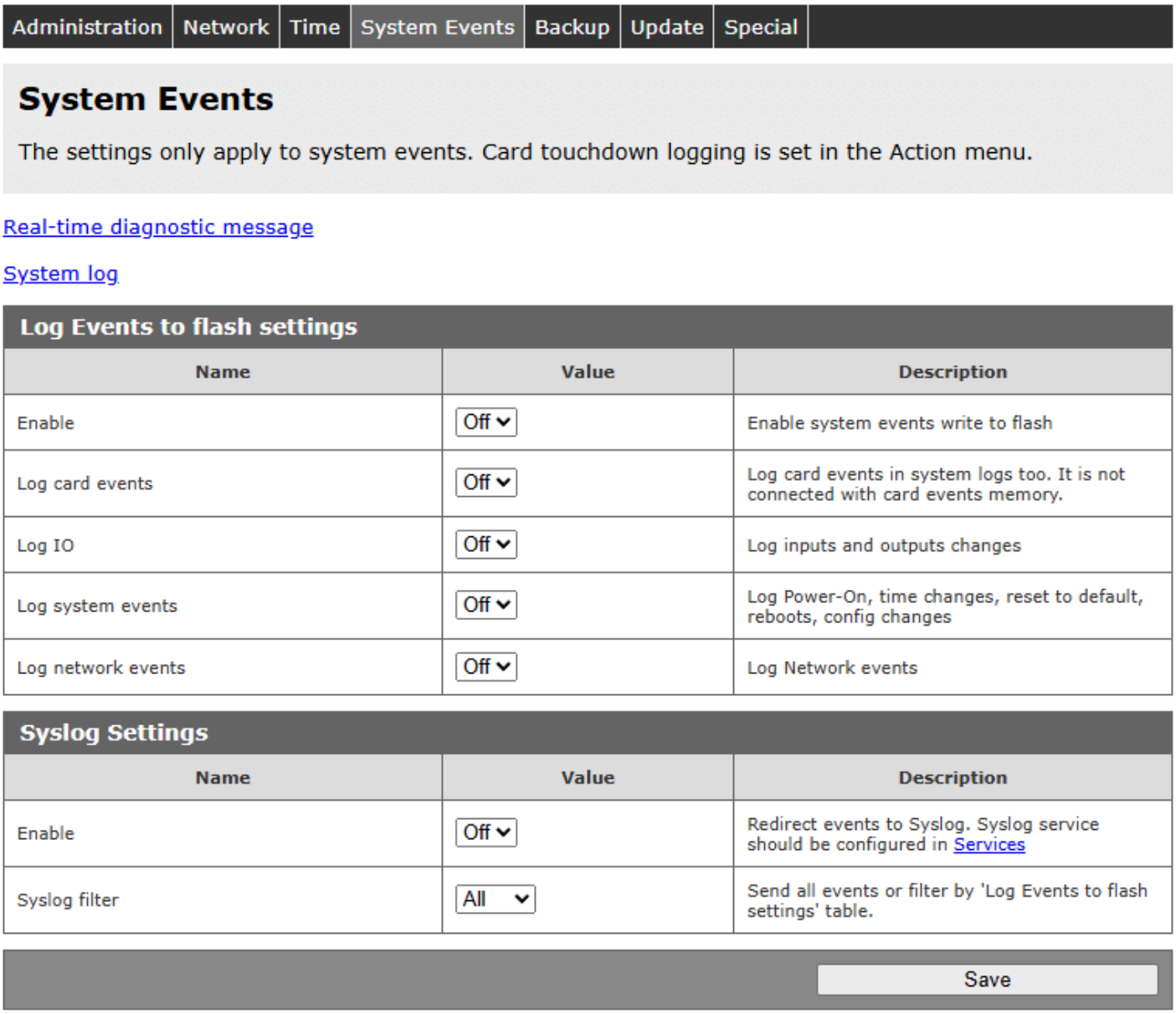
The device can act as an SNTP client. To configure the SNTP service, go to the **Services / SNTP** tab.



- **Enable** – Enable SNTP client,
- **Server** – IP address or domain name of the SNTP server,
- **Poll time** – Time server querying frequency.

Diagnostics, events

The module offers advanced diagnostic options, allowing for data logging and sending them to a Syslog server. The settings are available in the **Administration / System Events** tab.



Log Events to flash settings table allows the user to configure events, that are to be logged to flash memory.

- **Enable** – Enable events logging to flash memory,
- **Log cards events** – Log card events - independent of card memory,
- **Log IO** – Log input and output state changes,
- **Log system events** – Log system events eg. module shutdown, changes in time settings, reset, restart and configuration changes,
- **Log network events** – Log network events eg. establishing a Wi-Fi connection.

Logged data can be previewed by clicking **System log**. The **Realtime diagnostic message** link goes to the site with logs displayed in real-time.

Syslog

The device is able to forward data to a Syslog server. The user can choose what events will be sent. Configuration takes place in the **Syslog Settings** table located in the **Administration / Syslog** tab.

Syslog Settings		
Name	**Value**	**Description**
Enable	Off	Redirect events to Syslog. Syslog service should be configured in Services
Syslog filter	All	Send all events or filter by 'Log Events to flash settings' table.

- **Enable** – Enable syslog client,
- **Syslog filter:**
  - **All** – All events registered by the device are sent to the server,
  - **Filter** – Only events selected in **Log events to flash settings** table are sent to the server.

Special

The **Special** tab is used for inputting special configuration parameters per customer's individual needs.

I/O Settings

The I/O Settings tab enables the configuration of device input and output related settings. The reader is equipped with two relay outputs and two potential-free digital inputs. The user has the ability to adjust relay operations and input configuration in accordance to application needs.

Inputs

Input 1 Invert	Non invert	Invert input
Input 2 Invert	Non invert	Invert input
Input 1 Action	No Action	Input 1 action
Input 2 Action	No Action	Input 2 action
Input 1 Filter	0	Input transient filter [ms], set 0 to disable
Input 2 Filter	0	Input transient filter [ms], set 0 to disable

- **Input invert** – Software change of default input state,
- **Input action** – Defines the action that will be undertaken when the input is activated:
  - **No action**,
  - **Out 1** – Trigger the relay output No. 1,
  - **Out 2** – Trigger the relay output No. 2.

- **Input filter** - Allows defining a time (in ms) during which the device ignores short changes to input signal, treating them as interference. The value of 0 disables filtration, which introduces no delays.

Outputs

Inputs & outputs		
Name	**Value**	**Description**
Out 1 Mode	On Period	Output 1 action
Out 1 Time	4,0	Time [s]
Out 1 Time Long	10,0	Time [s]
Out 1 Invert	Non invert	Invert output

- **Out mode** – Relay output operation mode:
  - **None** – Software disabled relay control. The relay is inactive,
  - **Off** – Permanent relay deactivation,
  - **On** – Permanent relay activation,
  - **Toggle** – Change of relay state to the opposite (activation if it was deactivated, or deactivation if activated),
  - **On period** – Relay activation for a time defined in the **Out Time** field,
  - **On long period** – Relay activation for a time defined in the **Out Time Long** field.
- **Out time** – Time of relay activation for the **On Period** mode. The user can input a value in seconds, during which the relay will remain active,
- **Out time long** – Time of relay activation for the **On Long Period** mode. Sets the relay operation for a longer period of time,
- **Out invert** – Software change of relay state - default NO (Normally Open); selecting this option changes it to NC (Normally Closed).

The same settings apply to both inputs.

LCD display

For the reader fitted with an LCD screen, configuration of default text and card reaction is possible. Standby state text configuration is available in the **I/O Settings** tab, **LCD** section.

I/O Settings

In&Out | LCD

LCD

Name	Value	Description
Lcd Backlight	-1	0: Disable, -1: infinity, other: time in seconds
Lcd Default Text	%enter_inout% %enter_privwork%	
Show Clock	Yes	Show clock on first line.
Text Entrance	In	
Text Exit	Out	
Text Work	work	
Text Private	priv	

Save

- **LCD Backlight** – Screen backlight time, the value of 0 disables the backlight permanently, -1 activates the backlight permanently,
- **LCD Default Text** – Standby state display text, the user can use the available list of variables,
- **Show Clock** – Option to display a clock in the first line of the display.

The LCD screen can be configured to display the selected entry/exit mode *(RCP2 only)*.

- **Text Entrance**,
- **Text Exit**,
- **Text Work**,
- **Text Private**.

Function keys *(RCP2 only)*

The **Button** tab in the RCP2 with function keys is used to configure and adapt their operation to user's individual needs. The user can select several options that can affect the keypad operation and device communication.

Button

Name	Value	Description
Sound	Enable	Enable sound on press
Card Emulate	Enable	Treat entered code as a card

- **Sound** – Allows for keypad sound signal activation/deactivation,
- **Card Emulate** – Enables/disables the function to treat codes inputted with the kepyad as RFID cards.

Additional functions available when HTTP Client function is enabled:

HTTP Client method	GET	HTTP method
HTTP Content type	Default	HTTP Content type
HTTP Client request	btn=%btn%	

- **HTTP Client Method** – Allows selection of communication method: eg. GET, POST, and how the data will be transferred to an external server,
- **HTTP Content Type** – Allows definition of HTTP request content type, eg. JSON, XML, making integration with various systems easier,
- **HTTP Client Request** – Configuration of a HTTP request that will be sent after a key is entered. Thanks to this option, it is possible to, for example, send the code to an external server for verification or further processing.

Additional functions available when MQTT function is enabled:

MQTT	Disable	Publish MQTT
MQTT payload		

- **MQTT** – Enable/disable data publishing via MQTT,
- **MQTT Payload** – Configures the data contents transferred via MQTT. The user can customize the information that will be sent to MQTT brokers after a code is entered or other keypad actions are performed.

Predefined variables

The user has access to a set of predefined variables that can be used during LCD text and various functions configuration:

- `%usr_name1%` – `name` field from the card database (empty if no record),
- `%usr_name2%` – `surname` field from the card database (empty if no record),
- `%usr_id%` – ID from the card database (value `-1` if no record),
- `%uid%` – UID number,
- `%time%` – Current device time,
- `%date%` – Current device date,
- `%ts%` – Current device timestamp,
- `%mod_name%` – Device name,
- `%mod_model%` – Device model,
- `%eip%` – Ethernet - IP address,

- `%emac%` – Ethernet - MAC address,
- `%in1%` – State of input No. 1,
- `%in2%` – State of input No. 2,
- `%cnt1%` – Counter value for input No. 1,
- `%cnt2%` – Counter value for input No. 2,
- `%out1%` – State of output No. 1,
- `%out2%` – State of output No. 2,
- `%json_cardlogs%` – Log table,
- `%json_card%` – Last state of applied card,
- `%enter_state%` – *(RCP2 with function keys only)*,
- `%enter_inout%` – Function button: entry or exit *(RCP2 with function keys only)*,
- `%enter_privwork%` – Function button: privbate or official *(RCP2 with function keys only)*,
- `%btn%` – Last pressed button *(RCP2 with numeric keypad only)*:
  - digits **0 – 9** – Subsequent numerical keys,
  - **C** – ESC key,
  - **U** – Arrow pointing up,
  - **D** – Arrow pointing down,
  - **L** – Arrow pointing left,
  - **R** – Arrow pointing right,
  - **X** – ENTER key.

RFID reader

The tab can be used to configure RFID tag reading options, such as event logging and basic card functions.

Main settings

Name	Value	Description
Enable card log	Enable	Enable the card log. Note that each action should have the 'Log event' option set properly.
Action Hold time	5	The time (in seconds) of holding the card after which the 'Action hold card' will be performed

- **Enable card log** – Enable global card event logging. Different events that are to be logged should be configured individually,
- **Action hold time** – Time, after which the reader starts executing settings defined for **Action hold card** function.

Card data process		
Name	**Value**	**Description**
Swap id	No	Swap bytes of uid
Remove card timeout	10	The time (in 0.1s) of absence the card after which the 'Action remove card' will be performed
LUA script	No	Execute the user LUA script for each action. Go to LUA Editor

Mifare options >>

- **Swap id** – Allows changing of the byte order in RFID card's UID number,
- **Remove card timeout** – The time after taking the card from the reader's reading field, after which **Action remove card** will be executed (0.1s),
- **Lua script** – Enable Lua script response for each action *(option)*,
- **Lua Editor** – Allows access to the Lua script editor *(option)*,
- **Mifare Options** – Additional Mifare card readout options (only Mif version):
  - **Read Mifare block** – Enable block reading function,
  - **Block** – Read block address,
  - **Auth key A/B** – Authentication key selection (A or B),
  - **Key** – Authentication key in HEX format - the key is saved in secure memory (write only). The key should be written only when a change is required. Default key: 6xff (FFFFFFFFFFFF),
  - **Override UID** – Override card ID with read data,
  - **Override UID offset** – Character that the data readout begins from.

Card Action

The **Card Action** section allows for detailed configuration of RFID reader's reaction for RFID tag application. On this page the user can configure different settings such as:

- Controlling relay outputs,
- Configuration of HTTP requests,
- Defining the contents of MQTT messages,
- Event logging settings,
- Displaying announcements on the display and LED indicator reactions,
- Controlling the buzzer.

Action settings

On new card action		
Name	**Value**	**Description**
Search card in database	Yes	Does reader should compare the read card with internal database.
HTTP Client method	Disable	HTTP method
HTTP Content type	Default	HTTP Content type
HTTP Client request		
MQTT	Disable	Publish MQTT
MQTT payload		

The **On new card action** section specifies the reader's behaviour when a new RFID card is detected in the read area. Some settings will only be visible if **HTTP Client** and/or **MQTT** is enabled in **Services**.

- **Search card in database** – Option disables the function of comparing the applied tag to an internal database and treats all tags as unknown,
- **HTTP Client Method, HTTP Content type oraz HTTP Client request** – See **HTTP Client configuration** and **HTTP Client – card action event request**,
- **MQTT oraz MQTT payload** – See **MQTT Client**.

Action on server requests		
Name	**Value**	**Description**
Wait for server reply	No	'No' - Perform a Known action or an Unknown action depending on recognized card. 'Yes' - do below action then wait for server reply. Server should control I/O.

- **Wait for server reply:**
  - **No** – The device checks the card, and depending on the **Search card in database** goes to **Action known card** or **Action unknown card** tabs,
  - **Yes** – When using the HTTP Client, the device waits for data from the server for up to 3 seconds. The user can set the wait signaling eg. an LCD text. After this function is enabled, additional settings will appear. In some situations, the HTTP server can have a delay in responding to the reader.

Action on server requests		
Name	**Value**	**Description**
Wait for server reply	Yes	'No' - Perform a Known action or an Unknown action depending on recognized card. 'Yes' - do below action then wait for server reply. Server should control I/O.
Action by HTTP status code	Disable	Perform a Known action or an Unknown action depending on the HTTP response status code
HTTP Connection fail action	Disable	If connection fail - perform a Known action or an Unknown action depending on 'Search card in database' settings.
LCD text		
Sound	Tick	Sound action
Out 1	None	Output 1 action
Hide more options		
Out User Time 1	40	Output 1 user time period
Out 2	None	Output 2 action
Out User Time 2	0	Output 2 user time period
LCD backlight	None	LCD Backlight
LCD Text Timeout	4	0: Disable, -1: infinity, other: time in seconds
Log event	Enable	Log event to internal memory

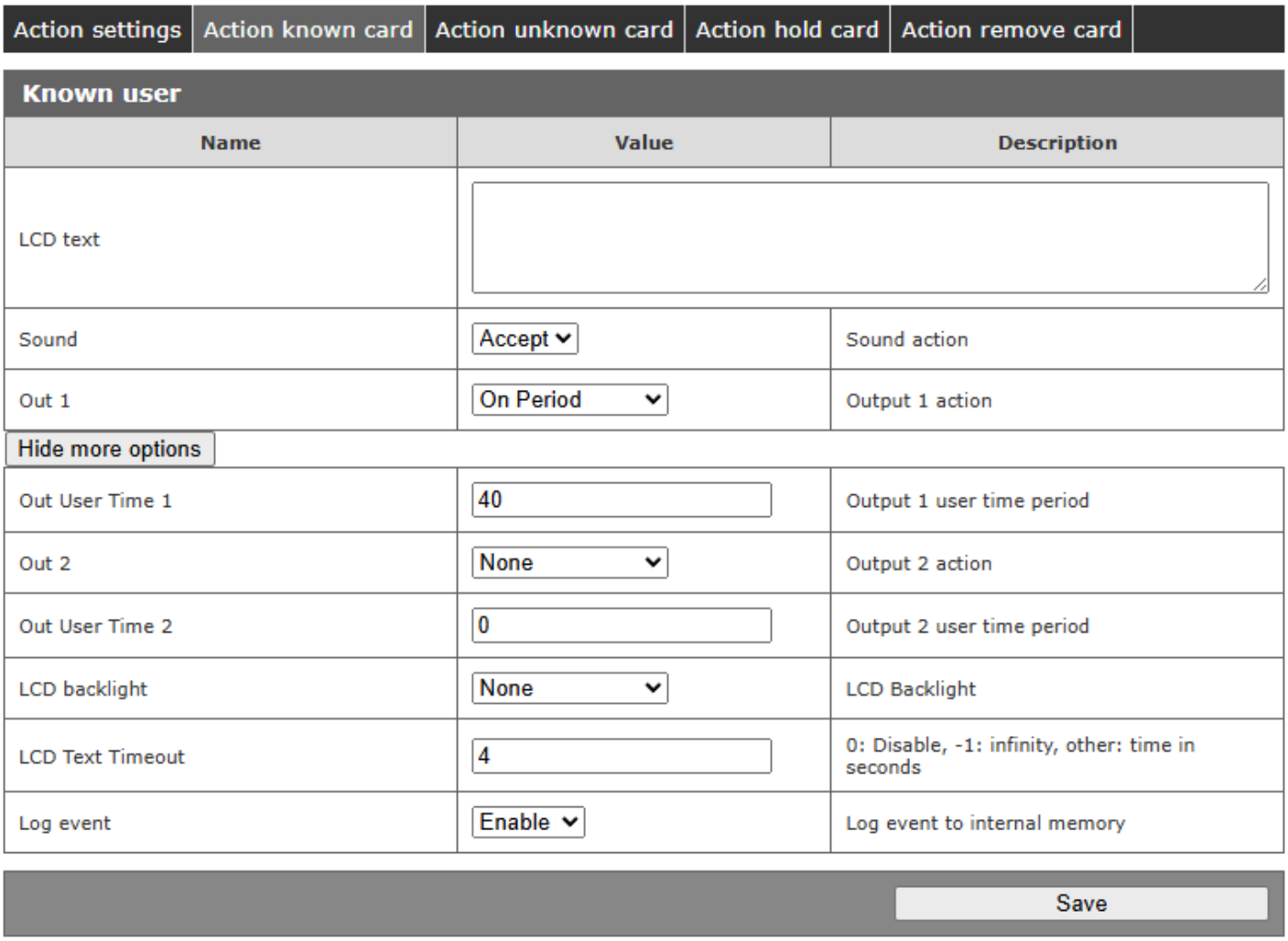
The **Action on server requests** section allows the user to define the reader's behaviour while waiting for a reply from server or while receiving it:

- **Wait for server reply**,
- **Led Green/Red** – Green/Red LED indicator *(IND2 LED only)*:
  - **None**,
  - **Off**,
  - **On**,
  - **Toggle** – Invert the LED state,
  - **On Period** – Activate the LED for the **Time** period,
  - **On Long Period** – Activate the LED for the **Time Long** period,
  - **Blink** – LED blink.
- **Sound** – Sound signal:
  - **None**,

    - **Accept**,
    - **Reject**,
    - **Tick** – Short sound signal.
- **Out 1/2** – Output 1/2 control:
    - **None** – Skip,
    - **Off** – Deactivate the relay output,
    - **On** – Activate the relay output,
    - **Toggle** – Change relay state to the opposite,
    - **On period** – Relay activation for a time defined in the **Out Time** field,
    - **On long period** – Relay activation for a time defined in the **Out Time Long** field,
    - **On User period** – Relay activation for a time defined in the **Out User Time 1** field.
- **Out User Time 1/2** – Relay No. 1 activation time for **On User Period** mode,
- **LCD text** – Message displayed when the card is applied,
- **LCD backlight:**
    - **None** – Skip,
    - **Off**,
    - **On**,
    - **Toggle** – Invert state,
    - **On Period** – Backlight activation for a time specified in the **Lcd Backlight** (**IO Settings** tab),
    - **On Long Period** – Backlight activation for a time specified in the **Out Time Long** field (**IO Settings** tab),
    - **Blink** – Backligh blink.
- **LCD Text Timeout:**
    - **0** – Do not display text,
    - **-1** – Display text continuously until another message is recalled,
    - **other value** – Text display time in seconds.
- **Log event** – Enable event logging for RFID tag application in protocol control mode.
- **Action by HTTP status code** – Automatic response depending on the HTTP response code received:
    - **Code 200:** the reader will execute the settings in **Action known card**,
    - **Other codes:** the reader will execute the settings in **Action unknown card**.
- **HTTP Connection fail action** – Reader waits 3 seconds for the server response:
    - **Disable** – If the sever does not respond, do nothing,
    - **Enable** – If the server does not respond, go to the **Action known card** or **Action unknown card** tab, depending on the **Search card in database** setting.

Action known card

This tab allows defining reactions of the reader after a card stored in the device's memory is applied.



- **Led Green/Red** – Green/Red LED indicator *(IND2 LED only)*:
  - **None**,
  - **Off**,
  - **On**,
  - **Toggle** – Invert the LED state,
  - **On Period** – Activate the LED for the **Time** period,
  - **On Long Period** – Activate the LED for the **Time Long** period,
  - **Blink** – LED blink.
- **LCD text** – Message displayed when the card is recognized,
- **Sound** – Sound signal:
  - **None**,
  - **Accept**,
  - **Reject**,
  - **Tick** – Short sound signal.
- **Out 1/2** – Output 1/2 control:
  - **None** – Skip,
  - **Off** – Deactivate the relay output,

    - **On** – Activate the relay output,
    - **Toggle** – Change relay state to the opposite,
    - **On period** – Relay activation for a time defined in the **Out Time** field,
    - **On long period** – Relay activation for a time defined in the **Out Time Long** field,
    - **On User period** – Relay activation for a time defined in the **Out User Time 1** field.
- **Out User Time 1/2** – Relay No. 1 activation time for **On User Period** mode,
- **LCD backlight:**
    - **None** – Skip,
    - **Off**,
    - **On**,
    - **Toggle** – Invert state,
    - **On Period** – Backlight activation for a time specified in the **Lcd Backlight** (**IO Settings** tab),
    - **On Long Period** – Backlight activation for a time specified in the **Out Time Long** field (**IO Settings** tab),
    - **Blink** – Backligh blink.
- **LCD Text Timeout:**
    - **0** – Do not display text,
    - **-1** – Display text continuously until another message is recalled,
    - **other value** – Text display time in seconds.
- **Log event** – Enable event logging for known RFID tag application in protocol control mode.

Action unknown card

This tab allows defining reactions of the reader after a card that isn't saved or has been deactivated in the device's memory is applied.

Action settings | Action known card | Action unknown card | Action hold card | Action remove card

Unknown user

Name	Value	Description
LCD text		
Sound	Tick	Sound action
Out 1	None	Output 1 action

Hide more options

Out User Time 1	40	Output 1 user time period
Out 2	None	Output 2 action
Out User Time 2	0	Output 2 user time period
LCD backlight	None	LCD Backlight
LCD Text Timeout	4	0: Disable, -1: infinity, other: time in seconds
Log event	Enable	Log event to internal memory

Save

Same settings as in **Action known card** are available.

Action hold card

This tab allows defining reactions of the reader for holding a card in the read area for time defined in the **Action Hold time** parameter (default 5 seconds). This parameter can be configured in the **Card Action / Action settings** section in the **Main settings** tab.

Action settings | Action known card | Action unknown card | Action hold card | Action remove card

Hold card

Name	Value	Description
LCD text		
Sound	None	Sound action
Out 1	On	Output 1 action

Hide more options

Out User Time 1	40	Output 1 user time period
Out 2	None	Output 2 action
Out User Time 2	0	Output 2 user time period
LCD backlight	None	LCD Backlight
LCD Text Timeout	4	0: Disable, -1: infinity, other: time in seconds
HTTP Client method	Disable	HTTP method
HTTP Content type	Default	HTTP Content type
HTTP Client request		
MQTT	Disable	Publish MQTT
MQTT payload		
Log event	Disable	Log event to internal memory

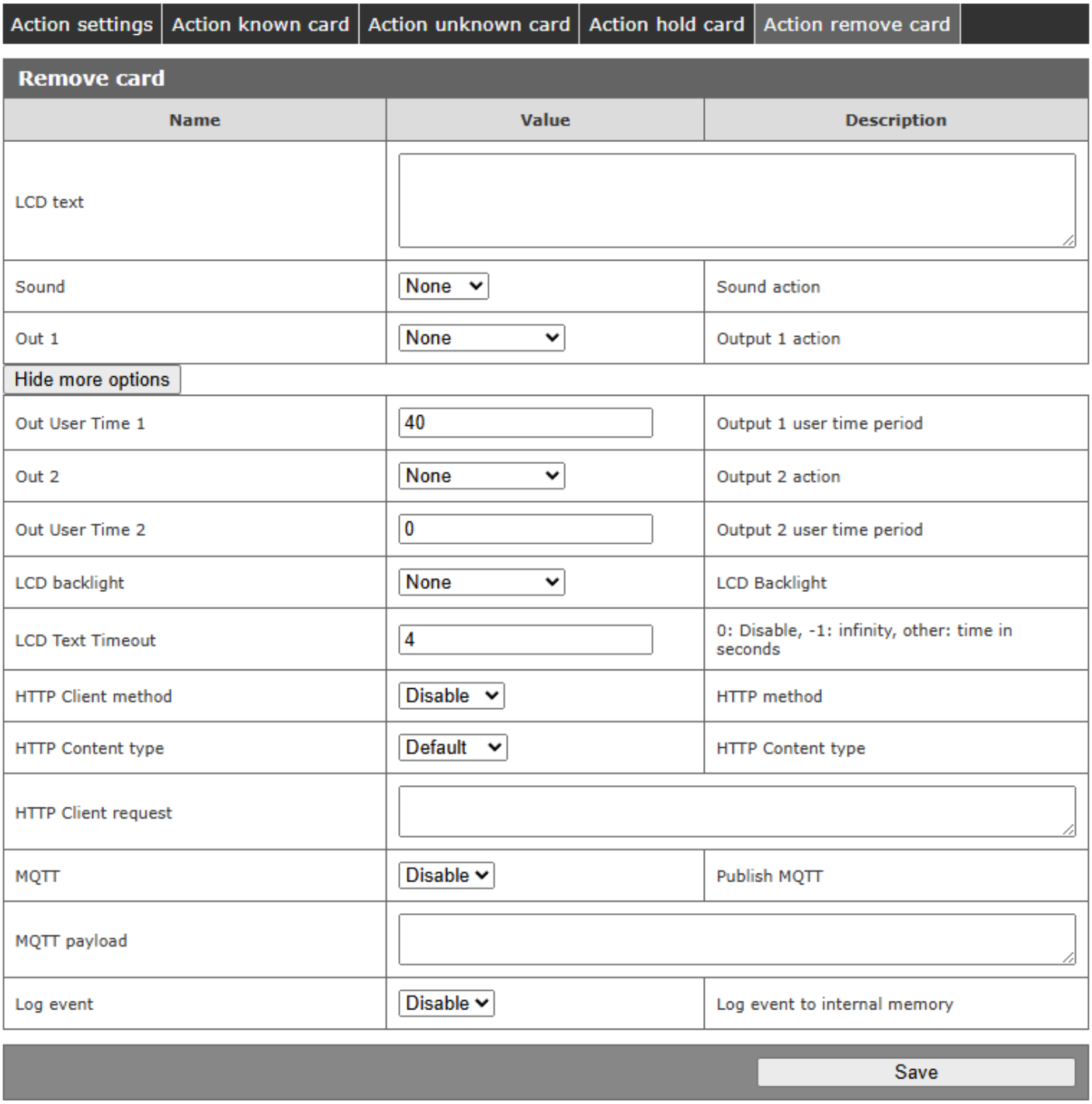
Save

Same settings as in **Action known card** are available. Additionally:

- **HTTP Client Method, HTTP Content type oraz HTTP Client request** – See **HTTP Client – card action event request**,
- **MQTT oraz MQTT payload** – See **MQTT Client**,
- **Log event** – Enable event logging if an RFID tag is held in range of the reader.

Action remove card

This tab allows defining reactions of the reader the moment a card is removed from the read area.



Same settings as in **Action hold card** are available.

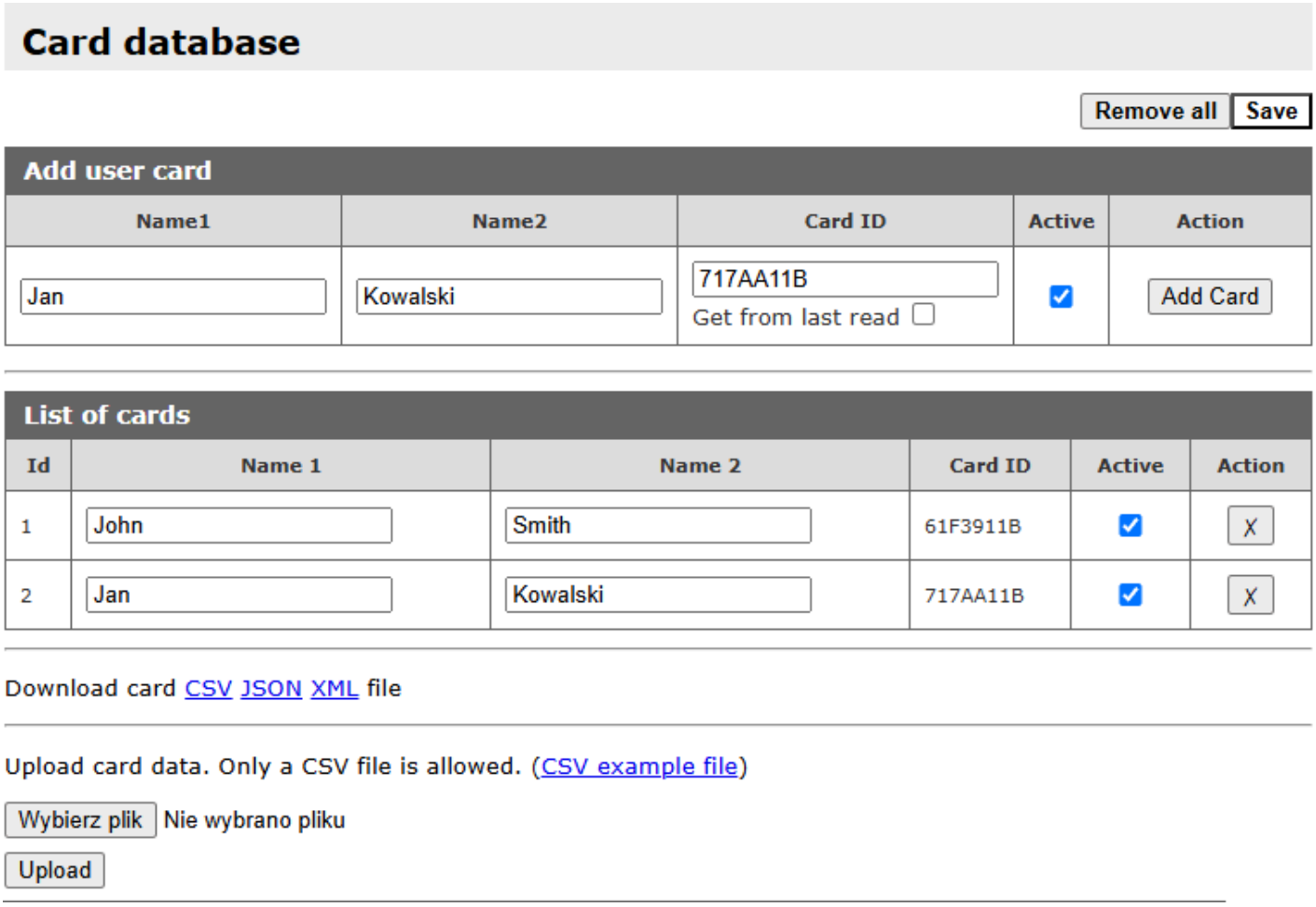
Managing cards and events

Card database

The RFID reader is equipped with internal memory for managing RFID cards. The user can add cards to the reader both manually and by importing a CSV file. Device's reaction may be different depending on whether a card is in the reader's memory or not.

Managing tags from a browser

The **Card Database** tab in the user interface allows for manual adding, editing and erasing RFID tags from the device's memory.



Fill the **Add user card** table to add a new user:

- **Name 1** – User's first name,
- **Name 2** – User's last name,
- **Card ID** – Read RFID tag UID number,
- **Get from last read** – Function inputting last read RFID tag,
- **Active** – Enable the RFID tag in reader's memory - disabling this option will cause the tag to skip searching the memory.
- **Add card** – Add record to a temporary RFID tag list.

Tip

The Add Card button causes a creation of a temporary RFID tags list, which is not immediately saved to the non-volatile memory. If a tag has been added or a record amended, the Save button changes its colour to red, indicating the need to press it in order to save all data to the non-volatile memory.

CSV upload

To upload an RFID card database to the device, make sure that the CSV file structure matches the **CSV Example File**. Next, upload the desired file to the device by using **Choose file** and clicking **Upload**.

```
uid;name;surname;userId;flag
12345678;John;Smith;1;1
87654321;Saul;Smith;2;1
```

The first row contains a header (ignored during file upload):

- **uid** – RFID tag ID,
- **name** – Name1 field from the card database,
- **surname** – Name2 field from the card database,
- **userId** – Unique user ID,
- **flag** – Value of the **active** field from the card database.

Warning

Size limit: maximum size of the uploaded file is 200 kB, which equals roughly 5000 users.

Downloading the database

The internal card database can be downloaded in the form of CSV, JSON and XML files.

To gain access to those files, refer to the corresponding resources (e.g. `192.168.111.15/card.csv`):

CSV file – `card.csv`

```
uid;name;surname;userId;flag
12345678;John;Smith;1;1
87654321;Saul;Smith;2;1
```

JSON file – `card.json`

```
{
    "card":[
        {
            "uid":"12345678",
            "name1":"John",
            "name2":"Smith",
            "userId":"1",
            "flag":"1"
        },
        {
            "uid":"87654321",
            "name1":"Saul",
            "name2":"Smith",
            "userId":"2",
            "flag":"1"
        }
    ]
}
```

XML file – `card.xml`

```
<card>
    <entry>
        <uid>12345678</uid>
        <name1>John</name1>
        <name2>Smith</name2>
        <userId>1</userId>
        <flag>1</flag>
    </entry>
    <entry>
        <uid>87654321</uid>
        <name1>Saul</name1>
        <name2>Smith</name2>
        <userId>2</userId>
        <flag>1</flag>
    </entry>
</card>
```

- **uid** – RFID tag ID,
- **name1** – name1 field from the card database,
- **name2** – name2 field from the card database,
- **userId** – Unique user ID,
- **flag** – Value of active field from the card database.

Event logging

The device can register events related to applying, holding or removing the card from the reader read area. Event log is available:

- On the website,

- In CSV format,
- In XML format,
- In JSON format.

Activation of event logging is possible in the **RFID reader** page, **Setup** tab, **Main settings** table.

Main settings		
Name	**Value**	**Description**
Enable card log	Disable	Enable the card log. Note that each action should have the 'Log event' option set properly.
Action Hold time	5	The time (in seconds) of holding the card after which the 'Action hold card' will be performed

- **Enable card log** – Activates the global event logging function. The events that are to be saved to the device's memory should be specified,
- **Action hold time** – The time after which the reader will start executing the actions assigned to the **Action hold card** function.

Types of logged events

- **Action Known Card** – Application of an RFID card stored in the reader's memory,
- **Action Unknown Card** – Application of an RFID card not stored or deactivated in the reader's memory,
- **Action Hold Card** – Holding an RFID card in the read area for minimum **Action Hold Time**,
- **Action Remove Card** – Removal of RFID card outside the read area,
- **HTTP/MQTT Communication** – Utilizing and control by HTTP or MQTT communication protocols, optionally after enabling **Wait for Server Replay**.

Logging activaton/deactivation: To activate or deactivate event logging, go to the appropriate tab in **Card Action** and click **Show More Options**. The **Log Event** row allows the user to enable or disable logging of the selected event.

Log event	Disable	Log event to internal memory

Events preview

The event log is available on the **Card logs** page. The logs available via the webpage have the following format:

Card Logs

Download log: CSV XML JSON

Remove logs

```
3 [2025-04-07 08:49:29]  action:cardKnown  mode:1030  userId:2  uid:717AA11B  name:Jan Kowalski
2 [2025-04-07 08:49:23]  action:cardKnown  mode:1030  userId:2  uid:717AA11B  name:Jan Kowalski
1 [2025-04-07 08:49:20]  action:cardKnown  mode:1030  userId:1  uid:61F3911B  name:John Smith
```

Column	Description
1	Sequential number
2	Date and time of recorded event
3	Action flag: - 1 – RFID tag read while in protocol control mode (HTTP/...) - 2 – action for a card stored in the device's memory - 4 – action for an unknown/inactive card - 8 – action for the hold card function - 16 – action for the remove card function
4	Mode value – decimal sum of events - consult the table below
5	ID number of user in card memory. „-1” means a card is unknown or inactive
6	UID of the read RFID tag
7	Combined value of Name and Surname fields from card memory

Mode value is the sum of a value assigned to the current input / output state:

Input/Output state	Description	binary	dec	hex
LOG_OUT1	Current state of output 1	0100 0000 0000	1024	400
LOG_OUT2	Current state of output 2	1000 0000 0000	2048	800
LOG_IN1	Current state of input 1	01 0000 0000 0000	4096	1000
LOG_IN2	Current state of input 2	10 0000 0000 0000	8192	2000

Function button	Description	dec	hex
ENTER_FLAG_PRIVATE	private	1	1
ENTER_FLAG_WORK	official	2	2
ENTER_FLAG_IN	entry	4	4
ENTER_FLAG_OUT	exit	8	8
ENTER_FLAG_EXT	Application of card to the external reader	16	10

Logs can be deleted by pressing **Remove logs** or with an HTTP command.

Log download

Logs can be downloaded in the form of CSV, XML or JSON files.

CSV file – `log.csv`

```
timestamp;time;date;action;user_id;uid;name1;name2;enter
1744008560;08:49:20;07.04.2025;2;1;61F3911B;John;Smith;1030
1744008563;08:49:23;07.04.2025;2;2;717AA11B;Jan;Kowalski;1030
1744008569;08:49:29;07.04.2025;2;2;717AA11B;Jan;Kowalski;1030
```

XML file – `log.xml`

```xml
<logs>
        <entry>
        <action>2</action>
        <timestamp>1744008560</timestamp>
        <time>08:49:20</time>
        <date>2025-04-07</date>
        <userId>1</userId>
        <uid>61F3911B</uid>
        <name1>John</name1>
        <name2>Smith</name2>
        <mode>1030</mode>
    </entry>
    <entry>
        <action>2</action>
        <timestamp>1744008563</timestamp>
        <time>08:49:23</time>
        <date>2025-04-07</date>
        <userId>2</userId>
        <uid>717AA11B</uid>
        <name1>Jan</name1>
        <name2>Kowalski</name2>
        <mode>1030</mode>
    </entry>
</logs>
```

JSON file – `log.json`


```
{
    "logs":[
        {
            "action":2,
            "mode":1030,
            "ts":1744008560,
            "time":"08:49:20",
            "date":"2025-04-07",
            "userId":1,
            "uid":"61F3911B",
            "name1":"John",
            "name2":"Smith"
        },
        {
            "action":2,
            "mode":1030,
            "ts":1744008563,
            "time":"08:49:23",
            "date":"2025-04-07",
            "userId":2,
            "uid":"717AA11B",
            "name1":"Jan",
            "name2":"Kowalski"
        }
    ]
}
```


Card log transmission

It is possible to activate periodic sending of card logs via HTTP(S). After this option is enabled, the logs – if present – will be sent in specified time intervals.
The following settings should be configured in **Services / HTTP Client**:

- **Send card log by HTTP(S)** – Activation / deactivation of periodic log sending via HTTP(S), data is sent in packets of up to 4kB in a single request,
- **Send card log period** – Frequency of logs sending in minutes,
- **Send card log request** – Syntax of the request sent by the device according to **Send card log period**. Must contain the variable `%json_cardlogs%` that recalls a table containing event logs.

If Internet access is unavailable, the device will send the saved events in 4kB packets (around 26 events) after the connection is restored.

Example

Send card log by HTTP(S)	Enable	Enable sending logs to server. Method POST, content-type JSON
Send card log period	1	The period (in minutes) to check and send logs
Send card log request	{"device":"%mod_model%","logs":%json_cardlogs%}	

According to the settings shown above, the device will send the card logs every minute (if they are present).

Sample data packet

```
{
    "device":"RFID reader ABCD",
    "logs":[
        {
            "action":"16",
            "mode":"0",
            "ts":"1682334001",
            "time":"13:00:01",
            "date":"2023-04-24",
            "userId":"6",
            "uid":"AA81DD9B",
            "name1":"Jack",
            "name2":"Smith"
        },
        {
            "action":"2",
            "mode":"0",
            "ts":"1682333999",
            "time":"12:59:59",
            "date":"2023-04-24",
            "userId":"4",
            "uid":"AB94DC5A",
            "name1":"John",
            "name2":"Smith"
        }
    ]
}
```

After each received log packet, the server will return the request defined by the user – example response:

```
{
"lcd_clr":0,
"lcd_text0":"Log packet transferred",
"cardlog_remove":12
}
```

After the reader receives this response, `Log packet transferred` will appear on the LCD. `"cardlog_remove":12` causes the 12 oldest logs to be removed from the device's memory. If any data remains in the memory, it will be sent in the next packet automatically. Data transfer will continue until no logs remain in the memory or the received command does not contain `cardlog_remove` (in this case the reader will send the next log packet after the **Send card log period** time passes).

Communication protocols

The device supports various communication protocols that can be adapted to individual needs. The configuration is available in the **Services** tab.

HTTP/HTTPS Server

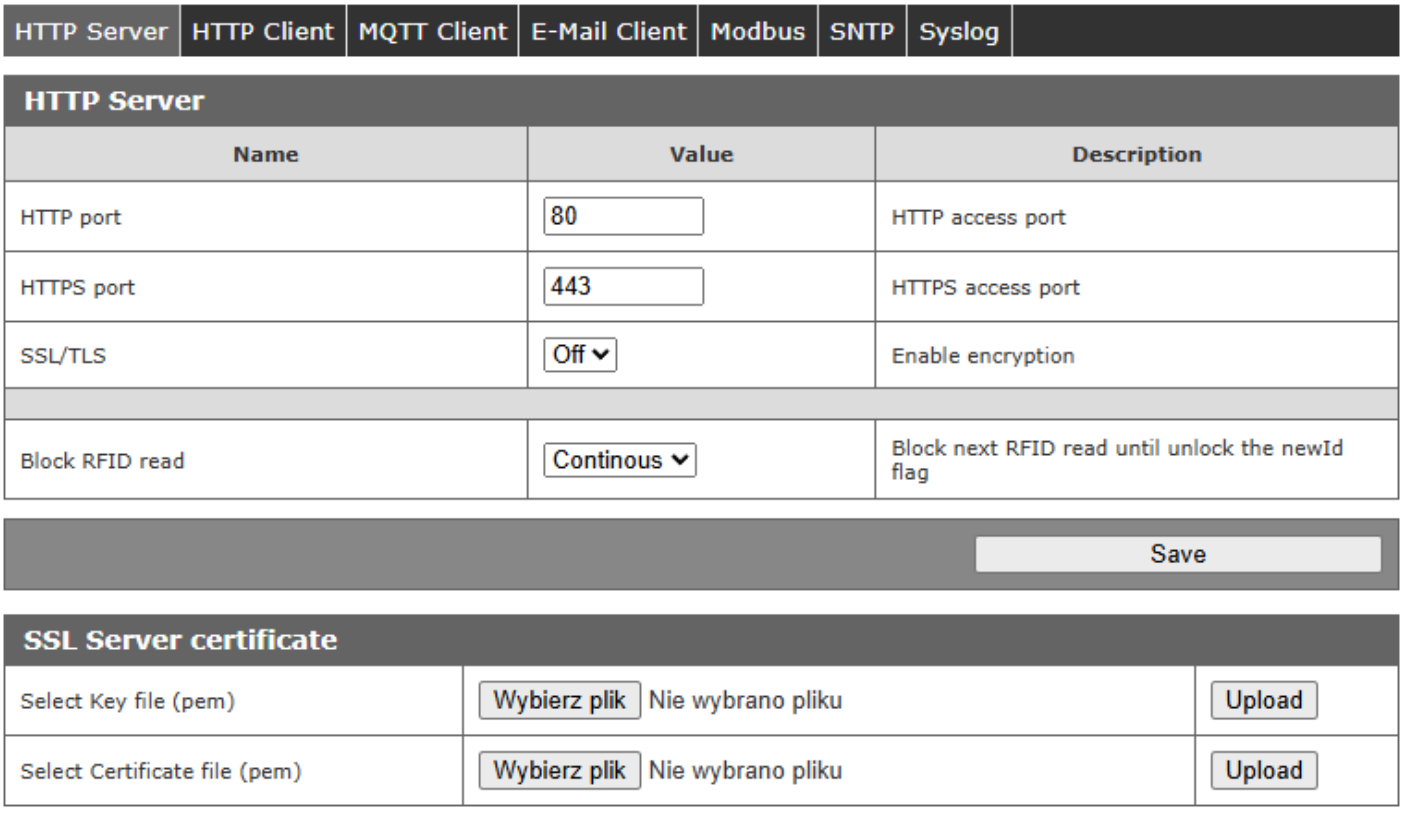
The reader can be configured as an HTTP or HTTPS server. In this mode, an external client connects with the reader and manages it with HTTP/HTTPS requests (GET method).
The client regularly downloads the `status.xml` or `status.json` file, which contains information about device condition such as:

- **Card UID**,
- **Reader's MAC address**,
- **Input and output states**.

After reading and decoding this data, the client can send commands to the reader, that will recall appropriate reactions, such as:

- **Sound signalling** (eg. card accepted/rejected),
- **LED indicators flashing**,
- **Door lock release**,
- **Displaying text on the screen**.

HTTP server configuration is available in the **Services** tab.



- **HTTP port** – HTTP server and device webpage access port,
- **HTTPS port** – HTTP server and device webpage access port (SSL/TLS encryption required),
- **Block RFID read** – Readout of a subsequent RFID tag is blocked until the reset flag is reset,
- **SSL/TLS** – Enable encrypted HTTPS connection support,
- **SSL Key file (pem)** – SSL key file location (pem format),
- **Certificate file (pem)** – CSR (Certificate Signing Request) file location (pem format).

HTTP GET – Status

To get the current module status, recall the resource under the address: `<IP_address>/data/status.xml` or `<IP_address>/data/status.json`.

In response, the resource will be displayed in the form of an XML or JSON file, containing basic information about the reader:

Example response in XML format

```
<status>
    <model>RFID IND2 LED Mif</model>
    <name/>
    <mac>00:00:00:00:00:00</mac>
    <id>40005EB4F4</id>
    <newId>1</newId>
    <rfid_enable>1</rfid_enable>
    <cnt>1</cnt>
    <n_logs>1</n_logs>
    <in1>0</in1>
    <in2>0</in2>
    <cnt1>0</cnt1>
    <cnt2>0</cnt2>
    <out1>0</out1>
    <out2>0</out2>
    <custom>0</custom>
    <fw>0.18.6</fw>
    <time>09:40:23</time>
    <date>15-06-2026</date>
</status>
```

Example response in JSON format


```
{
    "model": "RFID IND2 LED Mif",
    "name": "",
    "mac": "00:00:00:00:00:00",
    "id": "40005EB4F4",
    "newId": "40005EB4F4",
    "rfid_enable": 1,
    "cnt": 1,
    "n_logs": 1,
    "in1": 0,
    "in2": 0,
    "cnt1": 0,
    "cnt2": 0,
    "out1": 0,
    "out2": 0,
    "custom": 0,
    "fw": "0.18.6"
    "time": "09:40:23",
    "date": "15-06-2026"
}
```


Tag	Description
`model`	Device model
`name`	User-assigned module name
`mac`	Device MAC address
`id`	ID of the last RFID tag read
`newId`	Flag informing about new tag read - read flag
`rfid_enable`	RFID antenna status
`cnt`	Amount of read RFID tags since last device restart
`n_logs`	Current amount of logs in the device
`in1`	Current state of input No. 0
`in2`	Current state of input No. 1
`cnt1`	Current counter value for input No. 0
`cnt2`	Current counter value for input No. 1

`out1`	Current state of relay output No. 0
`out2`	Current state of relay output No. 1
`custom`	Special order version indicator
`fw`	Firmware revision
`time`	Current device time
`date`	Current device date

HTTP GET – Control

> **Tip**
>
> **All HTTP GET commands are mapped to XML/JSON tags for HTTP Client service**

> **Warning**
>
> **Special attention should be given to the correct order in which the commands are sent. For example: sending `lcd_text[0-3]` and then `lcd_clr` will clear the display. Correct order: `lcd_clr`, then `lcd_text[0-3]`. Similarly, sending `lcd_text_imeout` AFTER `lcd_text[0-3]` will display the text only for the default 5s (the command will not apply).**

`http://<IP_address>/stat.php`

Command	Value	Description
open1	INTEGER	Control of relay No. 1 in the mode defined in **I/O Settings**
open2	INTEGER	Control of relay No. 2 in the mode defined in **I/O Settings**
out0	INTEGER	Control of relay No. 1 in bistable mode
out1	INTEGER	Control of relay No. 2 in bistable mode
text	STRING	Writes text to the LCD. A new line is started after 20 subsequent characters
lcd_text0	STRING	Transcription of one LCD text line. Text line 1
lcd_text1	STRING	Transcription of one LCD text line. Text line 2

lcd_text2	STRING	Transcription of one LCD text line. Text line 3
lcd_text3	STRING	Transcription of one LCD text line. Text line 4
lcd_clr		Clear LCD
lcd_wrap	STRING	Transcription of the LCD text with wrapping
lcd_nowrap	STRING	Transcription of the LCD text without wrapping
lcd_word	STRING	Transcription of the LCD text with wrapping, text divided for entire words
lcd_text_timeout	INTEGER	Display text for a defined period of time
lcd_clock	BOOLEAN	Display or hide the clock, does not overwrite the webpage setting
bl	STRING	LCD backlight control on off int – time [s]
ledg	STRING	Green LED: on, off, int, int, int - TimeOn*0,1s, TimeOff*0,1s, CNT=255 - repeats endlessly example: `"ledg":5,5,5` will activate the LED for 0,5s, deactivate for 0,5s and repeat the whole sequence five times.
ledr	STRING	Red LED. Similar to ledg
buzz	INTEGER	Sound emission: 1 – accept, 2 – reject, 3 - short signal example: `"buzz":2` - buzzer will emit the card rejection sound signal.
userdb_store	INTEGER	Transcription of card data from **RAM** to **FLASH**. (1) This operation should be performed only after all user modifications have been made, and as rarely as possible. It should be noted that the amount of guaranteed memory write cycles is 10 000
userdb_ver	STRING	Allows readout of card database stored in RAM (if the base is sent to the reader with HTTP Client). If the card database is updated dynamically or dependent on specified conditions, this command allows to monitor its version and to check if changes have been made.
enable	BOOLEAN	Enable or disable card reading

removeLogsN	INTEGER	Delete last n logs eg.: `"removeLogsN":10` will delete 10 oldest logs
removeLogs	1	Delete all logs
releaseId	INTEGER	Release the newId flag (if block RFID read is enabled)
cardlog_remove_n	INTEGER	Delete last n card logs eg.: `"cardlog_remove_n":4` will delete 4 oldest logs
cardlog_remove_all	1	Delete all logs
rfid_timeout	INTEGER	RFID antenna timeout after specified time
enter_state	**FLAG_PRIV=1** **FLAG_WORK=2** **FLAG_IN=4** **FLAG_OUT=8** **FLAG_EXT=10** **FLAG_KBD=20**	**Setting of entry mode.** **FLAG_PRIV (0x01):** Sets the private mode. **FLAG_WORK (0x02):** Sets the work mode. **FLAG_IN (0x04):** Sets the mode to entrance – records the entrance time. **FLAG_OUT (0x08):** Sets the exit mode – records the exit time. **FLAG_EXT (0x10):** Card application to an external reader. **Example:** The setting for **private entrance** has a value of **5**, which is the result of combining **FLAG_PRIV (0x01)** with **FLAG_IN (0x04)**

Tip

Consecutive commands in HTTP GET are to be separated with `&`.

HTTP GET – card database

RFID cards can be appended, added or deleted by using HTTP commands.

Tip

After a power cycle, the reader transcribes the card data from flash memory to RAM

Warning

After all edits have been made, it's needed to write all data to the flash memory. The flash memory write procedure should be performed as rarely as possible.

Resource: `http://<IP_address>/card.cmd?`

Command	Value	Description
addCard	UID Name Surname ID Flag	Add or modify card `addCard=UID; Name;Surname;ID;flag` - if id=-1: the reader will find an empty record and add the card to it - if id>0: the record exists in memory – edits the record; if the record does not exist – adds a record with specified ID. Returns in JSON: - -2 – operation failed – the UID is already in the database (only for id=-1) - -1 – operation failed - >0 – ID number that has been modified or added
getCardByUid	STRING	Returns a record with the card containing selected UID
getCardByUserId	INTEGER	Rerurns a record with the card containing selected user ID
removeCard	INTEGER	Deletes a card with the selected user ID
removeAllCards	INTEGER	Deletes all cards from the reader's memory
reloadCards	INTEGER	Re-read RFID cards from flash to RAM
storeCards	INTEGER	Write card data from RAM to flash

HTTP Client

In the **HTTP Client** mode, the module operates similarly to a web browser – it initiates a connection with a server, sending requests and awaiting a reply.

Main features:

- **Automatic data sending**: After an RFID tag is read, the module automatically connects to the server and transfers information about the tag read,
- **Receiving commands from the server**: The server can send module control instructions such as: enable LED's, generate sounds, display text on the LCD screen,
- **Easy integration**: The mode allows easy integration with various server technologies such as PHP, Node.js or MySQL.

Benefits:

- **Instantaneous communication**: The module automatically initiates a connection after a card is read, allowing for quick data transfer to the server,
- **Flexible control**: The server can control different module elements in response to the commands sent.

HTTP Client operation can be configured in **Services**, **HTTP Client** tab.

HTTP Server | HTTP Client | MQTT Client | E-Mail Client | Modbus | SNTP | Syslog

HTTP Client

Name	Value	Description
Enable	No	
Server		Remote server address
HTTP port	0	Port
HTTP Method	GET	HTTP request default method
Content type	text/plain	Content-type header
Resource	/	Http resource i.e. /rfid.php
User		Auth user
Password		Auth password
HTTP ping server	0	Ping time in secs. 0-disable
HTTP ping request		

- **Enable** – Enable the **HTTP Client** service,
- **Server** – IP address of a server the module initiates a connection with,
- **HTTP port** – Port, on which the server listens (default **80** for HTTP or **443** for HTTPS),
- **HTTP Method** – HTTP method used for communicating: **GET, POST, PUT, DELETE**,
- **Content type** – Header determining the type of transferred data (eg. text/plain, json, xml, form),
- **Resource** – Access path to a server resource the module communicates with,
- **User** – User name (if using server authentication),
- **Password** – User password (if using server authentication),
- **HTTP ping server** – Server query frequency (poll time),
- **HTTP ping request** – Server query request syntax (poll request).

Send card log by HTTP(S)	Disable	Enable sending logs to server. Method POST, content-type JSON
Send card log period	1	The period (in minutes) to check and send logs
Send card log request	{"logs":%json_cardlogs%}	

- **Send card log by HTTP(S)** – Enable/disable periodic transfer of event logs via HTTP(S), data is sent in packets of up to 4kB per single request,
- **Send card log period** – Log transfer frequency in minutes,
- **Send card log request** – Syntax of the request sent by the device during **Send card log period**. Must contain the `%json_cardlogs%` variable, recalling a table containing event logs (see **Predefined variables**).

For communication security, the device supports connection encryption.

SSL/TLS	Off	Enable encryption
SSL Root certificate	No	Use CA ROOT certificate
Skip cert CN check	No	Skip certificate Common Name check
Use Client certificate	No	It needs upload client's key, password and certificate
Client key password		

Save

SSL Client certificate		
SSL server root certificate	Wybierz plik Nie wybrano pliku	Upload
Client certificate	Wybierz plik Nie wybrano pliku	Upload
Client key	Wybierz plik Nie wybrano pliku	Upload

- **SSL/TLS** – Enable encryption using SSL/TLS protocols,
- **SSL root certificate** – Enable use of CA ROOT for server verification,
- **Skip cert CN check** – Skip checking of CN (Common Name) field compatibility server certificate,
- **Use Client certificate** – Enable use of client certificate to authenticate on the server side,
- **Client key password** – Password securing the client key.

All certificate files required for establishing an encrypted connection can be transferred in the **SSL Client Certificate** section.

HTTP Client – Poll request

The reader can periodically send poll requests to the server, in order to check the status or keep the connection open. The content of such request is defined in the **HTTP Ping Request** field on the **Services / HTTP Client** page. The request can contain both static text and predefined variables, which allows for dynamic adaptation to its contents.

Example: Sending a "poll" request to the resource every 6 seconds:

- **GET request** using predefined variables

```
ip=%eip%&name=%mod_name%&in1=%in1%
```

Data received on the server side:

```
GET /resource?ip=192.168.0.176&name=Taw&in1=1 HTTP/1.1
User-Agent: Ind2
Host: 192.168.0.90
Content-Length: 0
```

The server can send data back in the form of a JSON (XML) file:

```
HTTP/1.1 200 OK
Content-Length: 33
Content-Type: application/json
{"lcd_text1":" Place an RFID tag"}
```

- **POST request** using predefined variables:

```
ip=%eip%
name=%mod_name%
in1=%in1%
```

Data received on the server side:

```
OST / HTTP/1.1
User-Agent: Ind2
Host: 192.168.0.90
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
ip=192.168.0.176
name=T
in1=0
```

The server can send data back in the form of a JSON (XML) file:

```
HTTP/1.1 200 OK
Content-Length: 14
Content-Type: application/xml
<buzz>3</buzz>
```

HTTP Client – Card action event request

Application of an RFID tag to the reader can trigger sending of a HTTP request to the configured server.

The **Card Action** page allows configuration of three HTTP request send triggers:

- **On-new** – Sends the request after each application of an RFID card to the reader,
- **On-hold** – Sends the request if the card is located in the read area for at least 5 seconds (**hold time** is set to 5 seconds by default),
- **On-remove** – Sends the request if the RFID tag is removed from the read area.

> **Information**
>
> **Each scenario shown above operates independently, which allows the user to choose what events are to trigger HTTP requests. Additionally, the user can individually configure the request contents, its type and a method for each trigger.**

The user configures the request content, its type and a method for each trigger separately.

HTTP Client method	GET	HTTP method
HTTP Content type	plain/text	HTTP Content type
HTTP Client request	mac=%emac%&ip=%eip%&id=%uid%	

HTTP request configuration parameters:

- **HTTP Client method** – HTTP method used for communicating:
  - **Disable** – The event will not be sent to the server,
  - **Default** – The default method defined in **Services** will be used,
  - **GET**,
  - **POST**,
  - **PUT**,
  - **DELETE**.
- **HTTP Content type** – Type of the content-type header:
  - **Default** – Default header defined in **Services**,
  - **plain/text**,
  - **json**,
  - **xml**.
- **HTTP request** – HTTP request syntax.

> **Tip**
>
> **If the Search card in database option (Card Action, On new card action) is set to All like unknown, the reader will not search the internal card database. User variables will assume the values of an unknown card.**

Example 1

For data from the screenshot and **On new card action** trigger:

HTTP Client request: `mac=%emac%&ip=%eip%&id=%uid%`

After a card is applied, data on the server side will look like this:

```
GET /?mac=00:00:00:00:00:00&ip=192.168.0.176&id=12005CB99F HTTP/1.1
User-Agent: Ind2
Host: 192.168.0.90
Content-Type: text/plain
Content-Length: 0
```

The server can return data in the form of a JSON (XML) file:

```
HTTP/1.1 200 OK
Content-Length: 41
Content-Type: application/json
{
"lcd_text1":" Welcome!",
"buzz":"3"
}
```

Example 2

Request: `ip=%eip%&id=%uid%&czas=%time%&state=VARIABLE`

VARIABLE assumes different values depending on the selected trigger

- **On new card action** → VARIABLE = put,
- **Action hold card** → VARIABLE = hold,
- **Action remove card** → VARIABLE = away.

For the HTTP request shown above, the server will receive the following data after an RFID card is applied to the reader for 10 seconds:

```
ip=192.168.0.176&id=12005EE365&czas=12:19:00&state=put //On new card action
ip=192.168.0.176&id=12005EE365&czas=12:19:05&state=hold //Action hold card
ip=192.168.0.176&id=12005EE365&czas=12:19:11&state=away //Action remove card
```

This HTTP request syntax allows for simultaneous reader control.

If using the reader with a card slot enclosure, the user will be registered in the system when a card is applied (put status). The hold status indicates that the card is present in the slot, granting access to a room / activating a device, while the remove status indicates a card removal action, canceling the authorization.

HTTP Client – Control

Controlling the module in HTTP Client mode is possible after enabling **HTTP Client** in the **Services** tab.

> **Warning**
>
> **Special attention should be given to the correct order in which the commands are sent. For example: sending `lcd_text[0-3]` and then `lcd_clr` will clear the display. Correct order: `lcd_clr`, then `lcd_text[0-3]`. Similarly, sending `lcd_text_imeout` AFTER `lcd_text[0-3]` will display the text only for the default 5s (the command will not apply).**

Command	Value	Description
open1	INTEGER	Control of relay No. 1 in the mode defined in **I/O Settings**
open2	INTEGER	Control of relay No. 2 in the mode defined in **I/O Settings**

out0	INTEGER	Control of relay No. 1 in bistable mode
out1	INTEGER	Control of relay No. 2 in bistable mode
text	STRING	Writes text to the LCD. A new line is started after 20 subsequent characters
lcd_text0	STRING	Transcription of one LCD text line. Text line 1
lcd_text1	STRING	Transcription of one LCD text line. Text line 2
lcd_text2	STRING	Transcription of one LCD text line. Text line 3
lcd_text3	STRING	Transcription of one LCD text line. Text line 4
lcd_clr		Clear LCD
lcd_wrap	STRING	Transcription of the LCD text with wrapping
lcd_nowrap	STRING	Transcription of the LCD text without wrapping
lcd_word	STRING	Transcription of the LCD text with wrapping, text divided for entire words
lcd_text_timeout	INTEGER	Display text for a defined period of time
lcd_clock	BOOLEAN	Display or hide the clock, does not overwrite the webpage setting
bl	STRING	LCD backlight control on off int – time [s]
ledg	STRING	Green LED: on, off, int, int, int - TimeOn*0,1s, TimeOff*0,1s, CNT=255 - repeats endlessly example: `"ledg":5,5,5` will activate the LED for 0,5s, deactivate for 0,5s and repeat the whole sequence five times.
ledr	STRING	Red LED. Similar to ledg
buzz	INTEGER	Sound emission: 1 – accept, 2 – reject, 3 - short signal example: `"buzz":2` - buzzer will emit the card rejection sound signal.

addCard	UID Name1 Name2 ID Flag	Add or edit card. `addCard=UID;Name1;Name2;ID;flag` - if `id=-1` : the reader will find an empty record and add the card there - if `id>0` : the record is present in the memory - it will be modified; if it's absent - a new record with the specified ID will be added Returns json: - -2 – operation failed – UID already in database (only for id=-1) - -1 – operation failed - >0 – new/modified UID
renewUser	[{ uid: STRING, name1: STRING name2: STRING userId: INTEGER flag: INTEGER }, ...]	Overwrite the entire card database to RAM in JSON format. - If `userId=-1` , the record will be added to the first free memory slot. - If `userId≠-1` , the card will be added to the slot specified in `userId` . `flag=1` indicates an active card. The `userId` and `flag` variables are not required and should be sent only after their modification
userdb_store	INTEGER	Transcription of card data from **RAM** to **FLASH**. (1) This operation should be performed only after all user modifications have been made, and as rarely as possible. It should be noted that the amount of guaranteed memory write cycles is 10 000
userdb_ver	STRING	Allows readout of card database stored in RAM (if the base is sent to the reader with HTTP Client). If the card database is updated dynamically or dependent on specified conditions, this command allows to monitor its version and to check if changes have been made.
enable	BOOLEAN	Enable or disable card reading
removeLogsN	INTEGER	Delete last n logs eg.: `"removeLogsN":10` will delete 10 oldest logs
removeLogs	1	Delete all logs
releaseId	INTEGER	Release the newId flag (if block RFID read is enabled)

cardlog_remove_all	1	Delete all logs
rfid_timeout	INTEGER	RFID antenna timeout after specified time
enter_state	**FLAG_PRIV=1** **FLAG_WORK=2** **FLAG_IN=4** **FLAG_OUT=8** **FLAG_EXT=10** **FLAG_KBD=20**	**Setting of entry mode.** **FLAG_PRIV (0x01):** Sets the private mode. **FLAG_WORK (0x02):** Sets the work mode. **FLAG_IN (0x04):** Sets the mode to entrance – records the entrance time. **FLAG_OUT (0x08):** Sets the exit mode – records the exit time. **FLAG_EXT (0x10):** Card application to an external reader. **Example:** The setting for **private entrance** has a value of **5**, which is the result of combining **FLAG_PRIV (0x01)** with **FLAG_IN (0x04)**

Example response to overwriting of user database:

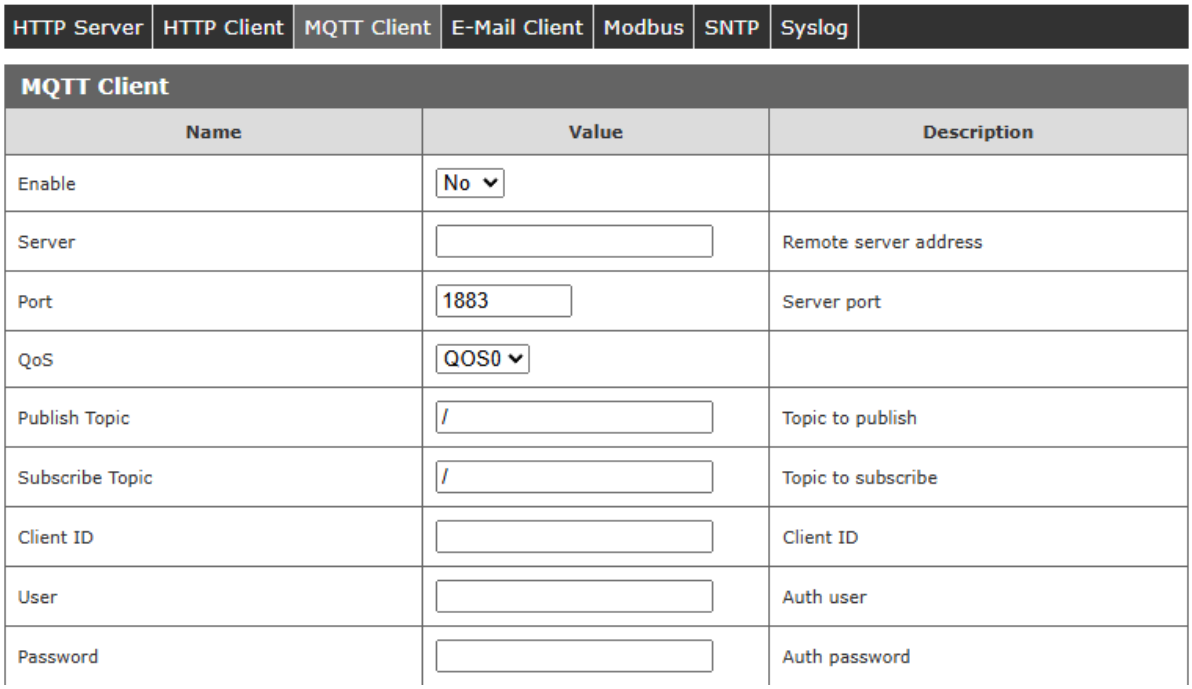

```
{
   "userdb_ver":"test",
   "renewUser":[
      {
         "uid":"12345678",
         "name1":"John",
         "name2":"Smith",
         "userId":-1,
         "flag":1,
         "parent":-1
      },
      {
         "uid":"ABCDEF01",
         "name1":"Jan",
         "name2":"Kowalski",
         "userId":-1,
         "flag":1,
         "parent":-1
      }
   ]
}
```

Example of php server support:

```
<?php
if( $_GET["id"] ) // receive id and MAC - $_GET["mac"]
{
$who=$_GET["id"];
// check id in DB and do some action
echo "<root>";
echo "<buzz>1</buzz>"; // sound signal
echo "<text>Card ID: $who</text>" // message ID on LCD
echo "<open1>1</open1>"; // door open
echo "</root>";
}
else // no id – poll request
{
echo "<text> Place an RFID tag</text>"; //standby text
}
```

MQTT Client

The device supports the MQTT protocol which allows data exchange in publication / subscription architecture. The settings can be configured in the **Services** tab under **MQTT client**.



- **Enable** – Enable MQTT service,
- **Server** – IP address or domain of the MQTT broker,
- **Port** – Port on which port the MQTT server listens (default 1883),
- **QoS** – Quality of Service, supported by the device:
  - **QoS 0** – At most once,
  - **QoS 1** – At least once,
  - **QoS 2** – Exactly once.
- **Publish Topic** – Topic to which the device sends data,

- **Subscribe Topic** – Topic, from which the device receives data,
- **Client ID** – MQTT client ID,
- **User** – User name (if authentication is needed),
- **Password** – User password (if authentication is needed),

The device supports the encrypted MQTT protocol. Available configuration options are similar to those available for HTTP Client configuration.

Publish topic

After the MQTT protocol is configured, triggers causing data to be sent to a defined topic (**Publish Topic**) should be declared. Similarly to HTTP Client, there are three triggers available:

- **On new card action** – Send the MQTT message every time an RFID card is applied to the reader,
- **Action hold card** – Send a MQTT message in a card remains in the read field for at least 5 seconds (default hold time),
- **Action remove card** – Send a MQTT message if the RFID tag is removed from the read area.

Each of the triggers operates independently, which allows configuring their operation to suit the user's needs. The user can choose what events cause a MQTT message to be sent.

For MQTT payload contents:

```
uid=%uid%&name1=%usr_name1%&name2=%usr_name2%
```

Data visible on the server side:

```
uid=87878A4C&name1=John&name2=Smith
```

Tip

The data visible on the server side depend on the contents of the sent message. If the Search card in database option (Card Action page in On new card action section) is set to All like unknown, the device will not search the interncal card database. In this case, all user variables will be treated as an unknown card.

Data visible on the server side with **Search card in database** disabled:

```
uid=878A8A4C&name1=&name2=
```

Subscribe topic

The device can be controlled via the MQTT protocol. To make this possible, Subscribe Topic from which the device will receive control commands must be configured. To do that, input the required MQTT broker settings on the **Services / MQTT** site and define the **Subscribe Topic**.

Example configuration:

MQTT Client		
Name	**Value**	**Description**
Enable	Yes	
Server	mqtt.inveo.com.pl	Remote server address
Port	1883	Server port
QoS	QOS0	
Publish Topic	/reader/prod2/sec1	Topic to publish
Subscribe Topic	/reader/prod2/sec1/c	Topic to subscribe

Information

The device sends data to the `reader/prod2/sec1` topic,
It receives the messages from `reader/prod2/sec1/c`

Tip

The device supports all HTTP protocol commands. They should be divided with `&`.

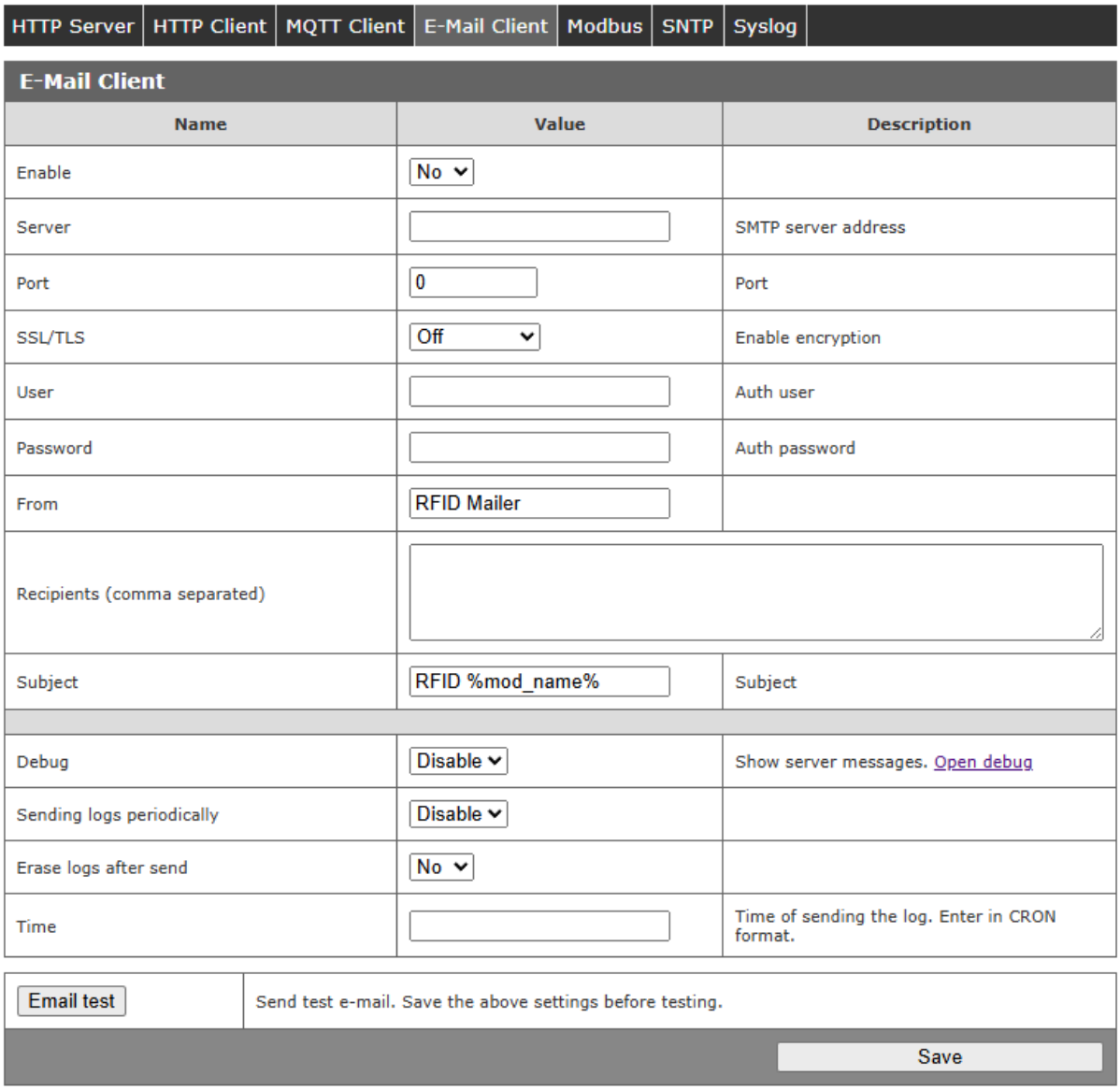
Example message sent to `reader/prod2/sec1/c`:

```
lcd_text1= Place an RFID tag&buzz=3
```

The device will display **Place an RFID tag** in the second LCD line and emit a short (**beep**). It's possible to use the same topic for publishing and subscribing – this enables instant device control after data is sent to **Publish Topic**.

E-mail

The device can send encrypted e-mail messages that can be configured in the **Services** tab.



- **E-mail** – Activate the SMTP service,
- **Server** – SMTP server address,
- **Port** – SMTP server port, on which messages are received,
- **SSL/TLS** – Encryption configuration:
  - **Off** – No encryption, connection unencrypted,
  - **SSL/TLS** – SSL/TLS Encryption,
  - **STARTTLS** – Enables the use of STARTTLS during connection.
- **User** – User name used for authentication in the SMTP server,
- **Password**,
- **From** – "From" field visible in the e-mail message header,
- **Recipients** – List of message recipients, e-mail addresses separated by commas,
- **Subject** – Subject of the e-mail message and name of the attachment file (eg. logs),
- **Debug** – Debugging mode enabling diagnostics of SMTP configuration errors.
- **Sending logs periodically** – Log file extension to be sent as an attachment,

- **Erase logs after send** – Option to erase the logs after they are successfully sent,
- **Time** – CRON format log sending schedule (seconds, minutes, hours, day, month, day of the week). For the syntax from the above screenshot: `0 0 0 ? * *` → every day at midnight.

Testing

After applying the settings, the user can test the configuration using the **E-mail test** button. The device will attempt to send a test message, that should reach the recipient immediately. If no message is received, check the SPAM folder.

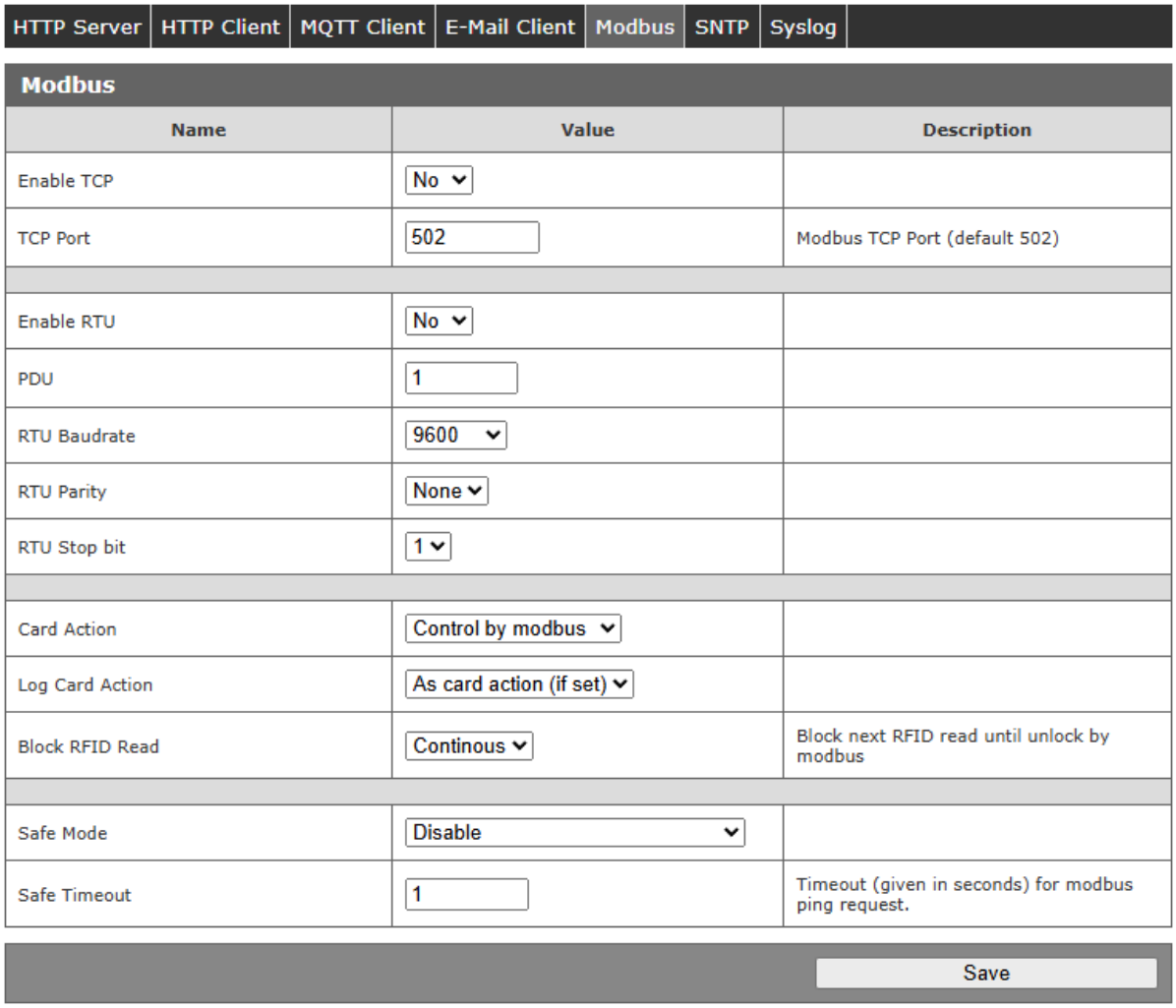
Troubleshooting

If problems with message sending arise, **Debug** mode can be enabled in order to diagnose the errors:

1. Open the debugging page in a new browser tab, typing in:
   `http://<IP_address>/protect/dmesg.html`
   or click **Open debugger here!** on the configuration page,
2. Click the **Send a test e-mail** button,
3. Details concerning the message sending problems will appear in the debugger's window. Analysis of this information will allow the problem to be identified and solved.

Modbus TCP/RTU

The device supports the Modbus TCP and Modbus RTU protocols. Connection configuration is available on the webpage under the **Services / Modbus** tab.



- **Modbus TCP** – Enable the Modbus TCP service,
- **TCP port** – Modbus TCP protocol port (default 502),
- **Modbus RTU** – Enable the Modbus RTU service,
- **PDU** – Device Modbus address,
- **RTU Baudrate** – Data transfer speed; available values:
  - 1200
  - 2400
  - 4800
  - 9600
  - 19200
  - 38400
  - 57600
  - 115200

- **RTU Parity** – Parity settings:
  - None
  - Odd
  - Even
- **RTU Stop bit** – Number of stop bits:
  - 1
  - 2
- **Card action** – Reaction of the reader for card application:
  - **Control by modbus** – Control of the device by Modbus only,
  - **Reader card action** – Reaction compliant with settings from the **Card Action** tab.
- **Log Card Action** – Logging of events during a correct connection with master (only for the **Card Action** → **Reader card action** setting):
  - **As card action (if set)** – Logging compliant with **Card Action** settings,
  - **Omit log action** – Omit the **Card Action** settings.
- **Block RFID read** – Block read of a next tag:
  - **Block** – After a tag is read, value 1 is sent to the Holding Register 1. Readout of another transponder is possible after a read flag is reset (value 0 in **Holding Register 1**),
  - **Continous** – Automatic erasure of the read flag – allows instantaneous reading of other RFID tags.
- **Safe mode** – Reaction of the reader in the event that Modbus connection is lost:
  - **Disable**,
  - **No action** – No reader reaction for an applied tag,
  - **Standalone (Reader Card Action)** – Transition to autonomous mode.
- **Safe Timeout** – Time until the reader enters **Safe mode**.

The device supports the following Modbus functions:

- 0x01 Read Coils
- 0x03 Read Holding Register
- 0x05 Write Single Coil
- 0x06 Write Single Register
- 0x0F Write Multiple Coils
- 0x10 Write Multiple Registers

Holding Registers addresses

Address	R/W	Description
1	R/W	Read flag Read: 1 – a new transponder has been read Write: 0 – zeroizing of read flag

2	R	ID_LEN
3	R	Card ID/UID/CSN [0]
4	R	Card ID/UID/CSN [1]
5	R	Card ID/UID/CSN [2]
6	R	Card ID/UID/CSN [3]
7	R	Card ID/UID/CSN [4]
8	R	Card ID/UID/CSN [5]
9	R	Card ID/UID/CSN [6]
10	R	Card ID/UID/CSN [7]
11	R	Card ID/UID/CSN [8]
12	R	Card ID/UID/CSN [9]
13	R	Reserve
14	R	User ID – **65535** indicates an unknown tag
20	R/W	Output #1 control according to **I/O Settings** tab
21	W	Output #1 control in bi-stable mode
22	W	Output #1 enable for a time set as a parameter (value * 0,1s)
23	W	blink_on time (value * 0,1s)
24	W	blink_off time (value * 0,1s)
25	W	Output #1 control in flashing mode (time set in registers 23/24)
26	R/W	Output #2 control according to **I/O Settings** tab
27	W	Output #2 control in bi-stable mode
28	W	Output #2 enable for a time set as a parameter (value * 0,1s)

29	W	blink_on time (value * 0,1s)
30	W	blink_off time (value * 0,1s)
31	W	Output #2 control in flashing mode (time set in registers 29/30)

Logs

Address	R/W	Description
1200	R/W	Log amount [MSB], writing 0 will delete all logs
1201	R	Log amount [LSB]
1202	R	Selected log [MSB]
1203	R	Selected log [LSB] (log numbering starts from 0)
1204	W	Delete oldest logs, amount set as a parameter
1205	R	Timestamp [MSB]
1206	R	Timestamp [LSB]
1207	R	Reserve
1208	R	Log source (main reader, auxiliary reader)
1209	R	Action
1210	R	UserID – **FFFF** indicates an unknown tag
1211	R	UID – Code saved in ASCII
1231	R	Last register containing possible data

Display

Address	R/W	Description
2000	R/W	LCD Mode: 0 – autonomous operation -1 – Modbus control

		Value from 1 to 65534 – text display time in seconds. After the time passes, the display will enter autonomous mode
2001	R/W	LCD TIME, if LCD Mode=1: 0 – do not display time on LCD 1 – display time on LCD
2002	R/W	LCD Clear: 1 – clear LCD contents
2003	W	Display backlight: 0 – disable backlight 1 – enable backlight >1 – backlight time (value * 0,1s)
2010-2029	W	LCD first line – UTF-8 encoding
2030-2049	W	LCD second line – UTF-8 encoding
2050-2069	W	LCD third line – UTF-8 encoding
2070-2089	W	LCD fourth line – UTF-8 encoding

Single Coil addresses

Address	R/W	Description
1	R/W	Output #1 control according to **I/O Settings**
2	R/W	Output #2 control according to **I/O Settings**
3	R	Output #1 coil state
4	R	Output #2 coil state
11	R	Input #1 state
12	R	Input #2 state
13	R/W	Red LED control
14	R/W	Green LED control

15	W	Sound generator control - ACCEPT sound
16	W	Sound generator control - REJECT sound
17		Reserve
18	R/W	Reset status Read: 1 – a restart occured (eg. because of power loss)

Lua scripts

Clicking the **Lua Editor** button in the **RFID reader / Card data** tab will open an editor window, which allows the user to define non-standard actions and control logic using the Lua script language.



> **Tip**
>
> **Select Yes in the Lua script prior to opening the editor.**

Interface description

The main tab section contains a code editor that can be used to create and edit Lua scripts, as well as a console window that enables preview of diagnostic information in real-time.

Control buttons

- **Load to module** – Upload the Lua script to the device (without saving to non-volatile memory),
- **Store in FLASH** – Save the Lua script to device memory. The script will remain active after a restart,
- **Clear console** – Clear the output console.

Available system functions

Lua functions can be used to make the device react to different events.

System functions

- `start()` – Recalled after the system start-up,
- `timeout()` – Recalled after `set_timeout(ms)` has passed,
- `set_timeout(ms)` – Sets the internal timer and recalls the `timeout()` function,
- `clear_timeout()` – Clears and disables the timer.

RFID functions

- `rfid_detect()` – Recalled when the system detects an RFID card and reads its UID,
- `action_oncard()` – The first function activated after a card is detected (before checking the database),

- `action_known()` – Recalled when the card is recognized as known,
- `action_unknown()` – Recalled when a card is not present in the database,
- `action_hold()` – Recalled when a card remains in the read area for a longer time,
- `action_remove()` – Recalled when a card is removed from the read area.

Lua libraries

The system offers several libraries enabling interaction with the device. The `card`, `mf`, `rfid`, and `dev` libraries are specific to Inveo devices. All other libraries are standard.

Biblioteka `card`

- `card.getFlag()` – Pobiera flagi związane z bieżącą kartą UID,
- `card.getUserId()` – Pobiera identyfikator użytkownika powiązany z bieżącą kartą.

Biblioteka `mf` (Mifare Classic)

- `mf.auth_ee(key_no, block, ab)` – Autoryzacja bloku Mifare Classic przy użyciu klucza z pamięci zabezpieczonej,
- `mf.auth_key(key, block, ab)` – Autoryzacja bloku Mifare Classic przy użyciu podanego klucza,
- `mf.rd(block)` – Odczyt bloku Mifare Classic,
- `mf.wr(block, data)` – Zapis bloku Mifare Classic (musi mieć 16 bajtów),
- `mf.deauth()` – Deautoryzacja.

Biblioteka `rfid`

- `rfid.getUidBin()` – Pobiera UID jako dane binarne,
- `rfid.getUidBinLen()` – Pobiera długość UID w formacie binarnym,
- `rfid.getUidAscii()` – Pobiera UID jako tekst ASCII (HEX),
- `rfid.getUidAsciiLen()` – Pobiera długość UID w formacie ASCII,
- `rfid.overwriteUidAscii(uid)` – Nadpisuje UID używany przez system,
- `rfid.pause(time)` – Zatrzymuje przetwarzanie RFID na podany czas (ms).

Biblioteka `dev`

- `dev.out_on(ch)` – Enable output on a given channel `ch`,
- `dev.out_off(ch)` – Disable the output on a given channel `ch`,
- `dev.out_single(ch, ton, toff)` – Sets a single pulse, `ton` long after a delay of `toff` (in 0,1s) on the output channel `ch`,
- `dev.out_blink(ch, ton, toff, cnt)` – Blinks the output `ch` `cnt` - times with pulses of `ton` length separeted by `toff` (in 0,1s),
- `dev.out_get(ch)` – Returns the `ch` output state,
- `dev.in_get(ch)` – Returns the `ch` input state,
- `dev.in_get_cnt(ch)` – Returns the `ch` input counter,

- `dev.delayms(ms)` – Delays the next step by the value in miliseconds,
- `dev.buzz(sound)` – Buzzer control (0 - accept, 1 - reject, 2 - pulse),
- `dev.lcd_clear()` – Clears the LCD screen,
- `dev.get_enter_state()` – Returns the card action flag (consult **Events preview using the webpage**).

Communication

E-mail

- `dev.email(recipient, subject, text)` – Sends an e-mail,
  - `recipient` – Recipient address,
  - `subject` – Message topic,
  - `text` – Message content.

HTTP

- `dev.http(url, data, data_length, method, content_type)` – Sends a HTTP request:
  - `url` – Address, the request will be sent to. Entering `null` will send the request to the address specified in the **HTTP Client** tab,
  - `data` – Data to be sent,
  - `data_length`,
  - `method`:
    - `0` - GET,
    - `1` - POST,
    - `2` - PUT,
    - `3` - DELETE.
  - `content_type`:
    - `0` - PLAIN,
    - `1` - JSON,
    - `2` - XML.

MQTT

- `dev.mqtt(topic, data, length)` – Publishes an MQTT message.

Logging and debugging

- `dev.log(message)` – Saves message to flash memory,
- `dev.console(message)` – Dispays the message in the Lua console.

HTTP Server – Request handling

- `dev.httpdscript(name, args)` – Function recalled when an HTTP request begins with `/data/script/`. Can be used to trigger a specific Lua script in response to a request sent to the reader,

- `dev.httpd_resp(text)` – Sets the HTTP response.

Example

```
function httpdscript(name, arg)
    dev.console("name:"..name.." arg:"..arg)

    local v1,v2,v3 = arg:match("v1=(%d+)&v2=(%d+)&v3=(%d+)")
    dev.console("parsed arg:"..v1.." "..v2.." "..v3)

    if name == "test.json" then
        dev.httpd_resp('{"result":"OK"}')
    elseif name == "test.xml" then
        dev.httpd_resp('<AastraIPPhoneStatus Beep="yes"><Session/><Message type="alert"
Color="darkgreen" Timeout="3">Door open</Message></AastraIPPhoneStatus>')
    else
        dev.httpd_resp("")
    end
end
```

`table` library

- `table.insert(t, [pos], value)` – Inserts `value` to array `t`. If `pos` is specified, inserts the element to a specified position, shifting other elements,
- `table.remove(t, [pos])` – Removes the element from `pos` in `t` array. If `pos` is not specified, removes the last added element,
- `table.sort(t, [, comp])` – Sorts the `t` array in ascending order. Optionally, the `comp(a, b)` comparison function can be added; it should return `true` if `a` should be before `b`,
- `table.concat(t [, sep [, i [, j]]])` – Combines the `t` array contents into a single string, separating them with `sep` from index `i` to `j`,
- `table.pack(...)` – Returns an array containing all function arguments. Remembers the amount of arguments as `n`,
- `table.unpack(t [, i[, j]])` – Returns the contents of `t` array as separate values for indexes `i` to `j`. Default: `i = 1`, `j = #t`.

`string` library

- `string.bytebyte(s [, i [, j]])` – Returns the byte code for `s` characters, from `i` to `j`,
- `string.char(...)` – Creates a string from the specified values,
- `string.len(s)` – Returns the character string length,
- `string.lower(s)` – Returns `s` with upper-case letters converted to lower-case,
- `string.upper(s)` – Returns `s` with lower-case letters converted do upper-case,
- `string.sub(s, i [, j])` – Returns the `s` substring from `i` to `j`,
- `string.rep(s, n)` – Returns `s` repeated `n` times,
- `string.reverse(s)` – Returns the inverted text of `s`,
- `string.find(s, pattern [, i [, plain]])` – Searches for `pattern` in `s`, returns the indexes or `nil`. Setting `plain` to `true` searches with no patterns,

- `string.match(s, pattern [, i])` – Returns the first pattern match,
- `string.gsub(s, pattern, repl [, n])` – Substitutes `pattern` with the `repl` string a maximum of `n` times. Returns the new string and the amount of substitutions.

`os` library

- `os.time([table])` – Returns current time in UNIX format or converts `table` to seconds,
- `os.date([format [, time]])` – Returns the date/time in the form of formatted text or a table,
- `os.clock()` – Returns the time spent for program exacution by the processor,
- `os.difftime(t1, t2)` – Returs the difference in seconds between `t1` and `t2`,

`math` library

- `math.abs(x)` – Absolute value of `x`,
- `math.ceil(x)` – Round up to the nearest whole number,
- `math.floor(x)` – Round down,
- `math.fmod(x, y)` – Remainder of the `x` by `y` division operation,
- `math.max(x, ...)` – The biggest number specified,
- `math.min(x, ...)` – The smallest number specified,
- `math.sqrt(x)` – Square root of `x`,
- `math.exp(x)` – `e` to the power of `x`,
- `math.log(x [, base])` – Natural logarithm `ln(x)` or `log_b(x)` (if `base` is specified),
- `math.pow(x, y)` – Exponentiation of `x^y`,
- `math.modf(x)` – Splits `x` into integer and fractional parts `int, frac`,
- `math.random([m [, n]])` – Random number between 0 and 1,
- `math.randomseed(seed)` – Sets a seed of the random number generator,
- `math.deg(x)` – Converts radians to degrees,
- `math.rad(x)` – Converts degrees to radiants,
- `math.sin(x)` – Sinus (`x` in radians),
- `math.cos(x)` – Cosinus (`x` in radians),
- `math.tan(x)` – Tangens (`x` in radians),
- `math.asin(x)` – Arcus sinus,
- `math.acos(x)` – Arcus cosinus,
- `math.atan(x)` – Arcus tangens,
- `math.atan2(y, x)` – Arcus tangens `y/x` including the quadrant.

`utf8` library

- `utf8.len(s [, i [, j]])` – Returns the number of UTF-8 characters in the substring `s` from `i` to `j`,
- `utf8.offset(s, n [, i])` – Returns the byte index in `s` of the beggining of the nth character (if `n` positive) or the nth from the end (if `n` negative),

- `utf8.codepoint(s [, i [, j]])` – Returns at least one Unicode point code of characters from `s` between `i` and `j`,
- `utf8.char(...)` – Creates an UTF-8 string from the given Unicode point codes,
- `utf8.codes(s)` – Returns the iterator, that returns subsequent byte indexes and Unicode point codes in `s`,
- `utf8.charpattern` – Lua pattern for matching a single UTF-8 character.

`pack` library

- `pack.pack` – Packs data to a binary string,
- `pack.unpack` – Unpacks data from a binary string,
- `pack.packsize` – Returns the size of a packet.

Factory settings, backup

Restoring factory settings

To restore the device to factory settings:

- Power up the device.
- Short the **RESET** jumper for a time between 10 and 15 seconds.
- Remove the **RESET** jumper when the signal sounds.

> **Information**
>
> **Factory settings can be restored by using the built-in www server. The user should go to the Administration tab, then in Backup tab press the Reset to default button, which is located at the bottom of the page.**

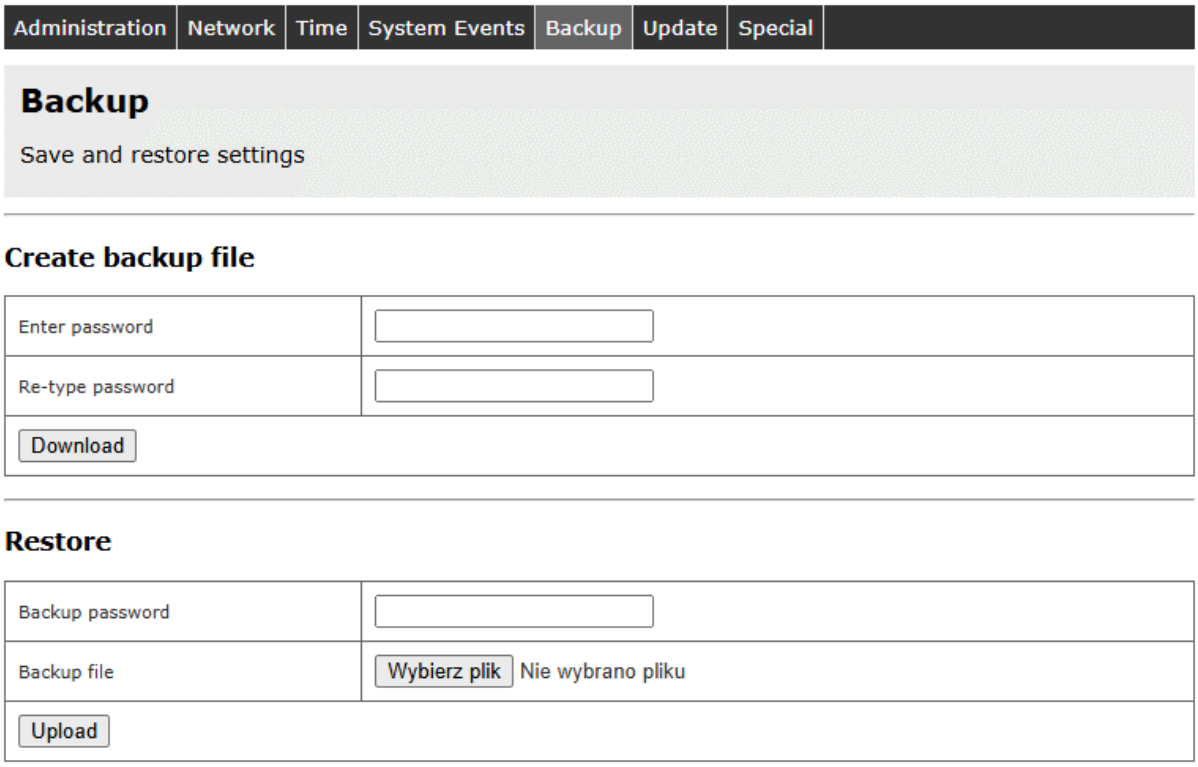
Once the above steps have been completed, the device will be set to the following parameters:

- **IP address:** 192.168.111.15
- **IP mask:** 255.255.255.0
- **User:** admin
- **Password:** admin

Backup

All configuration data can be saved to a file. The user can create/upload the backup file on the **Administration / Backup** page.

Creation of a configuration data save requires a password.



Emergency firmware restore

To restore firmware after a failure:

- Short the **RESET** jumper.
- Power up the device and connect it to LAN.

Once the above steps have been completed, the device will be set to the following parameters:

- **IP address**: 192.168.111.15
- **IP mask**: 255.255.255.0

> **Tip**
>
> **In order to access the 192.168.111.15 address, computer's IP address must be in the same subnet (example computer address: 192.168.111.1).**

Referring to the specified IP address will give access to the device's bootloader.



In this location, it is possible to upload firmware, reset the device to factory settings and reboot.

Warranty and manufacturer's liability

> **Warning**
>
> **The manufacturer provides a two-year warranty for the device and post-warranty service for a period of 10 years from the date of introduction of the device to the market. The warranty covers all material and production defects.**

The manufacturer undertakes to respect the warranty agreement, if the following conditions are met:

- All repairs, changes, expansions and device calibrations are carried out by the manufacturer or an authorized service center,
- The power supply system meets the applicable standards,
- The device is operated in accordance with the suggestions presented in this manual,
- The device is operated in accordance with its intended purpose.

The manufacturer assumes no responsibility for consequences resulting from improper installation, improper use of the device, failure to comply with the instruction manual, and repairs made by unauthorized personnel.

> **Warning**
>
> **The device contains no user serviceable parts inside.**

Storage, operation and transport conditions

The device should be stored in enclosed rooms, where the atmosphere is free from vapours and corrosive substances:

- Environment temperature from -30°C to +60°C (-22°F - 140°F),
- Humidity from 25% to 90% (condensation unacceptable),
- Atmospheric pressure from 700 to 1060 hPa.

The device is intended to operate in the following conditions:

- Environment temperature from -10°C do +55°C (14°F - 131°F),
- Humidity from 30% to 75%,
- Atmospheric pressure from 700 to 1060 hPa.

Recommended transport conditions:

- Environment temperature from -40°C do +85°C (-40°F - 185°F),
- Humidity from 5% to 95%,
- Atmospheric pressure from 700 to 1060 hPa.

Installation and device operation:

- The module should be operated in accordance with recommendations provided later in this manual.

Disposal and decommissioning

In an event the device needs to be decommissioned (eg. after its intended life period is surpassed), it is recommended to contact the manufacturer or his representative, who are responsible to respond appropriately, i.e., to collect the device from the user. The user can alternatively contact companies specializing in electronic device or computer equipment disposal and/or decommissioning. Under no condition should the device be placed with other waste.