User Manual

Revision 1.000
English

Programmable Unit

(Order Code: HD67H98-B2, HD67H98-G43-B2, HD67H98-xxx-B2, HD67H98-G43-xxx-B2)

For Website information:
www.adfweb.com?Product=HD67H98-B2
www.adfweb.com?Product=HD67H98-G43-B2
www.adfweb.com?Product=HD67H98-xxx-B2
www.adfweb.com?Product=HD67H98-G43-xxx-B2

For Price information:
www.adfweb.com?Price=HD67H98-B2
www.adfweb.com?Price=HD67H98-G43-B2
www.adfweb.com?Price=HD67H98-xxx-B2
www.adfweb.com?Price=HD67H98-G43-xxx-B2

Benefits and Main Features:

- Configurable via Webserver
- Power Supply 24 V DC
- Temperature range: -40°C/+85°C (-40°F/+185°F)
- Programmable in C, C++, C#, Python and others
- Integrated VPN service

For others MQTT Broker products, see also the following links:

Converter MQTT broker to

www.adfweb.com?Product=HD67C01	**(Ethernet)**
www.adfweb.com?Product=HD67C02	**(M-Bus Master)**
www.adfweb.com?Product=HD67C03	**(Serial)**
www.adfweb.com?Product=HD67C04	**(Modbus Master)**
www.adfweb.com?Product=HD67C05	**(Modbus Slave)**
www.adfweb.com?Product=HD67C06	**(Modbus TCP Master)**
www.adfweb.com?Product=HD67C07	**(Modbus TCP Slave)**
www.adfweb.com?Product=HD67C08	**(BACnet Master)**
www.adfweb.com?Product=HD67C09	**(BACnet Slave)**
www.adfweb.com?Product=HD67C10	**(CAN)**
www.adfweb.com?Product=HD67C11	**(CANopen)**
www.adfweb.com?Product=HD67C12	**(DALI)**
www.adfweb.com?Product=HD67C13	**(DeviceNet Master)**
www.adfweb.com?Product=HD67C14	**(DeviceNet Slave)**
www.adfweb.com?Product=HD67C15	**(DMX)**
www.adfweb.com?Product=HD67C16	**(EtherNet/IP Slave)**
www.adfweb.com?Product=HD67C17	**(J1939)**
www.adfweb.com?Product=HD67C18	**(KNX)**
www.adfweb.com?Product=HD67C19	**(NMEA 0183)**
www.adfweb.com?Product=HD67C20	**(NMEA 2000)**
www.adfweb.com?Product=HD67C21	**(PROFIBUS Master)**
www.adfweb.com?Product=HD67C22	**(PROFIBUS Slave)**
www.adfweb.com?Product=HD67C23	**(PROFINET Slave)**
www.adfweb.com?Product=HD67C24	**(SNMP Manager)**
www.adfweb.com?Product=HD67C25	**(SNMP Agent)**
www.adfweb.com?Product=HD67C26	**(EtherNet/IP Master)**
www.adfweb.com?Product=HD67C27	**(PROFINET Master)**

Do you need to choose a device? Do you want help?
www.adfweb.com?Cmd=helpme

INDEX

UPDATED DOCUMENTATION

Dear customer, we thank you for your attention and we remind you that you need to check that the following document is:

- Updated
- Related to the product you own

To obtain the most recently updated document, note the "document code" that appears at the top right-hand corner of each page of this document.

With this "Document Code" go to web page www.adfweb.com/download/ and search for the corresponding code on the page. Click on the proper "Document Code" and download the updates.

REVISION LIST

Revision	Date	Author	Chapter	Description
1.000	04/11/2024	Ff	All	First release version
1.001	10/03/2025	Ff	All	Revision
1.002	03/10/2025	Ff	All	Revision

WARNING

ADFweb.com reserves the right to change information in this manual about our product without warning.
ADFweb.com is not responsible for any error this manual may contain.

TRADEMARKS

All trademarks mentioned in this document belong to their respective owners.

SECURITY ALERT

General Information

To ensure safe operation, the device must be operated according to the instructions in the manual. When using the device, legal and safety regulation are required for each individual application. The same applies also when using accessories.

Intended Use

Machines and systems must be designed so the faulty conditions do not lead to a dangerous situation for the operator (i.e. independent limit switches, mechanical interlocks, etc.).

Qualified Personnel

The device can be used only by qualified personnel, strictly in accordance with the specifications.
Qualified personnel are persons who are familiar with the installation, assembly, commissioning and operation of this equipment and who have appropriate qualifications for their job.

Residual Risks

The device is state-of-the-art and is safe. The instruments can represent a potential hazard if they are inappropriately installed and operated by untrained personnel. These instructions refer to residual risks with the following symbol:

This symbol indicates that non-observance of the safety instructions is a danger for people that could lead to serious injury or death and / or the possibility of damage.

CE conformity

The declaration is made by our company. You can send an email to support@adfweb.com or give us a call if you need it.

CONNECTION SCHEME

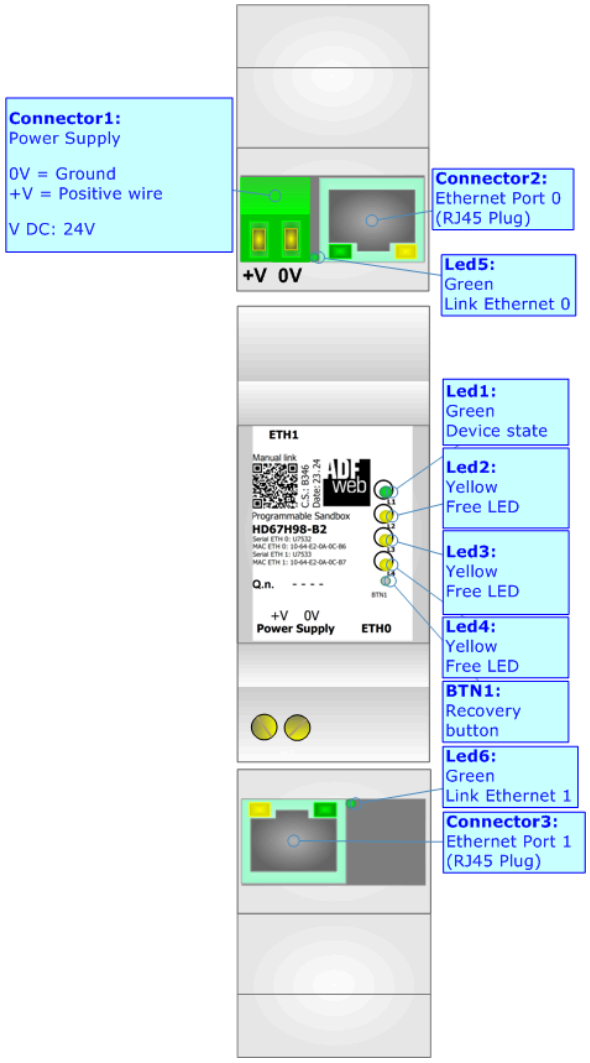


Fig. 1a: Connection Scheme for HD67H98-B2

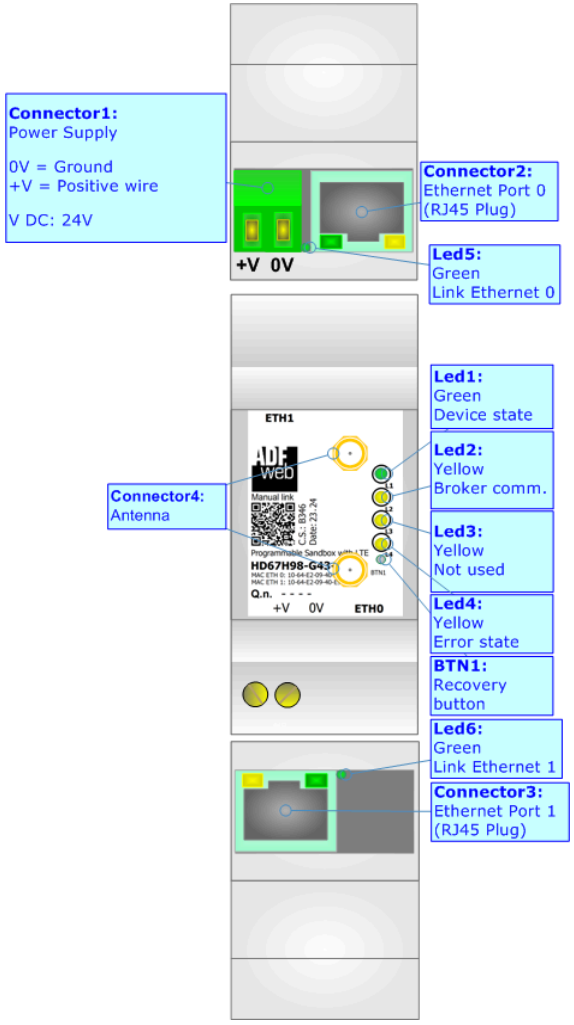


Fig. 1b: Connection Scheme for HD67H98-G43-B2

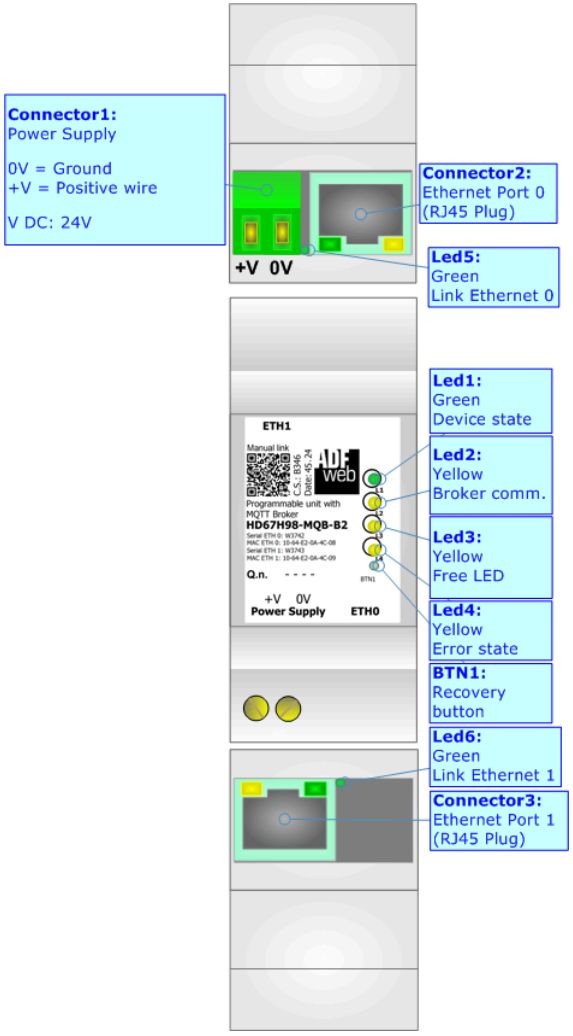


Fig. 1c: Connection Scheme for HD67H98-MQB-B2

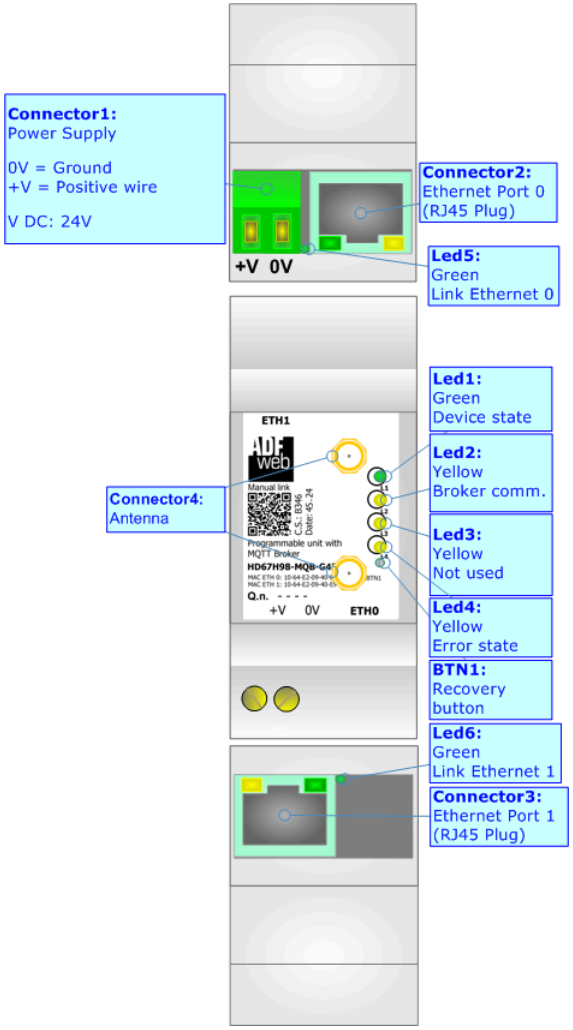


Fig. 1d: Connection Scheme for HD67H98-MQB-G43-B2

CHARACTERISTICS

The HD67H98-B2 is a programmable unit running on a Linux machine.

It allows the following characteristics:

- Hardware specifications:
  - Rev. 1:
    - 32-bit, 800Mhz CPU
    - 512 MB RAM
    - 5.5 GB User usable Flash
    - 2x 100Mb Ethernet
    - 1x LTE modem (for HD67H98-G43-B2 version)
- Electrical isolation between Ethernet and Power Supply;
- Mountable on 35mm Rail DIN;
- Power supply: 24V DC;
- Wide temperature range: -40°C / 85°C [-40°F / +185°F].

CONFIGURATION

You need "ADFweb Discovery Tool" software on your PC in order to discover the device into the network and see its IP address.
The configuration is made using an integrated webserver and it is used to:

- Define the parameter of Ethernet interfaces;
- Define the webserver settings;
- Generate the key pair for secure SSH connection;
- Enable VPN feature;
- Update the device.

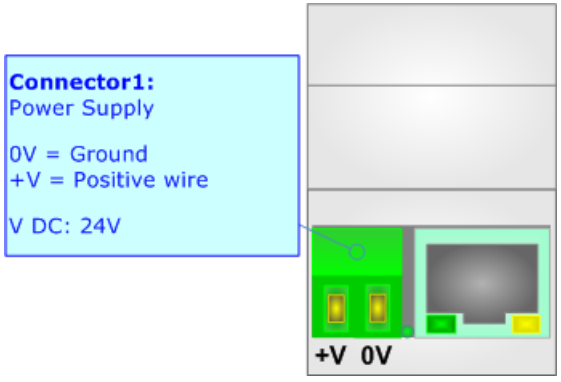
POWER SUPPLY

The devices can be powered at 12…35V DC. For more details see the two tables below.

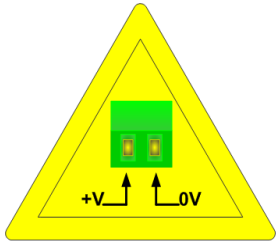
VDC ⎓
24V

Consumption at 24V DC:

Device	Consumption [W/VA]
HD67H98-xxx-B2	3.5



Caution: Not reverse the polarity power

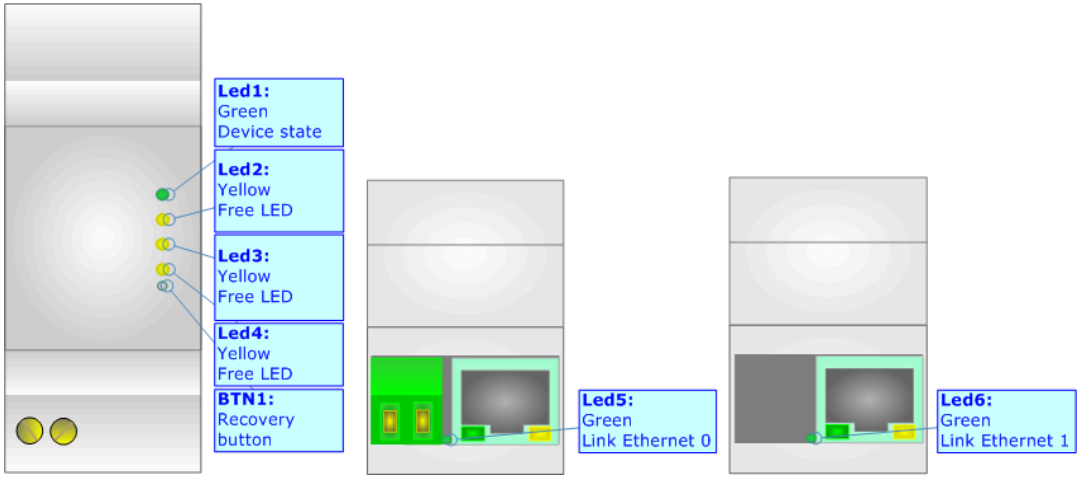


HD67H98-xxx-B2

LEDS (HD67H98 without expansions)

The device has got six LEDs that are used to give information of the functioning status. Meanings of the LEDs when the device is ON are described in the table below. During the start-up, there is a fixed animation of the LEDs (Scrolling LEDs) that will stop as soon as the hardware is ready.

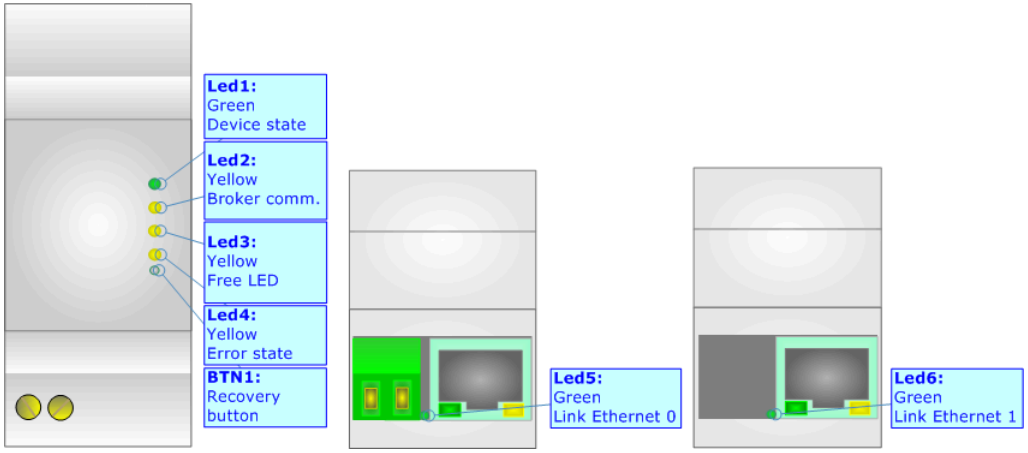
LED	Normal Mode	Recovery Mode
1: Device State (green)	Blinks slowly (~1Hz)	Blinks quickly
2: Free LED (yellow)	Programmable by user	Blinks quickly
3: Free LED (yellow)	Programmable by user	Blinks quickly
4: Free LED (yellow)	Programmable by user	Blinks quickly
5: Link Ethernet 0 (green)	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted
6: Link Ethernet 1 (green)	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted



LEDS (HD67H98 with MQTT Broker expansion)

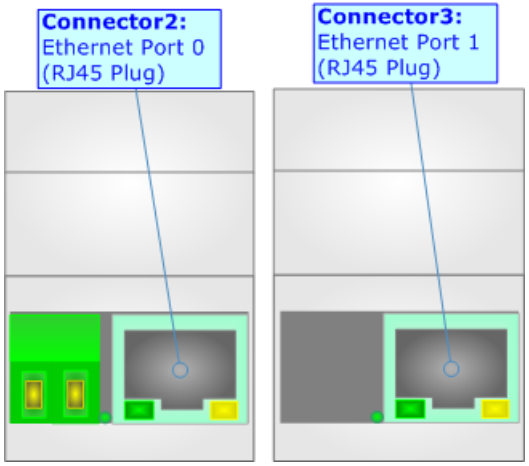
The device has got six LEDs that are used to give information of the functioning status. Meanings of the LEDs when the device is ON are described in the table below. During the start-up, there is a fixed animation of the LEDs (Scrolling LEDs) that will stop as soon as the hardware is ready.

LED	Normal Mode	Recovery Mode
1: Device State (green)	Blinks slowly (~1Hz)	Blinks quickly
2: Broker comm. (yellow)	It blinks when a MQTT message is received	Blinks quickly
3: Free LED (yellow)	Programmable by user	Blinks quickly
4: Error state (yellow)	**ON:** bridge connection is in error **OFF:** No errors are present	Blinks quickly
5: Link Ethernet 0 (green)	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted
6: Link Ethernet 1 (green)	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted	**ON:** Ethernet cable inserted **OFF:** Ethernet cable not inserted



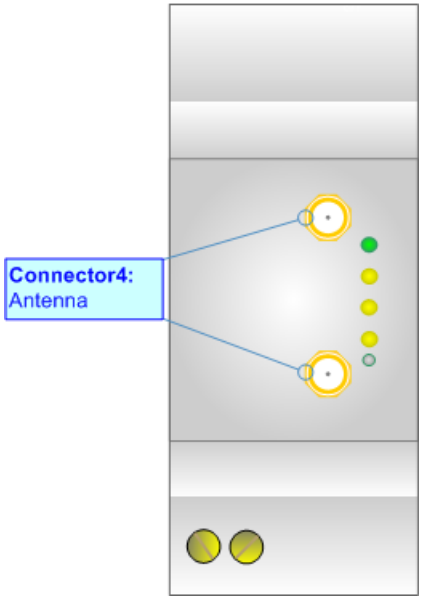
ETHERNET

The Ethernet port is used for programming the device and Ethernet communication.
The Ethernet connection must be made using Connector2 and/or Connector3 of HD67H98-xxx-B2 with at least a Category 5E cable. The maximum length of the cable should not exceed 100m. The cable has to conform to the T568 norms relative to connections in cat.5 up to 100 Mbps. To connect the device to an Hub/Switch is recommended the use of a straight cable, to connect the device to a PC is recommended the use of a cross cable.



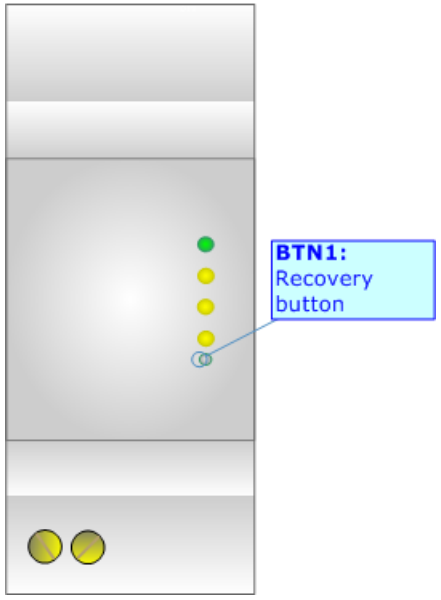
MOBILE

The HD67H98-G43-B2 uses LTE module. The Antenna connector is a SMA Female ('Female Outer Shell' and 'Female Receptacle') so the Antenna must have a SMA Male connector (there are two antennas for HD67H98-G43-B2). The SIM type is Micro-Sim.



RECOVERY BUTTON

In order to recover the device in case of wrong firmware updating or any error that compromises correct functioning of the device, it is necessary to press the BTN1. After pressing, the device will be switched in Recovery Mode and it will be possible to update again the firmware or reset the device using the default IP address 192.168.2.206 via webserver.



USE OF ADFWEB DISCOVERY TOOL SOFTWARE

To discover the device into the network and see its IP Address, use the available software that runs with Windows called “ADFweb Discovery Tool”. It is downloadable from here: www.adfweb.com/download/filefold/ADFweb_Discovery_Tool.zip.

The software works with MSWindows (XP, Vista, Seven, 8, 10, 11; 32/64bit).

USE OF WEB SERVER

To configure basic HD67H98-B2, it is possible to use the integrated web server too.
Communication will be done over the HTTPS protocol, so with encrypted communication.
The default IP address from factory is 192.168.2.206 and the device is reachable at this url: https://192.168.2.206/

As soon as the browser is launched and connected to the IP address of the device, the login page appears.
Default username and password: admin/admin.

It appears like below:

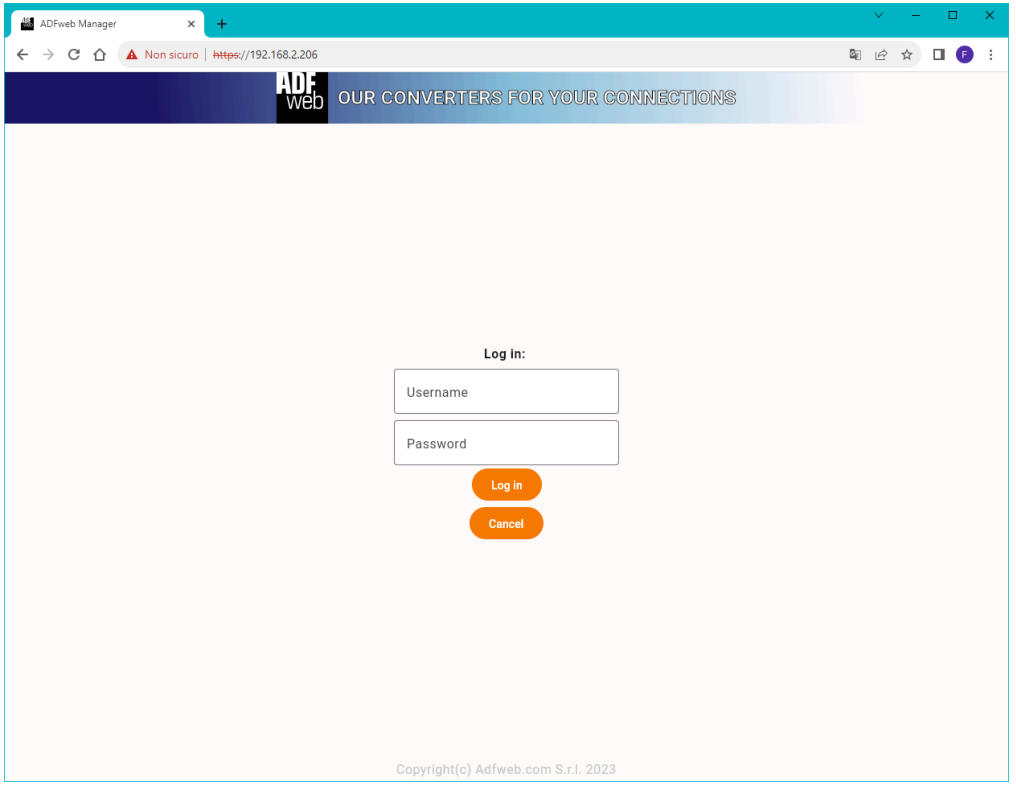


Figure 2: "Login"

When login is done, the main page with device info will appear. In the top-left corner, there will be a menu that allows to open a drawer with all sections of the webserver.

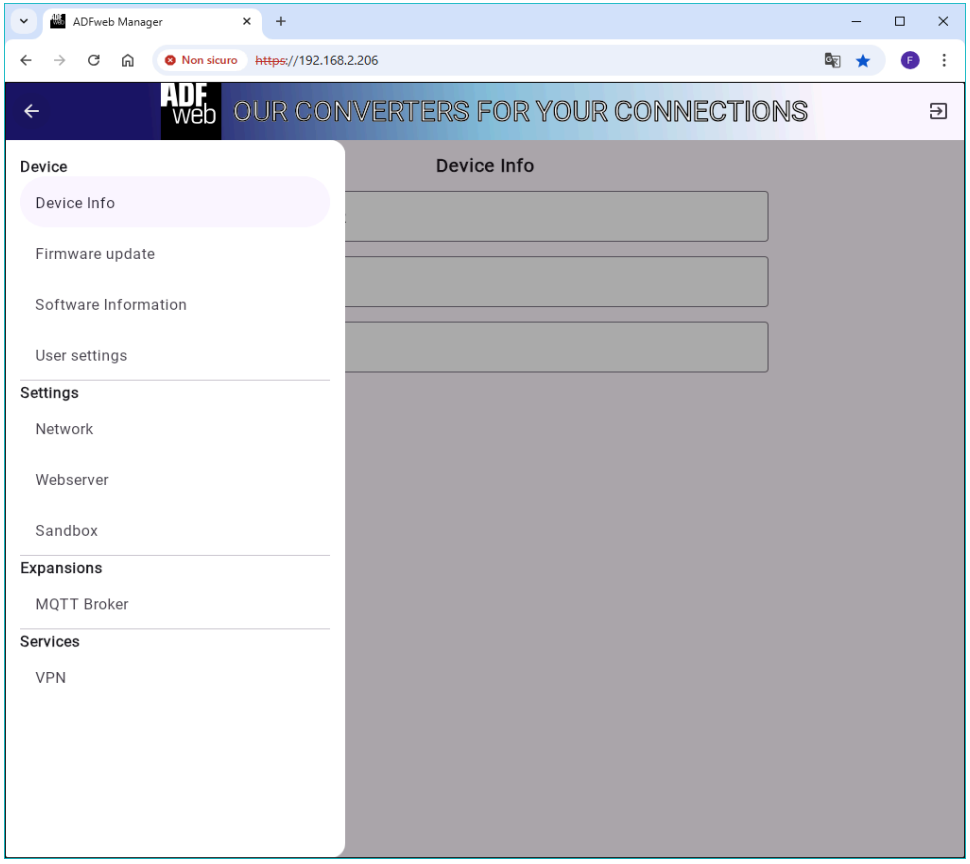


Figure 3: "Drawer"

The sections are described below:

- Device
  - Device info
  - Firmware update
  - Software information
  - User settings
- Settings
  - Network
  - Webserver
  - Sandbox
- Expansions
  - MQTT Broker
- Services
  - VPN

DEVICE → DEVICE INFO

This section shows main info about the device.

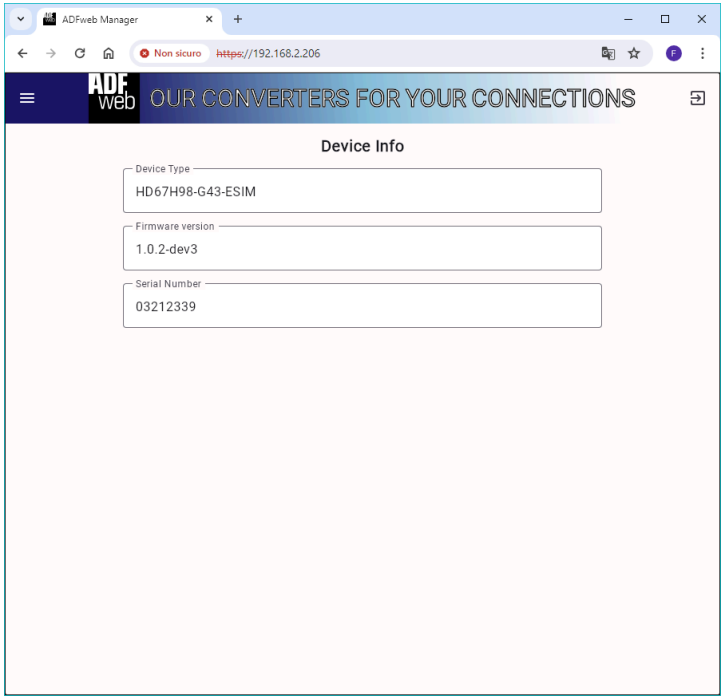


Figure 4: "Device Info" page

The meaning of the fields for the "Device Info" section are:

- In the field "**Device Type**" the name of the product is displayed;
- In the field "**Firmware version**" the version of the firmware is displayed;
- In the field "**Serial number**" the serial number is defined.

DEVICE → SOFTWARE INFORMATION

This section lists the software packages and their respective license.

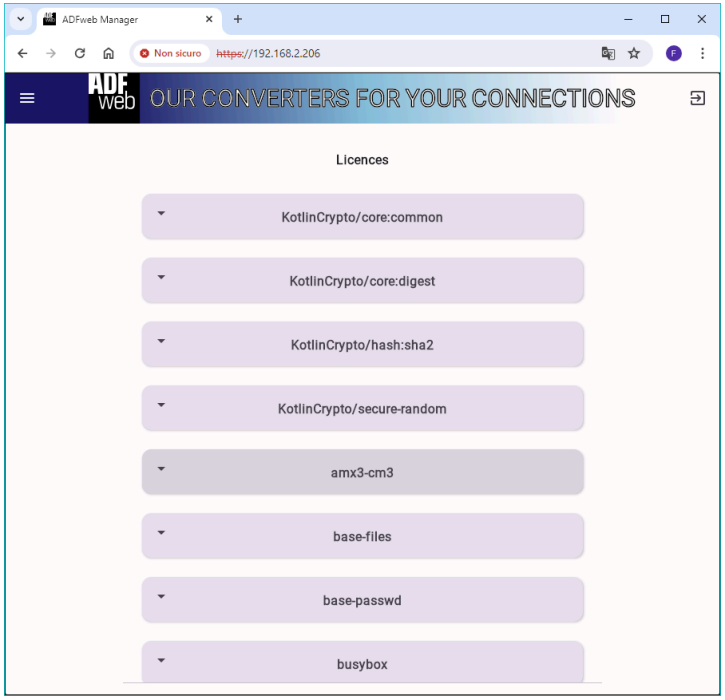


Figure 5: "Software information" page

Note:
If you need the sources of the packages subjected to a Copyleft, please contact us at support@adfweb.com.

DEVICE → FIRMWARE UPDATE

This section is used to download and update the firmware of the device and export/import full configuration.
If required, it is possible to reset the device to Factory settings.

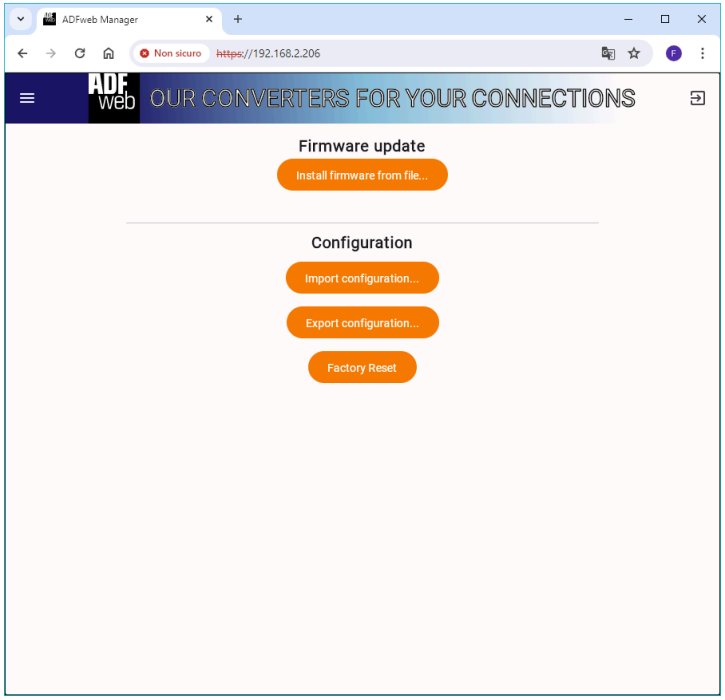


Figure 6: "Firmware update" page

DEVICE → USER SETTINGS

This section is used to change the username and/or password of the webserver.
All settings of this page can be imported or exported by pressing the related buttons.

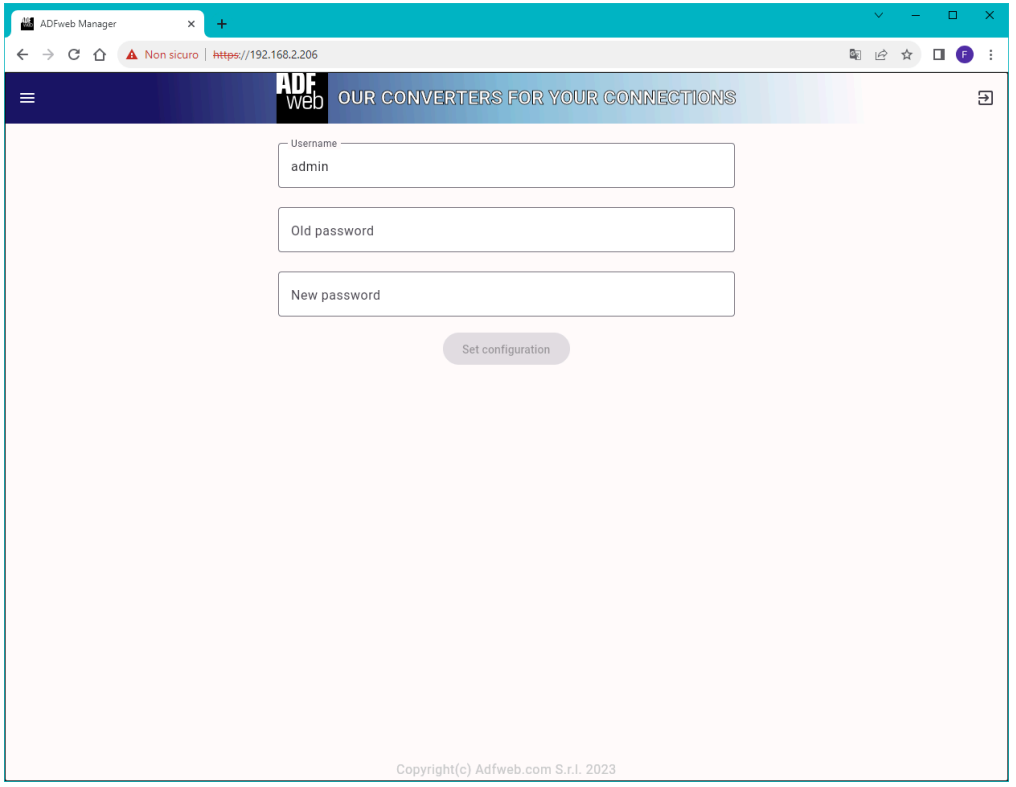


Figure 7: "User settings" page

SETTINGS → NETWORK

This section is used to set the Network interfaces of the device.
All settings of this page can be imported or exported by pressing the related buttons.

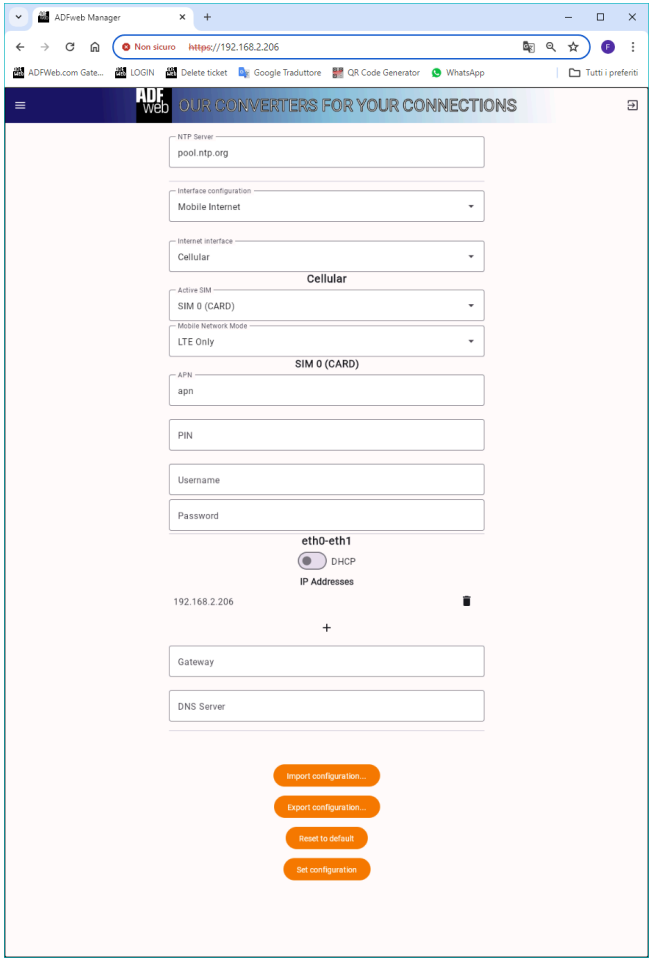


Figure 8: "Network settings" page

The meaning of the fields for the "Network settings" section are:

- In the field "**Server NTP**" it is possible to set the NTP server address;
- In the field "**Interface configuration**" it is possible to isolate the Ethernet interfaces (Separate Ethernet interfaces), bridge them to act like a single interface (Single Ethernet interface) or set LTE connection (Mobile Internet);
- In the field "**Internet interface**" it is possible to select the interface used for Internet connection;
- In the section "**Cellular**" the mobile settings of the interface are defined. The parameters are:
  - Active SIM: SIM slot used for the mobile connection;
  - Mobile Network Mode: it is possible to select the network type (LTE only or automatically selected);
  - APN: APN for the mobile connection;
  - PIN: PIN code of the SIM card;
  - Username: user for the mobile connection;
  - Password: password for the mobile connection.
- In the section "**Ethernet**" the IP settings of the interface are defined. It is possible to:
  - Set "DHCP" option;
  - Set a static IP Address by adding IP Address, SubNet Mask, Gateway and DNS.

SETTINGS → WEBSERVER

This section is used to update the certificate and key for TLS encryption. By default, the device has its own certificate, but it is possible to update it to be recognized, trusted and configured in the final installation. It is also possible to change the HTTP port.
All settings of this page can be imported or exported by pressing the related buttons.

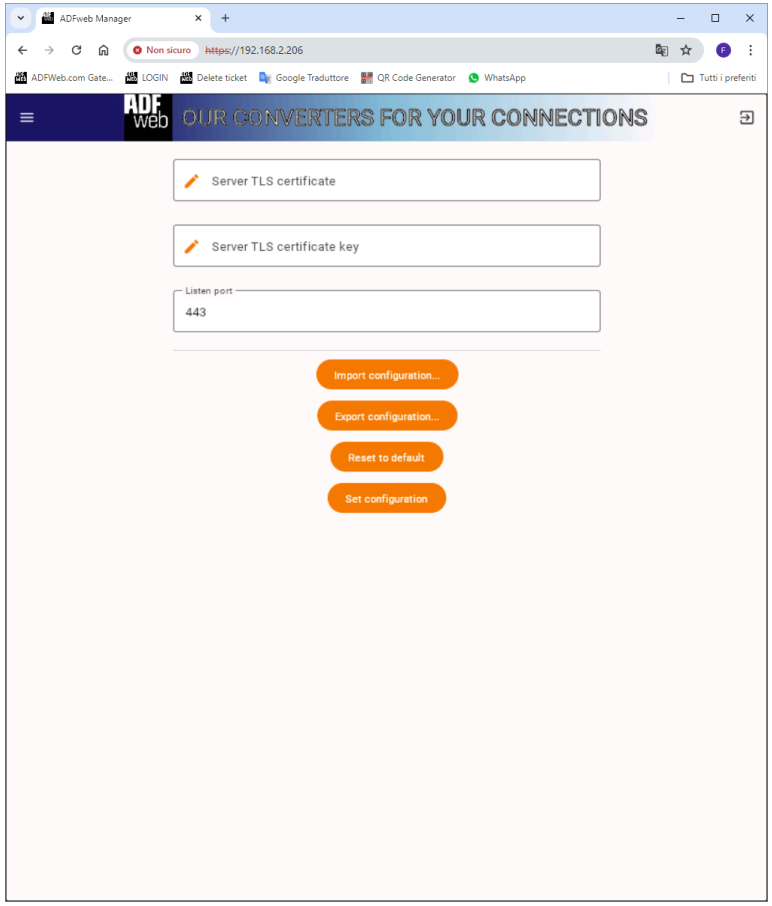


Figure 9: "Webserver settings" page

SETTINGS → SANDBOX

This section is used to set the parameters of the unit (Sandbox is the 'safe zone' where the user program will run).
All settings of this page can be imported or exported by pressing the related buttons.

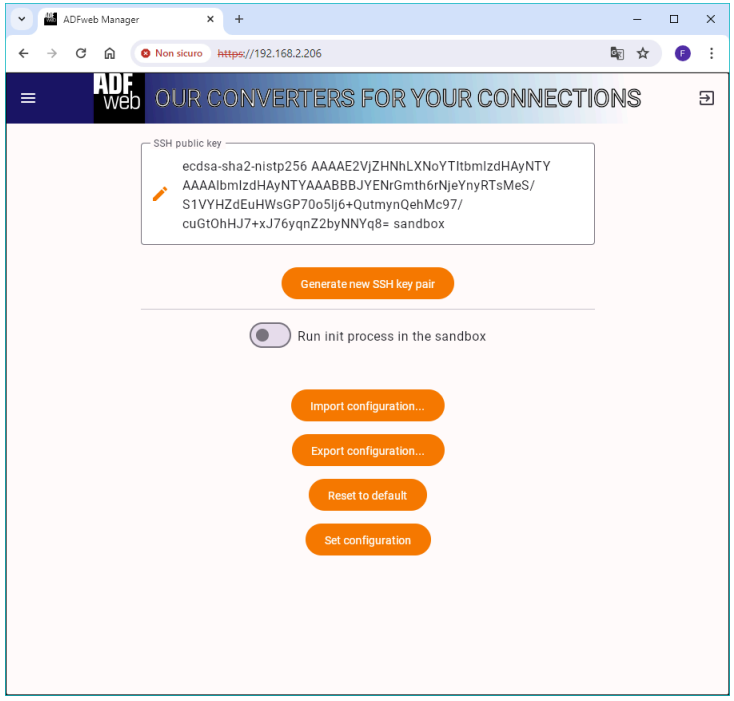


Figure 10: "Sandbox" page

The meaning of the fields for the "Sandbox" section are:

- In the field "**SSH public key**" the SSH public key is defined. It is possible to generate a new key pair with the "Generate new SSH key pair" button. The public key will populate the field while the private key will be downloaded. The key pair generation is local to the browser for security reasons. The keys are in OpenSSH format;
- With the switch "**Run init process in the sandbox**" it is possible to enable the execution of the user defined Init program placed in /home/sandbox/init.

EXPANSIONS → MQTT BROKER (for HD67H98-MQB version)

This section is used to set all parameters of MQTT broker and MQTT bridge. All settings of this page can be imported or exported by pressing the related buttons.

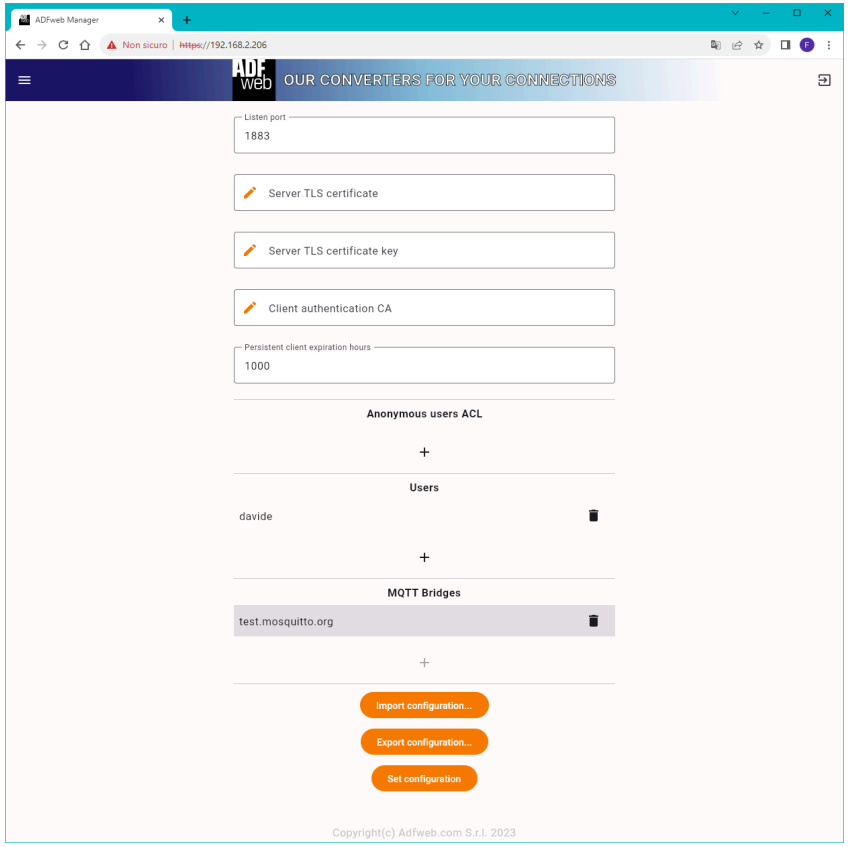


Figure 11: "MQTT Broker" page

The means of the fields for the "MQTT Broker" section are:

- In the field "**Local port**" the MQTT port of the broker is defined
- In the field "**Server TLS certificate**" the .pem certificate for MQTT communication over TLS is defined;
- In the field "**Server TLS certificate key**" the .pem private key for MQTT communication over TLS is defined;
- In the field "**Client authentication CA**" the .pem CA certificate used to generate the Client certificate and private key for Client authentication;
- In the field "**Persistent client expiration hours**" the amount of hours before a Client section expires is defined.

In the section "Anonymous users ACL" it is possible to define the list of accepted/denied topics from/to anonymous clients by clicking on '+'.
The means of the fields are described below:

In the field "**Topic**" the topic name is defined

In the field "**Topic permissions**" the access of the topics is defined.

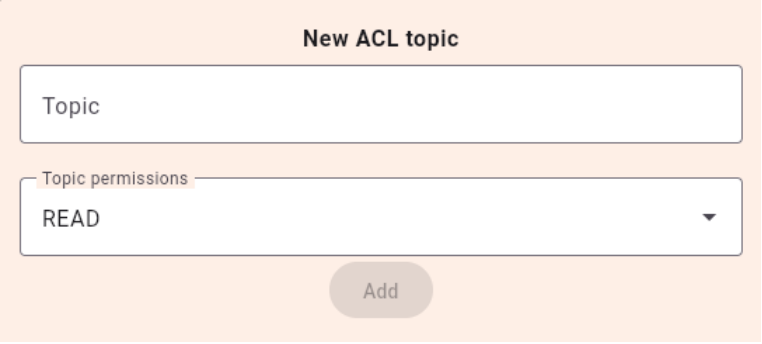


Figure 12: "New ACL topic" page

In the section "Users" it is possible to define the list of trusted clients and their accepted/denied topics by clicking on '+'. The means of the fields are described below:

- In the field "**Username**" the username of the Client is defined;
- In the field "**Password**" the password of the Client is defined.

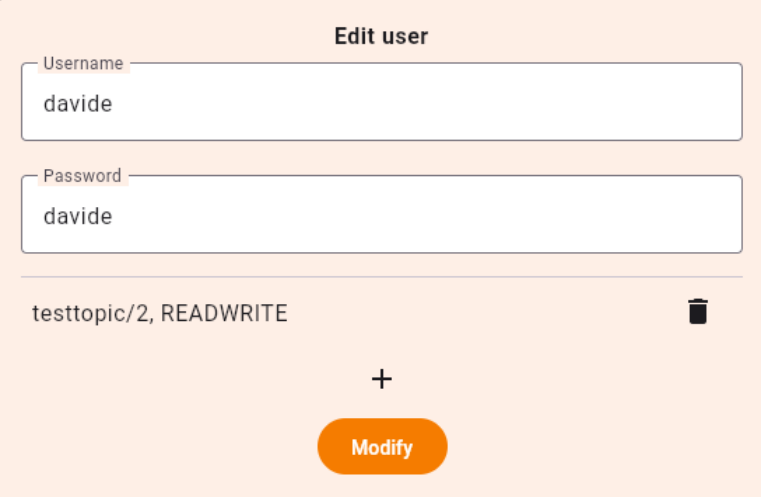


Figure 13: "New user" page

In order to add the accepted/denied topics by the user, just click on '+'. The meanings of the fields are the same described for Anonymous users.

In the section "MQTT Bridges" it is possible to define the connection to external broker/server for bridging functionalities. The means of the fields are described below:

- In the field "**Address**" the server url or IP Address is defined
- In the field "**Port**" the MQTT port is defined;
- In the field "**Bridge Protocol Version**" the MQTT varsion is defined;
- If the field "**Clean Session**" is checked, the last MQTT messages are deleted by the Server and the Client in case of missing ACK. If unchecked, the Server and the Client hold the last MQTT messages and, in case of incorrect disconnection or missing ACK, they try to send again them since all the ACK messages are exchanged correctly;
- In the field "**Keepalive interval in seconds**" the delay with which the Keep Alive message is sent on MQTT is defined;
- If the field "**Outgoing retain**" is checked, the published topics will have retain bit active, otherwise it will be inactive;
- In the field "**Remote Client ID**" the Client ID of the bridge is defined;
- In the field "**Remote Username**" the username of the bridge is defined;
- In the field "**Remote Password**" the password is defined;
- In the field "**CA Certificate**" the .pem CA Certificate of the remote broker/server used to open a secure connection is defined;In the field "**Client certificate**" the .pem Client's Certificate is defined;
- In the field "**Client certificate private key**" the corresponding .pem Client's Certificate key is defined.

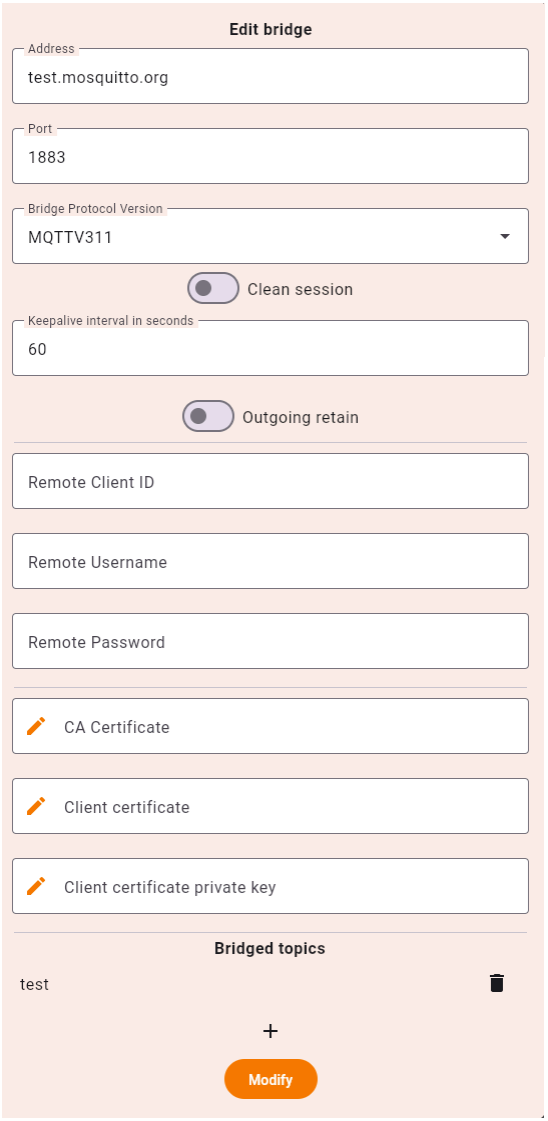


Figure 14: "Edit bridge" page

In the section "Bridged topics" it is possible to define the list of incoming and outgoing topics and the related associations. The means of the fields are described below:

- In the field "**Description**" the description of the rule is defined;
- In the fields "**Local topic**" and "**Remote topic**" it is possible to define the mapping between the local broker and the remote broker. The messages received on the local topic will be published on the remote broker in the remote topic and vice versa. Using the field "**Bridge direction**" it is possible to limit this behaviour to a single direction: either only from the local broker to the remote setting it to OUT or only from the remote broker to the local setting it to IN;
- In the field "**QoS**" the QoS for publishing and subscribing to the remote topic of the remote broker is defined.

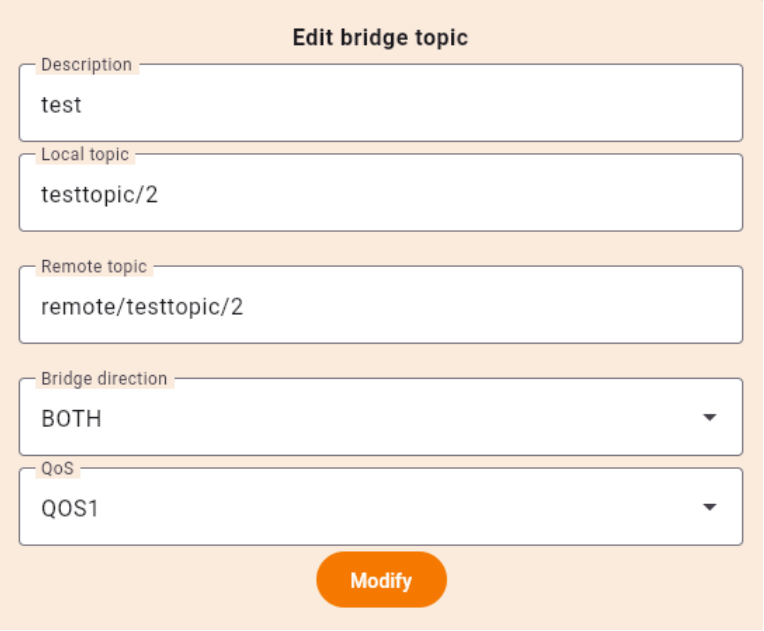


Figure 15: "Edit bridge topic" page

SERVICES → VPN

This section is used to set and enable/disable the VPN service integrated into the Sandbox.

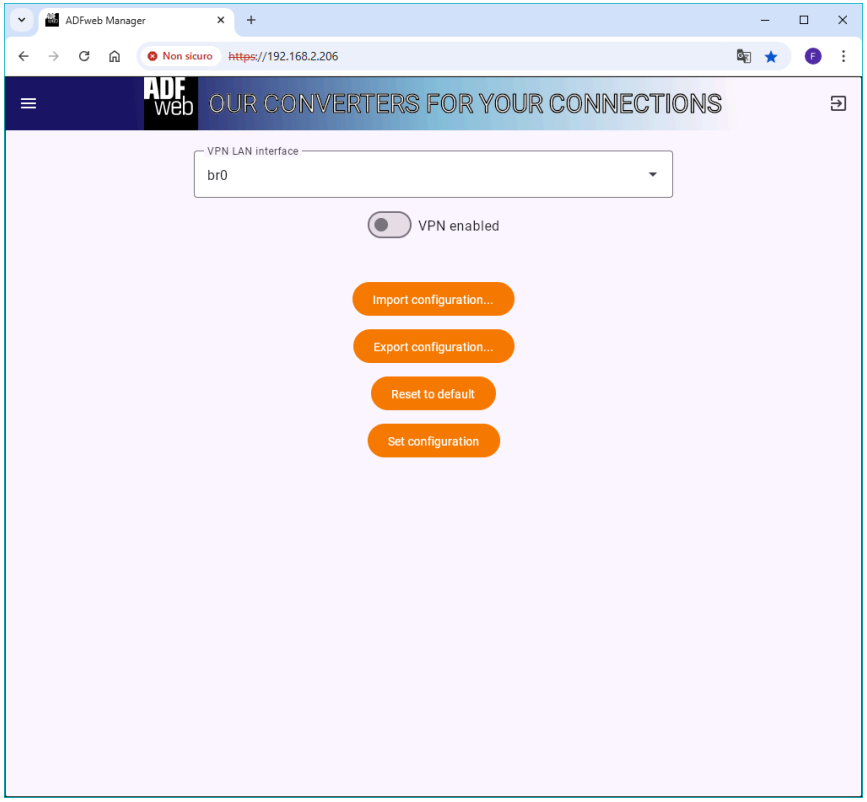


Figure 16: "VPN" page

VPN service is provided by https://cloud.movpn.eu/.

SANDBOX DEVELOPER INSTRUCTIONS

SSH

Add the private key generated from the device's webserver (Linux example):

- copy ssh_key to ~/.ssh/
- **chmod 600 ~/.ssh/ssh_key**
- **eval "$(ssh-agent -s)"**
- **ssh-add ~/.ssh/ssh_key**
- **ssh sandbox@192.168.2.206**

It is strongly suggested to use a more sophisticated software like Filezilla, WinSCP, PuTTY.

To ensure data integrity when writing to the file system, it is recommended to send the "sync" command to flush the pending changes to the flash memory. This may reduce the life of the flash memory, so it is recommended to use it just in case of critical operations (like data logging or installation of services) or when the power supply could be cut.

INIT

The sandbox has the capability of running an init script at the container start, if enabled from the Webserver → Sandbox configuration. The init script/program needs to be placed in /home/sandbox/init and should be set as executable via **chmod +x /home/sandbox/init**.

Example:
#!/bin/sh
echo 1 > /sandbox/leds/led2

C / C++ SDK

DOWNLOAD AND INSTALLATION

The SDK can be installed only on Linux computers. Optionally, it is possible to use a Virtual Machine or Windows WSL.

The SDK is available for download at this address:
https://www.adfweb.com/download/filefold/lin/adf-am335x/sdk/adf-x86_64-armv7at2hf-neon-toolchain.sh

There are software components within the SDK that are licensed as GPLv3. These software components are for development purposes only, and are intended to be removed before installing the application(s) code in your final product, if your product is meant to be redistributed.

Run the downloaded file to start the installation process.

SDK USAGE

To use the SDK from the command line it is possible to configure the environment with the following command:
source $INSTALLATION_DIR/environment-setup-armv7at2hf-neon-oe-linux-gnueabi
This command sets the standard environment variables used by build systems.

To use the compiler the partial command is the following:
$INSTALLATION_DIR/sysroots/x86_64-arago-linux/usr/bin/arm-oe-linux-gnueabi/arm-oe-linux-gnueabi-gcc -march=armv7-a -mfpu=neon -mfloat-abi=hard --sysroot="$INSTALLATION_DIR/sysroots/armv7at2hf-neon-oe-linux-gnueabi"
Or the equivalent shorthand **$CC** or **$CXX** for C and C++ respectively.

The " character is not correctly recognized by the terminal, make sure to type it by yourself.

APPLICATION COMPILATION

The example is available for download at the following address: https://github.com/ADFweb/sandbox-c-hello-world.
The following process describes how to manually build the example application:

- **git clone https://github.com/ADFweb/sandbox-c-hello-world**
- **mkdir build**
- **cmake -DCMAKE_BUILD_TYPE=Debug -DCMAKE_C_COMPILER=$INSTALLATION_DIR/sysroots/x86_64-arago-linux/usr/bin/arm-oe-linux-gnueabi/arm-oe-linux-gnueabi-**

gcc '-DCMAKE_CXX_COMPILER=$INSTALLATION_DIR/sysroots/x86_64-arago-linux/usr/bin/arm-oe-linux-gnueabi/arm-oe-linux-gnueabi-g++' -DCMAKE_SYSROOT=$INSTALLATION_DIR/sysroots/armv7at2hf-neon-oe-linux-gnueabi '-DCMAKE_C_FLAGS=-march=armv7-a -mthumb -mfpu=neon -mfloat-abi=hard' '-DCMAKE_CXX_FLAGS=-march=armv7-a -mthumb -mfpu=neon -mfloat-abi=hard' -B build

- **cmake --build build**
- The resulting executable will be in the build directory.

It is strongly recommended to use an IDE which supports CMake to automate the process.

DEBUGGING

The debugger executable is available for download at the following address: https://www.adfweb.com/download/filefold/lin/adf-am335x/sdk/gdbserver. It is possible to copy it to the device via SSH if necessary (remember to set it as executable with chmod).

It is strongly recommended to setup an IDE that supports GDB remote debugging and SSH file transfer to automate the process.
Manual steps:

- copy the executable on the device via SSH, for example in /home/sandbox/application;
- set the executable execute permission: **chmod +x /home/sandbox/application**;
- start GDB server: **/home/sandbox/gdbserver :1234 /home/sandbox/application**;
- run GDB client on the computer **$INSTALLATION_DIR/sysroots/x86_64-arago-linux/usr/bin/arm-oe-linux-gnueabi/arm-oe-linux-gnueabi-gdb application**;
- on the GDB console: **target remote 192.168.2.206:1234**.

PYTHON

The unit comes with Python 3.10.13 preinstalled.

PACKAGE INSTALLATION

The sandbox environment provides the PIP package manager. It is possible to use it with the command pip3 from the SSH command line, provided the device has an Internet connection. Otherwise it is possible to copy the packages manually via SSH or SFTP.

DEBUGGING

PyCharm

Tested with PyCharm Professional 2024.1.1.
PyCharm debugging first time setup:

- Click on the bottom right Python interpreter;
- Click on Add Interpreter, on SSH;
- Click New on SSH connection;
- In the Host field type the IP address;
- In the Username field type sandbox;
- Choose Key pair;
- Select the private key downloaded from the webserver;
- Click Next and Create to complete the procedure.

PyCharm debugging:

- Click the debug symbol on the top right.

C#

Install .NET Core in the unit:

- Go to https://dotnet.microsoft.com/it-it/download/dotnet/8.0;
- Download the binary Runtime .NET for Linux Arm32;
- Extract the archive and copy it to the sandbox via SSH;
- chmod +x dotnet.

DEBUGGING

Add the following lines to the application, to wait for the debugger to attach. Not needed for Visual Studio Code.

```
Console.WriteLine("Waiting for debugger to attach");
while (!Debugger.IsAttached) {
    Thread.Sleep(100);
}
Console.WriteLine("Debugger attached");
```

These lines must be commented or removed when the debugging is terminated, otherwise the application will not run.

Visual Studio

Tested with Visual Studio 17.10.

.NET Core visual studio debugging first time setup:

- generate the keys in the webserver and convert the private key from OpenSSH format to OpenSSL format:
  - **chmod 600 ssh_key**
  - **ssh-keygen -f ssh_key -p -m pem** (warning: it will replace the file)
- otherwise generate SSH keys and set the public key in the sandbox webserver section:
  - **ssh-keygen -m pem -t ecdsa -f vs_id_rsa**
  - Copy the content of the generate public key (.pub file);
  - Paste it in the webserver and set the configuration.
- in visual studio you can go to Debug > Options > Cross platform >Connection management > Add > Enter the information and the key;
- Run the following commands on the device (make sure you have Internet connection since it will take some time and data ~66MB to download the debugger):
  - **mkdir ~/.vs-debugger**
  - **wget https://aka.ms/getvsdbgsh -O ~/.vs-debugger/GetVsDbg.sh**
  - **chmod a+x ~/.vs-debugger/GetVsDbg.sh**
  - **~/.vs-debugger/GetVsDbg.sh -r linux-arm -v latest -l ~/.vs-debugger/vscode**

.NET Core visual studio debugging:

- move *PROJECT_NAME*.dll, *PROJECT_NAME*.pdb, *PROJECT_NAME*.runtimeconfig.json, *PROJECT_NAME*.deps.json and the dependencies via SSH;
- Run the following commands on the device:
  - **dotnet *PROJECT_NAME*.dll**

- Connect remotely with the following steps (the first time it will take some time and data ~66MB to download the debugger):
  - Click Debug in the top menu bar;
  - Connect to process…;
  - Connection Type: SSH;
  - Select the device from the dropdown menu;
  - Select the dotnet process;
  - Click Connect;
  - Choose Managed (.NET Core for UNIX).

Reference links:

- https://learn.microsoft.com/en-us/visualstudio/debugger/attach-to-running-processes-with-the-visual-studio-debugger?view=vs-2022#BKMK_Attach_to_a_process_on_a_remote_computer
- https://learn.microsoft.com/en-us/visualstudio/debugger/remote-debugging-dotnet-core-linux-with-ssh?view=vs-2022#attach-the-debugger

Visual Studio Code

- Make sure you have the C# Dev Kit extension installed.
- Run the following commands on the device (make sure you have Internet connection):
  - **mkdir ~/.vs-debugger**
  - **wget https://aka.ms/getvsdbgsh -O ~/.vs-debugger/GetVsDbg.sh**
  - **chmod a+x ~/.vs-debugger/GetVsDbg.sh**
  - **~/.vs-debugger/GetVsDbg.sh -r linux-arm -v latest -l ~/.vs-debugger/vscode**
- Prepare the tasks.json and launch.json in the .vscode folder of your project:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "build",
            "command": "dotnet",
            "type": "process",
```

```
            "args": [
                "build",
                "${workspaceFolder}/sandbox-csharp-hello-world/sandbox-csharp-hello-world.csproj"
            ],
            "problemMatcher": "$msCompile"
        },
        {
            "label": "transfer",
            "dependsOn": "build",
            "presentation": {
                "reveal": "always",
                "panel": "new"
            },
            "type": "process",
            "command": "scp",
            "args": [
                "-i",
                "C:/Users/david/Downloads/ssh_key",
                "-v",
                "-r",
                "./sandbox-csharp-hello-world/bin/Debug/net8.0/*",
                "sandbox@192.168.2.206:/home/sandbox/"
            ],
            "problemMatcher": []
        }
    ]
}
```

```
{
    "version": "2.0.0",
    "configurations": [
        {
            "name": ".NET Core Launch (remote console)",
            "type": "coreclr",
            "request": "launch",
            "preLaunchTask": "transfer",
            "program": "dotnet",
            "args": ["/home/sandbox/sandbox-csharp-hello-world.dll"],
            "cwd": "/home/sandbox",
            "stopAtEntry": true,
            "console": "internalConsole",
            "pipeTransport": {
                "pipeCwd": "${workspaceFolder}/sandbox-csharp-hello-world",
                "pipeProgram": "ssh",
                "pipeArgs": [
                    "-T",
                    "-i", "C:/Users/david/Downloads/ssh_key",
                    "sandbox@192.168.2.206"
                ],
                "debuggerPath": "~/.vs-debugger/vscode/vsdbg"
            }
        }
    ]
```

```
    }

```

Rider

Tested with Rider 2024.1.
.NET Core Rider debugging first time setup:

- Click on run options (3 dots in the top right);
- Attach to Remote Process…;
- Manage connections;
- Add the device IP address, username (sandbox) and private key in putty format.

.NET Core Rider debugging:

- Move *PROJECT_NAME*.dll, *PROJECT_NAME*.pdb, *PROJECT_NAME*.runtimeconfig.json, *PROJECT_NAME*.deps.json and the dependencies via SSH;
- Run the following commands on the device:
  - **dotnet *PROJECT_NAME*.dll**
- Connect remotely (the first time will take some time to copy the debugger ~360MB):
  - Run options (3 dots in the top right corner);
  - Attach to Remote Process…;
  - Select the device;
  - Choose the process;
  - Attach with .NET Debugger.

Reference links:

- https://www.jetbrains.com/help/rider/SSH_Remote_Debugging.html#i3xg9pg_33

Rust

On the development machine make sure to have the compiler installed:

- rustup install armv7-unknown-linux-gnueabihf;
- sudo apt install gcc-arm-linux-gnueabihf;
- create the file .cargo/config.toml inside the project with the following content:

```
[target.armv7-unknown-linux-gnueabihf]
rustflags = ["-C", "target-feature=+crt-static", "-C", "linker=arm-linux-gnueabihf-gcc"]
```

- cargo build --target=armv7-unknown-linux-gnueabihf

Cron

To enable cron in the sandbox container, a specific sequence of commands needs to be used:

- mkdir /home/sandbox/.crontabs
- crontab -e
- Then write your crontab file in the editor (example: * * * * * date >> /home/sandbox/test.txt)
- Leave the editor
- chmod 444 /home/sandbox/.crontabs/sandbox
- crond

Sandbox Files

Specific functionalities are exposed to the sandbox user via files depending on the hardware and software capabilities of the device:

- /sandbox/leds/led*: reading the led* files will return 0 if the LED is OFF or 1 if the LED is ON. Writing 1 to the file will turn ON the LED, viceversa writing 0 will turn it OFF;
- /sandbox/watchdog: the watchdog file controls a virtual watchdog that only resets the sandbox application. It provides an interface compatible with the Linux watchdog (https://www.kernel.org/doc/html/v5.9/watchdog/watchdog-api.html). The sample application provides a usage example. If the watchdog timer is triggered, the sandbox process tree will be terminated and the init process will be restarted if it was configured to do so (see configuration section);
- /sandbox/board: hardware board identifier;
- /sandbox/product: product code;
- /sandbox/serial: QN number.

Field bus files

If the sandbox has a field bus extensions, like ProfiNet Slave, the data will be available at the following files:

- /sandbox/converters/*converter_name*/bank0: allows reading the data from the field bus;
- /sandbox/converters/*converter_name*/bank1: allows writing data to the field bus.

The two files are in binary format and can be seeked to read/write the desired bytes.
Any programming language can be used to read/write bytes by using standard library functions.
Example to read from the shell: *hexdump -C /sandbox/converters/MQTT_CLIENT/bank0*
Example to write 0x31c0 from the shell: *printf '\x31\xc0' | dd of=/sandbox/converters/COMM_PROTO/bank1 bs=1 seek=0 count=2 conv= notrunc*

Sandbox HTTP endpoints

All the endpoints use basic authentication with the same username and password of the webserver UI.

GET https://ipaddress/api/state

Provides information about the device, including the allowed interface combinations.

```
{
   uptime: 123456,
   version: "1.0.0",
   model: "HD67xxx-G43",
   serial_number: "123456789",
   features: {
       interface_combinations: {
           "eth0-eth1,usb0": {
               name: "Mobile Internet",
               interfaces: {
                   "eth0-eth1": {
                       name: "Ethernet",
                       type: "ethernet",
                       if_name: "br0",
                       address_methods: ["dhcp", "static"]
                   },
                   "usb0": {
                       name: "Cellular",
                       type: "cellular",
                       if_name: "wwan0",
                       address_methods: ["dhcp"],
                       cellular: {
                           sim: [{
                               type: "card"
                           }, {
                               type: "esim"
                           }]
                       }
                   }
               }
           },
```

```
            "eth0,eth1": {
                name: "Separate Ethernet interfaces",
                interfaces: {
                    "eth0": {
                        name: "Ethernet 0",
                        type: "ethernet",
                        if_name: "eth0",
                        address_methods: ["dhcp", "static"]
                    },
                    "eth1": {
                        name: "Ethernet 1",
                        type: "ethernet",
                        if_name: "eth1",
                        address_methods: ["dhcp", "static"]
                    }
                }
            },
            "eth0-eth1": {
                name: "Single Ethernet interface",
                interfaces: {
                    "eth0-eth1": {
                        name: "Ethernet",
                        type: "ethernet",
                        if_name: "br0",
                        address_methods: ["dhcp", "static"]
                    }
                }
            }
        },
        functions: {
            "sandbox": {
                name: "Sandbox"
            },
            "vpn": {
                name: "Vpn"
            }
        }
    }
}
```

GET and POST https://ipaddress/api/configuration/network

Allows getting the current network configuration and setting a new one with the GET and POST requests respectively.


```
{
   interface_combination: {
       name: "Mobile Internet",
       interfaces: {
           "eth0-eth1": {
               name: "Ethernet",
               type: "ethernet",
               dhcp: false,
               dns: "8.8.8.8",
               gateway: "192.168.2.1",
               gateway_metric: 10,
               hostname: "device",
               ip_addresses: [{
                   ip: "192.168.2.206",
                   sm: "255.255.255.0"
               }],
               ip_if_no_lease: {
                   ip: "192.168.2.206",
                   sm: "255.255.255.0"
               },

               if_name: "br0"
           },
           "usb0": {
               name: "Cellular",
               type: "cellular",
               if_name: "usb0",
               ip_addresses: [],
               dhcp: true,
               cellular: {
                   active_sim: 0,
                   sim: [{
                       type: "card",
                       apn: "iliad",
                       pin: "1234",
```

```
                        username: "",
                        password: "",
                    }],
                    network_mode: "all",
                },
                gateway_metric: 5
            }
        }
    },
    ntp_server: "pool.ntp.org"
}
```

JSON

FIELD NAME	FORMAT	VALUES
interface_combination	object	See "Interface_combination" object section
ntp_server	string	Valid URL or IP Address

"Interface_combination" object

NAME	FORMAT	VALUES
name	string	Valid values: *Separate Ethernet interfaces* = Ethernet ports isolated each other *Single Ethernet interface* = Ethernet ports bridged each other *Mobile Internet* = LTE connection enabled
interfaces	JSON map	See "Interfaces"object section

"Interfaces" object

Valid keys (the value of the map is described in the next table):

- "eth0": settings for Ethernet 0 port (only if Interface_combination → name is "Separate Ethernet interfaces");
- "eth1": settings for Ethernet 1 port (only if Interface_combination → name is "Separate Ethernet interfaces");
- "usb0": settings for LTE modem (only if Interface_combination → name is "Mobile Internet");
- "eth0-eth1": settings for bridged ports (only if Interface_combination → name is "Single Ethernet interface" or "Mobile Internet").

NAME	FORMAT	VALUES
name	string	Generic text
type	string	Valid values: *ethernet* = Ethernet connection *cellular* = Mobile connection
dhcp	bool	True = Activated False = Deactivated
dns	string	Valid IP Address. Optional.
gateway	string	Valid IP Address. Optional.
gateway_metric	Int	Integer number > 0
hostname	string	Generic text (max 30 chars). Optional, works only if DHCP is True
ip_addresses	array of objects	See "ip_addresses" object section. Can be empty if DHCP is True
ip_if_no_lease	object	See "ip_if_no_lease" object section. Mandatory if DHCP is True
if_name	string	Valid values: *eth0* = valid for key "eth0" *eth1* = valid for key "eth1" *br0* = valid for key "eth0-eth1" *usb0* = valid for key "usb0"
cellular	object	See "cellular" object section. Mandatory if setting usb0 mobile network interface. If present, DHCP must be True.

"ip_addresses" object

NAME	FORMAT	VALUES
ip	string	Valid IP Address
sm	string	Valid SubNet Mask

"ip_if_no_lease" object

NAME	FORMAT	VALUES
ip	string	Valid IP Address
sm	string	Valid SubNet Mask

"cellular" object

NAME	FORMAT	VALUES
active_sim	Int	Index of SIM card defined into "sim" array
sim	array of objects	See "sim" object section
network_mode	string	Valid values: *all* = network type automatically selected *lte_only* = mobile connection on LTE only

"sim" object

NAME	FORMAT	VALUES
type	string	Valid values: *card* = external SIM card; *esim* = internal eSIM, if available.

apn	string	Generic text
pin	string	Text with numeric chars
username	string	Generic text
password	string	Generic text

GET https://ipaddress/api/state/cellular

Get the status of the LTE module.

Response body:

```
{
    is_registered: true,
    is_connected: true,
    imei: "0123456789",
    iccid: "0123456789",
    signal: {
        rssi: 25,
        quality: 99
    }
}
```

JSON

FIELD NAME	FORMAT	VALUES
is_registered	bool	The modem is registered to the cellular network
is_connected	bool	Data connection is open
imei	string	IMEI number of the modem
iccid	string	ICCID number of the selected SIM card
signal	object	

"signal" object

FIELD NAME	FORMAT	VALUES
rssi	int	Strength of the radio signal: 0-31
ber	int	Bit error rate: 0-7 or 99 if undetectable

GET and POST https://ipaddress/api/configuration/mqttBroker

Set and get the MQTT broker's configuration (for MQB version).

```
{
    listeners: [{
        port: 1883,
        bind_interface: null,
        cert_file: "",
        key_file: "",
        client_ca_file: "",
        anonymous_user_acl: {
            topic_list: [{
                permission: "readwrite",
                topic: "#"
            },
            {
                …
            }]
        },
        user_list: [{
            username: "federico",
            password: "federico",
            acl: {
                topic_list: [{
                    permission: "read",
                    topic: "test"
                },
                {
                    …
                }]
            }
        },
        {
            …
        }]
    }],
    persistent_client_expiration_hours: 1000,
    bridges: [{
```

```
        address: "test.mosquitto.org",
        port: 1883,
        bridge_protocol_version: "mqttv311",
        cleansession: true,
        keepalive_interval: 30,
        remote_clientid: "remoto",
        remote_username: "remoto",
        remote_password: "remoto",
        bridge_outgoing_retain: false,
        bridge_insecure: false,
        bridge_cafile: "",
        bridge_certfile: "",
        bridge_keyfile: "",
        topic_list: [{
            description: "test",
            topic: "",
            direction: "both",
            qos: "0",
            local_prefix: "test",
            remote_prefix: "testremoto"
        },
        {
            …
        }]
    }]
}
```

JSON

FIELD NAME	FORMAT	VALUES
listeners	array	See "listeners" object section
persistent_client_expiration_hours	Int	Number of hours after which the Client is removed if it does not reconnect
bridges	array	See "bridges" object section

"listeners" object

NAME	FORMAT	VALUES
port	Uint	Valid number of TCP port
bind_interface	string	Optional, if not present bind to all interfaces, otherwise the string representing the interface to bind to (i.e. "eth0")
cert_file	string	Broker's certificate, optional
key_file	string	Broker's private key, optional
client_ca_file	string	CA certificate for the clients, optional
anonymous_user_acl	object	See "anonymous_user_acl" object section
user_list	array	See "user_list" object section

"anonymous_user_acl" object

NAME	FORMAT	VALUES
topic_list	array	See "topic_list" object section

"topic_list" object

NAME	FORMAT	VALUES
permission	string	Valid values: *read* = Topic can be only read *write* = Topic can be only written readwrite = Topic can be read and written deny = Topic is blocked
topic	string	Topic's name

"user_list" object

NAME	FORMAT	VALUES
username	string	Client's username
password	string	Client's password
acl	object	See "acl" object section

"acl" object

NAME	FORMAT	VALUES
topic_list	object	See "topic_list" object section

"bridges" object

NAME	FORMAT	VALUES
address	string	Url or valid IP Address
port	Uint	Valid number of TCP port
bridge_protocol_version	string	Valid values: *mqttv31* = version 3.1 *mqttv311* = version 3.1.1 *mqttv50* = version 5.0
cleansession	bool	Enable or disable "Clean session" option
keepalive_interval	int	Time in seconds for Keep Alive
remote_clientid	string	Bridge's Client ID
remote_username	string	Bridge's username to connect to the remote broker, optional

remote_password	string	Bridge's password to connect to the remote broker, optional
bridge_outgoing_retain	bool	Enable or disable "Retain" option
bridge_insecure	bool	Enable or disable "Insecure connection" option
bridge_cafile	string	Bridge's CA certificate, optional
bridge_certfile	string	Bridge's certificate, optional
bridge_keyfile	string	Bridge's private key, optional
topic_list	object	See "bridge_topic_list" object section

"bridge topic_list" object

NAME	FORMAT	VALUES
description	string	Description for the topic
topic	string	Fixed to ""
direction	string	Valid values: *in* = Topic can be only received from remote broker *out* = Topic can be only transmitted to remote broker both = Topic can be received and transmitted from/to remote broker
qos	string	QoS level. Allowed values "0", "1" or "2"
local_prefix	string	Topic's name in the local broker
remote_prefix	string	Topic's name in the remote broker

GET and POST https://ipaddress/api/configuration/vpn

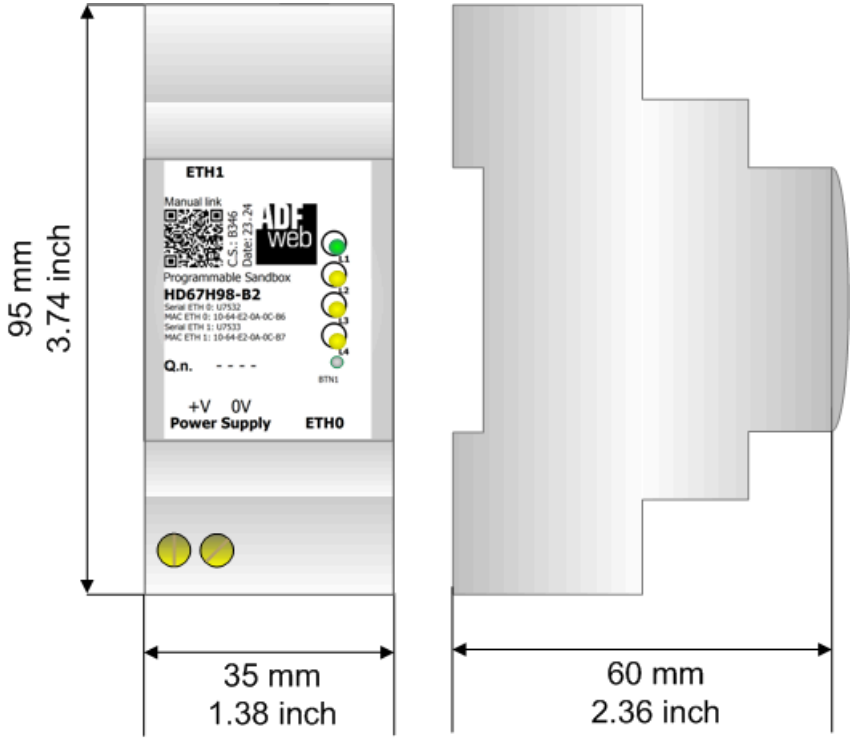
Allows getting the current VPN configuration and setting a new one with the GET and POST requests respectively.

```
{
   vpn_enabled: false,
   vpn_lan_interface: "br0"
}
```

JSON

FIELD NAME	FORMAT	VALUES
vpn_enabled	bool	true = VPN enabled false = VPN disabled
vpn_lan_interface	string	Valid values: *eth0* = VPN tunnel on Ethernet 0 port (only if Interface_combinations → name is "Separate Ethernet interfaces"); *eth1* = VPN tunnel on Ethernet 1 port (only if Interface_combinations → name is "Separate Ethernet interfaces"); *br0* = VPN tunnel on bridged ports (only if Interface_combinations → name is "Single Ethernet interface" or "Mobile Internet").

MECHANICAL DIMENSIONS



Housing: PVC
Weight: 200g
(Approx)

Figure 17: Mechanical Dimensions for HD67H98-xxx-B2

ORDERING INFORMATIONS

The ordering part number is formed by a valid combination of the following:

ACCESSORIES

Order Code: **AC34011** **-** 35mm Rail DIN - Power Supply 220/240V AC 50/60Hz – 12 V DC

Order Code: **AC34012** **-** 35mm Rail DIN - Power Supply 220/240V AC 50/60Hz – 24 V DC

DISCLAIMER

All technical content within this document can be modified without notice. The content of the document is a under continual renewal.
For losses due to fire, earthquake, third party access or other accidents, or intentional or accidental abuse, misuse, or use under abnormal conditions repairs are charged to the user. ADFweb.com S.r.l. will not be liable for accidental loss of use or inability to use this product, such as loss of business income. ADFweb.com S.r.l. shall not be liable for consequences of improper use.

OTHER REGULATIONS AND STANDARDS

WEEE INFORMATION

Disposal of old electrical and electronic equipment (as in the European Union and other European countries with separate collection systems).

This symbol on the product or on its packaging indicates that this product may not be treated as household rubbish. Instead, it should be taken to an applicable collection point for the recycling of electrical and electronic equipment. If the product is disposed correctly, you will help prevent potential negative environmental factors and impact of human health, which could otherwise be caused by inappropriate disposal. The recycling of materials will help to conserve natural resources. For more information about recycling this product, please contact your local city office, your household waste disposal service or the shop where you purchased the product.

RESTRICTION OF HAZARDOUS SUBSTANCES DIRECTIVE

The device respects the 2002/95/EC Directive on the restriction of the use of certain hazardous substances in electrical and electronic equipment (commonly referred to as Restriction of Hazardous Substances Directive or RoHS).

CE MARKING

The product conforms with the essential requirements of the applicable EC directives.

WARRANTIES AND TECHNICAL SUPPORT

For fast and easy technical support for your ADFweb.com SRL products, consult our internet support at www.adfweb.com.
Otherwise contact us at the address support@adfweb.com

RETURN POLICY

If while using your product you have any problem and you wish to exchange or repair it, please do the following:

- Obtain a Product Return Number (PRN) from our internet support at www.adfweb.com. Together with the request, you need to provide detailed information about the problem.
- Send the product to the address provided with the PRN, having prepaid the shipping costs (shipment costs billed to us will not be accepted).

If the product is within the warranty of twelve months, it will be repaired or exchanged and returned within three weeks. If the product is no longer under warranty, you will receive a repair estimate.

ADFweb.com S.r.l.
Via Strada Nuova, 17
IT-31010 Mareno di Piave
TREVISO (Italy)
Phone +39.0438.30.91.31
Fax +39.0438.49.20.99
www.adfweb.com