ELSETA

WCC Lite

User manual

Elseta

2020/04/07

Doc version: 1.4.0

HW version: 1.2

FW version: 1.4.0

COPYRIGHTS AND TRADEMARKS

SAFETY PRECAUTIONS

Any work related to the installation, configuration, commissioning and maintenance of the WCC Lite should be carried out by qualified personnel only. It is implied the person or group responsible for the said duties have adequate engineering knowledge.

It is crucial to adhere to laws and regulations of the jurisdiction the WCC Lite is being installed at. This product should not be implemented or resold to install in high-security areas such as: nuclear power plants, aircraft navigation, military equipment, transport traffic management or in other areas where equipment failure or malfunction can result in hazardous, life-threatening consequences of human injury or harm to environment.

This product is NOT safe to use in an explosive atmosphere.

Any installation or wiring should be carried out with the equipment fully powered off.

In order to prevent damage to the equipment, please carefully check the power supply, input and output ratings, as well as the operating conditions. Failure to observe this information may render input, output, or the whole device inoperable and may also void warranty.

SYMBOLS USED

Danger - critical notice which may affect the safety of the user or device.

Attention - notice on possible problems that may arise in individual cases.

Information notice - tips or other information.

WARRANTY

UAB Elseta has the right to terminate the warranty maintenance:

- If the WCC Lite components obtained mechanical damage.
- If the WCC Lite was disassembled not as described in service and installation manual.
- If the WCC Lite failure was due to deliberate or inadvertent user's fault.
- If the WCC Lite broke due to natural disasters.

Table of Contents

1 Overview **11**

2 Hardware and software requirements **11**

3 Technical information **12**

4 WCC Lite status indication and control **14**

4.1 Status indication 14

4.2 Reset button 14

5 Installing the WCC Lite **15**

5.1 Mounting 15

5.2 Power 15

5.3 SIM card slot 15

5.4 Dual-SIM card slot 16

6 Interfaces **17**

6.1 Serial port interfaces 17

6.2 Relay output 20

6.3 GSM 20

6.4 WiFi 22

7 Tags **23**

7.1 Single point 23

7.2 Double point 23

8 Internal web page **24**

8.1 Initial setup 24

8.1.1 Static IP address setup on Windows 24

8.1.2 Connecting to an internal web page 27

8.2 Site layout 28

8.3 Protocol HUB 29

8.3.1 Manage devices 29
8.3.2 Port settings 34
8.3.3 Import 35
8.3.4 Signals 35
8.3.5 Sequence of Events (SOE) 36
8.3.6 Imported signals 37
8.4 Status 38
8.4.1 Overview 38
8.4.2 Firewall 41
8.4.3 Routes 43
8.4.4 System Log 44
8.4.5 Kernel Log 44
8.4.6 Processes 45
8.4.7 Realtime graph 45
8.4.8 GSM status 48
8.4.9 VNSTAT Traffic monitor 50
8.5 System 52
8.5.1 System 52
8.5.2 Administration 54
8.5.3 Software 56
8.5.4 Startup 57
8.5.5 Scheduled tasks 57
8.5.6 Mount points 58
8.5.7 LED configuration 59
8.5.8 Backup/flash firmware 60
8.5.9 Reboot 62
8.6 Services 63
8.6.1 Telemetry agent 63
8.6.2 IPsec 63
8.6.3 L2TP/IPsec 68
8.6.4 OpenVPN 68

8.6.5 ser2net . . . 69
8.7 Network . . . 70
8.7.1 Interfaces . . . 70
8.7.2 Wireless . . . 75
8.7.3 DHCP and DNS . . . 77
8.7.4 Hostnames . . . 80
8.7.5 Static routes . . . 80
8.7.6 Firewall . . . 80
8.7.7 Diagnostics . . . 84
8.7.8 GSM . . . 85
8.7.9 Layer 2 Tunneling Protocol . . . 88
8.8 Logout . . . 90

9 API 91

10 SNMP 92

11 DNP3 93
11.1 DNP3 Master . . . 93
11.2 DNP3 Slave . . . 94

12 DLMS 95
12.1 Overview . . . 95
12.2 Configuration . . . 95
12.2.1 Devices section . . . 95
12.2.2 Signals section . . . 96

13 Modbus 97
13.1 Modbus Master . . . 97
13.1.1 Configuring datapoints . . . 97
13.1.2 Debugging a Modbus Master application . . . 100
13.2 Modbus Slave . . . 100
13.2.1 Configuring datapoints . . . 100

13.2.2 Mapping values to registers . . . 101
13.2.3 Debugging a Modbus Slave application . . . 102

14 IEC 60870-5 103

14.1 IEC 60870-5-103 Master . . . 103
14.1.1 Configuring datapoints . . . 103
14.1.2 Debugging a IEC 60870-5-103 Master aplication . . . 105
14.2 IEC 60870-5-104 . . . 105
14.2.1 Slave . . . 105

15 IEC 62056-21 106

15.1 Overview . . . 106
15.2 Configuration . . . 106
15.2.1 Devices section . . . 106
15.2.2 Signals section . . . 106

16 WCC Lite internal signals 107

16.1 Overview . . . 107
16.2 Configuration . . . 107
16.2.1 Devices section . . . 107
16.2.2 Signals section . . . 107

17 Excel configuration 109

17.1 Devices sheet . . . 109
17.1.1 Optional settings . . . 110
17.1.2 Serial port settings . . . 110
17.1.3 TCP/IP settings . . . 111
17.1.4 Protocol specific settings . . . 111
17.2 Signals sheet . . . 117
17.2.1 Required attributes . . . 117
17.2.2 Optional attributes . . . 117
17.2.3 Signal recalculation operation priority . . . 119
17.2.4 number_type field . . . 120

17.2.5 Protocol specific settings . . . 121
17.2.6 Linking signals . . . 124
17.2.7 Mathematical expressions . . . 124
17.3 Uploading configuration . . . 127
17.3.1 Importing an Excel file . . . 127
17.3.2 Generating .zip file . . . 128
17.3.3 Uploading configuration remotely . . . 128

18 Programmable logic controller 129
18.1 IEC 61499 . . . 129
18.1.1 4Diac framework . . . 130
18.1.2 Example project . . . 131
18.1.3 Configuring data endpoints . . . 135
18.1.4 Debugging an IEC 61499 application . . . 136
18.1.5 Generating and uploading FORTE logic file . . . 138
18.1.6 Distributed control application . . . 139

19 MQTT 143
19.1 Overview . . . 143
19.2 Using WCC Lite as MQTT Client . . . 143
19.3 MQTT data format . . . 145

20 Certificates 146

21 Cyber security 148
21.1 User rights . . . 148
21.1.1 User management and rights authentication . . . 148
21.1.2 Locally stored credentials management . . . 148
21.1.3 Authentication via external service . . . 150
21.1.4 Audit Log . . . 152
21.1.5 Secure your device's access . . . 152
21.2 Groups rights . . . 153
21.2.1 SSH Access . . . 153

21.2.2 Web interface permissions 153
21.3 Conformance to IEC 62351 standard 155

22 Changelog **157**

23 Information about the equipment manufacturer **159**

1 Overview

This document is intended to act as a user manual and explain WCC Lite usage in detail.

It is expected the person referring to this manual is experienced in programmable logic controllers (PLC), networking (IPv4, ethernet) and the use of the operating system of choice (Windows, Linux, Mac, etc.).

This document might not cover all of the use cases. For usage not described in this document please contact Elseta technical support (contact info available on the last page of this document).

2 Hardware and software requirements

In order to get the WCC Liteup and running, a PC/Mac is required, capable of running a web browser and an MS Excel compatible spreadsheet editor (e.g. LibreOffice or an online spreadsheet editor such as Google Sheets). A built-in or external Ethernet adapter is also required to connect to the WCC Lite.

3 Technical information

System	
Processor	ARM CPU (AR9331, 400MHz)
Memory	16 MB Flash/64 MB DDR2 RAM
Wireless	802.11 b/g/n
I/O	1x Relay output
	1x Binary input
Additional storage	SD card (2GB by default)
Ethernet	10/100 Base-T - RJ-45 connector up to 2 independent ports
Serial ports	1x RS-485 1x RS-485 / RS-232 (switchable)
Time synchronization	NTP client + server, IEC 60870-5-101, IEC 60870-5-104
GSM	2G(GPRS, EDGE) / 4G(LTE) 2G(GPRS, EDGE) / 3G(UMTS, HSDPA, HSUPA) 2G(GPRS, EDGE) / 3G(UMTS, HSDPA, HSUPA) / 4G(LTE) Single OR Dual SIM card modem
Power requirements	
Power supply	12 - 24 VDC
Power consumption	< 6W
Mechanical	
Dimensions	101 (H) x 22.5 (W) x 119 (L), mm
Mounting	Wall mount, Din rail
Environmental	
Operating temperature	-40°C to +85°C
Warranty	2 year
Software	
Compatibility with HMI (Human Machine Interface)	Compatible with cloud based SCADA system - CloudIndustries.eu
Routing	• Isolated LAN interface • Isolated LAN interface, but omitted to provide TCP / UDP ports or VPN mergers • Routed LAN internet connection masking data for GSM interfaces • Secure LAN data transfer via VPN • Secure LAN data transmission through VPN access to the Internet • Single OR Dual SIM card modem
Database	• File based database • Data buffering in case of network outage
Data security	• All data between WCC Lite and Cloud based SCADA exchange over secure encrypted VPN tunnel • Firewall to prevent intrusion and DoS attacks • VPN solution with VPN gateway can be used to manage (configure and update) and monitor VPN and WCC devices from a single place

Device maintenance	It is possible to configure and monitor devices and protocols connected to the WCC Lite through Elseta cloud based SCADA system CloudIndustries.eu or 3rd party SCADAs and see device-based alarms such as communication errors, etc.
Supported protocols	• Modbus master / slave (RTU / ASCII / TCP) • M-Bus master (Serial / TCP) • IEC 60870-5- 101 master / slave • IEC 60870-5- 103 master • IEC 60870-5- 104 master / slave • IEC 62056-31 master • IEC 62056-21 (since v1.2.13) • DNP3 master / slave • SMA Net • DLMS (since v1.3.0) • Resol VBus
Supported devices	• Other Elseta products • Aurora PV inverters • Delta inverters • Kaco PV inverters • SMA PV inverters • Ginlong PV inverters • Solplus PV inverters • Kostal devices • Windlog data logger • Vestas Wind turbines • Elgama elektronika electricity meters
Network features	• IPsec • OpenVPN • xl2tp • Firewall • Routing • RADIUS • SNMP • ser2net • API • NTP synchronization
Extra features	• Software update • Remote configuration via CloudIndustries.eu administration • Device fault notifications • Internal web page for configuration and diagnostics

4 WCC Lite status indication and control

4.1 Status indication

Name	Description
STATUS	On (green) System is powered on blinking (blue) Blinking in heartbeat pattern. Blinking is more intense when CPU load is high. On (red) Fault (if configured) Off System is not powered
RO	On (green) Relay is activated (COM connected to NO) Off Relay is deactivated (COM connected to NC)
RX/TX 1	On (yellow) PORT1 data TX On (green) PORT1 data RX Off No activity on PORT1
RX/TX 2	On (yellow) PORT2 data TX On (green) PORT2 data RX Off No activity on PORT2
ETH 0	On (green) ETH0 LINK/Activity Off ETH0 not linked/inactive
ETH 1	On (green) ETH1 LINK/Activity Off ETH1 not linked/inactive
WLAN	On (blue) Wi-fi enabled Blinking (blue) Connected Off Wi-fi disabled
GSM	Blinking (yellow) Registered to network Off Not registered to network On (green) GSM signal level is over -65dBm On (green) GSM signal level is over -85dBm On (green) GSM signal level is lover than -85dBm

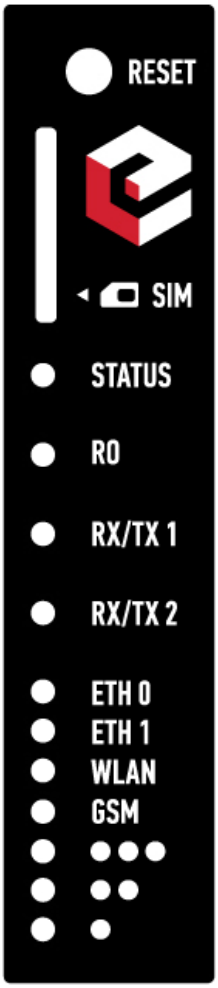


Figure 1: Front panel status indication

4.2 Reset button

The reset button is located on the front panel of WCC Lite, to access it, remove a transparent front panel cover. Different time lengths of button pressing call different behaviour.

Table 2: Possible Reset button behaviour

Pressing time	Description	Indication
Short Press	System reboot	Red STATUS LED starts blinking
Long press (>3s)	Reset to factory settings	Red STATUS LED turns on

5 Installing the WCC Lite

5.1 Mounting

To mount the device:

1. Secure the top of the mounting clip onto a DIN rail.
2. Push the bottom of a device forward to fix the clip in place.

To dismount the device:

1. Pull red coloured clip downwards (found at the bottom side of the DIN rail).
2. Pull back the bottom of the device.
3. Pull device upwards to dismount it.

Figure 2: WCC Lite DIN rail mounting clip

5.2 Power

WCC Lite It is recommended to power WCC Lite from 6W (minimum) 12-24V DC power supply. A full range is 5V to 36V.

Note: Make sure that device is compatible with your power source before proceeding! Check the label next to power connector or on the side of device.

5.3 SIM card slot

WCC Lite has push-push type microSIM card connector with card detection function. The connector is located on the front panel of WCC Lite. To access it, remove a transparent front panel cover. To insert a SIM card gently push it inside (see Figure below) until it locks in place. Press again to release and remove the card.

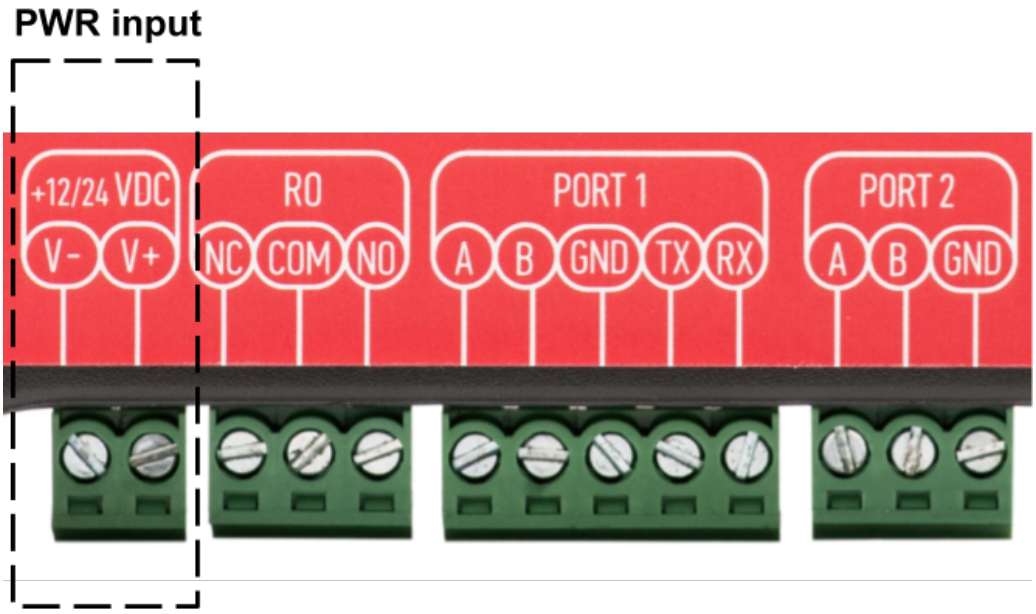


Figure 3: Power input connector

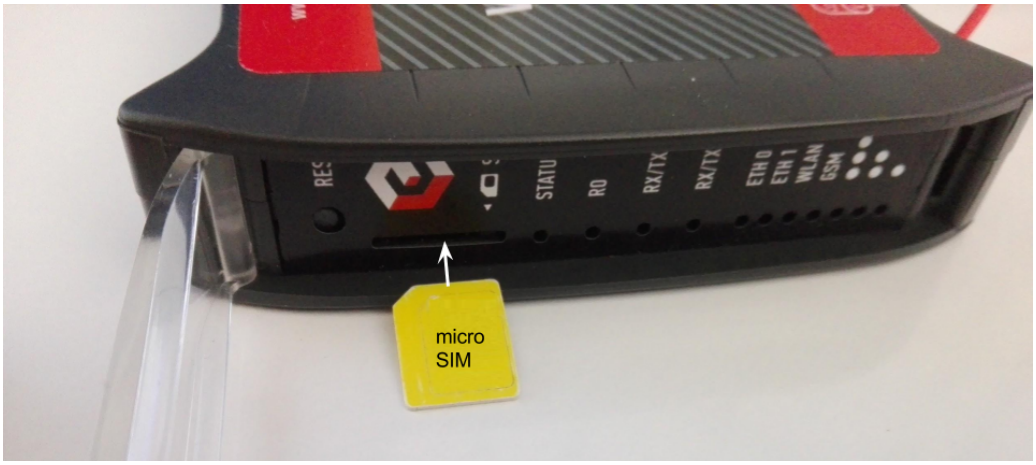


Figure 4: WCC Lite SIM card slot

5.4 Dual-SIM card slot

WCC Lite has optional Dual-SIM card modem. To access both SIM cards, remove a transparent front panel cover and press through marked hole with small tool until SIM holder pops out.

To insert SIM cards, remove Dual-SIM holder and fit SIM cards into it. Insert holder with SIM cards into slot.

Note: Be careful when removing or inserting DUAL-SIM holder, as SIM cards can fall out.

Note: WCC Lite will automatically detect a SIM card insertion or removing.

Figure 5: WCC Lite Dual-SIM card slot

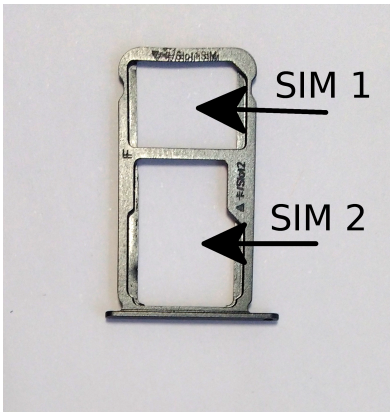


Figure 6: WCC Lite Dual-SIM card holder

6 Interfaces

6.1 Serial port interfaces

WCC Lite WCC Lite has 2 serial ports (Figure 7). Selectable RS485 (by default) or RS232 interface on PORT1 and RS485 interface on PORT2.

WCC Lite RS485 interface supports baud rates up to 115200 and has an integrated 120Ω termination resistor. It is recommended to use termination at each end of the RS485 cable. To reduce reflections, keep the stubs (cable distance from main RS485 bus line) as short as possible when connecting device. See typical RS485 connection diagram on figure 8.

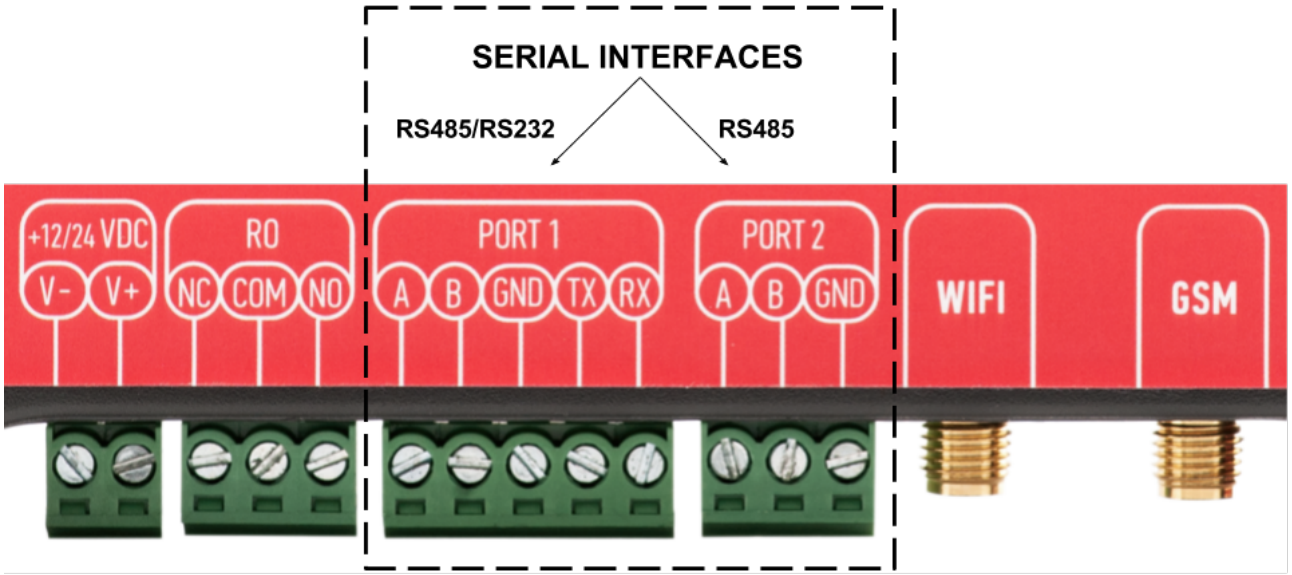


Figure 7: WCC Lite ports

Note: Double check if A and B wires are not mixed up.

WCC Lite 3-wire RS232 interface is available on PORT1 and can be selected by user (see Port settings). Baud rates up to 115200 are supported. See typical RS232 connection diagram on figure 9.

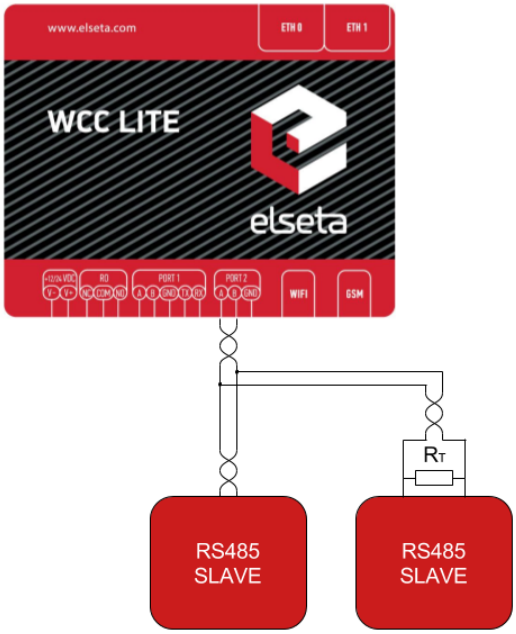


Figure 8: Typical WCC Lite RS-485 connection diagram

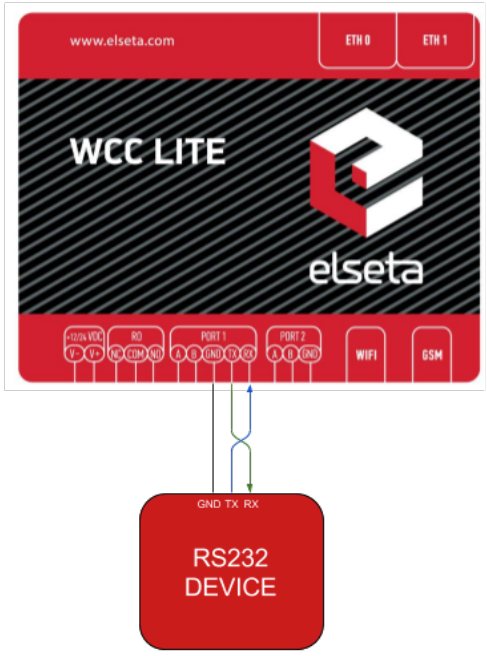


Figure 9: Typical WCC Lite RS-232 connection diagram

6.2 Relay output

WCC Lite integrates 1 signal relay (3-way RO connector) with COM (common), NC (normally closed) and NO (normally open) signals.

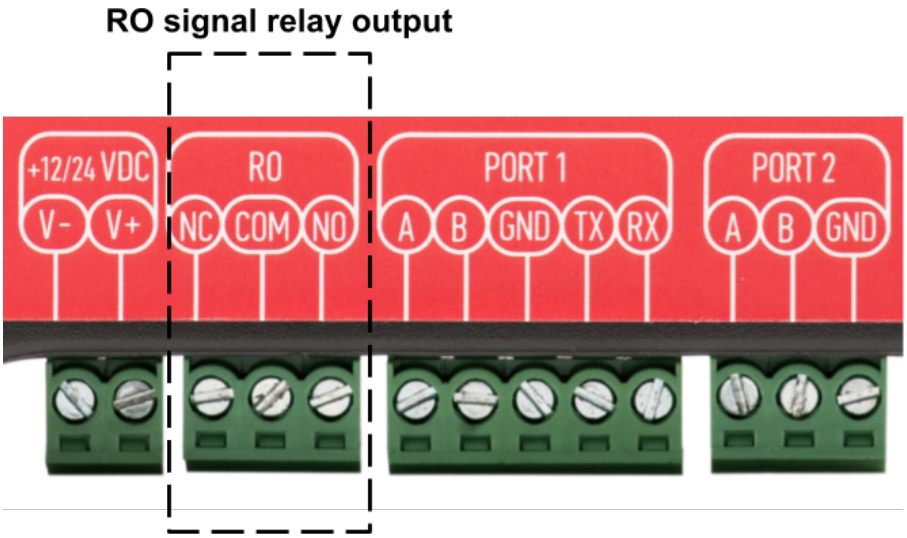


Figure 10: Signal relay connector

Maximum switching power is 60W, maximum contact current is 2A, maximum switching voltage is 60VDC/60VAC. The lower is switching power, the higher is lifecycle of RO.
Relay electrical endurance:

- resistive load, 30VDC / 1A - 30W min. $1x10^5$ operations;
- resistive load, 30VDC / 2A - 60W min. $1x10^4$ operations.

6.3 GSM

WCC Lite comes with an optional GSM module.

There are few hardware configurations available:

- Without GSM modem.
- With single SIM modem (HW version 1.0 - 1.2) - 2G/3G (GPRS, EDGE / UMTS, HSDPA, HSUPA) version - 5.76Mb/s upload, 7.2Mb/s download. UMTS/HSPA bands 900, 2100. GSM bands 900, 1800. Modem chip - Ublox Sara-U270.
- With single SIM modem (HW version 1.0 - 1.2) - 2G/4G (GPRS, EDGE / LTE) Cat 1 version - 10.3Mb/s upload, 5.2Mb/s download. LTE bands 3, 7, 20. GSM bands 900, 1800. Modem chip - Ublox Lara-R211.
- With dual SIM modem (HW version 1.0 - 1.2) - 2G/3G (GPRS, EDGE / UMTS, HSDPA, HSUPA) version - 5.76Mb/s upload, 7.2Mb/s download. UMTS/HSPA bands 900, 2100. GSM bands 900, 1800. Modem chip - Ublox Sara-U270.

- With dual SIM modem (HW version 1.0 - 1.2) - 2G/4G (GPRS, EDGE / LTE) Cat 1 version - 10.3Mb/s upload, 5.2Mb/s download. LTE bands 3, 7, 20. GSM bands 900, 1800. Modem chip - Ublox Lara-R211.
- With dual SIM modem (HW version 1.3 - 1.4) - 2G/3G/4G (GPRS, EDGE / UMTS, HSDPA, HSUPA / LTE) Cat 4 version - 50Mb/s (max) upload, 150Mb/s (max) download. LTE bands 1, 3, 5, 7, 8, 20, 38, 40, 41. GSM bands 3, 8. UMTS bands 1, 5, 8. Modem chip - Quectel EC25-E.

They are based on mini PCI-e standard connector and compatible with any other devices. Check label on package for current modification.

Connect an antenna to the SMA connector labeled “GSM“. Select a good antenna placement spot considering the operation environment and network coverage of your mobile provider in the area. Make sure the signal level is over -80dBm to have a stable connection to the network.

4G (LTE) Cat 1 version modem both antennas are used for LTE communication. In such case internal WIFI antenna is used. Network can be limited in distance and speed, especially in metal based panels.

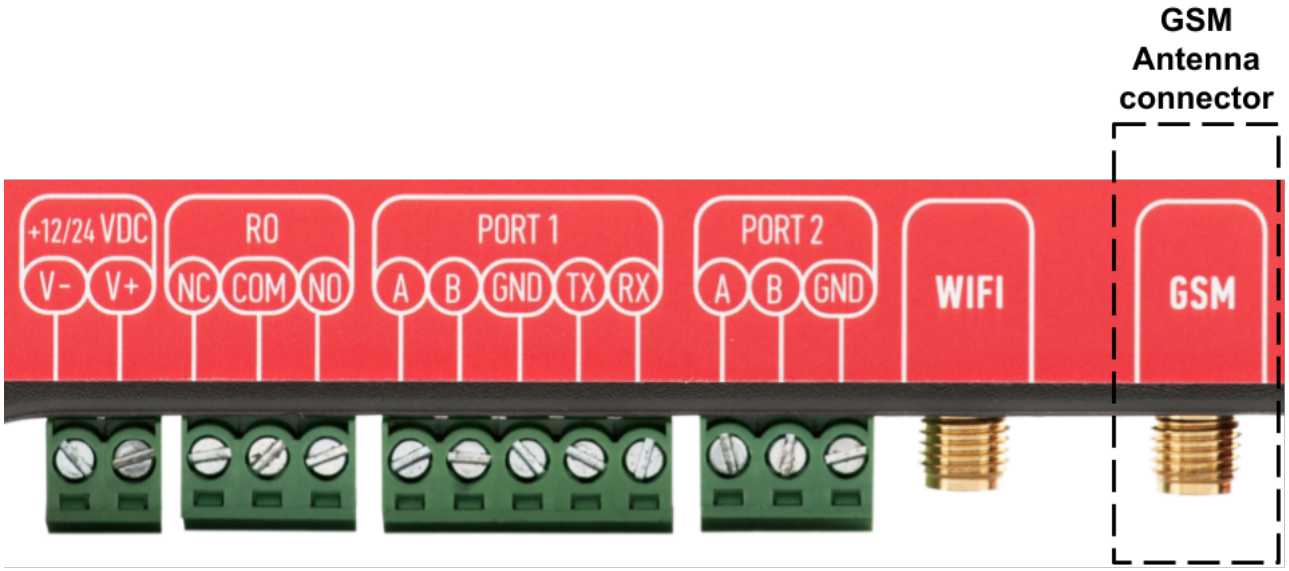


Figure 11: GSM antenna connector

6.4 WiFi

In case a Wi-fi connection is needed, connect a Wi-fi antenna to the SMA connector labeled “WIFI“. Select a good antenna placement spot considering the operation environment. Make sure the signal level is over -80dBm to have a stable connection to the network.

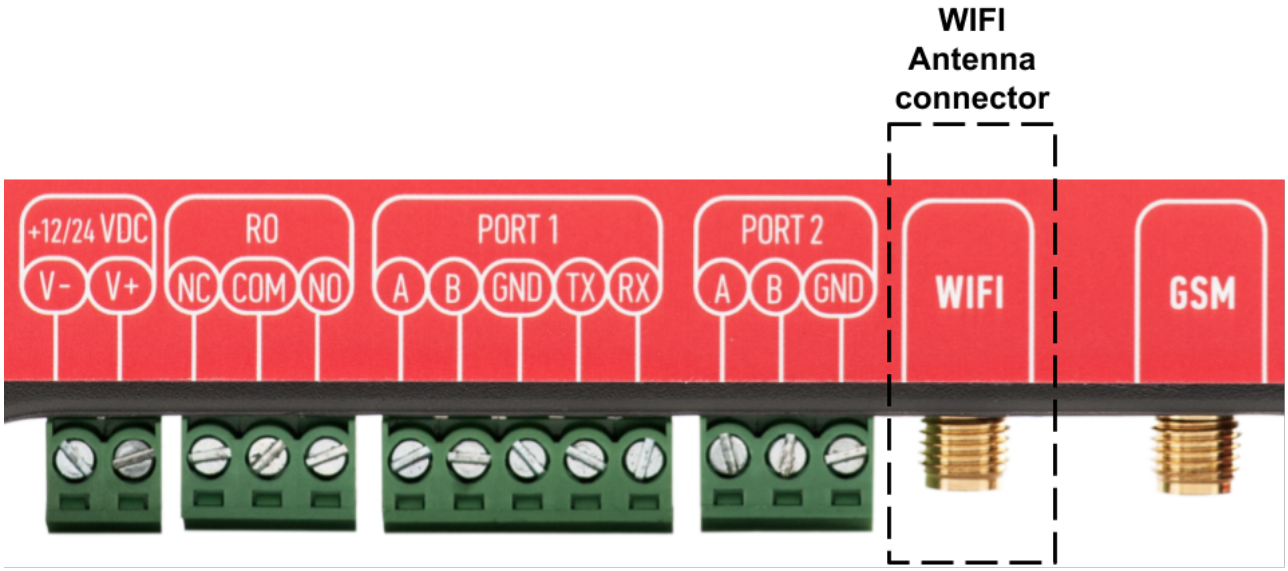


Figure 12: Wi-Fi antenna connector

7 Tags

7.1 Single point

Commonly used in storing digital states single point values have only one bit of information. The value of such tags can be either *one* or *zero*.

On the internal web of *WCC Lite* states of this type of tags are shown in colored boxes with customisable label.

Value	Representation
0	OFF
1	ON

7.2 Double point

Double point signals contain two bits of information that allow four different states, therefore they contain additional information compared to single point ones. INDETERMINATE state might, for example, mean that part of the equipment has been turned off or a mechanical part which does the switching has stuck between states. ERROR state might mean that both contacts are connected and there might be a short circuit in the equipment.

Value	Representation
00	INDETERMINATE
01	OFF
10	ON
11	ERROR

8 Internal web page

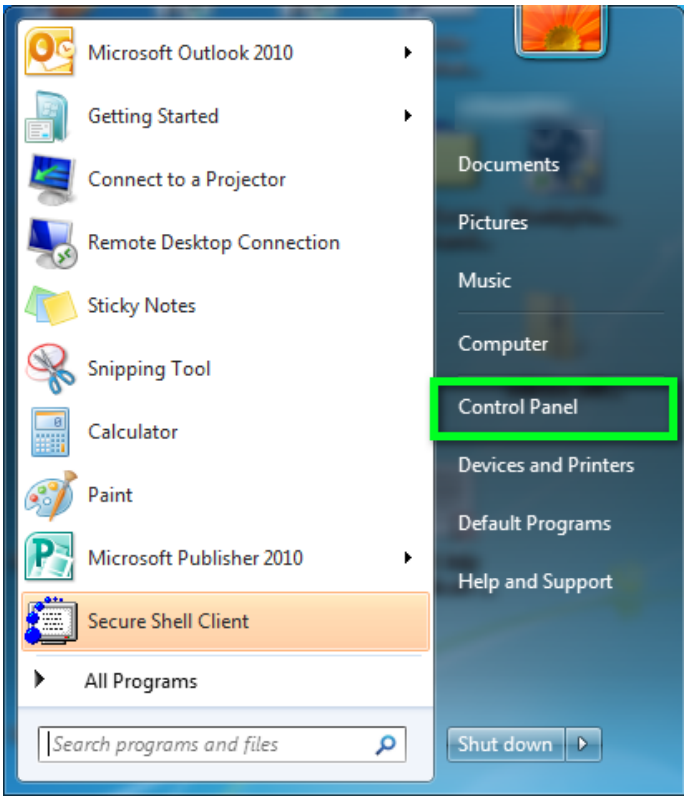
WCC Lite is configured via an internal web browser, so no additional software is required.

8.1 Initial setup

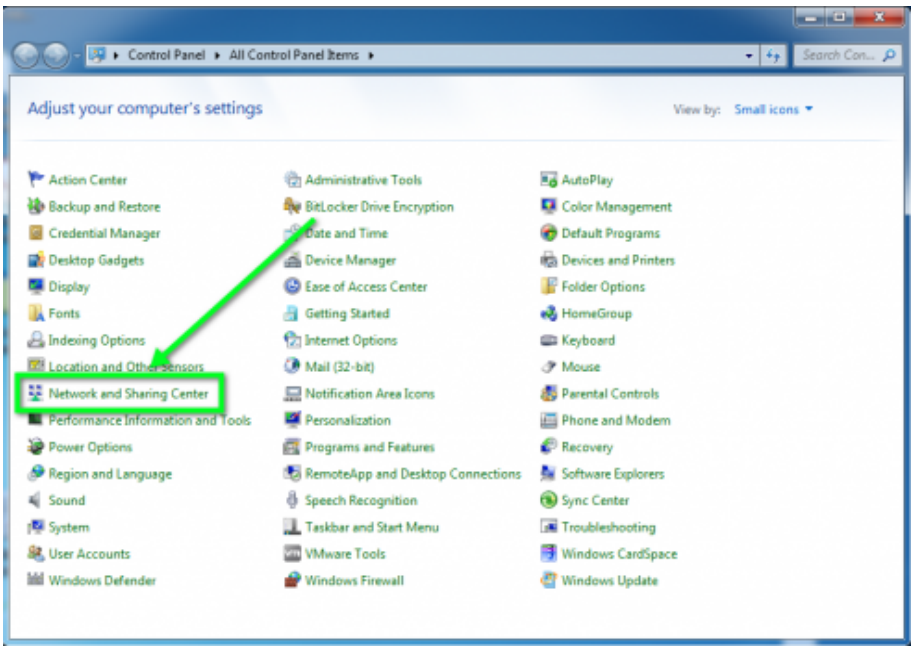
WCC Lite comes with static network configuration with its IP set to *192.168.1.1*. For initial setup set a static IP address on your computer and connect your network card to the *WCC Lite* with an ethernet cable.

8.1.1 Static IP address setup on Windows

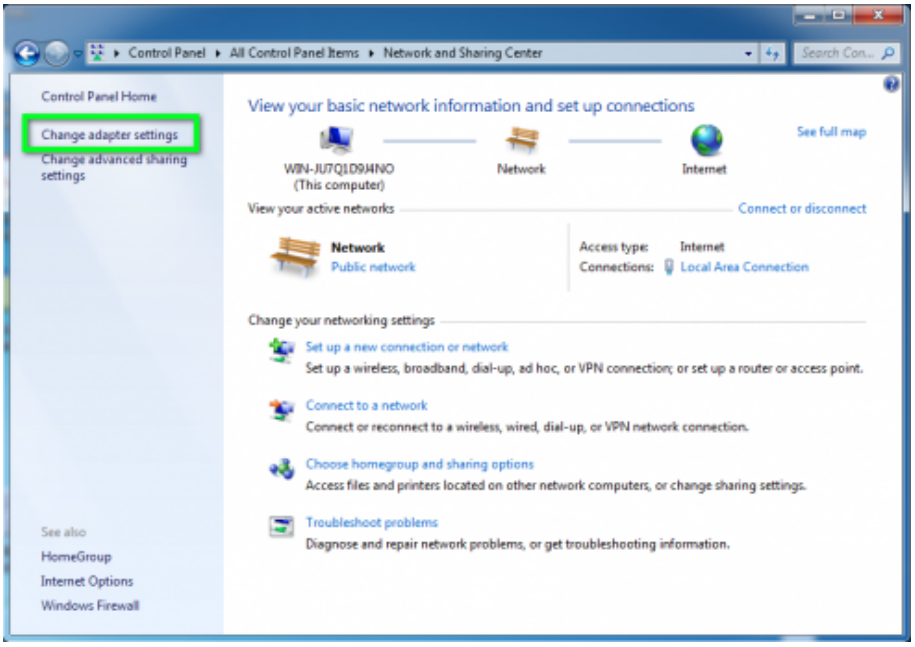
1. Click the Start menu. Next, click on the *Control Panel* option.



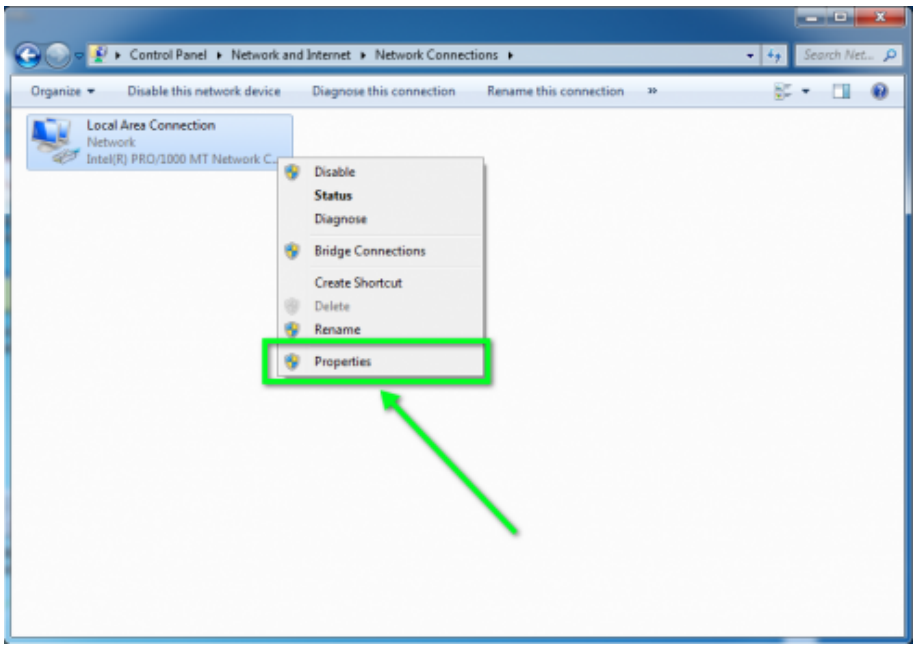
2. Click on the *Network and Sharing Center* option.



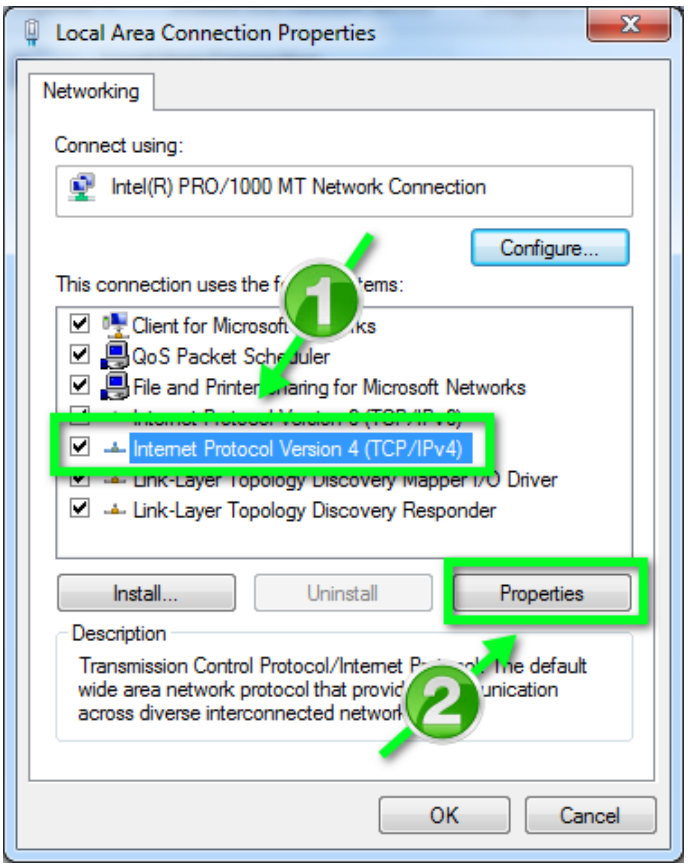
3. Click on *Change adapter settings* from the left side menu.



4. Right-click on the *Local Area Connection* icon, then select *Properties*.

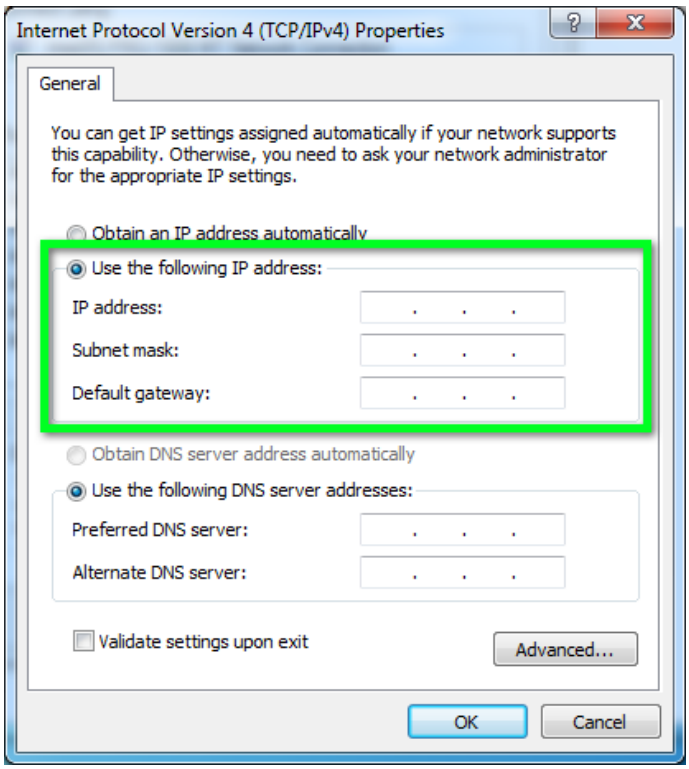


5. In the window that opens, click on the *Internet Protocol Version 4 (TCP/IPv4)* (you may need to scroll down to find it). Next, click on the *Properties* button.



6. In the window that opens, click the *Use the following IP address* radio button. Fill the following fields and click *OK*:

   - *IP address:* 192.168.1.2
   - *Subnet mask:* 255.255.255.0
   - *Default gateway:* (leave empty)



8.1.2 Connecting to an internal web page

If your computer IP address is set up and ethernet cable is connected power up the device. Wait a few minutes until the device boots. Then open your web browser and enter the following url: `http://192.168.1.1/`

Supported web browsers:

- Google Chrome (recommended)
- Mozilla Firefox
- Internet Explorer 8 or later

Log in with the root user:

- *Username:* root
- *Password:* wcclite

It is recommended to change the password immediately to avoid any unauthorized access.

Before plugging *WCC Lite* with a static IP address to the local computer network, make sure to check if such address is not already reserved by other devices.

8.2 Site layout

It provides the main navigation through the website. Contains the following sections:

- *PROTOCOL HUB*: configuration related to data exchange between *WCC Lite* and other devices.
- *STATUS*: system information and diagnostics.
- *SYSTEM*: basic system settings such as time setup.
- *SERVICES*: various other services.
- *NETWORK*: network related settings and services.
- *USERS*: existing user groups and management of their permissions
- *LOGOUT*: user logout.

8.3 Protocol HUB

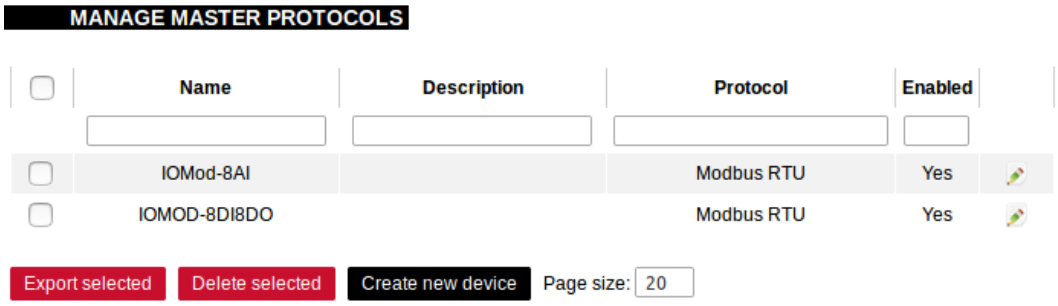
Full Protocol HUB section is only available on “Cloud gateway” firmware type. For “RTU” firmware type refer to Sequence of Events and Imported Signals subsections.

Protocol HUB section stores configuration for every connected device. There are three ways to configure these devices:

1. Manual configuration in *Manage devices* section.
2. Import settings from Excel file.
3. Remote configuration via CloudIndustries.eu.

Any changes made in this section will take effect only after being applied. A notification with apply button will appear after making any changes.

8.3.1 Manage devices



The ”Manage Master Protocols” section displays a list of configured devices. Devices can be edited, removed, added and exported from this window.

This functionality is only available on “Cloud gateway” firmware type.

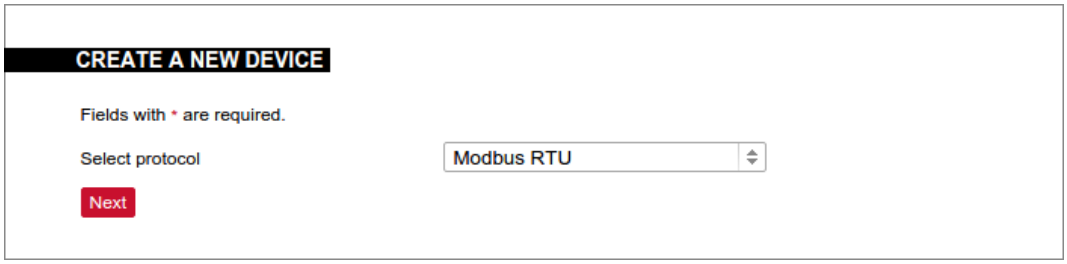
Further device configuration is described below in ”Manual device configuration” section.

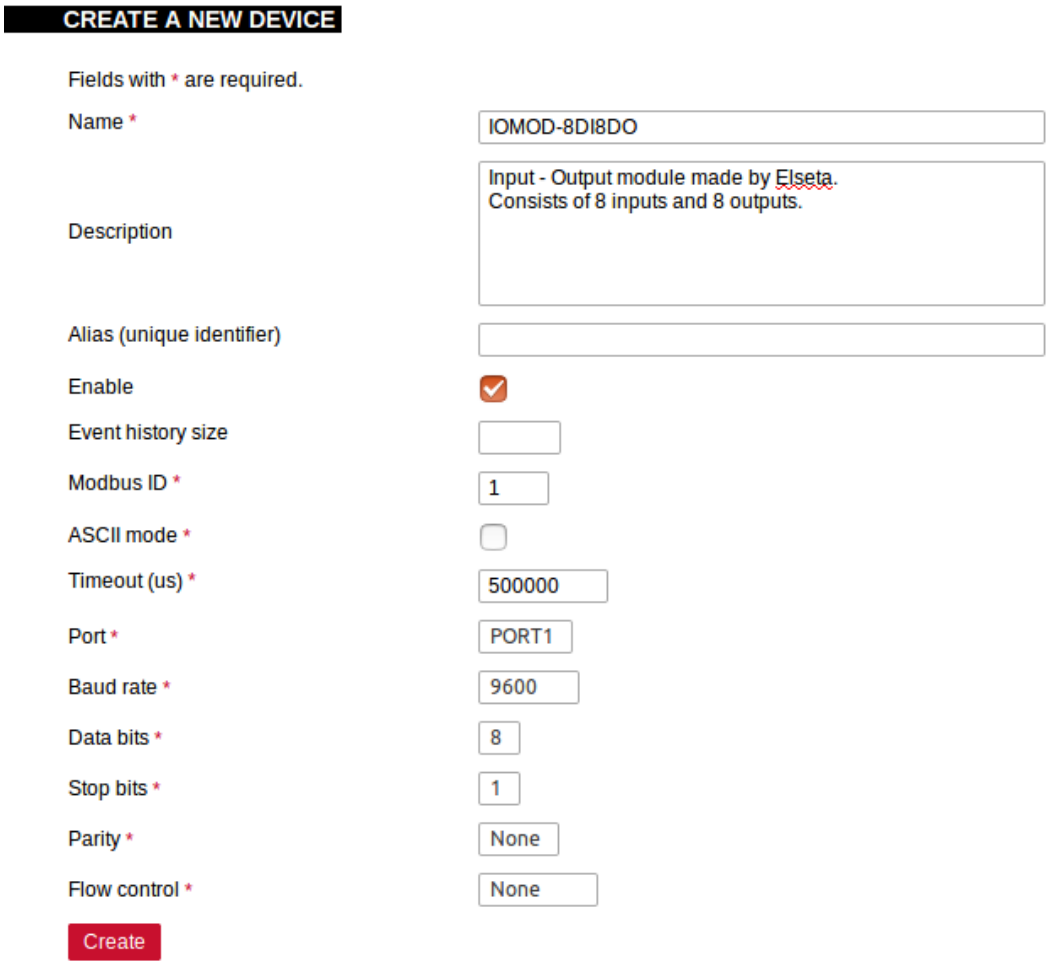
Manual device configuration

Manual configuration allows to create fine tuned device configuration that can later be exported as a template.

Create new device: Device creation is performed by selecting it’s working protocol and configuring it’s name, slave address and communication settings. Existing device configuration is

performed similary.





Name: Device name to be used further in signals and events

Description: Short device description

Alias: An unique alphanumeric string that identifies this device. If alias is left blank, it will be generated automatically.

Enable: If disabled, configuration for this device will be ignored and it's measurements will not be updated

Event history size: Number of device measurements to keep in events history. If this field is left blank, history is disabled.

Modbus ID: Modbus slave device unique identifier.

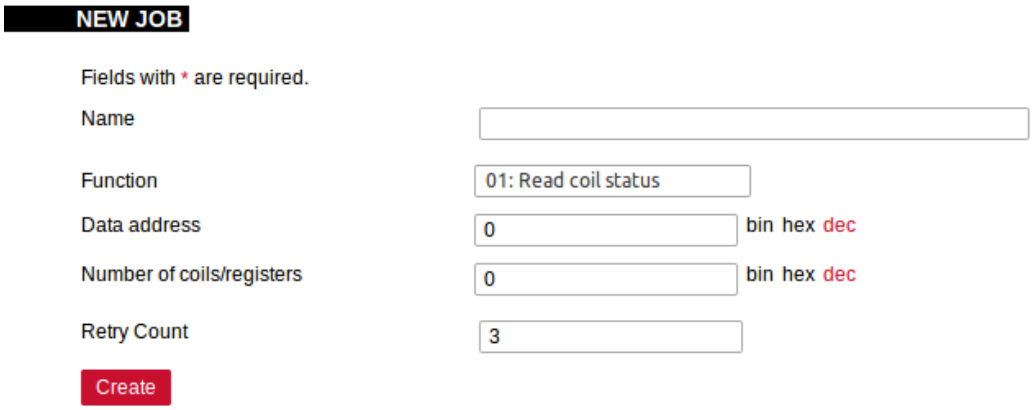
Modbus ASCII mode: Check to use Modbus ASCII mode. If left unchecked, RTU mode is used by default.

Timeout: Time limit to wait for a response from the device.

Port: Select port that device is connected to.

Communication settings: Serial port communication rate; Number of data bits; Number of stop bits; parity mode; flow control.

Manage jobs: A job is a software instruction to communicate with a device and get required data. Further data extraction is done with tags. Job creation and configuration is performed similarly.



Name: Job name

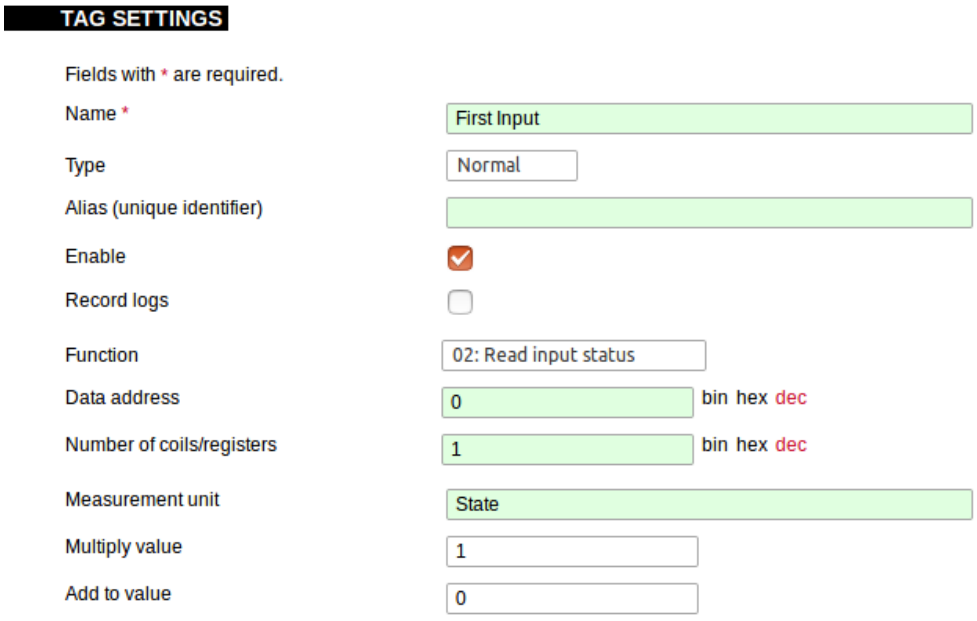
Description: Short device description

Function: A specific instruction to communicate with device. These instruction options are protocol specific

Retry Count: This number indicates the retry limit when communication has failed

When job is created and configuration is applied, WCC Lite immidietly starts sending data requests to configured port. Tag settings needs to be configured for data extraction from job.

Manage tags: Tag is one measurement for a device. Tags contain information how to obtain required values from job data. Tag configuration is divided into two panes - ”Tag settings” and ”Advanced”, the latter dedicated only for experienced users.



Name: Name for one measurement, e.g. ”Temperature” or ”Energy consumption”

Type: Tag type

Alias: An unique alphanumeric string that identifies this device. If alias is left blank, it will be generated automatically.

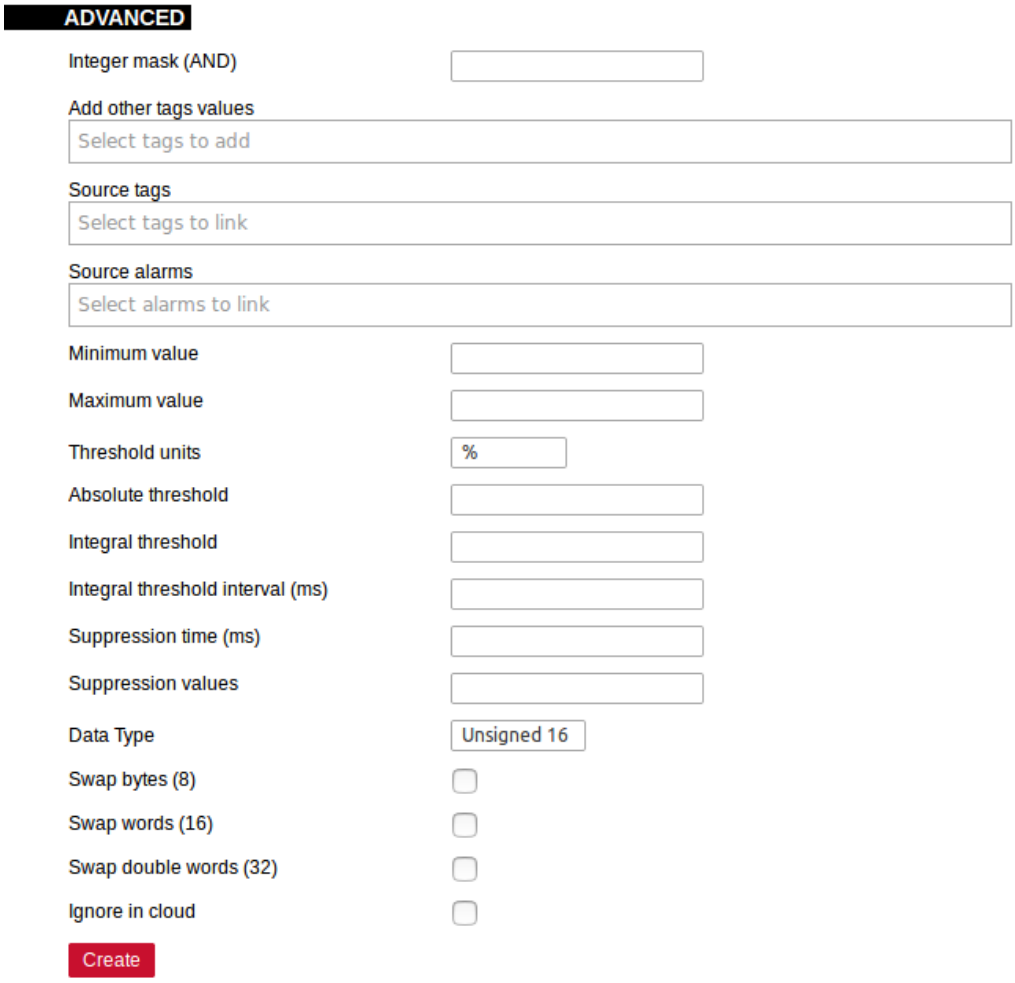
Enable: If disabled, configuration for this tag will be ignored and measurements will not be updated

Function: A specific instruction to communicate with device. These instruction options are protocol specific

Measurement unit: Units to show for this measurement, e.g. V, W or kg

Multiply value: Value to multiply by measurement. Use values below 1 to divide.

Add to value: Adds value to measurement. Use negative values to substract.



Measurement data format and parsing rules can be configured via the ”Advanced” pane.

Name: Name for one measurement, e.g. ”Temperature” or ”Energy consumption”

Integer mask (AND):

Add other tags values:

Source tags:

Source alarms:

Minimum value:

Maximum value:

Threshold units:

Absolute threshold:

Integral threshold:

Integral threshold interval (ms):

Suppression time (ms):

Suppression values:

Data Type: Selects data type e.g. Float, signed / unsigned integer.

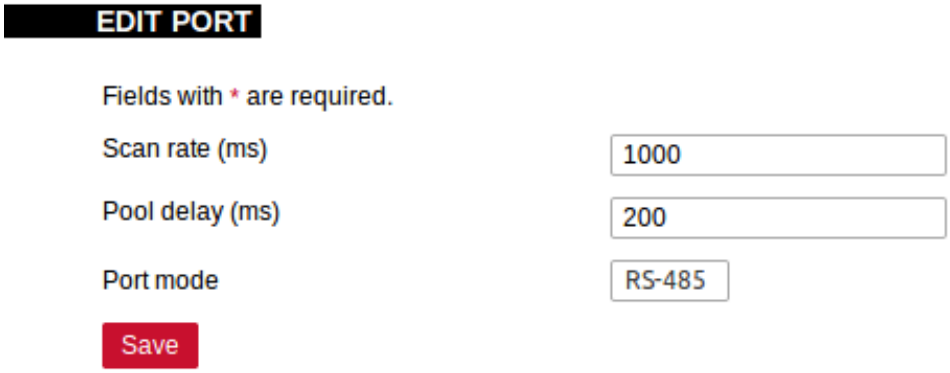
Swap bytes (8): changes byte sequence.

Swap words (16): changes word sequence.

Swap double words (32): changes double word sequence.

Ignore in cloud:

8.3.2 Port settings



These options affect how device data polling is scheduled each port. These settings do not affect *IEC 60870-5* protocols.

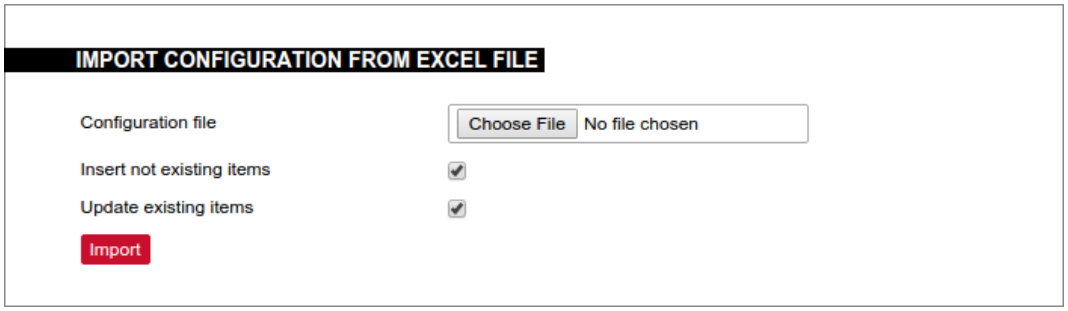
This functionality is only available on “Cloud gateway” firmware type.

Scan rate: Time duration in milliseconds when all jobs on current port should be done. This option directly affects measurement update speed on one port. For example, if this value is set to 10 seconds, every measurement will be updated every 10 seconds if possible.

Poll delay: Minimum time delay in milliseconds to wait before sending any data on port. This is useful when devices fail to respond when data is transmitted too fast.

Port mode: Mode selection for port. WCC Lite has first port selectable between RS-232 and RS-485 interfaces.

8.3.3 Import



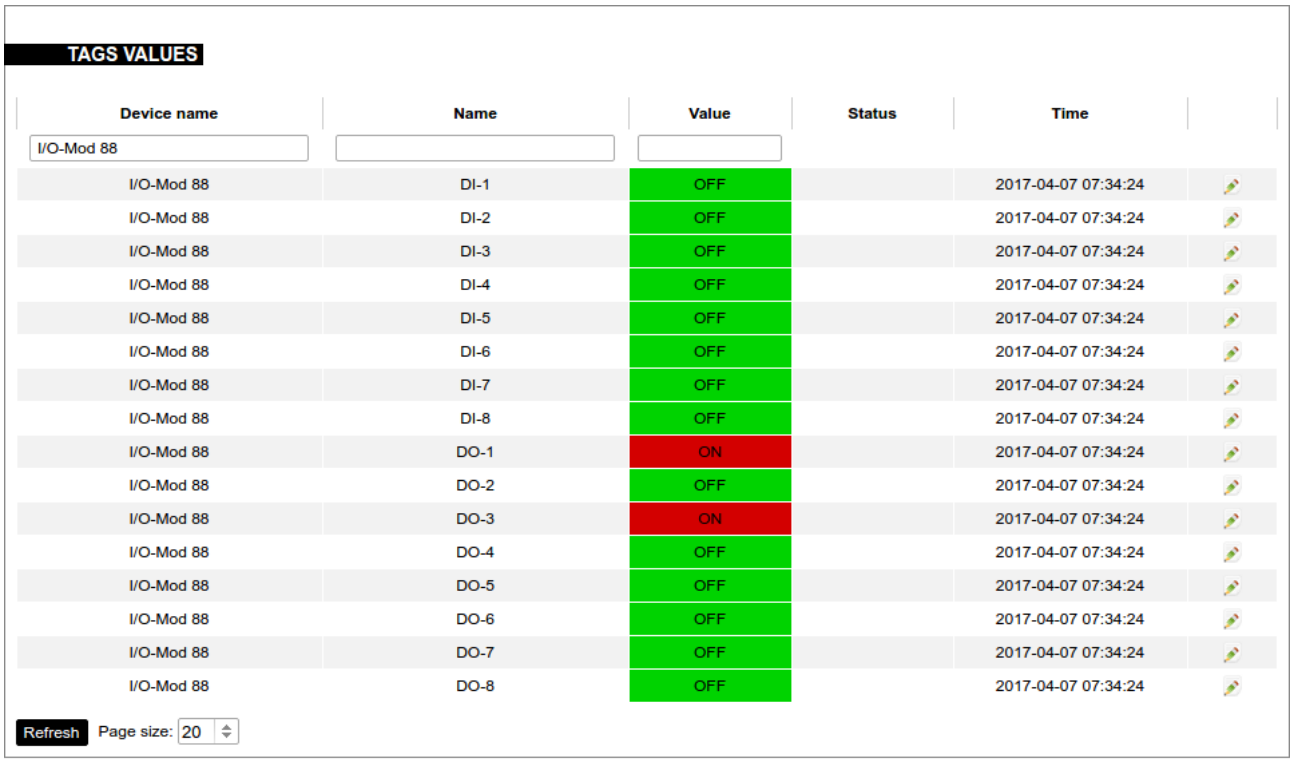
This functionality is only available on “Cloud gateway” firmware type.

Import new configuration from Excel file (*.xls*, *.xlsx* formats). If any errors in the file are found, device will not be imported and importing process will be stopped.

Insert: If this checkbox is selected, items that are not yet present in current configuration will be added. Otherwise new content will not be processed.

Update: If this checkbox is selected, any items that already exist in current configuration replaced with new configuration. Otherwise existing configuration will be left intact.

8.3.4 Signals



The ”Tag Values” window displays a measurement list that contains information about tag states. Several columns can be sorted and filtered. By clicking the magnifying glass icon measurement history is shown (if recording is enabled).

This functionality is only available on “Cloud gateway” firmware type.

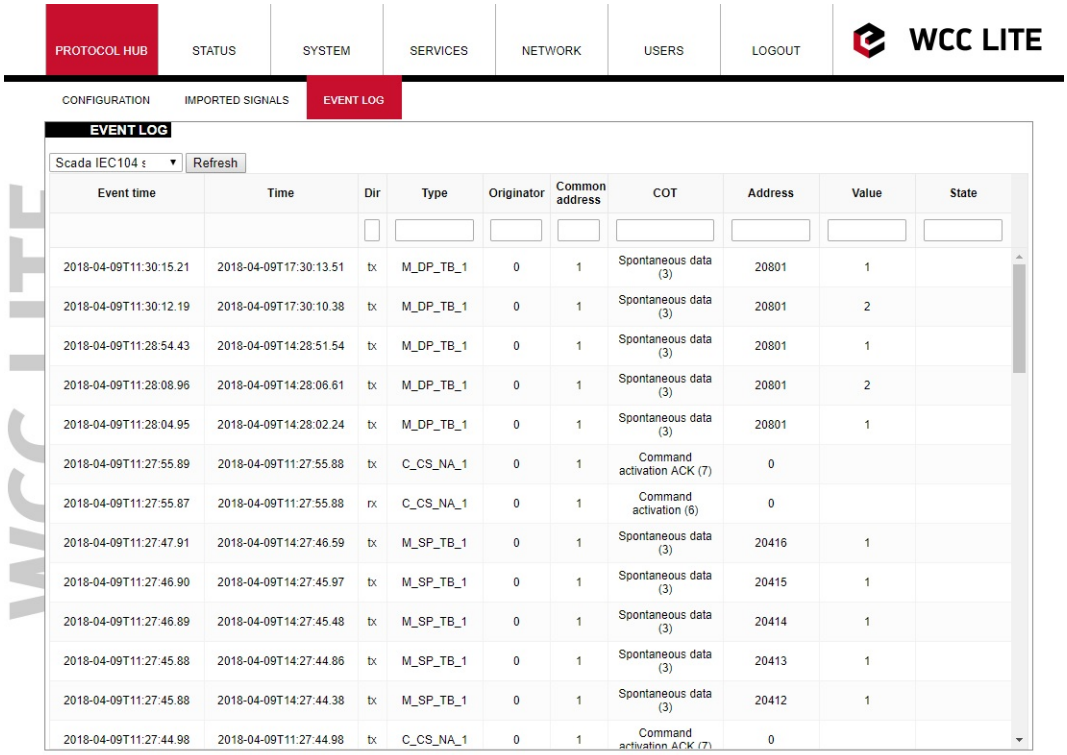
Name: Device or tag label. Sorting and filtering can be applied.

Value: Latest measured value. Sorting and filtering can be applied.

Status: Any error flags are listed here.

Time: The time measurement was updated. Sorting can be applied.

8.3.5 Sequence of Events (SOE)



SOE is the time-stamped status data. SOE allows to review latest events and changes for device’s state changes in chronological order. Newest events are shown at the top of the list. WCC Lite will time-stamp the status data with a time resolution of one millisecond.

Initially, all breakers, protection contacts digital status input points in the WCCLite; events captured from IEDs shall be configured as SOE points. It’s possible to assign any digital status input data point in the WCCLite as SOE point with Excel template during configuration.

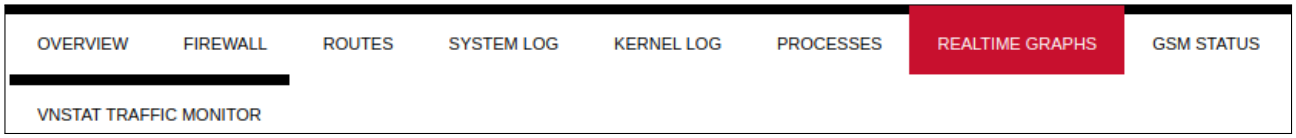
Each time a device changes state, the WCClite will save it with time-tag in internal storage. WCC Lite will maintain a SOE buffer within the configured history size limitations.

Events are recorded only for devices that have *Event history size* field set. When log size exceeds its limit, oldest records are deleted.

8.3.6 Imported signals

Imported signals section shows basic information about applied configuration. This section is view only.
For signals and their states refer to “Protocol Hub” section “Signals”

8.4 Status

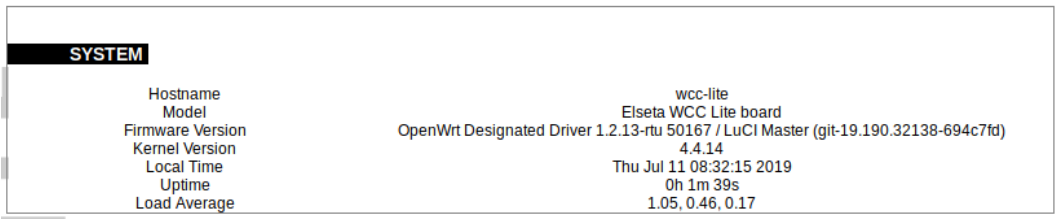


Status tab in a graphical interface includes various statuses of the device and contains the following subsections:

- *OVERVIEW*: brief summary of main system parameters;
- *FIREWALL*: current IPv4 and IPv6 firewall status;
- *ROUTES*: active route rules on the system;
- *SYSTEM LOG*: system log information;
- *KERNEL LOG*: kernel log information;
- *PROCESSES*: currently running system processes and their statuses;
- *REALTIME GRAPHS*: various real-time graphs for internal device data;
- *VNSTAT TRAFFIC MONITOR*: graphical network traffic representation;
- *GSM STATUS*: all information about gsm device (if it is present);

8.4.1 Overview

System



System section in status tab shows basic information about current status of the system.

Hostname: The label that is used to identify the device in the network.

Model: Model of the device.

Firmware version: Current firmware version.

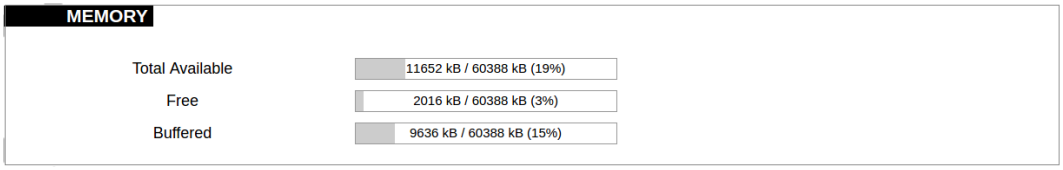
Kernel version: Current kernel version.

Local Time: Current local time.

Uptime: The time a device has been working.

Load average: Measure CPU utilization of the last 1, 5, and 15 minute periods. Load of 0.5 means the CPU has been 50% utilized over the last period. Values over 1.0 mean the system was overloaded.

Memory



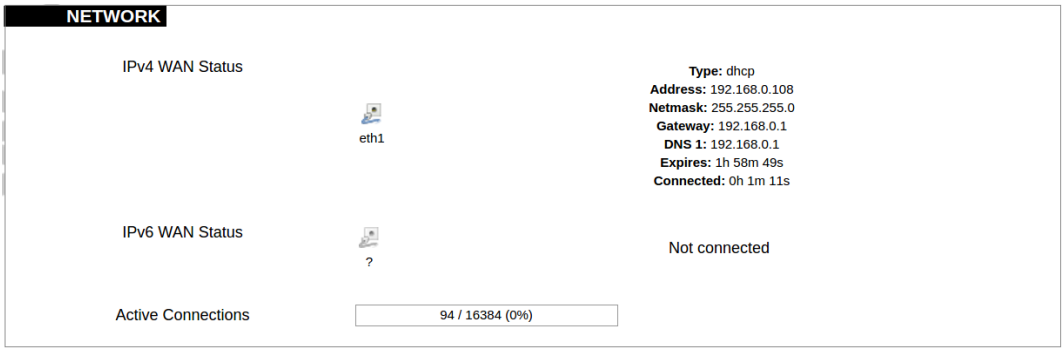
The ”Memory” window provides memory usage information on the device.

Total available memory: The amount of available memory that could be used over installed physical memory.

Free: The amount of physical memory that is not currently in use over installed physical memory.

Buffered: The amount of buffered memory that is currently in use for active I/O operations over installed physical memory.

Network



IPv4 WAN, IPv6 WAN status and active connections of the device.

Type: Type of addressing of IPv4 network interface – DHCP or static.

Address: IP address of the device.

Netmask: Netmask of the device.

Gateway: IP address of the Gateway.

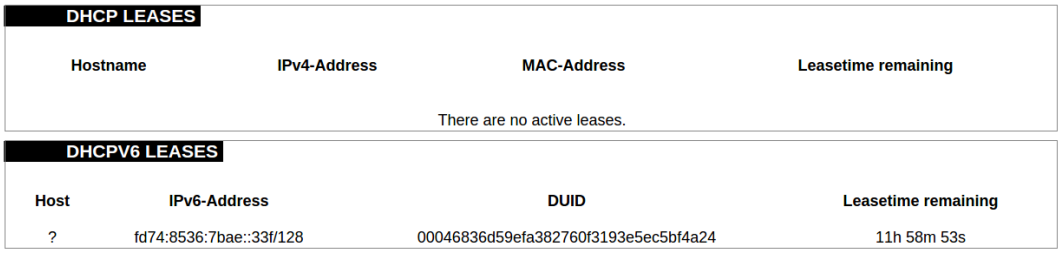
DNS: IP address of DNS server.

Expires: DHCP lease expiration time of the connection.

Connected: The time a device has been connected.

Active Connections: The number of the active connections with the device.

DHCP leases



DHCPv4 and DHCPv6 lease expiration time.

Hostname: The label that is used to identify the device in the network.

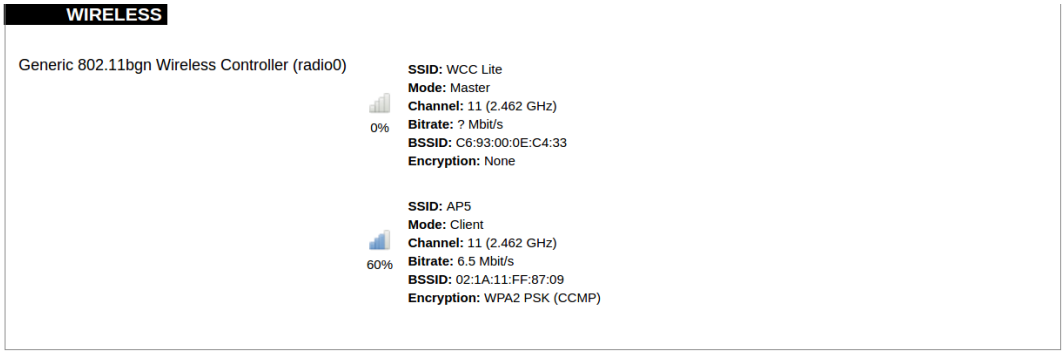
IPv4-Address: IPv4 address of network interface.

MAC-Address: The media access control address of IPv4 network interface.

DUID: DHCP Unique Identifier of IPv6 network interface.

Lease Time remaining: The amount of time the device will be allowed connection to the Router.

Wireless



WiFi interface information window.

SSID: The sequence of characters that uniquely names a wireless local area network.

Mode: Shows how the device is connected to the wireless network – Master or Client.

Channel: The number of channel and radio frequency for connection to access point.

Bitrate: The number of bits that pass the device in a given amount of time.

BSSID: The MAC address of the wireless access point.

Encryption: Security protocol for the wireless network.

Associated stations

List of associated stations (clients).

Network: Mode and SSID of network point.

MAC-Address: The media access control address of IPv4 network interface.

Hostname: The label or IP address that is used to identify the device in the network.

Signal/Noise: Received signal level over the background noise level. -30 dBm is the maximum achievable signal strength, -70 dBm is the minimum signal strength for reliable packet delivery in the wireless network

RX Rate/TX rate: Used measure data transmission in the wireless network over bandwidth. RX Rate represents the rate at which data packets being received by the device, TX Rate represents the rate at which data packets being sent from the device.

Board information

Board information provides the following details:

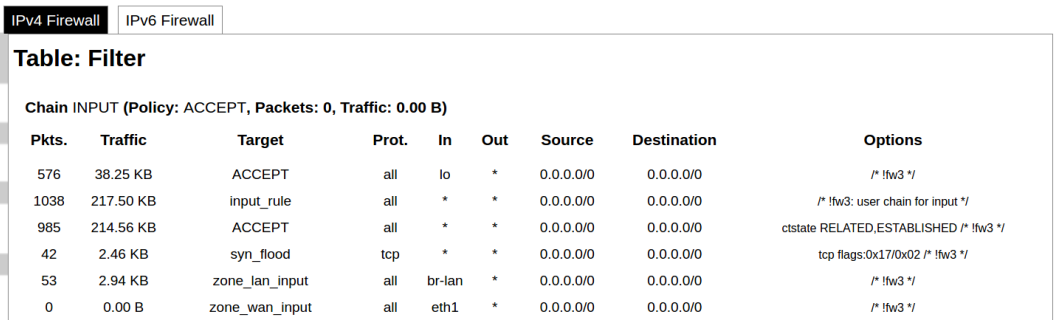
Hardware version: Current hardware version;

Serial number: Serial number of the board;

SoC ID: Unique identifier of CPU unit;

8.4.2 Firewall

IPv4 Firewall



Firewall rule list for IPv4 traffic.

Table: The four distinct tables which store rules regulating operations on the packet. Filter concerns filtering rules. NAT concerns translation of source or destination addresses and ports of packages. Mangle table is for specialized packet alteration. The raw table is for configuration exceptions.

Chain: The list of rules. Filter table has the following built-in chains: Input – concerns packets whose destination is the firewall itself, Forward – concerns packets transiting through the firewall, Output – concerns packets emitted by the firewall, Reject – reject the packet, Accept – allow the packet to go on its way. NAT table has the following built-in chains: Prerouting – to modify packets as soon as they arrive, Postrouting – to modify packets when they are ready to go on their way. Mangle table

has one built-in chain: Forward for transiting packets through the firewall.

Pkts.: The packets processed by the firewall.

Traffic: The amount of data processed by the firewall.

Target: The chain of the table of the firewall.

Prot.: The transport layer protocol processed by the firewall.

In: The network interface for the input chain processed by the firewall.

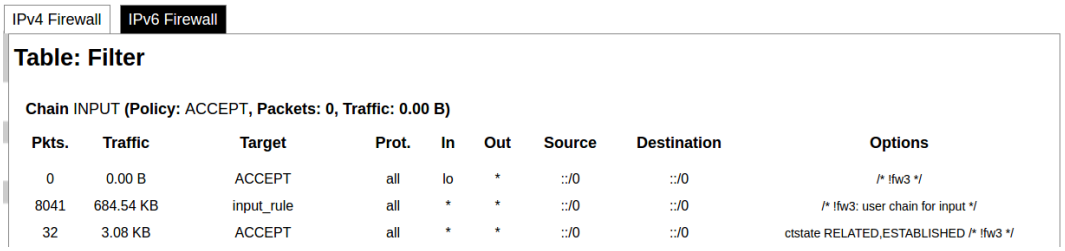
Out: The network interface for the output chain processed by the firewall.

Source: IPv4 address of the device that the packet comes from.

Destination: IPv4 address of the device that the packet goes to.

Options: The options for configuring the firewall.

IPv6 Firewall



Firewall rule list for IPv6 traffic.

Table: The three distinct tables which store rules regulating operations on the packet. Filter concerns filtering rules. Mangle table is for specialized packet alteration. The raw table is for configuration exceptions.

Chain: The list of rules. Filter table has the following built-in chains: Input – concerns packets whose destination is the firewall itself, Forward – concerns packets transiting through the firewall, Output – concerns packets emitted by the firewall, Reject – reject the packet, Accept – allow the packet to go on its way. Mangle table has one built-in chain: Forward for transiting packets through the firewall.

Pkts.: The packets processed by the firewall.

Traffic: The amount of data processed by the firewall.

Target: The chain of the table of the firewall.

Prot.: The transport layer protocol processed by the firewall.

In: The network interface for the input chain processed by the firewall.

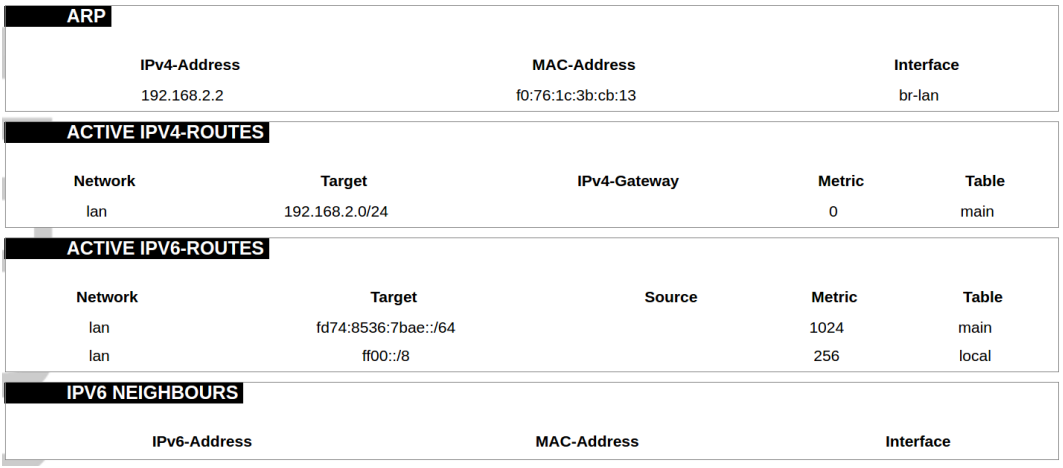
Out: The network interface for the output chain processed by the firewall.

Source: IPv6 address of the device that the packet comes from.

Destination: IPv6 address of the device that the packet goes to.

Options: The options for configuring the firewall.

8.4.3 Routes



ARP

IPv4-Address	MAC-Address	Interface
192.168.2.2	f0:76:1c:3b:cb:13	br-lan

ACTIVE IPV4-ROUTES

Network	Target	IPv4-Gateway	Metric	Table
lan	192.168.2.0/24		0	main

ACTIVE IPV6-ROUTES

Network	Target	Source	Metric	Table
lan	fd74:8536:7bae::/64		1024	main
lan	ff00::/8		256	local

IPV6 NEIGHBOURS

IPv6-Address	MAC-Address	Interface

The routing tables provide information on how datagrams are sent to their destinations.

ARP: An address Resolution Protocol which defines how IP address is converted to a physical hardware address needed to deliver packets to the devices.

Interface: The type of Network interface. br-lan refers to the virtual bridged interface: to make multiple network interfaces act as if they were one network interface.

Network: The type of network through which the traffic will be sent to the destination subnet.

Target: An address of the destination network. The prefix /24 refers the subnet mask 255.255.255.0.

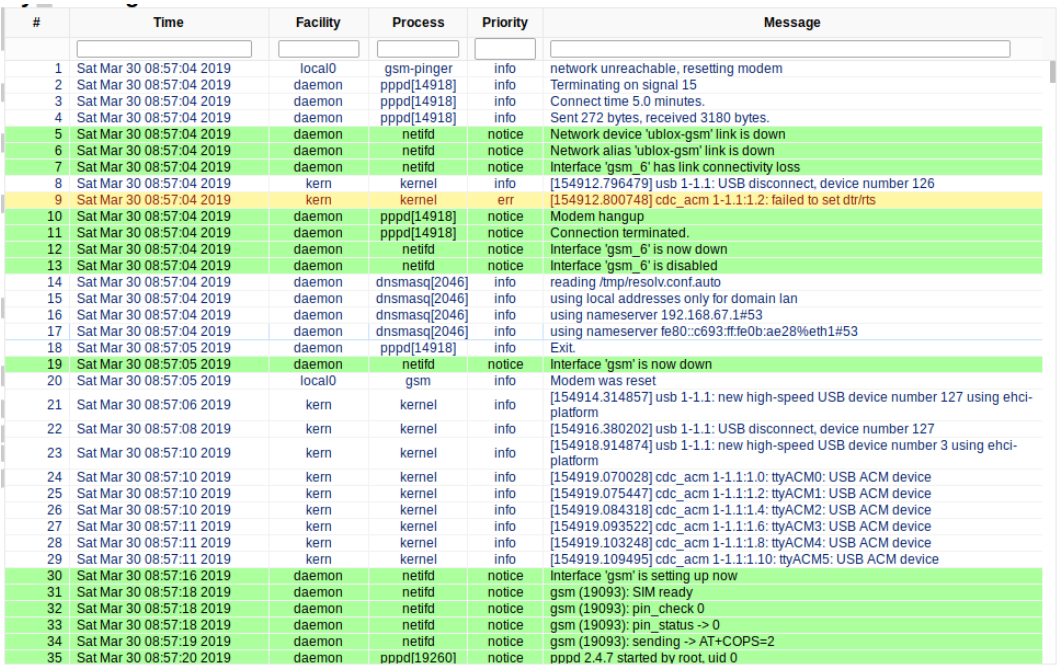
IPv4-Gateway: IP address of the gateway to which traffic intended for the destination subnet will be sent.

Metric: The number of hops required to reach destinations via the gateway.

Table: The type of routing tables: main (default), local (maintained by the kernel).

IPv6 Neighbours: The devices on the same network with IPv6 addresses.

8.4.4 System Log



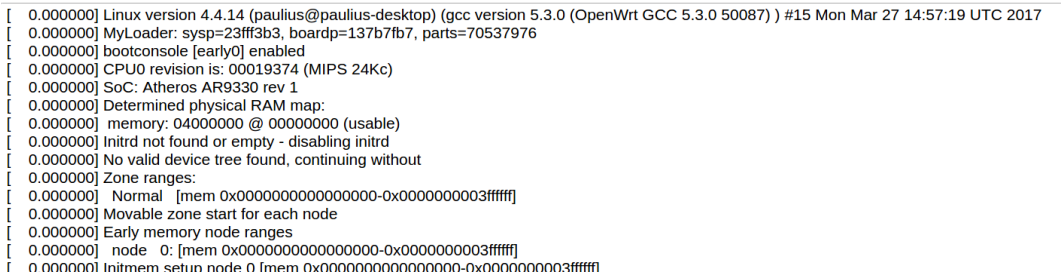
#	Time	Facility	Process	Priority	Message
1	Sat Mar 30 08:57:04 2019	local0	gsm-pinger	info	network unreachable, resetting modem
2	Sat Mar 30 08:57:04 2019	daemon	pppd[14918]	info	Terminating on signal 15
3	Sat Mar 30 08:57:04 2019	daemon	pppd[14918]	info	Connect time 5.0 minutes.
4	Sat Mar 30 08:57:04 2019	daemon	pppd[14918]	info	Sent 272 bytes, received 3180 bytes.
5	Sat Mar 30 08:57:04 2019	daemon	netifd	notice	Network device 'ublox-gsm' link is down
6	Sat Mar 30 08:57:04 2019	daemon	netifd	notice	Network alias 'ublox-gsm' link is down
7	Sat Mar 30 08:57:04 2019	daemon	netifd	notice	Interface 'gsm_6' has link connectivity loss
8	Sat Mar 30 08:57:04 2019	kern	kernel	info	[154912.796479] usb 1-1.1: USB disconnect, device number 126
9	Sat Mar 30 08:57:04 2019	kern	kernel	err	[154912.800748] cdc_acm 1-1.1:1.2: failed to set dtr/rts
10	Sat Mar 30 08:57:04 2019	daemon	pppd[14918]	notice	Modem hangup
11	Sat Mar 30 08:57:04 2019	daemon	pppd[14918]	notice	Connection terminated.
12	Sat Mar 30 08:57:04 2019	daemon	netifd	notice	Interface 'gsm_6' is now down
13	Sat Mar 30 08:57:04 2019	daemon	netifd	notice	Interface 'gsm_6' is disabled
14	Sat Mar 30 08:57:04 2019	daemon	dnsmasq[2046]	info	reading /tmp/resolv.conf.auto
15	Sat Mar 30 08:57:04 2019	daemon	dnsmasq[2046]	info	using local addresses only for domain lan
16	Sat Mar 30 08:57:04 2019	daemon	dnsmasq[2046]	info	using nameserver 192.168.67.1#53
17	Sat Mar 30 08:57:04 2019	daemon	dnsmasq[2046]	info	using nameserver fe80::c693:ff:fe0b:ae28%eth1#53
18	Sat Mar 30 08:57:05 2019	daemon	pppd[14918]	info	Exit.
19	Sat Mar 30 08:57:05 2019	daemon	netifd	notice	Interface 'gsm' is now down
20	Sat Mar 30 08:57:05 2019	local0	gsm	info	Modem was reset
21	Sat Mar 30 08:57:06 2019	kern	kernel	info	[154914.314857] usb 1-1.1: new high-speed USB device number 127 using ehci-platform
22	Sat Mar 30 08:57:08 2019	kern	kernel	info	[154916.380202] usb 1-1.1: USB disconnect, device number 127
23	Sat Mar 30 08:57:10 2019	kern	kernel	info	[154918.914874] usb 1-1.1: new high-speed USB device number 3 using ehci-platform
24	Sat Mar 30 08:57:10 2019	kern	kernel	info	[154919.070028] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
25	Sat Mar 30 08:57:10 2019	kern	kernel	info	[154919.075447] cdc_acm 1-1.1:1.2: ttyACM1: USB ACM device
26	Sat Mar 30 08:57:10 2019	kern	kernel	info	[154919.084318] cdc_acm 1-1.1:1.4: ttyACM2: USB ACM device
27	Sat Mar 30 08:57:11 2019	kern	kernel	info	[154919.093522] cdc_acm 1-1.1:1.6: ttyACM3: USB ACM device
28	Sat Mar 30 08:57:11 2019	kern	kernel	info	[154919.103248] cdc_acm 1-1.1:1.8: ttyACM4: USB ACM device
29	Sat Mar 30 08:57:11 2019	kern	kernel	info	[154919.109495] cdc_acm 1-1.1:1.10: ttyACM5: USB ACM device
30	Sat Mar 30 08:57:16 2019	daemon	netifd	notice	Interface 'gsm' is setting up now
31	Sat Mar 30 08:57:18 2019	daemon	netifd	notice	gsm (19093): SIM ready
32	Sat Mar 30 08:57:18 2019	daemon	netifd	notice	gsm (19093): pin_check 0
33	Sat Mar 30 08:57:18 2019	daemon	netifd	notice	gsm (19093): pin_status -> 0
34	Sat Mar 30 08:57:19 2019	daemon	netifd	notice	gsm (19093): sending -> AT+COPS=2
35	Sat Mar 30 08:57:20 2019	daemon	pppd[19260]	notice	pppd 2.4.7 started by root, uid 0

System log window shows a table containing the events that are logged by the device. It has the following columns:

- *#* (sequence number);
- *Time* (day of the week, month, day of the month, time and year);
- *facility*;
- *process* (who generated the message);
- *priority level*;
- *message*.

Messages can be sorted and filtered to extract a particular set of messages. This might be useful when debugging kernel or protocol level problems.

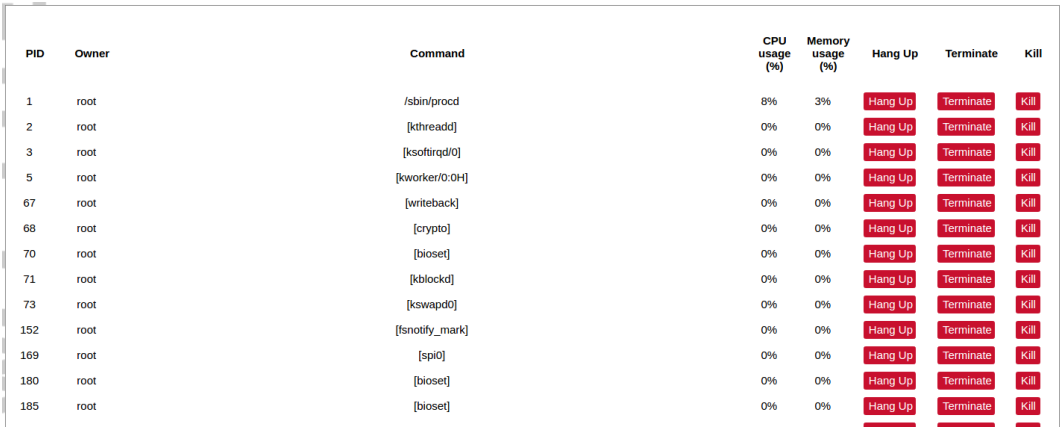
8.4.5 Kernel Log



```
[    0.000000] Linux version 4.4.14 (paulius@paulius-desktop) (gcc version 5.3.0 (OpenWrt GCC 5.3.0 50087) ) #15 Mon Mar 27 14:57:19 UTC 2017
[    0.000000] MyLoader: sysp=23fff3b3, boardp=137b7fb7, parts=70537976
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU0 revision is: 00019374 (MIPS 24Kc)
[    0.000000] SoC: Atheros AR9330 rev 1
[    0.000000] Determined physical RAM map:
[    0.000000]  memory: 04000000 @ 00000000 (usable)
[    0.000000] Initrd not found or empty - disabling initrd
[    0.000000] No valid device tree found, continuing without
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x0000000000000000-0x0000000003ffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000000000000-0x0000000003ffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000000000000-0x0000000003ffffff]
```

Kernel log shows a list of the events that are logged by the kernel of the device. Log format: time in seconds since the kernel started and message.

8.4.6 Processes



PID	Owner	Command	CPU usage (%)	Memory usage (%)	Hang Up	Terminate	Kill
1	root	/sbin/procd	8%	3%	Hang Up	Terminate	Kill
2	root	[kthreadd]	0%	0%	Hang Up	Terminate	Kill
3	root	[ksoftirqd/0]	0%	0%	Hang Up	Terminate	Kill
5	root	[kworker/0:0H]	0%	0%	Hang Up	Terminate	Kill
67	root	[writeback]	0%	0%	Hang Up	Terminate	Kill
68	root	[crypto]	0%	0%	Hang Up	Terminate	Kill
70	root	[bioset]	0%	0%	Hang Up	Terminate	Kill
71	root	[kblockd]	0%	0%	Hang Up	Terminate	Kill
73	root	[kswapd0]	0%	0%	Hang Up	Terminate	Kill
152	root	[fsnotify_mark]	0%	0%	Hang Up	Terminate	Kill
169	root	[spi0]	0%	0%	Hang Up	Terminate	Kill
180	root	[bioset]	0%	0%	Hang Up	Terminate	Kill
185	root	[bioset]	0%	0%	Hang Up	Terminate	Kill

List of processes running on the system.

PID: Process ID.

Owner: User to whom the process belongs.

Command: Process.

CPU usage: It is CPU usage of the individual process. CPU usage above 90 % is an indicator of insufficient processing power.

Memory usage: Memory usage of the individual process.

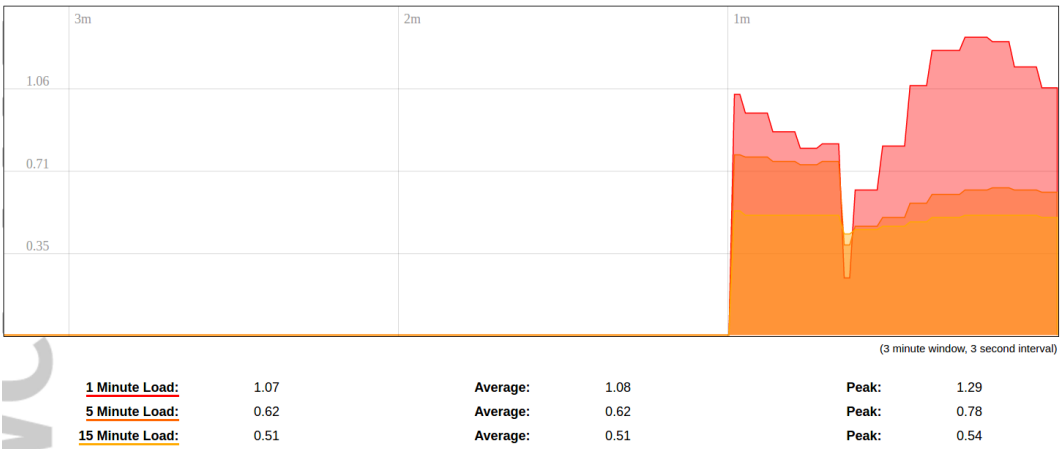
Hang Up: To freeze the process.

Terminate: To end the process cleanly.

Kill: To end the process immediately.

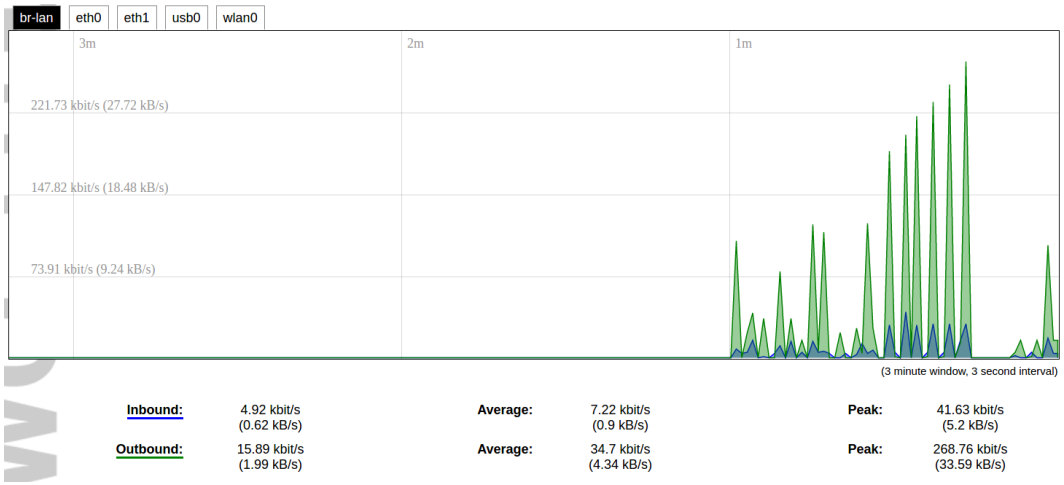
8.4.7 Realtime graph

Realtime Load



CPU utilization graph. Load of 0.5 means the CPU has been 50% utilized over the last period. Values over 1.0 mean the system was overloaded.

Realtime Traffic



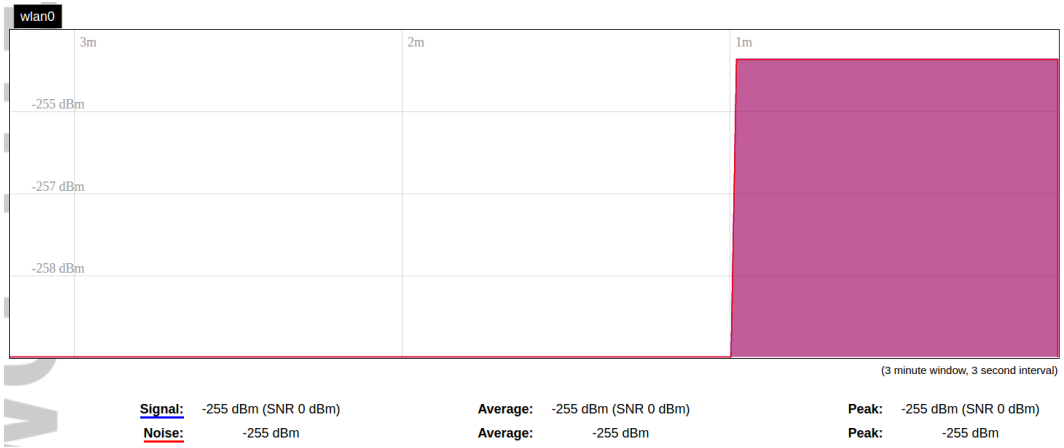
Graphs representing the status of the virtual and physical network interfaces of the device.

Inbound: The speed at which the incoming packets arrive at the device.

Outbound: The speed of the packets which were originated by the device.

Phy. Rate: The speed at which bits can be transmitted over the physical layer.

Realtime Wireless



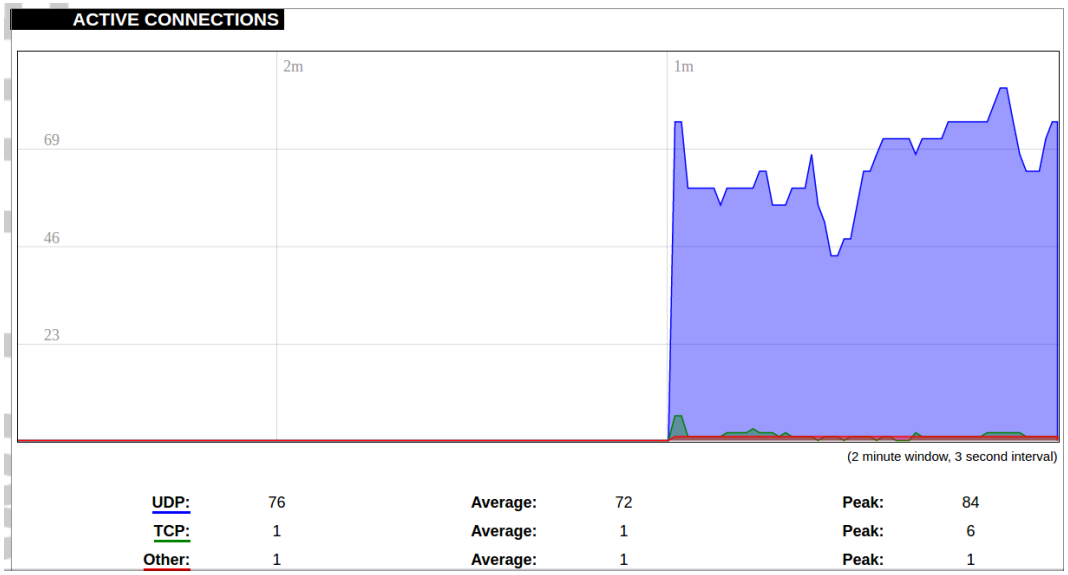
WiFi status graph.

Signal: Signal strength level.

Noise: Noise level.

Phy. Rate: The speed at which bits can be transmitted on the physical layer.

Active connections



Graph representation of active connections with the device.

UDP: Transport layer – User Datagram Protocol.

TCP: Transport layer – Transmission Control Protocol.

Network: Type of the network layer – IPv4 or IPv6.

Source, Destination: IP address and the port number.

Transfer: The amount of the transferred data in kB and packets.

GSM signal quality

Graph representation of gsm modem receiving signal quality. RSRP - RSRQ graph is showed, when connected to 4G/LTE network, RSSI - when 2G/3G networks are used.

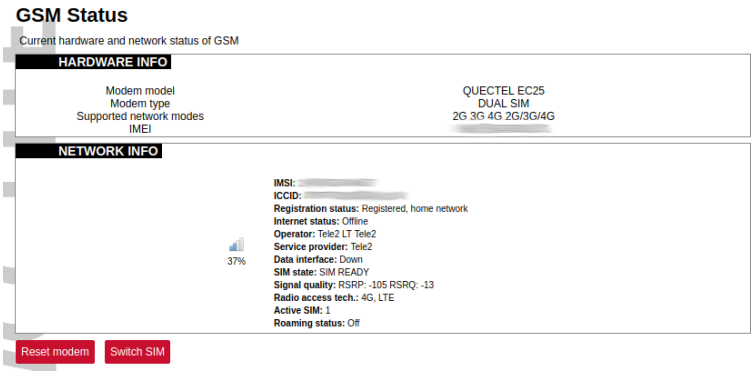
RSSI: Received Signal Strength Indicator in dBm.

RSRP: Received Signal Reference Power in dBm.

RSRQ: Received Signal Reference Quality in dBm.

8.4.8 GSM status

This page shows all information that is related to GSM modem.



Hardware info

All static information on GSM modem.

Modem model: Manufacturer and model of present modem.

Modem type: Single SIM or Double SIM modem.

Supported network modes: Shows which network modes (or their combinations) are supported (e.g. 2G 4G 2G/4G).

IMEI: IMEI (International Mobile Equipment Identity number).

Network info

All dynamic information on GSM modem and connected network.

IMSI: IMSI (International Mobile Subscriber Identity) number related to current SIM card user.

ICCID: ICCID (Integrated Circuit Card Identifier) number related to physical SIM card.

Registration status: Curren status of network connection.

Internet status: Status of connection to internet (valid, when gsm-pinger is enabled and can reach provided hosts).

Operator: Operator's name, to which modem is currently connected.

Service provider: IMEI (Service provider for SIM card.

Data interface: Shows, whether wcc-lite have a data connection through gsm or not (possible values: "Up", "Down").

SIM state: Shows current status of SIM card (needs PIN, needs PUK, not-inserted and etc.).

Signal quality: Shows current signal strength value in dBms. RSSI value is shown, when connected to 2G/3G networks, RSRP-RSRQ values - when connected to 4G network.

Radio access tech.: Current radio technology used (2G, 3G or 4G).

Active SIM: Shows which SIM card is active (if the modem is Dual SIM).

Roaming status: Current status of roaming ("Off", "On").

Little bars with percentage at the center left shows signal strength. It is calculated with the respect to current radio access technology used (RSSI or RSRP). Two buttons at the bottom can reset (cold-reset) modem or manually switch SIM cards (if it is Dual SIM modem and both cards are enabled).

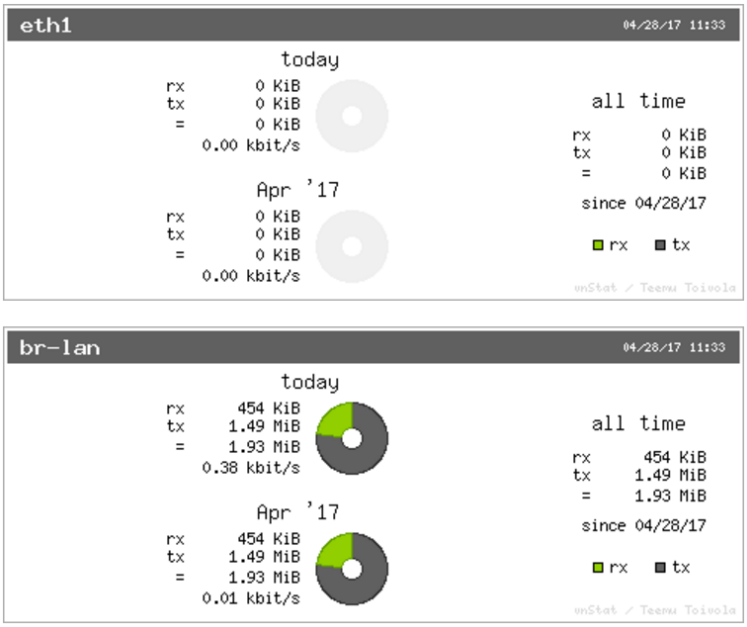
Signal quality is described in different ways for different type for different mobile services: Received Signal Strength Indication (RSSI) in GSM (2G) and UMTS (3G), the Reference Signal Received Quality (RSRQ) in LTE RAT.

The Reference Signal Received Power (RSRP) is a LTE specific measure that averages the power received on the subcarriers carrying the reference signal. The RSRP measurement bandwidth is equivalent to a single LTE subcarrier: its value is therefore much lower than the total received power usually referred to as RSSI. In LTE the RSSI depends on the currently allocated bandwidth, which is not pre-determined. Therefore the RSSI is not useful to describe the signal level in the cell.

8.4.9 VNSTAT Traffic monitor

To monitor the traffic of various network interfaces VNSTAT Traffic monitor can be used. Traffic tracking can be useful if user wants to have a precise information on how much data is used because it can have a dependance with data transmission costs, for example, mobile (cellular) data.

Graph



An example graph shows the statistics gathered for two network interfaces. In these graphs:

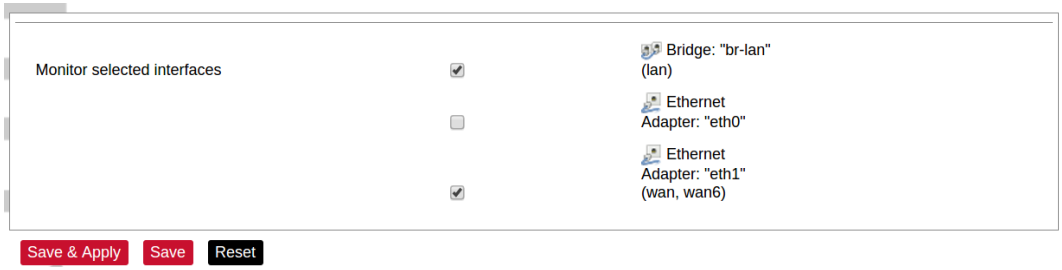
eth1: Network interface (e.g. Ethernet).

br-lan: Virtual network interface (bridge).

rx: Data packets received by the device.

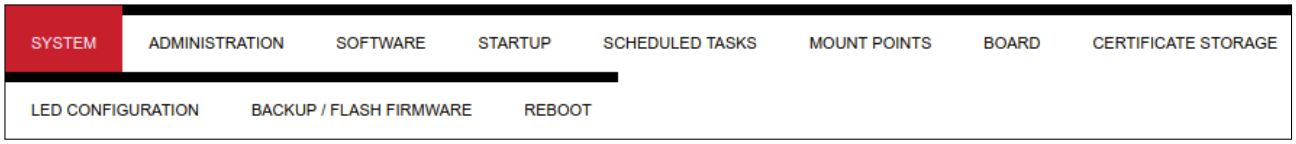
tx: Data packets sent from the device.

Configuration



Interfaces to be monitored can be selected in a configuration screen. It includes all the network interfaces configured in a system. To start or stop monitoring user should either select or unselect respective checkbox and save settings by pressing *Save & Apply*.

8.5 System



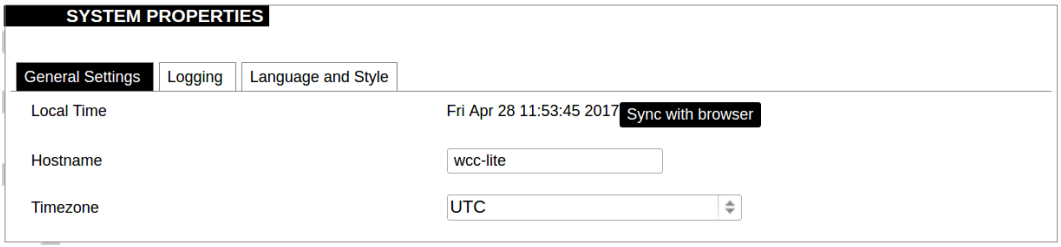
System tab includes various properties, configuration, and settings of the system and contains the following pages:

- *SYSTEM*: properties and settings of the system.
- *ADMINISTRATION*: settings of the administration for various services.
- *SOFTWARE*: settings of the packages.
- *STARTUP*: process management.
- *SCHEDULED TASKS*: settings of the scheduled tasks.
- *MOUNT POINTS*: settings for the mount points.
- *BOARD*: board configuration.
- *CERTIFICATE STORAGE*: certificate management panel.
- *LED CONFIGURATION*: settings for the LEDs.
- *BACKUP/FLASH FIRMWARE*: management of the configuration files and firmware image upgrade.
- *REBOOT*: device reboot page.

8.5.1 System

Basic aspects of the device can be configured. These include time settings, hostname, system event logging settings, language and theme selection.

System properties

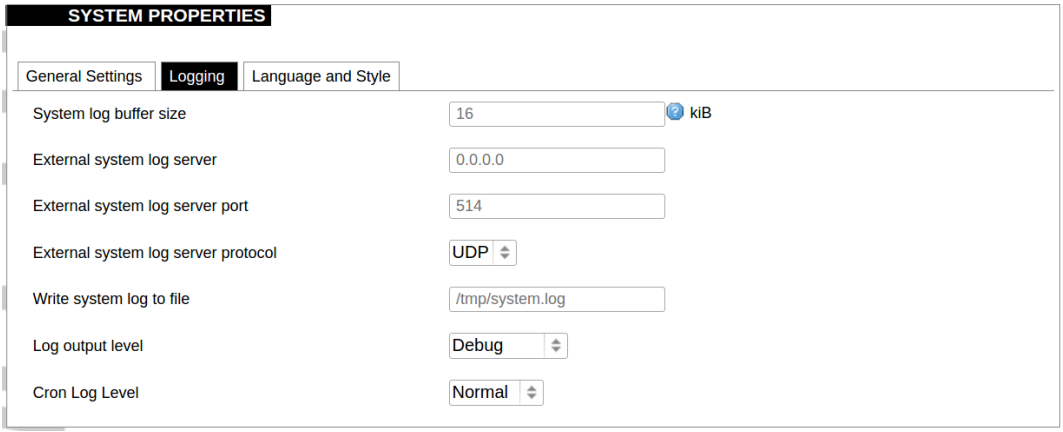


General settings of the WCC Lite device are defined as follows:

Local Time: Current local time.

Hostname: The label that is used to identify the device in the network.

Timezone: A region of the globe that observes a uniform standard time. The time zone number indicates the number of hours by which the time is shifted ahead of or behind UTC – Coordinated Universal Time. Some zones are, however, shifted by 30 or 45 minutes.



Logging settings of the WCC Lite device are defined as follows:

System log buffer size: The amount of the records before writing these data to the disk.

External system log server: IP address of the server.

External system log server port: An endpoint of communication with the server.

External system log server protocol: A standard that defines how to establish and maintain a network connection: UDP - User Datagram Protocol, TCP - Transmission Control Protocol.

Write system log to file: The name of the file with the path to it.

Log output level: Log output messages can be grouped by their importance to the user. Levels are described in a table below.

Table 3: Log output levels

Log output level	Description
Emergency	System is unusable
Alert	Action must be taken immediately
Critical	Critical conditions
Error	Error conditions
Warning	Potentially hazardous conditions
Notice	Normal conditions that might need action
Info	Information messages
Debug	Debugging messages

Cron Log Level: Cron has three output levels to choose from to write to its logs. Possible options are described in a table below.

Table 4: Cron log levels

Cron log level	Description
Debug	Debugging messages
Normal	General administrative messages
Warning	Potentially hazardous conditions

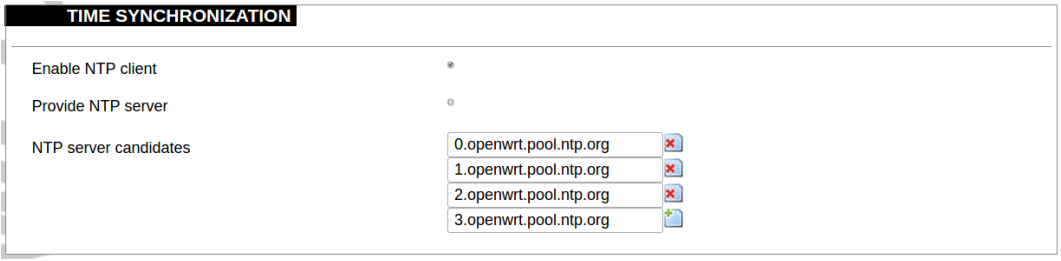
Language and Style settings are defined as follows:

Language: The language of the Web interface of the device.

Design: The theme of the Web interface of the device.

Time synchronization

WCC Lite has an NTP client to synchronize date and time with external sources. It is not the only source for synchronization, it can also be done using methods defined in IEC-60870-5 protocols.



Please take care choosing a time sync method. If both NTP and IEC 60870-5 protocol slave interface time sync methods are activated simultaneously, they can interfere if there is a time difference. We strongly recommend to use single time sync method to prevent time interference.

Time synchronization options are defined as:

Enable NTP client: The local time of the device will sync with external time servers.

Provide NTP server: Turn the device into a local NTP server.

NTP server candidates: The network time protocol servers.

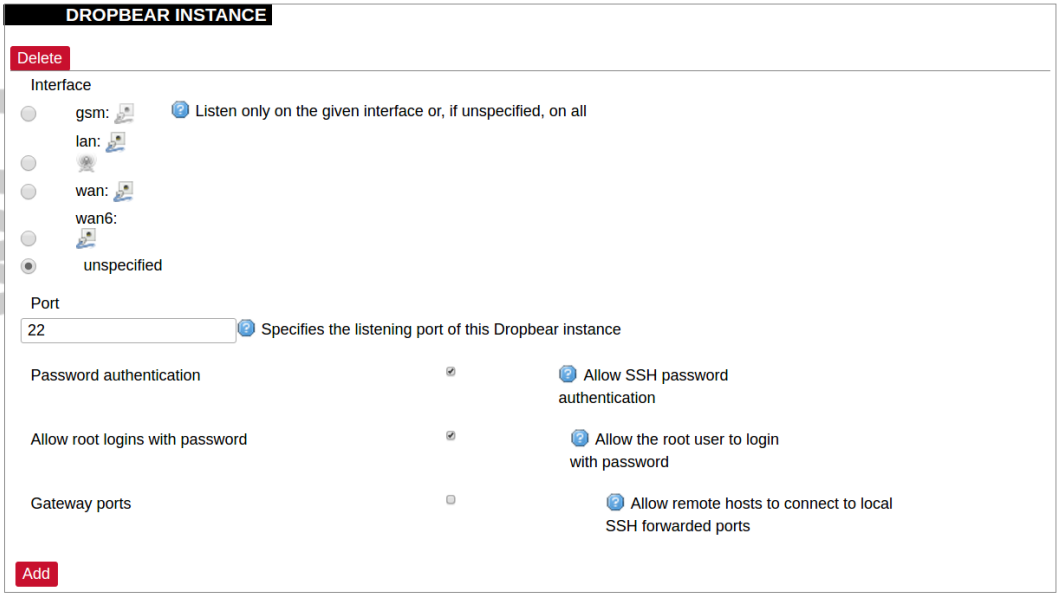
8.5.2 Administration

Administrator password can be changed. To change it the combination of digits and letters of the alphabet should be entered and then confirmed in *Confirmation* field by typing in again.

It is advised not to use the default password.

Dropbear instance

WCC Lite has a compact secure shell (SSH) server named *Dropbear*. Multiple options are, however, available to be changed via WCC Lite web interface, ranging from automatic firewall rules to authentification flexibility.



Dropbear options are defined as follows:

Interface: Listen only on the given interface or on all, in unspecified.

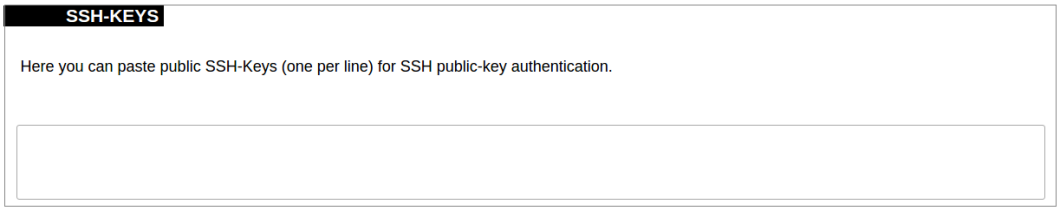
Port: Specifies the listening port of this interface.

Password authentication: Allow SSH password authentication.

Allow roots logins with password: Allow the root user to login with the password.

Gateway ports: Allow remote hosts to connect to local SSH forwarded ports.

SSH-keys



SSH keys can be added via WCC Lite web interface. They might be helpful if the user logs into device frequently and does not want to always have to write his credentials.

HTTPS certificate

WCC Lite by default is shipped with a default certificate for HTTPS connection. This certificate only enables connecting to device via web interface and might cause warnings from a web browser. To eliminate them, user can use his own certificate to secure access to web interface.

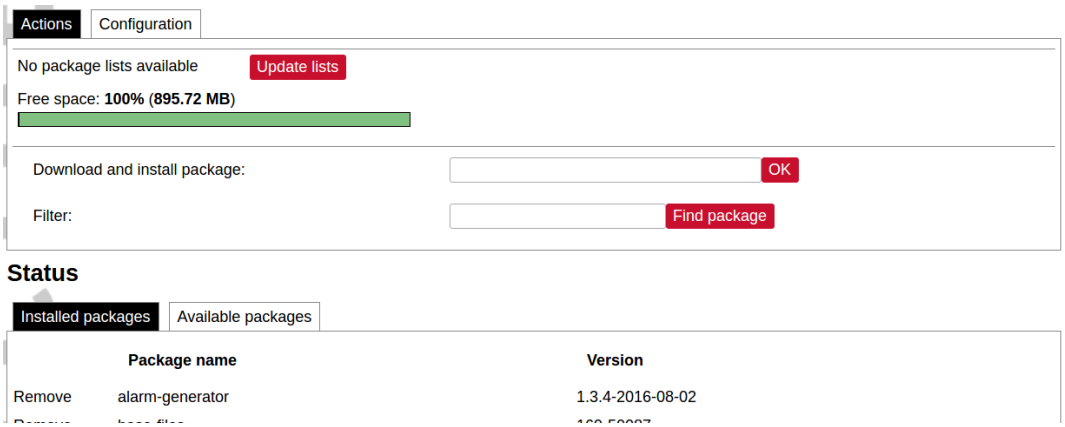
User can use certificates uploaded to a certificate storage. It should be noted that only valid

certificates with **.pem* extension can be used. Certificate to be used is validated every time device is restarted. If validation fails, default certificate is used. This is done to prevent user from losing device access via web interface.

For new certificate to come to effect user should restart the device.

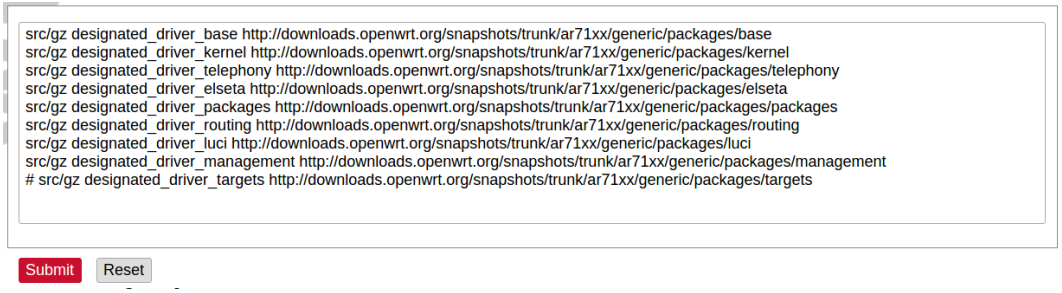
8.5.3 Software

Individual packages can be installed via WCC Lite web interface. They can either be installed using web link or selected from the pre-defined feeds.

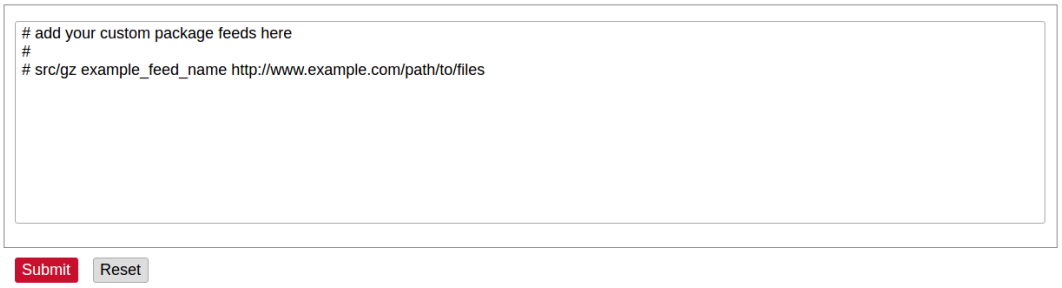


Various options can be selected when installing packages, however, default ones should work well enough and it's advised to only change them for advanced users.

Feeds from which packages are listed for update are defined in Open PacKaGe management (OPKG) configuration that can be changed easily from user interface.

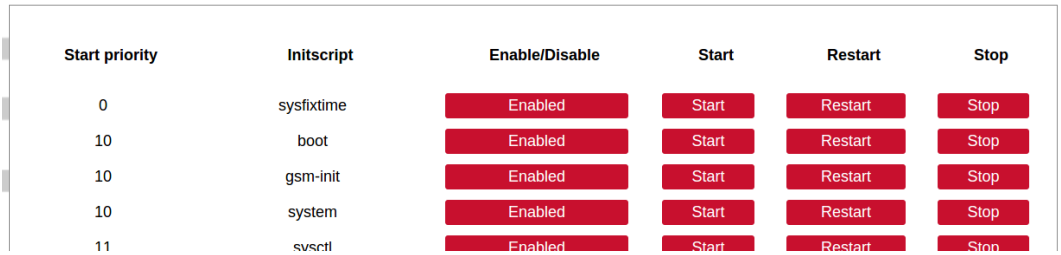


Specific distribution feeds can also be added for special cases if standard ones do not fit the needs.

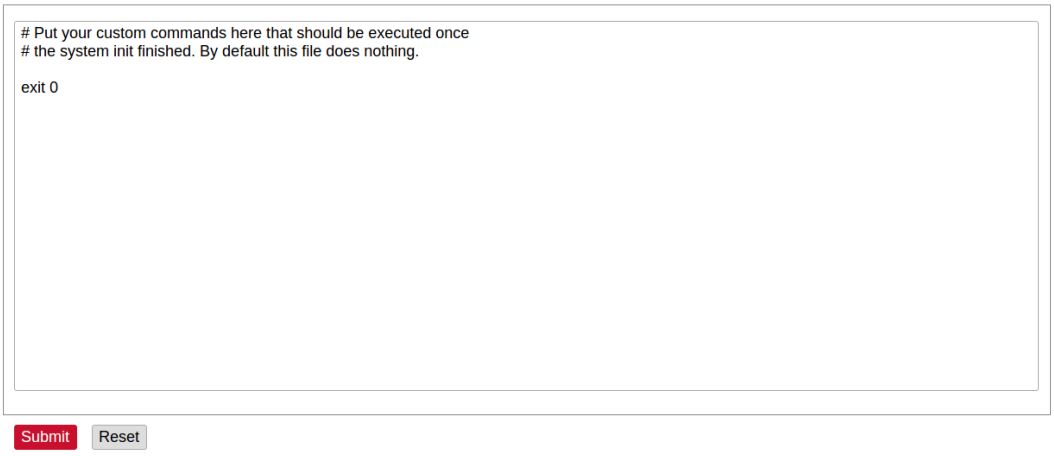


8.5.4 Startup

All of the processes that have *init.d* scripts can optionally enabled or disabled. This can be very useful if user only intends to use only part of the processes.



User should not disable processes that are essential for device operation as it can render the device unusable.



User can optionally run scripts and programs on device startup by putting them into a `/etc/rc.local` file. This file can be updated from WCC Web interface.

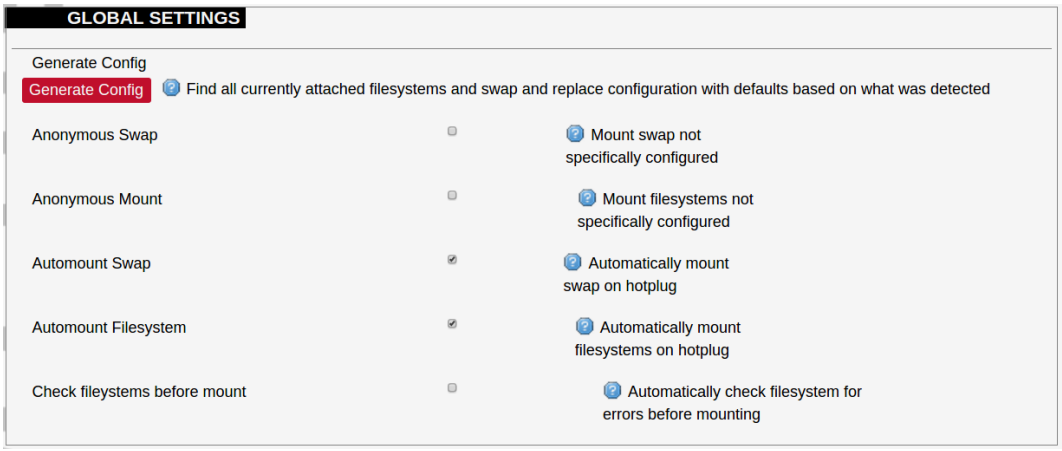
8.5.5 Scheduled tasks

Various tasks can be scheduled with the system crontab. New tasks can be included by creating and saving new rules conforming to *cron* rules. WCC Lite accepts full *cron* configuration functionality.

Example in the pictures shows how to execute the disk usage command to get the directory sizes every 6 p.m. on the 1st through the 15th of each month. E-mail is sent to the specified email address.

8.5.6 Mount points

Global settings



File system mount point configuration window.

Generate Config: Find all currently attached filesystems and swap and replace configuration with defaults based on what was detected.

Anonymous Swap: Mount swap not specifically configured.

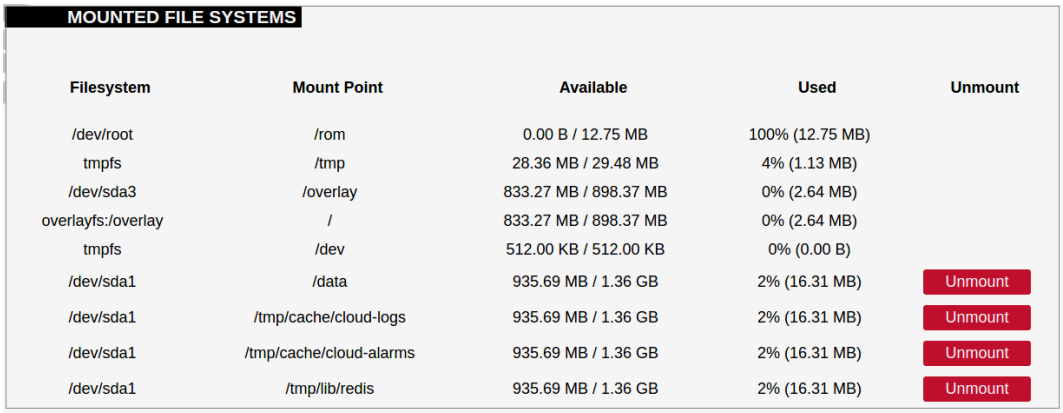
Anonymous Mount: Mount filesystems not specifically configured.

Automount Swap: Automatically mount swap on hotplug.

Automount Filesystem: Automatically mount filesystems on hotplug.

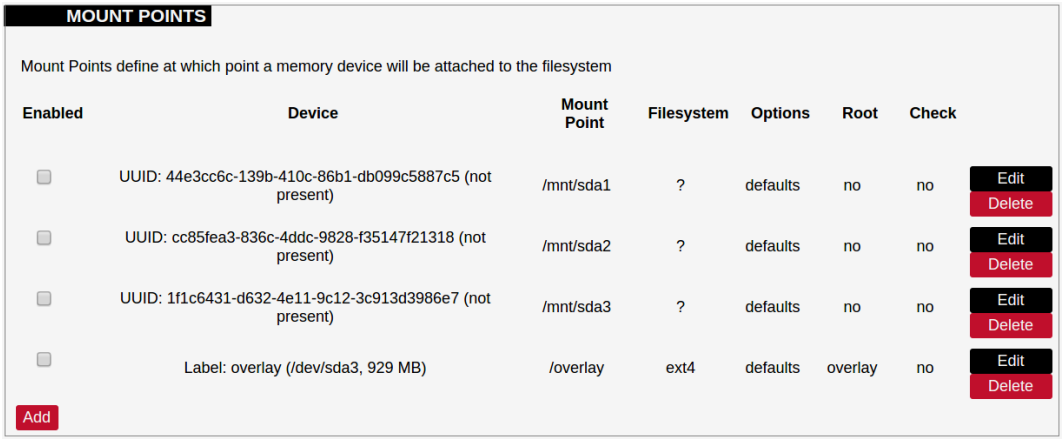
Check filesystems before mount: Automatically check filesystem for errors before mounting.

Mounted file systems



List of mounted file systems, some of which can be dismounted manually.

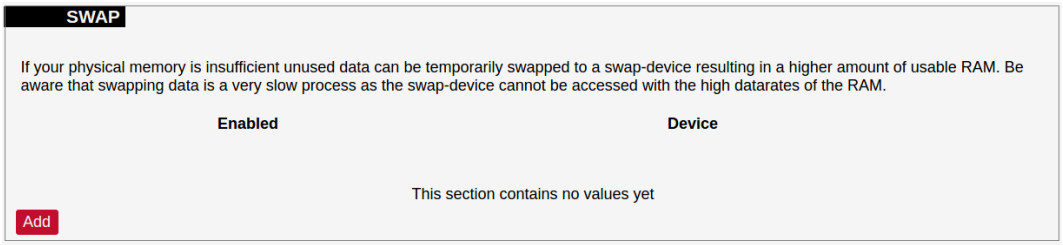
Mount points



List of mount points which can be enabled, disabled or deleted.

Swap

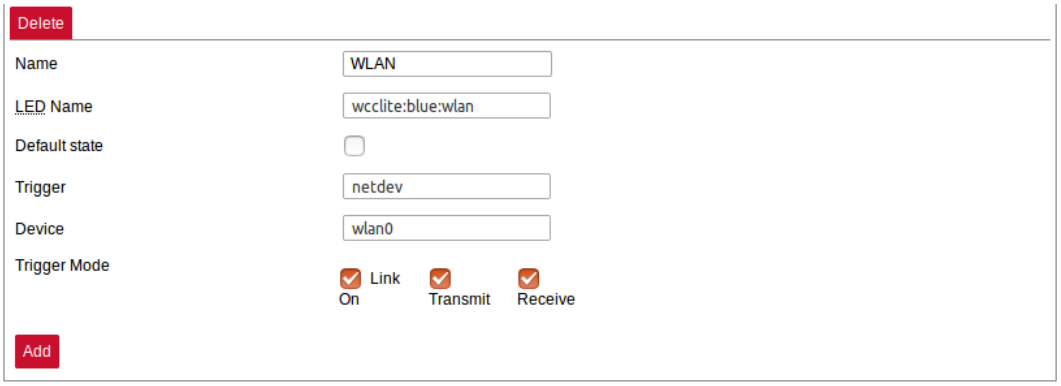
Swap section is used to describe the virtual memory that can be used if there's a lack of main memory. WCC Lite does not use any virtual memory by default.



It should be noted that virtual memory might do a lot of reading and writing operations. As WCC Lite uses SD card as an additional flash memory, it is highly advised to not use swap to reduce wearing.

8.5.7 LED configuration

WCC Lite has three LEDs that can be configured: WAN, LAN and WLAN. All of the LEDs have a default configuration which should fit most of the cases.



All possible LED configuration options:

Name: Name of the LED configuration.

LED Name: Colour and location of the LED. These can be changed, however, normally they should be left unchanged

Default state of the LED: On/Off.

Trigger: One of the various triggers can be assigned to an LED to changes its states. Possible values are shown in a table below.

Table 5: Possible trigger for an LED

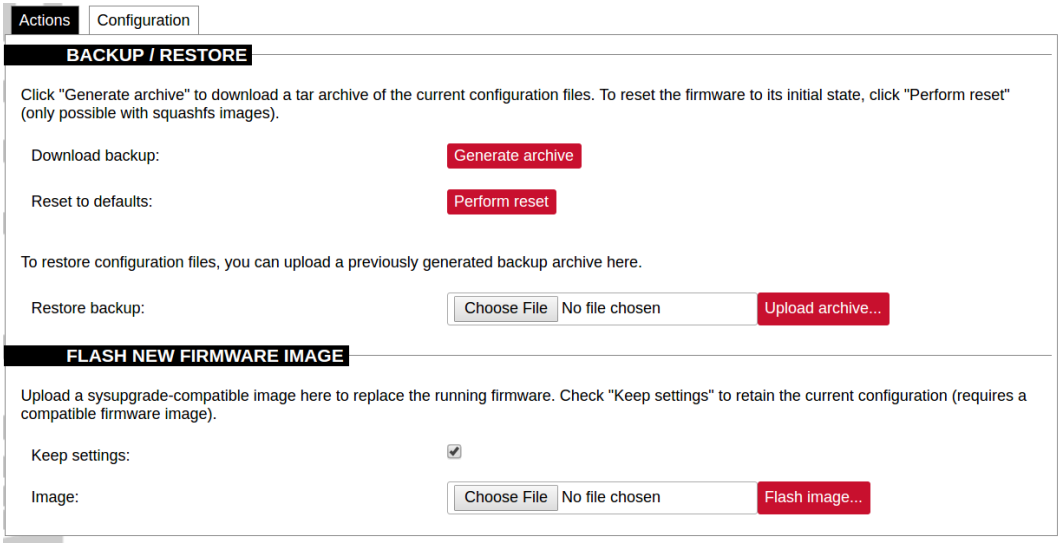
Trigger type	Description
none	No blinking function assigned to LED
defaulton	LED always stays on
timer	Blinking according to predefined timer pattern
heartbeat	Simulating actual heart beats
nand-disk	Flashed as data is written to flash memory
netdev	Flashes according to link status and send/receive activity
phy0rx, phy0tx, phy0assoc, phy0radio, phy0tpt	Flashed on WiFi activity events
usbdev	Turned on when USB device is connected. Applicable for modems

Device: Network interface which is going to be tracked.

8.5.8 Backup/flash firmware

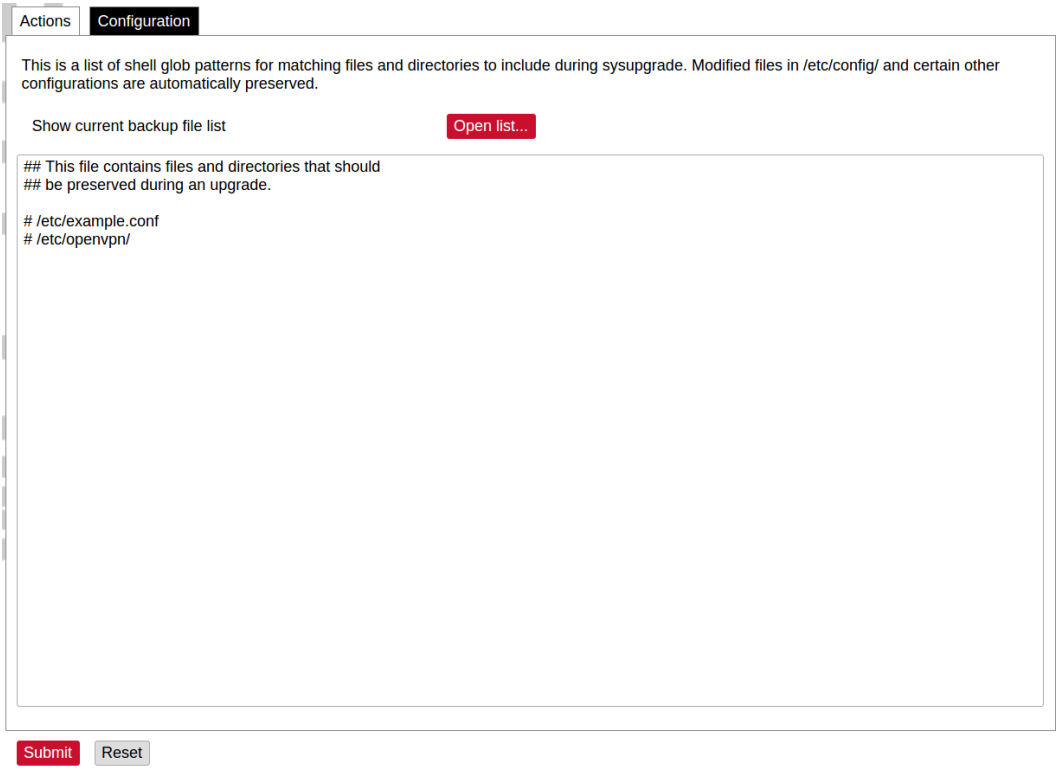
Software update allows to upgrade the software running in *WCC Lite*. It is recommended to keep the device up to date to receive the latest features and stability fixes.

Backup archives contain complete *WCC Lite* configuration that can be restored at any time. A file will be downloaded by your browser when creating a backup. This file can be later uploaded to the web page to restore configuration.



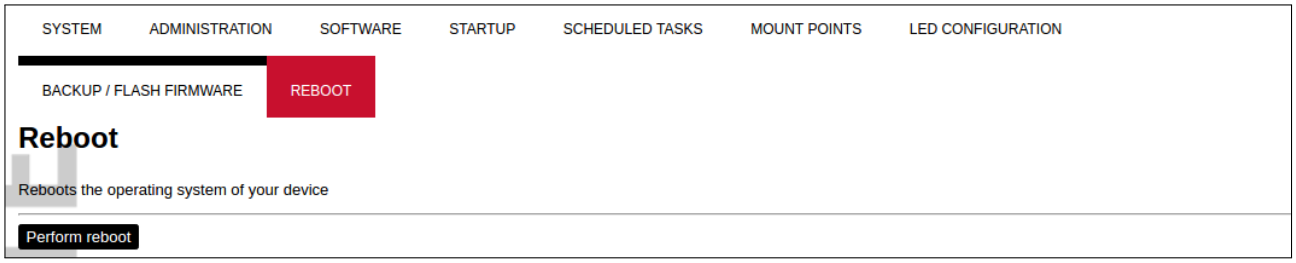
A user can choose to keep existing settings after an upgrade. Marking *Keep Settings* checkbox preserves files listed in `/etc/sysupgrade.conf` and `/lib/upgrade/keep.d/`. It is advised to do a clean install and use backup files to restore settings later if a user intends to make a major system upgrade.

Uploading firmware image, to preserve RAM memory, will stop all Protocol HUB processes. After upload, you will have 2 minutes to proceed with firmware flash or to cancel it. After 2 minutes, firmware file will be deleted and Protocol HUB processes will be restarted.



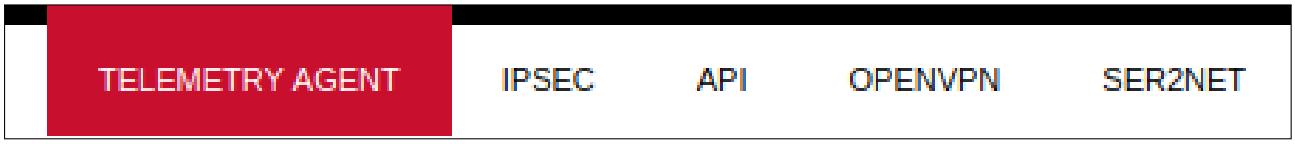
A file name `/etc/sysupgrade.conf` can be updated via WCC Web interface. To preserve additional file user should add them to backup file and press *Submit*. To get the whole list files that would be backed up press *Open list...*. It is advised to check it before doing a back-up or an upgrade while keeping settings.

8.5.9 Reboot



This reboots the operating system of the device.

8.6 Services



Services tab shows the services of the device and contains the following subsections:

- *TELEMETRY AGENT*: device telemetry sending to a remote server;
- *IPSEC*: encrypted virtual private network (VPN) configuration.
- *OPENVPN*: shows the open-source software application that implements virtual private network (VPN).
- *SER2NET*: network-to-serial proxy;

8.6.1 Telemetry agent

Having data about the device helps to easily maintain it. *Telemetry agent* gathers information in a compact and easily decodable way. It uses UDP packets therefore only small overhead is introduced. However, UDP does not guarantee the arrival of sent packets therefore not every message might reach the server saving these messages.

To start using *Telemetry agent* a user should configure and enable it. Four options are available:

- Enable agent;
- Server address;
- Port (UDP);
- Period (s).

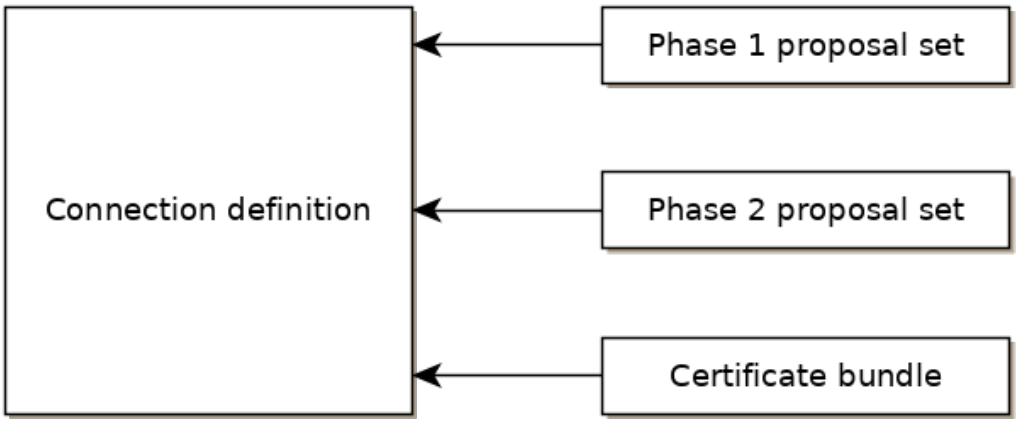
Every time timer of period length expires, a message is sent to a server of configured server if service is enabled .

Telemetry agent doesn't start as a service if *Enable agent* checkbox is unchecked. Enabling agent and saving the configuration automatically starts the process with the new configuration.

8.6.2 IPsec

Background

WCC Lite supports ipsec vpn, thus is able to deliver data securely over encrypted link. To establish ipsec vpn, a connection definition must be created by entering appropriate configuration settings. For advanced connection description auxiliary settings sets can be defined. They can be joined to the connection and can be reusable several times according to the need. Each configuration record is identified by a unique name, which is assigned in time of creation. The following diagram shows relations between connection and auxiliary sets.



Ipsec settings

Connection description

Options supported by wcclite is described below.

Item	Type	Description
Gateway	string	Host name or IP address of the remote peer.
Type	selector	Tunnel mode: full packet encryption, covers host-to-host, host-to-subnet, subnet-to-subnet situations or transport mode: ip payload encryption, secures host-to-host data only.
Local subnet	string	Specifies local network, in form network/netmask, for example 192.168.11.0/24
Remote subnet	string	Specifies remote network at another side of a tunnel.
Authentication	selector	Pre-shared key or RSA certificate
Pre-shared key	string	Available if Authentication set to Pre-shared key
Certificate set	selector	Available if Authentication set to RSA certificate. Selectable from configured auxiliary set.
Phase 1 proposal (IKE)	selector	Authentication-encryption schema, selectable from configured auxiliary set.
Phase 2 proposal (ESP)	selector	Authentication-encryption schema, selectable from configured auxiliary set.
Local ID	string	Specifies the identity of the local endpoint
Remote ID	string	Specifies the identity of the remote endpoint
Key exchange	selector	Sets method of key exchange IKEv2 or IKEv1. Default IKEv2.
Exchange mode	selector	Main or aggressive. Available if key exchange is set to IKEv1.
Use compression	checkbox	If selected a compression ability will be proposed to the peer.
DPD action	selector	Controls the use of dead peer detection protocol, values: • none – default, disables sending of DPD messages. • clear – the connection closed with no action. • hold – keeps description, tries re-negotiate connection on demand. • restart – will try to re-negotiate immediately.

Item	Type	Description
DPD delay	string	Time interval in seconds between peer check. Default 30.
DPD timeout	string	Time in seconds after which peer consider to be unusable. IKEv1 only. Default 150.
Key lifetime	string	Lifetime of data channel in seconds . Default 10800.
IKE lifetime	string	Lifetime of keying channel in seconds. Default 3600.

Auxiliary settings

Phase 1 proposals - IKE/ISAKMP cipher suite components.

Item	Type	Description	Note
Encryption algorithm	selector	Encryption algorithm – 3DES, AES128, AES192, AES256.	required
Hash algorithm	selector	Hash algorithm – MD5, SHA1, SHA256, SHA384 or SHA512.	required
DH exponentiation	selector	Specifies Diffie-Hellman groups – 1,2,5,14,15,16,18	required

Phase 2 proposals - ESP cipher suite components

Item	Type	Description	Note
Encryption algorithm	selector	Encryption algorithm – 3DES, AES128, AES192, AES256.	required
Hash algorithm	selector	Hash algorithm – MD5, SHA1, SHA256, SHA384 or SHA512.	required
DH exponentiation	selector	Specifies Diffie-Hellman groups – 1,2,5,14,15,16,18	optional

The following specification and topology map corresponds to settings used in further configuration walk-through example.

Creating a connection description

Site-to-Site VPN scenario

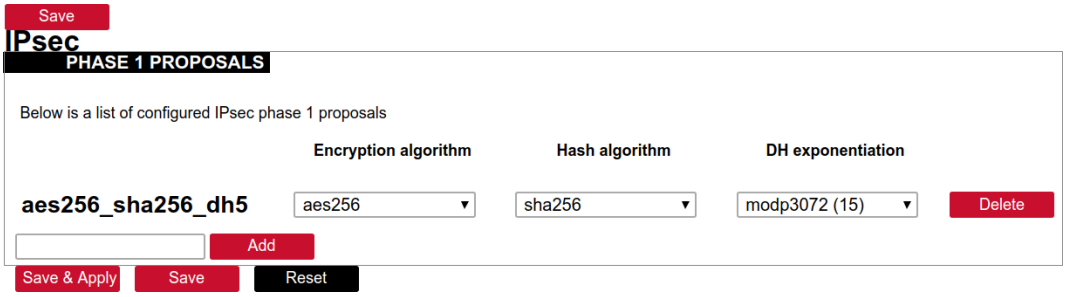
VPN connection details

Tunnel: demoo					
1	IPSec peer	*ipsec.vpn.net*	7	IKE authentication	*aes256*
2	Pre-shared key	*thebigsecret*	8	IKE hash	*sha256*
3	Mode	*tunnel*	9	IKE DH group	*5 (modp1536)*
4	Remote network	*10.10.10.0/24*	10	ESP authentication	*aes128*
5	Local network	*10.10.12.0/24*	11	ESP hash	*sha1*
6	Local ID	*wcclite*			

If auxiliary data is needed, it is recommended to check or define it first.

Creation of Phase 1 proposal

- Enter section “Phase 1 proposals”.
- Create a new record by assigning new name, for example “aes256-sha256-dh5” and click the button “Add”.
- Choose corresponding values: encryption, hash algorithm and DH exponentiation.
- Push “save” to save the data.



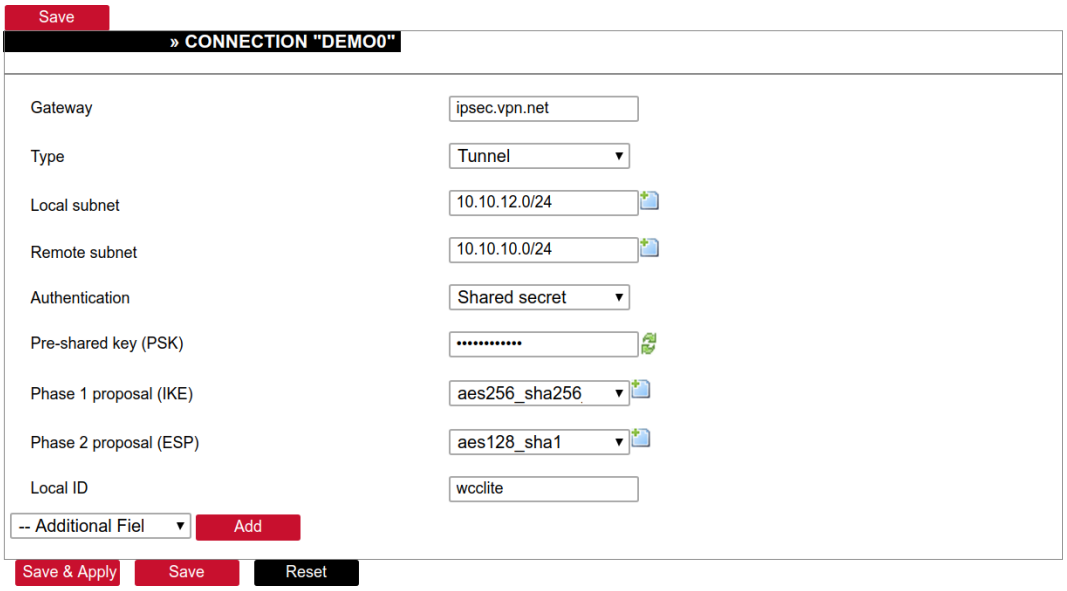
Creation of Phase 2 proposal

- Enter section “Phase 2 proposals”.
- Create a new record by assign new name for example “aes128-sha1” and click the button “Add”.
- Choose corresponding values: encryption, hash algorithm.
- Push “save” to save the data.

Creation of tunnel definition

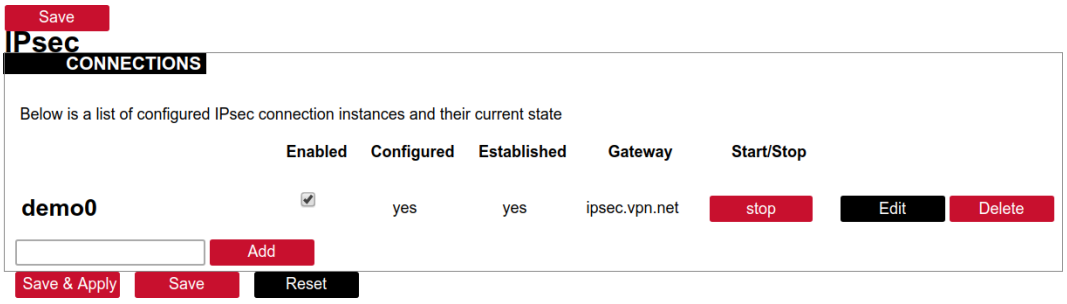
- Enter section connections

- Create a new record by assigning new name (e.g.“demo0”) and clicking “Add”.
- Call a detail form by pushing the button “edit”.
- Enter peer address into “Gateway”: “ipsec.vpn.net”.
- Ensure “Type” is set to: “Tunnel”.
- Fill local subnet to: 10.10.12.0/24.
- Fill remote subnet to: 10.10.10.0/24.
- Make sure authentication is set to: “Shared secret”.
- Enter Pre-shared key (PSK): thebigsecret.
- “Phase 1 proposal (IKE)”, choose a value: `aes256_sha256_dh5`.
- “Phase 2 proposal (ESP)”, choose a value: `aes128_sha1`.
- Locate combo box “additional field”, select “Local ID”, then set value to: wcclite.
- Push “Save”.



Activating the tunnel

- Return to the section “connections”.
- Check the checkbox “Enabled”.
- Push the button “save & apply”.
- Examine indicator “configured”, it should be “yes”, if not, review settings just entered.
- The tunnel should be prepared for operation and will be established on demand.
- Optionally, it is possible to establish tunnel operation by pressing button “start”.



8.6.3 L2TP/IPsec

Because of the lack of confidentiality inherent in the L2TP protocol, it is often implemented along with IPsec. This is referred to as L2TP/IPsec, and is standardized in IETFRFC 3193. The process of setting up an L2TP/IPsec VPN is as follows:

- Negotiation of IPsec security association (SA), typically through Internet key exchange (IKE). This is carried out over UDP port 500, and commonly uses either a shared password (so-called ”pre-shared keys”), public keys, or X.509 certificates on both ends, although other keying methods exist.
- Establishment of Encapsulating Security Payload (ESP) communication in transport mode. The IP protocol number for ESP is 50 (compare TCP’s 6 and UDP’s 17). At this point, a secure channel has been established, but no tunneling is taking place.
- Negotiation and establishment of L2TP tunnel between the SA endpoints. The actual negotiation of parameters takes place over the SA’s secure channel, within the IPsec encryption. L2TP uses UDP port 1701.

When the process is complete, L2TP packets between the endpoints are encapsulated by IPsec. Since the L2TP packet itself is wrapped and hidden within the IPsec packet, no information about the internal private network can be gathered from the encrypted packet. Also, it is not necessary to open UDP port 1701 on firewalls between the endpoints, since the inner packets are not acted upon until after IPsec data has been decrypted and stripped, which only takes place at the endpoints. A potential point of confusion in L2TP/IPsec is the use of the terms tunnel and secure channel. The term tunnel refers to a channel which allows untouched packets of one network to be transported over another network. In the case of L2TP/PPP, it allows L2TP/PPP packets to be transported over IP. A secure channel refers to a connection within which the confidentiality of all data is guaranteed. In L2TP/IPsec, first IPsec provides a secure channel, then L2TP provides a tunnel.

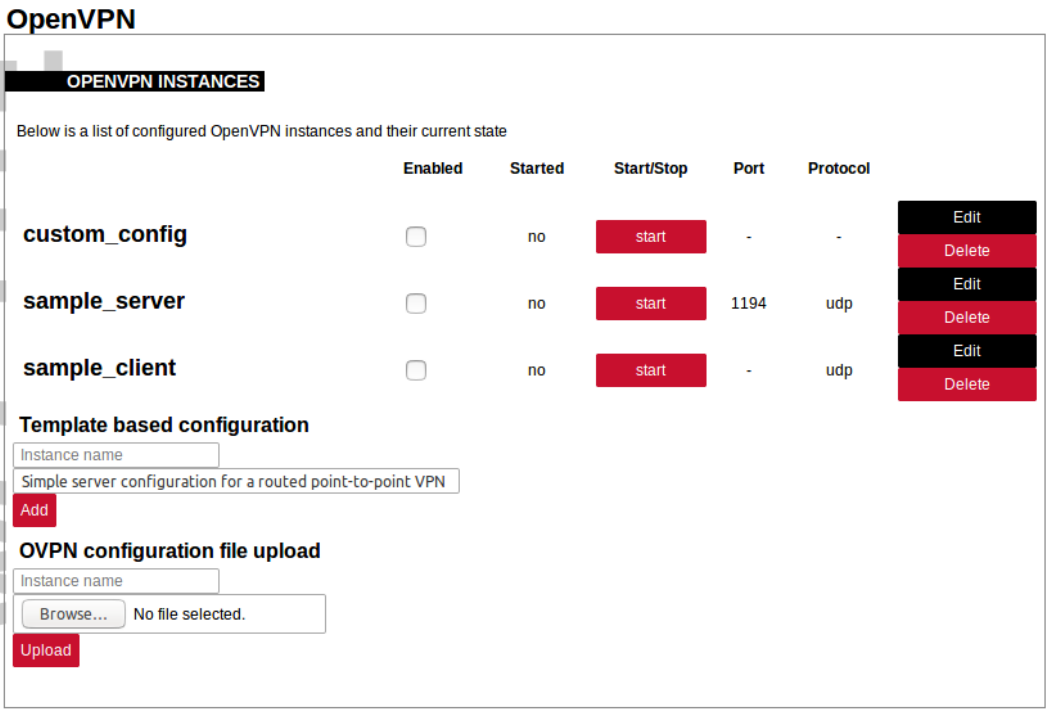
Refer to 8.7.9 for setting up L2TP network.

8.6.4 OpenVPN

OpenVPN Instances

The primary goal is to get a working WCC Lite tunnel and establish a basic platform for further customisation. Most users will require further configuration tailored to their individual needs. If you are creating an OpenVPN server (either type), you must create security certificates using the instructions below. If you are using OpenVPN as a client, the required certificates should have been provided with your configuration details. OpenVPN can be configured either by using WCC Lite Web interface or uploading the OVPN file containing necessary parameters. OpenVPN will automatically attempt to load all *.conf files placed in the /etc/openvpn folder.

Several OpenVPN *recipes* are suggested containing most used configurations that may only require minor changes. If a user intends setting up OpenVPN without OVPN file, it is highly advised to use these recipes and tweaking them up to individual needs.



OpenVPN instances page contains parameters to be configured.

Enabled: Flag to specify if a particular configuration should be enabled;

Started: Specifies if a particular configuration has been started by OpenVPN;

Start/*Stop*: Button to manually start or stop any configured tunnels;

Port: Specifies the listening port of this service;

Protocol: A standard that defines how to establish and maintain a network connection: UDP - User Datagram Protocol, TCP - Transmission Control Protocol.

More parameters for every instance can be changed by pressing *Edit* button, configuration can be removed with *Delete* button. Pressing *Edit* takes the user to main configuration screen containing the options usually used in particular OpenVPN *recipes*. To do more specific changes user should further select *Switch to advanced configuration*.

OVPN files contain configuration in a textual form therefore changing parameters requires having prior knowledge about different OpenVPN parameters. It is advised to used OVPN files, however, if configuration has been pre-built beforehand and is used without further changes.

8.6.5 ser2net

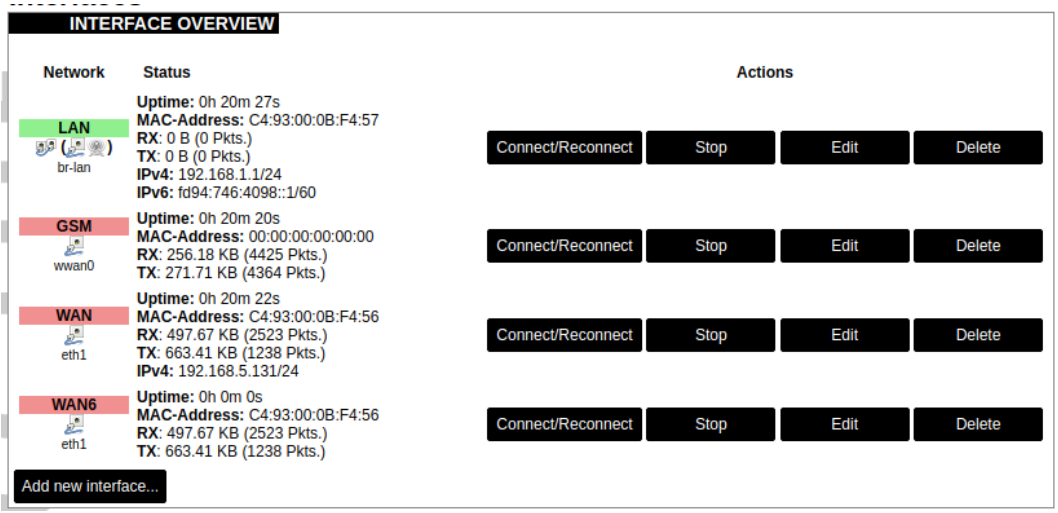
The ser2net daemon allows telnet and tcp sessions to be established with a device's serial ports. The program comes up normally as a daemon, opens the TCP ports specified in the configuration file, and waits for connections. Once a connection occurs, the program attempts to set up the connection and open the serial port. If another user is already using the connection or serial port, the connection is refused with an error message.

8.7 Network

The page shows information about current interface status, its configurations, provides various interface, network properties configuration capabilities and contains the following subsections:

- *INTERFACES*: shows information about current interface status, allows to create new and configure them.
- *WIRELESS*: shows information about wireless radio stations, covers physical settings of the wireless hardware.
- *DHCP AND DNS*: allows management of DHCP and DNS servers.
- *HOSTNAMES*: allows management of host names.
- *STATIC ROUTES*: allows management of IPv4 and IPv6 static routes.
- *FIREWALL*: allows management of firewall zones and various firewall properties.
- *DIAGNOSTICS*: provides network diagnostics utilities.
- *GSM*: allows management of gsm modem and SIM cards.

8.7.1 Interfaces



Current information and status of various network interfaces (GSM, LAN, WAN).

Uptime: Current interface uptime in hours, minutes and seconds.

MAC address: Physical interface address.

RX: Received data in bytes (packet count).

TX: Transmitted data in bytes (packet count).

IPv4: Internet protocol version 4 address.

IPv6: Internet protocol version 6 address.

In addition to the network interface status, several actions may be performed:

Connect/*Reconnect*: Connect to configured interface network if it does not do it automatically. If it already connected to the network it will be trying to reconnect to it.

Stop: Shutdown interface. If you are connected through this interface the connection may be lost.

Edit: Edit interface settings.

Delete: Delete interface.

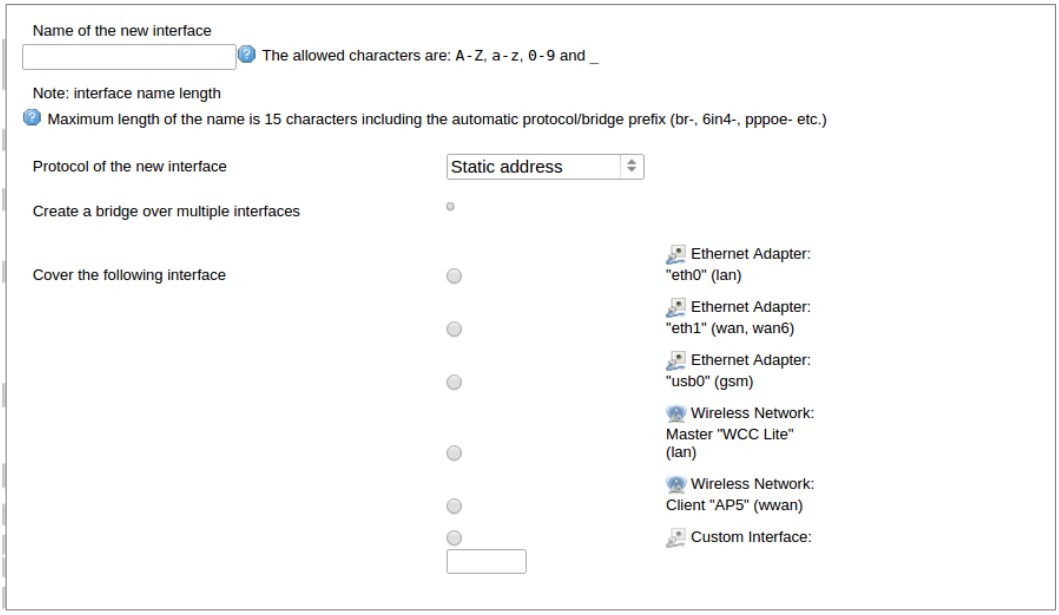
Add new interface: Adding new Ethernet, GSM or wireless interface with the custom name, protocol and etc.

Table 7: Default interface settings

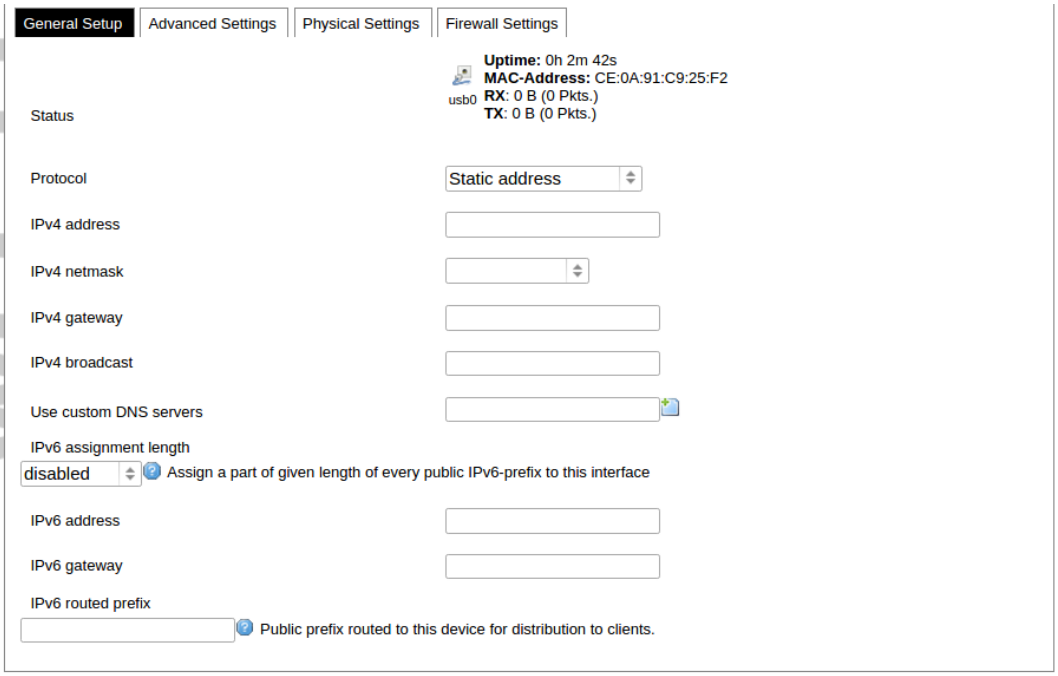
	eth0	eth1
Type	Static	DHCP
Address	192.168.2.1	
Subnet mask	255.255.255.0	
Gateway		

Changes will only take effect after device reboots.

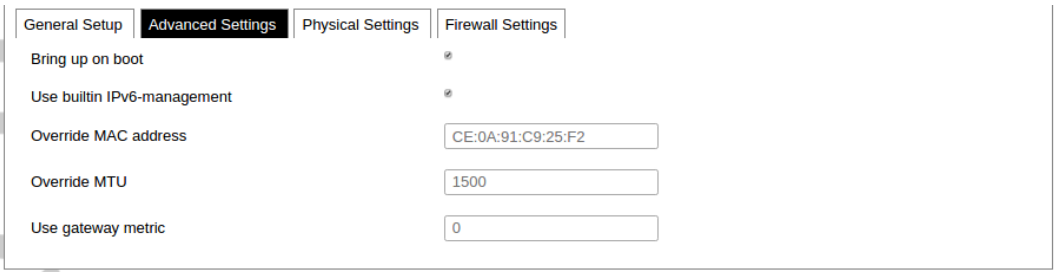
Network interfaces can be configured on the common page, which can be accessed through add new interface or edit button.



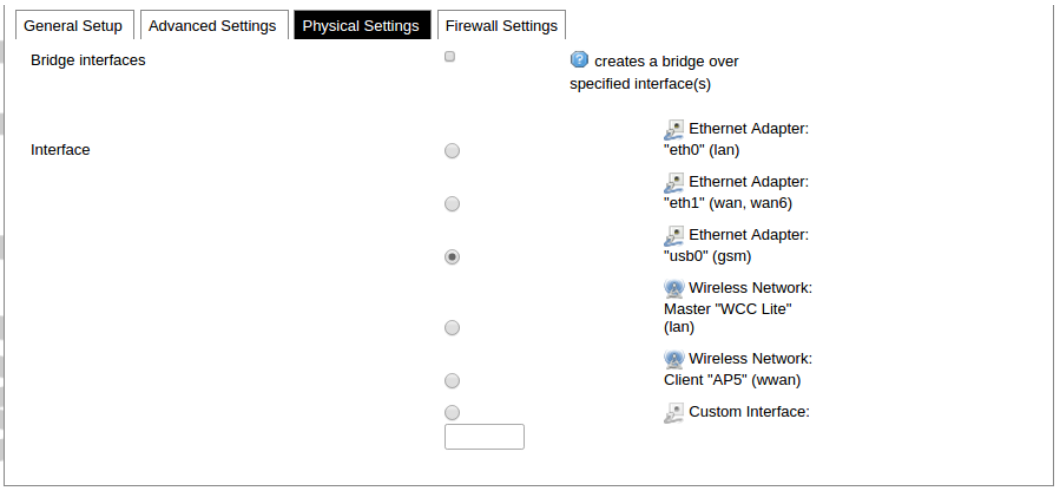
The following options can be defined in the interface creation panel: name of the interface, protocol, coverage of a particular interface or bridging with other interfaces. After the general setup is done, more detailed settings can be set.



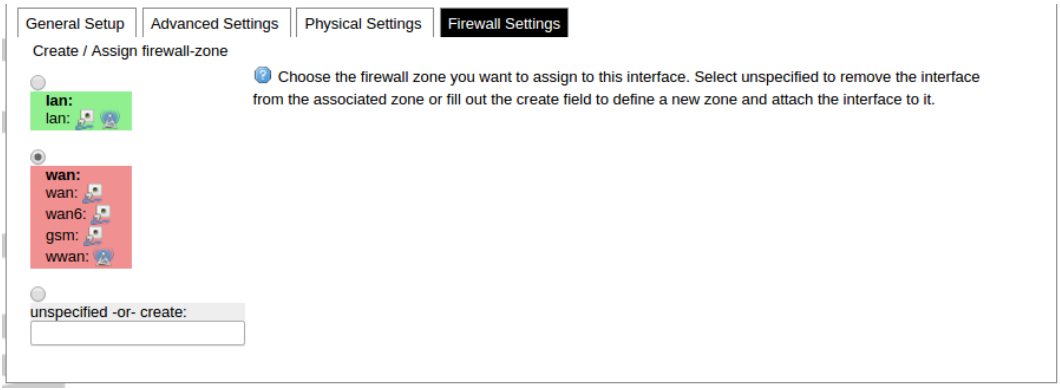
General common interface setup panel.



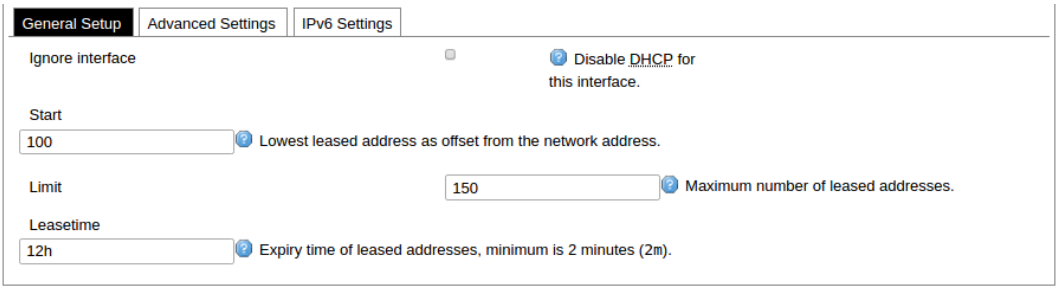
Advanced common interface setup panel.



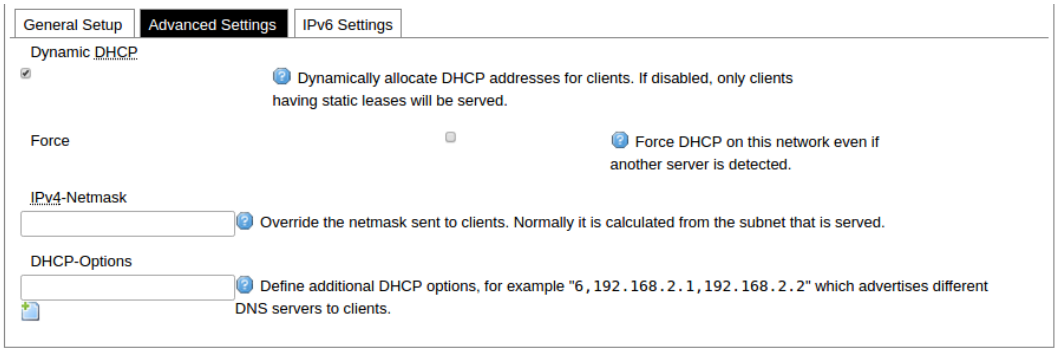
Physical common interface setup panel.



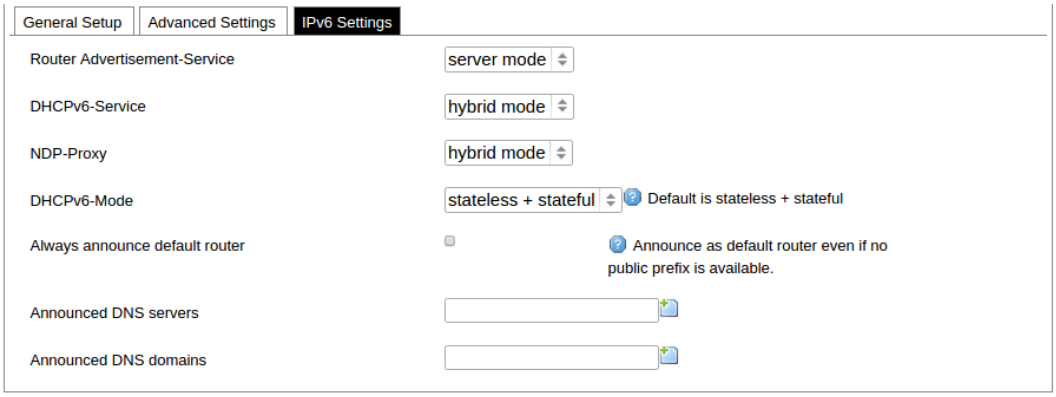
Firewall common interface setup panel.



DHCP server general setup panel.

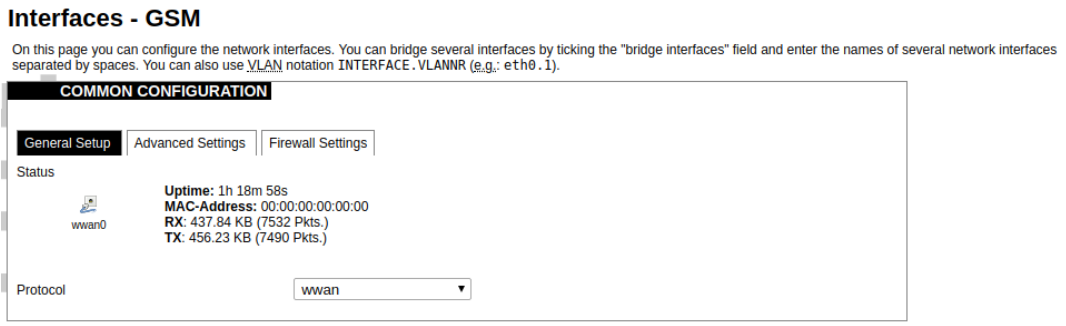


DHCP server advanced setup panel.



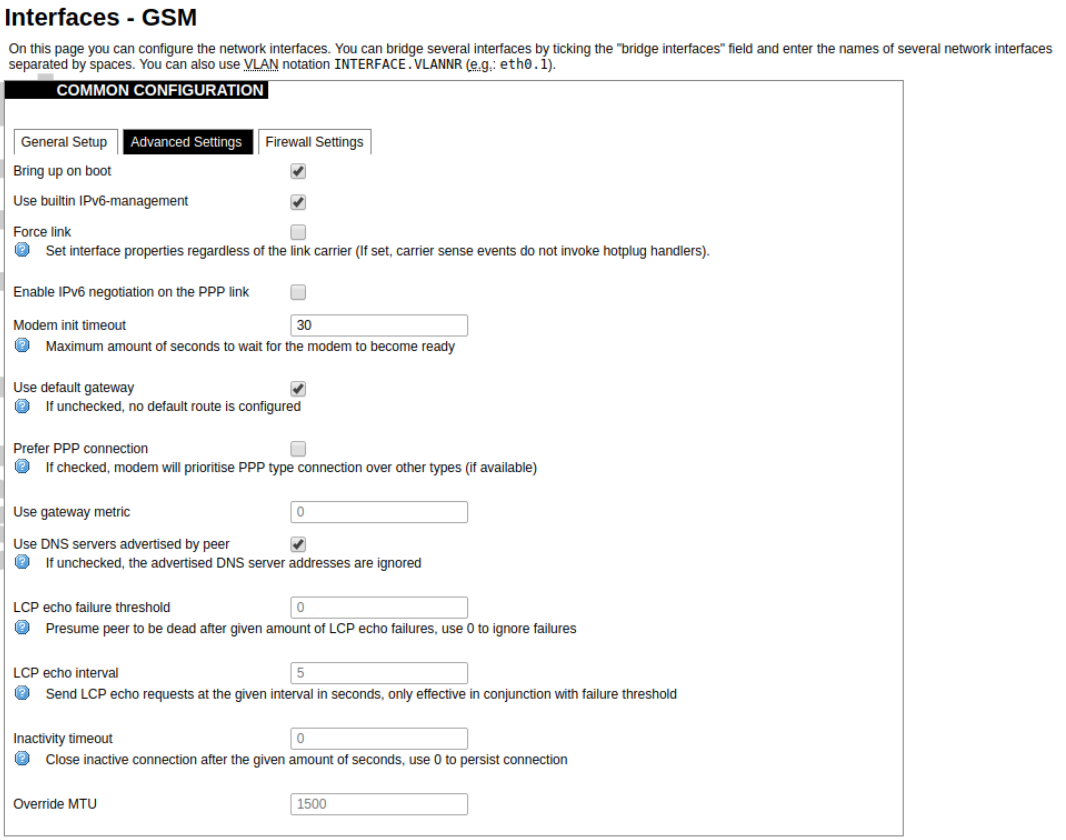
DHCP server IPv6 settings setup panel.

GSM



General Settings Information tab. Gives you name of physical GSM interface, lets you choose protocol (not recomended!).

Note: Make sure you won't change GSM interafce's protocol, which is set by default to WWAN. Changing this parameter will lead to undefined GSM modem behaviour.



Advanced Settings tab enables user to configure advanced settings for mobile communication. It includes the following options:

Bring up on boot: Checkbox to start a GSM interface on startup;

Use builtin IPv6-management: Checkbox to select if the device is going to use its own tools to manage IPv6 transport layer messages;

Force link: Specifies whether IP address, route, and gateway are assigned to the interface regardless of the link being active or only after the link has become active; when active, carrier sense events do not invoke hotplug handlers;

IPv6 support: User can select if IPv6 support is handled automatically, manually or disabled altogether;

Modem init timeout: Maximum amount of seconds before the device gives up on finishing initialization;

Use default gateway: Uses the default gateway obtained through DHCP. If left unchecked, no default route is configured;

Prefer PPP connection: If ,the modem, supports PPP and any other communication protocol (e.g. QMI, RNDIS and etc.), prioritise PPP type connection;

Use gateway metric: The WAN configuration by default generates a routing table entry. In this field you can alter the metric of that entry. Higher metric means higher priority;

Use DNS servers advertised by peer: Uses DNS servers obtained from DHCP. If left unchecked, the advertised DNS server addresses are ignored;

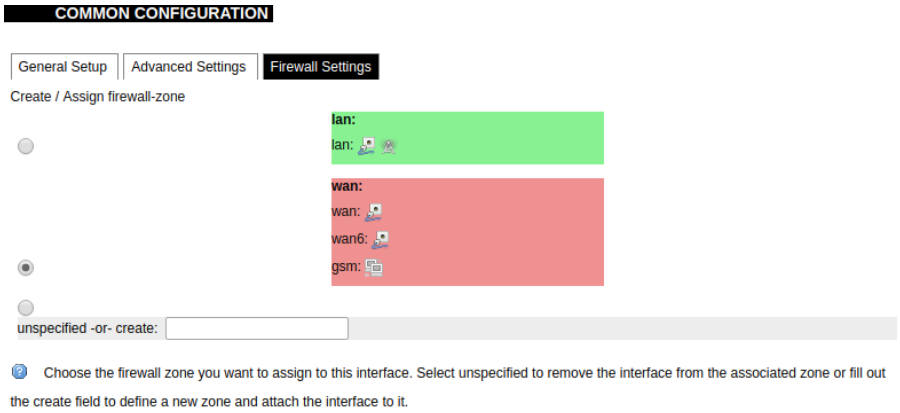
LCP echo failure threshold: LCP (link control protocol) is a part of PPP (Point-to-Point Protocol) and helps to determine the quality of data transmission. If enough failures happen, LCP presumes link to be dead. 0 disables failure count checking;

LCP echo interval: Determines the period of LCP echo requests. Only effective if LCP echo failure threshold is more than zero;

Inactivity timeout: Station inactivity limit in seconds: if a station does not send anything, the connection will be dropped. A value of 0 can be used to persist connection.

Override MTU: Set custom MTU to gsm interface.

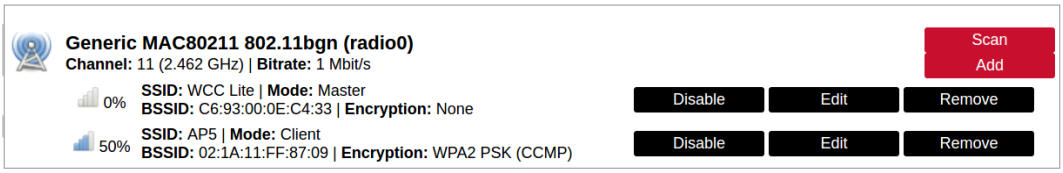
Note: If modem uses QMI connection protocol and user haven't defined custom MTU setting, the MTU on interface will be set to operator's defined MTU value.



GSM configuration ends with firewall settings. A user can assign an already defined firewall zone or create a new one.

8.7.2 Wireless

The wireless network interface parameters and configuration are described in this section.



Configured interfaces for the physical radio device.

Channel: Specifies the wireless channel to use.

Bitrate: Specifies transfer rate in Mbit/s.

SSID: The broadcasted service set identifier of the wireless network.

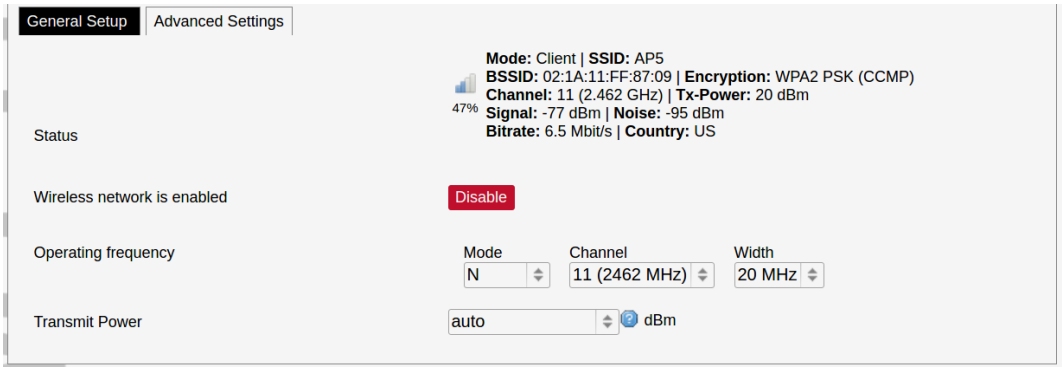
Mode: Selects the operation mode of the wireless network interface controller.

BSSID: The basic service set identification of the network, only applicable in adhoc or STA mode.

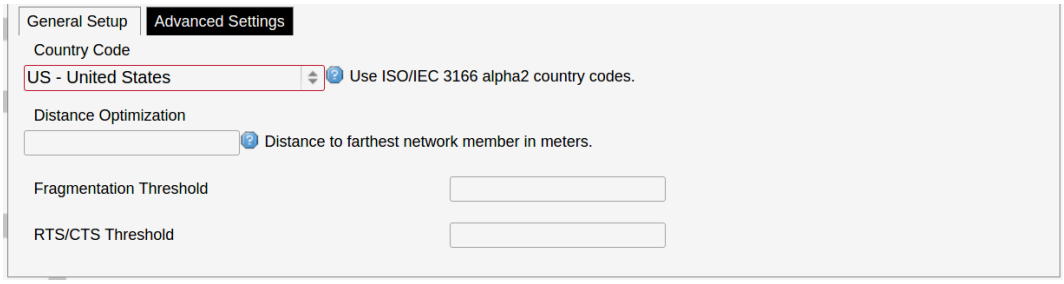
Encryption: Wireless encryption method.

List of associated wireless stations.

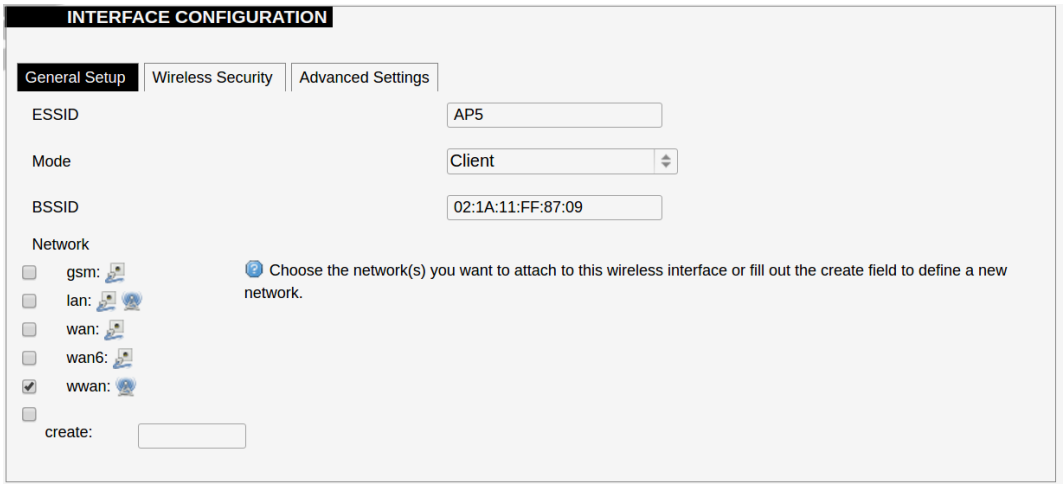
The Device Configuration section covers physical settings of the radio hardware such as channel, transmit power or antenna selection which are shared among all defined wireless networks (if the radio hardware is multi-SSID capable). Per network settings like encryption or operation mode are grouped in the Interface Configuration.



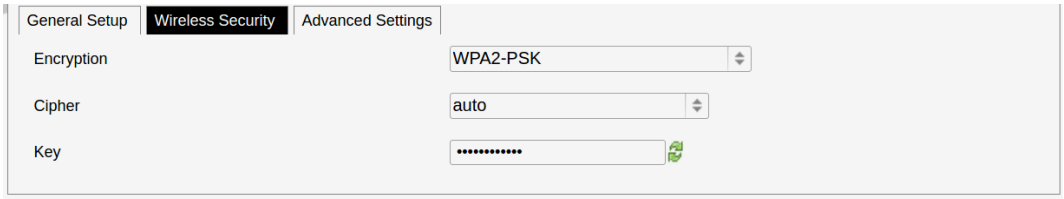
General device settings.



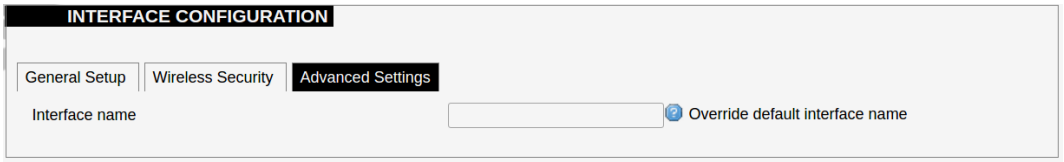
Advanced device settings.



General interface settings.



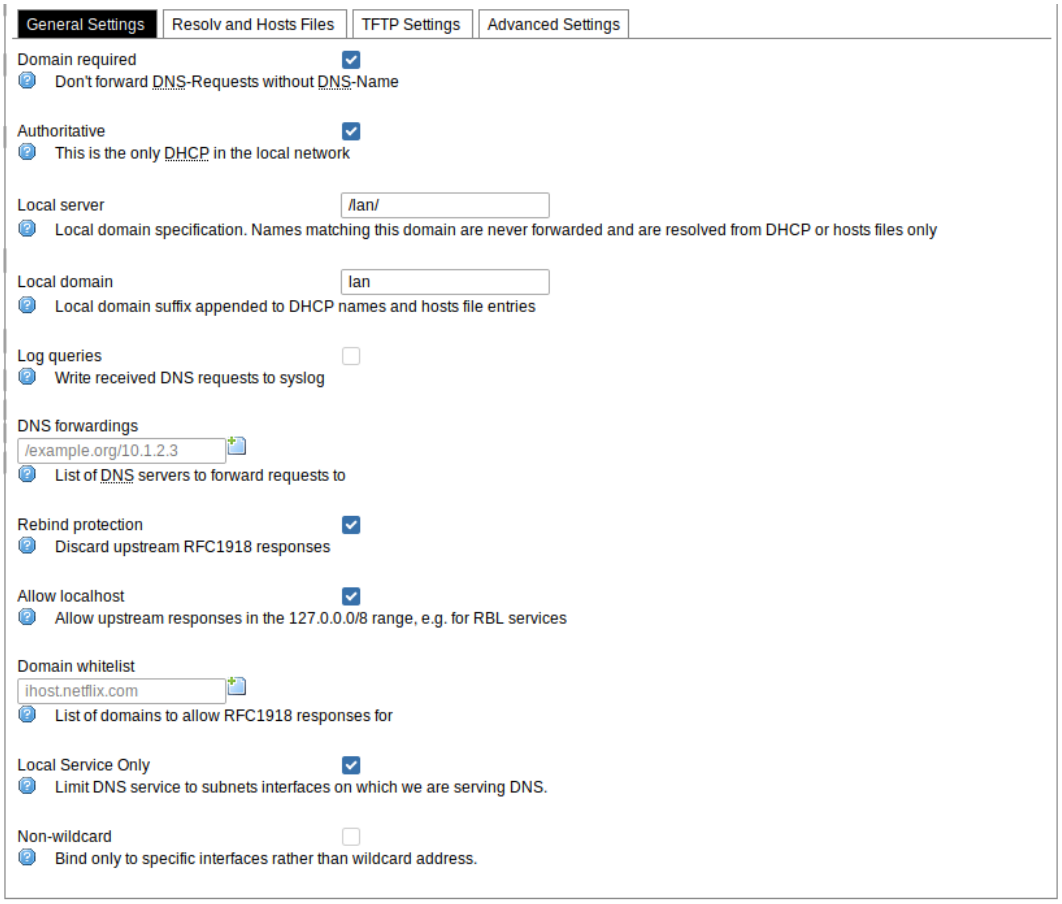
Wireless security interface settings.



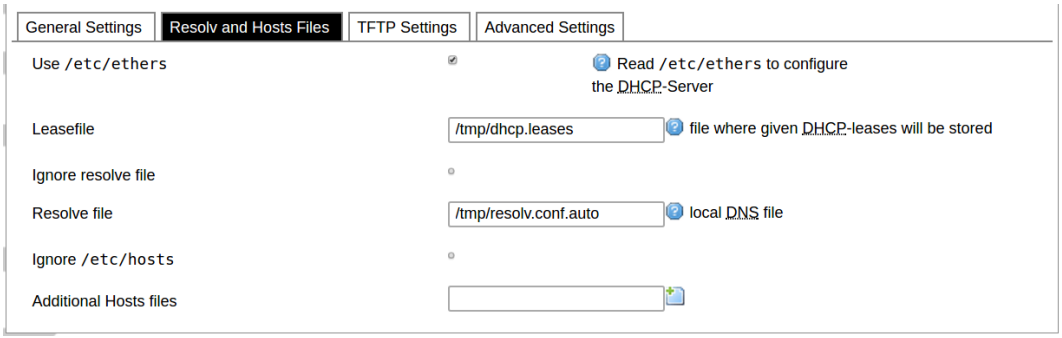
Advanced interface settings.

8.7.3 DHCP and DNS

DHCP server and DNS forward for NAT firewalls is described in this section.

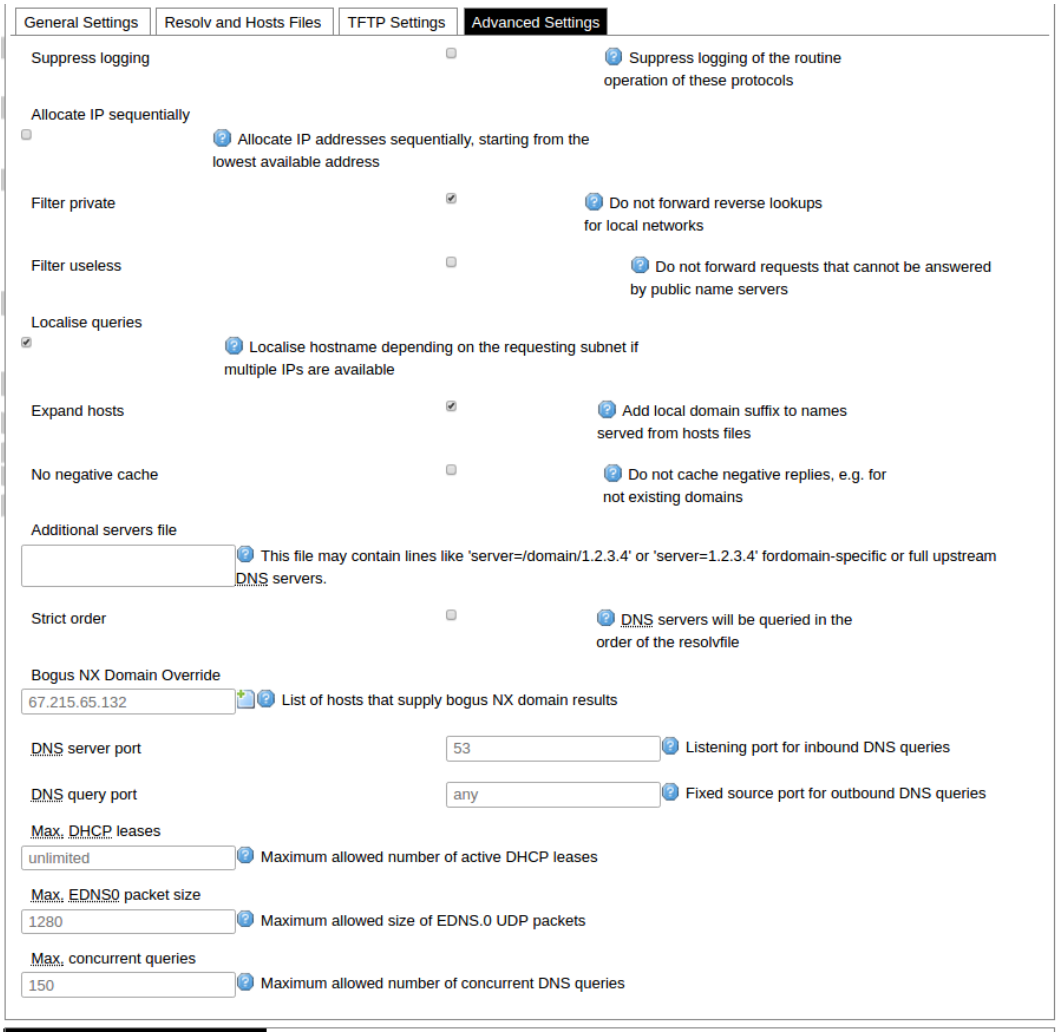


General DHCP settings.

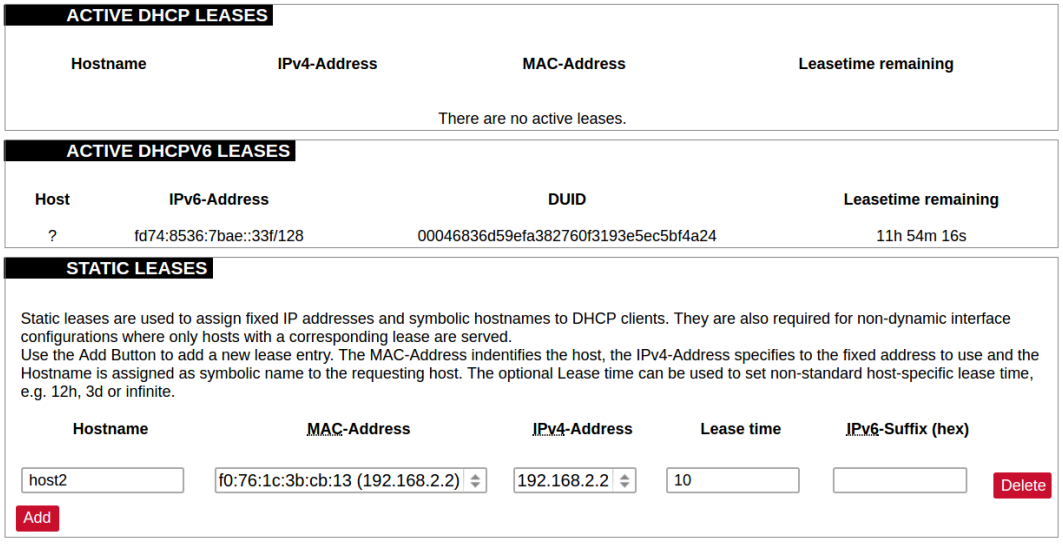


Resolve and hosts files settings.

TFTP server settings.



Advanced settings.



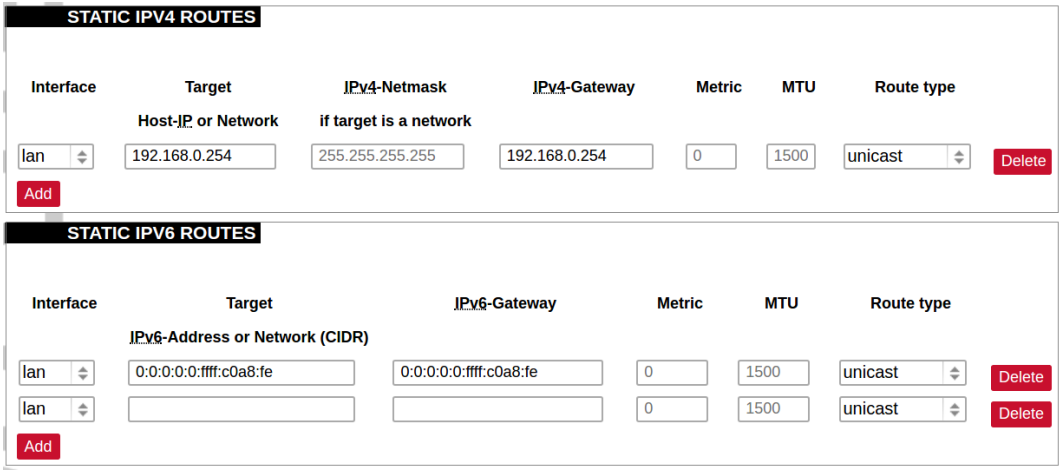
List of active DHCP and static leases. It is also possible to assign fixed IP addresses to hosts on the network, based on their MAC (hardware) address.

8.7.4 Hostnames

List of existing host names. Addition or deletion is allowed for the user.

8.7.5 Static routes

Routes specify over which interface and gateway a certain host or network can be reached.



Current IPv4 and IPv6 static routes configuration.

Interface: Lets to chose for which interface static route is created.

Target: Defines target host IP or network.

IPv4 Netmask: Defines netmask if the target is a network.

IPv4/IPv6 Gateway: Defines IPv4 or IPv6 gateway.

Metric: Specifies the route metric to use for the route.

MTU: Maximum Transmit/Receive Unit, in bytes.

Route type: All incoming packets can be: accepted, rejected, dropped.

8.7.6 Firewall

This subsection is divided into four categories: general settings, port forwards, traffic rules and custom rules.

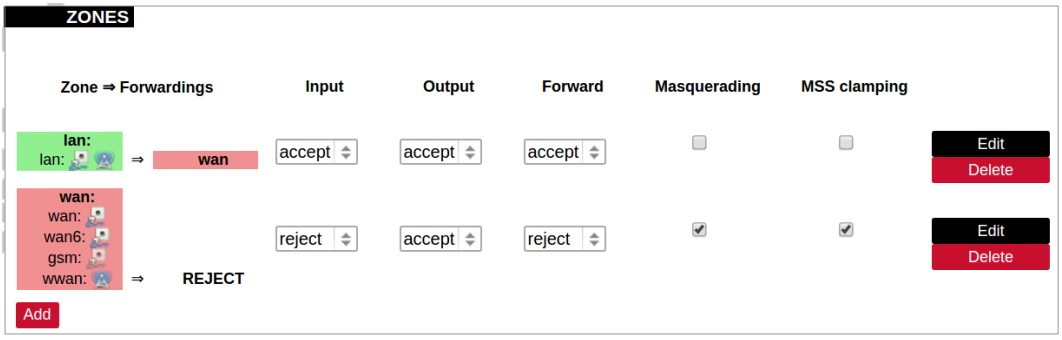
General settings

General Settings for firewall can be changed in *General Settings* screen. These settings are defined as follows:

Input: All incoming packets can be: accepted, rejected, dropped.

Output: All outgoing packets can be: accepted, rejected, dropped.

Forward: All packets being sent to another device can be: accepted, rejected, dropped.



Additional zones for firewall can be created, edited or deleted.

Zone => Forwardings: Defines zones and their traffic flow.

Input: All incoming packets can be: accepted, rejected, dropped.

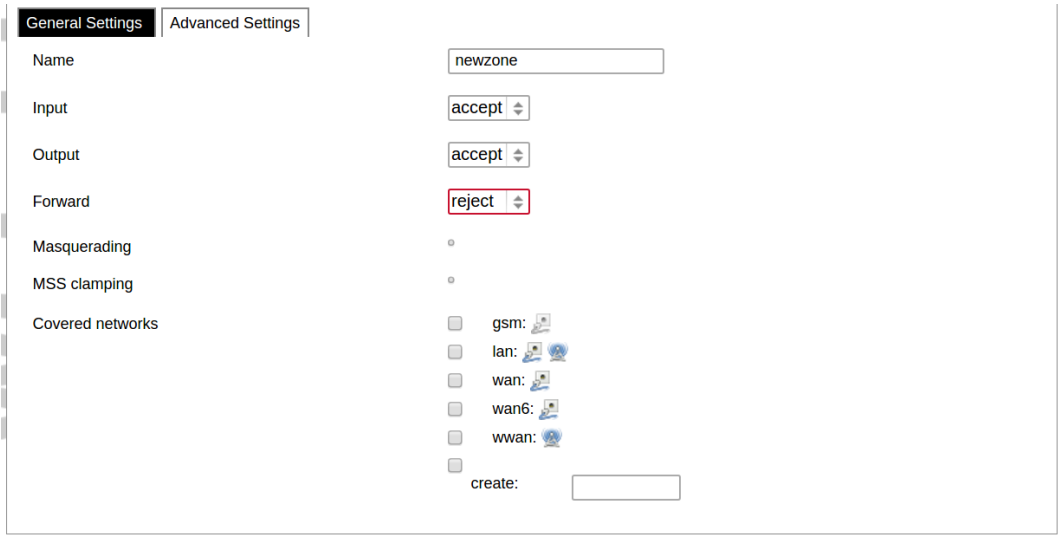
Output: All outgoing packets can be: accepted, rejected, dropped.

Forward: All packets being sent to another device can be: accepted, rejected, dropped.

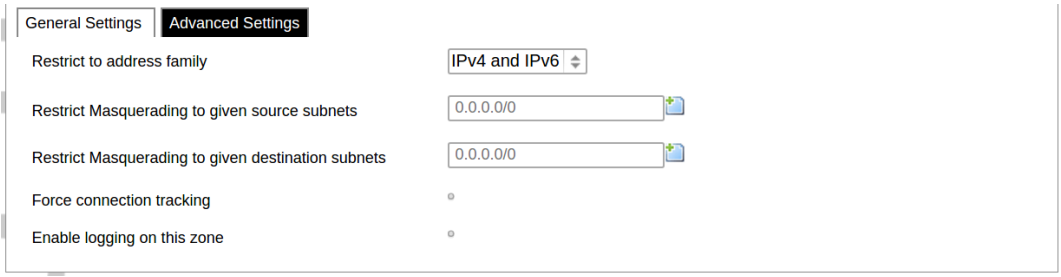
Masquerading: Allows one or more devices in a zones network without assigned IP addresses to communicate with the Internet.

MSS clamping: Change the maximum segment size (MSS) of all TCP connections passing through this zone with MTU lower than the Ethernet default of 1500.

Additional actions can be performed with zones: add, edit, delete.



Common properties of newly created or edited zones chan be edited in this panel. The input and output options set the default policies for traffic entering and leaving this zone while the forward option describes the policy for forwarded traffic between different networks within the zone. Covered networks specify which available networks are members of this zone.

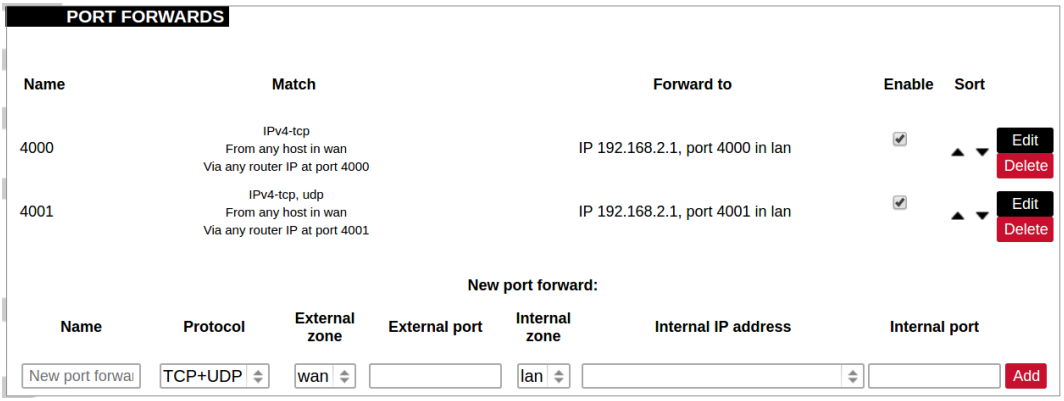


Advanced settings of new created or edited zone. Restrict to address family option defines to what IP families the zone belongs to IPv4, IPv6 or both. Restrict masquerading to given source/destination subnets defines one or more subnets for which the masquerading option is applied to. Connection tracking and logging options enable additional information gathering on the zone.

Controls of the forwarding policies between new/edited zone and other zones. Destination zones cover forwarded traffic originating from the new/edited zone. Source zones match forwarded traffic

from other zones targeted at the new/edited zone. The forwarding rule is unidirectional, e.g. a forward from LAN to WAN does not imply a permission to forward from WAN to LAN as well.

Port forwards



Port forwarding allows remote computers on the Internet to connect to a specific computer or service within the private LAN. It is done in a way of routing network packets within a private network created by the device. Settings for the port forwarding of the device are defined as follows:

Name: The name of the port forwarding rule.

Match: Informs what port forward is matched to.

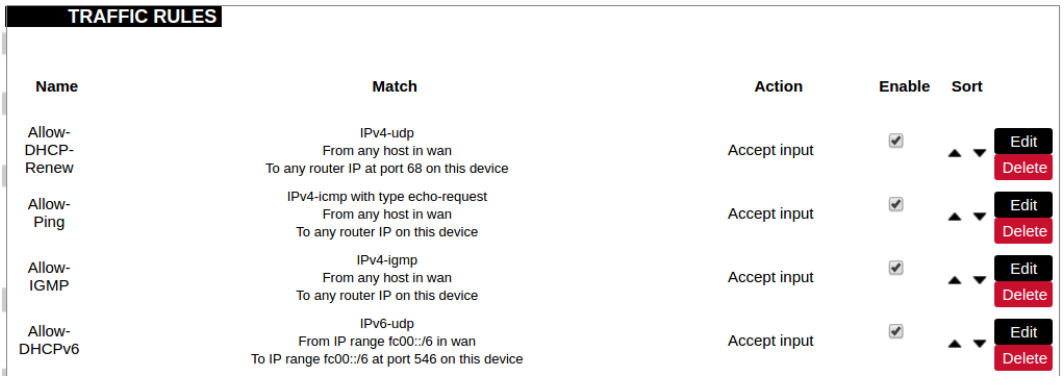
Forward to: Informs where the port is forwarded to.

Enable: Enable (checked) or disable port forward.

Sort: Allows to sort port forwarding.

The user can add, edit or delete port forwarding rules.

Traffic rules



Traffic rules which define policies for packets traveling between different zones.

Name: The name of the traffic rule.

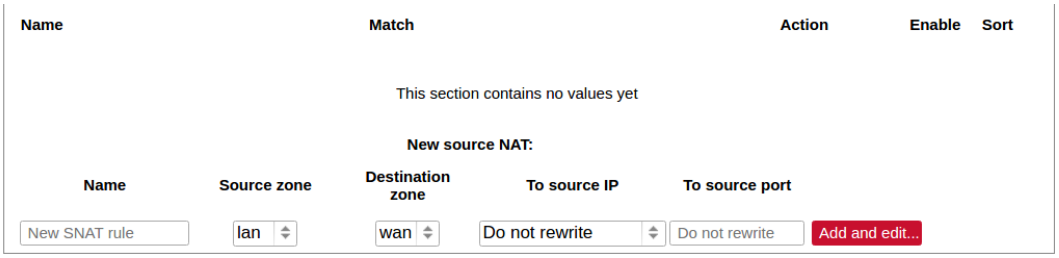
Match: Informs what ICMP types are matched.

Action: Informs what action would be performed.

Enable: Enable (checked) or disable the rule.

Sort: Allows to sort rules.

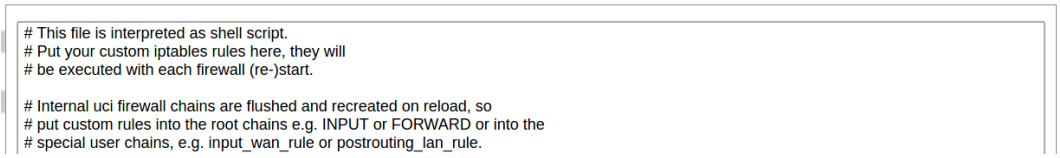
The user can add, edit or delete traffic rules. For every rule can be defined these options: name, restrict to address family, protocol, match ICMP type, source and destination zones, source MAC, IP addresses and port, destination IP address and port, action and extra arguments, month and weekdays for which rule will apply, start/stop dates and times, time in UTC.



Source NAT, which is a specific form of masquerading which allows fine grained control over the source IP used for outgoing traffic, for the example to map multiple WAN addresses to internal subnets.

The user can add, edit or delete source NAT rules. For every rule can be defined these options: name, protocol, source and destination zones, source, destination, SNAT IP addresses, ports, extra arguments, month and weekdays for which rule will apply, start/stop dates and times, time in UTC.

Custom rules

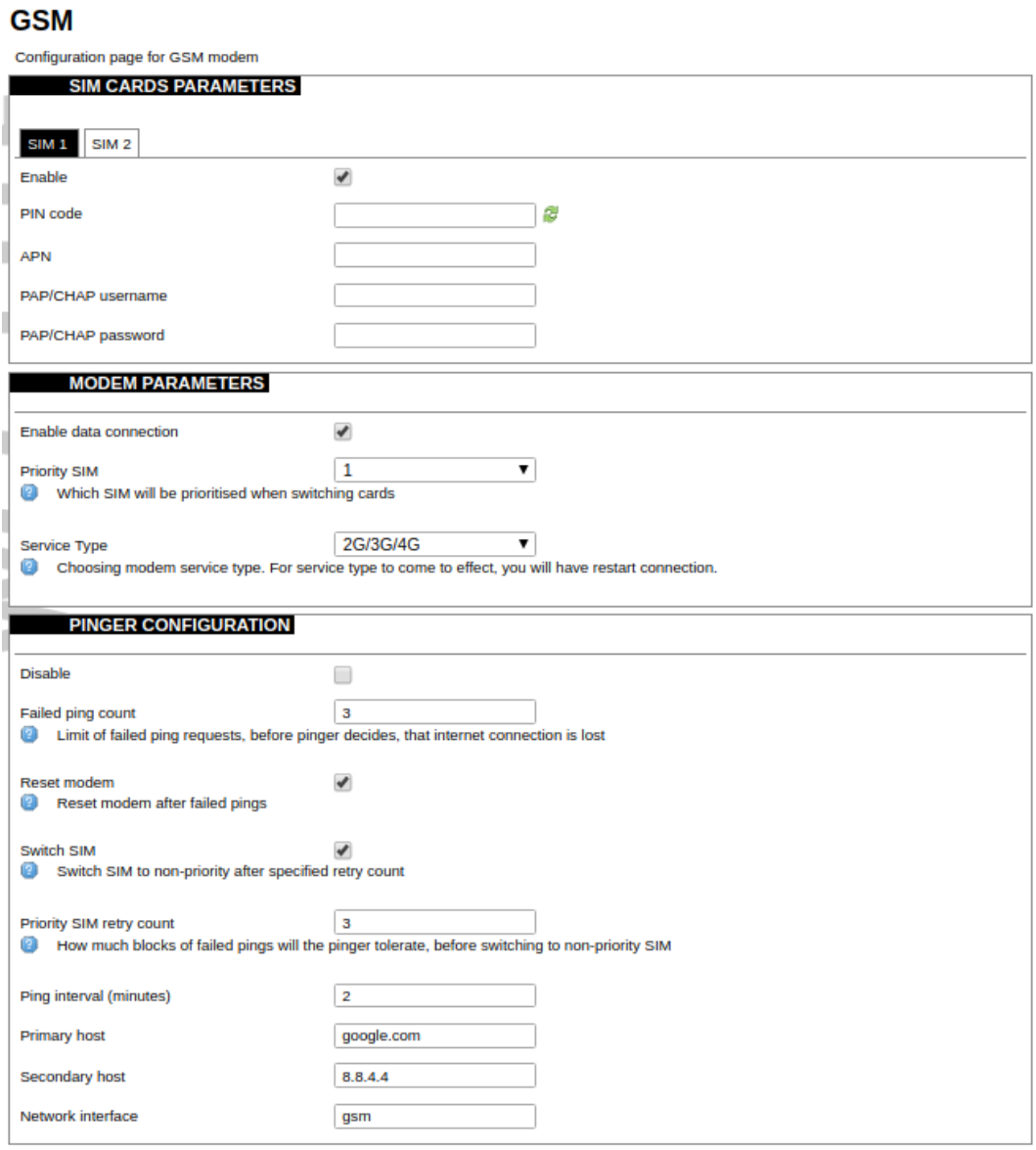


Custom rules allow to executing arbitrary iptables commands which are not otherwise covered by the firewall framework. The commands are executed after each firewall restart, right after the default ruleset has been loaded.

8.7.7 Diagnostics

Diagnostics tools which can be used to diagnose some of the networking problems: ping, traceroute and nslookup.

8.7.8 GSM



SIM cards parameters

Parameters for SIM card. If single SIM modem is used, there won't be "SIM 1" and "SIM 2" tabs.

Enable: Enable or disable this SIM card.

PIN code: PIN code to use on that SIM card.

APN: APN to use on that SIM car.

PAP/CHAP username: Username (if configured).

PAP/CHAP password: Password (if configured).

Modem parameters

Enable data connection: Enable or disable data connection through gsm modem.

Priority SIM: Primary SIM card (if Dual SIM modem is used). Mainly used for pinger configuration.

Service Type: Which radio access technology will be used when connecting to gsm network.

Pinger configuration

Pinger is a service which pings two hosts (primary and secondary) to check internet connection. If both of these hosts are unreachable pinger will wait and restart modem (or switch SIM card, if Dual-SIM modem is installed in WCC Lite)

Disable: Disable pinger functionality.

Failed ping count: Limit of failed ping requests, before pinger decides, that internet connection is lost.

Reset modem: If checked, pinger resets gsm modem after ”Failed ping count”.

Switch SIM: If checked, pinger switches SIM to non-priority after ”Priority SIM retry count”. If internet connection is not available with non-priority SIM as well, pinger switches back to priority SIM after one failed ping attempt.

Priority SIM retry count: How much blocks of failed pings will the pinger tolerate, before switching to non-priority SIM.

Ping interval (minutes): Interval between ping requests.

Primary host: The host, that will be pinged first.

Secondary host: The host, that will be pinged second, if primary host fails.

Network interface: GSM network interface name.

GSM Pinger is used to detect the status of network connection via cellular network. This status is written to file (*/var/run/board/internet-status*) and can be configured to be sent to SCADAs. If pinger is disabled, status is always set equal to zero and should not be trusted to represent internet status. Additionally, this status is reflected in ”Status”->”GSM Status” window.

This is Pinger functionality described step by step:

- Pinger will ping primary host every 2 minutes.
- If primary host fails, pinger redirects to secondary host immediately.
- If either primary or secondary host is responding to ping requests, pinger will continue testing connection every ”Ping interval (minutes)” parameter and no further action is taken.
- If both primary and secondary host are unreachable, pinger will start pinging these hosts every ”Ping interval (minutes) / 2” minute for ”Failed ping count” times.
- If hosts are still unreachable, pinger will try to switch SIM and restart modem (if corresponding parameters are set) or will restart immediately if single SIM modem is used.
- SIM card is switched to non-priority SIM after ”Priority SIM retry count” failed modem restarts with priority SIM. If non-priority SIM fails, it is swtiched to priority SIM in next pinger action.

Dual SIM start procedure

Table below shows, which card is expected on boot, when selectiom is made between Enable/Disable SIM cards and Primary card.

Table 8: Default SIM on boot

SIM 1 Enabled	SIM 2 Enabled	Priority SIM	SIM on boot
X		1	1
X		2	1
	X	1	2
	X	2	2
X	X	1	1
X	X	2	2
		1	Undefined
		2	Undefined

8.7.9 Layer 2 Tunneling Protocol

In computer networking, Layer 2 Tunneling Protocol (L2TP) is a tunneling protocol used to support virtual private networks (VPNs) or as part of the delivery of services by ISPs. It does not provide any encryption or confidentiality by itself. Rather, it relies on an encryption protocol that it passes within the tunnel to provide privacy.

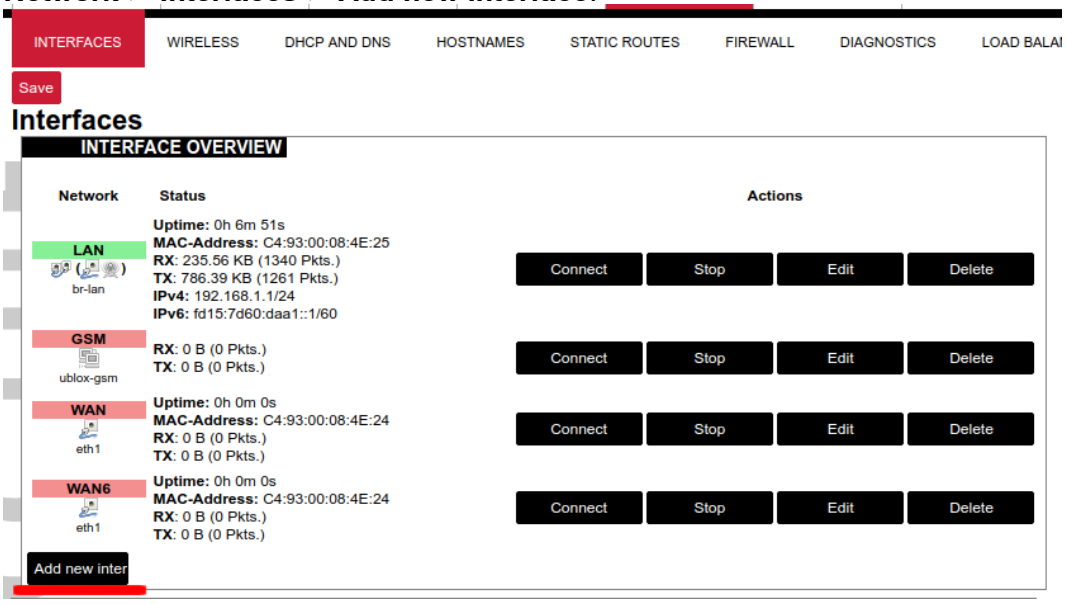
Description

The entire L2TP packet, including payload and L2TP header, is sent within a User Datagram Protocol (UDP) datagram. It is common to carry PPP sessions within an L2TP tunnel. L2TP does not provide confidentiality or strong authentication by itself. IPsec is often used to secure L2TP packets by providing confidentiality, authentication and integrity. The combination of these two protocols is generally known as L2TP/IPsec (discussed below). The two endpoints of an L2TP tunnel are called the LAC (L2TP Access Concentrator) and the LNS (L2TP Network Server). The LNS waits for new tunnels. Once a tunnel is established, the network traffic between the peers is bidirectional. To be useful for networking, higher-level protocols are then run through the L2TP tunnel. To facilitate this, an L2TP session (or 'call') is established within the tunnel for each higher-level protocol such as PPP. Either the LAC or LNS may initiate sessions. The traffic for each session is isolated by L2TP, so it is possible to set up multiple virtual networks across a single tunnel. MTU should be considered when implementing L2TP. The packets exchanged within an L2TP tunnel are categorized as either control packets or data packets. L2TP provides reliability features for the control packets, but no reliability for data packets. Reliability, if desired, must be provided by the nested protocols running within each session of the L2TP tunnel. L2TP allows the creation of a virtual private dialup network (VPDN) to connect a remote client to its corporate network by using a shared infrastructure, which could be the Internet or a service provider's network.

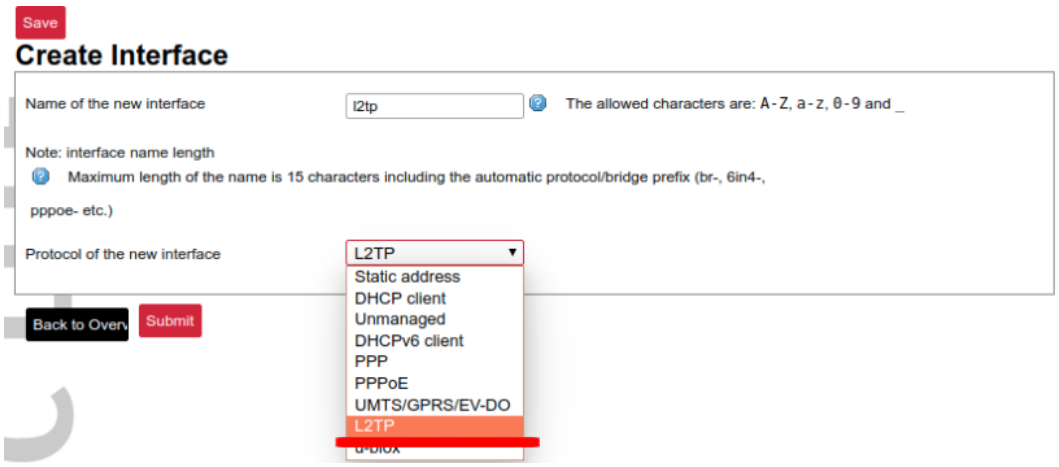
Setting up L2PT interface

In order to create a L2TP tunnel following steps are required:

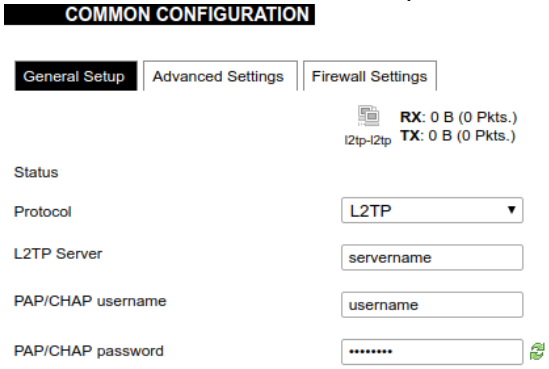
1. Go to **Network > Interfaces > Add new interface**:



2. Enter interface name and selet L2TP protocol:



3. Enter server name and authorization parameters:



4. Save and apply the new configuration. A new network interface will appear.

8.8 Logout

To log out of the device graphical user interface a logout button in interface's upper right corner should be pressed. A user is automatically disconnected after ten minutes of inactivity. This ensures that the device would not be suspect to any deliberate damage made by unauthorized access.

9 API

The firmware of the WCC Lite features a built-in API which is accessible via the web interface. As of version 1.2.11, it does not implement any access restriction features apart from those provided by the firewall functionality.

Individual API endpoints can be enabled or disabled via the web configuration interface at Services->API. All endpoints are disabled by default.

Available API endpoints are shown in the table below.

Table 9: Available API functions

Path	Description
/api/version	Version of the API
/api/actions	List of available points
/api/syncVersion	Version of the sync service
/api/sync	Protocol hub configuration sync (name=”file”)*
/api/syslog	Prints out the syslog
/api/systemInfo	General system info
/api/gsmInfo	GSM modem information
/api/devices	List of configured devices
/api/device/info	Device information (name=”device_alias”)**
/api/device/tags	List of tags on particular device (name=”device_alias”)**
/api/device/tag/value	Tag value (name=”device_alias”, name=”signal_alias”)**
/api/tags	List of configured tags
/api/sysupgrade	Firmware upgrade (name=”file”)*

* Endpoints accepting files
** Endpoints accepting field data

The API accepts data and files as POST requests encoded as ”multipart/form-data”.

10 SNMP

SNMP (Simple Network Management Protocol) is an internet-standard protocol for managing devices on IP networks. SNMP exposes management data in the form of a hierarchy of variables in a MIB (Management Information Base).

WCC Lite supports SNMP service which is not added to default build of firmware but can be installed as a module. It enables user to collect data on various parameters of system:

- *CPU time* - time spent for calculations of various processes:
  user - time for user processes;
  system - time for system processes;
  idle - time spent idling;
  interrupts - time spent handling interrupts.
- *CPU load average* - CPU load average for 1, 5 and 15 minutes respectively;
- *Disk usage*:
  total - total amount of storage in the device (in kB)
  available - amount of storage available to store data (in kB)
  used - amount of storage used in the device (in KB)
  blocks used percentage - blocks (sectors) used to store data in a disk (in kB)
  inodes used percentage - the inode (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block location(s) of the object's data.
- *Memory usage* - RAM usage statistics:
  total - total amount of RAM in the device (in kB);
  available - unused amount of RAM in the device (in kB);
  shared - shared amount of RAM between multiple processes (in kB);
  buffered - refers to an electronic buffer placed between the memory and the memory controller;
  cached - a portion of memory made of high-speed static RAM (SRAM) instead of the slower dynamic RAM (DRAM) used for main memory;
- *Network interfaces:*
  MTU - maximum transmission unit to be sent over network;
  speed - rate of network transmission;
  physical address - unique MAC address assigned to a device;
  tx/*rx*: byte, packet, drop, error count;
- *System properties:*
  uptime - time since the device was turned on;
  process uptime - time since the process has been started;
  hostname - a label that is assigned to a device connected to a computer network;
  name - name of the device (if defined);
  location - location of the device (if defined).

11 DNP3

DNP3 (Distributed Network Protocol) is a set of communications protocols used between components in process automation systems. Its main use is in utilities such as electric and water companies. It was developed for communications between various types of data acquisition and control equipment. It plays a crucial role in SCADA systems, where it is used by SCADA Master Stations (a.k.a. Control Centers), Remote Terminal Units (RTUs), and Intelligent Electronic Devices (IEDs). It is primarily used for communications between a master station and RTUs or IEDs. ICCP, the Inter-Control Center Communications Protocol (a part of IEC 60870-6), is used for inter-master station communications.

Elseta's DNP3 stack has both Master and Slave protocols implemented. Both of them are able to serve multiple serial (over physical RS-485 line), TCP or TLS (over TCP) connections with high efficiency.

IEEE-1815 defines 4 subset levels (1-4) that consist of the objects and function codes that must be supported by the master and outstation. Levels 1-3 are supported fully and level 4 is supported partially. To get more information about how DNP3 works and what capabilities are supported one should get a copy of protocol specification and/or check *Slave Interoperability List*/*Configuration guides* for both Master and Slave protocols.

To set up TLS connection for both DNP3 Master and Slave, refer to sections *Excel configuration* and *Certificates*. All keys and certificates should be provided in the PEM format.

If no configuration is set up, DNP3 Master and Slave services are not started.

11.1 DNP3 Master

Default group and variation sets are used to send commands. If slave devices support different groups and variations, they can be adjusted in Excel configuration. For more information check section *Excel configuration*.

Table 10: Default command variations

Signal Type	Command Variation
Binary Output Command	Group12 Var1
Analog Output Command	Group41 Var1

11.2 DNP3 Slave

Default group and variation sets are used to send static and event values. If master devices support different groups and variations, they can be adjusted in Excel configuration. For more information check section *Excel configuration*.

Table 11: Default signal variations

Signal	Static Variation	Event Variation
Binary	Group1 Var2	Group2 Var1
Analog	Group30 Var1	Group32 Var1
Double Bit Binary	Group3 Var2	Group4 Var1
Binary Output Status	Group10 Var2	Group11 Var1
Counter	Group20 Var1	Group22 Var1
Frozen Counter	Group21 Var1	Group23 Var1
Analog Output Status	Group40 Var1	Group42 Var1
Octet String	Group110 Var0	Group111 Var0

12 DLMS

12.1 Overview

DLMS (Device Language Message Specification) is a suite of standards developed and maintained by the DLMS User Association. COSEM (Companion Specification for Energy Metering) includes a set of specifications that define the transport and application layers of the DLMS protocol.

In DLMS/COSEM all the data in electronic utility meters and devices are represented by means of mapping them to appropriate classes and related attribute values.

Objects are identified with the help of OBIS (OBject Identification System) codes (as per IEC 62056-61).

The DLMS driver allows only for readout and displaying only numeric values of DLMS object data fields. Connection via TCP or serial (RS232/RS485) port are supported.

The setup of the DLMS driver consists of communication and tag configuration. Protocol specific parameters (except for DLMS/IEC handshake mode) apply for both serial and IP connections.

12.2 Configuration

12.2.1 Devices section

serialnumber, **server_address** and **id** define the meter addressing parameters. Either **serialnumber** (meter serial number) or a combination of **server_address** (physical server address) and **id** (logical server address) is used. If a serial number is provided, physical and logical server addresses are ignored.
master_address defines the client address. This usually depends on the authentication used. Most meters support 16 for no authentication.
type defines the object referencing. SN should be used for short name referencing and LN for logical name referencing.
mode defines the initial handshake mode. IEC initiates the connection according to IEC 62056-21 ('/?!'), at the default initial baud rate (300 7E1). DLMS initiates the connection in DLMS mode. The IEC setting is irrelevant when an IP connection is used.
timeout_ms defines the reply timeout for telegrams both via serial and TCP.
auth and **password** define the authentication mode and password. This can be set to None, or other authentication variant (see table below), depending on the mode configured and supported by the particular meter.
ip and **port** define the IP address and TCP port for DLMS communication via IP.

When **ip** and **port** are configured, any serial port settings are ignored and connection is initiated only via IP.

Connection parameters are device specific and can differ between makes, models and utility companies. For initial connection settings please refer to the configuration of the particular meter.

Before configuring the Device section it is best to first check the connection parameters with a 3rd party DLMS utility.

Table 12: DLMS device configuration

Parameter	Description	Type	Default value	Example
serialnumber	Meter serial number	unsigned long	0	1122334455
slave_address	Meter physical server address	unsigned long	0	1600
id	Meter logical server address	unsigned long	0	1
master_address	Client address	int	16	1
type	Meter object referencing: SN - short referencing, LN - logical referencing	string	SN	LN
mode	Initial handshake mode: DLMS or IEC	string	DLMS	IEC
timeout_ms	Timeout in milliseconds	int	2500	1500
auth	Authentication: None, Low, High, HighMd5, HighSha1, HighSha256, HighGmac HighEcdsa	string	None	Low
password	Password for authentication	string	n/a	MyPass123
ip	IP address	string	n/a	192.168.0.141
port	TCP port	int	n/a	4059

12.2.2 Signals section

tag_job defines the tag job. A list of comma separated OBIS codes (or a single OBIS) should be used. Attribute indexes for objects of types register and extended register are selected automatically. Any other object types should include the attribute index in the form of OBIS:index.
tag_job_todo defines the job sub-job. This field should contain an OBIS code from within the list of the tag_job.

Table 13: DLMS tag configuration

Parameter	Description	Type	Default value	Example
tag_job	Tag job as single or multiple comma separated OBIS codes	string	n/a	”1.0.1.8.0.255, 1.0.15.8.1.255, 1.0.31.7.0.255:2”
tag_job_todo	Tag sub job	string	n/a	”1.0.15.8.1.255”

13 Modbus

Modbus is a serial communications protocol for use with its programmable logic controllers (PLCs). Modbus has become a de facto standard communication protocol and is now a commonly available means of connecting industrial electronic devices. It was developed for industrial applications, is relatively easy to deploy and maintain compared to other standards, and places few restrictions other than size on the format of the data to be transmitted.

Modbus enables communication among many devices connected to the same network, for example, a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from industry usage of Ladder logic and its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact.

WCC Lite supports both Modbus Master and Slave protocols. One can select between transmission over TCP/IP or serial connection (RS-485). Bytes to transmit can either be encoded according to both RTU and ASCII parts of standard.

13.1 Modbus Master

Modbus communication contains a single Master and may include more than 1, but not more than 247 devices. To gather data from peripheral devices, master device request a cluster of slave devices for data. If any device understand that this message is addressed for it, replies with data. As no timestamp is sent along with data, having recent data requires frequent polling. WCC Lite can be configured to acquire data periodically in custom-defined intervals.

13.1.1 Configuring datapoints

To use Modbus Master in WCC Lite, it has to be configured via an Excel configuration. This configuration contains two Excel sheets where parameters have to be filled in - *Devices* (Table 14) and *Signals* (Table 15).

Table 14: Modbus Master parameters for *Devices* tab

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	No
description	string	Description of a device	No
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used (”Modbus RTU”/”Modbus TCP””)	Yes
host (ip)	string (IP address format)	IP address of TCP slave device	Yes (for TCP). host parameter has a higher precedence
bind_address	string	IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”)	Yes (for TCP)

id	integer	Slave ID	
ascii	bool	Modbus ASCII mode (when Modbus RTU selected).	No (for RTU/ASCII)
timeout_ms (timeout)	integer	Response timeout in milliseconds	Yes. ”timeout” parameter has a higher precedence
device	string	Communication port (”PORT1”/”PORT2”)	Yes (for RTU/ASCII)
baudrate	integer	Communication speed, baud/s	Yes (for RTU/ASCII)
databits	integer	Data bit count for communication	Yes (for RTU/ASCII)
stopbits	integer	Stop bit count for communication	Yes (for RTU/ASCII)
parity	string	Communication parity option (”none”/”even”/”odd”)	Yes (for RTU/ASCII)
flowcontrol	string	Communication device flow control option. Available options (case insensitive): ”no”/”none”, ”sw”/”software”, ”hw”/”hardware”	Yes (for RTU/ASCII)
scan_rate_ms	integer	If provided and positive - all jobs will have similar scan rate - all reads and writes will be executed within this timeframe (parameter scan_rate_ms in *Signals* tab will be ignored)	Yes/No
poll_retry_count	integer	Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued	No
poll_delay_ms	integer	RS485 delay between read and write operations in milliseconds (Default: 50)	No (for RTU/ASCII). ”serial_delay” parameter has a higher precedence
bind_address	string (IP address format)	IP address to bind to specific network adapter (Default: 0.0.0.0)	No (for TCP)
port	integer	TCP communication port (Default: 502)	No (for TCP)
event_history_size	integer	Event log size	
modbus_multi_write	bool	Use 15/16 functions to write 1 register/coil (Default: 0)	No
comm_restart_delay	integer	Time delay between disconnecting from slave device and restarting connection (in milliseconds) (Default: 500)	No (for TCP)

Table 15: Modbus Master parameters for *Signals* tab

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly signal name	Yes
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique alphanumeric name of the signal to be used	Yes
enable	boolean	Enabling/disabling of an individual signal	Yes
job_todo	string	Request to send according to modbus specification without device address and checksum. This field can be identical on several tags to fetch them in single request	Yes
tag_job_todo	string	Similar format to **job_todo** field. Address and length must be a subset of job field. Defines the individual tag's resgister(s) or coil(s). Can be described in HEX or DEC formats	Yes
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes
log_size	Integer	Size of this signal's log in Event log.	
scan_rate_ms	integer	If **scan_rate_ms** in devices tab is not provided or is a positive number, read or write job will be executed within this timeframe in milliseconds	Yes/No
pulse_short_time_ms	integer	Time interval for short output pulse to stay active	No
pulse_long_time_ms	integer	Time interval for long output pulse to stay active	No

Different device vendors can have different implementations of a Modbus protocol stack. A register table can be a one of the primary differences. WCC Lite Modbus Master transmits the most significant word (byte) first, however, devices from some vendors might require transmitting the least significant word (byte) first. If that is the case, make sure to switch bytes as needed. To find out more about setting a correct number format, one should consult a section number_type (17.2.4).

Modbus job or tag (as a task to be completed) can be built in a two different formats - user can select a more convenient way for him:

- hexadecimal format with every single byte separated by | symbol. Device address, bytes containing output information and CRC (LRC) bytes should be excluded from the message;
- decimal format containing function number, first address and address count, separated by *;* symbol. All other information should be excluded from the message;

job_todo can group several **tag_job_todo**'s. That way one Modbus message can be used to extract several tags. Grouping is accomplished dynamically meaning that if several identical jobs are found, their tags are grouped automatically.

Modbus Master has an additional signal which can be configured to show communication status. It is used to indicate if the slave device has disconnected from master (WCC Lite). To configure such signal, two columns should be filled with particular values. To a newly created additional signal one should make **job_todo** equal to *device_status* and **tag_job_todo** equal to *communication_status*. Communication error status is set when a predefined count of messages (three by default, defined in **poll_retry_count** column) fail to be received or are considered invalid.

13.1.2 Debugging a Modbus Master application

If configuration for Modbus Master is set up, handler for protocol will start automatically. If configuration is missing or contains errors, protocol will not start. It is done intentionally to decrease unnecessary memory usage.

Table 16: Modbus Master command line debugging options

Option	Description
-h [–help]	Display help information
-V [–version]	Show version
-d<debug level>	Set debugging level
-c [–config]	Config path
-r [–raw]	Show raw telegram data
-f [–frame]	Show frame data
-s [–serial]	Show serial port data
–tcp	Show tcp packets
–ascii	Show ASCII messages
–rtu	Show RTU messages
-e [–redis]	Show redis debug information
-R [–readyfile]	Ready notification file

If Modbus Master does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly. To launch a debugging session, a user should stop modbus-master process and run modbus-master command with respective flags as in Table 16.

13.2 Modbus Slave

WCC Lite can act as one (or several) of slave devices in a communication line. This can be used to transmit data to SCADA systems or other RTU devices. It can reply to a messages from Modbus Master with matching device and register addresses.

13.2.1 Configuring datapoints

To use Modbus Slave in WCC Lite, it has to be configured via an Excel configuration. This configuration contains two Excel sheets where parameters have to be filled in - *Devices* (Table 17) and *Signals* (Table 18).

If TCP/IP is used as a trasmission medium, only devices with IPs predefined in *host* column are allowed to connect. All other connections are rejected

Table 17: Modbus Slave parameters for *Devices* tab

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	
description	string	Description of a device	
device_alias	string	Alphanumeric string to identify a device	Yes

enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used (”Modbus TCP Slave”/”Modbus serial Slave”)	Yes
device	string	Communication port (PORT1 or PORT2)	Serial
baudrate	integer	Communication speed, bauds/s	Serial
databits	integer	Data bit count for communication	Serial
stopbits	integer	Stop bit count for communication	Serial
parity	integer	Communication parity option (none/even/odd)	Serial
flowcontrol	string	Communication device’s flow control option. Available options (case insensitive) - ”no” or ”none”, ”sw” or ”software”, ”hw” or ”hardware”.	Serial
mode	string	Choosing between RTU (”rtu”) and ASCII (”ascii”) modes	
bind_address	string	Local IP addresses to bind the server to	TCP/IP
host	string	Space separated host IP addresses of master devices	TCP/IP
port	integer	TCP port to listen for incoming connections	TCP/IP

Table 18: Modbus Slave parameters for *Signals* tab

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly name of a signal	
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique signal name to be used	Yes
number_type	string	Number (variable) type (e.g. ”DIGITAL”)	Yes
common_address	integer	Address of a device	Yes
function	integer	Function number	Yes
info_address	integer	Register address	Yes
size	integer	Register/Coil size	Yes

13.2.2 Mapping values to registers

Internally stored values aren’t organised in a register-like order, therefore mapping should be done by the user. This mapping includes setting an address of the device WCC Lite is simulating as well as function number, register number and how much 16-bit registers are used to store a value. These values should be set in *common_address*, *function*, *info_address* and *size* columns respectively in the Excel configuration.

To find out how many register should be used for storing a values, how values can have their values swapped, a user should consult a section number_type (17.2.4).

If a Modbus master device requests a data from a register that is mapped but doesn’t yet have initial value, ILLEGAL DATA ADDRESS error code will be returned. The same error code is returned if a requested size of value is bigger that defined or if register is not configured at all.

13.2.3 Debugging a Modbus Slave application

If configuration for Modbus Slave is set up, handler for protocol will start automatically. If configuration is missing or contains errors, protocol will not start. It is done intentionally to decrease unnecessary memory usage.

Table 19: Modbus Slave command line debugging options

Option	Description
-h [–help]	Display help information
-V [–version]	Show version
-d<debug level>	Set debugging level
-c [–config]	Config path
-r [–raw]	Show raw telegram data
-f [–frame]	Show frame data
-s [–serial]	Show serial port data
–tcp	Show tcp packets
–ascii	Show ASCII messages
–rtu	Show RTU messages
-e [–redis]	Show redis debug information
-R [–readyfile]	Ready notification file

If Modbus Slave does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly. To launch a debugging session, a user should stop modbus-slave process and run modbus-slave command with respective flags as in Table 19.

14 IEC 60870-5

14.1 IEC 60870-5-103 Master

The IEC 60870-5-103 protocol is a companion standard for the informative interface of protection equipment.Standard IEC 60870-5-103 was prepared by IEC technical committee 57 (Power system control and associated communications).It is a companion standard for the basic standards in series IEC 60870-5:

Standard IEC 60870-5-103 defines communication between protection equipment and devices of a control system (supervisor or RTU) in a substation.

Standard IEC 60870-5-103 defines a multipoint communication protocol via which information can be exchanged between a control system (supervisor or RTU) and one or more protection devices. The control system is the master and the protection devices are the slaves. Each slave is identified by a unique address between 1 and 254. Address 255 is reserved for broadcast frames.

WCC Lite supports IEC 60870-5-103 Master protocol over serial link (according EIA RS-485). Its full functionality list can be found in a IEC 60870-5-103 PID Interoperability List which can be downloaded separately from this user manual.

14.1.1 Configuring datapoints

To use IEC 60870-5-103 Master in WCC Lite, it has to configured via an Excel configuration. This configuration contains two Excel sheets where parameters have to be filled in - *Devices* (Table 20) and *Signals* (Table 21).

Table 20: IEC 60870-5-103 parameters for *Devices* tab

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	
description	string	Description of a device	
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used (IEC 60870-5-103 master)	Yes
device	string	Communication port (PORT1 or PORT2)	Yes
baudrate	integer	Communication speed, bauds/s	Yes
databits	integer	Data bit count for communication	Yes
stopbits	integer	Stop bit count for communication	Yes
parity	integer	Communication parity option (none/even/odd)	Yes
flowcontrol	string	Communication device's flow control option. Available options (case insensitive) - "no" or "none", "sw" or "software", "hw" or "hardware".	
link address	integer	Address of device (link)	Yes
asdu_address	integer	Application Service Data Unit adress	Yes
time_sync_interval_sec	integer	Time frame between Time Synchronization requests in seconds	Yes

gi_interval_sec	integer	Time frame between General Interrogation requests in seconds	Yes
poll_interval_ms	integer	Polling interval in milliseconds. Time frame between two telegrams from master. Default - 100	
event_history_size	integer	Maximum count of events in event log. Default - 0	
poll_timeout_ms	integer	Timeout of waiting for incoming request	
serial_delay	integer	Communication device's serial delay in milliseconds. Time frame in which master station is not TX'ing after last RX byte. Default: 50	
poll_retry_count	integer	Number of retries of failed requests before announcing that device is in Error state	

Table 21: IEC 60870-5-103 parameters for *Signals* tab

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly name of a signal	
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique signal name to be used	Yes
source_device_alias	string	device_alias of a source device	For commands
source_signal_alias	string	signal_alias of a source signal	For commands
enable	boolean	Enabling/disabling of a signal	Yes
log_size	integer	Space for signal in event log	
gi	boolean	Including/excluding signal from General Interrogation. Default - 0 (exclude)	
common_address	integer	Address of a device	
function	integer	Function number	Yes
info_address	integer	Information address	Yes
info_number	integer	Information number	Yes
data_type	integer	ASDU type identificator	Yes
fleeting	boolean	Mark signal as fleeting type. Fleeting signals have go to DPI::OFF after defined time	
normalise_time_ms	integer	Time in milliseconds between station receiving DPI::ON and automatically switching to DPI::OFF. Default - 100.	If fleeting is used

IEC 60870-5-103 has an additional signal which can be configured to show communication status. It is used to indicate if the slave device has disconnected from master (WCC Lite). To configure such signal, two columns should be filled with particular values. To a newly created additional signal one should make **job_todo** equal to *device_status* and **tag_job_todo** equal to *communication_status*.

14.1.2 Debugging a IEC 60870-5-103 Master aplication

If configuration for IEC 60870-5-103 devices is set up, handler for protocol will start automatically. If configuration is missing or contains errors, protocol will not start. It is done intentionally decrease unnecessary memory usage.

If IEC 60870-5-103 does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly. To launch a debugging session, a user should stop *iec103-master* process and run *iec103-master* command with respective flags as in Table 22.

Table 22: IEC 60870-5-103 command line debugging options

Option	Description
-h [–help]	Display help information
-V [–version]	Show version
-d<debug level>	Set debugging level
-c [–config]	Config path
-r [–raw]	Show raw telegram data
-f [–frame]	Show frame data
-R [–readyfile]	Ready notification file

14.2 IEC 60870-5-104

IEC 60870-5-104 protocol (in short IEC 104) is a part of IEC Telecontrol Equipment and Systems Standard IEC 60870-5 that provides a communication profile for sending basic telecontrol messages between two systems in electrical engineering and power system automation. Telecontrol means transmitting supervisory data and data acquisition requests for controlling power transmission grids.

IEC 104 provides the network access to IEC 60870-5-101 (in short IEC 101) using standard transport profiles. In simple terms, it delivers IEC 101 messages as application data (L7) over TCP, usually port 2404. IEC 104 enables communication between control station and a substation via a standard TCP/IP network. The communication is based on the client-server model.

To set up TLS connection for both IEC104 Master and Slave, refer to sections *Excel configuration* and *Certificates*. All keys and certificates should be provided in the PEM format.

If no configuration is set up, IEC104 Master and Slave services are not started.

14.2.1 Slave

IEC 60870-5-104 Slave is designed not to lose data acquired from Master protocols. The data that arrives from Master protocols is stored in cache. This data is checked every second to manage further data sending. The data that leaves IEC-60870-5-104 Slave has output caches. They're built to provide switching between multiple sessions (redundant Scadas). If a new connection arrives, the old one is dropped, but data, that is stored in cache, not sent and not confirmed by Scada is transfered to new connection.

15 IEC 62056-21

15.1 Overview

The IEC 62056-21 standard defines protocol specifications for local meter data exchange.
Data is read out via serial port in modes A, B or C. The default initial serial port settings are 300 bps 7E1, as per standard, but can be user configured.
The driver implementation additionally allows for communication via TCP/IP, which is not described in the standard. In this case, baud rate acknowledgement is allowed however actual switchover between baud rates is not possible.
Mode A: data is requested and read out at the configured baud rate.
Mode B: data is requested at the configured baud rate and mutually switched to the baud rate proposed by the meter. Baud rate confirmation is absent.
Mode C: data is requested at the configured baud rate, new baud rate is proposed by the meter and, if acknowledged, data is read out at the proposed baud rate.
Currently data readout is supported in modes A, B and C.

For data readout it is necessary to know the port settings and the format of OBIS code representation as they can slightly differ (see table) depending on the configuration of the meter.

15.2 Configuration

15.2.1 Devices section

serialnumber defines the serial number of the meter. 0 (zero) will result in a '/?!' handshake string and may cause issues if more than one meter is wired to the serial port.
baudrate defines the initial connection baud rate. In modes B and C this will be switched to what ever baud rate is proposed by the meter.
meter_model defines the meter profile. This is reserved for future use and should be set to 1.
type defines the connection mode. Modes A, B and C are supported.

Table 23: IEC 62056-21 device configuration

Parameter	Description	Type	Default value	Example
serialnumber	Meter serial number	unsigned long	n/a	1122334455
baudrate	Serial port baud rate	int	300	9600
databits	Serial port byte length	int	7	8
stopbits	Serial port stop bits	int	1	2
parity	Serial port parity	string	EVEN	NONE
ip	IP address	string	n/a	192.168.1.123
port	TCP port	int	n/a	1000

If ip or port parameters are configured, any serial port settings are ignored and connections is initiated via TCP.

15.2.2 Signals section

tag_job defines the tag job. This is not used for this protocol and should be set to "'1".
tag_job_todo defines the job sub-job. This field should contain the exact representation of the OBIS code as it is configured in the meter. E.g. if the parameter of interest is represented as "1.8.0*24(0147238.4*kWh)", the value of the configuration field should be "1.8.0*24" (excluding quotation marks).

Table 24: IEC 62056-21 tag configuration

Parameter	Description	Type	Default value	Example
tag_job	Tag job	string	n/a	1
tag_job_todo	Tag sub job	string	n/a	1.8.0, 1-1.8.0

For **tag_job_todo** configuration it is best to first manually read the meter via PC or HHU (hand-held unit) to determine the exact OBIS representation format of the parameter as they can differ between meter manufacturers and utility companies.

16 WCC Lite internal signals

16.1 Overview

The WCC Lite contains several internal data points for readout and control which can be accessed via the Pooler service.

16.2 Configuration

16.2.1 Devices section

In the **devices** section, only the **protocol**, **scan_rate_ms** and **poll_delay_ms** are to be configured for this type of device.

Table 25: WCC Lite internal signals

Parameter	Description	Value
protocol	Protocol identifier	Internal data
scan_rate_ms	Update rate	>5000
poll_delay_ms	Poll delay	200

It is advised to set **scan_rate_ms** to a value greater than 5000 ms as frequent scans may result in significant overload of the pooler process.

16.2.2 Signals section

tag_job defines the tag job. This can be set to **gpio**, **board**, **netstat**, **led** and **process**.
tag_job_todo defines the job sub-job. This field should address the particular point of interest (see table).

Table 26: WCC Lite internal signal tag configuration

job_todo	Description	tag_job_todo	Description
gpio	Get GPIO	GPIO number	Use number 11 for onboard digital input
board	Board info	active-sim-iccid	Active SIM ICCID
		active-sim	Active SIM card
		cpu-usage	CPU usage
		duid	DUID
		gsm-current-rat-num	GSM current radio access technology identifier
		gsm-imsi	GSM IMSI number
		gsm-internet-status	GSM Internet status
		gsm-operator-num	GSM operator number
		gsm-roaming-status	GSM roaming status
		gsm-signal-quality-num	GSM signal quality (dBm)
		modem-imei	Modem IMEI number
		ram-usage	RAM usage

Table 26: WCC Lite internal signal tag configuration

job_todo	Description	tag_job_todo	Description
netstat|[interface]	Network statistics	TX	Bytes transferred
		RX	Bytes received
led	LED status/control	wcclite:blue:wlan	WLAN LED
		wcclite:green:eth0	ETH0 LED
		wcclite:green:eth1	ETH1 LED
		wcclite:red:fault	Fault LED
		wcclite:relay	Relay LED & Output
process	Check if process is running	[process name]	1 or 0 is returned

Assigning source signals to tags other than wcclite:relay may cause undesirable effects. Signals other than wcclite:relay should be used for monitoring only.

17 Excel configuration

Protocol HUB uses configuration in excel file format. Each sheet represents a specific part of configuration:

- **Devices** contains device list and protocol related configuration.
- **Signals** contains a list of signals and their options.

First line on each sheet is a header row that contains parameter name for each column. Header order determines parameter names for each following row. Every line after the header is a new entry. An empty row is interpreted as end of sheet. Any rows after empty row are discarded.

17.1 Devices sheet

Devices sheet contains all devices to be configured on gateway. Each row represents one device and its settings. Following options are required for each device:

- **name** - Name of the device. Used for representation only.
- **description** - A short description for the device. Used for representation only.
- **device_alias** - A unique short name for the device. It is used for linking signals to a device. Alias can only contain alphanumeric characters and dashes (- and _). Alias must be unique for each device.
- **protocol** - Protocol type to use on device. Following values are valid:
  - Modbus RTU
  - Modbus TCP
  - Aurora
  - SMA Net
  - Kaco
  - Ginlong
  - Fault Passage Indicator (SMS)
  - Elgama
  - MBus Serial
  - MBus TCP
  - Solplus
  - Internal data
  - VBus
  - Windlog
  - Vestas
  - IEC 60870-5-101 master
  - IEC 60870-5-101 slave
  - IEC 60870-5-103 master
  - IEC 60870-5-104 master

- IEC 60870-5-104 slave
- IEC 62056-21 (since FW 1.2.13)
- ComLynx
- PowerOne
- Delta
- Modbus TCP Slave
- Modbus serial Slave
- DNP3 Master
- DNP3 Slave
- DLMS (since FW 1.3.0)
- IEC 61499 (since FW 1.4.0)
- MQTT (since FW 1.4.0)

Although device **name** rules aren't strictly enforced, it is highly advised to give a unique name for every new device. Identical device names might introduce confusion while searching for signal in *Imported Signals* tab.

17.1.1 Optional settings

- **enable** - Flag to enable or disable device on system. Can contain values 0 or 1.
- **event_history_size** - Maximum number of signal events to save on device for later review. Older records will be erased. This feature is only available on cloud firmware.

17.1.2 Serial port settings

Required for any protocol that uses serial line communication.

- **device** - Serial port for communication **(PORT1/PORT2)**
- **baudrate** - Serial port speed. Valid values:
  - 300
  - 600
  - 1200
  - 2400
  - 4800
  - 9600
  - 19200
  - 38400
  - 57600
  - 115200
- **databits** - Number of data bits **(6-9)**
- **stopbits** - Number of stop bits **(1-2)**
- **parity** - Parity mode **(none/even/odd)**
- **flowcontrol** - Flow control method **(none/hardware/software)**

17.1.3 TCP/IP settings

Settings for any protocol that uses communication over TCP/IP. Note that all TLS certificates and keys are stored in single folder therefore only name and not the path should be filled in respective fields. TLS fields are only supported for IEC-60870-5-104 Slave and DNP3 Master and Slave.

- **ip** - IP address for master protocol to connect to;
- **bind_address** - one of local IP addresses to bind the server to. Connections through other network devices will be ignored;
- **host** - space separated host IP addresses of master devices;
- **port** - TCP port to listen for incoming connections;
- **tls_local_certificate** - name of local TLS certificate;
- **tls_peer_certificate** - name of certificate authority (CA) TLS certificate;
- **tls_private_key** - name of private key for making TLS connections.

17.1.4 Protocol specific settings

Protocol	Attribute	Type	Description
Modbus RTU	id	0 - 255	Modbus slave device unique identifier
	ascii	boolean	Modbus serial mode: • 0 - RTU mode • 1 - ASCII mode
	timeout	0 - 6e7	Response timeout (us)
Modbus TCP	id	0 - 255	Modbus slave device address
	ip	IPv4	IP address
	port	0-65535	TCP port
	timeout	0 - 6e7	Response timeout (us)
Aurora, Ginlong, Delta	id	0 - 255	Inverter ID
	timeout	0 - 6e7	Response timeout (us)
SMA Net	serialnumber	0 - 2^{32}	Inverter serial number
Kaco	id	0 - 32	Inverter serial number
	ext_device	boolean	• 0 - Inverter is connected directly • 1 - Inverter is connected via remote terminal
	timeout	0 - 6e7	Response timeout (us)

Protocol	Attribute	Type	Description
Fault Passage Indicator	type	string	Beacon type: • linetroll3100 • linetrollr400d • elseta-beacon • sipronikaLok200
	phone_number	integer	Phone number of beacon modem
	heartbeat_timeout_min	0 - 86400	Heartbeat timeout (min)
	counter_threshold_min	0 - 60	Missing counter delay (min)
	timeout	0 - 6e7	Response timeout (us)
Elgama	id	integer	Meter serial number
	meter_model	integer	Meter type: • **0** - EPQM/LZQM • **1** - EPQS • **2** - GAMA300 • **3** - GAMA100 • **4** - ITS cl
	use_time	boolean	• 0 - Use system time • 1 - Use meter's time
	timeout	0 - 6e7	Response timeout (us)
MBus Serial	address	0 - 1e10	Device address
	timeout	0 - 6e7	Response timeout (us)
MBus TCP	id	0 - 1e10	Device address
	ip	IPv4	IP address
	port	0 - 65535	TCP port
	timeout	0 - 6e7	Response timeout (us)
Solplus	url	URI	URL
	timeout	0 - 360000	Response timeout (ms)
VBus, Vestas	master_address	0 - 255	Master address
	slave_address	0 - 255	Slave address
	timeout_ms	0 - 60000	Response timeout (ms)
Windlog	timeout_ms	0 - 60000	Response timeout (ms)
IEC 60870-5-101 master	link_address	0 - 65535	Link address
	asdu_address	0 - 65535	Common address of ASDU
	link_size	1 - 2	Link address size in bytes
	asdu_size	1 - 2	Common address size in bytes
	cot_size	1 - 2	Cause of transmission (COT) size in bytes
	ioa_size	1 - 3	Information object address (IOA) size in bytes
	time_sync_interval_sec	integer	Time synchronization interval (sec)

Protocol	Attribute	Type	Description
	gi_interval_sec	integer	General interrogation interval in seconds
	poll_interval_ms	integer	Class request interval in milliseconds
	poll_timeout_ms	integer	Device response timeout in milliseconds
	poll_retry_count	integer	Number of failed poll retries before link is reset
IEC 60870-5-101 slave	link_address	0 - 65535	Link address
	link_size	1 - 2	Link address size in bytes
	asdu_size	1 - 2	Common address size in bytes
	cot_size	1 - 2	Cause of transmission (COT) size in bytes
	ioa_size	1 - 3	Information object address (IOA) size in bytes
	time_sync	boolean	Allow time synchronization
	sp_time	boolean	Add CP56Time2a information to single point signals
	dp_time	boolean	Add CP56Time2a information to double point signals
	me_time	boolean	Add CP56Time2a information to measurements
	message_size	0 - 255	Maximum length of a message
	cache_size	0 - 1000	Maximum number of unsent events to store in a buffer
	respond_delay	0 - 10e6	Time in microseconds to wait before sending responses
	single_byte_ack	boolean	Use single character acknowledge
IEC 60870-5-103 master	link_address	0 - 65535	Link address
	asdu_address	0 - 65535	Common address of ASDU
	time_sync_interval_sec	integer	Time synchronization interval (sec)
	gi_interval_sec	integer	General interrogation interval in seconds
	poll_interval_ms	integer	Class request interval in milliseconds
	poll_timeout_ms	integer	Device response timeout in milliseconds
	poll_retry_count	integer	Number of failed poll retries before link is reset
IEC 60870-5-104 master	host	IPv4	Host IP address
	port	0 - 65535	TCP port
	asdu_address	0 - 65535	Common address of ASDU
	asdu_size	1 - 2	Common address size in bytes
	cot_size	1 - 3	Cause of transmission (COT) size in bytes
	ioa_size	1 - 3	Information object address (IOA) size in bytes
	time_sync_interval_sec	integer	Time synchronization interval (sec)

Protocol	Attribute	Type	Description
	gi_interval_sec	integer	General interrogation interval in seconds
	t1	integer	Acknowledge timeout t1 (sec)
	t2	integer	Connection ACKRSN clock t2 (sec)
	t3	integer	Connection TESTFR clock t3 (sec)
	rwt	integer	Receive window (RWT)
	swt	integer	Send window (SWT)
IEC 60870-5-104 slave	bind_address	IPv4	Bind to local IP address
	host	IPv4	Space separated remote host IP addresses
	port	0 - 65535	TCP port
	asdu_size	1 - 2	Common address size in bytes
	cot_size	1 - 3	Cause of transmission (COT) size in bytes
	ioa_size	1 - 3	Information object address (IOA) size in bytes
	time_sync	boolean	Allow time synchronization
	sp_time	boolean	Add CP56Time2a information to single point signals
	dp_time	boolean	Add CP56Time2a information to double point signals
	me_time	boolean	Add CP56Time2a information to measurements
	message_size	0 - 255	Maximum length of a message
	cache_size	0 - 1000	Maximum number of unsent events to store in a buffer
	t1	integer	Acknowledge timeout t1 (sec)
	t2	integer	Connection ACKRSN clock t2 (sec)
	t3	integer	Connection TESTFR clock t3 (sec)
	rwt	integer	Receive window (RWT)
	swt	integer	Send window (SWT)
ComLynx	network	1 - 14	Network address
	subnet	0 - 14	Subnet address
	address	0 - 254	Device address
	timeout_ms	0 - 60000	Response timeout (ms)
PowerOne	serial	integer	Serial number
	type	string	• **CU** - Collecting unit • **CB** - Normal CB • **HID** - HID with integrated CB
	timeout_ms	0 - 60000	Response timeout (ms)

Protocol	Attribute	Type	Description
Modbus serial Slave	mode	string	• **rtu** - RTU mode • **ascii** - ASCII mode
Modbus TCP Slave	bind_address	IPv4	Bind to local IP address
	port	0 - 65535	TCP port
	mode	string	• **tcp** - TCP mode • **rtu** - RTU mode • **ascii** - ASCII mode
DNP3 Slave	time_sync_interval_sec	64-bit integer	Defines how often (in seconds) slave will request time synchronization. If greater than 0 - slave will request synchronizations, will reset timer if master did it earlier. If zero - slave won't request timesyncs, but will allow them. If -1 - timesyncs are not supported - requests will be dropped. Default - 0.
	timeout_ms	64-bit integer	Timeout in ms for solicited confirm, unsolicited confirm and unsolicited retry. Default - 2000.
	transport_type	string	Transport type - "tcp", "tls", "serial" - any other value will throw runtime error. Defines transport type.
	source_address	unsigned 16-bit integer	Address of a slave (local) station. Default - 1.
	destination_address	unsigned 16-bit integer	Address of a master station. Default - 1.
	max_tx_frag_size	unsigned 32-bit integer	Maximum size of a transmitted fragment. Can't be bigger than 2048. If bigger - reduced to default - 2048.
	unsol_classes	string	Defines for which classes slave (outstation) would send unsolicited actions. String must contain numbers 1, 2 or 3. Order is not important. Default - [] (none of the classes will have unsolicited option). It is advised to leave default value - master should enable unsolicited action sending.

Protocol	Attribute	Type	Description
	keep_alive_timeout	64-bit integer	Keep alive timer in seconds. Defaults to 60.
	select_ms	64-bit integer	Select command timeout in milliseconds. Valid for all signals. Default - 10000.
DNP3 Master	time_sync_interval_sec	64-bit integer	Periodic time sync interval in seconds. If the value is positive time syncs are forced and periodic, if zero - time syncs react to IIN bits from slave, if negative - time syncs are disabled. Default - 0.
	poll_timeout_ms	64-bit integer	Poll action timeout in milliseconds. Default - 2000.
	transport_type	string	Transport type -"tcp", "tls", "serial" - any other value will throw runtime error. Defines transport type.
	source_address	unsigned 16-bit integer	Address of master station. Default - 1.
	destination_address	unsigned 16-bit integer	Address of slave station. Default - 1.
	max_rx_frag_size	unsigned 32-bit integer	Maximum size of received fragment. Can't be bigger than 2048. IF bigger - reduced to default. Default - 2048.
	unsol_classes	string	Defines for which classes master will initiate unsolicited action on startup. String must contain numbers 1, 2 or 3. Order is not important. Default - "" (none of the classes will have unsolicited option).
	integrity_scan_interval	64-bit integer	Time between integrity scans (classes 0,1,2,3) in seconds (general interrogation). To disable write 0. Default - 0.
	exception_scan_interval	64-bit integer	Time between exception scans (classes 1,2,3) in seconds. To disable write 0. Default - 0.
	keep_alive_timeout	64-bit integer	Time interval in which keep-alive messages will be started to send. Default - 60.
IEC 62056-21	serialnumber	long	Serial number
	type	string	IEC 62056-21 mode (A/B/C)
DLMS	id	integer	Logical server address
	serialnumber	unsigned long	Serial number of the device. If set to zero, id and slave_address are used.

Protocol	Attribute	Type	Description
	type	string	Name referencing: SN - short name referencing, LN - logical name referencing
	master_address	integer	HDLC client ID. Public (no authentication): 16 (0x10); Broadcast: 177 (0x7F). Other client IDs may be available - check with the meter configuration
	slave_address	integer	Physical server address
	timeout_ms	integer	Timeout in milliseconds
	mode	string	Initial handshake mode: DLMS or IEC
	auth	string	Authentication mode. Available modes are: None, Low, High, HighMd5, HighSha1, HighGmac, HighSha256
	password	string	Authentication password
	ip	string	IP address for DLMS/IP
	port	int	TCP port for DLMS/IP

17.2 Signals sheet

Signals sheet contains all signals linked to devices. Each signal is defined in single row. Signal list can be split in multiple sheets. Each sheet name may start as *Signals*.

17.2.1 Required attributes

- **signal_name** - Name of the signal. Used for repesentation only.
- **device_alias** - Alias of a device defined in *Devices* sheet. Signal is linked to a matching device.
- **signal_alias** - A unique short name for the signal. It is used for linking signal to other signals. Alias can only contain alphanumeric characters and dashes (- and _). Device and signal alias combination must be unique.

17.2.2 Optional attributes

- **source_device_alias** - Alias of a source device defined in *Devices* sheet. If a user intends to use several signals and combine them via mathematical or logical function, every alias should be seperated by a newline symbol (in the same cell). An operation used must also be defined in an operation column.
- **source_signal_alias** - Alias of a source signal defined in *Signals* sheet. If a user intends to use several signals and combine them via mathematical or logical function, every alias should be seperated by a newline symbol (in the same cell). An operation used must also be defined in an operation column. Every **source_signal_alias** should be posted in the same line as its respective **source_device_alias**. Aliases can only contain alphanumeric characters and dashes (- and _). Device and signal alias combination must be unique.

- **enable** - Flag to enable or disable signal on system. Can contain values 0 or 1.
- **tag_type** - Tag type. *Simple* signals are polled from device. *Virtual* signals are computed internally.
- **off_message** - Message to display when single point or double point signals are in OFF state.
- **on_message** - Message to display when single point or double point signals are in ON state.
- **units** - Signal value measurements units.
- **multiply** - Multiply value by this number.
- **add** - Add this number to a value.
- **sum_signals** - Define other signal values to add to current signal. This field uses following format: **dev_alias**/**tag_alias**. Multiple signals can be defines usign commas.
- **min_value** - Minimum expected value. If result is lower than this value, invalid flag is raised.
- **max_value** - Maximum expected value. If result is higher than this value, overflow flag is raised.
- **absolute_threshold** - Absolute threshold level.

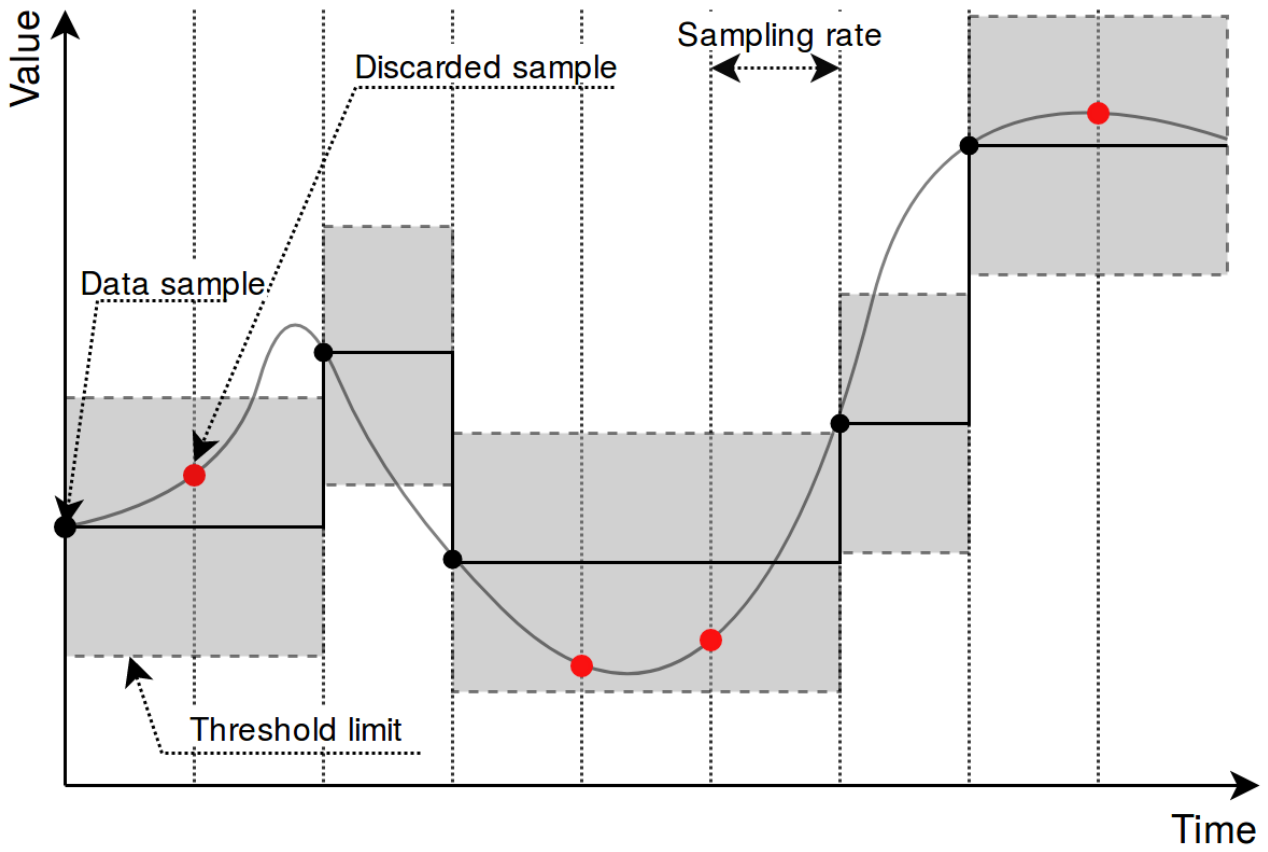


Figure 13: Result of using an absolute threshold

- **integral_threshold** - Integral threshold level.
- **integral_threshold_interval** - Integral threshold addition interval in milliseconds.
- **threshold_units** - Units used in threshold level fields (percent/real).
- **log_size** - Maximum number of records for this tag to keep in storage for CloudIndustries logging.

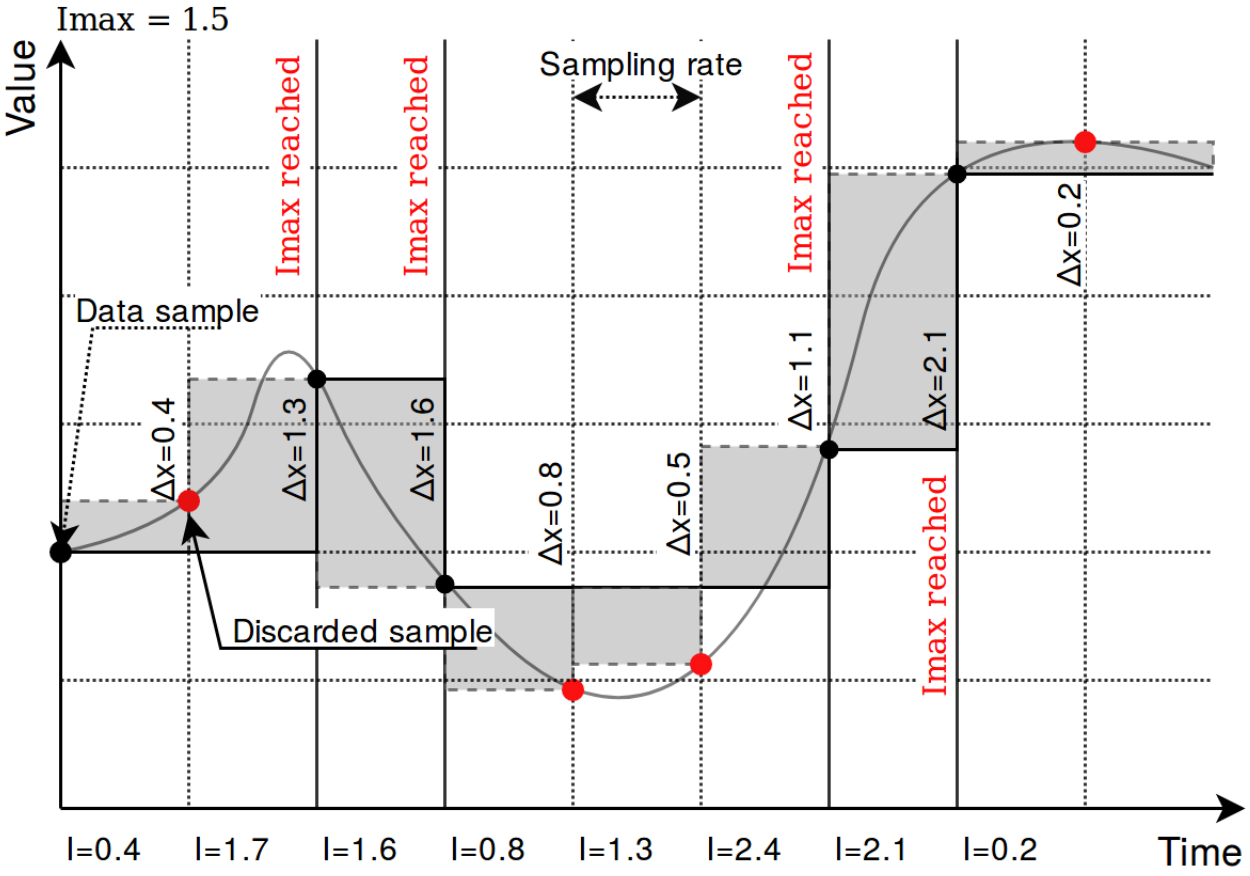


Figure 14: Result of using an integral threshold

- **suppression_values** - Space separated numeric values to be used in suppression.
- **suppression_time_ms** - Suppression time in milliseconds.
- **operation** - Mathematical or logical operation to be used for signals defined in **source_signal_alias** column. Following mathematical operations for source signal values can be used: *avg* (average of all values), *min* (lowest value), *max* (highest value), *median* (median value) and *sum* (all values accumulated to a single number). Logical operations, intended for unsigned integers only, are *or* and *and* operations.
- **bit_select** - selecting an individual bit of an integer number; bit numeration starts from zero.
- **math_expression** - a mathematical expression for signal value to be evaluated against. Explained in detail in section 17.2.7.

17.2.3 Signal recalculation operation priority

A value generated by some protocol usually has to be recalculated in one way or another. This might mean changing the value of an argument as well as adding flags needed for other protocols to correctly interpret results. As recalculation is a sequential process, some actions are done before others. The sequence of operations done to a value is as follows:

- *Edition of attributes*. Attributes for further interpretation are added. This might, for example, include flag to show that a signal resembles an answer to a command;
- *Mathematical calculations*. **multiply**, **add**, **bit_select** and **math_expression** columns are evaluated here;

- *Usage of last value*. Decision if last value for a signal should be used if a new value of a signal is not a number (NaN) or contains a non-topical (NT) flag;
- *Limiting of values*. If a value exceeds a lower or higher configured limit, value is approximated not be lower (or higher) than the limit. An additional invalid (IV) or overflow (OV) flag is added as frequently used in IEC-60870-5 protocols;
- *Suppresion of values*. As electrical circuits can be noisy, protocols may generate multiple values in a short amount of time. What is more, some values are considered as intermediary and ideally should not be sent to SCADA unless they stay in the same state for some amount of time. **suppression_values** and **suppression_time_ms** are used to configure this functionality;
- *Treshold checking*. If a new signal doesn't cross a threshold target value, value is supressed and not used in further stages. **absolute_threshold**, **integral_threshold**, **integral_threshold_interval**, **threshold_units** columns are used to configure this functionality.

Not all of the elements in this sequence have to configured, missing operation are skipped and values are fed to a further stage of signal recalculation.

17.2.4 number_type field

This field is required for some protocols to determine a method to retrieve a signal value from hexadecimal form. Available values:

- **FLOAT** - 32-bit single precision floating point value according to IEEE 754 standard
- **DOUBLE** - 64-bit double precision floating point value according to IEEE 754 standard
- **DIGITAL** - 1-bit boolean value
- **UNSIGNED8** - 8-bit unsigned integer (0 - 255)
- **SIGNED8** - 8-bit signed integer (-128 - 127)
- **UNSIGNED16** - 16-bit unsigned integer (0 - 65535)
- **SIGNED16** - 16-bit signed integer (-32768 - 32767)
- **UNSIGNED32** - 32-bit unsigned integer (0 - 4294967295)
- **SIGNED32** - 32-bit signed integer (-2147483648 - 2147483647)
- **UNSIGNED64** - 64-bit unsigned integer (0 - 18446744073709551615)
- **SIGNED64** - 64-bit signed integer (-9223372036854775808 - 9223372036854775807)

Number conversion uses **big endian** byte order by default. Converted data will be invalid if byte order on connected device side is different. In such case byte swap operations can be used. Adding swap prefixes to number type will set different a byte order while converting values. Following swap operations are available:

- **SW8** - Swap every pair of bytes (8 bits) (e.g., **0xAABBCCDD** is translated to **0xBBAADDCC**)
- **SW16** - Swap every pair of words (16 bits) (e.g., **0xAABBCCDD** is translated to **0xCCDDAABB**)
- **SW32** - Swap every pair of two words (32 bits) (e.g., **0x1122334455667788** is translated to **0x5566778811223344**)

Table 28: Example of using different swapping functions

Address	0	1	2	3	4	5	6	7
Original number	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
SW8	Byte 1	Byte 0	Byte 3	Byte 2	Byte 5	Byte 4	Byte 7	Byte 6
SW16	Byte 2	Byte 3	Byte 0	Byte 1	Byte 6	Byte 7	Byte 4	Byte 5
SW32	Byte 4	Byte 5	Byte 6	Byte 7	Byte 0	Byte 1	Byte 2	Byte 3
SW8.SW16	Byte 3	Byte 2	Byte 1	Byte 0	Byte 7	Byte 6	Byte 5	Byte 4
SW8.SW32	Byte 5	Byte 4	Byte 7	Byte 6	Byte 1	Byte 0	Byte 3	Byte 2
SW8.SW16.SW32	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

Add a dot separated prefix to number format to use byte swapping. Multiple swap operations can be used simultaneously. For example, use **SW8.SW16.SIGNED32** to correctly parse a 32-bit signed integer in a little endian format. Table 28 shows in detail how bytes, words or double words can be swapped and how swapping functions can be combined to make different swapping patterns. Table shows how byte swap is done for 64-bit (8-byte) numbers. It doesn't matter if it is an unsigned/signed integer or double, byte swapping is considered a bit-level operation. If a number is shorter than 64 bits, the same logic applies, the only difference is unavailability of some swapping operations (SW32 for 32-bit and smaller numbers). Using such unavailable operation might lead to an undefined behaviour.

17.2.5 Protocol specific settings

Protocol	Attribute	Type	Description
Modbus RTU, Modbus TCP	job_todo	string	Request to send according to modbus specification without device address and checksum. This field can be identical on several tags to fetch them in single request.
	tag_job_todo	string	Similar format to job field. Address and length must be a subset of job field.
	number_type	string	see 17.2.4
SMA Net	tag_job_todo	string	Parameter name (e.g., B.Ms.Amp)
Kaco, MBus	tag_job_todo	string	Parameter index
Fault Passage Indicator	type	string	Signal type: • **fault** - Raise fault signal if received value maches • **timeout** - Raise fault signal when there are no events for longer than counter_threshold_min • **value** - Use value directly
	name	string	Name of fault or value parameter. Leave empty for timeout signal.
	timeout_enabled	boolean	If enabled, this signal will have notopical flag on timeout condition.

Protocol	Attribute	Type	Description
	fault_value	string	Required value to be received to activate fault. Leave blank for value/timeout signals.
Elgama	job_todo	string	Request address in hexadecimal form. Use predefined addresses form templates only
	tag_job_todo	string	Request address and parameter offset in hexadecimal form. Use predefined addresses form templates only
IEC 60870-5-101, IEC 60870-5-104	common_address	integer	Common address of ASDU
	info_address	integer	IOA address
	gi	boolean	Enable responses to general interrogation
	data_type	integer	ASDU type id. Types are identified automatically if this field is set to zero.
	select_ms	integer	Time limit in milliseconds for command execution. Command select has to be performed before execution if this parameter is specified. Direct command execution can be performed only if this field is left empty or set to zero.
IEC 60870-5-103	common_address	integer	Common address of ASDU
	function	integer	Function code
	info_address	integer	Information address
	info_number	integer	Information offset
	gi	boolean	Enable responses to general interrogation
	data_type	integer	ASDU type id. Types are identified automatically if this field is set to zero.
Modbus Slave	common_address	integer	Modbus Slave address
	function	integer	Function code
	info_address	integer	Coil / input / register address
	size	integer	Register size in bits
	number_type	string	see 17.2.4
DNP3 Slave	select_ms	integer	Default command behaviour. If -1 – DirectOperateNoAck, 0 – DirectOperate, any other number – default – SelectBeforeOperate.

Protocol	Attribute	Type	Description
	signal_type	string	DNP3 signal type. Available values: Binary, Analog, Double Bit Binary, Binary Output Status, Counter, Frozen Counter, Analog Output Status, Octet String, Binary Output Command, Analog Output Command. Case and spaces insensitive. If no or invalid value provided - crashes with signal_type definition error
	static_variation	unsigned 16-bit integer	Override default signal's static variation. Valid for *Status* mode signals.
	event_variation	unsigned 16-bit integer	Override default signal's event variation. Valid for *Status* mode signals.
	index	unsigned 16-bit integer	Index of signal.
	class_num	integer	Class assignment of this signal. Default (or if assigned incorrectly) - 0.
	deadband	double	Deadband for Analog, Analog Output Status, Counter, Frozen Counter signals. Default - 0.
DNP3 Master	select_ms	integer	Default command behaviour. If -1 – DirectOperateNoAck, 0 – DirectOperate, any other number –SelectBeforeOperate.
	signal_type	string	DNP3 signal type. Available values: Binary, Analog, Double Bit Binary, Binary Output Status, Counter, Frozen Counter, Analog Output Status, Octet String, Binary Output Command, Analog Output Command. Case and spaces insensitive. If no or invalid value provided - crashes with signal_type definition error
	command_variation	unsigned 16-bit integer	Override default signal's command variation. Valid for *Command* mode signals.
	index	unsigned 16-bit integer	Index of signal.
	class_num	integer	Class assignment of this signal. Default (or if assigned incorrectly) - 0.
IEC 62056-21	tag_job_todo	string	OBIS code
DLMS	job_todo	string	Job in OBIS:index format. Should be supplied in full OBIS format with the particular index of the value following.

17.2.6 Linking signals

Signals can be linked together to achieve data transfer between several protocols. If a signal source is defined, all output from that source will be routed to the input of target signal. This way events polled from a modbus device (e.g., Modbus, IEC 60870-5, etc.) can be delivered to external station over a different protocol. A signal source is required if a signal is created on a slave protocol configuration to link events between protocols.

Example 1
To read a coil state from a Modbus device and transfer it to IEC 60870-5-104 station, following steps may be taken:

1. Create a Modbus master configuration in Devices sheet.
2. Create a IEC 60870-5-104 slave configuration in Devices sheet.
3. Create a signal on master device to read coil status (function 1).
4. Create a signal on slave device with single point type (data_type = 1).
5. Set **source_device_alias** and **source_signal_alias** fields on slave device signal to match **device_alias** and **ignal_alias** on master device's coil signal.

Example 2
To write a coil state to a Modbus device on a command from IEC 60870-5-104 station, following steps may be taken:

1. Follow steps 1-3 from example 1.
2. Create a signal on slave device with single command type (data_type = 45).
3. Set **source_device_alias** and **source_signal_alias** fields on master configuration coil signal to match **device_alias** and **signal_alias** on slave device's command signal. Coil will be written to a value received by a command.
4. Set **source_device_alias** and **source_signal_alias** fields on command signal to match **device_alias** and **signal_alias** on master device's coil signal. A command termination signal will be reported to the station on coil write result.

For additional information regarding configuration of IEC 60870-5-101/103/104 protocols, please refer to "IEC 60780-5-101/103/104 PID interoperability for WCC Lite devices", accordingly.

17.2.7 Mathematical expressions

Signal value might require some recalculation or signal update prior to being sent. Understandably, existing columns in Excel configuration like **multiply**, **add**, **bit_select** might not be flexible enough. To overcome these limitations, symbolic mathematical expressions can be configured to do calculations automatically on every update of a signal.

It should be noted that filling mathematical expression disables other mathematical scalar operations for a single value such as **multiply**, **add** or **bit_select**. Other functions (primarily between several signals) are still available such as **operation**.

Feature list:

- Optimized for speed
  - High parsing performance
  - if-then-else operator with lazy evaluation
- Default implementaion with many features
  - 25 predefined functions
  - 18 predefined operators
- Unit support
  - Use postfix operators as unit multipliers (3m -> 0.003)

Table 30: Supported mathematical functions

Name	Argument count	Explanation
sin	1	sine function (rad)
cos	1	cosine function (rad)
tan	1	tangent function (rad)
asin	1	arcus sine function (rad)
acos	1	arcus cosine function (rad)
atan	1	arcus tangens function (rad)
sinh	1	hyperbolic sine function
cosh	1	hyperbolic cosine
tanh	1	hyperbolic tangens function
asinh	1	hyperbolic arcus sine function
acosh	1	hyperbolic arcus tangens function
atanh	1	hyperbolic arcur tangens function
log2	1	logarithm to the base 2
log10	1	logarithm to the base 10
log	1	logarithm to base e (2.71828...)
ln	1	logarithm to base e (2.71828...)
exp	1	e raised to the power of x
sqrt	1	square root of a value
sign	1	sign function -1 if x<0; 1 if x>0
rint	1	round to nearest integer
abs	1	absolute value
min	variable	min of all arguments
max	variable	max of all arguments
sum	variable	sum of all arguments
avg	variable	mean value of all arguments

It should be noted that trigonometric functions (excluding hiperbolic functions) only support arguments in radians. This means that arguments for this function have to be recalculated if angle is defined in degress.

Value recalculation is only triggered on signal change of the preconfigured signal. That means that using other signals (via *TagValue()* call) does not trigger value update.

Some mathematical expression cannot be mathematically evaluated in some conditions, for example, square root cannot be found for negative numbers. As complex numbers are not supported, result is then equal to Not a Number (NaN). These results are marked with an invalid (IV) flag.

Table 31: Supported binary operators

Operator	Description	Priority
=	assignment	-1
»	right shift	0
«	left shift	0
&	bitwise and	0
|	bitwise or	0
&&	logical and	1
||	logical or	2
<=	less or equal	4
>=	greater or equal	4
!=	not equal	4
==	equal	4
>	greater than	4
<	less than	4
+	addition	5
-	subtraction	5
*	multiplication	6
/	division	6
^	raise x to the power of y	7

Ternary operators can be used. This expression can be compared to the operator supported by C/C++ language (Table 32). Condition is written before a question (*?*) sign. If condition is true, result after question sign is selected. If condition is false, result after colon (*:*) is selected.

Table 32: Supported ternary operators

Operator	Description	Remarks
?:	if then else operator	C++ style syntax

Table 33: Example expressions

Expression	Description
value * 0.0001	Multiply the tag by a constant.
value + TagValue(”tag/dev_alias/sig_alias/out”)	Add value of tag/dev_alias/sig_alias/out to the current tag.
sin(value)	Return a predefined sine function value of the tag.
(value > 5) ? 1 : 0	If value is greater than 5, result should be equal to 1, otherwise - equal to 0

User can construct his own equation by using the aforementioned operators and functions. These examples can be seen in Table 33.

Variable called **value** is generated or updated on every signal change and represent the signals being configured. If another value from tag list is intended to be used, one should use *TagValue()* function to retrieve its last value.

The inner argument of *TagValue()* function has to described in a Redis topic structure of WCC Lite. That means that it has to be constructed in a certain way. Quotes should be used to feed the topic name value, otherwise expression evaluation will fail.

Every Redis topic name is constructed as *tag/[device_alias]/[signal_alias]/[direction]*. Prefix *tag/* is always used before the rest of argument. *device_alias* and *signal_alias* represent columns in Excel configuration. *direction* can have one of four possible values - *rout*, *out*, *in*, *rin*; all of which depend on the direction data is sent or acquired device-wise. For example, *out* keyword marks data sent out of WCC Lite device, whereas *in* direction represents data that WCC Lite is waiting to receive, for example, commands. Additional *r* before either direction means that data is **r**aw, it was is presented the way it was read by an individual protocol.

Several functions are defined make tag operations possible:

- *TagValue(key)* - returns last known value of tag identified by redis key;
- *TagFlag(key)* - returns 1 if tag flag exists. Name format is: ”key flag”. For example to check if tag is notopical, name would be ”tag/19xxxxxx/x/x nt”;
- *TagAttribute(key)* - similar to TagFlag, but returns a numeric value of a tag attribute;
- *TagTime(key)* - returns unix timestamp in milliseconds of a last know tag value.

17.3 Uploading configuration

As of WCC Lite version v1.4.0 there are three separate ways to import the configuration: import an Excel file via web interface, generate compressed configuration files and later upload them via web interface; or generate compressed configuration files and upload them via utility application.

For WCC Lite versions v1.4.0, name of the file is shown in *Protocol Hub->Configuration*. Older versions only allow configuration file to be stored to a file called *phub.xlsx* and later downloaded with a custom-built name reflecting date of a download. Upgrade process from older version to versions v1.4.0 and above when preserving configuration files automatically makes the neccessary changes to enable this new functionality of WCC Lite.

If a user intends to **downgrade** firmware to versions older than version v1.4.0 from newer versions, he/she must first download the configuration files and later reupload the configuration after finishing the upgrade process.

17.3.1 Importing an Excel file

Excel file can be imported without any external tools. This option can be used where there is no internet connection or only minor change has to be applied. This way of importing is not suitable for validation of Excel configuration file.

Generating configuration is a resource-intensive task. It might take up to five minutes depending on configuration complexity

Supported types of an Excel configuration: .xlsx, .xlsm, .xltm, .xltx

To upload an Excel file, open *Protocol Hub->Configuration* screen in Web interface, select *Configuration file* and press *Import configuration*.

17.3.2 Generating .zip file

To accelerate a task of generating configuration a computer can be used. For this user should download *WCC Excel Utility* application. Upon opening an application, user should search for a field called *Excel file* which lets to choose an Excel file for which a conversion should be made. *Output file* should be filled out automatically, however, this value can be edited.

To make a conversion press *Convert*. If there are no errors found in the configuration, output file should contain the generated configuration, otherwise, error message is shown to a user.

This .zip file can be uploaded via Web interface, using the same tools as used for import of an Excel file.

17.3.3 Uploading configuration remotely

As of WCC Lite version v1.4.0 generated configuration files can be uploaded by a click of button. There are four parameters (not counting the configuration file itself) that have to be filled in before starting upload:

- *Hostname*: an IP address for device to connect to. This field conforms to hostname rules, therefore, if invalid value is selected, it is reset to default (192.168.1.1);
- *Port*: a PORT number to which a SSH connection can be made; valid values fall into a range between 1 and 65535; if invalid value is selected, it is reset to default (22);
- *Username*: a username which is used to make a SSH connection; make sure this user has enough rights, preferably *root*;
- *Password*: a password of a user used for establishing a SSH connection;

Configuration can only be uploaded if a port used for SSH connection is open for IP address filled in hostname entry field. Please check WCC Lite firewall settings in case of connection failure;

To upload a configuration remotely, press *Upload configuration*. If no errors occur, you should finally be met with text output mentioning configuration has been applied. During the course of upload process the aforementioned button is disabled to prevent spanning multiple concurrent processes.

18 Programmable logic controller

A programmable logic controller (PLC) is a digital device adapted for control of processes which require high reliability, ease of programming and real-time responses. Such functionality has long since replaced hard-wired relays, timers and sequencers which would be required to complete various tasks.

Programmable logic controllers usually had to conform to IEC 61131-3 standard which defines four programming languages: function block diagram (FBD), ladder diagram (LD), structured text (ST) and sequential function chart (SFC). This standard does not support distributed control systems therefore IEC 61499 standard was published in 2005. The standard is considered an extension of IEC 61131-3 standard.

WCC Lite supports PLC functionality while conforming to specifications of IEC 61499 standard.

18.1 IEC 61499

IEC 61499-1 defines the architecture for distributed systems. In IEC 61499 the cyclic execution model of IEC 61131 is replaced by an event driven execution model. The event driven execution model allows for an explicit specification of the execution order of function blocks. If necessary, periodically executed applications can be implemented by using the E_CYCLE function block for the generation of periodic events.

IEC 61499 enables an application-centric design, in which one or more applications, defined by networks of interconnected function blocks, are created for the whole system and subsequently distributed to the available devices. All devices within a system are described within a device model. The topology of the system is reflected by the system model. The distribution of an application is described within the mapping model. Therefore, applications of a system are distributable but maintained together.

Like IEC 61131-3 function blocks, IEC 61499 function block types specify both an interface and an implementation. In contrast to IEC 61131-3, an IEC 61499 interface contains event inputs and outputs in addition to data inputs and outputs. Events can be associated with data inputs and outputs by WITH constraints. IEC 61499 defines several function block types, all of which can contain a behavior description in terms of service sequences:

- *Service interface function block* – SIFB: The source code is hidden and its functionality is only described by service sequences;
- *Basic function block* - BFB: Its functionality is described in terms of an Execution Control Chart (ECC), which is similar to a state diagram (UML). Every state can have several actions. Each action references one or zero algorithms and one or zero events. Algorithms can be implemented as defined in compliant standards.
- *Composite function block* - CFB: Its functionality is defined by a function block network.
- *Adapter interfaces*: An adapter interface is not a real function block. It combines several events and data connections within one connection and provides an interface concept to separate specification and implementation.
- *Subapplication*: Its functionality is also defined as a function block network. In contrast to CFBs, subapplications can be distributed.

To maintain the applications on a device IEC 61499 provides a management model. The device manager maintains the lifecycle of any resource and manages the communication with the software tools (e.g., configuration tool, agent) via management commands.

18.1.1 4Diac framework

Figure 15: 4Diac IDE 1.11.3

The PLC functionality in the WCC Lite is implemented using Eclipse 4diac framework, consisting of the 4diac IDE and the 4diac FORTE runtime. The system corresponds to IEC 61499, an extension of IEC 61131-3. For more in-depth instructions and function block reference please see the 4diac manual - this document is merely a quick start guide that emphasizes the specifics of tailoring the applications to run on the WCC Lite.

The 4diac IDE application is used to model logic sequences. An output file, *.fboot, is then generated and either loaded into the runtime for debugging purposes (functionality available from within the IDE), or uploaded into the controller for normal use via web interface.

During debugging, the output logic is executed directly in the runtime. Any logic loaded during debugging will be discarded after a reboot of the controller. Logic applications for regular use should be uploaded via the web interface.

It is possible to run multiple tasks at once. These tasks can either be implemented in the same screen or split into separate tasks. Please note, however, that all elements should have unique names, even between different tasks. As of 4diac IDE 1.11.3 this is not enforced between separate apps, however, 4Diac runtime application rejects such file purely because of naming issues.

The 4diac FORTE runtime is able to execute the aforementioned fboot files containing the logic. The FORTE runtime can be run on both the WCC Lite and a PC for debugging purposes. The runtime is integrated to interact with the REDIS database.

18.1.2 Example project

The best way to understand basics of 4Diac and WCC Lite collaboration is through an example project. This user manual intends to show the pieces needed to run PLC applications on WCC Lite. It is not intended to be definitive guide on how to use 4Diac IDE or how to interpret IEC 61499 standard.

During (at least) the first start of the IDE user will be asked to select a directory for the workspace as in Figure 16. Workspace is used to save files needed for projects.

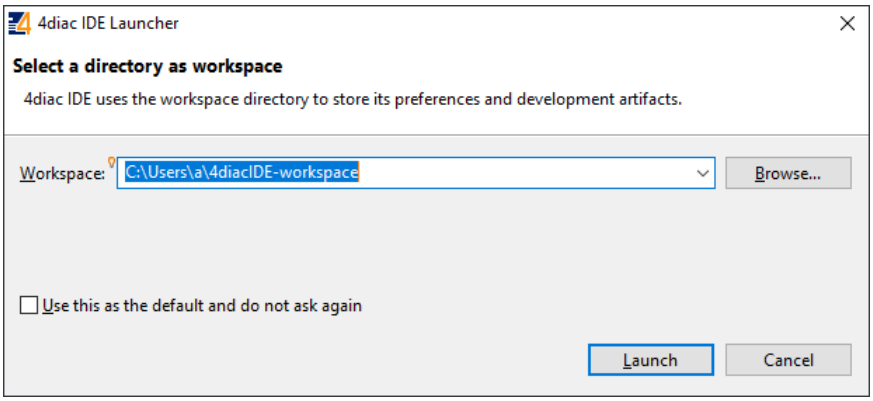


Figure 16: Selecting 4Diac IDE workspace

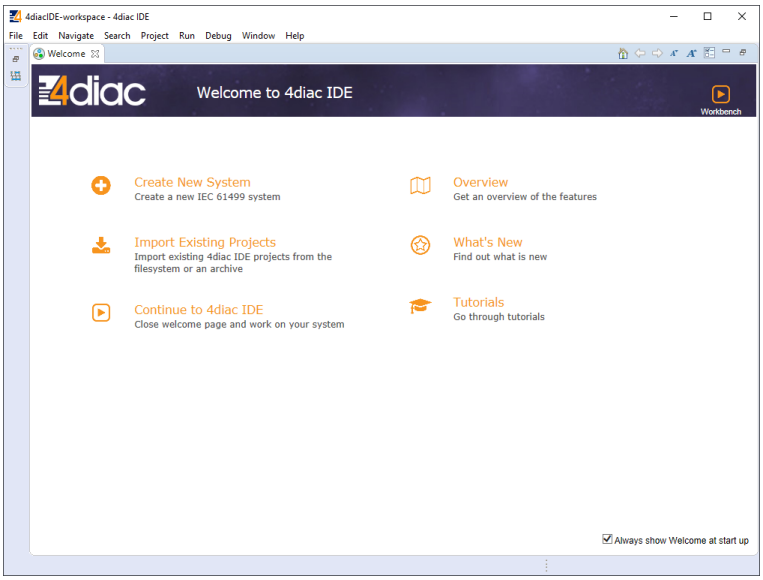


Figure 17: 4Diac welcome screen

After that a user should be met by the welcome window as in Figure 17. If such window is not shown, one can create create project by selecting *File->New->Project* and filling in the required fields (figure 18).

To create a simple application, simply drag and drop objects from the palette to the canvas and wire them accordingly. Event trigger and data pathways cannot be connected to one another. Displayed below is an example of a simple blinker application (figure 19).

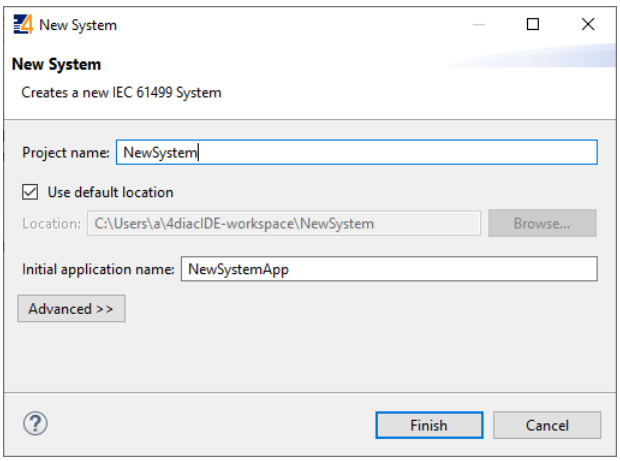


Figure 18: 4Diac new project window

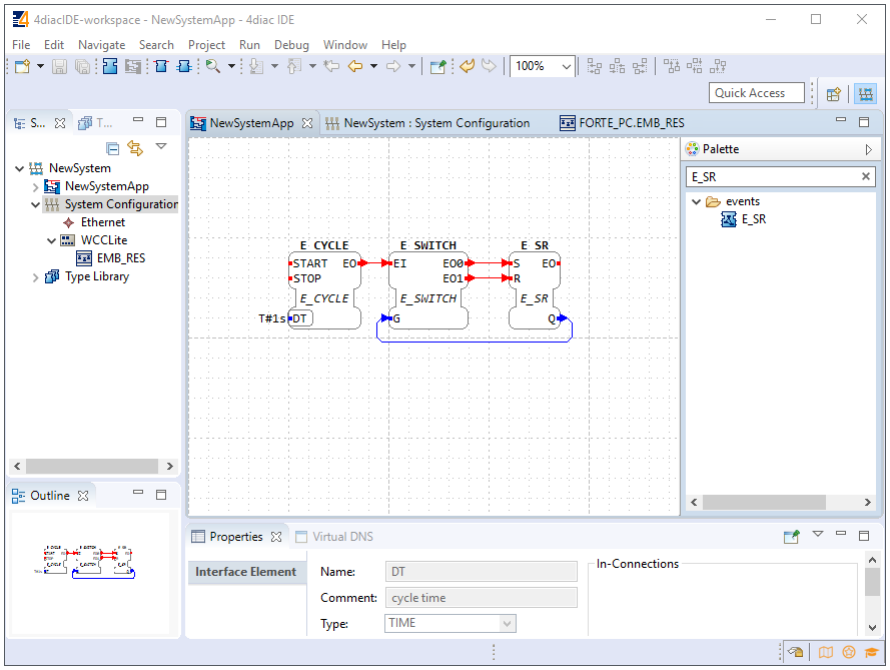


Figure 19: Simple blinker application

Having less wiring by connecting several signals to same subnet as PCB designer (such as Altium Designer) as of 4Diac IDE 1.11.3 is not supported. However, if some parts are used frequently, it is highly advised to have less wiring by simply compiling several elements into a subapplication. For this, you would have to select elements to be grouped, press right key and select *New Subapplication*. You can later change names of such elements and its pins.

In the System Configuration section, drag and drop a FORTE_PC device, an Ethernet segment and link them (figure 20). For debugging in the local (PC) runtime, leave the address ”localhost:61499”. For testing on a WCC Lite, enter the IP address of the device, along with the port number (which by default is 61499 as well).

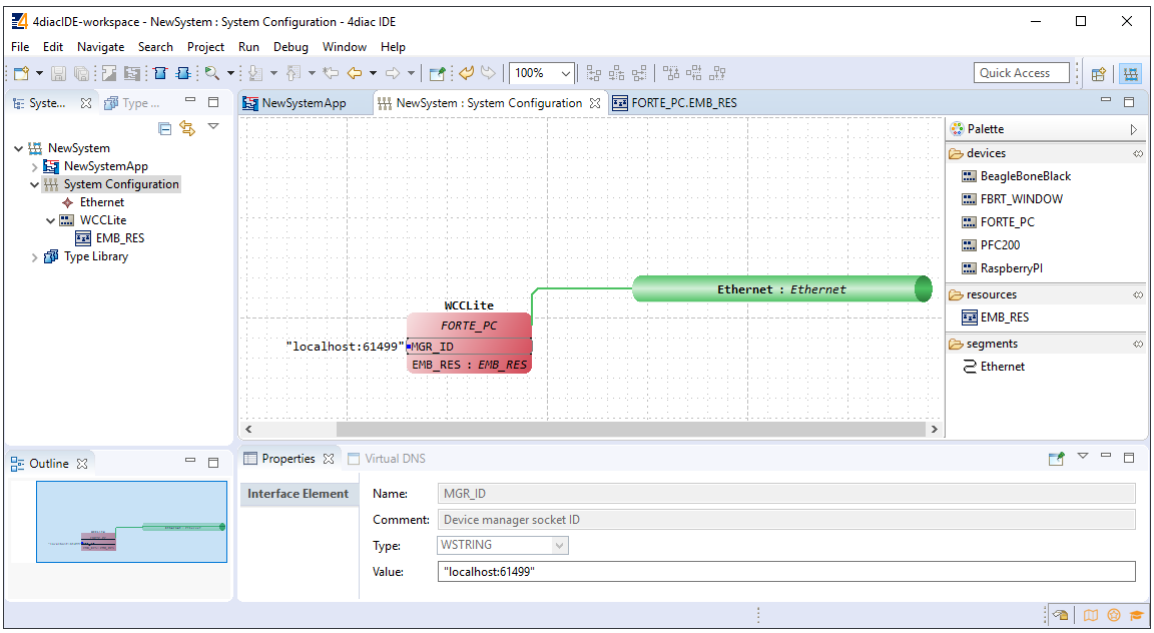


Figure 20: System configuration and network settings

In order to deploy the application, the circuit needs to be mapped to the controller. For a non-distributed application (distributed application cases will not be discussed in this chapter), all the FBs of the application need to be selected and mapped to the configured controller as shown in figure 21.

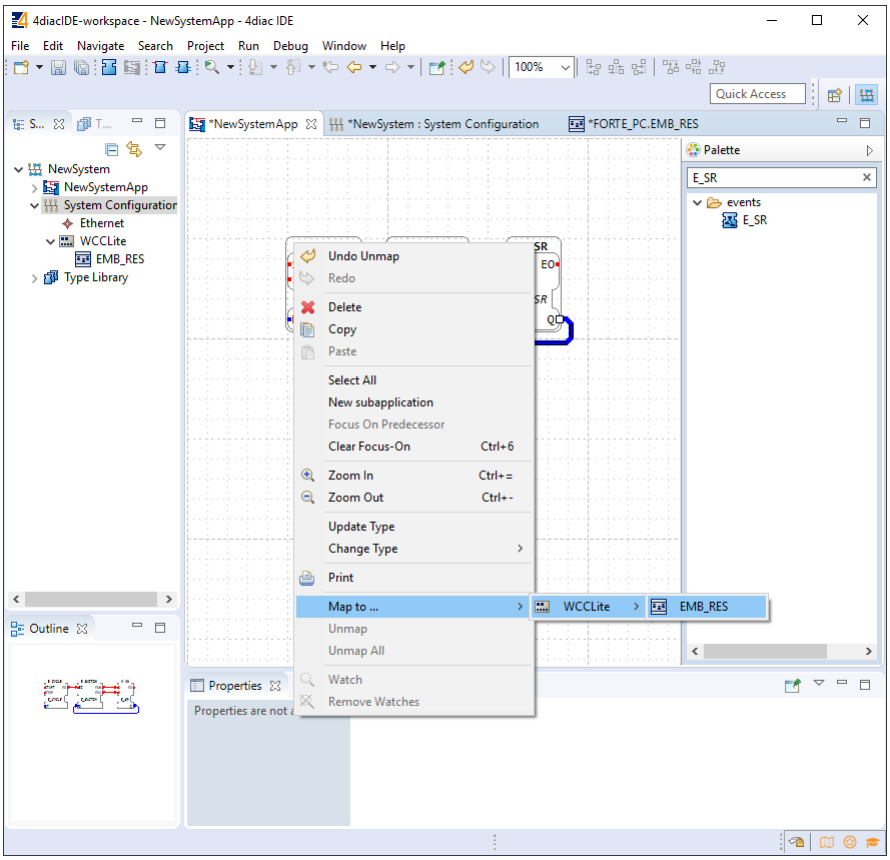


Figure 21: Mapping the logic to the controller

To start the application execution, an initial trigger needs to be present. For a non-distributed

application, the initial event trigger needs to be wired from the START function block in the resource section as shown in figure 22.

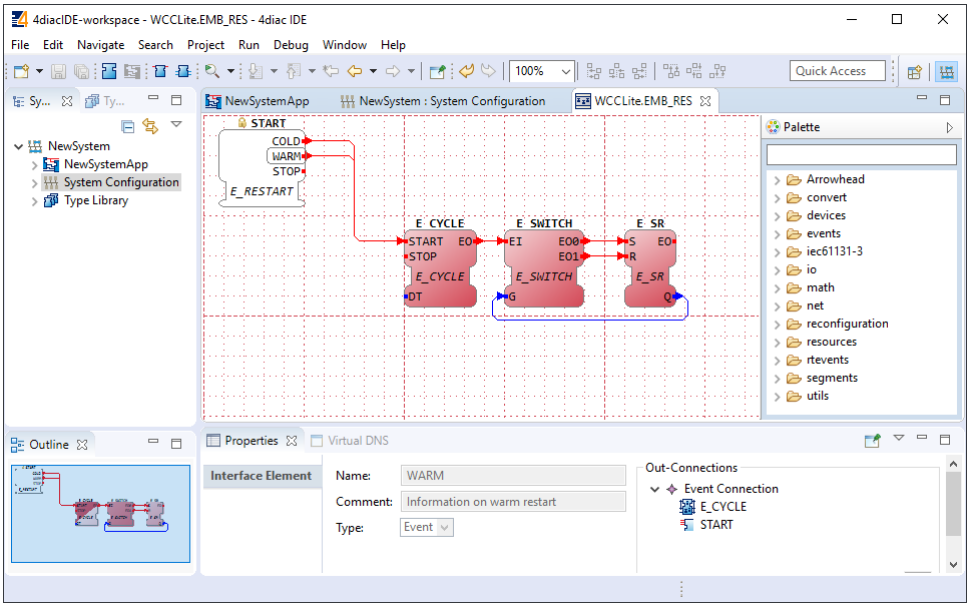


Figure 22: Mapping the logic to the controller

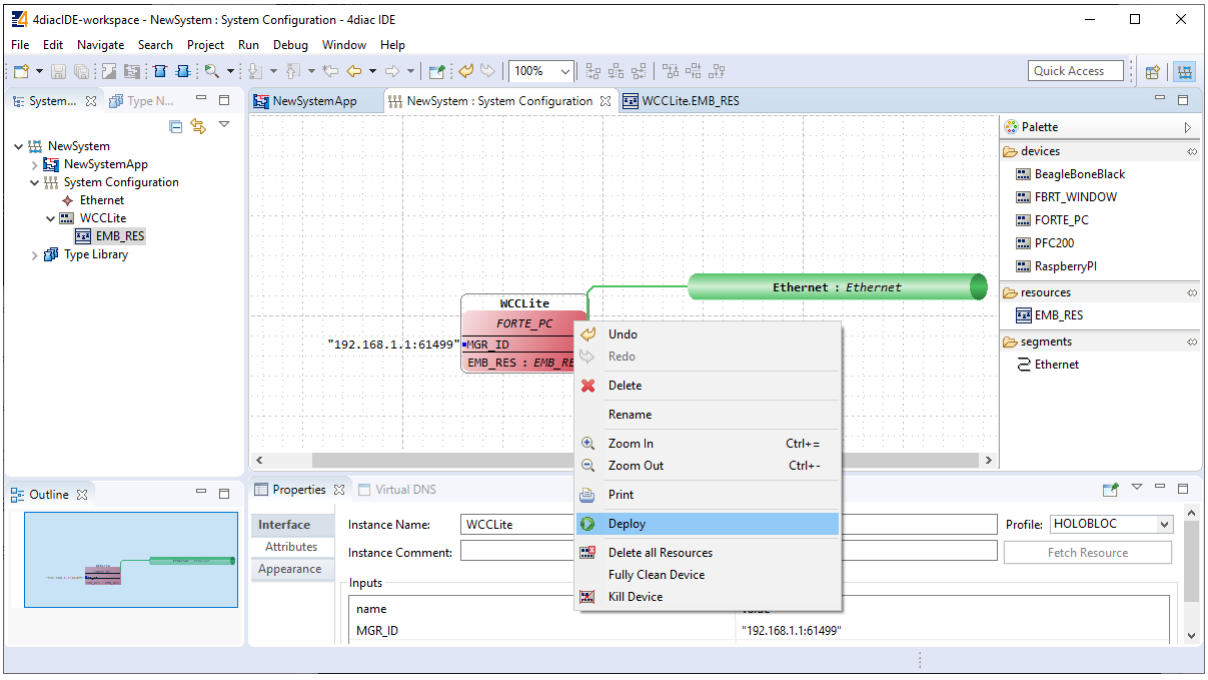


Figure 23: Deploying of the application

To deploy the application, go to the System Configuration tab and simply select ”Deploy” from the right-click menu of the controller device (figure 23). If a running application exist in the runtime, you may be asked whether you want to replace it. This will only overwrite the application in the memory and not the storage. If the controller is restarted, the old application will be loaded from the non-volatile memory of the controller.

18.1.3 Configuring data endpoints

To use WCC Lite as a programmable logic controller, it needs to be configured in a particular way. The PLC functionality of the WCC Lite only allows for the use of data that is has been configured in the Excel configuration spreadsheet. This has been done for security purposes and to preserve transmission medium only for data that is available. Only topics defined in the configuration can post or get data. If a certain data entry exists but it has not been linked to a PLC program, all calls from PLC runtime application to Redis database will be ignored. Therefore it is highly advised to prepare and upload the Excel configuration before using this signal in the PLC application.

Some parameters are mandatory for PLC usage. These parameters are shown in two tables below (one for *Devices*, one for *Signals* tab). Please note that other parameters can be used as well, but are not covered because they aren't specific to PLC functionality.

Table 34: Mandatory parameters for *Devices* tab

Parameter	Type	Description
name	string	User-friendly device name
device_alias	string	Device alias to used in configuration
enable	boolean	Enabling/disabling of a device
protocol	string	Selection of protocol (IEC 61499)

Table 35: Mandatory parameters for *Signals* tab

Parameter	Type	Description
signal_name	string	User-friendly signal name
device_alias	string	Device alias from a Devices tab
signal_alias	string	Unique signal name to be used
source_device_alias	string	device_alias of a source device
source_signal_alias	string	signal_alias of a source signal
enable	boolean	Enabling/disabling of an individual signal

If an upload consisting of configuration for IEC 61499 has been succesful, one should be able to access a configuration stored in */etc/iec61499.json* file where protocol-specific parameters are shown in a JSON format. If the file is missing, make sure you have a correct firmware version installed and haven't made any typing errors.

Parameters mentioned earlier, namely *device_alias* and *signal_alias*, are the only parameters one needs to fill to bind Excel configuration to 4Diac framework. Two types of blocks are used for data transmission - PUBLISH blocks to write data to REDIS database and SUBSCRIBE blocks to acquire data from database as soon as it changes its value. Both of them have an ID connection. To connect a block to a datapoint, one should set this pin as *raw[].redis[device_alias,signal_alias]*, e.g. *raw[].redis[example_plc_device,example_plc_signal_alias]*.

An example with SUBSCRIBE and PUBLISH function blocks is shown below in figure 24.

Outputs of variable type ANY cannot be directly wired to inputs of the same type and therefore need to be explicitly typed using transitional function blocks

If every step until now has been succesful, a user could now start debugging a PLC application.

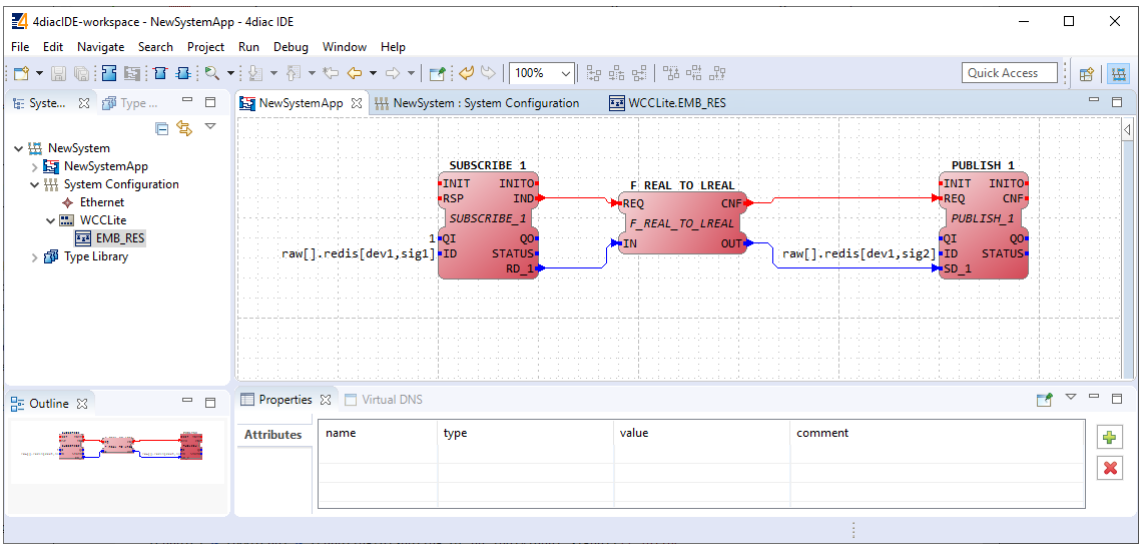


Figure 24: Subscribe and publish examples

18.1.4 Debugging an IEC 61499 application

After a project has been built and binded to an existing Excel configuration, a user would normally want to check if every part is working according to the prior requirements before compiling finished project and uploading it to production. Both 4Diac framework and WCC Lite offer tools for flexible debugging.

There is a possibility that 4Diac FORTE might not start as a process. It may happen if multiple faults occured and process has stopped. Process is also programmed to not start if no excel configuration file is found, therefore a user should make sure that Excel configuration is uploaded and ready for use.

Individual function blocks can be set to Watch mode: events can be triggered and values can be forced at inputs or outputs (figures 25 and 26). To monitor the function blocks, the application should be deployed and the IDE should be in Online mode (*Debug -> Monitor System -> NewSystem*).

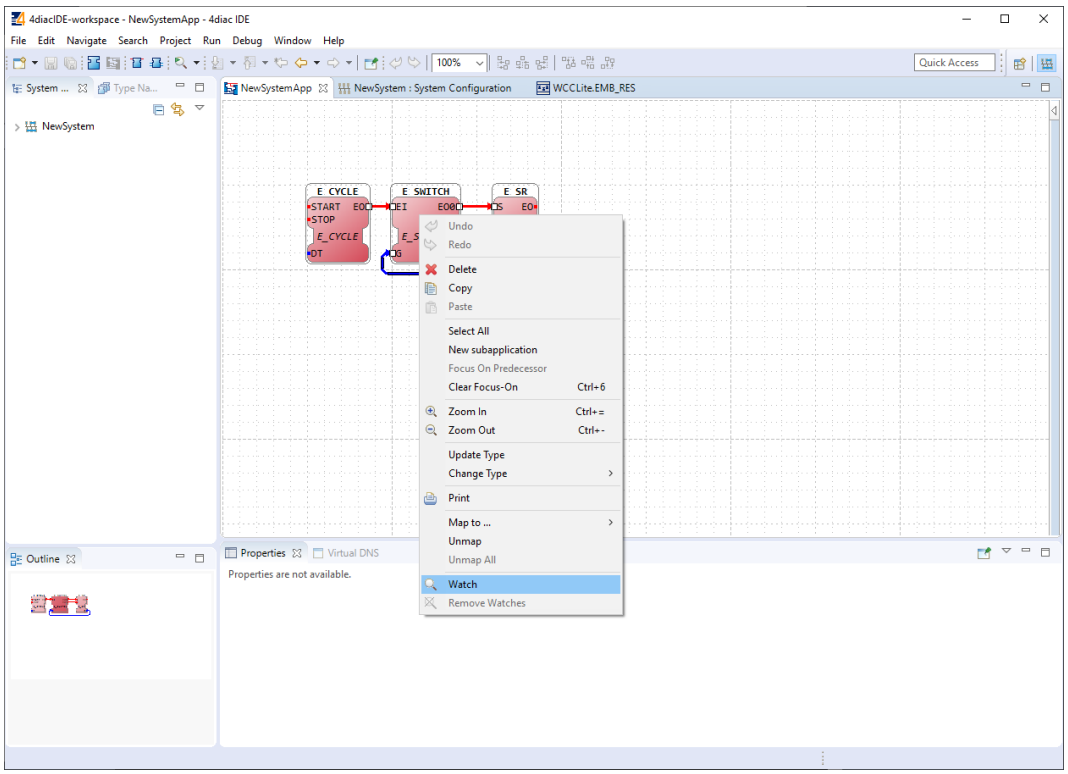


Figure 25: Selecting watch mode

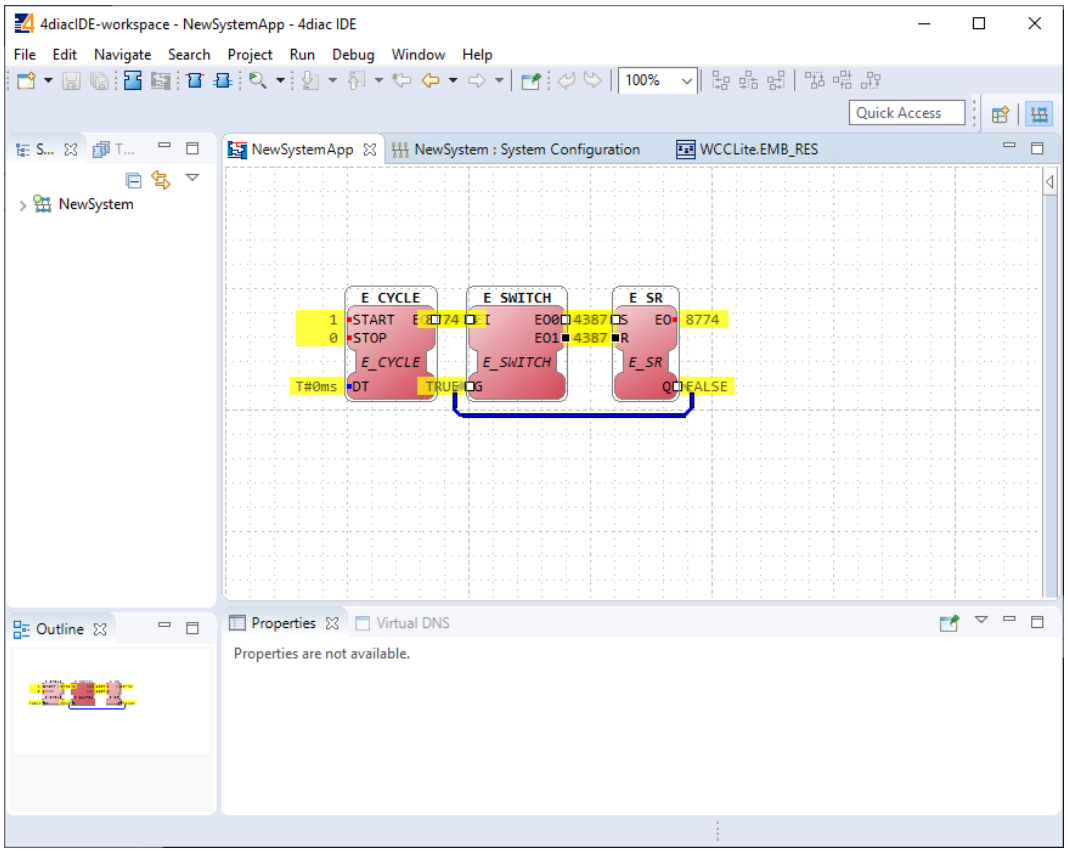


Figure 26: Function blocks in watch mode

Seeing information dynamically updated on 4Diac IDE might be very informative, however, some applications might require accesing WCC Lite via command-line interface. For example, in case of

Table 36: 4Diac FORTE command line debugging options

Option	Description
-h	Display help information
-c <IP>:<port>	Set the listening IP and port for the incoming connections
-r	Show redis messages
-d<debug level>	Set debugging level
-f	Set the boot-file where to read from to load the applications

information not being updated one would want to assure that 4Diac FORTE in WCC Lite is not filtering data out but sending it to internal database (Redis). To run 4Diac FORTE debug from command-line interface, a user should write *forte* and press *Enter*. All possible choices are shown by adding *-h* flag. More flags are shown in a Table 36. Make sure to stop any running process that could use the address that 4Diac framework is going to use.

18.1.5 Generating and uploading FORTE logic file

After the PLC design is finished and debugged, such design can be compiled into FBOOT file and uploaded to one or multiple devices to be used in production. As application being debuggged is not automatically considered as a default application, one should be uploaded explicitly via web interface.

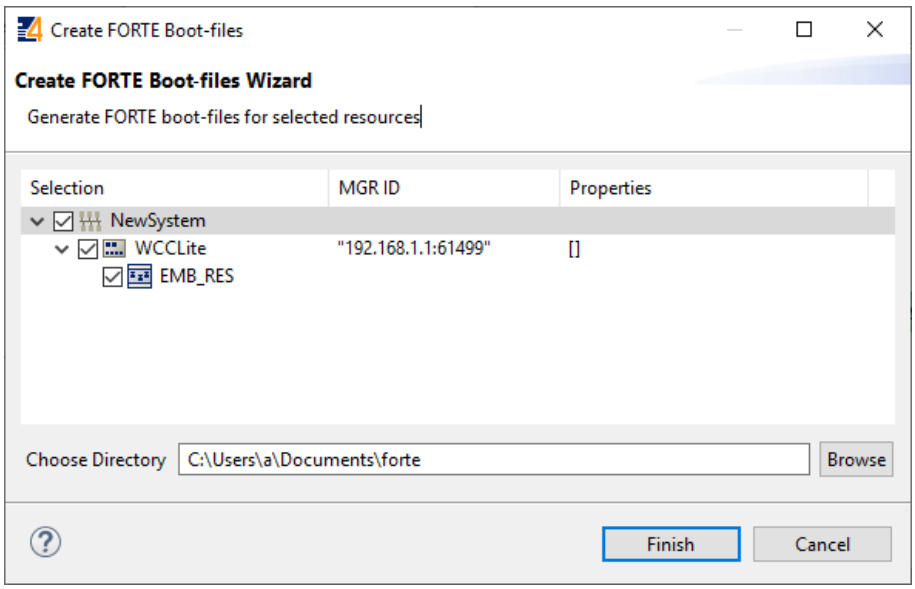


Figure 27: Generating FBOOT file

To generate FORTE boot-files a user should select *Run->Create FORTE boot-file...*. After that one should select devices which should have their boot files created as well as additional devices' properties and directory where these files should be stored as in Figure 27.

Upload button for FORTE file in web interface can be found in *Protocol Hub* tab, *Configuration* screen (FORTE boot file upload supported for versions v1.4.0 and above). You should see a view as in Figure 28. Please be noted that only files with *.fboot extension are allowed.

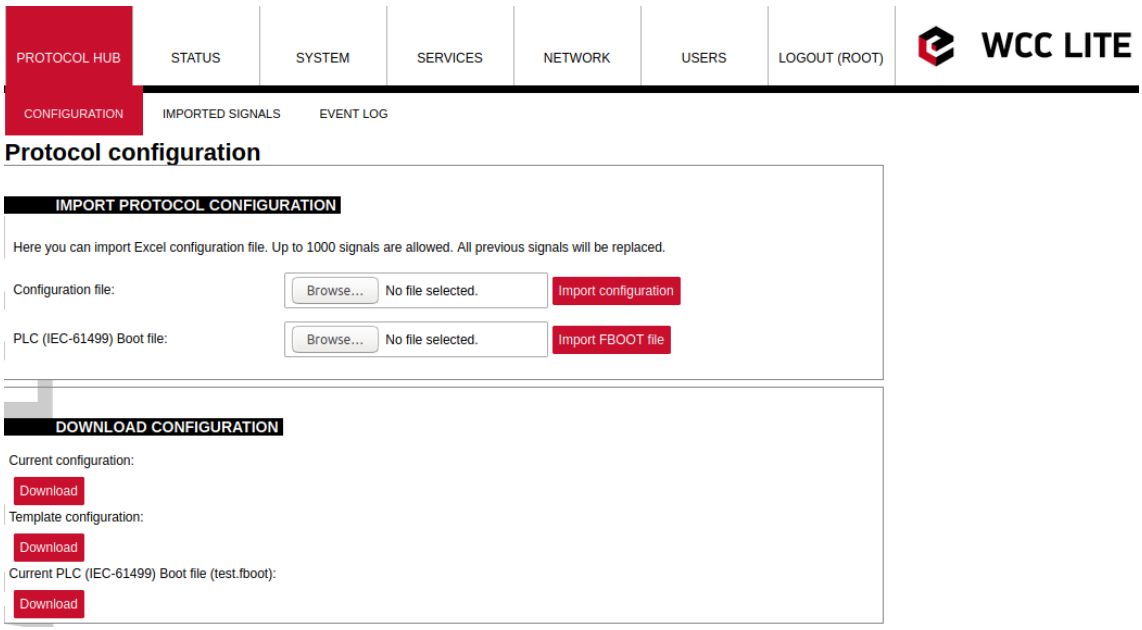


Figure 28: Upload and download of 4Diac configuration files

After the file has been imported one should be able to download it from the same screen as seen in Figure 28.

Uploading file saves its name and shows it in web interface. It is advised to carefully choose filename to separate different versions of PLC application files.

18.1.6 Distributed control application

IEC 64199 standard introduced requirements for a distributed control. This means that multiple devices can change information between them and make their own decisions based on the data they receive from other sources. This enables distributed applications between multiple WCC Lite devices and all other devices that support IEC 61499.

Communication between devices can be configured using:

- Publish/Subscribe function blocks (via UDP packets);
- Client/Server function blocks (via TCP packets).

A Publish block can publish data messages using UDP multi-cast addresses meaning that multiple devices would be able to simultaneously get the same data. However, one would have to make sure that all of the devices support multi-cast option.

This user manual will only cover setting up point-to-point communication between devices via Publish/Subscribe blocks. For more information on communication between several IEC 61499 devices please check documentation for *Eclipse 4diac framework*.

Let's say we would like to count how many times the light has been turned on. For this we can add counting functionality to application shown in Figure 19. The application should run on 2 devices. The blinking part of the application will run on a 4diac FORTE and the count on another 4diac FORTE, see the architecture below in Figure 29. The two different programs running on two separate WCC

Lite devices emulate two PLCs. Two different devices can be identified by different colors of function blocks. One can identify device and it properties by accessing System Configuration screen as seen in Figure 30. Yellow function blocks belong to WCC_212 device which can be accessed through 192.168.4.212 (port number 61499) whereas brown function blocks belong to WCC_218 device which can accessed through 192.168.4.212 (port number 61499).

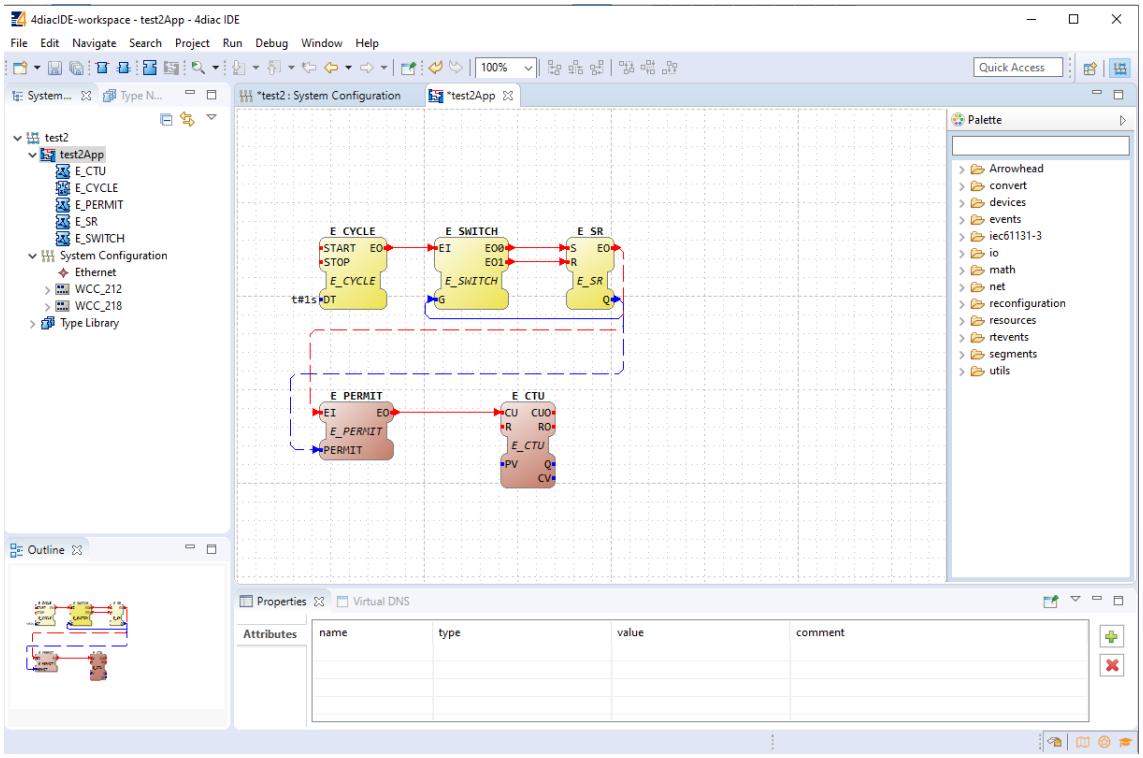


Figure 29: Example blinking application as a distributed system

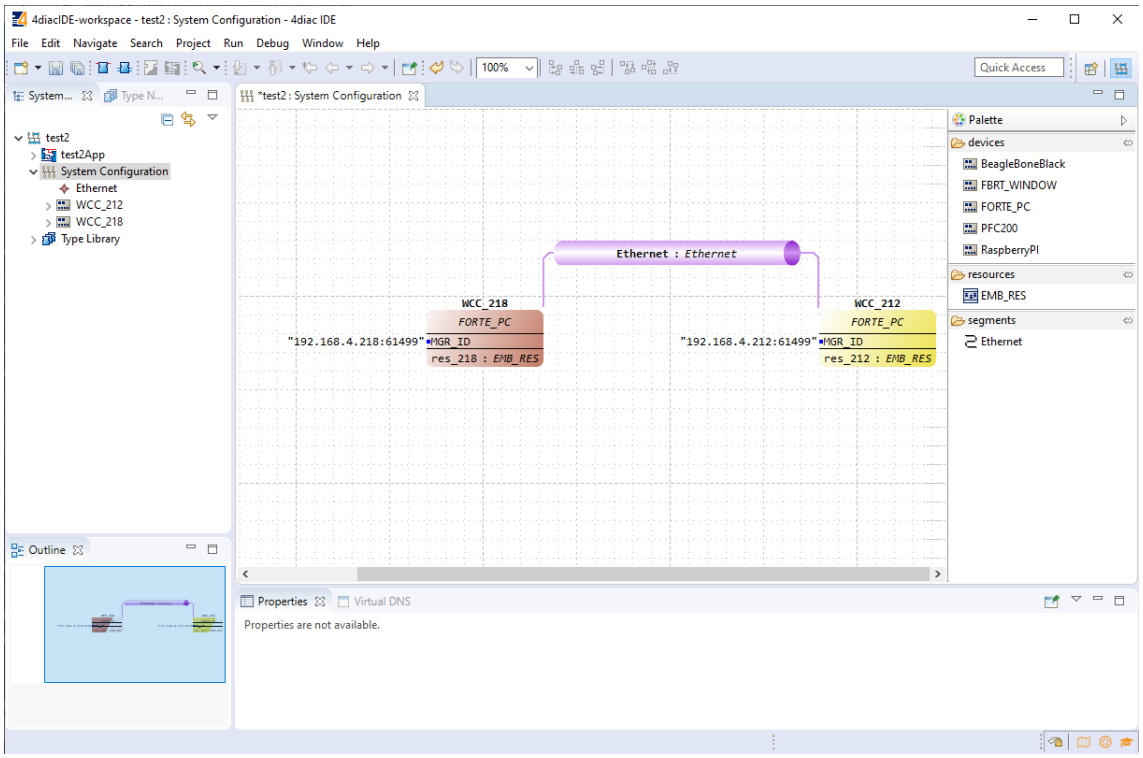


Figure 30: Example system configuration for a distributed system

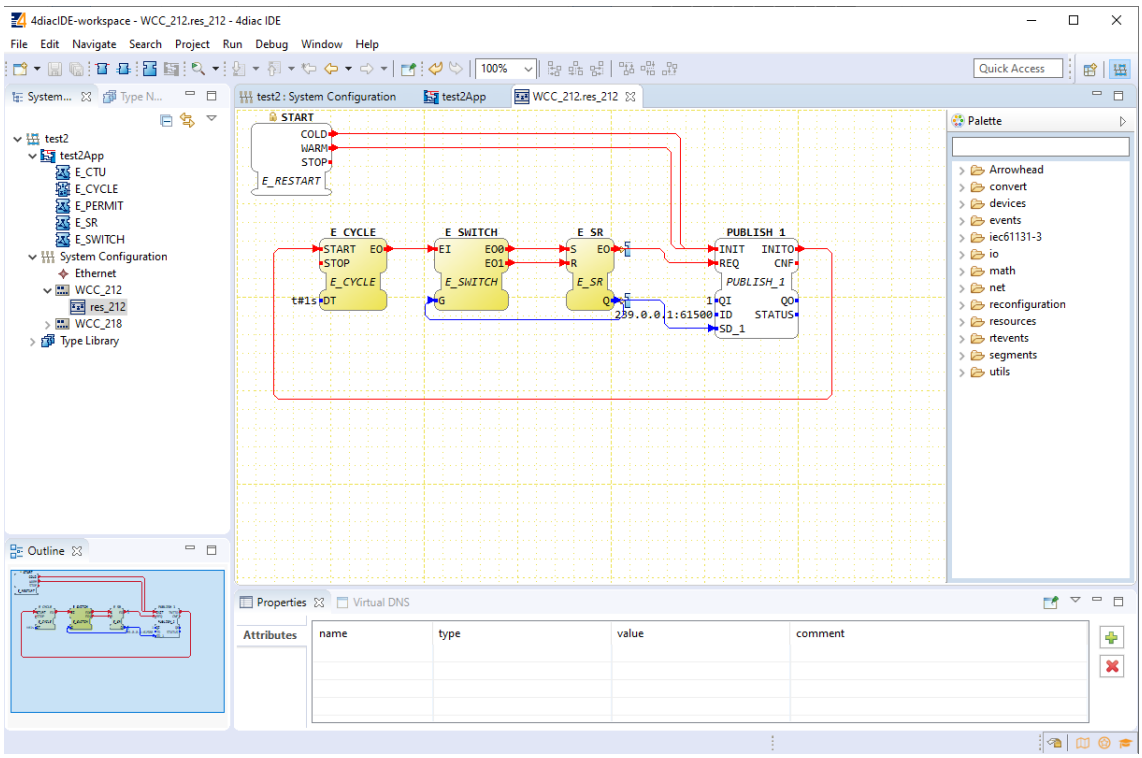


Figure 31: Example app for blinking part of a distributed system

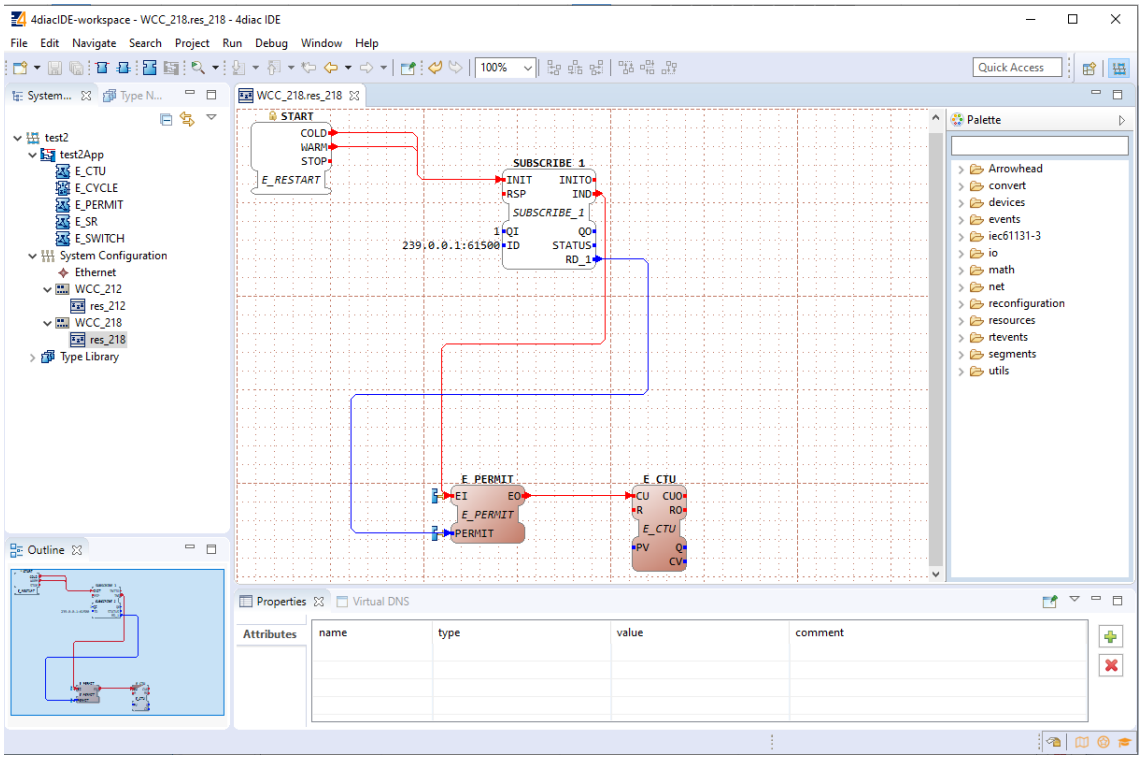


Figure 32: Example app for counting part of a distributed system

One part of the distributed application, simple blinker is acquired from a Figure 19 and can be found in 31. Counting part is done in another device as seen in application in Figure 32.

To count the blinking, two new Function Blocks (FBs) have been added to the existing application for a different device (WCC_218):

- E_PERMIT
- E_CTU

To communicate between devices, an additional PUBLISH_X/SUBSCRIBE_X pair must be used. As one can identify, these blocks are not seen when looking at a whole distributed system and should be seen as an intermediary between devices.

The PUBLISH_X FB is used to send messages over the network which are received by an according SUBSCRIBE_X FB. Every time a REQ is triggered, a message is sent according to the ID input. With the value of the ID input you can specify what specific network protocol you would like to use (e.g., MQTT). If you don't specify a dedicated protocol the default as defined in the "IEC 61499 Compliance Profile for Feasibility Demonstrations" is used. The number X in PUBLISH_X is the number of data elements that you want to send in the message. Since we are only sending one value we used PUBLISH_1.

The used ID value specifies an IP:PORT pair.

19 MQTT

19.1 Overview

MQTT (short for MQ Telemetry Transport) is an open OASIS and ISO standard (ISO/IEC PRF 20922) lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP, although its variant, MQTT-SN, is used over other transports such as UDP or Bluetooth. It is designed for connections with remote locations where a small code footprint is required or the network bandwidth is limited.

The broker acts as a post office, MQTT doesn't use the address of the intended recipient but uses the subject line called "Topic", and anyone who wants a copy of that message will subscribe to that topic. Multiple clients can receive the message from a single broker (one to many capability). Similarly, multiple publishers can publish topics to a single subscriber (many to one).

Each client can both produce and receive data by both publishing and subscribing, i.e. the devices can publish sensor data and still be able to receive the configuration information or control commands. This helps in both sharing data, managing and controlling devices.

With MQTT broker architecture the devices and application becomes decoupled and more secure. MQTT might use Transport Layer Security (TLS) encryption with user name, password protected connections, and optional certifications that requires clients to provide a certificate file that matches with the server's. The clients are unaware of each others IP address.

The broker can store the data in the form of retained messages so that new subscribers to the topic can get the last value straight away.

The main advantages of MQTT broker are:

- Eliminates vulnerable and insecure client connections
- Can easily scale from a single device to thousands
- Manages and tracks all client connection states, including security credentials and certificates
- Reduced network strain without compromising the security (cellular or satellite network)

Each connection to the broker can specify a quality of service measure. These are classified in increasing order of overhead:

- At most once - the message is sent only once and the client and broker take no additional steps to acknowledge delivery (fire and forget).
- At least once - the message is re-tried by the sender multiple times until acknowledgement is received (acknowledged delivery).
- Exactly once - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received (assured delivery).

19.2 Using WCC Lite as MQTT Client

WCC Lite supports MQTT messaging compatible with MQTT v3.1 standard (starting from version 1.4.0). Such messaging is possible via mapping of Redis and MQTT data therefore data can be

transmitted from any protocol that is supported by WCC Lite. MQTT serves as an alternative for protocols conforming to IEC standards, for example, to send data to a cloud infrastructure that supports MQTT instead of IEC-60870-5-104.

All standard functions, except for data encryption, are supported. Encrypted messages are not supported yet, therefore to ensure security a user would have to use a VPN service. A user can choose from three different Quality of Service levels, select if messages are to be retained, authenticate users and optionally send Last Will messages.

To configure WCC Lite a user can fill in the needed parameters in Excel configuration. These parameters are shown in two tables below.

Table 37: MQTT parameters for *Devices* tab

Parameter	Type	Description	Mandatory
name	string	User-friendly device name	
device_alias	string	Device alias to used in configuration	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Selection of protocol (MQTT)	Yes
host	string	MQTT broker IP address selection	Yes
port	integer	MQTT broker port selection (default - 1883)	
enable_threshold	bool	A parameter to determine if identical values should not be sent multiple times in a row. Default - 1 (true).	
gi_interval_sec	integer	Parameter to determine how frequently should **all** values be sent at once. Disabled if equal to 0. Default - 0	
mqtt_qos	integer	MQTT Quality of Service for message as in standard (default - 0)	
mqtt_retain	boolean	Selecting if MQTT broker should retain last received messages (default - False)	
user	string	MQTT user name	Yes
password	string	MQTT user password	Yes
use_last_will	boolean	Selecting if MQTT should use Last Will and Testament functionality (default - False)	
last_will_topic	string	Topic to which an MQTT message would be sent if the device abruptly disconnected message broker	If use_last_will=True
last_will_message	string	Message to be sent over MQTT if the device abruptly disconnected message broker	
last_will_qos	integer	MQTT Quality of Service selection as in standard (default - 0)	
last_will_retain	boolean	Selecting if MQTT broker should retain last will message (default - False)	

To map the signal to send through MQTT client, it should have its **device_alias** and **signal_alias** mapped to **source_device_alias** and **source_signal_alias** respectively.

If MQTT is configured but does not send data, a user can use command line interface to debug transmission. All options for MQTT process which transmits data over MQTT (called *mqtt-client* as

Table 38: MQTT parameters for *Signals* tab

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly signal name	
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique signal name to be used	Yes
source_device_alias	string	device_alias of a source device	
source_signal_alias	string	signal_alias of a source signal	
enable	boolean	Enabling/disabling of an individual signal	Yes
topic	string	Topic name to override the value built by default	

Table 39: MQTT (*mqtt-client*) command line debugging options

Option	Description
-h [–help]	Display help information
-c [–config]	Configuration file location (default - /etc/elseta-mqtt.conf)
-V [–version]	Show version
-d<debug level> [–debug]	Set debugging level
-r [–redis]	Show REDIS output
-m [–mqtt]	Show MQTT output

it works as an adapter between REDIS and MQTT data) are shown in Table 39. It is necessary to make sure that no copies of the same process are running in a background because broker can stop old session because of multiple connections from the same user from the same IP.

19.3 MQTT data format

The format of a MQTT message is a bit different than Redis messages. Redis messages are supported as CSV strings: *value,timeStamp,flags* (where *value* can be float, integer or *nan*; *timeStamp* - Unix timestamp in milliseconds; *flags* contain additional information about a measurement). MQTT messages are supported as *value,timestamp,quality* (where *value* can be float, integer or *nan*; *timeStamp* - Unix timestamp in milliseconds; *quality* shows if a value is to be considered as valid). Quality parts of a string is always equal to 1 except for Redis messages containing invalid (IV), non-topic (NT) and/or overflow (OV) flags.

As mentioned, MQTT client acts as an adapter between Redis and MQTT, therefore data from topic in Redis is written to a topic in MQTT. Therefore *mqtt-client* has to know the mapping table before starting. This table is saved at */etc/elseta-mqtt.json*. Every Redis topic name is constructed as *tag/[device_alias]/[signal_alias]/[direction]*. Prefix *tag/* is always used before the rest of argument. *device_alias* and *signal_alias* represent columns in Excel configuration. Direction can have one of four possible values - *rout*, *out*, *in*, *rin*; all of which depend on the direction data is sent or acquired protocol-wise. The same Redis topic structure is preserved in MQTT by default making it easier to find matching signals, however, as no recalculation is done by MQTT and only PUBLISH messages are now supported, only Redis signals with *in* direction have their MQTT mappings.

A user can create and select his own topic name in Excel configuration, in *topic* column. As no recalculation is done by MQTT and only PUBLISH messages are now supported, only Redis signals with *in* direction have their MQTT mappings.

20 Certificates

Devices that send unencrypted data are susceptible to attacks which might cause deliberate damage to the user system. Therefore it is highly advised to use cryptography to secure the sensitive data. WCC Lite offers means to easily store certificates for their later usage.

Some protocols, namely IEC-60870-5-104 Slave, DNP v3.0 Slave and Master might be configured to send data over TCP/IP. For these protocols, secured connection over TCP/IP using TLS certificates can be made. For this purpose, certificate storage has been created and is available since firmware version 1.3.0.

To make storage secure, multiple steps have been taken:

- By default certificate storage is only accessible for root user and users with group level 15 permissions;
- By default certificates are not added to backup to avoid private key leakages; private keys should never be revealed to public;
- By default certificates are deleted after system upgrade;
- Only basic information is shown on a web interface; certificates can be uploaded, deleted but not downloaded.

Certificates can be split into three parts - local (private) certificate, certificate from peer (usually called Certificate Authority (CA)) and private key. It has to be noted that all of these certificates sometimes can be found in one file, therefore ideally a user should have at least minimal understanding about formats in which certificates are stored.

Certificates should conform to X509 standard. The difference between local certificate and certificate authority certificates is that only certificate authority generates certificates for others. Therefore *Issuer* and *Subject* fields are always the same for certificate authority certificate whereas they differ for local certificates. Both of these certificates are usually stored in a device to validate if incoming connections have valid certificates and are to be trusted. Both of the certificates have the public key which together with public key enable having encrypted connections.

Some applications might let user make secured connections without certificate authority validation however for maximum security it is advised to store both local and peer certificates in the device.

The private key is a text file used initially to generate a Certificate Signing Request (CSR), and later to secure and verify connections using the certificate created per that request. It usually contains a unique hash made in a way that chances of guessing it by using brute force are technically infeasible. The private key should be closely guarded, since anyone with access to it use it in nefarious ways. If you lose your private key, or believe it was compromised in any way, it is recommended to *re-key* your certificate – reissue it with a new private key.

To make certificate upload more intuitive, certain restrictions are imposed. Only files with certain extensions (*.crt, *.pem, *.der, *.key) can be uploaded. Trying to upload other files will result in an error message. Certificate storage should be considered a folder with certain access restrictions, therefore file names should be unique for every file.

It should be noted that this chapter only reviews main certificates and suggest means to use them for *Protocol Hub* services. Certificates can also be used for other causes, e.g. to secure VPN

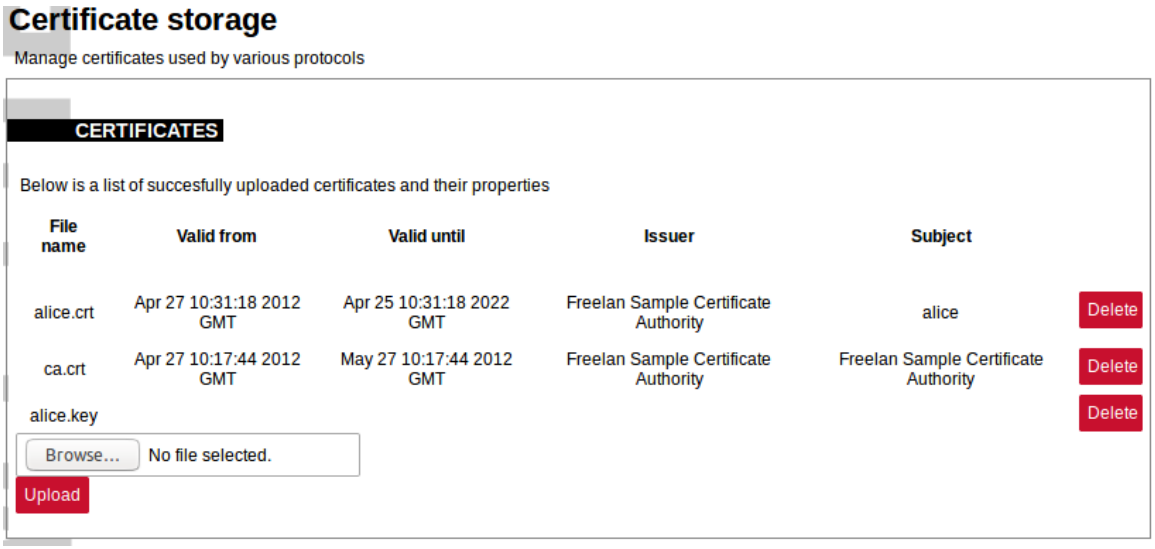


Figure 33: Interface for certificate storage

connections. For the sake of simplicity, uploading certificates and their usage are explained in their respective chapters where applicable.

To get more details about how one could use TLS for Protocol Hub protocols please check section *Excel configuration format*.

To find out more about why certificates help keep device secure please check section *Cyber security* or check X.509 and RFC 5755 standard.

21 Cyber security

WCC Lite is based on OpenWRT operating system. OpenWrt is described as a Linux distribution for embedded devices. WCC Lite has same functionality as Linux OS including user management.

Basic configuration on WCC Lite can be done using web based frontend. More advanced configuration is available over terminal interface. For secure web access, WCC Lite can be accessed via HTTPS (TLS) instead of the unencrypted HTTP protocol. You can use *openssl* utility to generate your own certificate authority and certificates to be used on web interface. Certificates can also be named or placed in whatever directory you wish by editing */etc/lighttpd/lighttpd.conf*.

Terminal is accessible over Telnet or SSH. For security reasons we strongly recommend to use SSH. SSH, also known as *Secure Socket Shell*, is a network protocol that provides administrators with a secure way to access a remote computer. SSH also refers to the suite of utilities that implement the protocol. Secure shell provides strong authentication and secure encrypted data communications between two computers connecting over an insecure network such as the Internet. SSH is widely used by network administrators for managing systems and applications remotely, allowing them to log in to another computer over a network, execute commands and move files from one computer to another.

21.1 User rights

Depending on the user name, different rights are defined: admin is generally entitled to make changes while user does not have any editing permissions, the relevant buttons are disabled. User can be assigned to one of fifteen user groups that can access different amounts of device parameters. Highest (fifteenth) permision level grants the same permission as root user has. User group rights can be edited to give more rights or restrictions, except for highest (15th) level.

21.1.1 User management and rights authentication

WCC Lite provides different authentication mechanisms:

- Authentication via locally stored credentials. In this scenario all users, passwords and permissions are encrypted and stored in internal WCC Lite storage.
- Authentication via external RADIUS Server. In this scenario all users, passwords and permissions (profiles) are defined in remote RADIUS Server. Login into WCC Lite is available only if RADIUS Server will grant authentication and will provide user profile with user rights on that device (more detailed description below). This also means that a password for such user cannot be changed remotely.
- Authentication via external RADIUS Server with fallback option. In this scenario users will be authenticated via RADIUS server. If server fails to respond (configured timeout is passed) WCC will use locally stored credentials. Fallback options are selected with PAM configuration.

By default only authentication via locally stored credentials is allowed. For authentication via external RADIUS server a user should at first enable RADIUS process and configure at least one server.

21.1.2 Locally stored credentials management

Device has predefined default users like *root* and *user*.

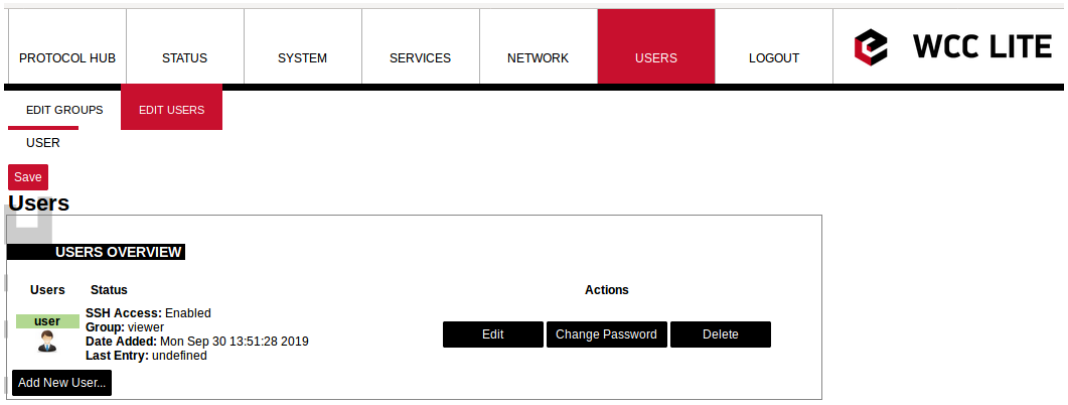


Figure 34: Screen containing all users

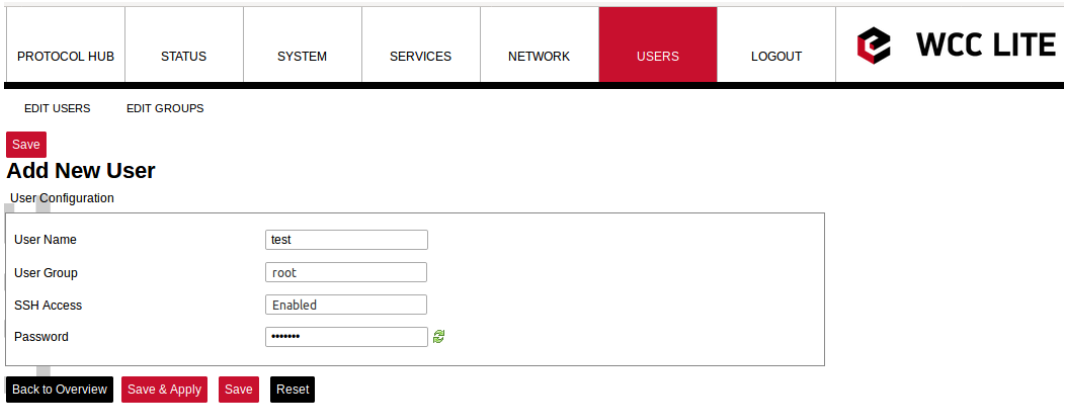


Figure 35: Screen for new user configuration

root user has full permission set to connect to WCC Lite over web interface and SSH or Telnet. This user is default user on WCC Lite and cannot be deleted. However, it is highly advised to change the default password to a different one less susceptible for attacks.

user is limited user on system and can't get root rights. A default password for access via command-line interface and web interface is *wcclite*. It is advised to change this password to increase a level of security.

System allows customer to set up even more users with well known commands like *adduser*, *passwd* and *userdel*. More users can also be added or edited via web interface as shown in Figure34 and Figure 35. User should enter user name, user groups for which the user should belong (the group must be preconfigured first), SSH access permision as well as password. When editing user settings, only *User Group* and *SSH Access* permission can be changed. To change user password, *Change Password* button should be pressed as seen in Figure34 to lead user to a screen seen in Figure 36.

A user needs to be assigned to **root** group for admin rights and have root access.

It should be noted that assigning user to a *root* group only gives complete authority over web interface. Permissions for a command-line interface should be given by a *root* user via command-line interface.

Following commands may be used in comamnd line interface for user control:

adduser - create a new user or update default new user information

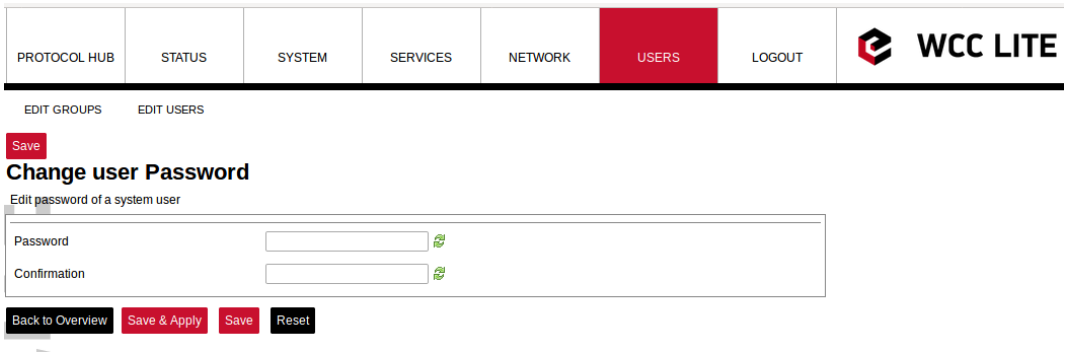


Figure 36: Changing user password

When invoked without the **-D** option, the *adduser* command creates a new user account using the values specified on the command line plus the default values from the system. Depending on command line options, the useradd command will update system files and may also create the new user's home directory and copy initial files.

passwd - change user password

The *passwd* command changes passwords for user accounts. A normal user may only change the password for his/her own account, while the superuser may change the password for any account. *passwd* also changes the account or associated password validity period.

deluser - delete a user account and related files

The *deluser* command modifies the system account files, deleting all entries that refer to the user name LOGIN. The named user must exist.

If a user intends to use newly created user account via both command-line interface and web interface he should create and delete users via web interface and not using *adduser* and *deluser* commands as they don't create uci entries.

For more information about controlling users via command line interface one should refer to Linux documentation.

21.1.3 Authentication via external service

WCC Lite support external authentification via RADIUS service. Remote Authentication Dial-In User Service (RADIUS) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA or Triple A) management for users who connect and use a network service. RADIUS is a client/server protocol that runs in the application layer, and can use either TCP or UDP as transport. Network access servers, the gateways that control access to a network, usually contain a RADIUS client component that communicates with the RADIUS server. RADIUS is often the back-end of choice for 802.1X authentication as well. The RADIUS server is usually a background process running on a UNIX or Microsoft Windows server. In WCC Lite RADIUS Client is implemented since WCC Lite software version v1.2.4. The user sends a request to a WCC Lite to gain access to get access using access credentials posted in an HTTP/HTTPS WCCLite web login form.

This request includes access credentials, typically in the form of username and password. Additionally, the request may contain other information which the Device knows about the user, such as its network address or information regarding the user's physical point of attachment to the device. The RADIUS server checks that the information is correct using authentication schemes

such as PAP, CHAP or EAP. The user's proof of identification is verified, along with, optionally, other information related to the request, such as the user's network address, account status, and specific network service access privileges. Historically, RADIUS servers checked the user's information against a locally stored flat file database. Modern RADIUS servers can do this, or can refer to external sources—commonly SQL, Kerberos, LDAP, or Active Directory servers—to verify the user's credentials. The RADIUS server then returns one of two responses to the WCC Lite:

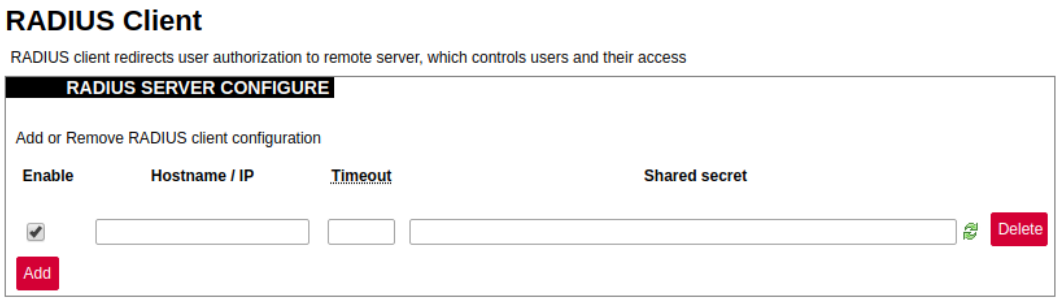
1. **Access Reject** - The user is unconditionally denied access to all requested resources. Reasons may include failure to provide proof of identification or an unknown or inactive user account.
2. **Access Accept** - The user is granted access. Once the user is authenticated, the RADIUS server will periodically check if the user is authorized to use the service requested. A given user may be allowed to get admin rights or user rights depending on permissions set on RADIUS Server. Again, this information may be stored locally on the RADIUS server, or may be looked up in an external source such as LDAP or Active Directory.

To use this mechanism a RADIUS server must be configured. The parameter Radius Authentication must be Enabled on WCC Lite.

As of firmware version 1.2.13, the RADIUS service is disabled by default. The service can be enabled at System->Startup.

If the RADIUS authentication is enabled, WCC Lite uses the RADIUS server IP address and the RADIUS shared secret key for communication with External RADIUS Server. After entering the login credentials and login attempt, WCC Lite sends these credentials to the RADIUS server for authentication. If the RADIUS server is available, it compares the login credentials:

- If the comparison is successful, the RADIUS server returns the specific user role and Access-Accept;
- If the login credentials are invalid, Radius Server returns Access-Reject and the logon fails.
- If the RADIUS server is not available and fallback option is disabled login into WCC Lite will be imposible. If RADIUS server is not available and timeout occurs, login will be attempted via local login credentials.



Enabled: Enables or disables this server.

Hostname/IP: Hostname or IP address of RADIUS server.

Timeout: Timeout in seconds to wait for server response.

Shared secret: Key shared between RADIUS server and RADIUS client.

Add: Adds auxiliary (backup) server

21.1.4 Audit Log

WCC Lite OS with version >1.2.0 has integrated Audit logging for important events such as:

- Login/logout.
- Wrong password attempts to login into system.
- Device boot event, when system was started.
- Device reboot/halt event.
- Configuration changes.
- Firmware changes.
- Date and time changes in system (excluding automatic system time updates over NTP or IEC 60870-5-10x protocol).

Enabling external system log server setup in System properties -> Logging is recomended. System stores logs in RAM memory by default due to limited flash storage. Rebooting or powering off the device will result in loss of log history.

21.1.5 Secure your device's access

There are some possibilities to grant access to the device (or to any PC/Server):

1. ask for nothing: anybody who can establish a connection gets access
2. ask for username and password on an unsecured connection (e.g. telnet)
3. ask for username and password on an encrypted connection (e.g. SSH) (e.g. by following firstlogin)
4. ask for username and merely a signature instead of a password (e.g. SSH with signature.authentication)

If you ask for username/password, an attacker has to guess the combination. If you use an unencrypted connection, he could eavesdrop on you and obtain them.

If you use an encrypted connection, any eavesdropper would have to decrypt the packets first. This is always possible. How long it takes to decrypt the content, depends on the algorithm and key length you used.

Also, as long as an attacker has network access to the console, he can always run a brute-force attack to find out username and password. He does not have to do that himself: he can let his computer(s) do the guessing. To render this option improbable or even impossible you can:

1. not offer access from the Internet at all, or restrict it to certain IP addresses or IP address ranges
   (a) by letting the SSH server dropbear and the web-Server lighttpd not listen on the external/WAN port
   (b) by blocking incoming connections to those ports (TCP 22, 80 and 443 by default) in your firewall

2. make it more difficult to guess:
   (a) don't use the username root
   (b) don't use a weak password with 8 or less characters
   (c) don't let the SSH server dropbear listen on the default port (22)
3. use the combination of
   (a) username different than root
   (b) tell dropbear to listen on a random port (should be >1024): **System > Administration > Dropbear Instance > Port**
   (c) public key authentication. Your public keys can be specified in **Administation > System > SSH-keys**. An older guide to DropBear SSH public key authentication has detailed information on generating SSH keypairs which include the public key(s) you should upload to your configuration.

21.2 Groups rights

If user is logged on via external server, its authentification level is acquired. As no direct mapping to existing users is used, authentification levels are a way to grant proper permissions for external users. WCC Lite uses a CISCO-like authentification system, meaning that there are fifteen different permission set level settings, of which the first 14 can be configured to enable or disable View and Edit permissions.

21.2.1 SSH Access

SSH Access of WCC Lite is made by Dropbear software package. To extend the basic functionality, Pluggable Authentification Module (PAM) for RADIUS is used. This enables user to add his own authentification modules as long as they are properly configured.

Fifteen levels of authorization are mapped for SSH access, meaning that user should be able to access SSH with credentials used to log into web interface. However, one should note that permissions in command line interface are not configurable via web interface. This means that first fourteen levels are restricted to basic permissions made my creating group by default. Highest level user has all the permissions *root* user has.

If a user intends to change permissions for user groups, it should be done via command line interfaces. It is only advised for advanced users.

21.2.2 Web interface permissions

Fifteen levels of authorization permission are mapped for web interface access, meaning that user should be able to access web interface with credentials used to log into command line interface. User assigned to a highest authorization level group is able to access every possible screen therefore this groups cannot be edited.

Figure 37 shows a screen containing already existing groups in a device. Pressing *Add New Groups...* guides user to an *Edit group* screen, *Edit* and *Delete* buttons respectively Edit and Delete configuration of a given user group.

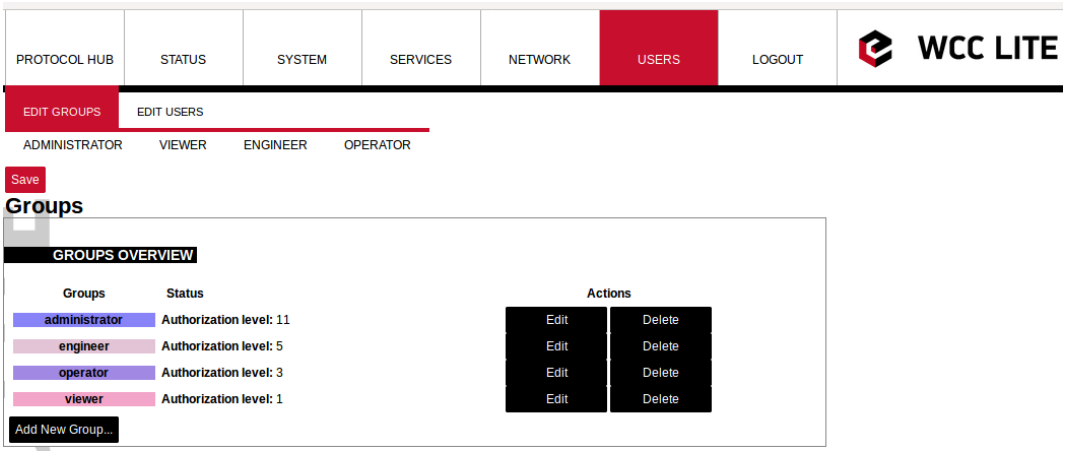


Figure 37: Screen showing existing user groups

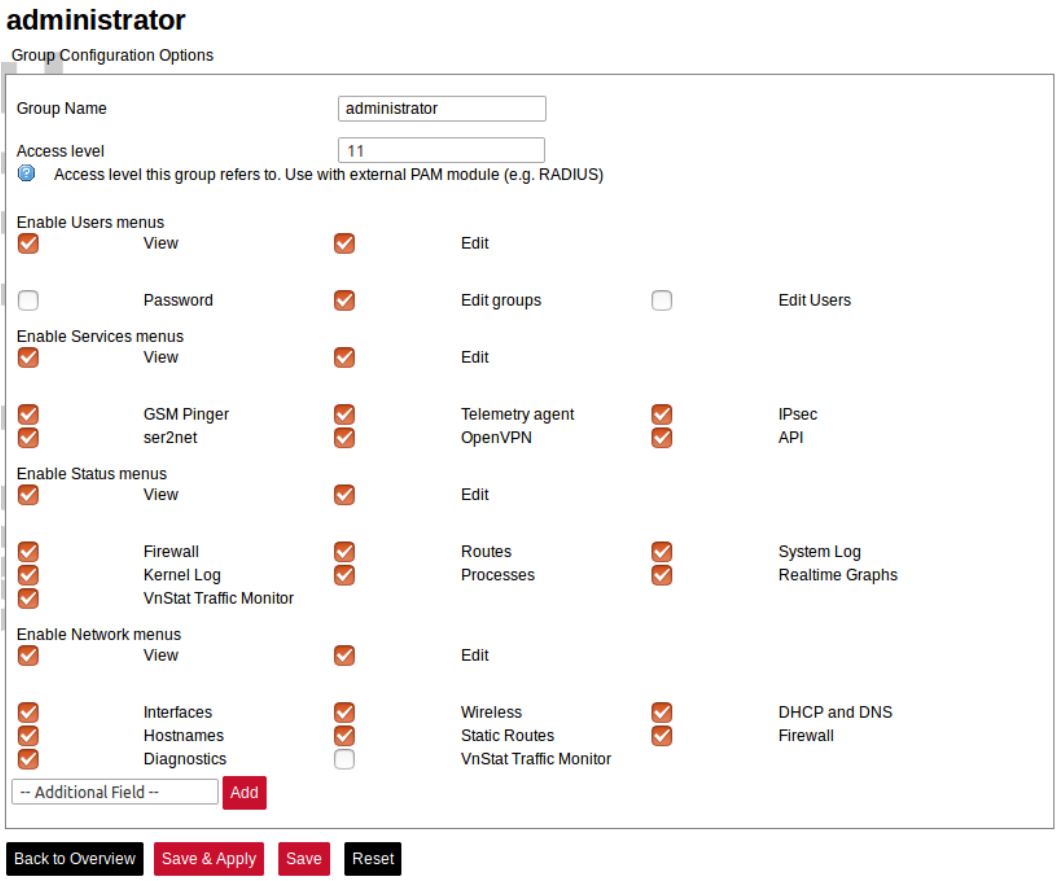


Figure 38: Screen for user group editing

Edit group screen for an individual group can be seen in Figure 38. Group name doesn't have any specific purpose for RADIUS, but it enables naming groups with words most meaningful for a given context. Access level values can only be integers between 1 and 14, other values will result in an error messages; only unconfigured levels are shown in a dropdown list when configuring. Other fields are dedicated for an individual menu configuration. To add more first level menus user should select from a dropdown list at the bottom named *–Additional Field–* and press *Add*.

Permissions for web interface are split into to parts: *View* and *Edit*.

View permissions can be assigned to second level menus meaning that only allowed subtabs are shown for a user. Selecting *View* checkbox show more parameters containing all the subtabs (submenus). If a user can access a given screen, it means all of the actions in that screen are available to be executed. Therefore, if a user with a lot of restrictions shouldn't, for example, import Excel configuration to WCC Lite, a tab containing this action (*Protocol Hub->Configuration*) should be disabled in his group's configuration.

Edit permissions can be assigned to first level menus meaning that if this permission is given, every configuration in the first level menu can be saved and applied succesfully.

21.3 Conformance to IEC 62351 standard

IEC 62351 is a standard developed by WG15 of IEC TC57. This is developed for handling the security of TC 57 series of protocols including IEC 60870-5 series, IEC 60870-6 series, IEC 61850 series, IEC 61970 series and IEC 61968 series. The different security objectives include authentication of data transfer through digital signatures, ensuring only authenticated access, prevention of eavesdropping, prevention of playback and spoofing, and intrusion detection.

Conformance to IEC 62351 standard of WCC Lite devices is described in a table below.

Table 40: Conformance to IEC 62351 standard

Standard	Description	Topic	Implemented	Version
IEC 62351-3	Security for any profiles including TCP/IP	TLS Encryption	Yes	>=1.3
		Node Authentication by means of X.509 certificates	Yes	>=1.3
		Message Authentication	Yes	>=1.3
IEC 62351-4	Security for any profiles including MMS	Authentication for MMS	No	
		TLS (RFC 2246)is inserted between RFC 1006 & RFC 793 to provide transport layer security	No	
IEC 62351-5	Security for any profiles including IEC 60870-5	TLS for TCP/IP profiles and encryption for serial profiles	No	
IEC 62351-6	Security for IEC 61850 profiles	VLAN use is made as mandatory for GOOSE	No	
		RFC 2030 to be used for SNTP	No	
IEC 62351-7	Security through network and system management	Defines Management Information Base (MIBs) that are specific for the power industry, to handle network and system management through SNMP based methods	No	

Table 40: Conformance to IEC 62351 standard

Standard	Description	Topic	Implemented	Version
IEC 62351-8	Role-based access control	Covers the access control of users and automated agents to data objects in power systems by means of role-based access control (RBAC)	Yes	>=1.2.6
IEC 62351-9	Key Management	Describes the correct and safe usage of safety-critical parameters, e.g. passwords, encryption keys.	No	
		Covers the whole life cycle of cryptographic information (enrolment, creation, distribution, installation, usage, storage and removal)	No	
		Methods for algorithms using asymmetric cryptography	No	
		A secure distribution mechanism based on GDOI and the IKEv2 protocol is presented for the usage of symmetric keys, e.g. session keys.	No	
IEC 62351-10	Security Architecture	Explanation of security architectures for the entire IT infrastructure	No	
		Identifying critical points of the communication architecture, e.g. substation control center, substation automation	No	
		Appropriate mechanisms security requirements, e.g. data encryption, user authentication	No	
		Applicability of well-proven standards from the IT domain, e.g. VPN tunnel, secure FTP, HTTPS	No	
IEC 62351-11	Security for XML Files	Embedding of the original XML content into an XML container	No	
		Date of issue and access control for XML data	No	
		X.509 signature for authenticity of XML data	No	
		Optional data encryption	No	

22 Changelog

Changelog section represents changes made between different versions of this document. Please be advised that these versions do not represent firmware version of a device. Changes in functionality between versions are described in their respective paragraphs.

1.4.0 (2020-04-02)

- Added sections for MQTT, PLC, Modbus;
- Added information about Excel configuration import;
- Moved all gsm parameters and pinger info to Network->GSM;
- Added Status->GSM Status window;
- Added EC25 modem description;
- Added Status->Realtime graphs->GSM description;
- Added WCC Lite internal signal description

1.3.5 (2020-01-31)

- Explained IEC 60870-5-103, GSM and API in detail;
- Updated WCC Lite technical information.

1.3.3 (2020-01-20)

- IEC62056-21 over IP implemented;
- Changes in DLMS tag configuration, scaler automatically applied to regular and extended registers;
- Comments about pinger functionality;
- Extensive description about mathematical expressions in WCC Lite without PLC.

1.3.1 (2019-10-15)

- Updated group permissions, user system description, group configuration description

1.3.0 (2019-09-19)

- Put information that can be grouped into tables;
- Additional comment on what to do if firmware image upload fails;
- Explained LEDs and cron in more detail;
- Description of DLMS, DNP3; notes to set up TLS for TCP/IP connections;

- Explained OpenVPN setup in finer detail;
- Added changelog representing changes from March 2018;
- Added conformance table to IEC 62351

1.2.8 (2019-07-11)

- Removed a part of document not representing a true functionality - serial-to-network interface; functionality in later versions;
- Deleted load balancing part as it's no longer part of the firmware;
- Updated formatting for all parts of document;
- Fixed mistake - VPN and network configurations are saved by default;
- Updated system-log screen in the web, board-information photo;
- Started writing firmware version which this document is compatible with;
- Added SNMP section;
- Added device installation information;
- Description on telemetry-agent, ser2net, IEC62056-21;
- Update *Status* tab;
- Updated DHCP, system and RADIUS photos;

1.2.7 (2019-03-20)

- Update photos of login, LED setup web header;

1.2.6 (2019-03-18)

- Changed company name from Aedilis to Elseta;
- Updated contact information;

1.2.5 (2018-11-29)

- Full specification of mathematical and logical operations;
- RADIUS user rights' specification;
- Images for RADIUS configuration through web interface;
- Sequence of events (SOE) described in more detail;
- GSM controller web interface description added;

23 Information about the equipment manufacturer

Office address:
L. Zamenhofo g. 3
LT-06332 Vilnius
Lithuania
Tel.: +370 5 2032302
Email: support@elseta.com
In the web: elseta.com

Work hours: I-V 8:00 - 17:00