# ELSETA

## WCC Lite

## User manual




Elseta

2022/10/04

Doc version: 1.6.3

HW version: 1.4

FW version: 1.6.3

# COPYRIGHTS AND TRADEMARKS

Elseta is a trademark of UAB Elseta that identifies products manufactured by UAB Elseta. All of the products copyrights belong to UAB Elseta. These documents and product properties cannot be changed without the knowledge and written consent from UAB Elseta. This document may be modified by company UAB Elseta without additional notice.

# 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.

# 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 ..... 8
2 Hardware and software requirements ..... 8
3 Technical information ..... 9
4 WCC Lite status indication and control ..... 11
4.1 Status indication ..... 11
4.2 Reset button ..... 11
5 Installing the WCC Lite ..... 12
5.1 Mounting ..... 12
5.2 Power ..... 12
5.3 SIM card slot ..... 13
5.4 Dual-SIM card slot ..... 14
6 Interfaces ..... 15
6.1 Serial port interfaces ..... 15
6.2 Relay output ..... 17
6.3 Digital Input ..... 17
6.4 GSM ..... 18
6.5 Wi-Fi ..... 19
7 Tags ..... 20
Single point ..... 20
Double point ..... 20
8 Internal web page ..... 20
8.1 Initial Setup ..... 21
8.1.1 Static IP address setup on Windows ..... 21
8.1.2 Connecting to an internal web page ..... 23
8.2 Site layout ..... 24
8.3 Protocol Hub ..... 25
8.4 Status ..... 28
Overview ..... 28
GSM status ..... 37
VNSTAT Traffic monitor ..... 38
8.5 System ..... 40
8.6 Services ..... 50
8.7 Network ..... 59
8.8 Logout ..... 78
9 API ..... 79
10 SNMP ..... 79
11 SMS sender ..... 80
General ..... 80
Configuring SMS sender ..... 80
12 DNP 3.0 ..... 83
12.1 Introduction ..... 83
12.2 DNP 3.0 Master ..... 84
Configuring datapoints ..... 84

Debugging the DNP3 Master service ... 88
12.3 DNP 3.0 Slave ... 89
Debugging the DNP3 Slave service ... 92
13 Modbus ... 94
13.1 Introduction ... 94
13.2 Modbus Master ... 95
13.3 Modbus Slave ... 99
14 IEC 60870-5-10X ... 102
14.1 Introduction ... 102
14.2 IEC 60870-5-101 Master ... 103
Configuring datapoints (master) ... 104
Debugging a IEC 60870-5-101 Master application ... 107
14.3 IEC 60870-5-101 Slave ... 107
Configuring datapoints (slave) ... 107
Debugging an IEC 60870-5-101 slave application ... 110
14.4 IEC 60870-5-103 Master ... 110
IEC 60870-5-103 Master Configuring datapoints ... 110
Debugging an IEC 60870-5-103 Master application ... 113
14.5 IEC 60870-5-104 Master ... 115
Configuring IEC 104 Master datapoints ... 115
Debugging an IEC 60870-5-104 Master application ... 117
14.6 IEC 60870-5-104 Slave ... 118
Configuring IEC 104 Slave datapoints ... 118
Debugging an IEC 60870-5-104 Slave application ... 120
15 IEC 61850 ... 122
15.1 Introduction ... 122
15.2 IEC 61850 Server ... 123
Commands ... 123
Configuring datapoints ... 123
Converting and uploading data model ... 125
Debugging an IEC 61850 server application ... 126
15.3 IEC 61850 Client ... 127
Acquiring data via report control blocks ... 127
Number Types ... 128
Controlling remote equipment via commands ... 129
Configuring datapoints ... 130
Debugging a IEC 61850 Client application ... 132
16 Specific protocols ... 133
16.1 At command ... 134
16.2 Aurora ... 136
16.3 COMLYNX ... 138
Overview ... 138
16.4 Delta ... 140
16.5 GINLONG ... 141
Overview ... 141

16.6 Kaco ..... 143
16.6 KOSTAL ..... 146
Configuration ..... 146
16.7 POWERONE ..... 148
Overview ..... 148
Configuration ..... 148
16.8 SMA NET ..... 149
Overview ..... 149
Configuration ..... 150
16.9 SOLPLUS ..... 151
Configuration ..... 151
16.10 VBUS ..... 154
Configuration ..... 154
16.11 VESTAS ..... 156
Configuration ..... 156
16.12 Windlog ..... 158
16.13 M-Bus ..... 160
17 Metering protocols ..... 162
17.1 DLMS/COSEM ..... 162
Introduction ..... 162
DLMS Master ..... 163
17.2 IEC 62056-21 ..... 166
Introduction ..... 166
Overview ..... 167
Configuration ..... 167
17.3 Elgama ..... 169
18 Excel Configuration ..... 172
18.1 Devices sheet ..... 173
18.2 Optional parameters for signals ..... 174
18.3 Mathematical functions ..... 179
18.4 Uploading configuration ..... 183
18.5 Virtual device ..... 184
19 WCC Lite internal signals ..... 184
20 Programmable logic controller ..... 188
20.1 IEC 61499 ..... 189
20.2 4Diac framework ..... 190
20.3 Example project ..... 190
20.4 Configuring data endpoints ..... 196
20.5 Debugging an IEC 61499 application ..... 198
20.6 Generating and uploading FORTE logic file ..... 199
20.7 Distributed control application ..... 201
21 MQTT ..... 203
Introduction ..... 203
Using WCC Lite as MQTT Client ..... 204
MQTT data format ..... 206

Debugging a MQTT protocol ..... 206
22 Data Export ..... 208
General ..... 208
Overview ..... 208
Using WCC Lite for data export ..... 208
Debugging data export service ..... 210
Host URL format rules ..... 210
Format of exported data ..... 211
23 Certificates ..... 213
24 Cyber security ..... 214
User rights ..... 214
Groups rights ..... 217
Conformance to IEC 62351 standard ..... 219

# 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 Lite up 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 builtin 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 Binary input (hardware version >=1.4)<br>1x Relay output |
| Additional storage SD card (2GB by default) | SD card (2GB by default) |
| Ethernet 10/100 BaseT RJ45 connector up to 2 independent ports | 10/100 BaseT RJ45 connector up to 2 independent ports |
| Serial ports | 1x RS485<br>1x RS485 / RS232 (switchable) |
| Time synchronization | NTP client + server, IEC 608705101, IEC 608705104 |
| GSM | 2G(GPRS, EDGE) / 4G(LTE)<br>2G(GPRS, EDGE) / 3G(UMTS, HSDPA, HSUPA)<br>2G(GPRS, EDGE) / 3G(UMTS, HSDPA, HSUPA) / 4G(LTE)<br>Single OR Dual SIM card modem |
| **Power requirements** | |
| Power supply | 12 - 24 VDC |
| Power consumption | < 6W |
| 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<br>• Isolated LAN interface, but omitted to provide TCP / UDP ports or VPN mergers<br>• Routed LAN internet connection masking data for GSM interfaces<br>• Secure LAN data transfer via VPN<br>• Secure LAN data transmission through VPN access to the Internet• Single OR Dual SIM card modem |
| Database | • File based database<br>• Data buffering in case of network outage |
| Data security | • All data between WCC Lite and Cloud based SCADA exchange over secure encrypted VPN tunnel<br>• Firewall to prevent intrusion and DoS attacks<br>• 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<br>CloudIndustries.eu or 3rd party SCADAs and see devicebased alarms such as communication errors, etc. |
| Supported protocols | • Modbus master / slave (RTU / ASCII / TCP)<br>• MBus master (Serial / TCP)<br>• IEC 608705 101 master / slave<br>• IEC 608705 103 master<br>• IEC 608705 104 master / slave<br>• IEC 6205631 master<br>• IEC 6205621 (since v1.2.13)<br>• DNP3 master / slave<br>• SMA Net<br>• DLMS (since v1.3.0)<br>• Resol VBus<br>• IEC 61499 (since v1.4.0)<br>• IEC 61850 master / slave (since v1.5.0) |
| Supported devices | • Other Elseta products<br>• Aurora PV inverters<br>• Delta inverters<br>• Kaco PV inverters<br>• SMA PV inverters<br>• Ginlong PV inverters<br>• Solplus PV inverters<br>• Kostal devices<br>• Windlog data logger<br>• Vestas Wind turbines<br>• Elgama elektronika electricity meters |
| Network features | • IPsec<br>• OpenVPN<br>• xl2tp<br>• Firewall<br>• Routing<br>• RADIUS• SNMP<br>• ser2net<br>• API<br>• NTP synchronization |
| Extra features | • Software update<br>• Remote configuration via CloudIndustries.eu administration<br>• Device fault notifications<br>• 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<br>blinking (blue) Blinking in heartbeat pattern.<br>Blinking is more intense when CPU load is high.<br>On (red) Fault (if configured)<br>Off System is not powered |
| RELAY | On (green) Relay is activated (COM connected to NO)<br>Off Relay is deactivated (COM connected to NC) |
| RX1/TX 1 | On (yellow) PORT1 data TX<br>On (green) PORT1 data RX<br>Off No activity on PORT1 |
| RX1/TX 2 | On (yellow) PORT2 data TX<br>On (green) PORT2 data RX<br>Off No activity on PORT2 |
| ETH 0 | On (green) ETH0 LINK/Activity<br>Off ETH0 not linked/inactive |
| ETH 1 | On (green) ETH1 LINK/Activity<br>Off ETH1 not linked/inactive |
| WLAN | On (blue) Wi-fi enabled<br>Blinking (blue) Connected<br>Off Wi-fi disabled |
| GSM | Blinking (yellow) Registered to network<br>Off Not registered to network<br>On (green) GSM signal level is over -65dBm<br>On (green) GSM signal level is over -85dBm<br>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 behavior.

| 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-colored clip downwards (found at the bottom side of the DIN rail).
2. Pull back the bottom of the device.
3. Pull the device upwards to dismount it.




Figure 2: WCC Lite DIN rail mounting clip

## 5.2 Power

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

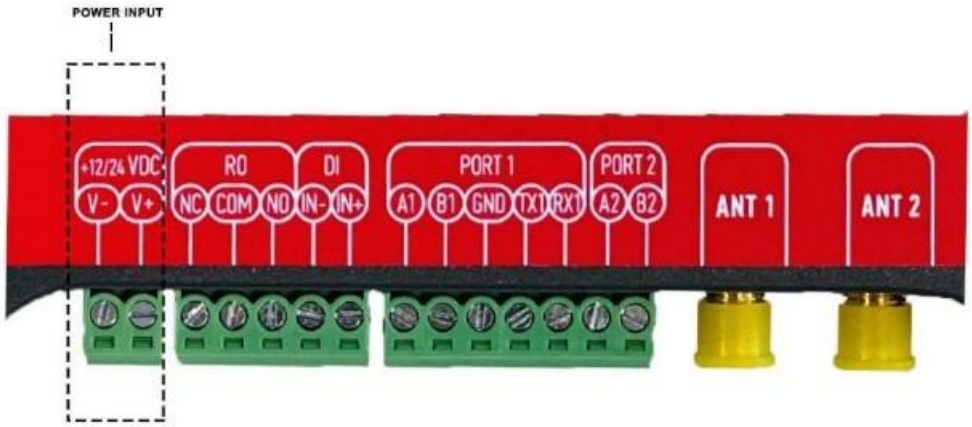



Figure 3: Power input connector

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

# 5.3 SIM card slot

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




Figure 5: WCC Lite Dual-SIM card slot

# 5.4 Dual-SIM card slot

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

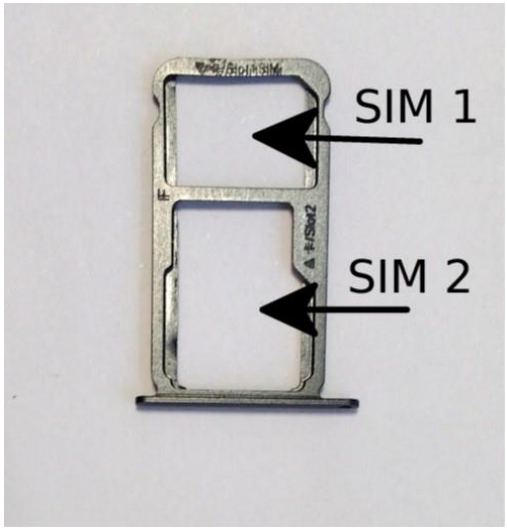



Figure 6: WCC Lite Dual-SIM card holder

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

# 6 Interfaces

WCC lite supports various interfaces to acquire data and control external circuitry. That includes two serial port interfaces, relay output, digital input, and external cellular connection antenna.

# 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.

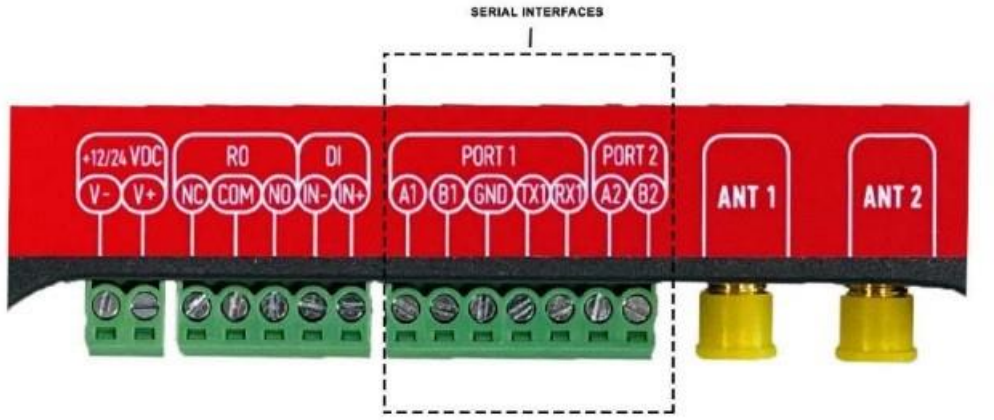



Figure 7: WCC Lite ports

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

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 the user (see Port settings). Baud rates up to 115200 are supported. See the typical RS232 connection diagram in figure 9.

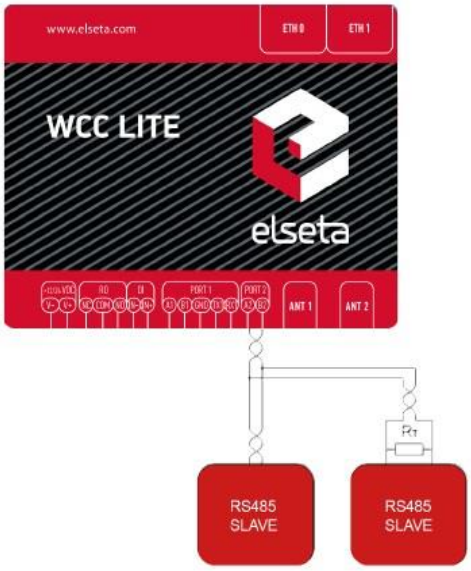



Figure 8: Typical WCC Lite RS-485 connection diagram

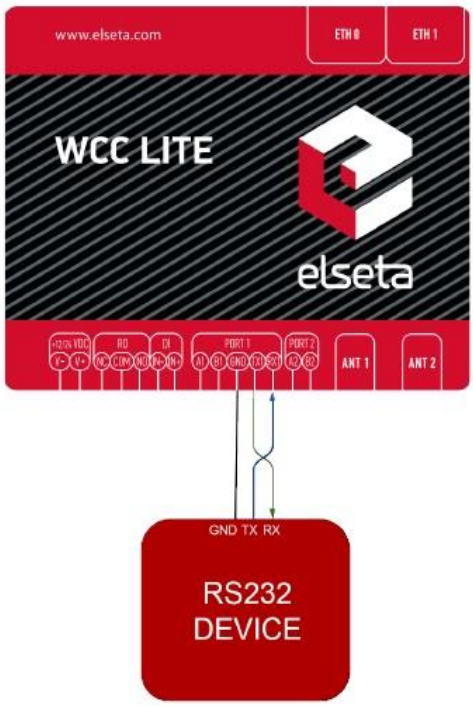



Figure 9: Typical WCC Lite RS-232 connection diagram

# 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.

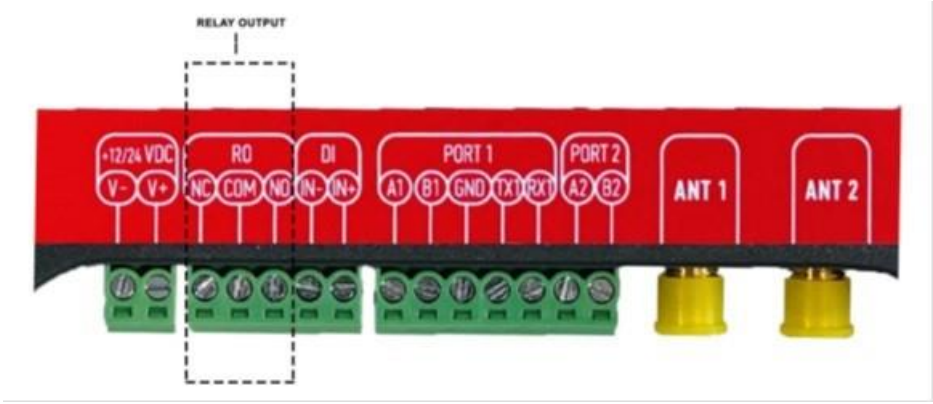



Figure 10: Signal relay connector

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

Relay electrical endurance:

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

# 6.3 Digital Input

With WCC Lite hardware version 1.4 a digital input functionality has been introduced. Software configuration guidelines are discussed later in this document.

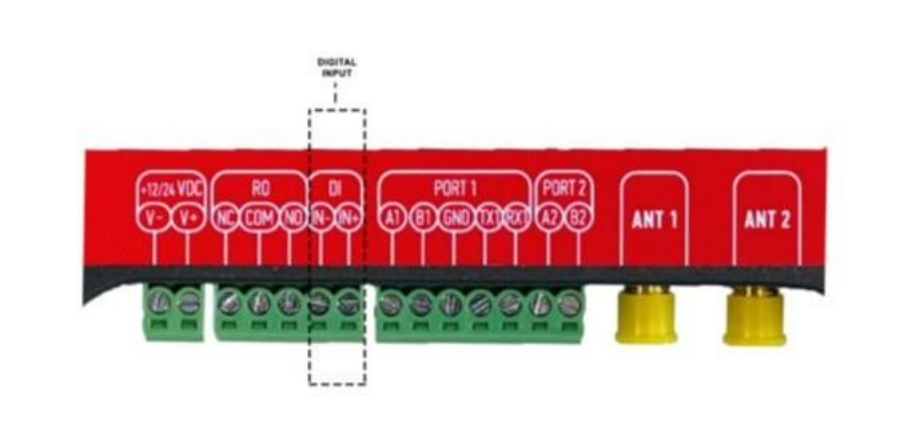



Figure 11: Signal relay connector

# 6.4 GSM

WCC Lite comes with an optional GSM module.
There are a 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 SaraU270.
- 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 a mini PCI-e standard connector and compatible with any other device. Check the label on the package for current modifications.

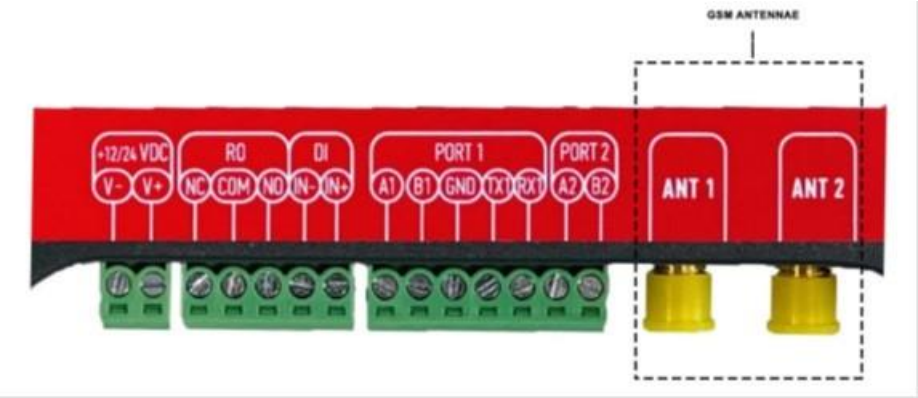



Figure 12: GSM antennae connectors

Connect an antenna to the SMA connector labeled “ANT1“. Select a good antenna placement spot considering the operation environment and network coverage of your mobile provider in the area. To enable MIMO functionality for 4G (LTE) modems a second antenna should be connected to the SMA connector labeled ”ANT2”.

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

Make sure the signal level is over -80dBm to have a stable connection to the network.

# 6.5 Wi-Fi

For hardware versions older than version 1.3, 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.

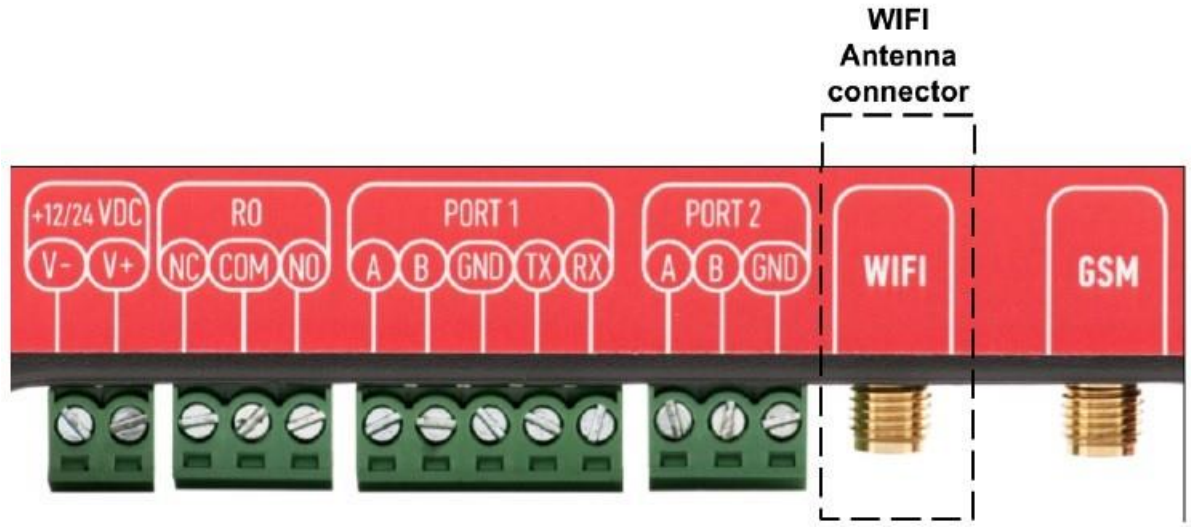



Figure 13: Wi-Fi antenna connector (hardware version older than 1.3)

Newer hardware versions don’t have an option of connecting an external Wi-Fi antenna as MIMO capability for cellular modems has been introduced. In case stronger reach is needed, a user should contact the manufacturer to provide possible solutions.

Make sure the signal level is over -80dBm to have a stable connection to the network.

# 7 Tags

## 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 customizable label.

| Value | Representation |
|---|---|
| 0 | OFF |
| 1 | ON |

## 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

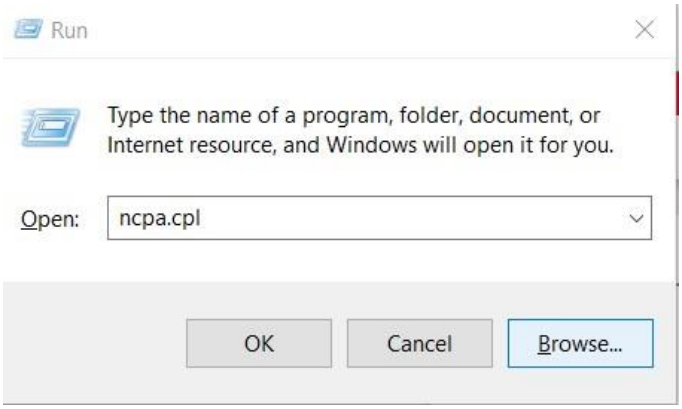
WCC Lite is configured via an internal web browser, so no additional software is required.

# 8.1 Initial Setup

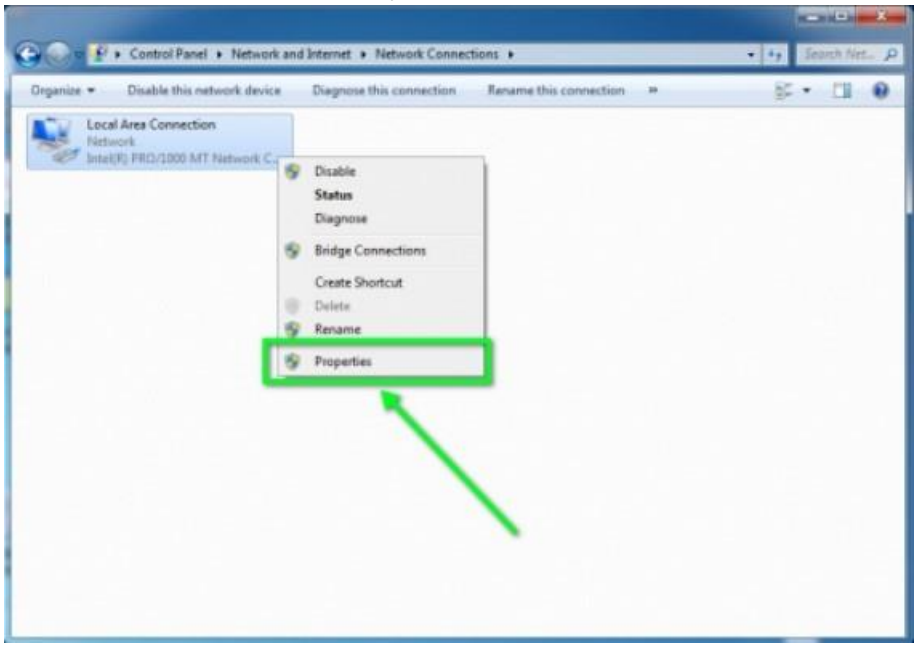
WCC Lite comes with a static network configuration with its IP set to 192.168.1.1. For initial setup set a static IP address on your computer and connect your network card to the WCC Lite with an ethernet cable.

## 8.1.1 Static IP address setup on Windows

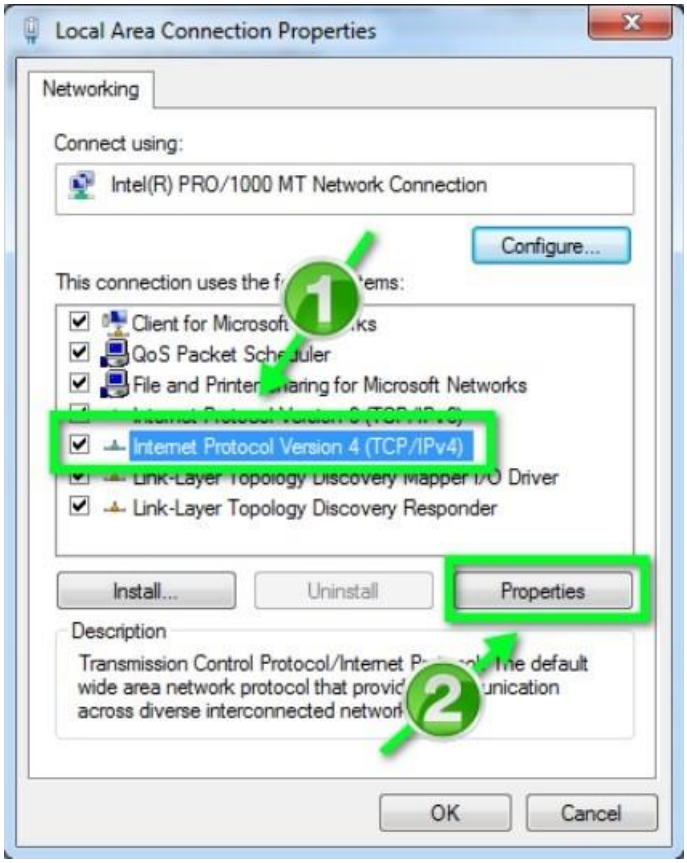
1. Press Win+R on your keyboard. This will open the run window. Enter ncpa.cpl and press OK. This will open the Network Connections window.




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

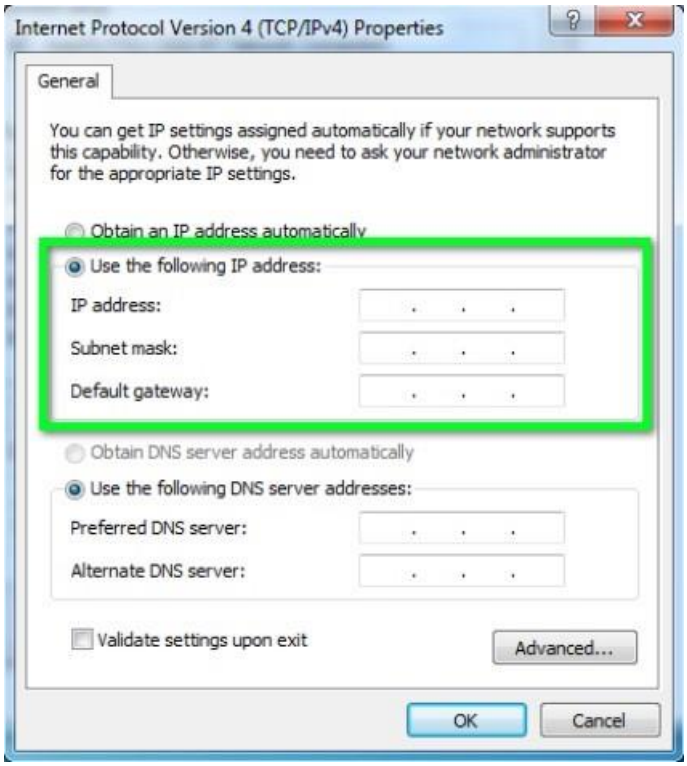



3. In the window that opens, click on the Internet Protocol Version 4 (TCP/IPv4) (you may need to scroll down to find it). Next, click on the Properties button.




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

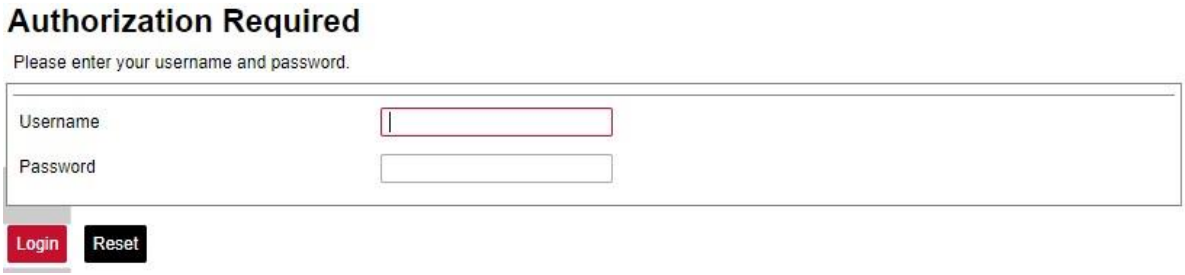
- IP address: 192.168.1.2
- Subnet mask: 255.255.255.0
- Default gateway: (leave empty)



# 8.1.2 Connecting to an internal web page

If your computer IP address is set up and an 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




Login 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



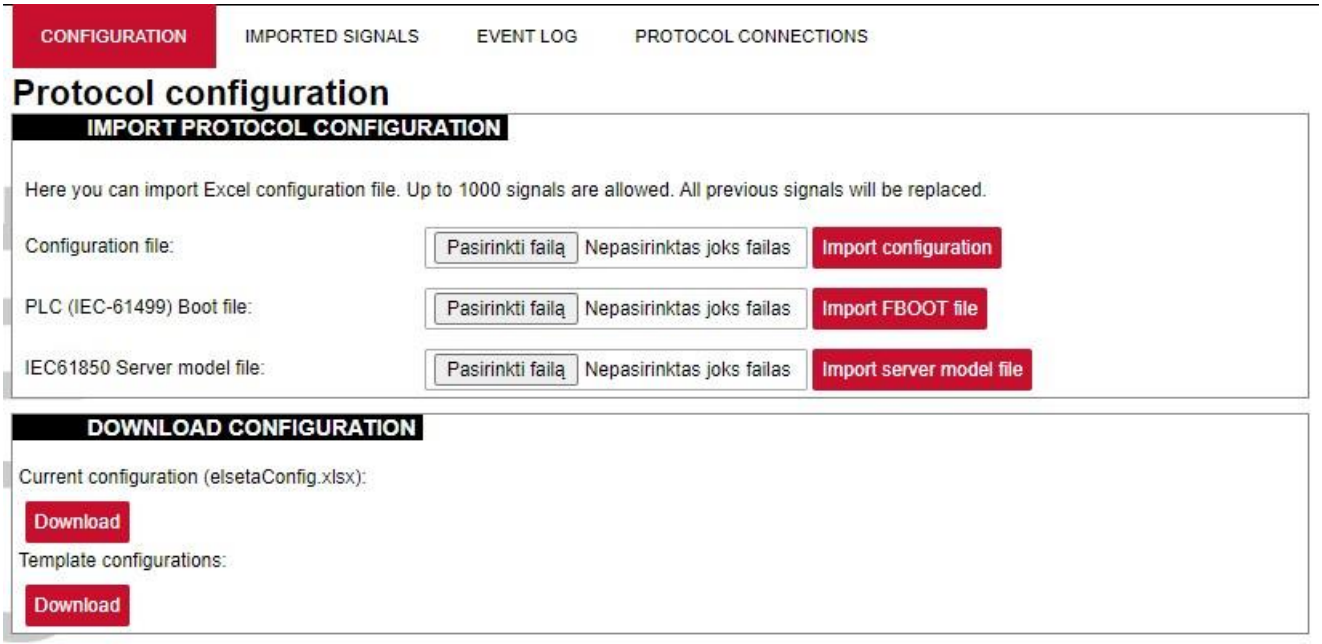

It provides the main navigation through the website. Contains the following sections:

- 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

Protocol HUB section stores configuration for every connected device. You can configure it by importing settings from an Excel file.

## Configuration

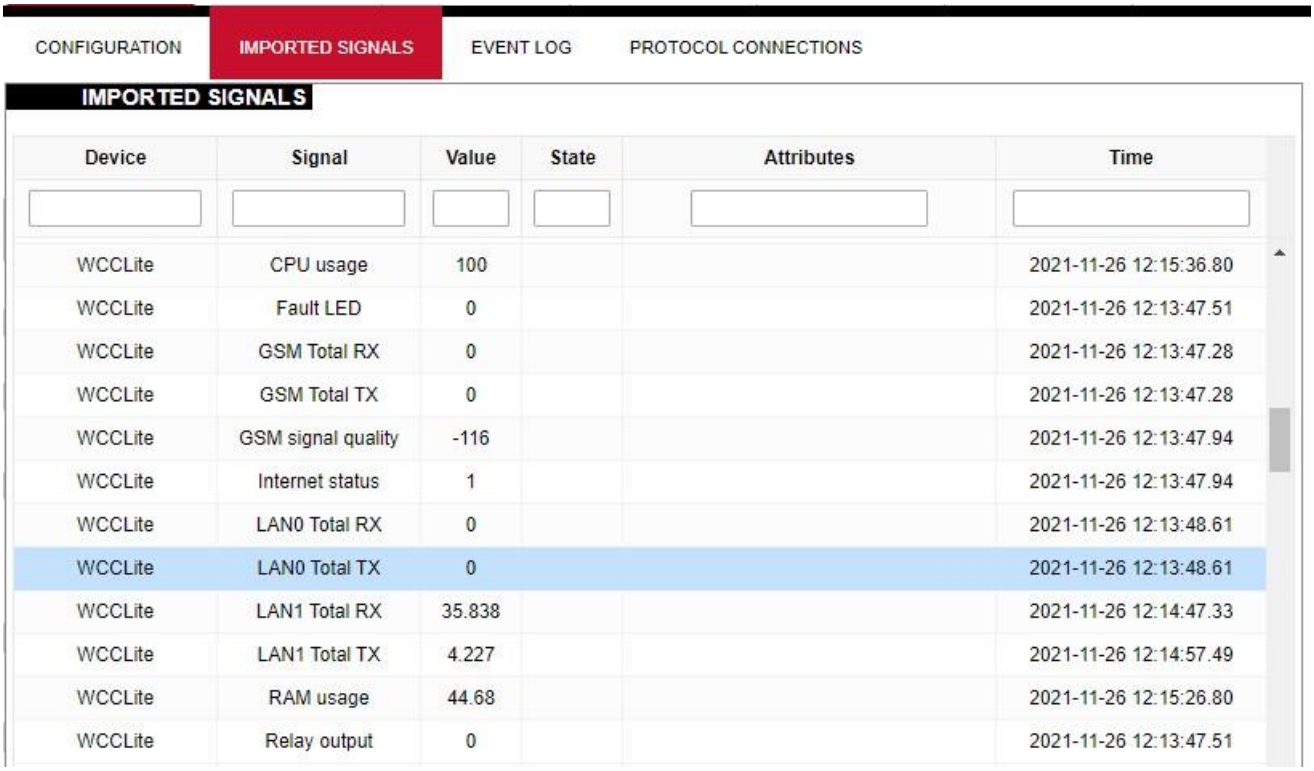



In this tab a user can:

- Import new configuration from Excel file (.xls, .xlsx formats). If any errors in the file are found, the device will not be imported, and importing process will be stopped.
- Import .fboot file for PLC.
- Import IEC61850 Server model file
- Download current configuration Excel file.

Download a template configuration Excel file.

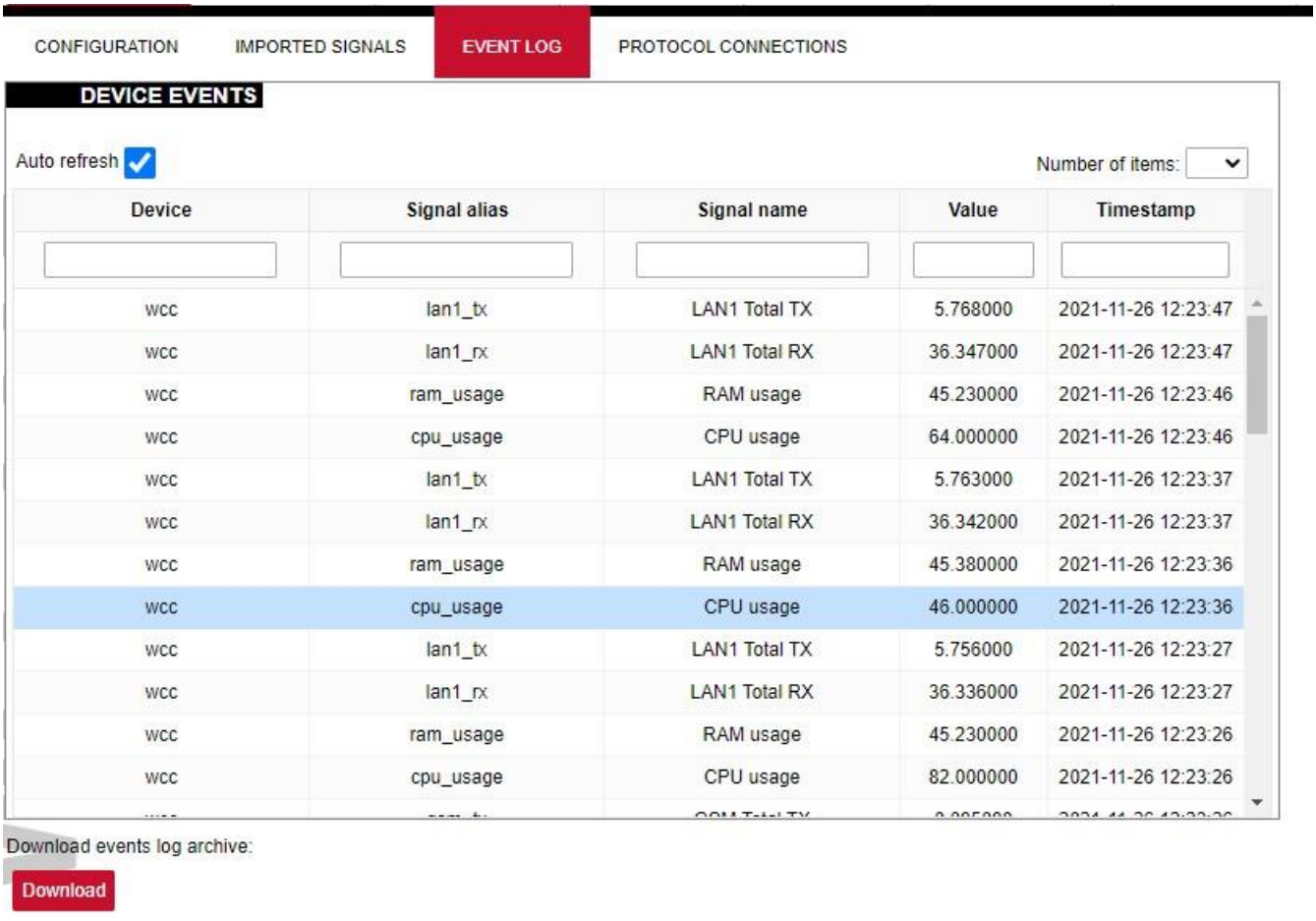
## Imported Signals




| Device | Signal | Value | State | Attributes | Time |
|---|---|---|---|---|---|
| WCCLite | CPU usage | 100 | | | 2021-11-26 12:15:36.80 |
| WCCLite | Fault LED | 0 | | | 2021-11-26 12:13:47.51 |
| WCCLite | GSM Total RX | 0 | | | 2021-11-26 12:13:47.28 |
| WCCLite | GSM Total TX | 0 | | | 2021-11-26 12:13:47.28 |
| WCCLite | GSM signal quality | -116 | | | 2021-11-26 12:13:47.94 |
| WCCLite | Internet status | 1 | | | 2021-11-26 12:13:47.94 |
| WCCLite | LAN0 Total RX | 0 | | | 2021-11-26 12:13:48.61 |
| WCCLite | LAN0 Total TX | 0 | | | 2021-11-26 12:13:48.61 |
| WCCLite | LAN1 Total RX | 35.838 | | | 2021-11-26 12:14:47.33 |
| WCCLite | LAN1 Total TX | 4.227 | | | 2021-11-26 12:14:57.49 |
| WCCLite | RAM usage | 44.68 | | | 2021-11-26 12:15:26.80 |
| WCCLite | Relay output | 0 | | | 2021-11-26 12:13:47.51 |

The imported signals section shows basic information about applied configuration. This section is used for viewing only.

## Event Log




| Device | Signal alias | Signal name | Value | Timestamp |
|---|---|---|---|---|
| wcc | lan1_tx | LAN1 Total TX | 5.768000 | 2021-11-26 12:23:47 |
| wcc | lan1_rx | LAN1 Total RX | 36.347000 | 2021-11-26 12:23:47 |
| wcc | ram_usage | RAM usage | 45.230000 | 2021-11-26 12:23:46 |
| wcc | cpu_usage | CPU usage | 64.000000 | 2021-11-26 12:23:46 |
| wcc | lan1_tx | LAN1 Total TX | 5.763000 | 2021-11-26 12:23:37 |
| wcc | lan1_rx | LAN1 Total RX | 36.342000 | 2021-11-26 12:23:37 |
| wcc | ram_usage | RAM usage | 45.380000 | 2021-11-26 12:23:36 |
| wcc | cpu_usage | CPU usage | 46.000000 | 2021-11-26 12:23:36 |
| wcc | lan1_tx | LAN1 Total TX | 5.756000 | 2021-11-26 12:23:27 |
| wcc | lan1_rx | LAN1 Total RX | 36.336000 | 2021-11-26 12:23:27 |
| wcc | ram_usage | RAM usage | 45.230000 | 2021-11-26 12:23:26 |
| wcc | cpu_usage | CPU usage | 82.000000 | 2021-11-26 12:23:26 |

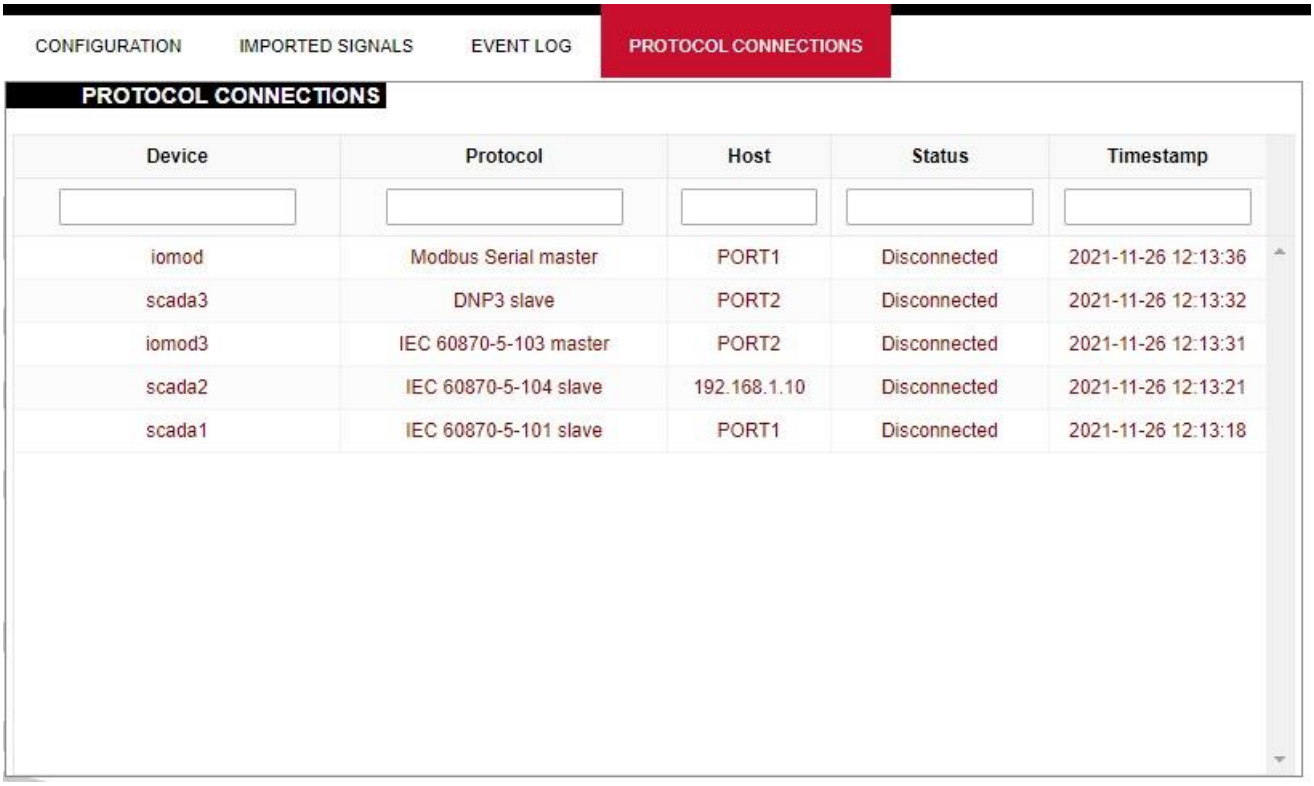
Event Log is the timestamped status data. It allows reviewing of the latest events and changes for devices state changes in chronological order. Newest events are shown at the top of the list. WCC Lite will timestamp the status data with a time resolution of one millisecond.

Initially, all breakers, protection contacts digital status input points in the WCCLite; events captured from IEDs (Intelligent electronic devices) shall be configured as Event Log points. It's possible to assign any digital status input data point in the WCCLite as an SOE point with an Excel template during configuration.

Each time a device changes state, the WCClite will save it with timetag in internal storage. Event Log can also be downloaded by pressing the download button at the bottom of the page.

Events are recorded only for devices that have the log field set to 1.

## Protocol Connections

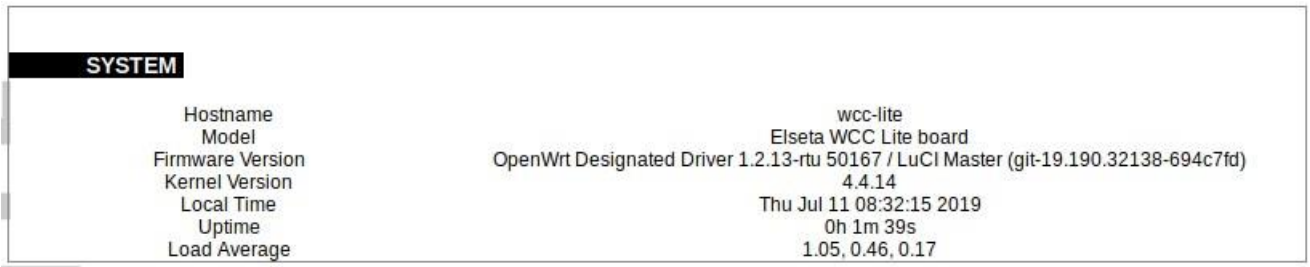



The protocol connections section shows configured devices and their respective ports, statuses

# 8.4 Status

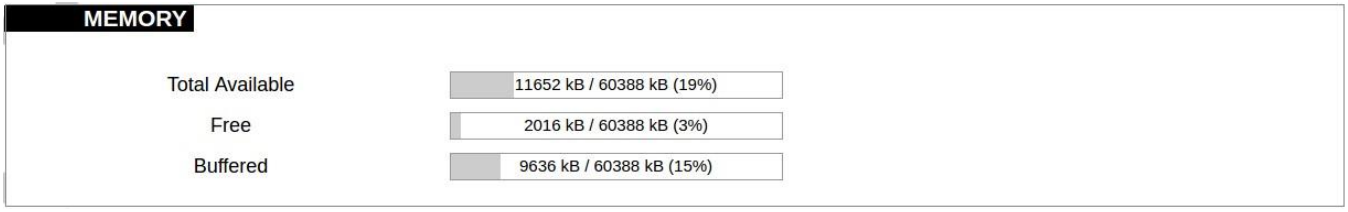
## Overview

### System




System section in the status tab shows basic information about the 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 active connections with the device.

## DHCP leases

**DHCP LEASES**

| Hostname | IPv4-Address | MAC-Address | Leasetime remaining |
|---|---|---|---|
| There are no active leases. | | | |

**DHCPV6 LEASES**

| Host | IPv6-Address | DUID | Leasetime remaining |
|---|---|---|---|
| ? | fd74:8536:7bae::33f/128 | 00046836d59efa382760f3193e5ec5bf4a24 | 11h 58m 53s |

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 the 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

**WIRELESS**

Generic 802.11bgn Wireless Controller (radio0)

0%
**SSID:** WCC Lite
**Mode:** Master
**Channel:** 11 (2.462 GHz)
**Bitrate:** ? Mbit/s
**BSSID:** C6:93:00:0E:C4:33
**Encryption:** None

60%
**SSID:** AP5
**Mode:** Client
**Channel:** 11 (2.462 GHz)
**Bitrate:** 6.5 Mbit/s
**BSSID:** 02:1A:11:FF:87:09
**Encryption:** WPA2 PSK (CCMP)

Wi-Fi 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 channels 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

**ASSOCIATED STATIONS**

| | Network | MAC-Address | Host | Signal / Noise | RX Rate / TX Rate |
|---|---|---|---|---|---|
| wlan0 | Client "AP5" | 02:1A:11:FF:87:09 | 192.168.43.1 | -71 / -95 dBm | 1.0 Mbit/s, 20MHz<br>6.5 Mbit/s, 20MHz, MCS 0 |

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.

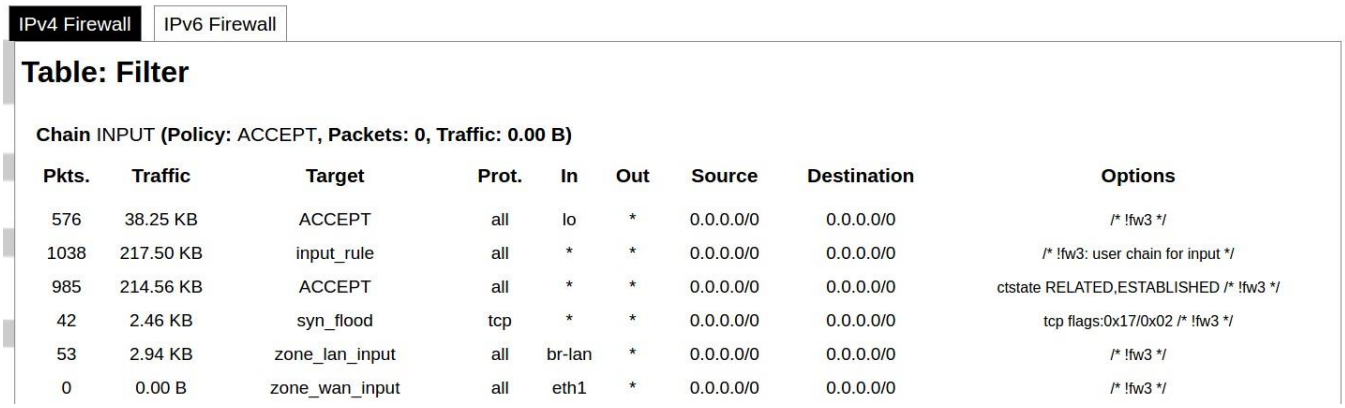
## Board information




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;

# 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

IPv4 Firewall | IPv6 Firewall

**Table: Filter**

**Chain** INPUT **(Policy:** ACCEPT**, Packets: 0, Traffic: 0.00 B)**

| Pkts. | Traffic | Target | Prot. | In | Out | Source | Destination | Options |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.00 B | ACCEPT | all | lo | * | ::/0 | ::/0 | /* !fw3 */ |
| 8041 | 684.54 KB | input_rule | all | * | * | ::/0 | ::/0 | /* !fw3: user chain for input */ |
| 32 | 3.08 KB | ACCEPT | all | * | * | ::/0 | ::/0 | ctstate RELATED,ESTABLISHED /* !fw3 */ |

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.

# 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 Neighbors: The devices on the same network with IPv6 addresses.

# 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.

# 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.

# 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 the 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.

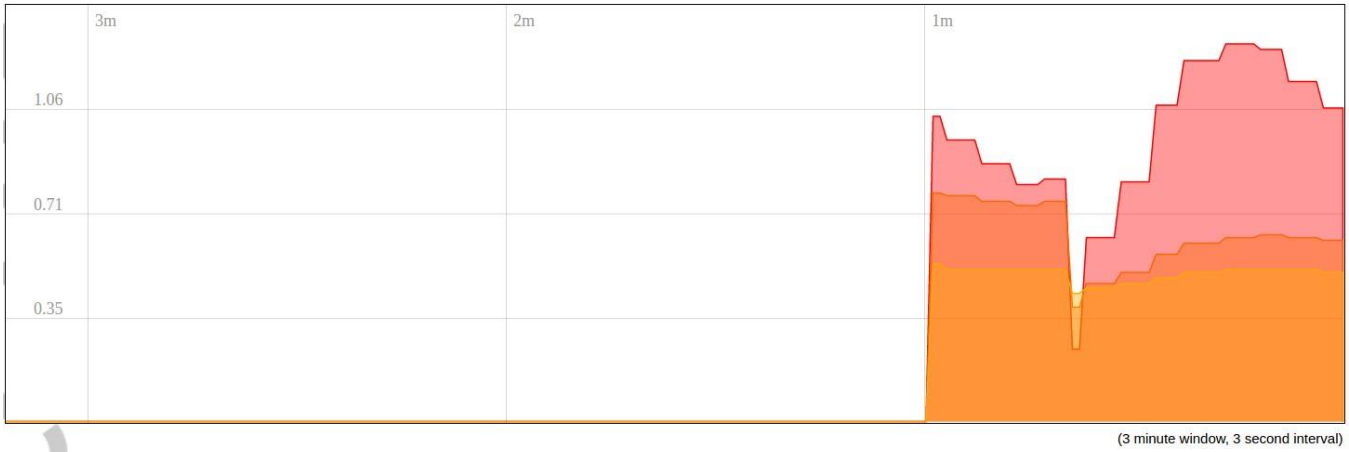
Hang Up: To freeze the process.

Terminate: To end the process cleanly.

Kill: To end the process immediately.

# Realtime graph

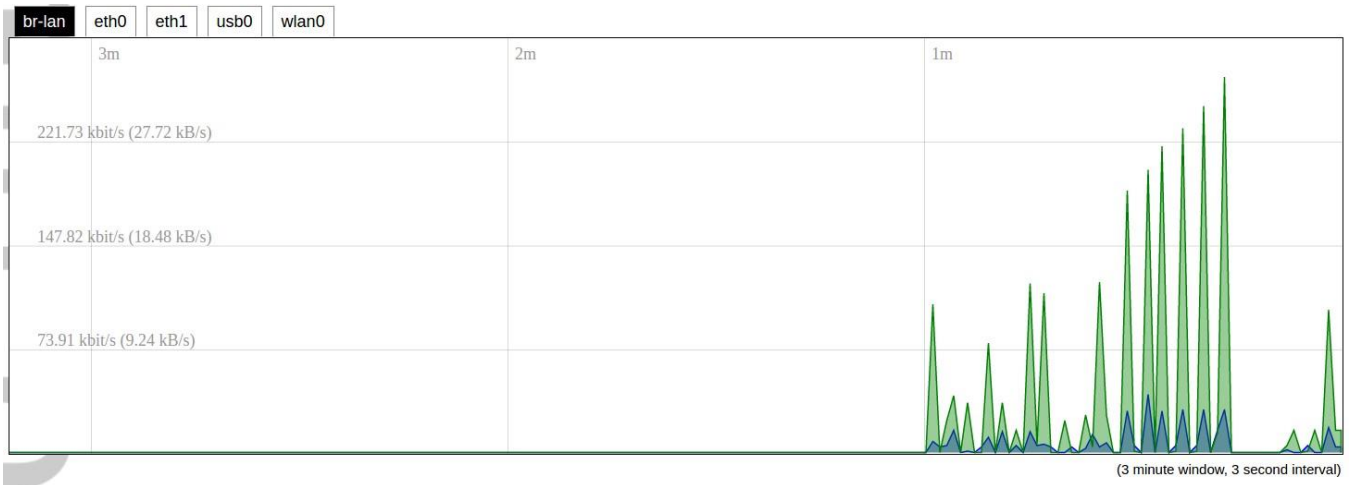
## Realtime Load




| | | | | | |
|---|---|---|---|---|---|
| **1 Minute Load:** | 1.07 | **Average:** | 1.08 | **Peak:** | 1.29 |
| **5 Minute Load:** | 0.62 | **Average:** | 0.62 | **Peak:** | 0.78 |
| **15 Minute Load:** | 0.51 | **Average:** | 0.51 | **Peak:** | 0.54 |

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




| | | | | | |
|---|---|---|---|---|---|
| **Inbound:** | 4.92 kbit/s (0.62 kB/s) | **Average:** | 7.22 kbit/s (0.9 kB/s) | **Peak:** | 41.63 kbit/s (5.2 kB/s) |
| **Outbound:** | 15.89 kbit/s (1.99 kB/s) | **Average:** | 34.7 kbit/s (4.34 kB/s) | **Peak:** | 268.76 kbit/s (33.59 kB/s) |

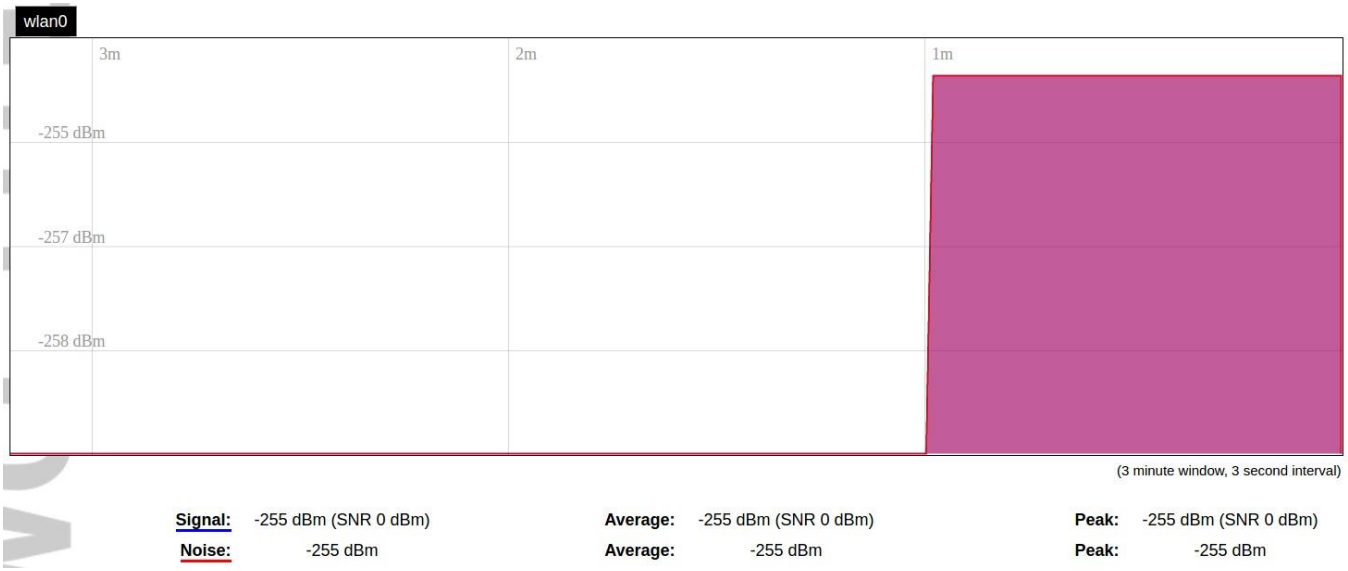
Graphs representing the status of the virtual and physical network interfaces of the device.

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



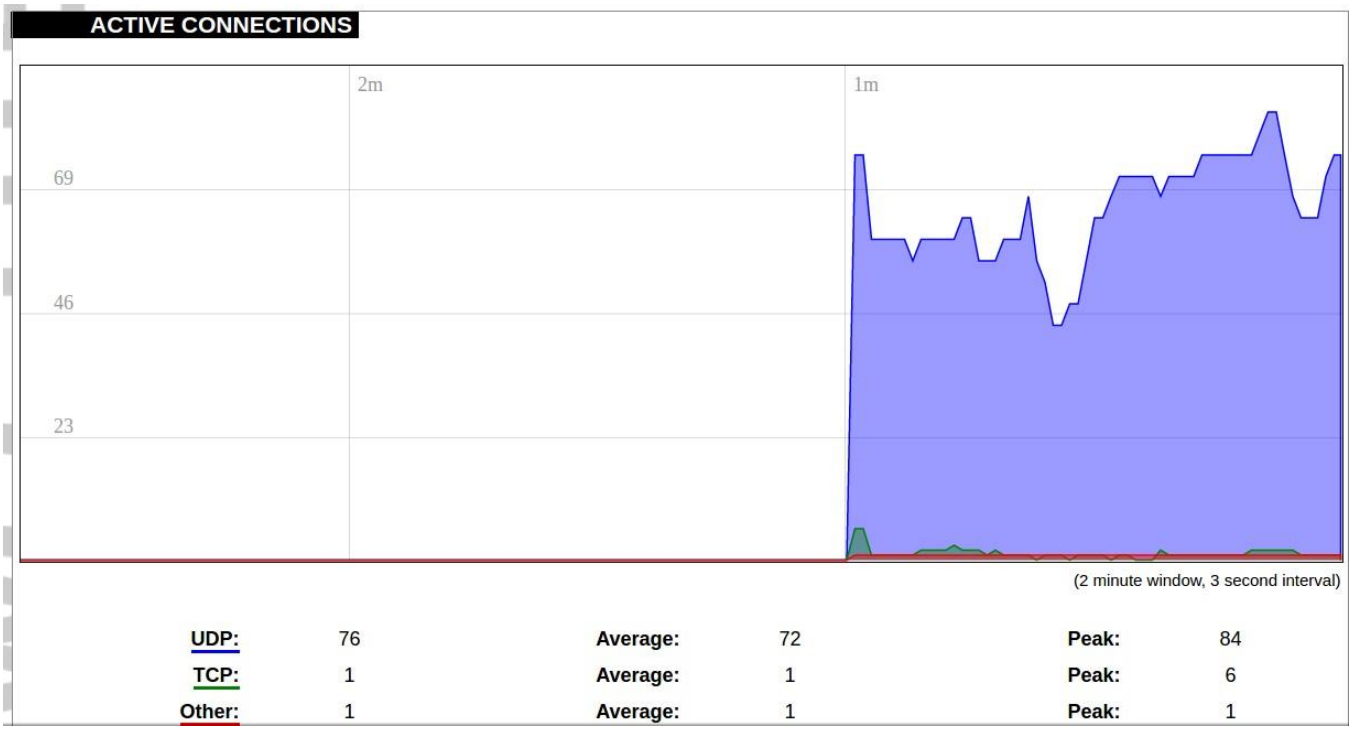

Wi-Fi status graph.

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.

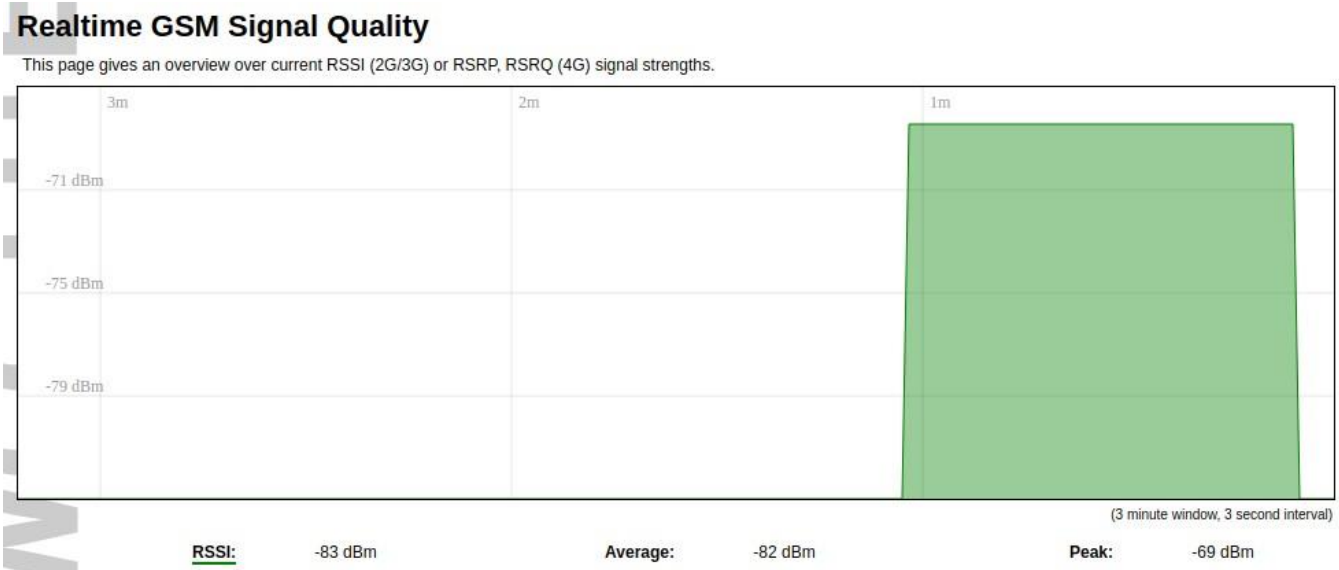
TCP: Transport layer – Transmission Control Protocol.

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.

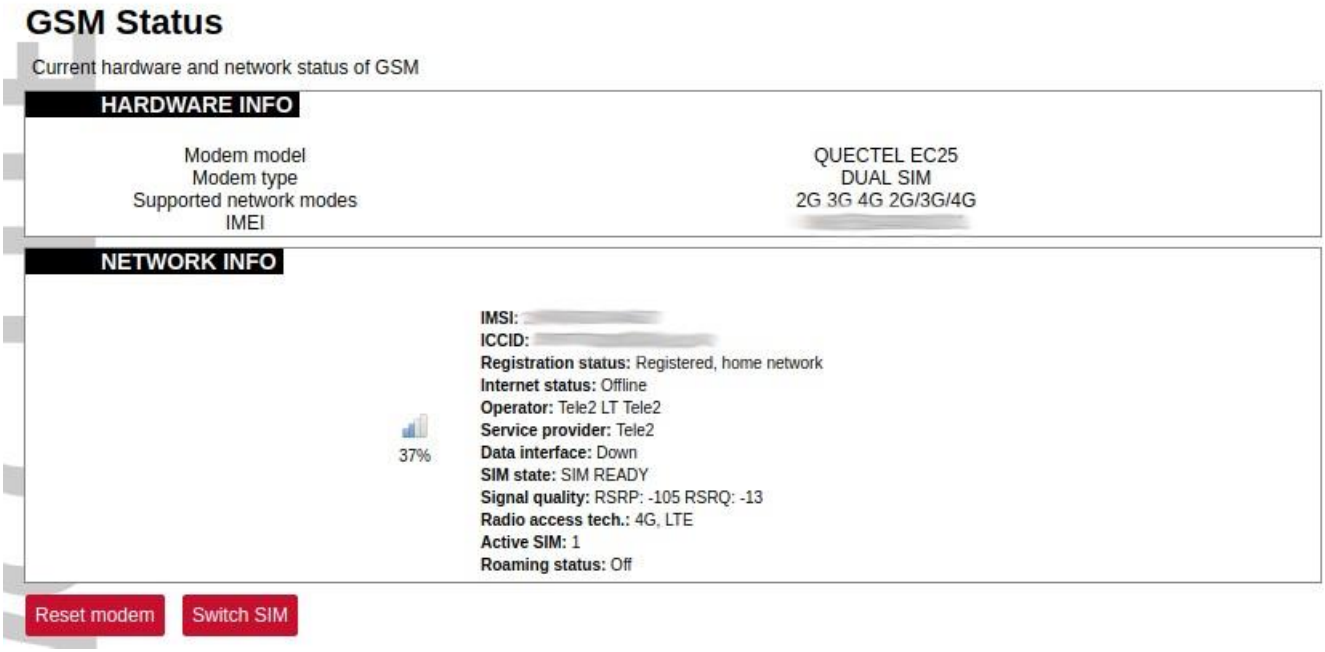
RSSI: Received Signal Strength Indicator in dBm.

RSRP: Received Signal Reference Power in dBm.

RSRQ: Received Signal Reference Quality in dBm.

# GSM status

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




## Hardware info

All static information on the 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: Current status of network connection.

Internet status: Status of connection to the 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 has 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.).

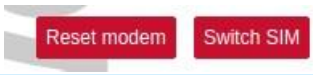
Signal quality: Shows current signal strength value in dBms. The RSSI value is shown, when connected to 2G/3G networks, RSRP-RSRQ values - when connected to 4G network.

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 a 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 a Dual SIM modem and both cards are enabled).



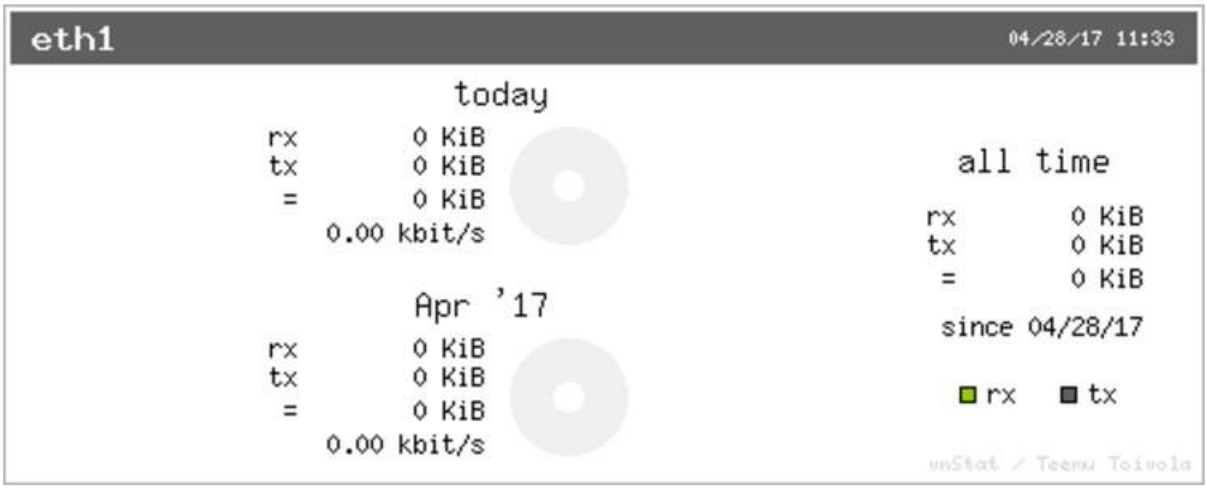

Signal quality is described in different ways for different types of different mobile services: Received Signal Strength Indication (RSSI) in GSM (2G) and UMTS (3G), the Reference Signal Received Quality (RSRQ) in LTE RAT.

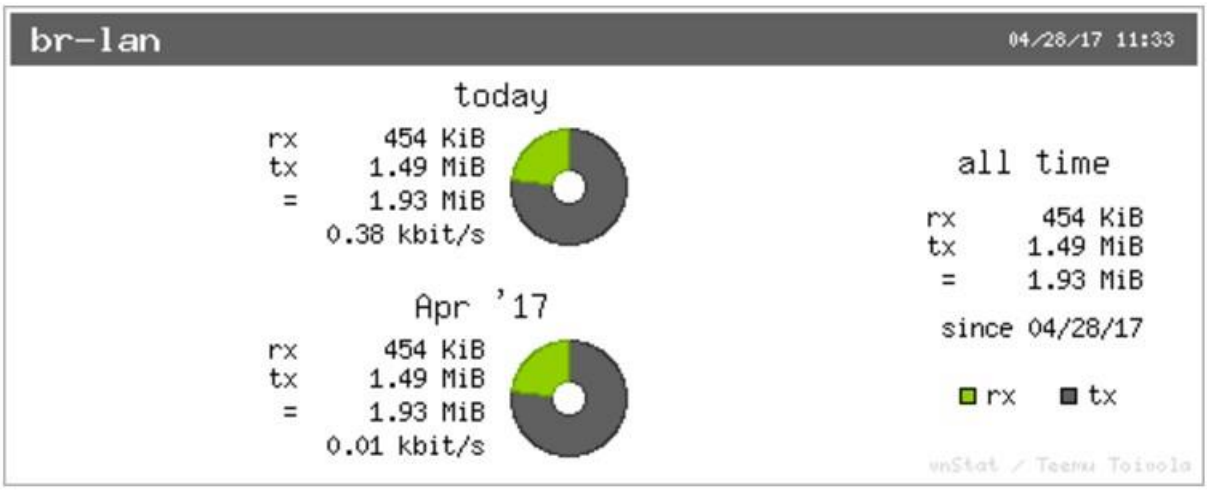
The Reference Signal Received Power (RSRP) is a LTE-specific measure that averages the power received on the subcarriers carrying the reference signal. The RSRP measurement bandwidth is equivalent to a single LTE subcarrier: its value is therefore much lower than the total received power usually referred 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.

# VNSTAT Traffic monitor

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

## Graph



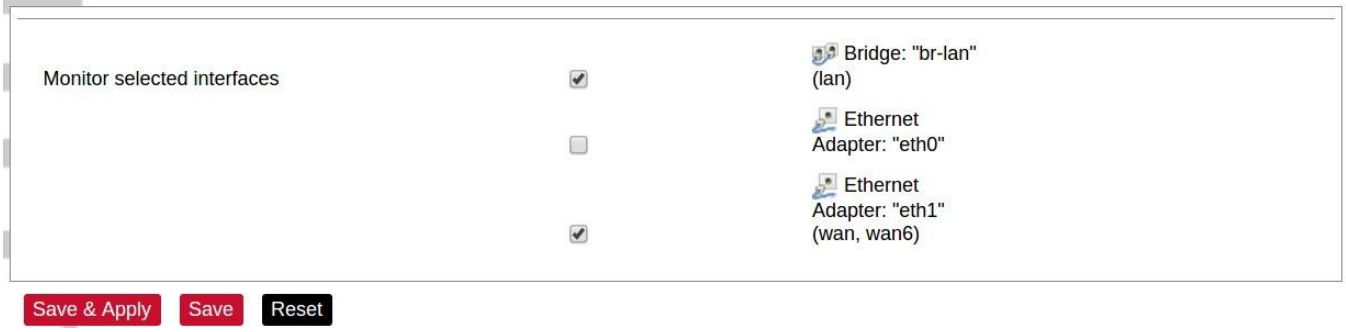




An example graph shows the statistics gathered for two network interfaces. In these graphs: eth1: Network

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 the respective checkbox and save settings by pressing Save & Apply.

# 8.5 System

## System

The system tab includes various properties, configurations, and settings of the system and contains the following pages:

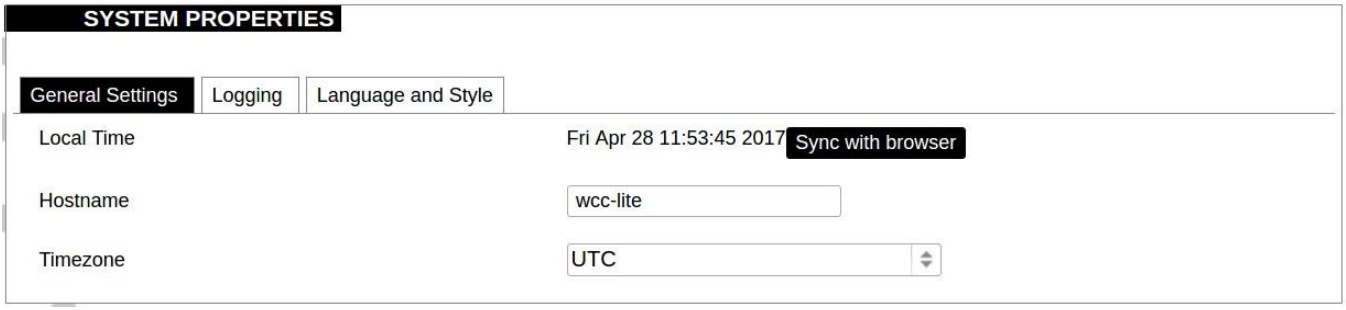



- SYSTEM: properties and settings of the system.
- ADMINISTRATION: settings of the administration for various services.
- SOFTWARE: settings of the packages.
- STARTUP: process management.
- SCHEDULED TASKS: settings of the scheduled tasks.
- MOUNT POINTS: settings for the mount points.
- BOARD: board 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.

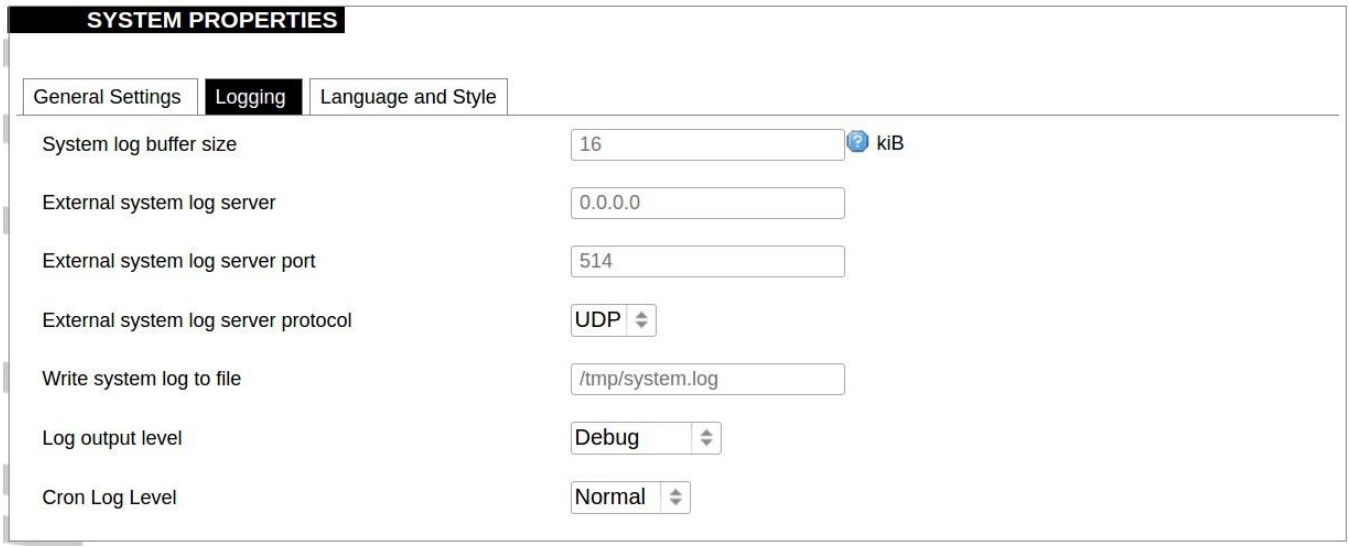
## System

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.
Time zone: 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 the table below.

| 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 the table below.

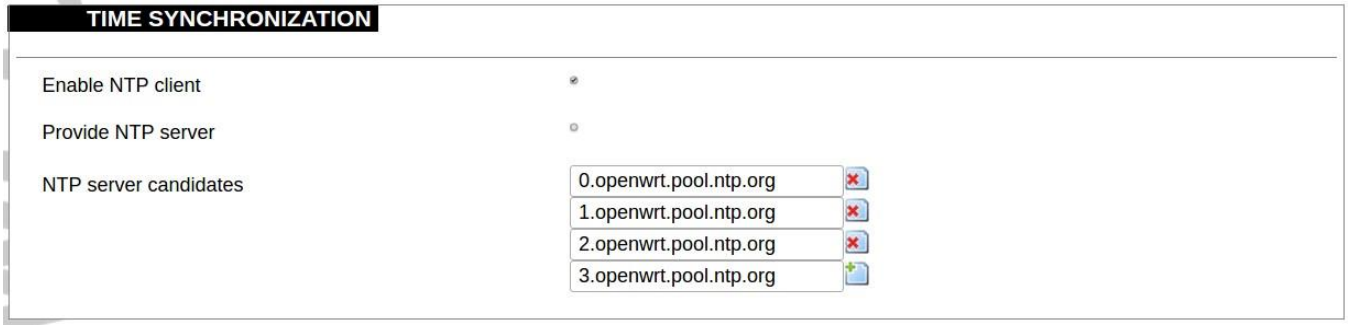
| Cron log level | Description |
|---|---|
| Debug | Debugging messages |
| Normal | General administrative messages |
| Warning | Potentially hazardous conditions |




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.

# Administration

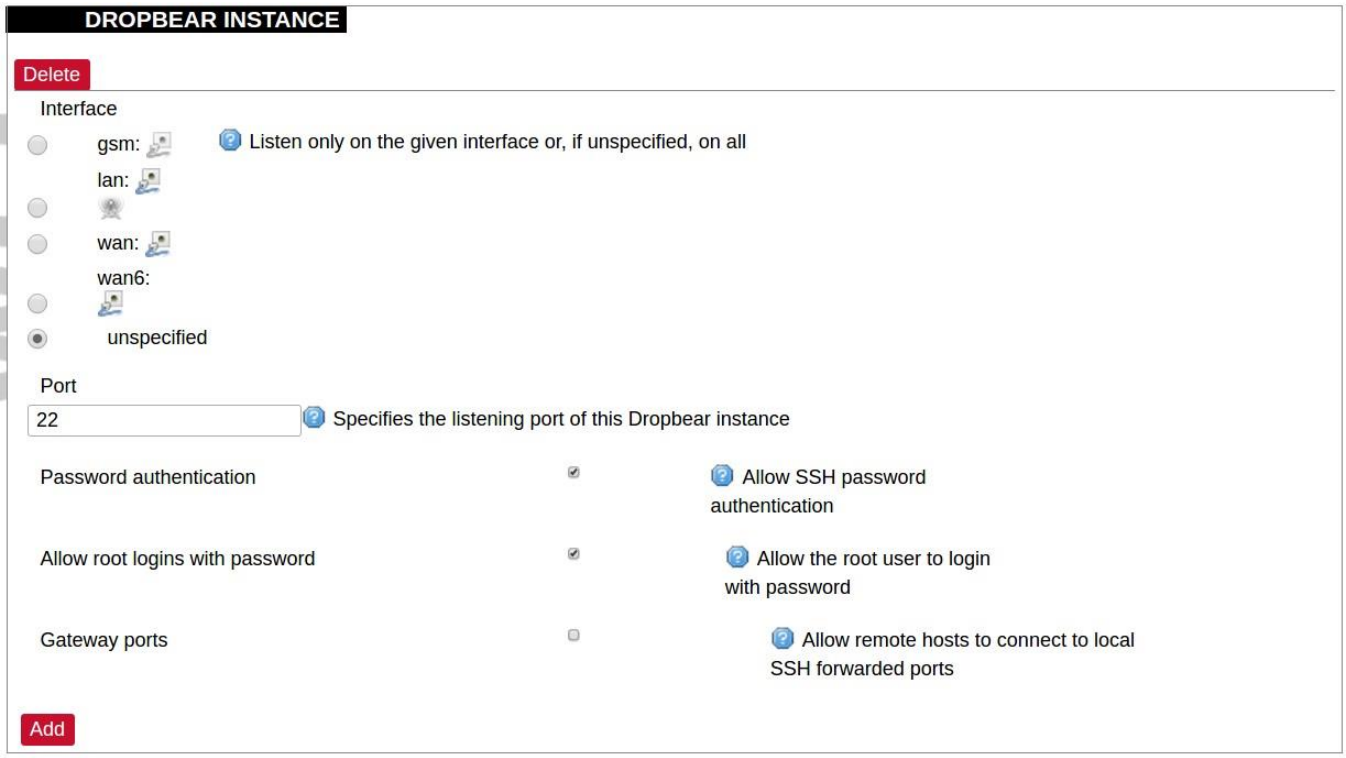



Administrator password can be changed. To change it the combination of digits and letters of the alphabet should be entered and then confirmed in the confirmation field by typing in again.

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 authentication flexibility.



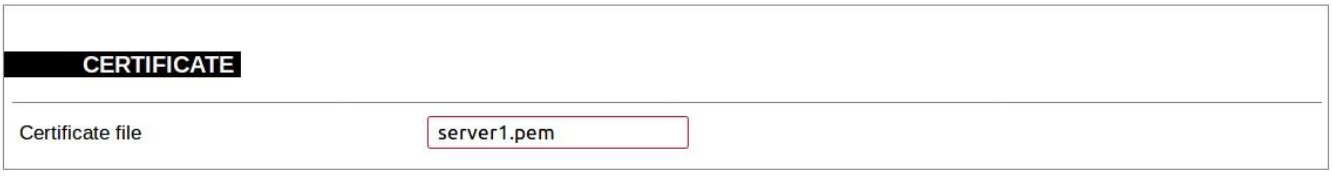

Dropbear options are defined as follows:
Interface: Listen only on the given interface or on all, in unspecified.
Port: Specifies the listening port of this interface.
Password authentication: Allow SSH password authentication.
Allow roots logins with password: Allow the root user to login with the password. 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



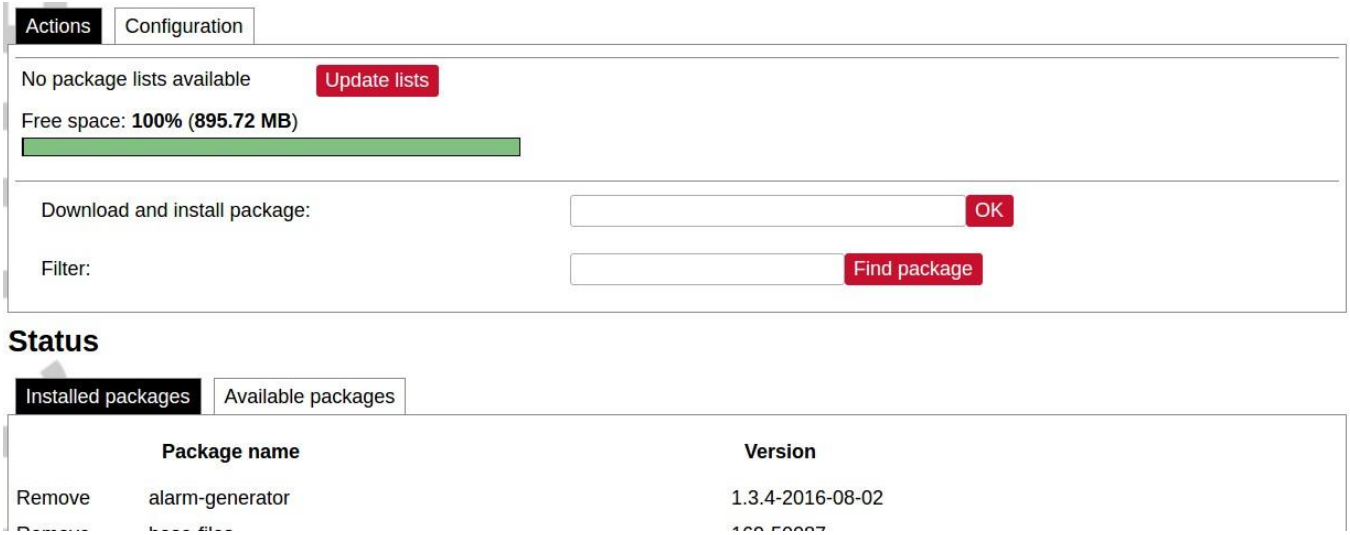

WCC Lite by default is shipped with a default certificate for HTTPS connection. This certificate only enables connecting to device via web interface and might cause warnings from a web browser. To eliminate them, user can use his own certificate to secure access to web interface.

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.

# Software

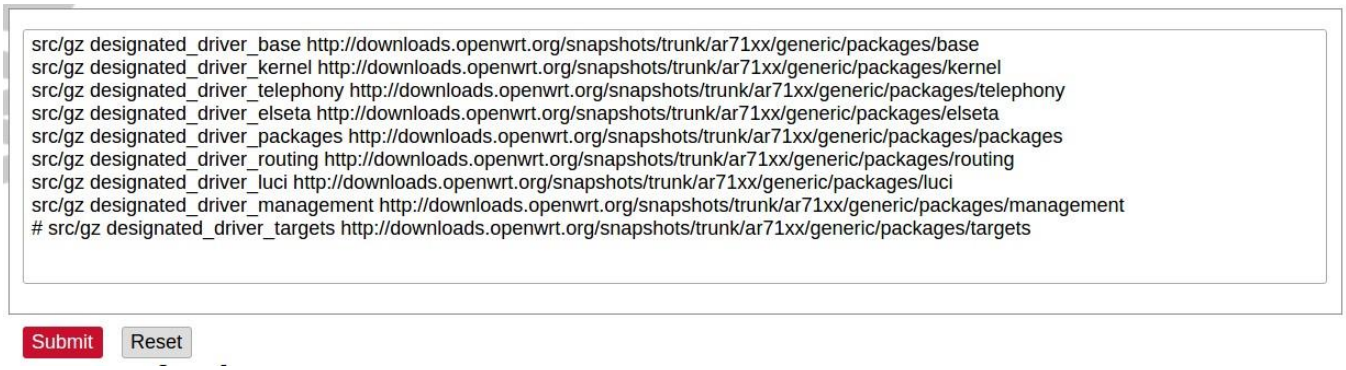
Individual packages can be installed via WCC Lite web interface. They can either be installed using web link or selected from the pre-defined feeds.




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.

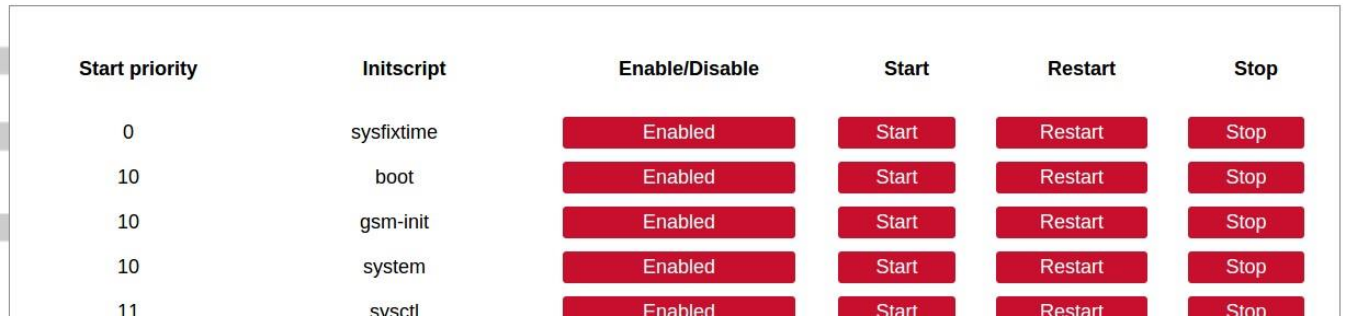



Specific distribution feeds can also be added for special cases if standard ones do not fit the needs.

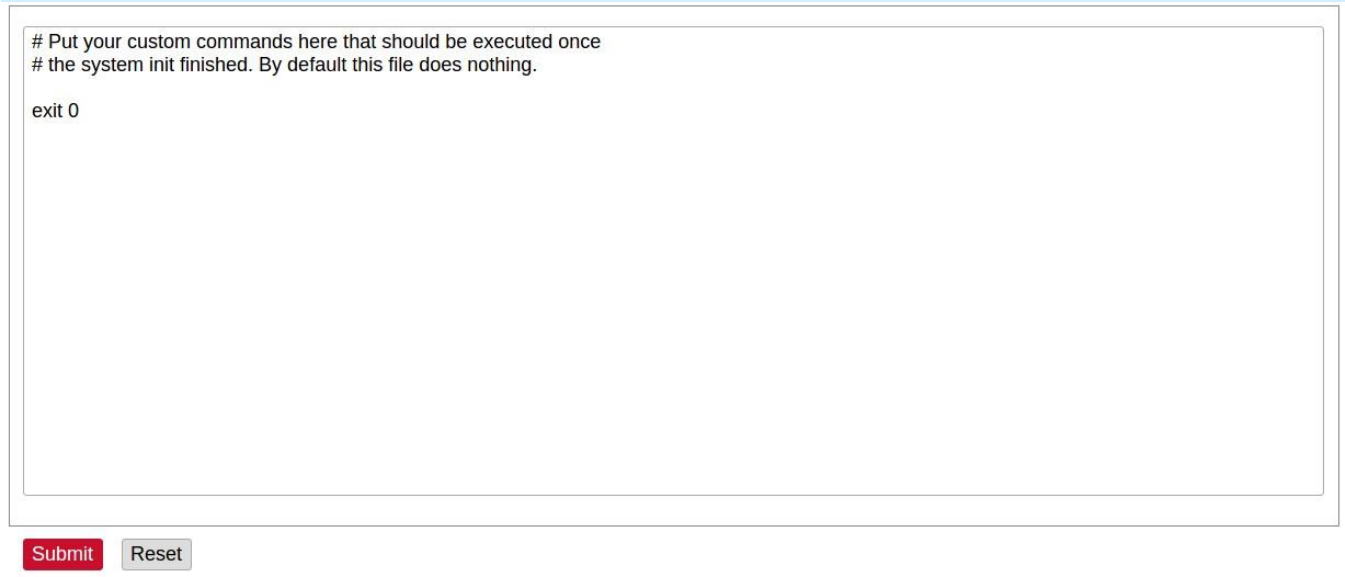



# Startup

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.

# Scheduled tasks

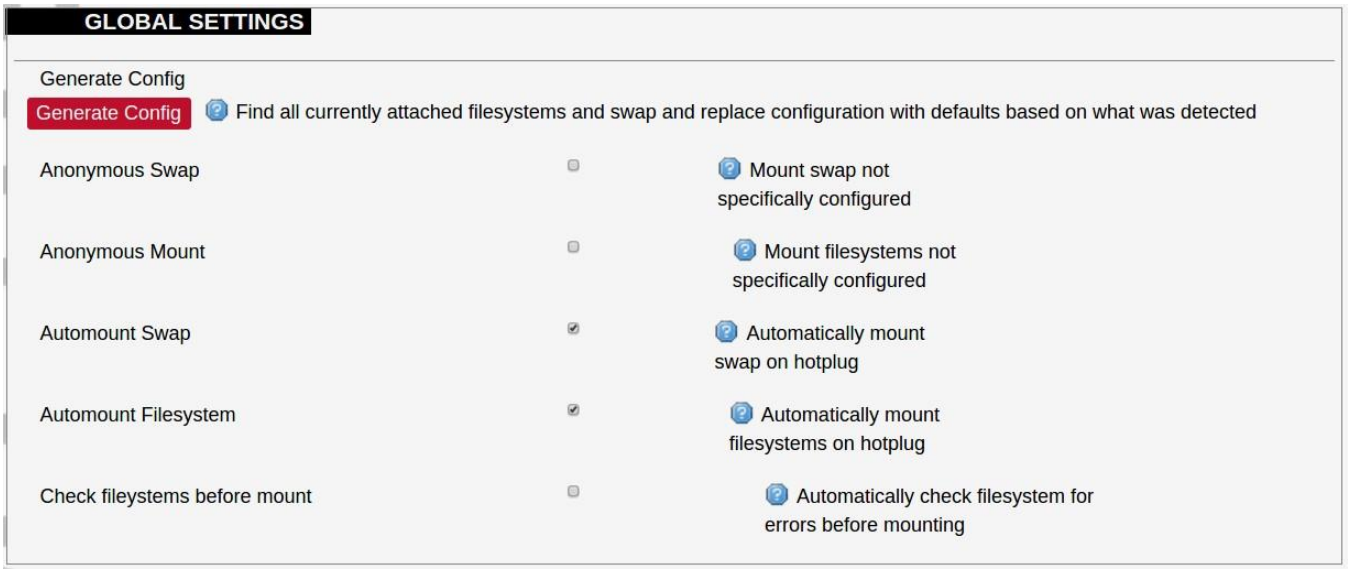



Various tasks can be scheduled with the system crontab. New tasks can be included by creating and saving new rules conforming to cron rules. WCC Lite accepts full cron configuration functionality.

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.

# 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 hot plug.
Automount Filesystem: Automatically mount filesystems on hot plug.
Check filesystems before mount: Automatically check filesystem for errors before mounting.

## Mounted file systems

**MOUNTED FILE SYSTEMS**

| Filesystem | Mount Point | Available | Used | Unmount |
|---|---|---|---|---|
| /dev/root | /rom | 0.00 B / 12.75 MB | 100% (12.75 MB) | |
| tmpfs | /tmp | 28.36 MB / 29.48 MB | 4% (1.13 MB) | |
| /dev/sda3 | /overlay | 833.27 MB / 898.37 MB | 0% (2.64 MB) | |
| overlayfs:/overlay | / | 833.27 MB / 898.37 MB | 0% (2.64 MB) | |
| tmpfs | /dev | 512.00 KB / 512.00 KB | 0% (0.00 B) | |
| /dev/sda1 | /data | 935.69 MB / 1.36 GB | 2% (16.31 MB) | Unmount |
| /dev/sda1 | /tmp/cache/cloud-logs | 935.69 MB / 1.36 GB | 2% (16.31 MB) | Unmount |
| /dev/sda1 | /tmp/cache/cloud-alarms | 935.69 MB / 1.36 GB | 2% (16.31 MB) | Unmount |
| /dev/sda1 | /tmp/lib/redis | 935.69 MB / 1.36 GB | 2% (16.31 MB) | Unmount |

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

## Mount points

**MOUNT POINTS**

Mount Points define at which point a memory device will be attached to the filesystem

| Enabled | Device | Mount Point | Filesystem | Options | Root | Check | |
|---|---|---|---|---|---|---|---|
| ☐ | UUID: 44e3cc6c-139b-410c-86b1-db099c5887c5 (not present) | /mnt/sda1 | ? | defaults | no | no | Edit Delete |
| ☐ | UUID: cc85fea3-836c-4ddc-9828-f35147f21318 (not present) | /mnt/sda2 | ? | defaults | no | no | Edit Delete |
| ☐ | UUID: 1f1c6431-d632-4e11-9c12-3c913d3986e7 (not present) | /mnt/sda3 | ? | defaults | no | no | Edit Delete |
| ☐ | Label: overlay (/dev/sda3, 929 MB) | /overlay | ext4 | defaults | overlay | no | Edit Delete |

Add

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.

**SWAP**

If your physical memory is insufficient unused data can be temporarily swapped to a swap-device resulting in a higher amount of usable RAM. Be aware that swapping data is a very slow process as the swap-device cannot be accessed with the high datarates of the RAM.

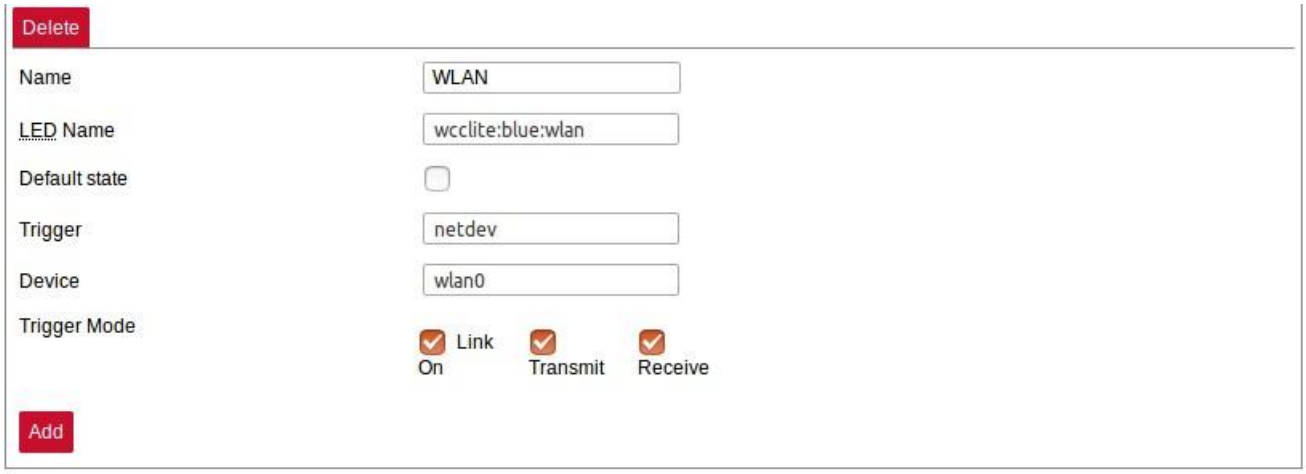
| Enabled | Device |
|---|---|
| This section contains no values yet | |

Add

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.

# 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: Color 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. 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, phy0radio, phy0tpt, phy0assoc | Flashed on Wi-Fi activity events |
| usbdev | Turned on when USB device is connected. Applicable for modems |

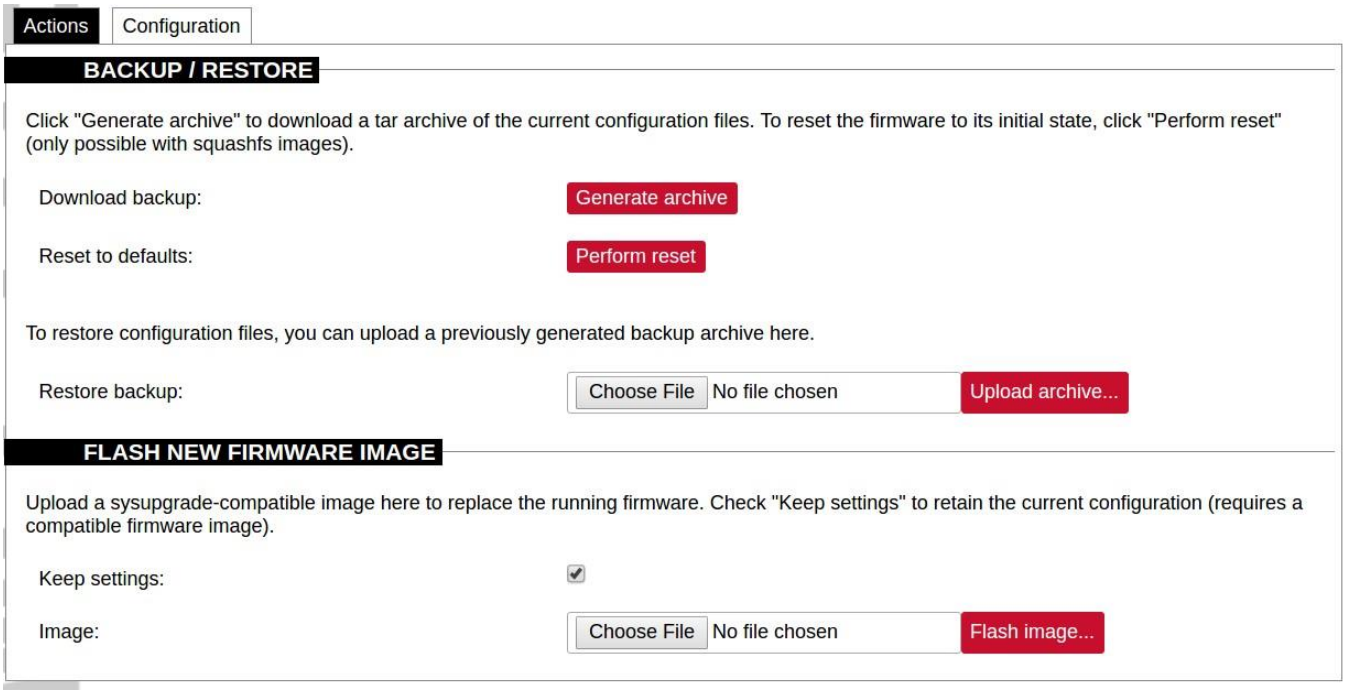
Device: Network interface which is going to be tracked.

# 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.

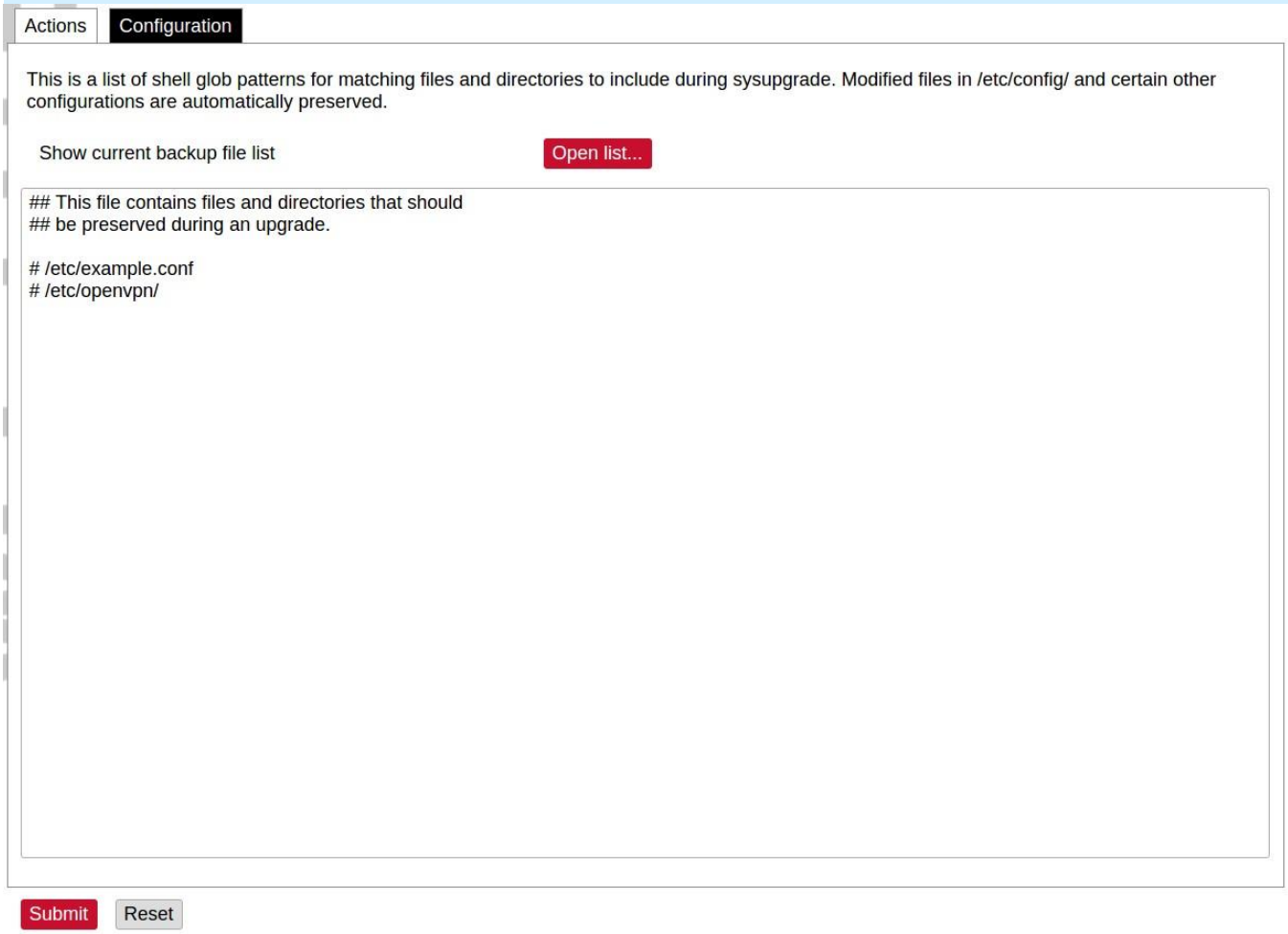
Backup archives contain complete WCC Lite configuration that can be restored at any time. A file will be downloaded by your browser when creating a backup. This file can be later uploaded to the web page to restore configuration.

⚠ Generated backup archive should only be applied to the same firmware version it was generated. Applying backup to a different firmware version might render some parts of operating system unstable or even unusable



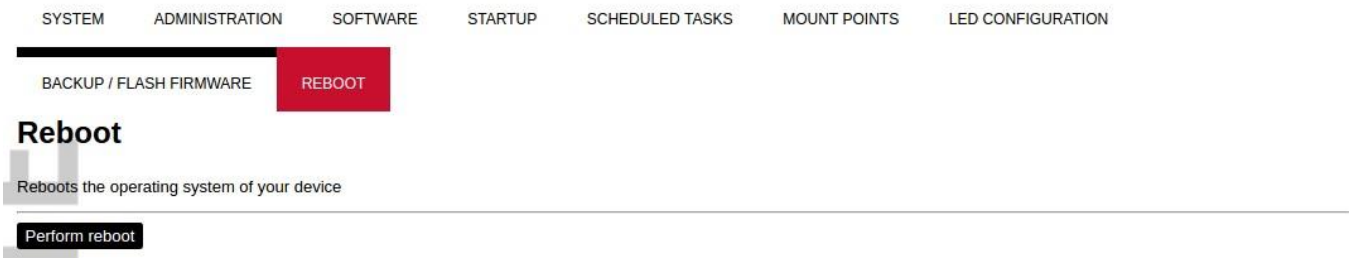

A user can choose to keep existing settings after an upgrade. Marking Keep Settings checkbox preserves files listed in /etc/sysupgrade.conf and /lib/upgrade/keep.d/. It is advised to do a clean install and use backup files to restore settings later if a user intends to make a major system upgrade.

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.

# 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:




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;

# 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.

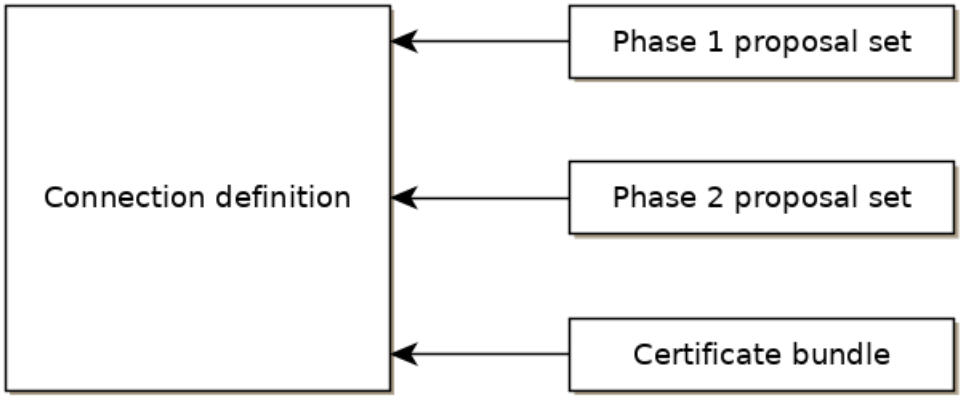
ℹ Enabling agent and saving the configuration automatically starts the process with the new configuration.

# 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:<br>• none – default, disables sending of DPD messages.<br>• clear – the connection closed with no action.<br>• hold – keeps description, tries renegotiate connection on demand. restart – will try to re-<br>• negotiate immediately. |
| 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 |

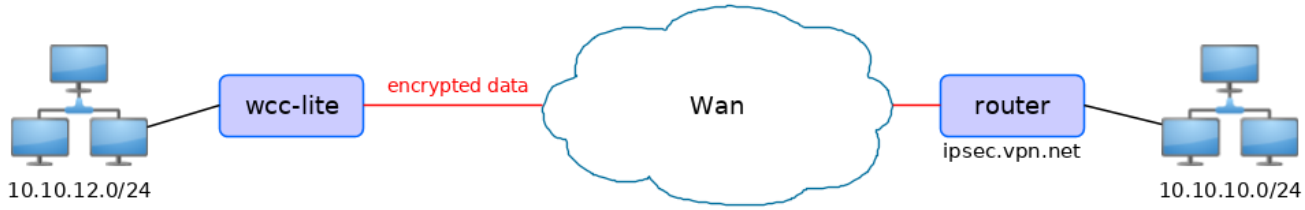
Phase 2 proposals - ESP cipher suite components:

| 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

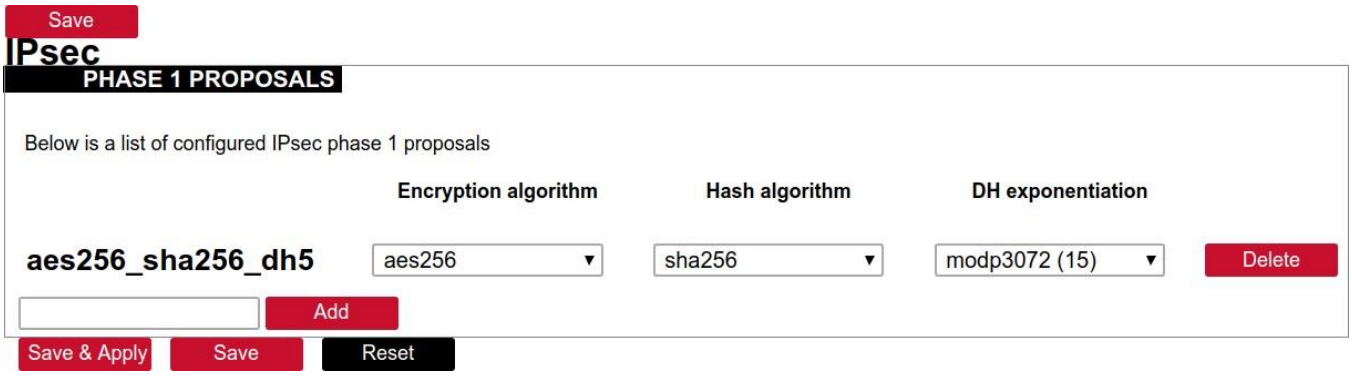
Tunnel: demo

```
IPsec peer: ipsec.vpn.net
Pre-shared key: thebigsecret
Mode: tunnel
Remote network: 10.10.10.10/24
Local network: 10.10.12.0/24
Local ID: wcclite
IKE authentication: aes256 IKE hash: sha256
IKE DH group: 5 (modp1536)
ESP authentication: aes128
ESP hash: sha1
```

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

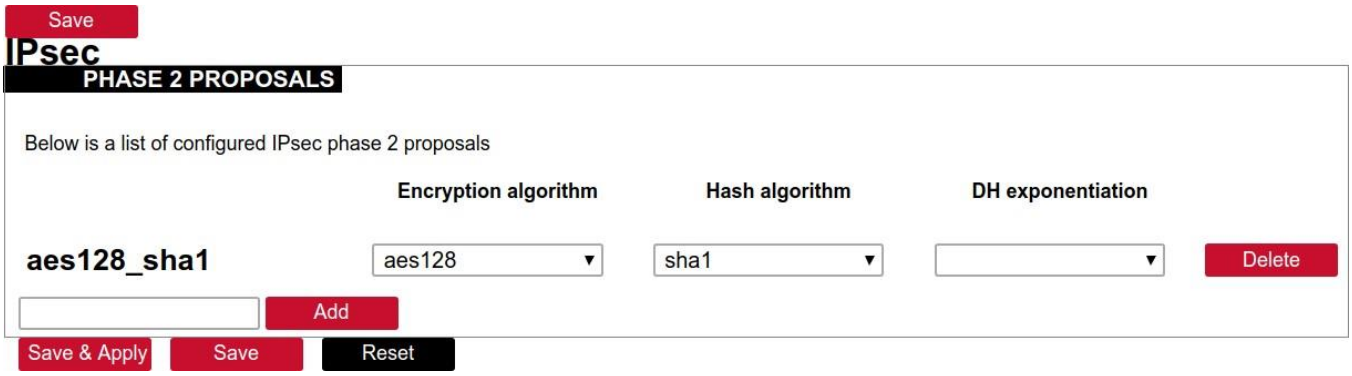
## Creation of Phase 1 proposal

- 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.

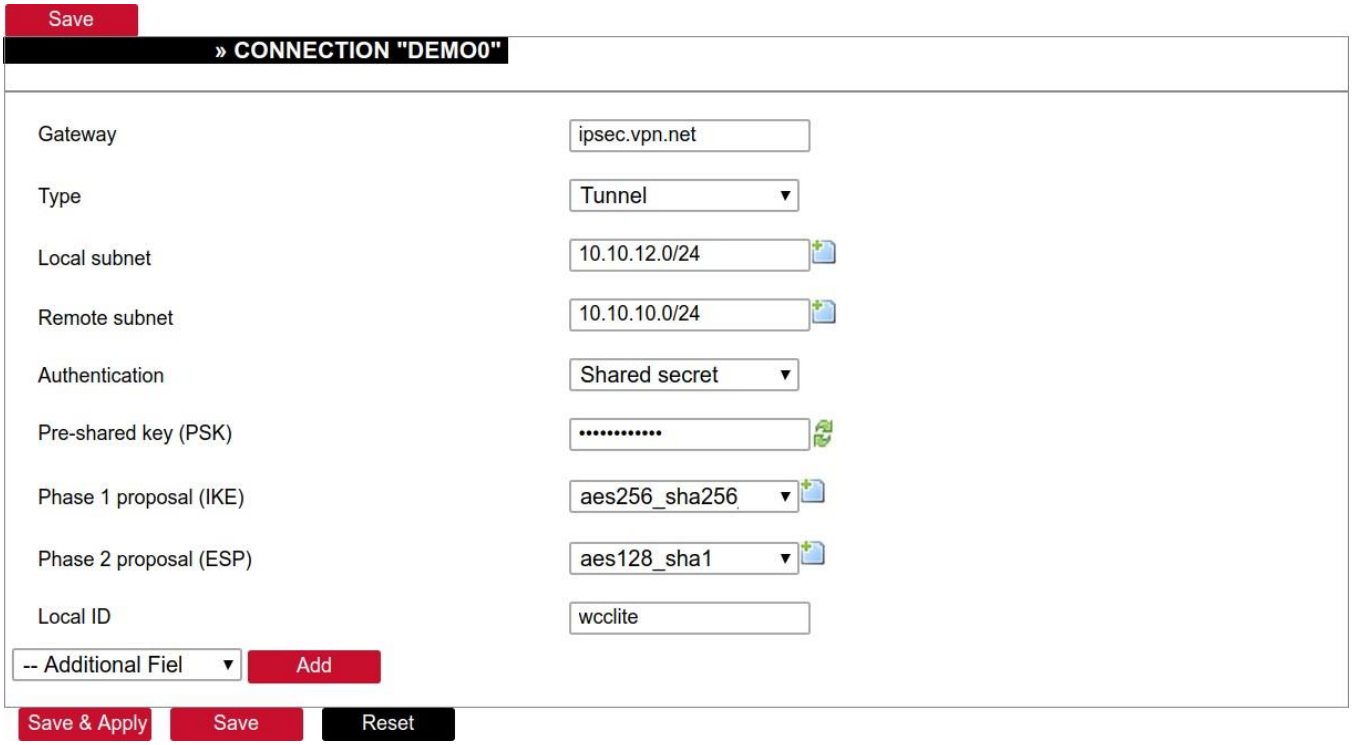



## Creation of tunnel definition

Enter section connections

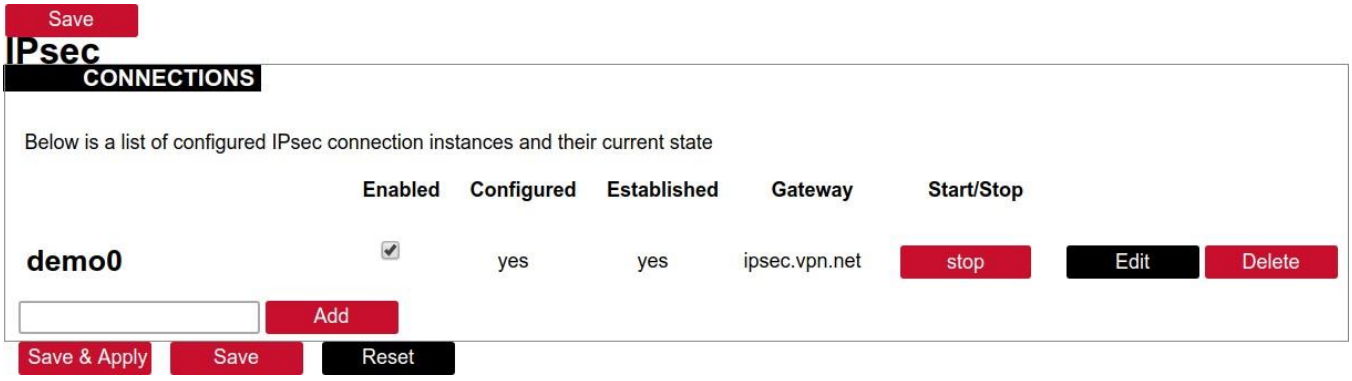
- Create a new record by assigning new name (e.g.“demo0”) and clicking “Add”.
- Call a detail form by pushing the button “edit”.
- Enter peer address into “Gateway”: “ipsec.vpn.net”.
- Ensure “Type” is set to: “Tunnel”.
- Fill local subnet to: 10.10.12.0/24.
- Fill remote subnet to: 10.10.10.0/24.
- 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”.




# 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.

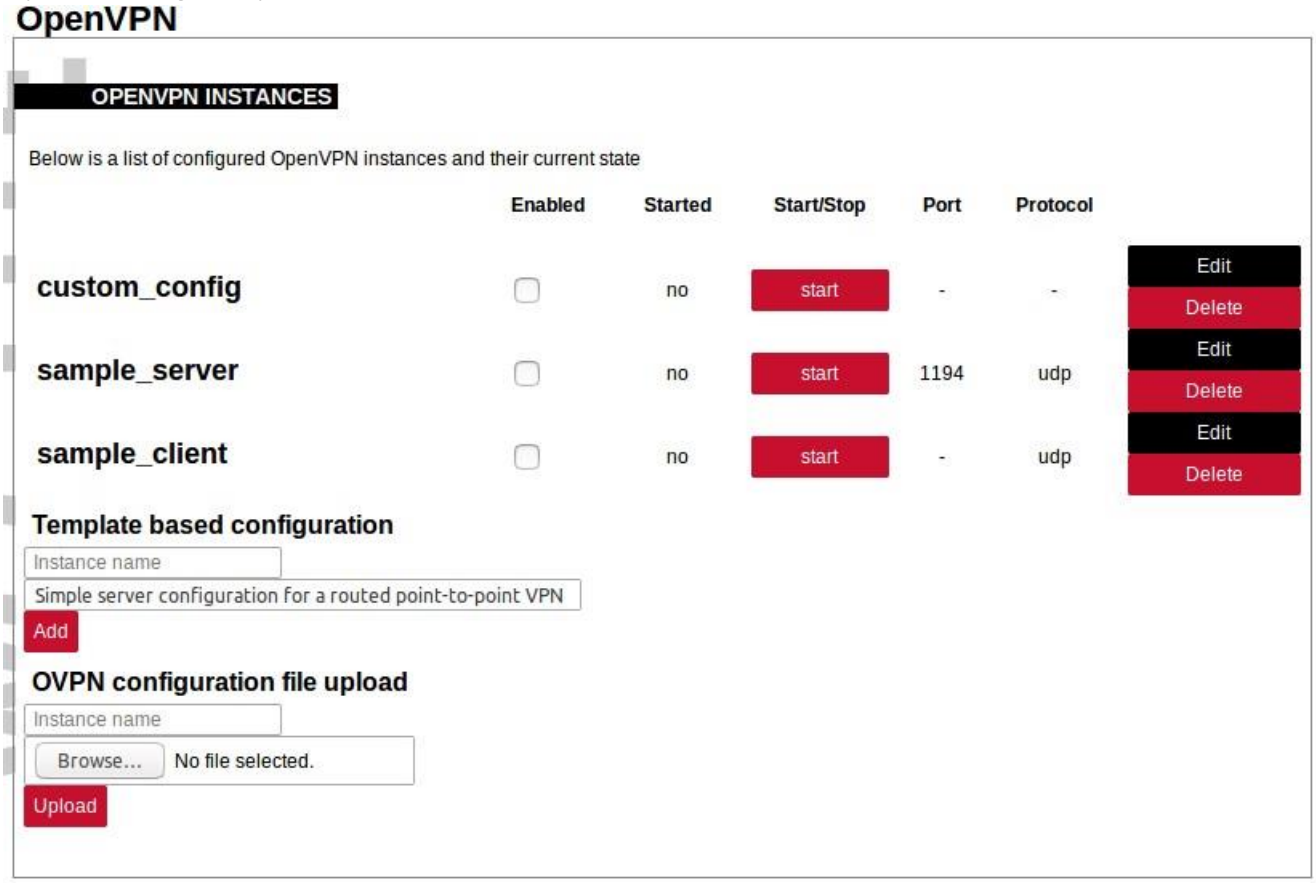
When the process is complete, L2TP packets between the endpoints are encapsulated by IPsec. Since the L2TP packet itself is wrapped and hidden within the IPsec packet, no information about the internal private network can be gathered from the encrypted packet. Also, it is not necessary to open UDP port 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.

# OpenVPN

## OpenVPN Instances

The primary goal is to get a working WCC Lite tunnel and establish a basic platform for further customization. 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.

# 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.

# 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.

> ℹ All endpoints are disabled by default.

Available API endpoints are shown in the table below.

Table. Available API endpoints:

| Endpoint | 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".

# 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 filesystem 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).

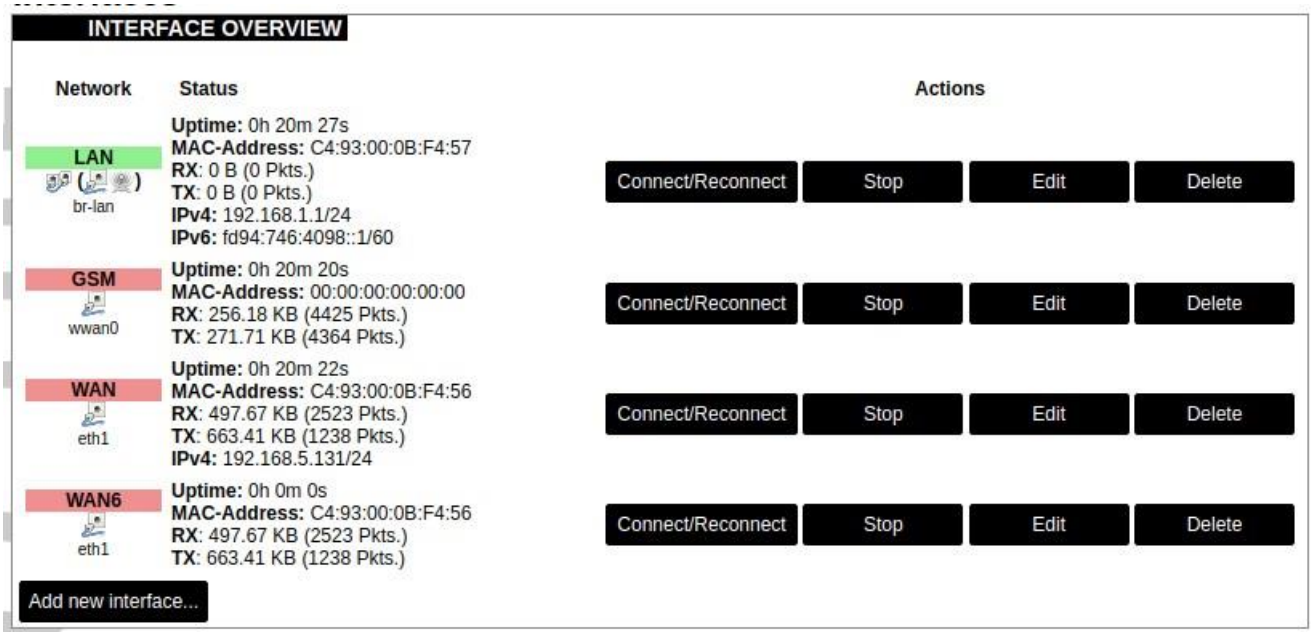
# 8.7 Network




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.

## 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.

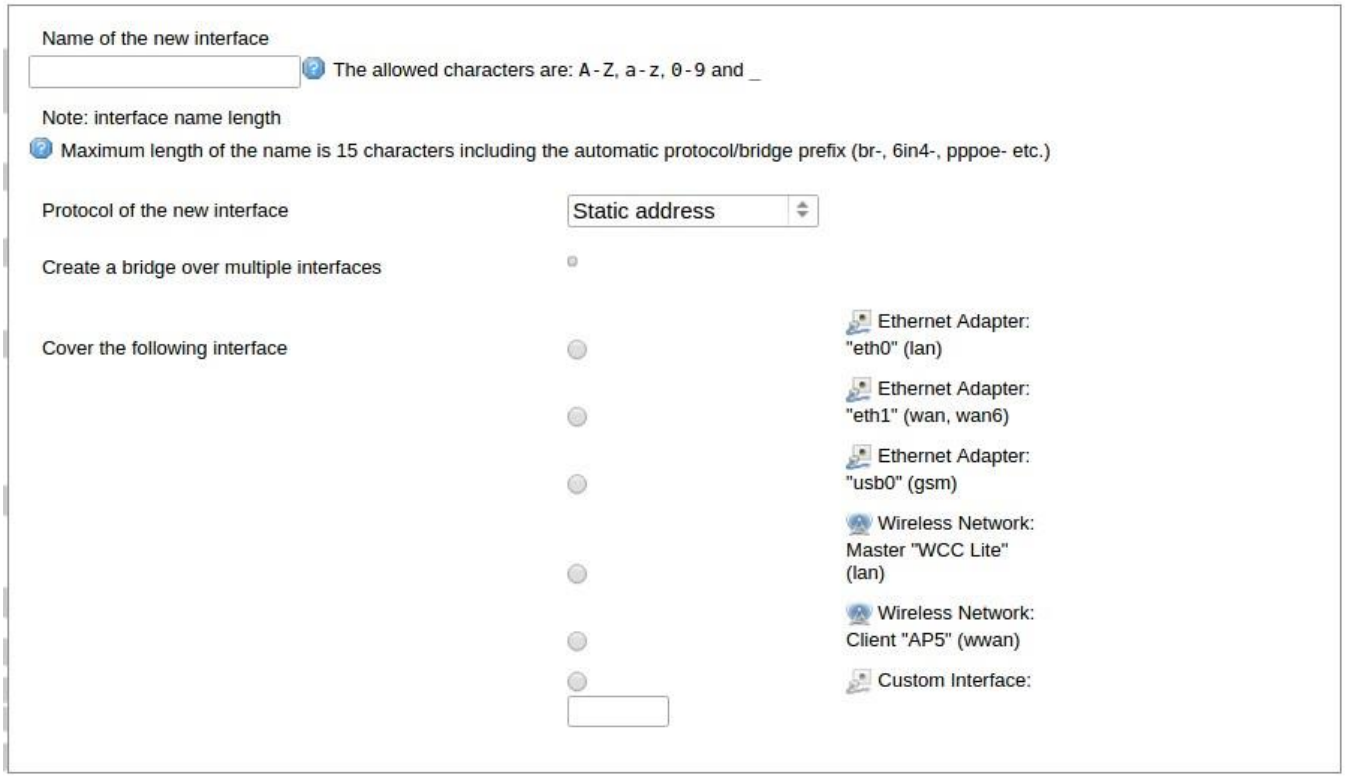
In addition to the network interface status, several actions may be performed:
Connect/Reconnect: Connect to configured interface network if it does not do it automatically. If it already connected to the network it will be trying to reconnect to it.
Stop: Shutdown interface. If you are connected through 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.

| | | |
|---|---|---|
| | Eth0 | eth1 |

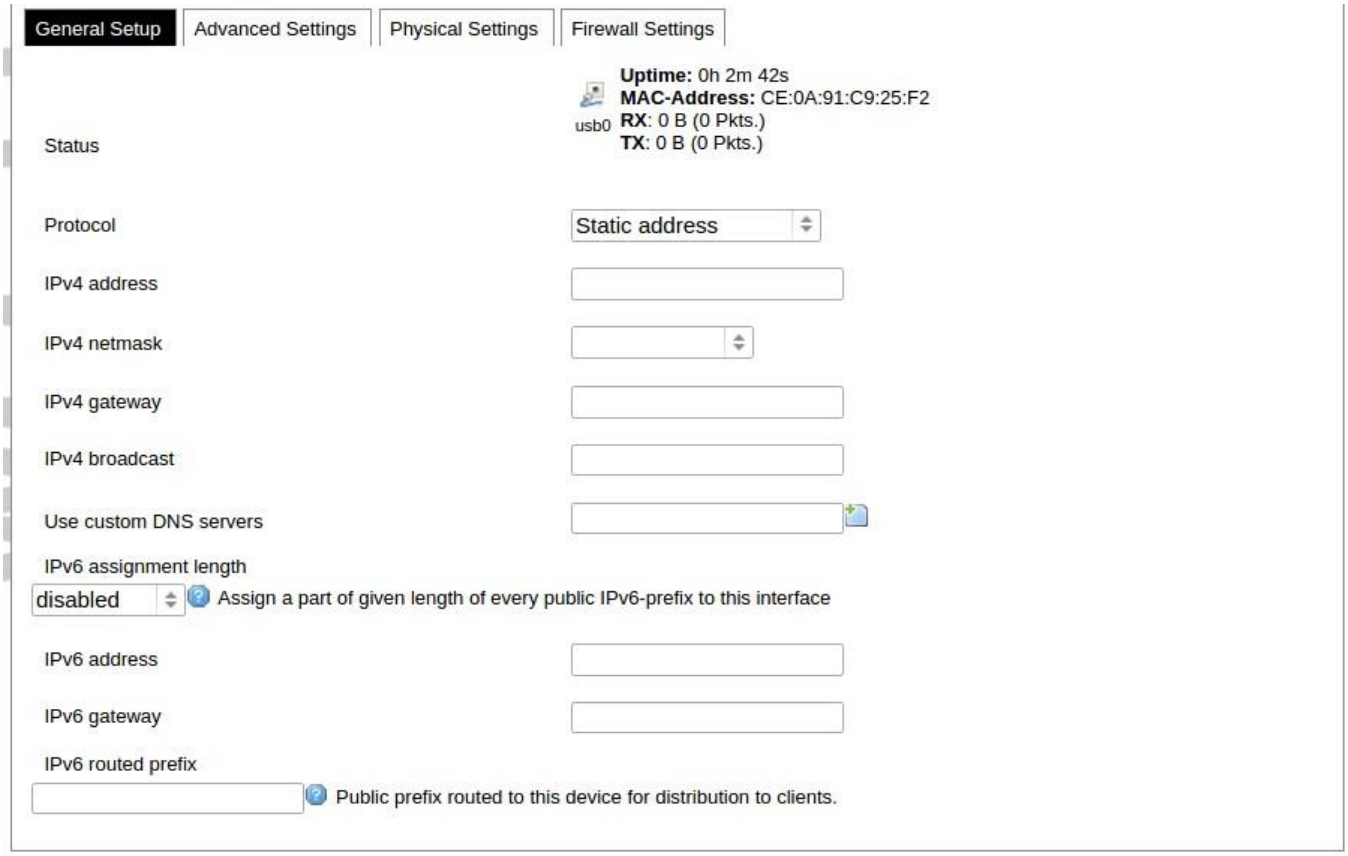
| | | |
|---|---|---|
| Type | Static | DHCP |
| Address | 192.168.1.1 | |
| Subnet mask | 255.255.255.0 | |
| Gateway | | |

Changes will only take effect after device reboots.

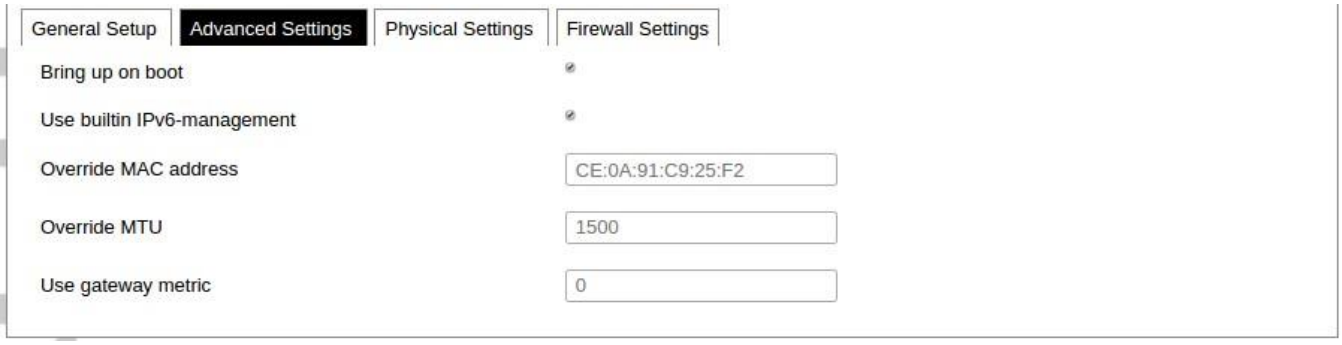
Network interfaces can be configured on the common page, which can be accessed through add new interface or edit button.



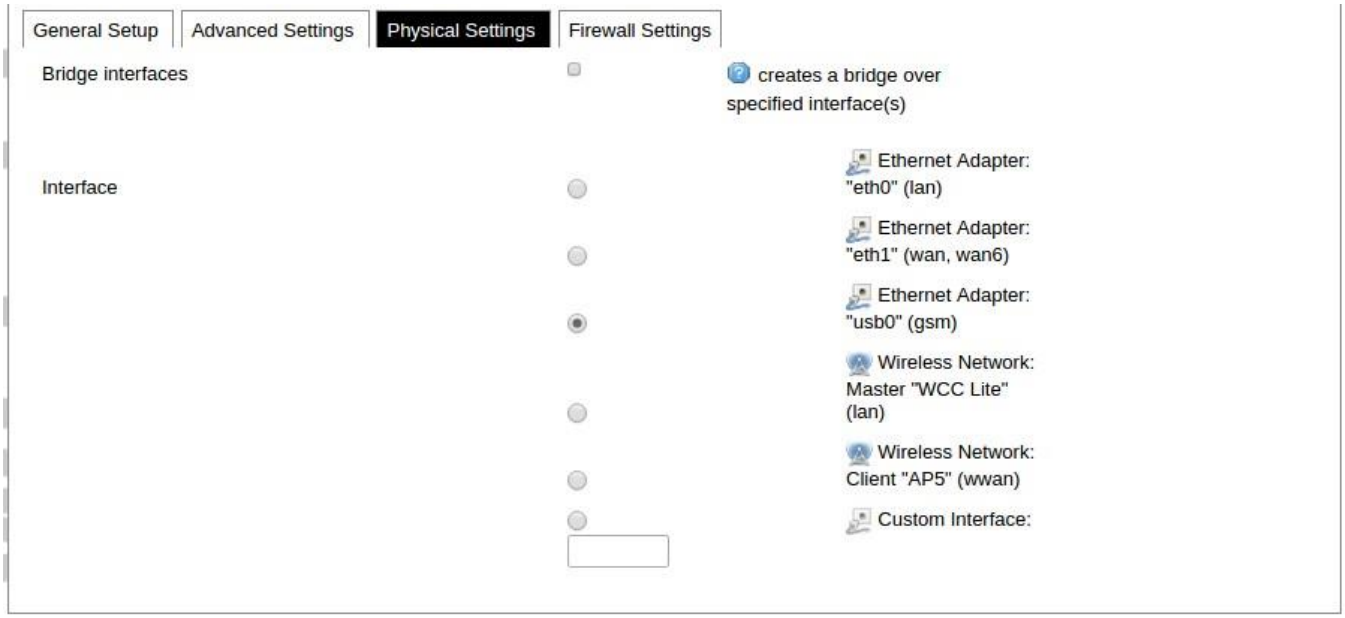

The following options can be defined in the interface creation panel: name of the interface, protocol, coverage of a particular interface or bridging with other interfaces. After the general setup is done, more detailed settings can be set.



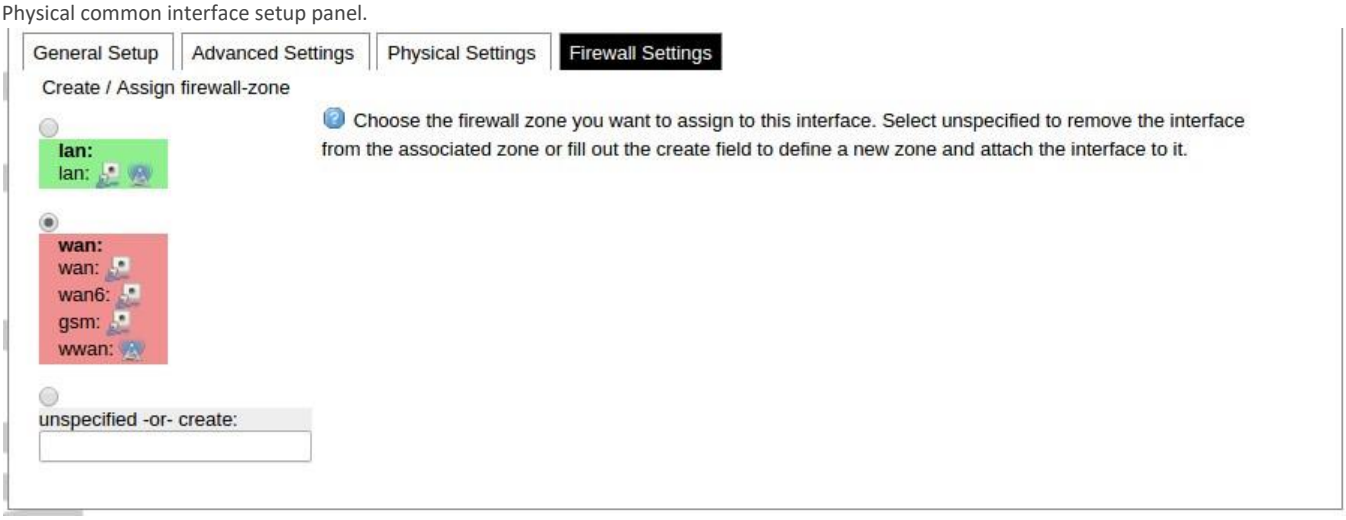

General common interface setup panel.



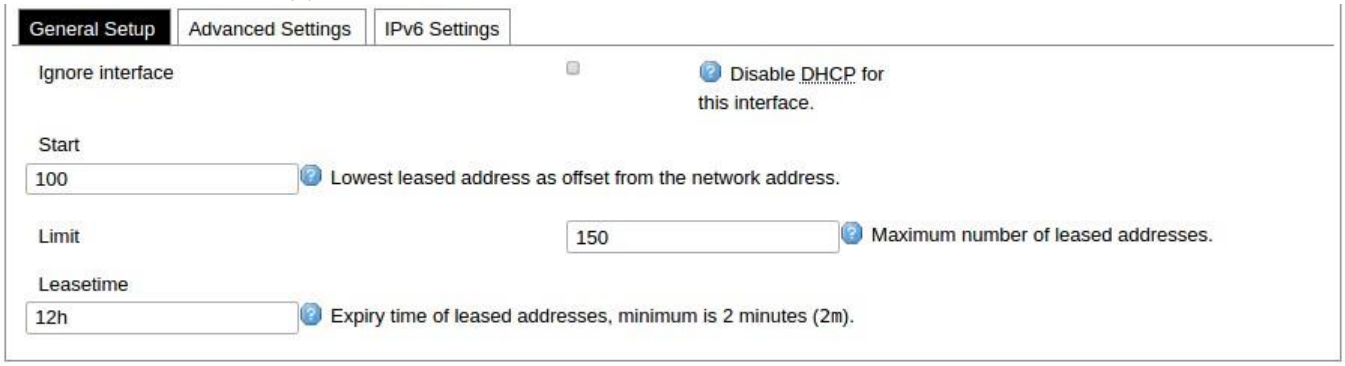

Advanced common interface setup panel.



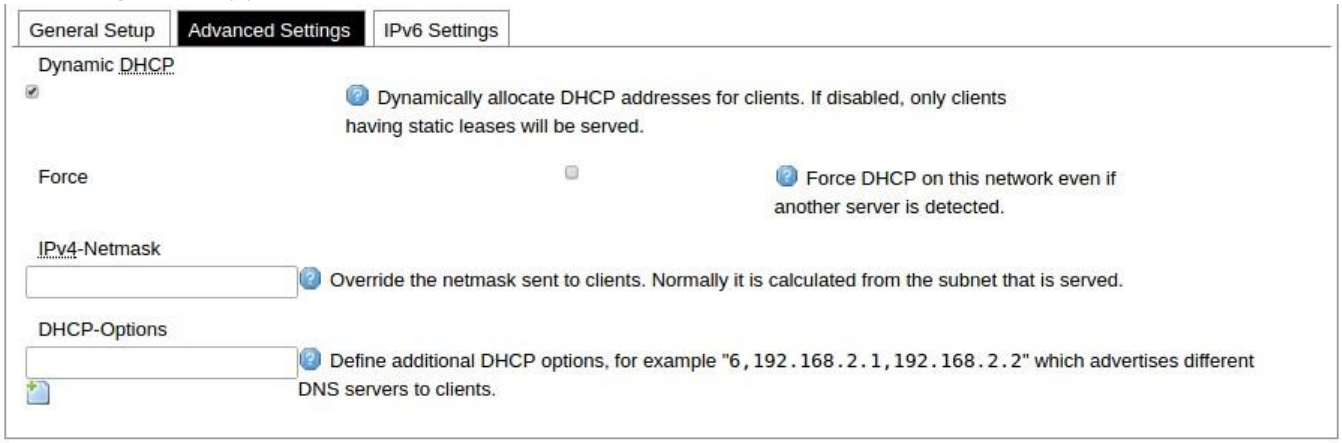
Physical common interface setup panel.



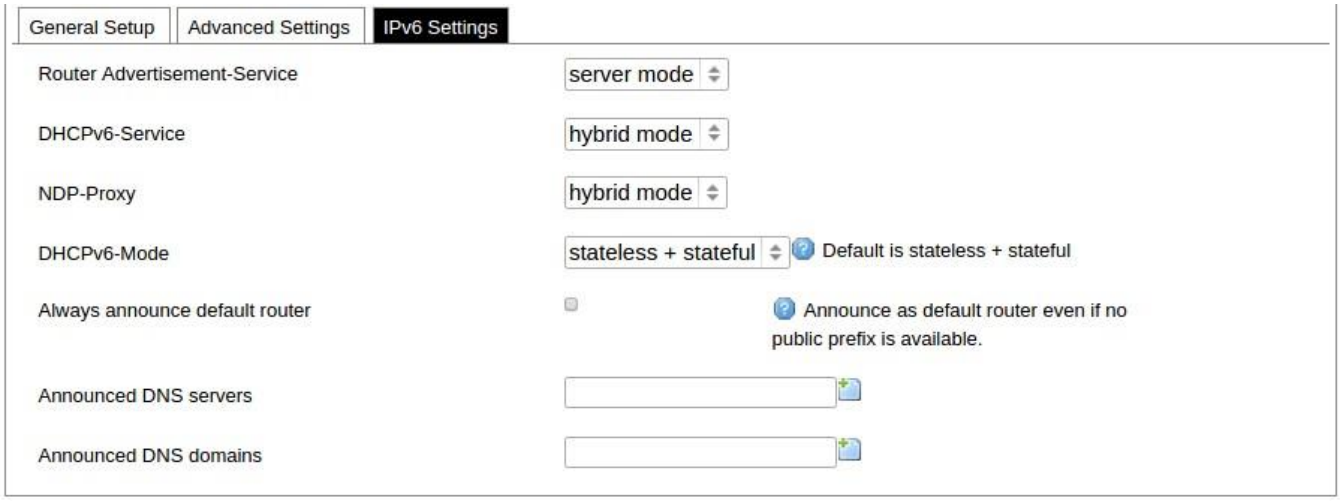

Firewall common interface setup panel.




DHCP server general setup panel.

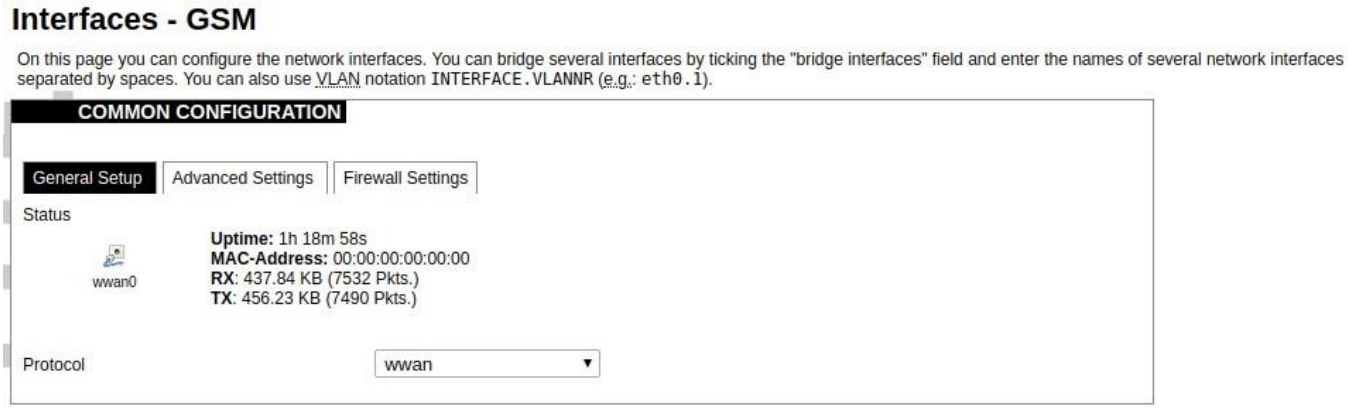



DHCP server advanced setup panel.



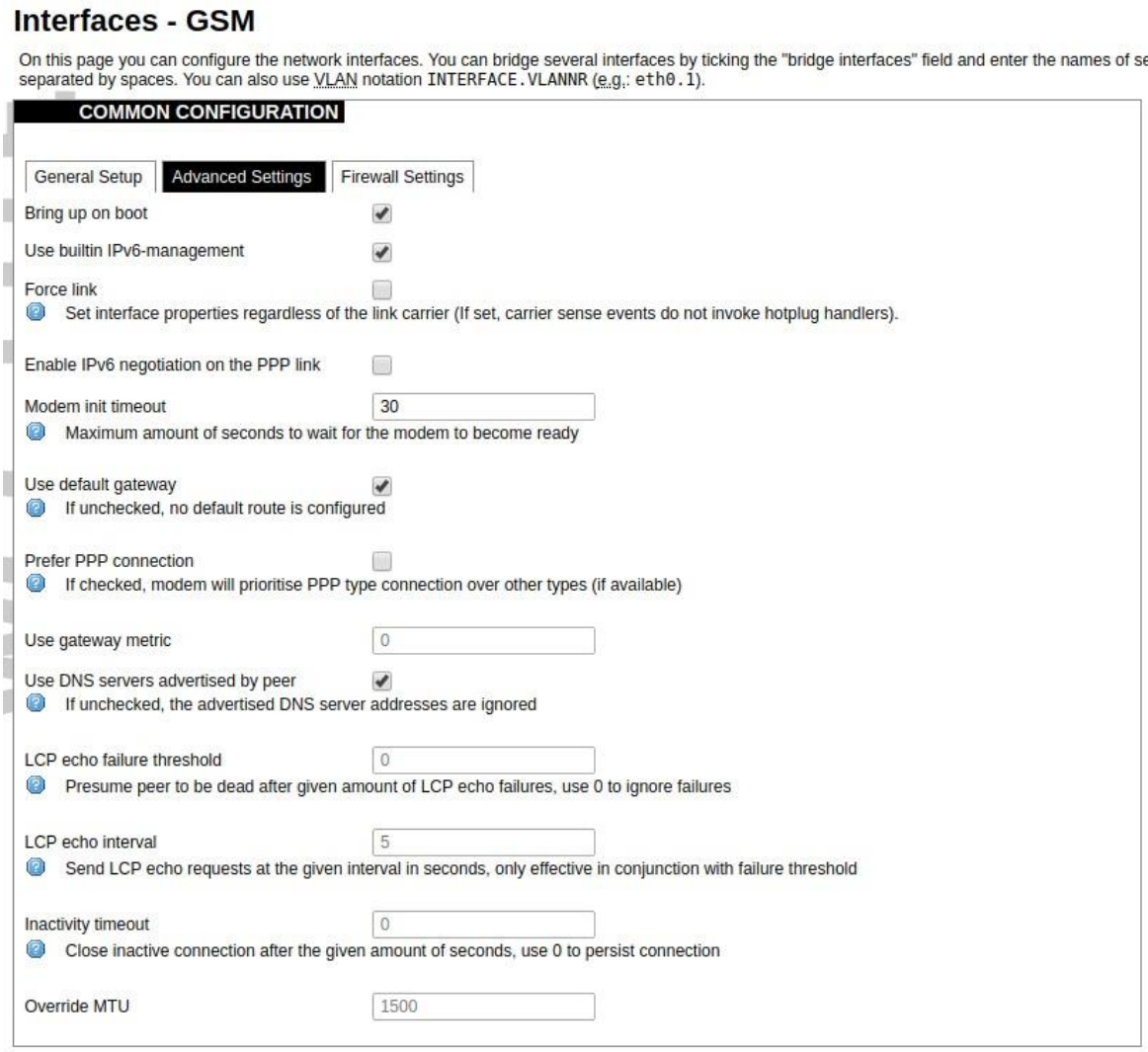

DHCP server IPv6 settings setup panel.

**GSM**



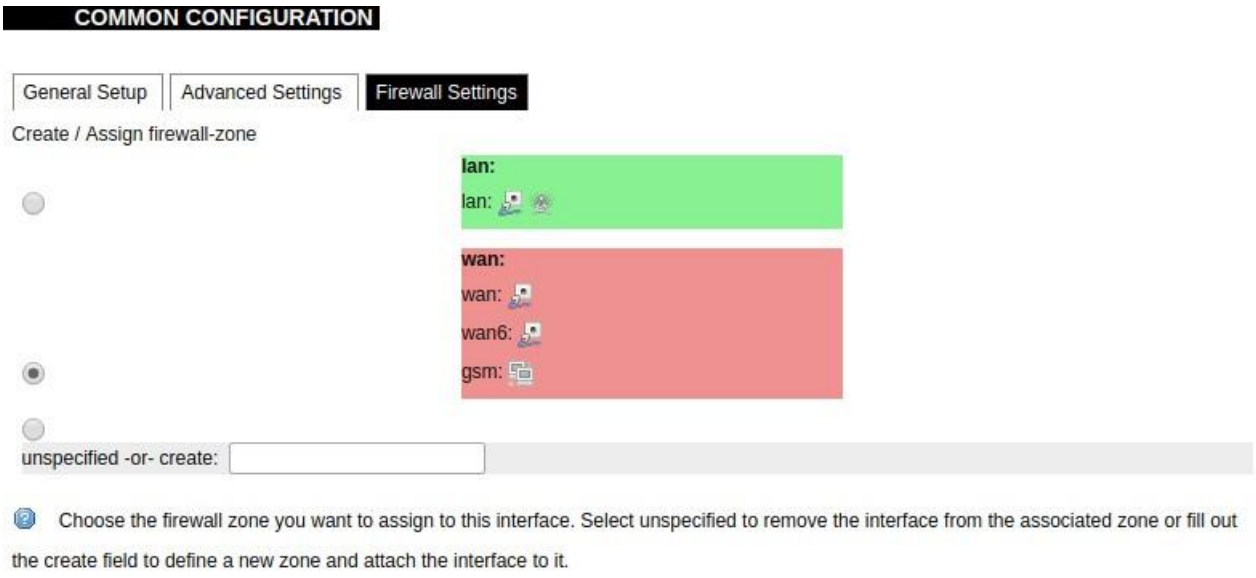

General Settings Information tab. Gives you name of physical GSM interface, lets you choose protocol (not recommended!).

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




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 built-in 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.), prioritize 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.

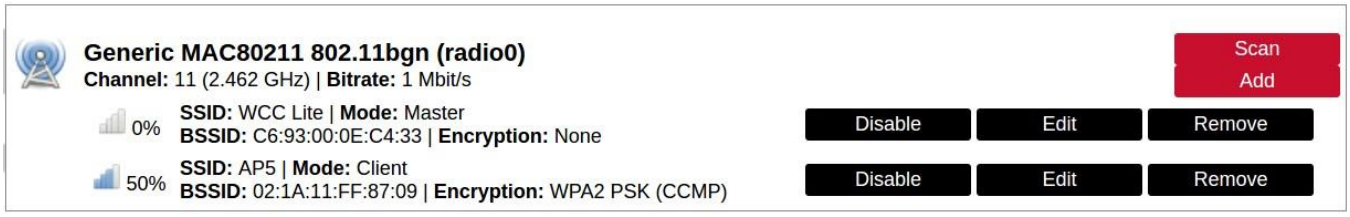
Note: If modem uses QMI connection protocol and user haven't defined custom MTU setting, the MTU on interface will be set to operator's defined MTU value.




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

# Wireless

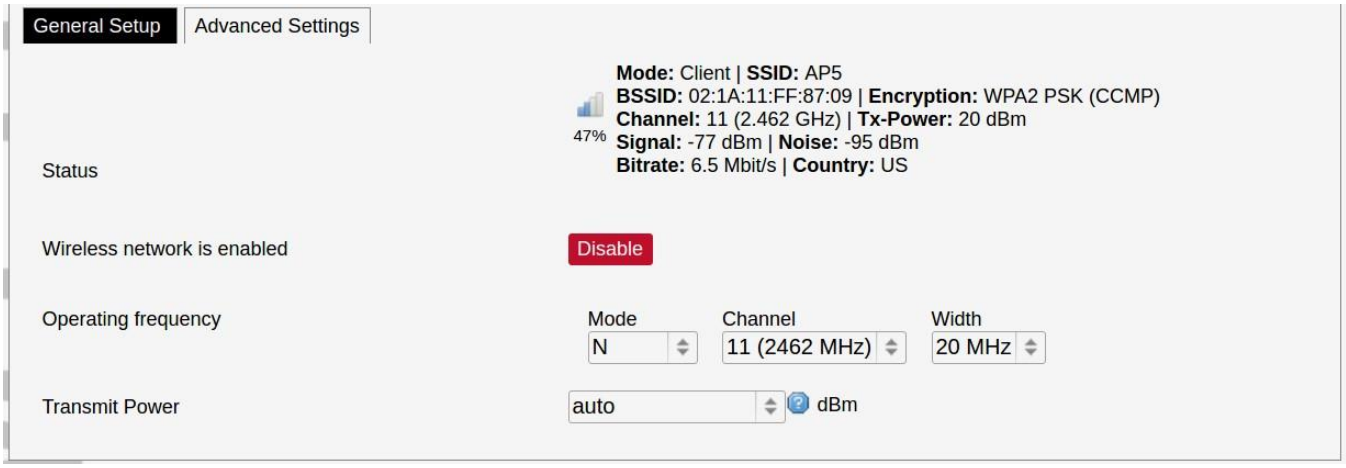
The wireless network interface parameters and configuration are described in this section.



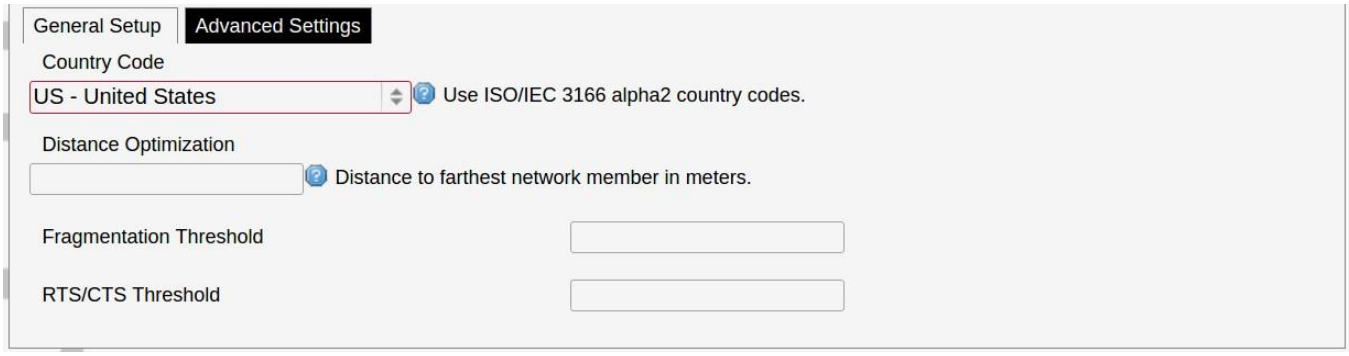

Configured interfaces for the physical radio device.
Channel: Specifies the wireless channel to use.
Bitrate: Specifies transfer rate in Mbit/s.
SSID: The broadcasted service set identifier of the wireless network.
Mode: Selects the operation mode of the wireless network interface controller.
BSSID: The basic service set identification of the network, only applicable in ad hoc or STA mode. Encryption: Wireless encryption method.



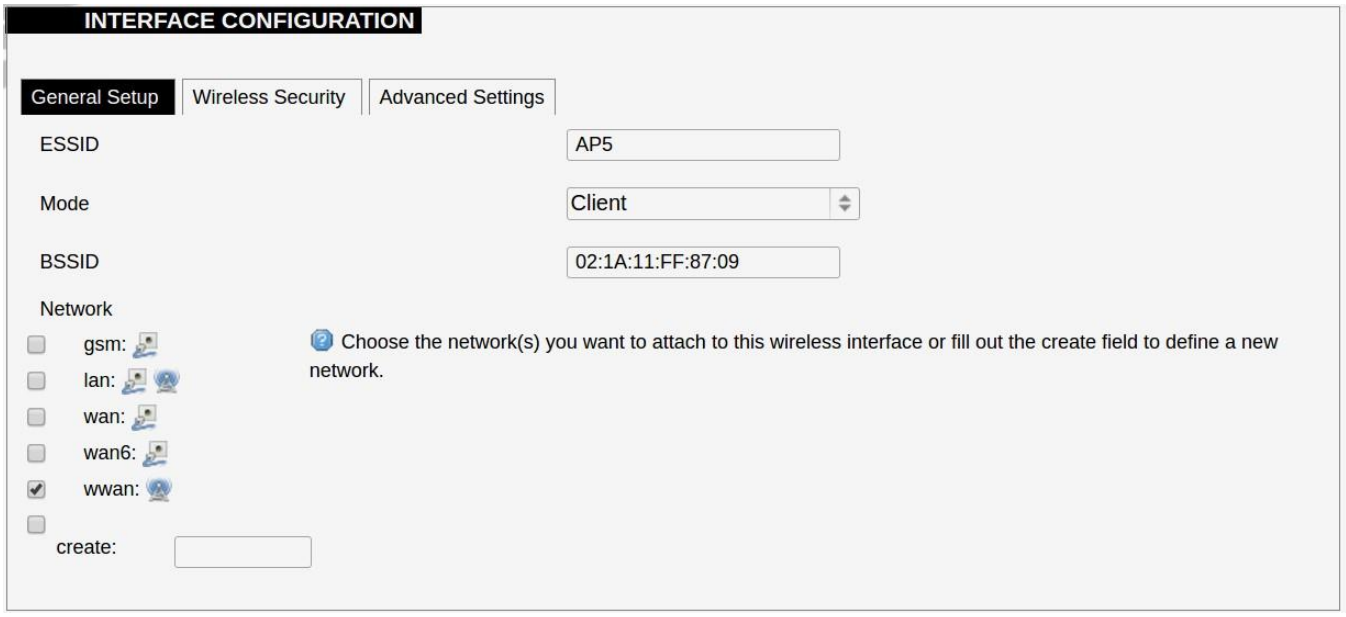

List of associated wireless stations.
The Device Configuration section covers physical settings of the radio hardware such as channel, transmit power or antenna selection which are shared among all defined wireless networks (if the radio hardware is multi-SSID capable). Per network settings like encryption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



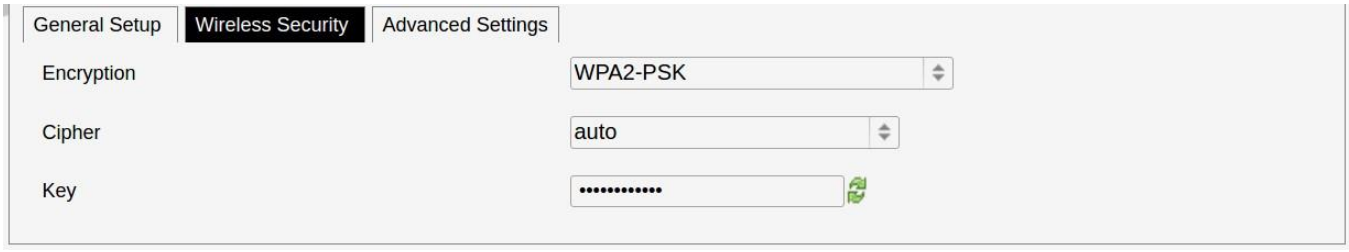

General device settings.



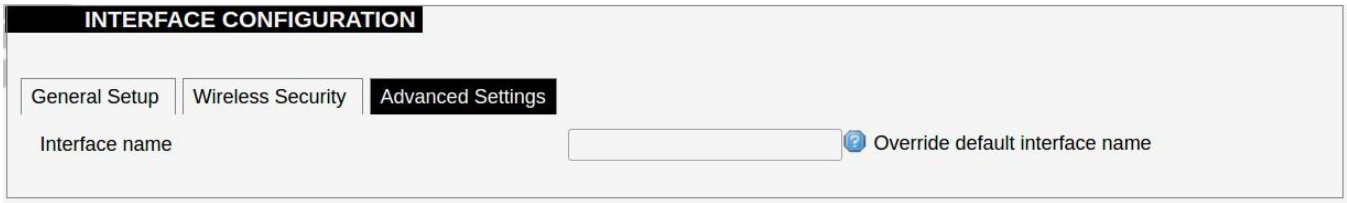

Advanced device settings.




General interface settings.



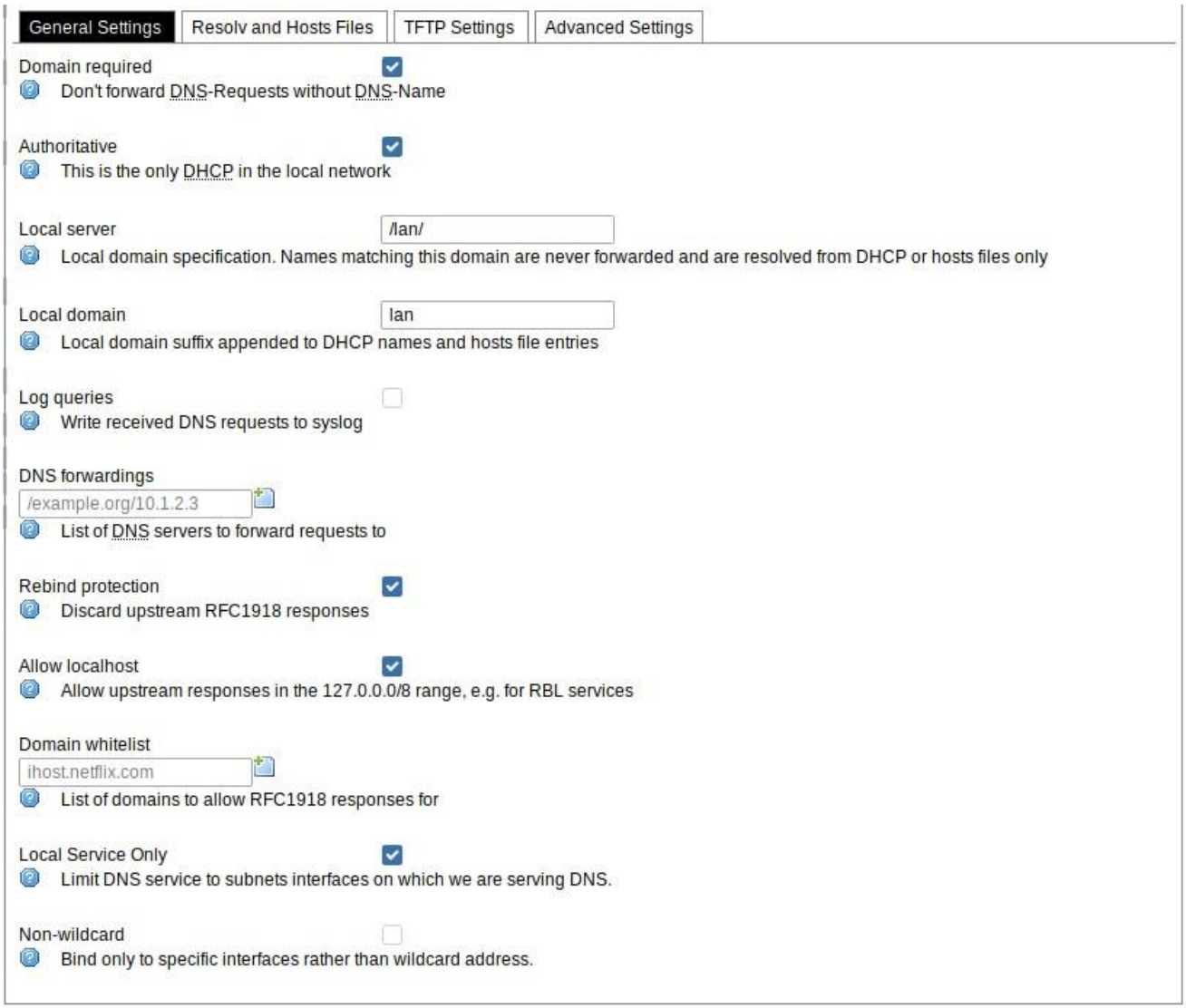

Wireless security interface settings.



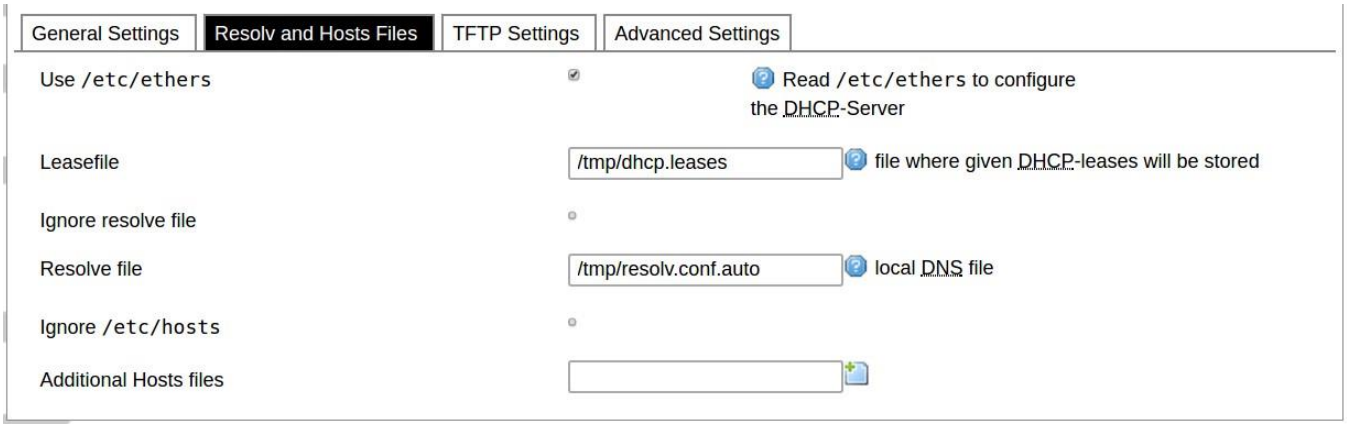

Advanced interface settings.

# DHCP and DNS

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



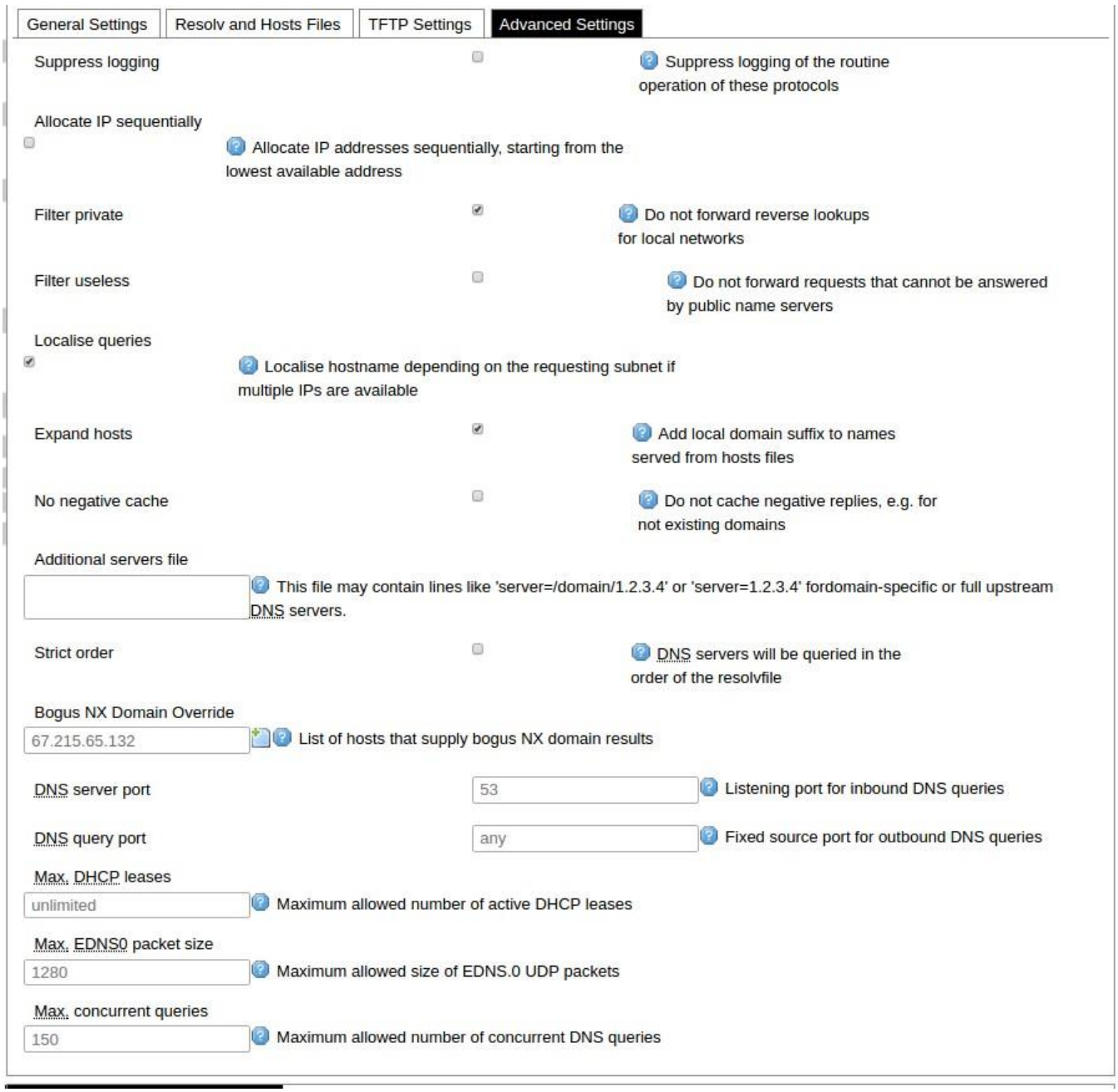

General DHCP settings.



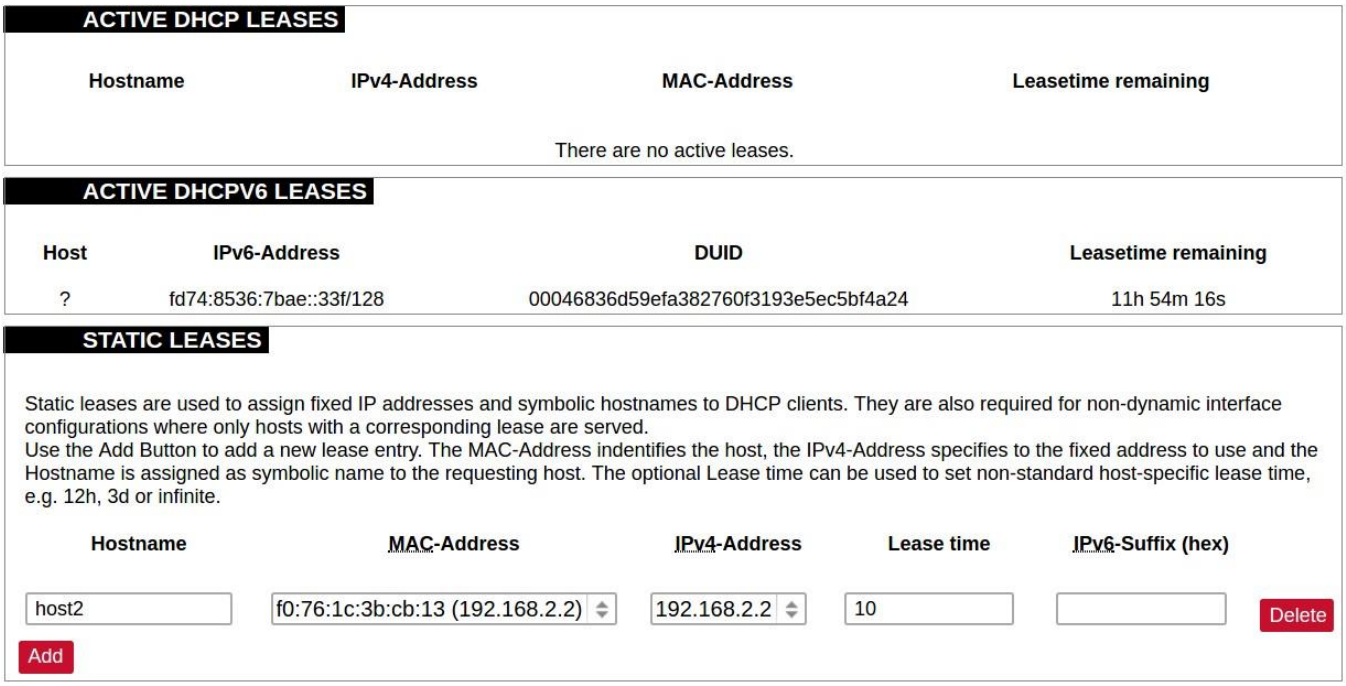

Resolve and hosts files settings.




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.

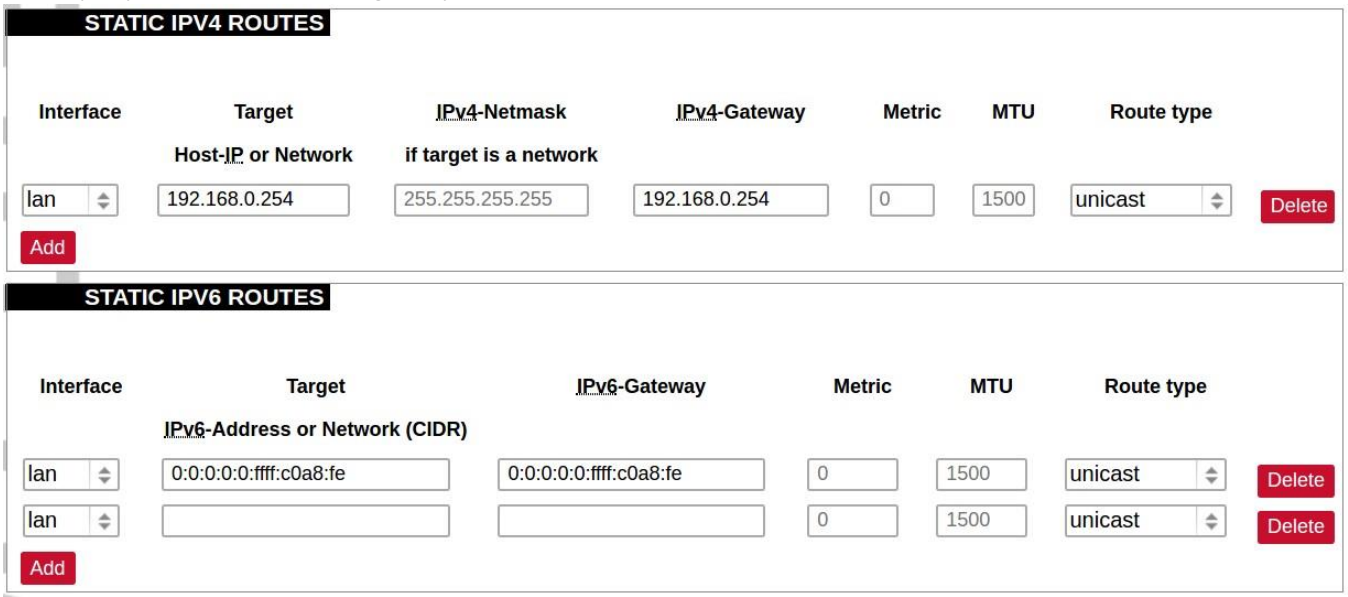
# Hostnames




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

# Static routes

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

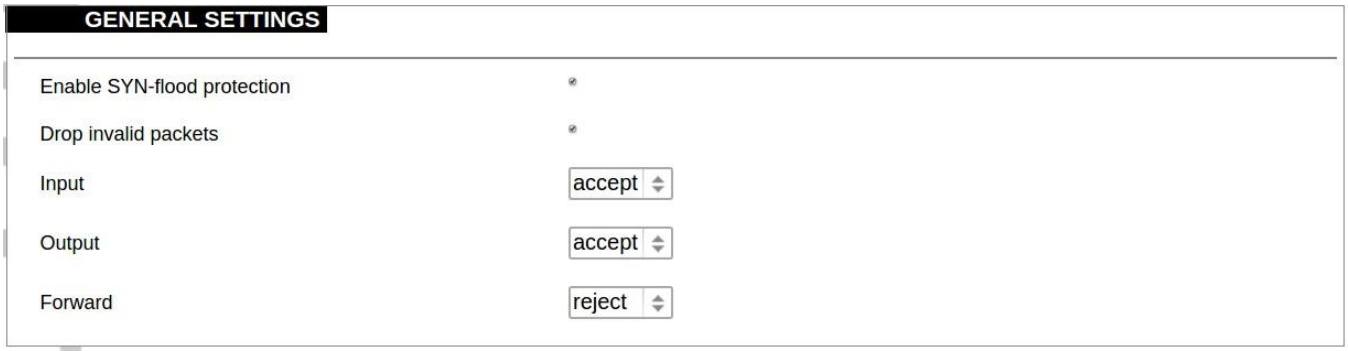


Current IPv4 and IPv6 static routes configuration.
Interface: Lets to chose for which interface static route is created.
Target: Defines target host IP or network.
IPv4 Netmask: Defines netmask if the target is a network.
IPv4/IPv6 Gateway: Defines IPv4 or IPv6 gateway.
Metric: Specifies the route metric to use for the route.
MTU: Maximum Transmit/Receive Unit, in bytes.
Route type: All incoming packets can be: accepted, rejected, dropped.

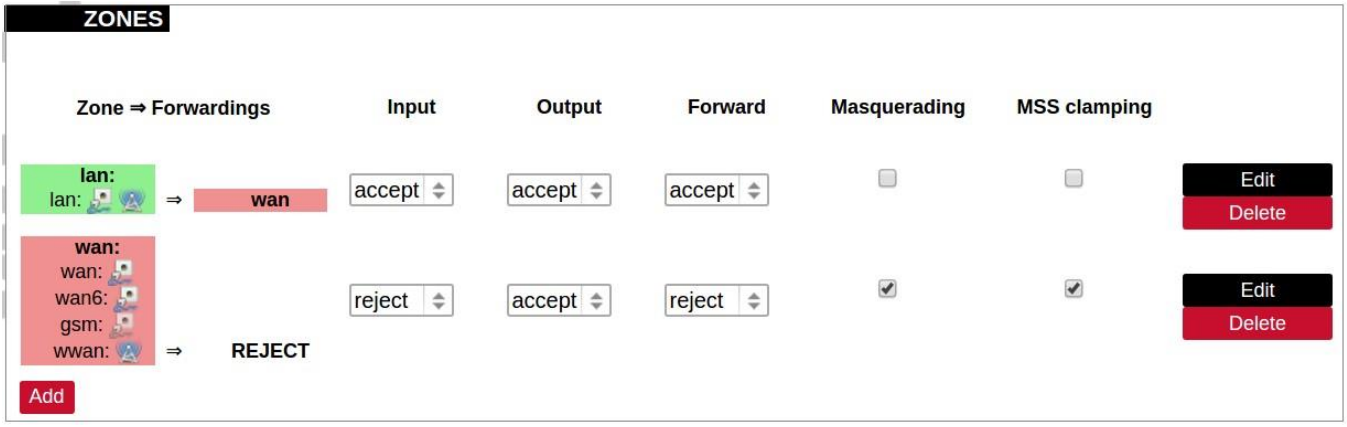
# Firewall

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

## General settings

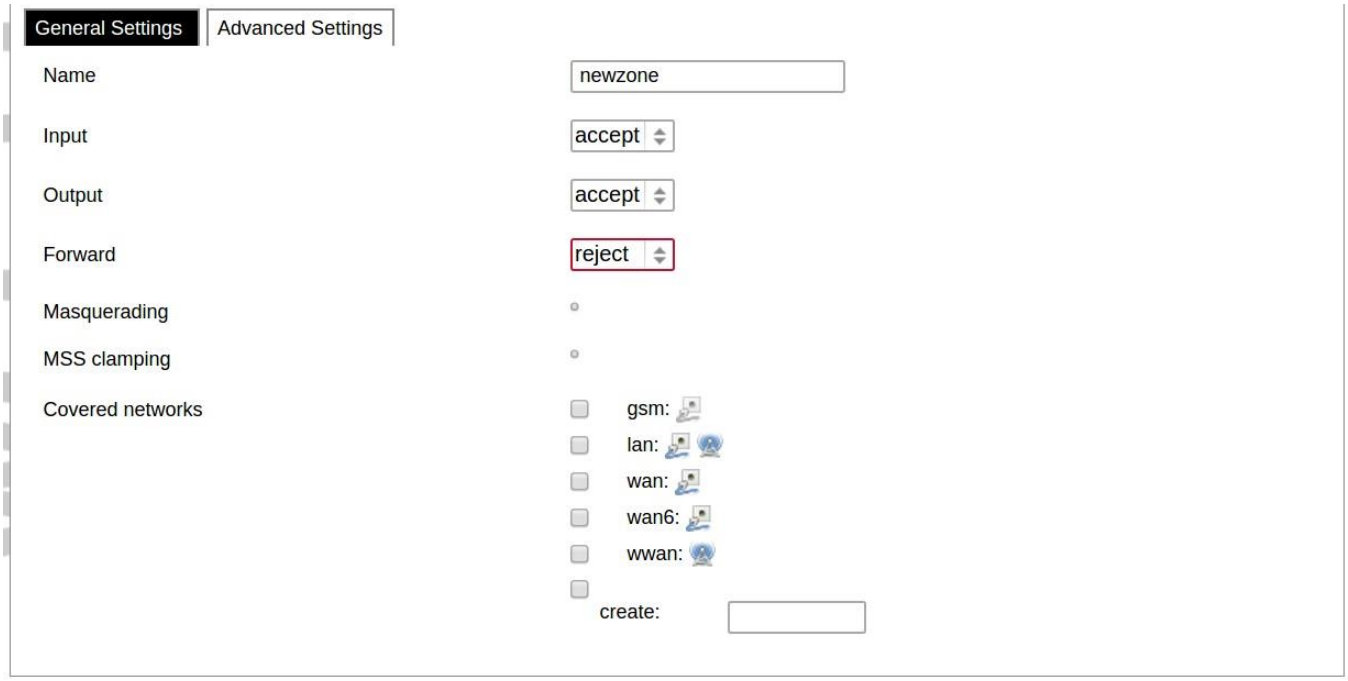



General Settings for firewall can be changed in General Settings screen. These settings are defined as follows:
Input: All incoming packets can be: accepted, rejected, dropped.
Output: All outgoing packets can be: accepted, rejected, dropped.
Forward: All packets being sent to another device can be: accepted, rejected, dropped.

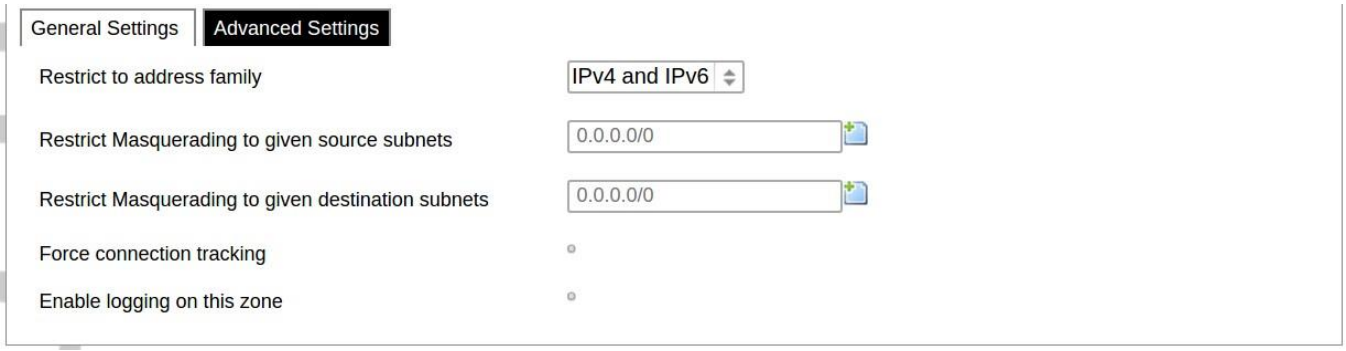



Additional zones for firewall can be created, edited or deleted.
Zone => Forwardings: Defines zones and their traffic flow.
Input: All incoming packets can be: accepted, rejected, dropped.
Output: All outgoing packets can be: accepted, rejected, dropped.
Forward: All packets being sent to another device 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.

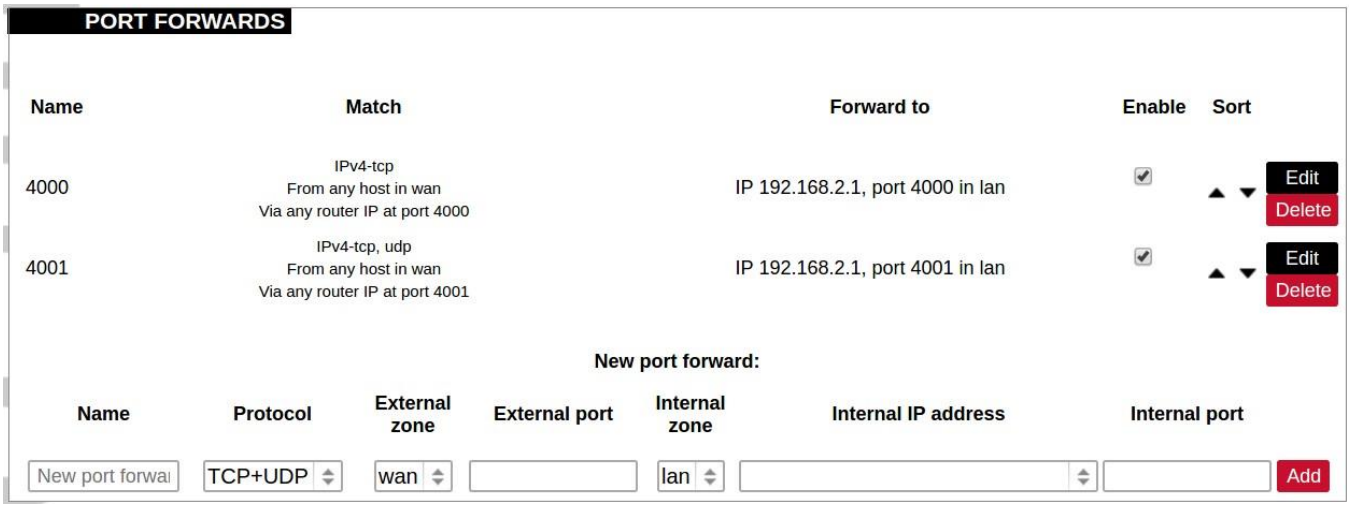



Advanced settings of new created or edited zone. Restrict to address family option defines to what IP families the zone belongs to IPv4, IPv6 or both. Restrict masquerading to given source/destination subnets defines one or more subnets for which the masquerading option is applied to. Connection tracking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



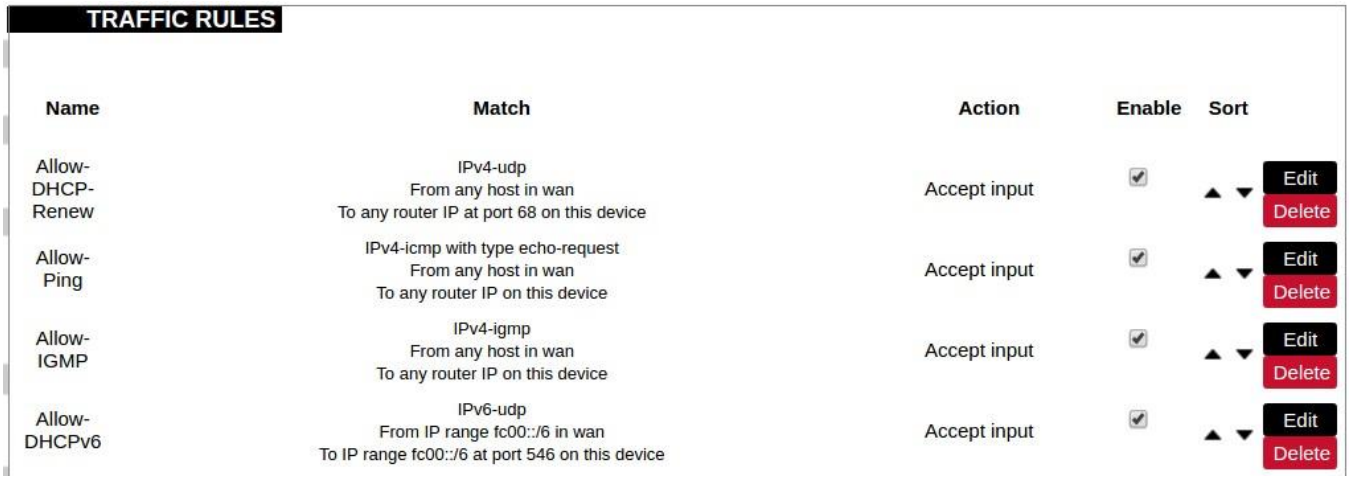

Controls of the forwarding policies between new/edited zone and other zones. Destination zones cover forwarded traffic originating from the new/edited zone. Source zones match forwarded traffic from other zones targeted at the new/edited zone. The forwarding rule is unidirectional, e.g. a forward from LAN to WAN does not imply a permission to forward from WAN to LAN as well.

## Port forwards



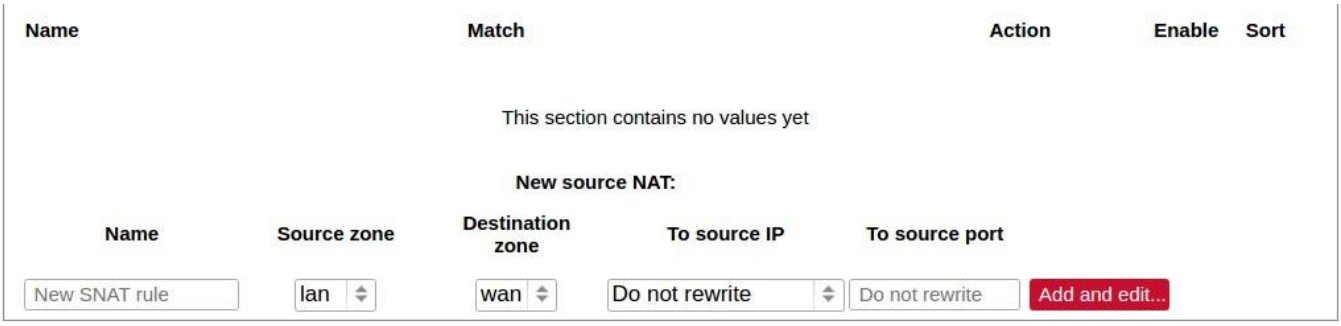

Port forwarding allows remote computers on the Internet to connect to a specific computer or service within the private LAN. It is done in a way of routing network packets within a private network created by the device. Settings for the port forwarding of the device are defined as follows:
Name: The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



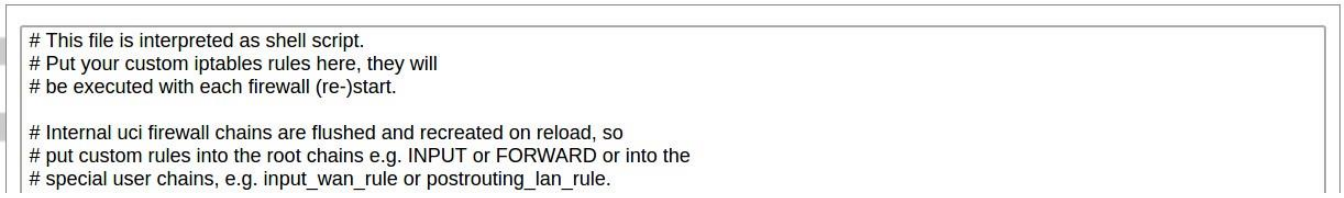

Traffic rules which define policies for packets traveling between different zones.
Name: The name of the traffic rule.
Match: Informs what ICMP types are matched.
Action: Informs what action would be performed.
Enable: Enable (checked) or disable the rule. Sort: Allows to sort rules.

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.



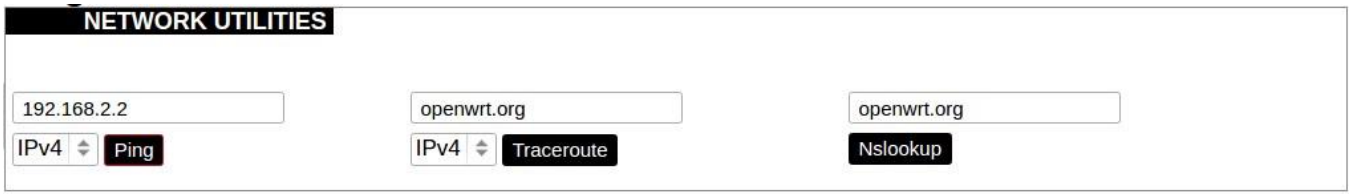

Source NAT, which is a specific form of masquerading which allows fine grained control over the source IP used for outgoing traffic, for the example to map multiple WAN addresses to internal subnets.
The user can add, edit or delete source NAT rules. For every rule can be defined these options: name, 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



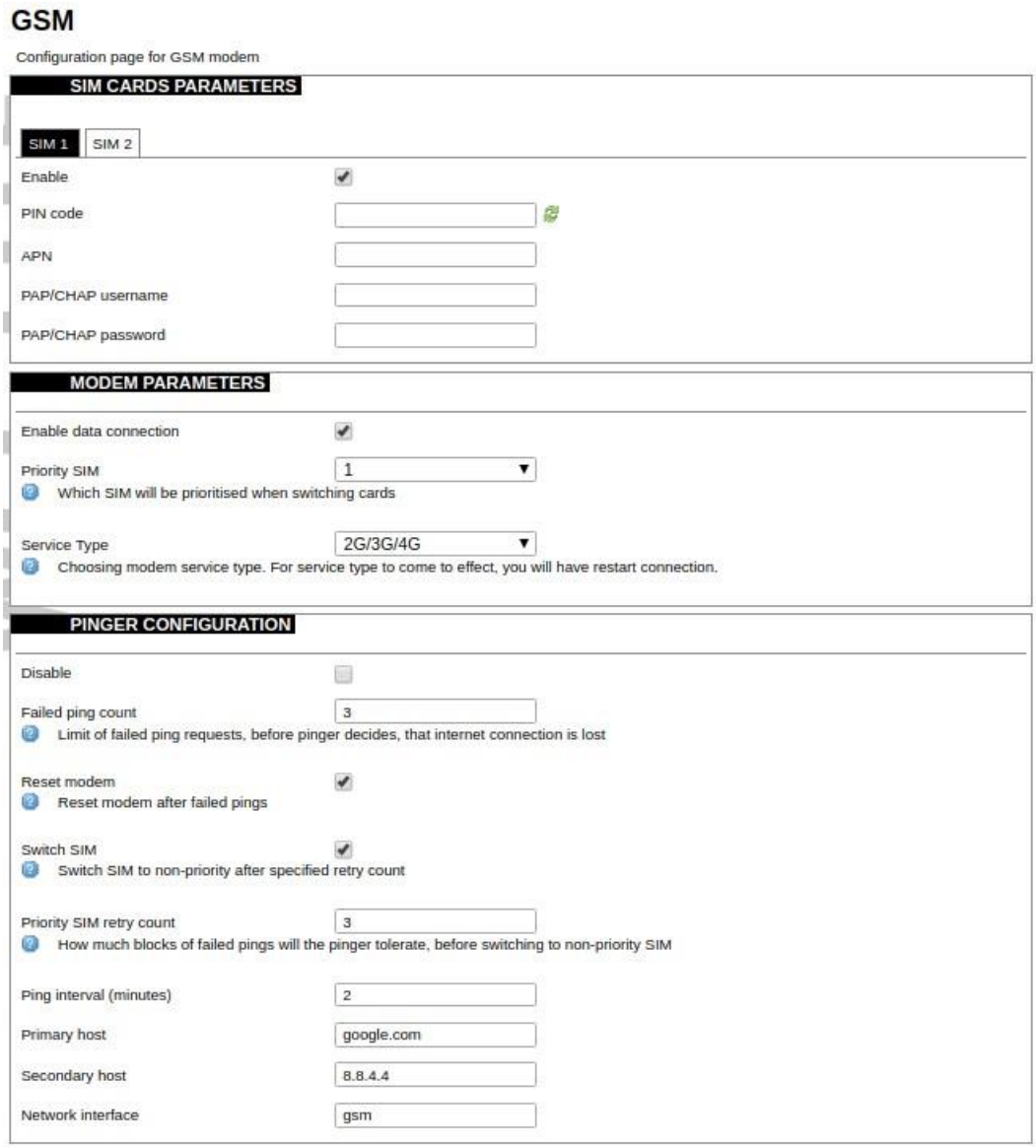

Custom rules allow to executing arbitrary iptables commands which are not otherwise covered by the firewall framework. The commands are executed after each firewall restart, right after the default ruleset has been loaded.

# Diagnostics




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

# 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 (optional). PAP/CHAP password: Password (optional).

## 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 the gsm network.

## Pinger configuration

Pinger is a service which pings defined hosts 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 many 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 the 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 the ”Status”->”GSM Status” window.

This is Pinger functionality described step by step:

- Pinger will ping the primary host every 2 minutes.
- If the primary host fails, pinger redirects to the 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 hosts 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) orwill 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 anon-priority SIM fails, it is switched to priority SIM in the next pinger action.

## Dual SIM start procedure

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

| 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 |

# 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.

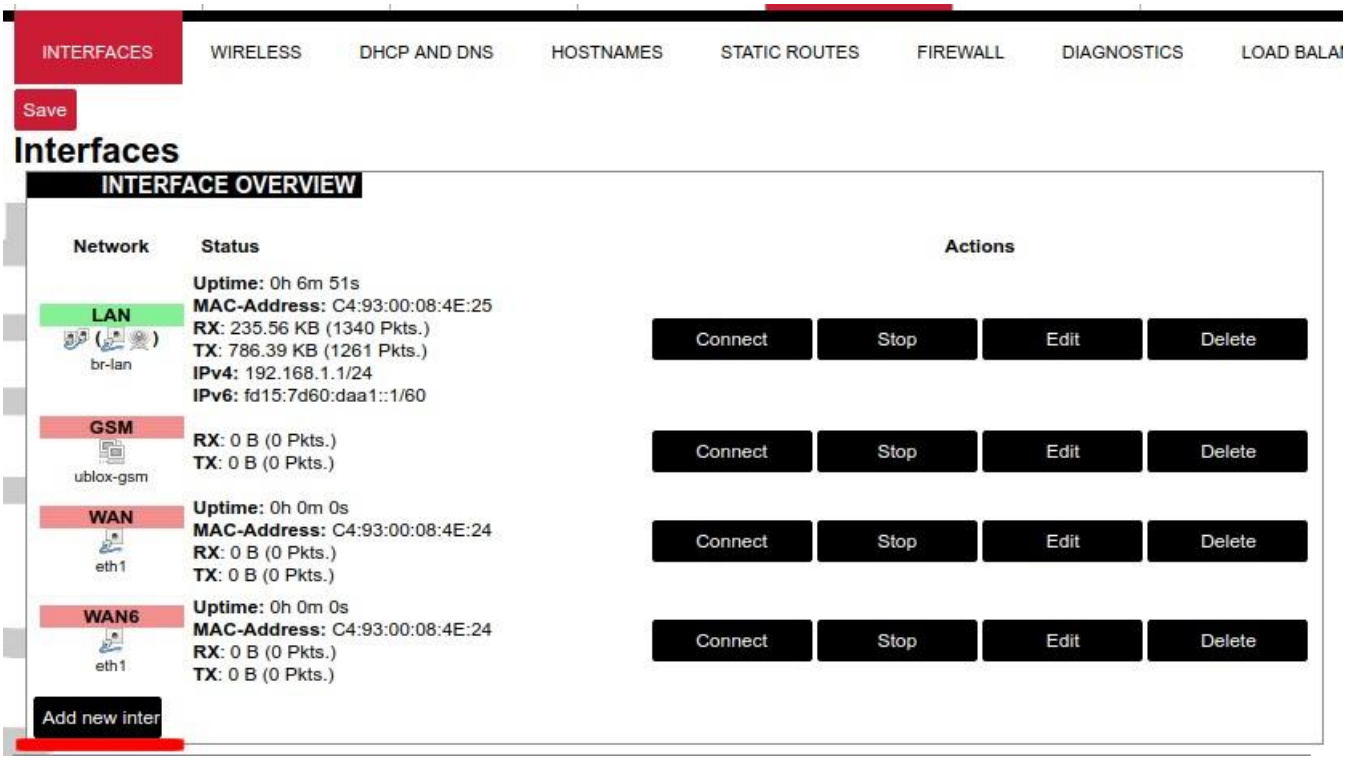
## Description

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.

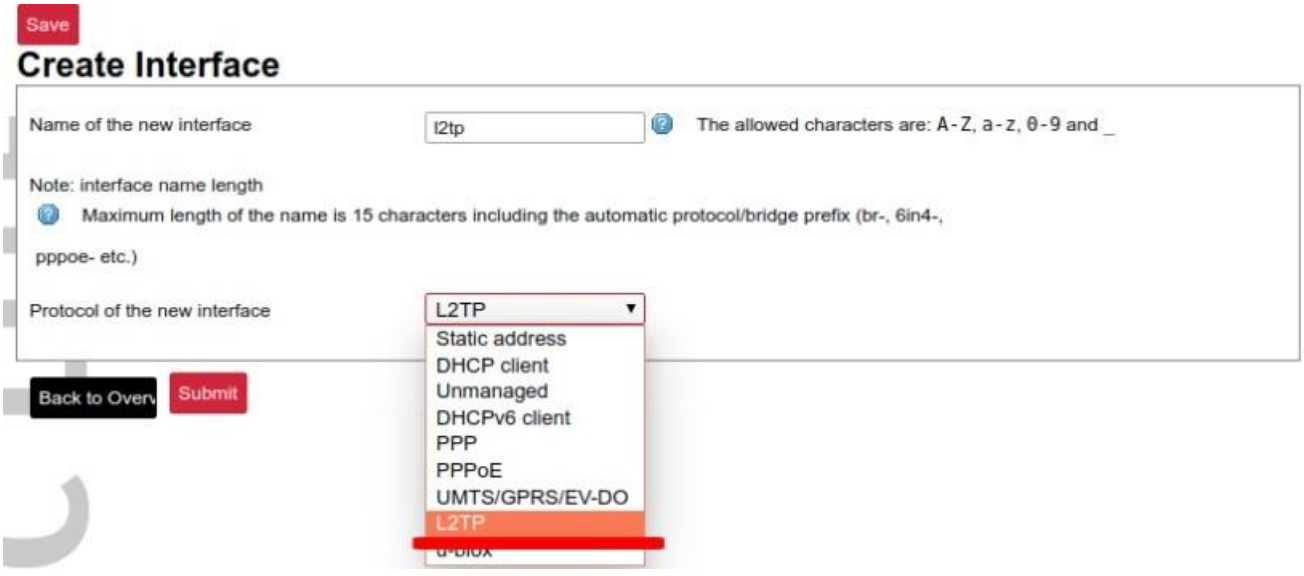
## Setting up L2TP interface

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

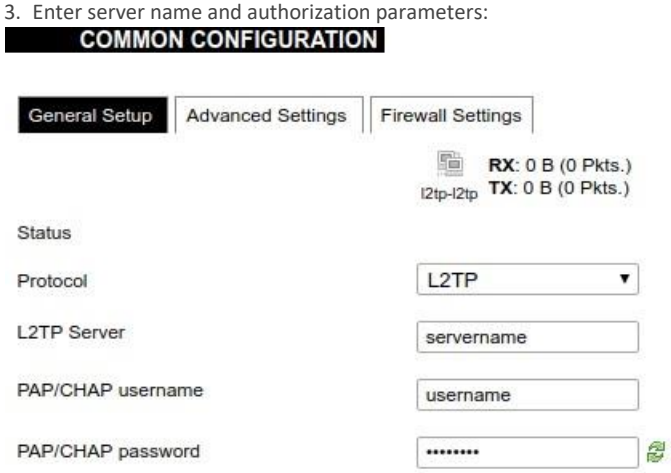
1. Go to **Network > Interfaces > Add new interface:**




2. Enter interface name and select L2TP protocol:



3. Enter server name and authorization parameters:




4. Save and apply the new configuration. A new network interface will appear.8 Internal web page

# 8.8 Logout




To log out of the device graphical user interface a logout button in the 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 builtin 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.

| 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/formdata”.

# 10 SNMP

SNMP (Simple Network Management Protocol) is an internetstandard 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 Unixstyle file system that describes a filesystem 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 highspeed 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 SMS sender

## General

SMS sender is a service that lets user configure WCC Lite to send SMS on a set tag triggers

SMS sender functionality is available since firmware version v1.5.4, of WCC Lite.

## Configuring SMS sender

To configure WCC Lite to use SMS sender user must fill in the needed parameters in Excel configuration. These parameters are shown in the tables below.

SMS sender parameters for Devices tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | **SMS sender** | |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| host | string | List of phone numbers to send SMS to, separated by space. | Yes | | | | | | | | |

| name | description | device_alias | enable | protocol | host | id | scan_rate_ms | poll_delay_ms | timeout_ms | port | ip |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sample Device | Modbus tcp | Sample_device | 1 | Modbus TCP | | 1 | 2000 | 200 | 1000 | 502 | 192.168.1.2 |
| SMS Sender | Service to send SMS | SMS_Sender | 1 | SMS sender | 860123456 +37060123456 | | | | | | |

SMS sender parameters for Signals tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| 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 the source device | No | | | |
| source_signal_alias | string | source_alias of the source signal | No | | | |
| enable | boolean | Enabling/disabling of a signal | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | 0 | |
| job_todo | string | Specific SMS sender tag type | Yes | | send-sms, device-control, device-status | |
| tag_job_todo | string | SMS sender tag for **send-sms**: text message | Yes | | | |
| trigger | string | Trigger expression for the SMS to be sent | No (Only for send_sms) | value!=0 | | |

To configure SMS sender, 3 types of signals are mandatory. These values should be written inside the job_todo field for each signal:

- **send-sms** - This signal takes value from the provided source_signal_alias field and checks if the value evaluates as true against the trigger field. If its true, the SMS sender will send the text from the
- tag_job_todo field to the specified phone numbers.

**device-control** - This signal controls if the SMS sender is enabled or disabled. Its tag_job_todo parameter should be set to enable. It takes value from the source_signal_alias field and evaluates it against the

trigger field. If the trigger is evaluated as true, the SMS sender will be enabled, otherwise it will disable the SMS sender. • **device-status** - This signal indicates if the service is enabled or not. Its tag_job_todo parameter should be set to enabled.

Trigger expressions can be configured with basic comparison operators:

- Less than **<**
- Greater than **>**
- Less than or equal to **<=**
- Greater than or equal to **>=**
- Equal **==**

Not equal **!=**

Example configuration of SMS sender:

| signal_name | device_alias | signal_alias | source_device_alias | source_signal_alias | enable | tag_type | job_todo | tag_job_todo | trigger |
|---|---|---|---|---|---|---|---|---|---|
| Sample measurement | Sample_device | Sample_data | | | 1 | Normal | 03\|00\|00\|00\|01 | 03\|00\|00\|00\|01 | |
| SMS sender enable/disable command | Sample_device | sms_switch | | | 1 | Normal | 02\|00\|00\|00\|01 | 02\|00\|00\|00\|01 | |
| SMS send | SMS_Sender | sms_send | Sample_device | Sample_data | 1 | Normal | send-sms | SMS text | value>100 |
| SMS switch | SMS_Sender | sms_control | Sample_device | sms_switch | 1 | Normal | device-control | enable | value!=0 |
| SMS status | SMS_Sender | sms_status | | | 1 | Normal | device-status | enabled | |

# 12 DNP 3.0



# 12.1 Introduction

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 InterControl Center Communications Protocol (a part of IEC-608706), is used for intermaster station communications.

Elseta's DNP3 stack has both Master and Slave protocols implemented. Both of them are able to serve multiple serial (over physical RS485 line), TCP or TLS (over TCP) connections with high efficiency.

IEEE1815 defines 4 subset levels (14) 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.

# 12.2 DNP 3.0 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.

## Configuring datapoints

To use DNP3 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 and Signals.

### DNP3 Master parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used (”dnp3 serial”/”dnp3 tcp” (case insensitive)) | Yes | | | |
| mode | string | Choosing between TCP, TLS and SERIAL modes . If protocol provided DNP3 TCP mode defaults to tcp and if DNP3 serial is provided mode defaults to serial | No | TCP or SERIAL | TCP, SERIAL | |
| ip | string | IP address of TCP slave device | Yes (for TCP). | | | |
| bind_address | string | IP address of network adapter used to connect to slave device | No (for TCP) | 0.0.0.0 | | |
| port | integer | TCP communication port | No (for TCP) | 20000 | | |
| device | integer | Communication port (”PORT1” or ”PORT2”) | Yes (for SERIAL) | | | |
| baudrate | integer | Communication speed, bauds/s | No (for SERIAL) | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| databits | integer | Data bit count for communication | No (for SERIAL) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No (for SERIAL) | 1 | 0 | 2 |
| parity | string | Communication parity option | No (for SERIAL) | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. | No (for SERIAL) | none | none | |
| tls | boolean | Enable/disable use of TLS | Yes (for TLS) | 0 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| tls_local_certificate | string | Local certificate for TLS connection | Yes (for TLS) | | | |
| tls_peer_certificate | string | Certificate authority file for TLS connection | No (for TLS) | | | |
| tls_private_key | string | File consisting of private key for TLS connection | No (for TLS) | | | |
| max_rx_frag_size | integer | Maximum size of a received fragment. | No | 2048 | 0 | 2048 |
| destination_address | integer | Address of a master station | No | 1 | 0 | 65535 |
| source_address | integer | Address of a slave (local) station. | No | 1 | 0 | 65535 |
| unsol_disable | bool | Disables unsolicited messages on startup. Overrides **unsol_classes** parameter. | No | 0 | 0 | 1 |
| unsol_classes | string | Defines which classes will have unsolicited actions on startup. Can be overridden with **unsol_disable**. (Example: "1,3,2") | No | no class | 1 | 3 |
| groups_scan_mask | integer | Bitmask for enabling separate group scans with x06 qualifier (all objects):<br>0 - Binary,<br>1 - Doublebit Binary,<br>2 - Binary Output Status,<br>3 - Counter,<br>4 - Frozen Counter,<br>5 - Analog,<br>6 - Analog Output Status,<br>7 - Octet String | No | 0 | 0 | 7 |
| groups_scan_interval | integer, string | Time between separate groups scans intervals in seconds. Set to 0 to disable. | No | 0 | 0 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| exception_scan_interval | integer, string | Time between exception scan (classes 1,2,3) intervals in seconds. Set to 0 to disable. | No | 0 | 0 | |
| integrity_scan_interval | integer, string | Time between integrity scan (classes 0,1,2,3) intervals in seconds (general interrogation). Set to 0 to disable. | No | 0 | 0 | |
| timesync_mode | string | Will override masters default setting for choosing timesync procedure | No | NON-LAN(for Serial)<br>LAN (for tcp) | LAN, NON-LAN | |
| time_sync_interval_sec | integer, string | Periodic time sync interval in seconds.<br>If 0 < - time syncs are forced and periodic.<br>If = 0 - time syncs react to IIN bits from slave.<br>If < 0 - time syncs are disabled. | No | 0 | 0 | |
| select_ms | integer | Select command timeout. Valid for all signals. | No | 10000 | | |
| timeout_ms | integer | Response timeout in milliseconds | No | 2000 | | |
| keep_alive_timeout | integer | Time interval for sending a keep alive packet in milliseconds. | No | 60 | | |

## DNP3 Master parameters for Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |
| signal_alias | | Unique alphanumeric name of the signal to be used | Yes | | | |
| enable | boolean | Enabling/disabling a device | No | 1 | 0 | 1 |
| index | integer | Index of a signal. | Yes | | 0 | 65535 |

| | | | | | | |
|---|---|---|---|---|---|---|
| log | boolean | Enable logging in the event log | No | 0 | 0 | |
| signal_type | string | DNP3 signal type. (case insensitive) | Yes | | ”BINARY”,<br>”ANALOG”,<br>”DOUBLEBITBINARY”<br>”BINARYOUTPUTSTATUS”,<br>”COUNTER”,<br>”FROZENCOUNTER”,<br>”ANALOGOUTPUTSTATUS,<br>”OCTETSTRING”,<br>”TIMEANDINTERVAL”,<br>”BINARYOUTPUTCOMMAND”,<br>”ANALOGOUTPUTCOMMAND” | |
| command_variation | integer | DNP3 command variation. Supported variations depend on signal type and are provided in the table below | No | 0 | 0 | 4 |
| static_variation | integer | DNP3 command variation (). Supported variations depend on signal type and are provided in the table below. | No | | 0, 1, 2, 3, 4, 5, 6, 9, 10 | |
| event_variation | integer | DNP3 command variation. Supported variations depend on signal type and are provided in the table below. | No | | 0 | 8 |
| control_code | integer | DNP3 control model code of CROB signal. TripClose and Pulse control model requires **PulseOn/off** times to be set | No | | LATCH, PULSE, TRIPCLOSE | |
| pulse_on_time_ms | integer | Pulse ON time in milliseconds, when using Pulse or TripClose control models must be set | No | | | |
| pulse_off_time_ms | integer | Pulse OFF time in milliseconds, when using Pulse or TripClose control models must be set | No | | | |
| class_num | integer | Class assignment of this signal. | No | 0 | 0 | 3 |
| operate_type | integer | | No | 1 | -1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| job_todo | string | The device status signal can be configured by providing one of the given values. | No | | COMMUNICATION_STATUS,<br>DEVICE_RUNNING,<br>DEVICE_ERROR,<br>UNKNOWN_ERROR |

## Debugging the DNP3 Master service

If configuration for DNP3 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.

DNP3 protocol runs a service called **dnp3-master**. If DNP3 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 **dnp3-master** process and run **dnp3master** command with respective flags as in the table given below.

Procedure for DNP3 Master protocol service debugging:

- **Step 1**: Service must be stopped by entering the following command into the wcclite: **/etc/init.d/dnp3-master stop**
- **Step 2**: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **dnp3-master -c /etc/dnp3-master/dnp3master.json -d7**Additional output forming options described in the table below.
  **Step 3**: Once the problem is diagnosed normal operations can be resumed with the following command: **/etc/init.d/dnp3-master start**
-

dnp3-master command line debugging options

| Option | Description |
|---|---|
| -h [ –help ] | Display help information |
| -V [ –version ] | Show version |
| -p [ –port ] | Show output for one port only |
| -d <debug level> | Set debugging level |
| -c [ –config ] | Config path |
| -a [ –app ] | Show application layer data |
| –l [ –link ] | Show link layer data |
| –t [ –transport ] | Show transport layer data |
| -r [ –redis ] | Show Redis messages |
| -R [ –readyfile ] | Ready notification file |

# 12.3 DNP 3.0 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. Wcc-Lite supported variations are provided in Static and Event variations and Command variations.

DNP3 Slave parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | dnp3 tcp slave dnp3 serial slave | |
| mode | string | Choosing between TCP, TLS and SERIAL modes. If protocol provided DNP3 TCP mode defaults to tcp and if DNP3 serial is provided mode defaults to SERIAL | No | TCP or SERIAL | TCP, SERIAL, TLS | |
| host | string | IP address of TCP slave device | Yes (for TCP). | | | |
| bind_address | string | IP address of network adapter used to connect to slave device | No (for TCP) | 0.0.0.0 | | |
| port | integer | TCP communication port | No (for TCP) | 20000 | | |
| device | integer | Communication port (”PORT1” or ”PORT2”) | Yes (for SERIAL) | | | |
| baudrate | integer | Communication speed, bauds/s | No (for SERIAL) | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| databits | integer | Data bit count for communication | No (for SERIAL) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No (for SERIAL) | 1 | 0 | 2 |
| parity | string | Communication parity option | No (for SERIAL) | none | none, even, odd | |

| | | | | | | |
|---|---|---|---|---|---|---|
| flowcontrol | string | Communication device flow control option. | No (for SERIAL) | none | none | |
| tls | boolean | Enable/disable use of TLS | Yes (for TLS) | 0 | 0 | 1 |
| tls_local_certificate | string | Local certificate for TLS connection | Yes (for TLS) | | | |
| tls_peer_certificate | string | Certificate authority file for TLS connection | No (for TLS) | | | |
| tls_private_key | string | File consisting of private key for TLS connection | No (for TLS) | | | |
| max_tx_frag_size | integer | Maximum size of a received fragment. | No | 2048 | 0 | 2048 |
| destination_address | integer | Address of a master station | No | 1 | 0 | 65535 |
| source_address | integer | Address of a slave (local) station. | No | 1 | 0 | 65535 |
| unsol_classes | string | Defines which classes will have unsolicited actions on startup. Can be overriden with **unsol_disable**. (Example: "1,3,2") | No | no class | 1 | 3 |
| time_sync_interval_sec | integer, string | Periodic time sync interval in seconds. If 0 < - time syncs are forced and periodic. If = 0 - time syncs react to IIN bits from slave. If < 0 - time syncs are disabled. | No | 0 | 0 | |
| select_ms | integer | Select command timeout. Valid for all signals. | No | 10000 | | |
| timeout_ms | integer | Response timeout in milliseconds | No | 2000 | | |
| keep_alive_timeout | integer | Time interval for sending a keep alive packet in milliseconds. | No | 60 | | |

## DNP3 Slave parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| signal_alias | | Unique alphanumeric name of the signal to be used | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| index | integer | Index of a signal. | Yes | | 0 | 65535 |
| log | boolean | Enable logging in event log | No | 0 | 0 | |
| deadband | double | Deadband for Analog, Analog Output Status, Counter, Frozen Counter signals. | No | 0 | | |
| signal_type | string | DNP3 signal type. (case insensitive) | Yes | | ”BINARY”, ”ANALOG”, ”DOUBLEBITBINARY” ”BINARYOUTPUTSTATUS”, ”COUNTER”, ”FROZENCOUNTER”, ”ANALOGOUTPUTSTATUS, ”OCTETSTRING”, ”TIMEANDINTERVAL”, ”BINARYOUTPUTCOMMAND”, ”ANALOGOUTPUTCOMMAND” | |
| command_variation | integer | DNP3 command variation. Supported variations depend on signal type and are provided in table below | No | 0 | 0 | 4 |
| static_variation | integer | Override default signal’s static variation. Valid for Status mode signals. | No | | 0, 1, 2, 3, 4, 5, 6, 9, 10 | |
| event_variation | integer | Override default signal’s event variation. Valid for Status mode signals. | No | | 0 | 8 |
| control_code | integer | DNP3 control model code of CROB signal. TripClose and Pulse control model requires **PulseOn/off** times to be set | No | | LATCH, PULSE, TRIPCLOSE | |
| pulse_on_time_ms | integer | Pulse ON time in milliseconds, when using Pulse or TripClose control models must be set | No | | | |
| pulse_off_time_ms | integer | Pulse OFF time in milliseconds, when using Pulse or TripClose control models must be set | No | | | |
| class_num | integer | Class assignment of this signal. | No | 0 | 0 | 3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| operate_type | integer | Default command behavior. IF selected ”**-1**” - DirectOperateNoAck, **”0”** - DirectOperate, "**1**" -- SelectBeforeOperate. | No | 1 | -1 | 1 |
| job_todo | string | Device status signal can be configured by providing one of the given values. | No | | COMMUNICATION_STATUS, DEVICE_RUNNING, DEVICE_ERROR, UNKNOWN_ERROR | |

## Command variations

| Signal Type | Available Command Variation | Default Command Variation |
|---|---|---|
| Binary Output Command (Group12) | 0, 1 | 1 |
| Analog Output Command (Group41) | 0, 1, 2, 3, 4 | 1 |

## Static and Event variations

| Signal Type | Available Variations | Default Variations |
|---|---|---|
| Binary | Static variation (Group1) 1, 2<br>Event variation (Group2) 1, 2, 3 | Static variation 2 Event variation 1 |
| Double Binary | Static variation (Group3) 1, 2<br>Event variation (Group4) 1, 2, 3 | Static variation 2 Event variation 1 |
| Binary Output Status | Static variation (Group10) 2<br>Event variation (Group11) 1, 2 | Static variation 2 Event variation 1 |
| Counter | Static variation (Group20) 1, 2, 5, 6<br>Event variation (Group22) 1, 2, 5, 6 | Static variation 1 Event variation 1 |
| Frozen Counter | Static variations (Group21) 1, 2, 5, 6, 9,10 Event variation (Group23) 1, 2, 5, 6 | Static variation 1 Event variation 1 |
| Analog | Static variation (Group30) 1, 2, 3, 4, 5, 6<br>Event variation (Group32) 1, 2, 3, 4, 5, 6, 7, 8 | Static variation 1 Event variation 1 |
| Analog Output Status | Static variation (Group40) 1, 2, 3, 4<br>Event variation (Group42) 1, 2, 3, 4, 5, 6, 7, 8 | Static variation 1 Event variation 1 |
| Time and Interval | Static variation (Group50) 1 | Static variation 1 |
| Octet String | Static variation (Group110) 0<br>Event variation (Group111) 0 | Static variation 0 Event variation 0 |

# Debugging the DNP3 Slave service

If configuration for DNP3 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.

DNP3 protocol runs a service called **dnp3-slave** . If DNP3 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 **dnp3-slave** process and run **dnp3slave** command with respective flags as in the table given below.

Procedure for DNP3 Master protocol service debugging:

- **Step 1**: Service must be stopped by entering the following command into the wcclite: **/etc/init.d/dnp3-slave stop**
- **Step 2**: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **dnp3-slave -c /etc/dnp3-slave/dnp3slave.json -d7** Additional output forming options described in the table below.
  **Step 3**: Once the problem is diagnosed normal operations can be resumed with the following command: **/etc/init.d/dnp3-slave start**
-

dnp3-slave command line debugging options

| Option | Description |
|---|---|
| -h [ –help ] | Display help information |
| -V [ –version ] | Show version |
| -p [ –port ] | Show output for one port only |
| -d <debug level> | Set debugging level |
| -c [ –config ] | Config path |
| -a [ –app ] | Show application layer data |
| –l [ –link ] | Show link layer data |
| –t [ –transport ] | Show transport layer data |
| -r [ –redis ] | Show Redis messages |
| -R [ –readyfile ] | Ready notification file |

# 13 Modbus13 Modbus

# 13.1 Introduction

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/RS232). Bytes to transmit can either be encoded according to both RTU and ASCII parts of standard.

# 13.2 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.

## 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 and Signals

### Modbus Master parameters for Devices tab

| **Parameter** | **Type** | **Description** | **Required** | **Default Value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | Modbus RTU, Modbus TCP | |
| ip | string | IP address of TCP slave device | Yes (for TCP). | | | |
| port | integer | TCP communication port | No (for TCP) | 502 | | |
| bind_address | string | IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”) | No (for TCP) | 0.0.0.0 | | |
| id | integer | Modbus Slave ID | Yes | | | |
| mode | string | Choosing between RTU (”rtu”), ASCII (”ascii”) and TCP(”tcp”) modes. ASCII is the same as RTU, but with ASCII symbols. | No | TCP (for TCP) RTU (for Serial) | rtu, ascii, tcp | |

| | | | | | | |
|---|---|---|---|---|---|---|
| timeout_ms | integer | Response timeout in milliseconds | No | 10000 | | |
| device | string | Communication port (”PORT1”/”PORT2 ”) | Yes (for RTU/ASCII) | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, baud/s | No (for RTU/ASCII) | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No (for RTU/ASCII) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No (for RTU/ASCII) | 1 | 1 | 2 |
| parity | string | Communication parity option | No (for RTU/ASCII) | none | none, even, odd | |
| flowcontrol | string | Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued | No (for RTU/ASCII) | none | none | |
| 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) | No | 300 | | |
| retry_count | integer | Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued | No | 3 | | |
| serial_delay | integer | RS485 delay between read and write operations in milliseconds | No (for RTU/ASCII) | 50 | | |
| keep_alive_timeout | integer | Time interval for sending a keep alive packet (in milliseconds) | No (for TCP) | 60 | | |
| modbus_multi_write | boolean | Use 15/16 functions to write 1 register/coil (Default: 0) | No | 0 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| comm_restart_delay | integer | Time delay between disconnecting from slave device and restarting connection (in milliseconds) (Default: 500) | No (for TCP) | 500 | | |
| update | boolean | Enable to keep updating the tags even if they have the same value. | No | 0 | 0 | 1 |

## Modbus Master parameters for Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default Value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes used | Yes | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| 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 | integer | Size of this signal's log in Event log. | No | 0 | | |
| 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 .

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.

## 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.

Modbus Master command line debugging options

modbus-master

```
-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 shown above.

# 13.3 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.

## 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 and Signals

⚠ 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

Modbus Slave parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | Modbus serial Slave, Modbus TCP Slave | |
| host | string | Space separated host IP addresses of master device | Yes (for TCP). | | | |
| port | integer | TCP port to listen for incoming connections | Yes (for TCP) | | | |
| bind_address | string | IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”) | No (for TCP) | 0.0.0.0 | | |
| keep_alive_timeout | integer | Minimum time a connection can be idle without being closed in miliseconds | No (for TCP) | 60 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| mode | string | Choosing between RTU (”rtu”), ASCII (”ascii”) and TCP(”tcp”) modes. ASCII is the same as RTU, but with ASCII symbols. | No | TCP (for TCP) RTU (for Serial) | rtu, ascii, tcp | |
| device | string | Communication port (”PORT1”/”PORT2 ”) | Yes (for serial) | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, baud/s | No (for serial) | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No (for serial) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No for serial) | 1 | 1 | 2 |
| parity | string | Communication parity option | No (for serial) | none | none, even, odd | |
| flowcontrol | string | Communication device’s flow control option. | No (for serial) | none | none | |

## Modbus Slave parameters for Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes used | Yes | | | |
| enable | boolean | Enabling/disabling an individual signal | No | 1 | 0 | 1 |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.). This defines the size that will be read. | Yes | | | |
| log | integer | Size of this signal’s log in the Event log. | No | 0 | | |
| slave_id | integer | Address of a slave device | Yes | | | |
| function | integer | Function number | Yes | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| register_address | integer | Register address | Yes | | | |

Modbus Slave has an additional signal which can be configured to show communication status. It is used to indicate if the slave device has disconnected from the master or vice versa. To configure such a signal, one column should be filled with particular value. To create an additional signal, one should make a tag_job_todo column equal to device_status or communication_status. Communication error status is set when a predefined count of messages (three by default, defined in keep_alive_timeout column) fail to be received or are considered invalid.

## 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 (18.2.4).

> ⚠ 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.

## 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.

Modbus Slave command line debugging options
modbus-slave

```
-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.

> ℹ To launch a debugging session, a user should stop modbus-slave process and run modbus-slave command with respective flags as shown above.

# 14 IEC 60870-5-10X



# 14.1 Introduction

**IEC 60870 part 5** is one of the IEC 60870 set of standards which define systems used for telecontrol (supervisory control and data acquisition) in electrical engineering and power system automation applications. Part 5 provides a communication profile for sending basic telecontrol messages between two systems, which uses permanent directly connected data circuits between the systems. The IEC Technical Committee 57 (Working Group 03) has developed a protocol standard for telecontrol, teleprotection, and associated telecommunications for electric power systems. The result of this work is IEC 60870-5. Five documents specify the base IEC 60870-5:

- IEC 60870-5-1 Transmission Frame Formats
- IEC 60870-5-2 Data Link Transmission Services
- IEC 60870-5-3 General Structure of Application Data
- IEC 60870-5-4 Definition and Coding of Information Elements
- IEC 60870-5-5 Basic Application Functions
- IEC 60870-5-6 Guidelines for conformance testing for the IEC 60870-5 companion standards

IEC TS 60870-5-7 Security extensions to IEC 60870-5-101 and IEC 60870-5-104 protocols (applying IEC 62351)

The IEC Technical Committee 57 has also generated companion standards:

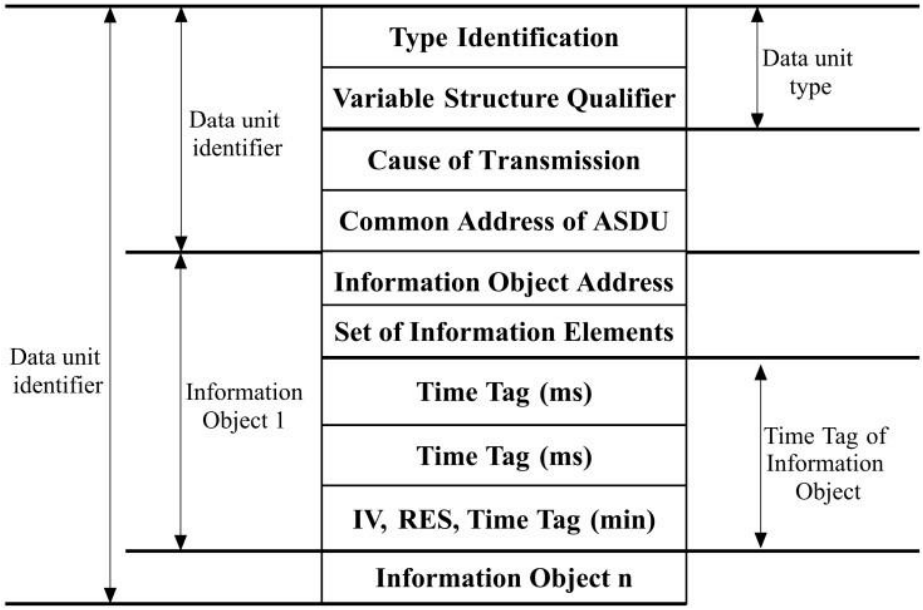
- IEC 60870-5-101 Transmission Protocols - companion standards especially for basic telecontrol tasks IEC 60870-5-102
- Transmission Protocols - Companion standard for the transmission of integrated totals in electric power systems (this standard is not widely used)
- IEC 60870-5-103 Transmission Protocols - Companion standard for the informative interface of protection equipment
- IEC 60870-5-104 Transmission Protocols - Network access for IEC 60870-5-101 using standard transport profiles
- IEC TS 60870-5-601 Transmission protocols - Conformance test cases for the IEC 60870-5-101 companion standard
- IEC TS 60870-5-604 Conformance test cases for the IEC 60870-5-104 companion standard
- IEC 60870-5-101/102/103/104 are companion standards generated for basic telecontrol tasks, transmission of integrated totals, data exchange from protection equipment & network access of IEC101 respectively.

Source: https://en.wikipedia.org/wiki/IEC_60870-5

# 14.2 IEC 60870-5-101 Master

The IEC 60870-5-101 protocol is a companion standard for power system monitoring, control associated communications for telecontrol, teleprotection, and associated telecommunications for electric power systems. Standard IEC 60870-5-101 was prepared by IEC technical committee 57 (Power system control and associated communications).

Standard IEC 60870-5-101 defines an **Application Service Data Unit** (**ASDU** Figure below). In ASDU there is an ASDU identifier(with the type of ASDU in it) and information objects.

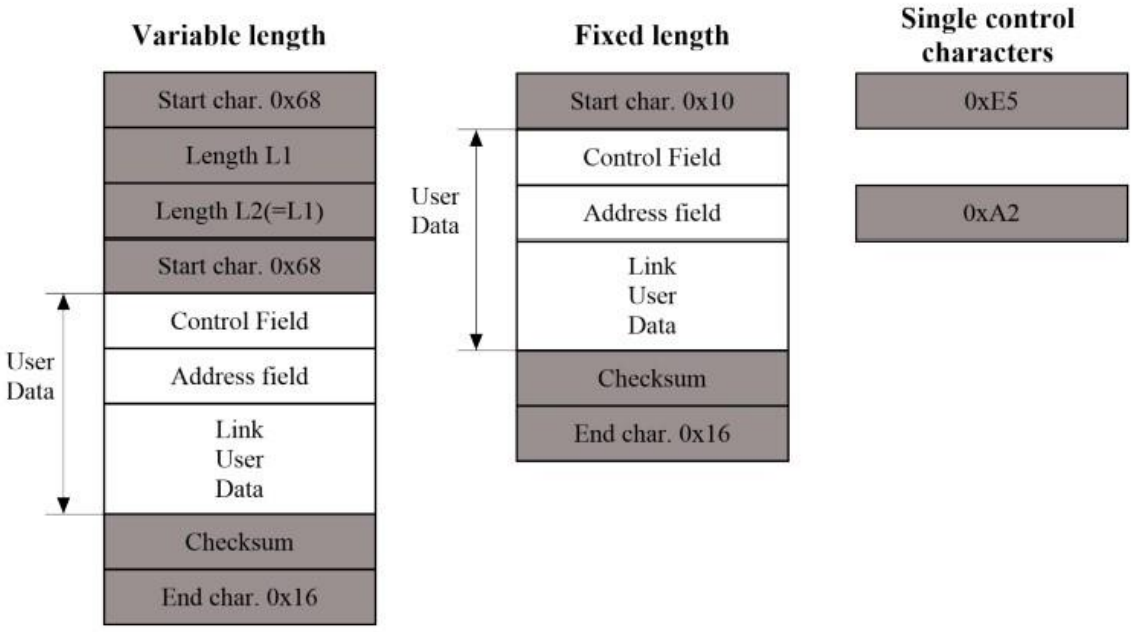



IEC 60870-5-101 ASDU structure

**Common Address of ASDU** Defines the stations' address and can be configured in Devices asdu_address field for source and Signals common_address field for the destination.

**Information Object Address** Used as destination object address in the control direction and as source object address in monitor direction can be configured in Signals info_address field.

Standard IEC 60870-5-101 transmission frames are separated into 3 different types: **frame with variable length**, **frame with fixed length,** and **single control characters**



**Control field** provides information about the message direction, type of service, and checksum.
**Address field** specifies the link address which points to the message's destination. WCC Lite supports IEC 60870-5-101 Master protocol over a serial link (according to EIA RS485). Its full functionality list can be found in an IEC 608705-101 PID Interoperability List which can be downloaded separately from this user manual.

# Configuring datapoints (master)

To use IEC 60870-5-101 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 and Signals.

## IEC 60870-5-101 master parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 60870-5-101 master | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, baud/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | | No | none | none | |
| link_address | integer | Destination address when in transmit and source address when broadcasting | Yes | | 0 | 65535 |
| link_size | integer | Link address size in bytes | No | 1 | 1 | 2 |
| asdu_address | integer | Application Service Data Unit address | Yes | | 0 | 65535 |
| asdu_size | integer | Common address size in bytes | No | 1 | 1 | 3 |
| ioa_size | integer | Information object address (IOA) size in bytes | No | 2 | 1 | 3 |
| cot_size | integer | Cause of transmission (COT) size in bytes | No | 1 | 1 | 2 |
| time_sync_interva l_sec | integer | Defines how often (in seconds) slave will request time synchronization. **If greater than 0** slave will request synchronizations, will reset the timer if the master did it earlier. **If 0** slave won't request timesyncs, but will allow them. **If 1** timesyncs are not supported requests will be dropped.  | No | 60 | | |
| gi_interval_sec | integer | Time frame between General Interrogation requests in seconds, if 0 requests are disabled | No | 300 | | |
| scan_rate_ms | integer | Polling interval in milliseconds. Time frame between two telegrams from master | No | 100 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| timeout_ms | integer | Response timeout in milliseconds | No | 1000 | | |
| retry_count | integer | Number of retries of failed requests before announcing that device is in Error state | No | 1 | | |

IEC 60870-5-101 master parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes used | Yes | | | |
| source_device_ali as | string | device_alias of a source device | For commands | | | |
| source_signal_alia s | string | signal_alias of a source signal | For commands | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | | |
| gi | boolean | Including/excludin g (1 or 0) signal from General Interrogation | No | 0 | 0 | 1 |
| common_address | integer | Address of a destination device | Yes | | | |
| info_address | integer | Information object address | Yes | | | |
| data_type | integer | ASDU type identificator | Yes | | | |

IEC 60870-5-101 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.

# Debugging an IEC 60870-5-101 Master application

If configuration for IEC 60870-5-101 devices is set up, handler for protocol will start automatically. If the configuration is missing or contains errors, protocol will not start. It is done intentionally decrease unnecessary memory usage.

If IEC 60870-5-101 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 iec101-master process and run iec101-master command with respective flags as shown in the table below.

Procedure for IEC 60870-5-101 master service debugging:

- **Step 1**: Service must be stopped by entering the following command into the wcclite: **/etc/init.d/iec101-master stop**
- **Step 2**: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **iec101-master -c /etc/iec101-master/iec101master.json -d7**
  Additional output forming options described here: Command line arguments.
- **Step 3**: Once the problem is diagnosed normal operations can be resumed with the following command: **/etc/init.d/iec101-master start**

IEC 60870-5-101 command line debugging options

```
-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
```



# 14.3 IEC 60870-5-101 Slave

## Configuring datapoints (slave)

To use IEC 60870-5-101 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 and Signals.

IEC 60870-5-101 slave parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 60870-5-101 slave | |

| | | | | | | |
|---|---|---|---|---|---|---|
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, baud/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued | No | none | none | |
| link_address | integer | Destination address when in transmit and source address when broadcasting | Yes | | 0 | 65535 |
| link_size | integer | Link address size in bytes | No | 1 | 1 | 2 |
| asdu_size | integer | Common address size in bytes | No | 1 | 1 | 3 |
| ioa_size | integer | Information object address (IOA) size in bytes | No | 2 | 1 | 3 |
| cot_size | integer | Cause of transmission (COT) size in bytes | No | 1 | 1 | 2 |
| time_sync | boolean | Allow time synchronization, 1 to enable and 0 to disable | No | 0 | 0 | 1 |
| message_size | integer | Maximum length of a message | Yes | | 0 | 255 |
| cache_size | integer | Maximum number of events to store in a buffer | No | 100 | 0 | 1000 |

| | | | | | | |
|---|---|---|---|---|---|---|
| respond_delay | integer | Time in microseconds to wait before sending responses | Yes | | 0 | 1000000 |
| single_byte_ack | boolean | Use single character acknowledge, 1 to enable and 0 to disable | No | 0 | 0 | 1 |
| keep_alive_timeout | integer | Time interval in seconds before serial connection is considered offline | No | 60 | | |

**keep_alive_timeout** timer is used for connection tracker to display protocol status. This parameter has no effect on protocol functionality and is only used to track it's status in connection tracker.

## IEC 60870-5-101 slave parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes 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 an individual signal | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | | |
| gi | boolean | Including/excluding (1 or 0) signal from General Interrogation | No | 0 | 0 | 1 |
| common_address | integer | Address of a destination device | Yes | | | |
| info_address | integer | Information object address | Yes | | | |
| data_type | integer | ASDU type identificator | Yes | | | |

IEC 608705101 has an additional signal which can be configured to show communication status. It is used to indicate if the slave device has disconnected from the master (WCC Lite). To configure such a 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.

# Debugging an IEC 60870-5-101 slave application

If the configuration for IEC 60870-5-101 devices is set up, the handler for the protocol will start automatically. If the configuration is missing or contains errors, the protocol will not start. It is done intentionally to decrease unnecessary memory usage.

If IEC 60870-5-101 does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from the command-line interface and find out why the link is not functioning properly. To launch a debugging session, a user should stop the iec101-slave process and run the iec101-slave command with respective flags as shown in the table below.

Procedure for IEC 60870-5-101 slave service debugging:

- **Step 1**: Service must be stopped by entering the following command into the wcclite: **/etc/init.d/iec101-slave stop**
- **Step 2**: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **iec101-slave-c /etc/iec101-slave/iec101slave.json -d7** Additional output forming options described here: Command line arguments.
- **Step 3**: Once the problem is diagnosed normal operations can be resumed with the following command: **/etc/init.d/iec101-slave start**

IEC 60870-5-101 command line debugging options

| |
|---|
| -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 |



# 14.4 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.

# IEC 60870-5-103 Master Configuring datapoints

WCC Lite supports IEC 60870-5-103 Master protocol over a serial link (according to EIA RS-485). Its full functionality list can be found in an IEC 60870-5-103 PID Interoperability List.

To use IEC 60870-5-103 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 and Signals.

IEC 60870-5-103 parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range |
|---|---|---|---|---|---|

| | | | | | Min | Max |
|---|---|---|---|---|---|---|
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 60870-5-103 master | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, baud/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 8 | |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued | No | none | none | |
| link_address | integer | Destination address when in transmit and source address when broadcasting | Yes | | 0 | 65535 |
| asdu_address | integer | Application Service Data Unit address | Yes | | 0 | 65535 |
| time_sync_interval_sec | integer | Time frame between Time Synchronization requests in seconds | No | 60 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| gi_interval_sec | integer | Time frame between General Interrogation requests in seconds, if 0 requests are disabled | No | 300 | | |
| scan_rate_ms | integer | Polling interval in milliseconds. Time frame between two telegrams from master | No | 100 | | |
| timeout_ms | integer | Response timeout in milliseconds | No | 1000 | | |
| serial_delay | integer | Communication device's serial delay in milliseconds. Time frame in which master station is not TX'ing after last RX byte | No | 50 | | |
| retry_count | integer | Number of retries of failed requests before announcing that device is in Error state | No | 3 | | |
| retry_delay_ms | integer | Time before the next retry in milliseconds | No | 500 | | |

## IEC 60870-5-103 master parameters for Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes used | Yes | | | |
| source_device_ali as | string | device_alias of a source device | For commands | | | |
| source_signal_alia s | string | signal_alias of a source signal | For commands | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| log | integer | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | | |
| gi | boolean | Including/excludin g (1 or 0) signal from General Interrogation | No | 0 | 0 | 1 |
| common_address | integer | Address of a destination device | Yes | | | |
| function | integer | Function number | No | 0 | | |
| info_address | integer | Information object address | Yes | | | |
| info_number | integer | Information number | Yes | | | |
| data_type | integer | ASDU type identificator | No | 0 | | |
| fleeting | boolean | Mark signal as fleeting type (1 or 0). Fleeting signals have go to DPI::OFF after defined time | No | | 0 | 1 |
| normalise_time_m s | integer | Time in milliseconds between station receiving DPI::ON and automatically switching to DPI::OFF | If fleeting is used | 100 | | |

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 the master (WCC Lite). To configure such a 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.

# Debugging an IEC 60870-5-103 Master application

If the configuration for IEC 60870-5-103 devices is set up, the handler for the protocol will start automatically. If a configuration is missing or contains errors, the protocol will not start. It is done intentionally to 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 the command-line interface and find out why the link is not functioning properly or use WCC Utility to do that.

To launch a debugging session, a user should stop the iec103-master process and run the iec103-master command with respective flags.

- Step 1: Service must be stopped by entering the following command into the wcclite:
- **/etc/init.d/iec103-master stop**
- Step 2: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **iec103-master -c /etc/iec103-master/iec103-master.json**
- **-d7**

Step 3: Once the problem is diagnosed normal operations can be resumed with the following command:
**/etc/init.d/iec103-master start**

## IEC 60870-5-103 command line debugging options

```
-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
```

# 14.5 IEC 60870-5-104 Master

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 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 the communication between the 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.

## Configuring IEC 104 Master datapoints

To use IEC 60870-5-104 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 and Signals.

### IEC 60870-5-104 Master parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 60870-5-104 master | |
| asdu_address | integer | Application Service Data Unit address | Yes | | 0 | 65535 |
| asdu_size | integer | Common address size in bytes | No | 2 | 1 | 3 |
| time_sync_interval_sec | integer | Time frame between Time Synchronization requests in seconds | No | 60 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| gi_interval_sec | integer | Time frame between General Interrogation requests in seconds. If 0 requests are disabled | No | 300 | | |
| port | integer | TCP port | Yes | | 0 | 65535 |
| ioa_size | integer | Information object address (IOA) size in bytes | No | 3 | 1 | 3 |
| swt | integer | Send window (SWT) | Yes | | | |
| rwt | integer | Receive window (RWT) | Yes | | | |
| cot_size | integer | Cause of transmission (COT) size in bytes | No | 2 | 1 | 2 |
| ip | string | Host IP address (ipv4) | Yes | | | |
| t1 | integer | Acknowledge timeout t1 (sec) | No | 15 | | |
| t2 | integer | Connection ACKRSN clock t2 (sec) | No | 10 | | |
| t3 | integer | Connection TESTFR clock t3 (sec) | No | 20 | | |
| originator | integer | Provides a means for a controlling station to explicitly identify itself | No | 0 | 0 | 255 |

## IEC 60870-5-104 Master parameters for Signals

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes used | Yes | | | |
| source_device_ali as | string | device_alias of a source device | For commands | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| source_signal_alia s | string | signal_alias of a source signal | For commands | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | | |
| gi | boolean | Including/excludin g (1 or 0) signal from General Interrogation | No | 0 | 0 | 1 |
| common_address | integer | Address of a destination device | Yes | | | |
| function | integer | Function number | No | 0 | | |
| info_address | integer | Information object address | Yes | | | |
| data_type | integer | ASDU type identificator | No | | | |
| 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. | No | 0 | | |

# Debugging an IEC 60870-5-104 Master application

If the configuration for IEC 60870-5-104 devices is set up, the handler for the protocol will start automatically. If a configuration is missing or contains errors, the protocol will not start. It is done intentionally to decrease unnecessary memory usage.

If IEC 60870-5-104 does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from the command-line interface and find out why the link is not functioning properly or use WCC Utility to do that.

To launch a debugging session, a user should stop the iec104-master process and run the iec104-master command with respective flags.

- Step 1: Service must be stopped by entering the following command into the wcclite:
- **/etc/init.d/iec104-master stop**
- Step 2: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **iec104-master -c /etc/iec104-master/iec104-master.json**
- **-d7**

Step 3: Once the problem is diagnosed normal operations can be resumed with the following command:
**/etc/init.d/iec104-master start**

## IEC 60870-5-104 command-line debugging options

```
-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
-e [ –redis ] Show redis message
-R [ –readyfile ] Ready notification file
```



# 14.6 IEC 60870-5-104 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 the 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 SCADA). If a new connection arrives, the old one is dropped, but data, that is stored in a cache, not sent and not confirmed by SCADA is transferred to the new connection.

## Configuring IEC 104 Slave datapoints

To use IEC 60870-5-104 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 and Signals.

### IEC 60870-5-104 Slave parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 60870-5-104 slave | |
| asdu_size | integer | Common address size in bytes | No | 2 | 1 | 2 |
| time_sync | boolean | Enable/disable (1 or 0) time synchronization | Yes | | | |
| port | integer | TCP port | No | 2404 | 0 | 65535 |
| ioa_size | integer | Information object address (IOA) size in bytes | No | 3 | 1 | 3 |
| swt | integer | Send window (SWT) | No | 12 | | |
| rwt | integer | Receive window (RWT) | No | 8 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| cot_size | integer | Cause of transmission (COT) size in bytes | No | 2 | 1 | 2 |
| host | string | Space-separated remote host IP addresses (ipv4) | Yes | | | |
| bind_address | string | Bind to local IP address (ipv4) | No | 0.0.0.0 | | |
| t1 | integer | Acknowledge timeout t1 (sec) | No | 15 | 1 | 255 |
| t2 | integer | Connection ACKRSN clock t2 (sec), t2 should be less than t1 | No | 10 | 1 | 254 |
| t3 | integer | Connection TESTFR clock t3 (sec) | No | 20 | 1 | 172800 |
| message_size | boolean | The maximum length of a message | Yes | | 0 | 255 |
| cache_size | integer | Amount of data to be cached | Yes | | 0 | 1000 |
| tls | boolean | Enable/disable the use of TLS | No | 0 | 0 | 1 |
| tls_local_certificate | string | Local certificate for TLS connection | Yes (for TLS) | | | |
| tls_peer_certificat e | string | Certificate authority file for TLS connection | No | | | |
| tls_private_key | string | A file consisting of the private key for TLS connection | No | | | |
| command_timeout_ms | integer | Time to execute a command before responding negatively. | No | 30000 | 0 | |
| command_age_ms | integer | The amount of time shift allowed for the command to still be executed. | No | 0 | 0 | |

## IEC 60870-5-104 Slave parameters for Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| signal_alias | string | Unique alphanumeric name of the signal to be Yes used | Yes | | | |
| source_device_ali as | string | device_alias of a source device | For commands | | | |
| source_signal_alia s | string | signal_alias of a source signal | For commands | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | 0 | 1 |
| gi | boolean | Including/excludin g (1 or 0) signal from General Interrogation | No | 0 | 0 | 1 |
| common_address | integer | Address of a destination device | Yes | | | |
| info_address | integer | Information object address | Yes | | | |
| data_type | integer | ASDU type id. | Yes | | | |
| 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. | No | 0 | | |

# Debugging an IEC 60870-5-104 Slave application

If the configuration for IEC 60870-5-104 devices is set up, the handler for the protocol will start automatically. If a configuration is missing or contains errors, the protocol will not start. It is done intentionally to decrease unnecessary memory usage.

If IEC 60870-5-104 does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from the command-line interface and find out why the link is not functioning properly or use WCC Utility to do that.

To launch a debugging session, a user should stop the iec104-slave process and run the iec104-slave command with respective flags.

- Step 1: Service must be stopped by entering the following command into the wcclite:
- **/etc/init.d/iec104-slave stop**
- Step 2: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/ folder) and a debug level 7: **iec104-slave-c /etc/iec104-slave/0_0_0_0_502.json -d7;** (0_0_0_0 - bind_address, 502 - port)

Step 3: Once the problem is diagnosed normal operations can be resumed with the following command:
**/etc/init.d/iec107-slave start**

## IEC 60870-5-10 command-line debugging options

```
-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
 - e [ –redis ] Show redis message
-R [ –readyfile ] Ready notification file
```

# 15 IEC 6185015 IEC 61850

# 15.1 Introduction

IEC 61850 is an international standard defining communication protocols for intelligent electronic devices at electrical substations. It is a part of the International Electrotechnical Commission's (IEC) Technical Committee 57 reference architecture for electric power systems. The abstract data models defined in IEC 61850 can be mapped to a number of protocols. Possible mappings in the standard can be MMS (Manufacturing Message Specification), GOOSE (Generic Object Oriented Substation Event), SMV (Sampled Measured Values). These protocols can run over TCP/IP networks or substation LANs using high speed switched Ethernet to obtain the necessary response times below four milliseconds for protective relaying.

As of version v1.5.0, WCC Lite supports MMS type messaging. Logging and groups setting services are not supported.

# 15.2 IEC 61850 Server

WCC Lite can act as a IEC 61850 server to serve data to remote SCADA systems. For example, WCC Lite can be used to acquire data from various protocols (Modbus, IEC 60870-5-103, etc.), this data can be redirected and propagated further to a single or multiple IEC 61850 clients. IEC 61850 Server supports TCP and TLS connection types. TCP connection can be secured with password authentication.

## Commands

WCC Lite **IEC 61850 Server** implementation defines four command types which are described by their control model:

- **Case 1**: Direct control with normal security (direct-operate);
- **Case 2**: SBO control with normal security (operate-once or operate-many);
- **Case 3**: Direct control with enhanced security (direct-operate);
- **Case 4**: SBO control with enhanced security (operate-once or operate-many).

Normal security commands are considered for execution if the command signal is found in Excel configuration. There aren't any additional checks in command execution in any master protocol.

Enhanced security commands need feedback from master protocol to either to succeed or fail. If feedback is not received within **command_ack_timeout_ms** timeframe, the command is considered as failed.

Command value attributes (e.g. stVal) must be updated separately (if they need to be updated).

When using SBO commands, select is not routed to master protocol and select logic is performed only in IEC 61850 Server protocol.

## Configuring datapoints

To use IEC 61850 Server in WCC Lite, it has to be configured via an Excel configuration and data model must be uploaded. This configuration contains two Excel sheets where parameters have to befilled in - Devices and Signals.

IEC 61850 Server parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 61850 Server | |
| tls | string | Selecting if TLS should be used | No | 0 | 0 | 1 |
| bind_address | string (IP address format) | IP address of and interface to use with server | No | 0.0.0.0 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| host | string (IP address format) | IP address list of allowed IPs (separated with spaces) | Yes | | | |
| port | integer | TCP communication port | Yes | | | |
| tls_local_certificat e | string | Local certificate for TLS connection | Yes (for TLS) | | | |
| tls_peer_certificat e | string | Certificate authority file for TLS connection | Yes (for TLS) | | | |
| tls_private_key | string | File consisting of private key for TLS connection | Yes (for TLS) | | | |
| event_history_size | integer | Event log size | No | | | |
| ied_name | string | Name of an Intelligent Electronic Device | Yes | | | |
| authorization | string | Authorization type | No | | password | |
| password | string | Authorization password for server device | Yes (if authorization is yes) | | | |
| model_filename | string | Filename of data model uploaded to WCC (with or without file extension) | Yes | | | |
| edition | string | Which IEC61850 edition to use. | No | 2 | 1,2, 2.1 | |
| command_ack_ti meout_ms | integer | Timeframe (ms) in which enhancedsecurity commands must be acknowledged (Default: 3000) | No | 3000 | | |
| report_buffered_si ze | integer | Report control blocks buffer size in bytes (Default: 65536) | No | 65536 | | |
| report_unbuffered _size | integer | Unbuffered report control blocks buffer size in bytes (Default: 65513) | No | 65513 | | |

## IEC 61850 Server parameters for Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |

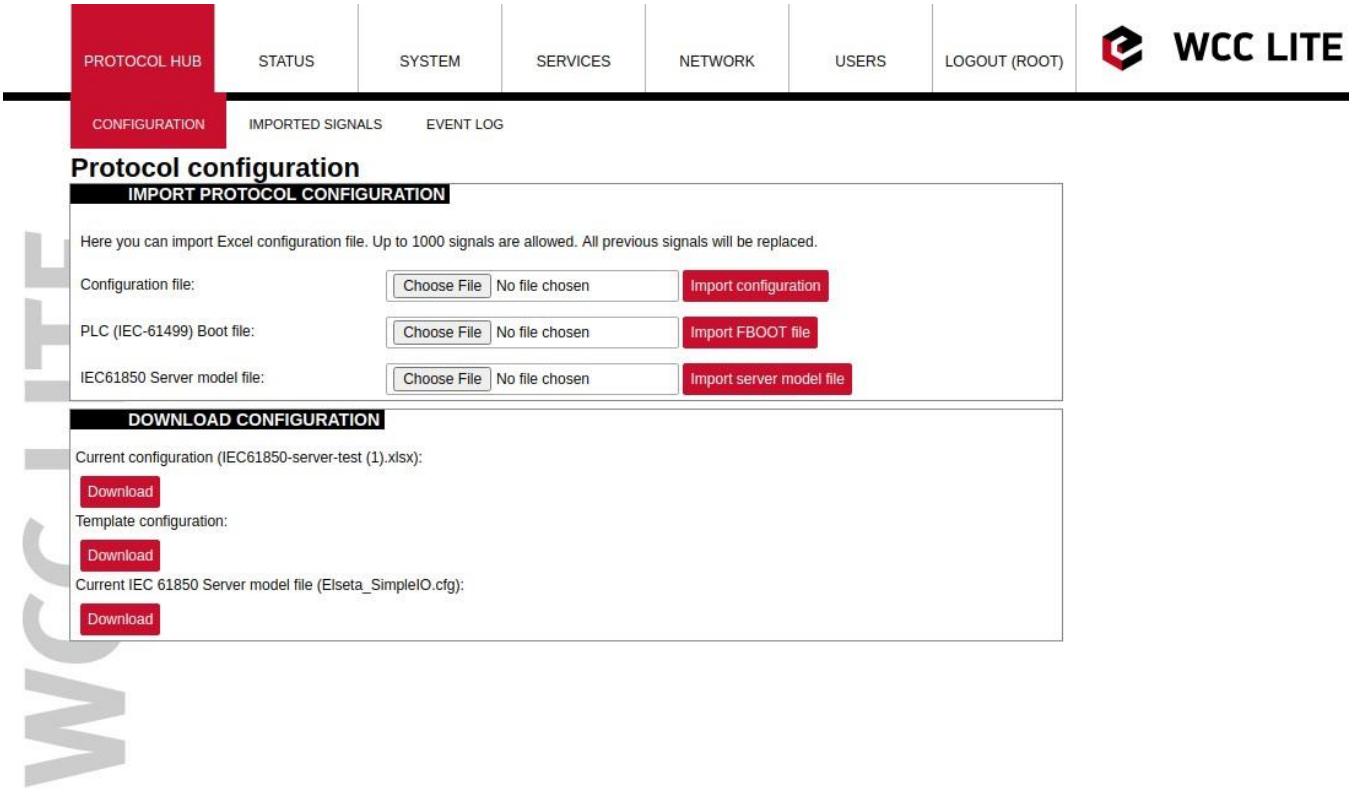
| | | | | | | |
|---|---|---|---|---|---|---|
| signal_alias | string | Unique alphanumeric name of the signal to be used | Yes | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | boolean | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | | |
| number_type | string | Number format type (BOOLEAN, FLOAT, INT16, etc.) | Yes | | | |
| ld_instance | string | Instance of a logical device | Yes | | | |
| ln_class | string | Logical node class type | Yes | | | |
| ln_instance | integer | Instance of a logical node | No | | | |
| ln_prefix | string | Prefix of logical node string | No | | | |
| cdc | string | Common Data Class (CDC) name | Yes | | SPS, DPS, INS, ENS, ACT, ACD, MV, CMV, SAV, SPC, DPC, INC, ENC, BSC, ISC, APC, BAC | |
| data_object | string | Name of data object in dataset | Yes | | | |
| da_value | string | Name of a data attribute value node | Yes | | | |
| da_time | string | Name of a data attribute time node | No | | | |
| da_quality | string | Name of a data attribute quality node | No | | | |
| da_fc | string | Functional constrain for data object | Yes | | ST,MX, CO, SP | |
| control_model | string | Model of output control | No | status-only | status-only, direct-with-normal-security, sbo-with-normal-security, direct-with-enhanced-security, sbo-withenhanced-security | |

# Converting and uploading data model

To use IEC61850 Server protocol in WCC Lite, user must upload a data model in specific format (file extension .cfg). These data models can be converted from SCL files (.icd or .cid files). To convert a data model, the user must use WCC Excel Utility. There's a separate tab for this operation as shown in picture below.




Converted file can be uploaded in WCC Lite web interface, Protocol Hub section. Current model can be also downloaded in the same page as shown in picture below.




## Debugging an IEC 61850 server application

If the configuration for IEC 61850 Server is set up, a handler for the protocol will start automatically. If the configuration is missing or contains errors, the protocol will not start. It is done intentionally to decrease unnecessary memory usage.

⚠ If IEC 61850 Server 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 the link is not functioning properly.

ℹ To launch a debugging session, a user should stop iec61850-server process and run iec61850-server command with respective flags as you can see below:

```
iec61850-server


-h [--help] Show help message
-c [--config] arg Configuration file location
-V [--version] Show version
```

```
-d [--debug] arg Set Debug level
-r [--redis] Show Redis messages
-C [--commands] Show command messages
-R [--readyfile] arg Ready notification file
```



# 15.3 IEC 61850 Client

WCC Lite can be used as a master station to collect data from IEC 61850 compatible server devices such as protection relays. As relays require fast, secure and responsive interfaces, WCC Lite can be considered as a valid option. For additional security a user can use encrypted transmission (TLS) or set up a password.

As TCP (TLS) connection can encounter issues and break, automatic reconnection is implemented. After every failed reconnection attempt the fallback delay is doubled starting from 1 second up until 32 seconds. After that connection reestablishment will be attempted every 32 seconds until a successful connection.

## Acquiring data via report control blocks

As per IEC 61850 standard, the report control block controls the procedures that are required for reporting values of data objects from one or more logical nodes to one client. Automatic reporting enables data servers (slave devices) to only send data on its (or its quality) change, thus saving network bandwith. Instances of report control blocks are configured in the server at configuration time.

Report control blocks send information that is defined in their respective datasets. Dataset is a set of data elements grouped to represent some data group. For example, it is a common practice to group measurements and events into different groups.

A server restricts access to an instance of a report control block to one client at a time. That client exclusively shall own that instance and shall receive reports from that instance of report control blocks. There are two classes of report control blocks defined, each with a slightly different behaviour:

- buffered-report-control-block (BRCB) - internal events (caused by trigger options data-change, quality-change, and data-update) issue immediate sending of reports or buffer the events (to some practical limit) for transmission, such that values of data object are not lost due to transport flow control constraints or loss of connection. BRCB provides the sequence-of-events (SOE) functionality;
- unbuffered-report-control-block (URCB) - internal events (caused by trigger options data-change, qualitychange, and data-update) issue immediate sending of reports on a best efforts basis. If no association exists, or if the transport data flow is not fast enough to support it, events may be lost.

Buffered report control blocks are therefore useful to keep event data, for example, keeping the last known state of a relay switch where a loss of information might lead to a confusion and even financial losses. Unbuffered report control blocks are particularly useful for data which is useful only momentarily, e.g. measurements of voltages, current or power. This information can change frequently and old measurements might not reflect the real state of a substation.

To allow multiple clients to receive the same values of data object, multiple instances of the report control classes shall be made available.

Buffered report control blocks are usually configured to be used by a specific client implementing a well-defined functionality, for example, a SCADA master. The client may know the ObjectReference of the BRCB by configuration or by the use of a naming convention.

Parsing of report control blocks is based on types of Common Data Class (CDC). Some of these types can have more then one data point of interest. Table below shows what data attributes are supported from various Common Data Classes. To select which data attribute should be used a da_value column should be filled with a data attribute name. Common Data Classes consist of data attributes with different Functional Constraints therefore to get the status points of interest correctly the user must fill in a correct value in da_fc column.

IEC 61850 Client supported data attributes:

| Common Data Class | Function Constraint | Data attributes |
|---|---|---|
| SPS<br>DPS INS<br>ENS | ST | stVal |
| ACT | ST | general<br>phsA phsB<br>phsC neut |

| | | |
|---|---|---|
| ACD | ST | general<br>dirGeneral<br>phsA dirPhsA<br>phsB dirPhsB<br>phsC dirPhsC<br>neut dirNeut |
| MV | MX | instMag mag |
| CMV | MX | instCVal cVal |
| SAV | MX | instMag |
| SPC<br>DPC INC<br>ENC | ST | stVal |
| BSC<br>ISC | ST | valWTr |
| APC BAC | MX | mxVal |

Some of data attributes are structures themselves, for example, mag attribute is a struct that can hold integer or float values. To select a fitting attribute the user should extend da_value parameter with additional attributes, for example, if float magnitude value is to be selected from MV Common Data Class, da_value column should be filled with mag.f value; if the user intends cVal magnitude value in float format from CMV Common Data Class, da_value should be filled with cVal.mag.f value. See IEC 61850-7-3 for more information about Common Data Classes.

To ensure the integrity of configuration, WCC Lite has additional checks implemented at configuration time. If report control block (or its dataset) with a predefined ObjectReference doesn't exist, it is considered that IEC 61850 Client has not been configured properly or configuration has been changed in either of IEC 61850 devices and cannot be matched, therefore should be considered invalid.

## Number Types

IEC 61580 has a distinct number_type field when compared to other protocols.

| number_type |
|---|
| BOOLEAN |
| INT8 |
| INT16 |
| INT32 |
| INT64 |
| INT128 |
| INT8U |
| INT24U |
| INT32U |
| FLOAT32 |
| FLOAT64 |

| |
|---|
| ENUMERATED |
| OCTET STRING 6 |
| OCTET STRING 8 |
| OCTET STRING 64 |
| VISIBLE STRING 32 |
| VISIBLE STRING 64 |
| VISIBLE STRING 65 |
| VISIBLE STRING 129 |
| UNICODE STRING 255 |
| TIMESTAMP |
| QUALITY |
| CHECK |
| CODEDENUM |
| GENERIC BITSTRING |
| CONSTRUCTED |
| ENTRY TIME |
| PHYCOMADDR |
| CURRENCY |
| OPTFLDS |
| TRGOPS |

## Controlling remote equipment via commands

The control model provides a specific way to change the state of internal and external processes by a client. The control model can only be applied to data object instances of a controllable Common Data Class (CDC) and whose ctlModel DataAttribute is not set to status - only. Such data objects can be referred to as control objects. If controls are enabled in a IEC 61850 Server device the user can configure controls by filling control_model column in Excel configuration with a control model (direct-with-normal-security, sbo-with-normal-security, direct-with-enhanced-
security, sbo-with-enhanced-security) as well as setting functional constraint in da_fc column to CO.

Depending on the application, different behaviours of a control object shall be used. Therefore, different state machines are defined. Four cases are defined:

- **Case 1**: Direct control with normal security (direct-operate);
- **Case 2**: SBO control with normal security (operate-once or operate-many);
- **Case 3**: Direct control with enhanced security (direct-operate);
- **Case 4**: SBO control with enhanced security (operate-once or operate-many).

IEC 61850 standard enables the user to plan command transmission in advance - set the timer when the command should be issued. However, as this possibility is rarely used in practice, it is not implemented as of version v1.5.0. All issued commands are executed immediately.

For more information on control class model, please consult IEC 61850-7-2 standard.

If ctlModel is read-only, messages from internal database will be ignored for this point, otherwise a subscribe callback will be launched to handle commands as soon as they are sent. If CDC of a signal does not have means of control, ctlModel parameter is ignored.

Originator identification can be attached to a station so that replies to command requests could be forwarded to only one device. To use this functionality a user should select an origin identificator by filling value in Excel configuration, originator column. Originator category is always enforced to tell that remote control command is issued.

# Configuring datapoints

To use IEC 61850 Client 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 and Signals tables.

Table IEC 61850 Client parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 61850 Client | |
| tls | string | Selecting if TLS should be used | No | 0 | 0 | 1 |
| ip | string (IP address format) | IP address of server device | Yes | 0.0.0.0 | | |
| port | integer | TCP communication port | Yes | 102 | | |
| tls_local_certificate | string | Local certificate for TLS connection | Yes (for TLS) | | | |
| tls_peer_certificat e | string | Certificate authority file for TLS connection | Yes (for TLS) | | | |
| tls_private_key | string | File consisting of private key for TLS connection | Yes (for TLS) | | | |
| event_history_size | integer | Event log size | No | | | |
| ied_name | string | Name of an Intelligent Electronic Device | Yes | | | |
| authorization | string | Authorization type | No | | password | |
| password | string | Authorization password for server device | No | | | |
| originator | string | Origin identificator for device | No | | | |

## Table IEC 61850 Client parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | boolean | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | | |
| number_type | string | Number format type | Yes | | | |
| ld_instance | string | Instance of a logical device | Yes | | | |
| ln_class | string | Logical node class type | Yes | | | |
| ln_instance | integer | Instance of a logical node | No | | | |
| ln_prefix | string | Prefix of logical node string | No | | | |
| cdc | string | Common Data Class (CDC) name | Yes | | SPS, DPS, INS, ENS, ACT, ACD, MV, CMV, SAV, SPC, DPC, INC, ENC, BSC, ISC, APC, BAC | |
| data_object | string | Name of data object in dataset | Yes | | | |
| da_value | string | Name of a data attribute value node | Yes | | | |
| da_fc | string | Functional constrain for data object | Yes | | ST,MX, CO, SP | |
| control_model | string | Model of output control | No | status-only | status-only, direct-with-normal-security, sbo-with-normal-security, direct-with-enhanced-security, sbo-withenhanced-security | |
| dataset | string | Full object reference of a dataset | Yes | | | |
| report_control_blo ck | string | Full object reference of a report control block | Yes | | | |
| intgPd | integer | Integrity period in milliseconds | No | 0 | | |

It should be noted that ACT and ACD messages can only be parsed from report if either only ‘general’ attribute or all attributes attached to all three phases and neutral can be found in report

IEC 61850 Client has an additional signal which can be configured to show communication status. It is used to indicate if the server device has disconnected from client (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 after a disconnection of a server
device.

## Debugging a IEC 61850 Client application

If configuration for IEC 61850 Client 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.

IEC 61850 Client command line debugging options

`iec61850-client`

```
-h [ –help ] Show help message
-c [–config] arg Configuration file location
-V [–version] Show version
-d [–debug] arg Set debugging level
-r [–redis] Show Redis messages
-C  [–commands] Show command messages
-D  [–datasets] Show dataset messages
–report Show report messages
-R [–readyfile] arg Ready notification file
```

If IEC 61850 Client 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 `iec61850-client` process and run `iec61850-client` command with respective flags as was shown above.

# 16 Specific protocols

– Aurora (ABB PV inverters protocol) – PowerOne (ABB PV inverters protocol) – SMA Net (SMA PV inverters protocol) – Kaco (Kaco PV inverters protocol) – Ginlong (Ginlong PV inverters protocol) – Solplus (Solutronic AG PV inverters protocol) – ComLynx (Danfoss PV inverters protocol) – Delta (Delta PV inverters protocol) – Windlog (Wind sensors from RainWise Inc.) – Vestas ( Wind turbines protocol) – VBus.

# 16.1 At command

## Overview

At command protocol is used for communications with AT Commands.

## Configuration

At command parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | at command | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. | No | none | none | |
| timeout_ms | integer | Timeout of waiting for incoming request in miliseconds | Yes | | 0 | 60000 |

| | | | | | | |
|---|---|---|---|---|---|---|
| serial_close_delay | integer | Delay before closing serial port | No | 400 | | |

## At command parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |

# 16.2 Aurora

## Overview

The Aurora Protocol is a link layer communications protocol for use on pointtopoint serial links. It is intended for use in highspeed (gigabits/second and more) connections internally in a computer or in an embedded system. It uses either 8b/10b encoding or 64b/66b encoding

### Aurora parameters for Device tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Aurora | |
| baudrate | integer | Communication speed, bauds/s (See values 33.1.2) | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option (”none”/”even”/”odd”) | No | none | | |
| flowcontrol | string | Communication device flow control option. | No | none | | |
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout in milliseconds | No | 2500 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| id | integer | Inverter ID | No | 0 | | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |

## Aurora parameters for Signals tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly device name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log (Default: 0) | No | 0 | 0 | |
| signal_alias | string | Unique alphanumeric name of the signal to be used | Yes | | | |
| job_todo | boolean | Define tagfunction | Yes | | | |
| tag_job_todo | string | Define tag action that depends on tag function | Yes | | | |
| number_type | integer | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| pulse_short_time_ms | integer | Time interval for short output pulse to stay active | No | 0 | | |
| pulse_long_time_ms | integer | Time interval for long output pulse to stay active | No | 0 | | |

# 16.3 COMLYNX

## Overview

Comlynx protocol is used to communicate with Comlynx inverters over serial communication.

### Comlynx parameters for Device tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Comlynx | |
| address | integer | Device address | No | 1 | | |
| subnet | integer | Subnet address | No | 0 | | |
| network | integer | Network address | No | 0 | | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 19200 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option (”none”/”even”/”odd”) | No | none | | |
| flowcontrol | string | Communication device flow control option. (Default: (casesensitive): ”none”) | No | none | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout in milliseconds | Yes | | 0 | 60000 |

## Comlynx parameters for Signals tab:

| **Parameter** | **Type** | **Description** | **Required** | | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|---|
| | | | | | | Min | Max |
| signal_name | string | User-friendly device 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 | No | | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. | No | | 0 | | |
| job_todo | boolean | Define tagfunction | Yes | | | | |
| tag_job_todo | string | Define tag action that depends on tag function | Yes | | | | |
| number_type | integer | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | | |
| pulse_short_time_ms | integer | Time interval for short output pulse to stay active | No | | 0 | | |
| pulse_long_time_ms | integer | Time interval for long output pulse to stay active | No | | 0 | | |

# 16.4 Delta

## Overview

Delta protocol is used to communicate with Delta inverters over serial communication.

## Configuration

Delta parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Delta | |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option (”none”/”even”/”odd”) | No | none | | |
| flowcontrol | string | Communication device flow control option. (Default: (casesensitive): ”none”) | No | none | | |
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| timeout_ms | integer | Timeout in milliseconds | No | | 0 | 60000 |
| id | integer | Inverter ID | Yes | 0 | | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |

### Delta parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |
| 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 | | | |

# 16.5 GINLONG

## Overview

Ginlong protocol is used to communicate with Ginlong inverters over serial communication.

## GINLONG parameters for Device tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Ginlong | |
| baudrate | integer | Communication speed, bauds/s (See values 33.1.2) | No | 9600 | 300 | 115200 |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option (”none”/”even”/”odd”) | No | none | | |
| flowcontrol | string | Communication device flow control option. (Default: (casesensitive): ”none”) | No | none | | |
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout in milliseconds | No | 2500 | | |
| id | integer | Inverter ID | Yes | 0 | | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |

GINLONG parameters for Signals tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly device 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 | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. | No | 0 | | |
| job_todo | boolean | Define tagfunction | Yes | | | |
| tag_job_todo | string | Define tag action that depends on tag function | Yes | | | |
| number_type | integer | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| pulse_short_time_ms | integer | Time interval for short output pulse to stay active | No | 0 | | |
| pulse_long_time_ms | integer | Time interval for long output pulse to stay active | No | 0 | | |

# 16.6 Kaco

## Overview

This protocol is meant to be used by inverters that convert the DC power generated by the photovoltaic (PV) modules into AC power and feed this into the power grid.

This protocol handles serial communication parameters (baudrate, databits, stopbits, etc.) automatically.

## Configuration

Kaco parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | Kaco | |
| scan_rate_ms | integer | All reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout of waiting for incoming request in miliseconds | No | 2500 | 0 | 60000 |
| subid | integer | Inverter serial number display | No | 0 | | |
| ext_device | boolean | 0 - Inverter isconnected directly<br>1 - Inverter is connected via remote terminal | No | 0 | 0 | 1 |
| id | integer | Inverter serial number | Yes | | | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |

## Kaco parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |
| 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 | | | |

**Required**

# 16.6 KOSTAL

## Overview

Kostal protocol is used to communicate with Kostal devices over serial communication.

## Configuration

Kostal parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | kostal | |
| id | integer | Kostal device id | Yes | | | |
| device | | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| scan_rate_ms | integer | Delay before closing serial port in miliseconds | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| timeout_ms | integer | Timeout of waiting for incoming request in miliseconds | Yes | | 0 | 60000 |

## Kostal parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |
| 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 | | | |

# 16.7 POWERONE

## Overview

PowerOne protocol is used to communicate with Aurora inverters over serial communication. Serial communication parameters (baudrate, parity, etc.) are handled automatically by the protocol.

## Configuration

PowerOne parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| Min | Max | | | | | |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | powerone | |
| serialnumber | integer | Inverter serial number | Yes | | | |
| type | integer | Inverter type :<br>• CU<br>• <br>• <br>Collect ing unit<br>CB - Norma l CB<br>HID - HID with integra ted CB | No | CU | | |
| device | | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| scan_rate_ms | integer | Delay before closing serial port in miliseconds | No | 10000 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout of waiting for incoming request in miliseconds | No | 1000 | 0 | 60000 |

PowerOne parameters for Signals tab

| Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|
| | | | | Min | Max |
| | User-friendly signal name | Yes | | | |
| | Device alias from a Devices tab | Yes | | | |
| | Unique alphanumeric name of the signal to be used | Yes | | | |
| an | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| r | Enable logging in event log | No | 0 | | |
| | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| | Tag sub job | Yes | | | |
| r | Time interval for short output pulse to stay active | No | 0 | | |
| r | Time interval for long output pulse to stay active | No | 0 | | |

# 16.8 SMA NET

## Overview

SMA Net can transfer SMA Data, TCP/IP and many more telegrams due to its multiprotocol capability. Thus, it is the preferred telegram frame in case of new developments.

# Configuration

## SMA NET parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | sma net | |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. | No | none | none | |
| scan_rate_ms | integer | Delay before closing serial port in miliseconds | No | 10000 | | |
| poll_delay_ms | integer | | No | 200 | | |
| timeout_ms | integer | Timeout of waiting for incoming request in miliseconds | No | 2500 | | |
| serial_number | integer | Inverter serial number | Yes | | | |
| device | | Communication port | Yes | | PORT1 | PORT2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| serial_close_delay | integer | Delay before closing serial port | No | 400 | | |

SMA NET parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |
| 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 | | | |

# 16.9 SOLPLUS

## Overview

Solplus protocol is used to download inverter data from Solplus inverters using a HTTP client.

## Configuration

Solplus parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | Solplus | |
| scan_rate_ms | integer | All reads and writes will be executed within thetimeframe in miliseconds | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout of waiting for incoming request in miliseconds | No | 2500 | 0 | 60000 |
| url | string | HTTP server location URL | Yes | | | |

## Solplus parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |
| 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 | | | |

# 16.10 VBUS

## Overview

Vbus is a protocol used for communication with solar station automation via serial link.

## Configuration

VBUS parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Vbus | |
| slave_address | integer | Slave device address | Yes | | 0 | 255 |
| master_address | integer | Master device address | Yes | | 0 | 255 |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. | No | none | none | |
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout in milliseconds | No | 2500 | 0 | 60000 |

VBUS parameters for Signals tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly device 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 | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. | No | 0 | 0 | |
| job_todo | boolean | Define tagfunction | Yes | | | |
| tag_job_todo | string | Define tag action that depends on tag function | Yes | | | |
| number_type | integer | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| pulse_short_time_ms | integer | Time interval for short output pulse to stay active | No | 0 | | |
| pulse_long_time_ms | integer | Time interval for long output pulse to stay active | No | 0 | | |

# 16.11 VESTAS

## Overview

Vestas is a protocol used for communication with solar station automation via serial link.

## Configuration

Vestas parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Vestas | |
| slave_address | integer | Slave device address | Yes | | 0 | 255 |
| master_address | integer | Master device address | No | 0 | 0 | 255 |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option (”none”/”even”/”odd”) | No | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. (Default: (casesensitive): ”none”) | No | none | | |
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in miliseconds. | No | 10000 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout in milliseconds | No | 2500 | | |

Vestas parameters for Signals tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly device 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 | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. | No | 0 | | |
| job_todo | boolean | Define tagfunction | Yes | | | |
| tag_job_todo | string | Define tag action that depends on tag function | Yes | | | |
| number_type | integer | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| pulse_short_time_ms | integer | Time interval for short output pulse to stay active | No | 0 | | |
| pulse_long_time_ms | integer | Time interval for long output pulse to stay active | No | 0 | | |

# 16.12 Windlog

## Overview

Windlog protocol is used for communications with the Windlog data logger.

## Configuration

Windlog parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | Windlog | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 115200 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| databits | integer | Data bit count for communication | No | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No | 1 | 1 | 2 |
| parity | string | Communication parity option | No | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. | No | none | none | |
| timeout_ms | integer | Timeout of waiting for incoming request in milliseconds | Yes | | 0 | 60000 |
| serial_close_delay | integer | Delay before closing serial port | No | 400 | | |

Windlog parameters for the Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range |
|---|---|---|---|---|---|

| | | | | | Min | Max |
|---|---|---|---|---|---|---|
| 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 an individual signal | No | 1 | 0 | 1 |
| log | integer | Enable logging in the event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma-separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |

# 16.13 M-Bus

## Overview

M-Bus or Meter-Bus is a protocol for the remote reading of water, gas, or electricity meters. M-Bus is also usable for other types of consumption meters, such as heating systems or water meters. The M-Bus interface is made for communication on two wires, making it cost-effective. M-bus over TCP is also supported. When configured, meters will deliver the data they have collected to a WCCLite RTU that is connected at periodic intervals (scan_rate_ms) to read all utility meters.

## Configuration

M-Bus parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | mbus serial, mbus tcp | |
| scan_rate_ms | integer | All reads and writes will be executed within the timeframe in milliseconds. | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in milliseconds to wait before sending any data on port. | No | 200 | | |
| timeout_ms | integer | Timeout of waiting for an incoming response in milliseconds | Yes | | 0 | 60000 |
| address | integer | Device address | Yes | | | |
| device | string | Communication port | Yes (for serial) | | PORT1 | PORT2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| baudrate | integer | Communication speed, baud/s | No (for serial) | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No (for serial ) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No (for serial) | 1 | 1 | 2 |
| parity | string | Communication parity option | No (for serial) | none | none, even, odd | |
| serial_close_delay | integer | Delay before closing the serial connection. | No (for serial) | 400 | | |
| ip | string | The IP address of the TCP slave device | Yes (for TCP). | | | |
| port | integer | TCP communication port | Yes (for TCP) | | 0 | 65535 |

## M-Bus parameters for the Signals tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Range** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in the event log | No | 0 | | |
| number_type | string | Type of a number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma-separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |

# 17 Metering protocols

- DLMS/COSEM - IEC 62056-21 - MBus Serial/TCP - Elgama (Meters based on IEC 62056-21 / 31 protocols)

# 17.1 DLMS/COSEM

## Introduction

**IEC 62056** is a set of standards for electricity metering data exchange by International Electrotechnical Commission.

The IEC 62056 standards are the international standard versions of the DLMS/COSEM specification.

**DLMS** or **Device Language Message Specification** (originally Distribution Line Message Specification),[1] is the suite of standards developed and maintained by the DLMS User Association (DLMS UA) and has been adopted by the IEC TC13 WG14 into the IEC 62056 series of standards. The DLMS User Association maintains a D Type liaison with IEC TC13 WG14 responsible for international standards for meter data exchange and establishing the IEC 62056 series. In this role, the DLMS UA provides maintenance, registration and compliance certification services for IEC 62056 DLMS/COSEM.

**COSEM** or **Companion Specification for Energy Metering**, includes a set of specifications that defines the transport and application layers of the DLMS protocol. The DLMS User Association defines the protocols into a set of four specification documents namely Green Book, Yellow Book, Blue Book and White Book. The Blue Book describes the COSEM meter object model and the OBIS object identification system, the Green Book describes the architecture and protocols, the Yellow Book treats all the questions concerning conformance testing, the White Book contains the glossary of terms. If a product passes the conformance test specified in the Yellow Book, then a certification of DLMS/COSEM compliance is issued by the DLMS UA.

The IEC TC13 WG14 groups the DLMS specifications under the common heading: "Electricity metering data exchange The DLMS/COSEM suite". DLMS/COSEM protocol is not specific to electricity metering, it is also used for gas, water and heat metering.

Source: https://en.wikipedia.org/wiki/IEC_62056

# DLMS Master

## 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 (HDLC or WRAPPER) 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.

## Configuration

### Devices section

**serial_number**, **physical_address** and **logical_address** define the meter addressing parameters. Either **serial_number** (meter serial number) or a combination of **physical_address** and **logical_address** is used. If a serial number is provided, physical and logical server addresses are ignored.

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

**client_address** is defined in hex and usually depends on the authentication used. Most meters support hex 11 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 communications mode. If IEC is used along with comms settings for serial readout, the connection is initiated as per IEC 62056-21, at the default initial baud rate (300 7E1). DLMS-HDLC shall be used for HDLC connections via IP. DLMS-WRAPPER is also supported for IP connections. The default setting is DLMS-HDLC. **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.

⚠ 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.

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

Device configuration parameters for DLMS meters acquisition:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | DLMS | |
| serial_number | integer | Meter serial number | No | 0 | | |
| physical_address | integer | Meter physical server address | No | 1600 | | |
| logical_address | integer | Meter logical server address | No | 0 | | |
| address_size | integer | Meter address size in bytes | No | 1 | 1 | 4 |
| client_address | integer | Client address | Yes | | | |
| type | string | Meter object referencing: SN short referencing, LN - logical referencing | No | SN | SN, LN | |
| mode | string | Initial handshake mode. | Yes | DLMS-HDLC | IEC, DLMS-HDLC or DLMS-WRAPPER | |
| timeout_ms | integer | Timeout in milliseconds | No | 2500 | | |
| auth | string | Authentication. | No | None | None, Low, High, HighMd5, HighSha1, HighSha256, HighGmac, HighEcdsa | |
| password | string | Password for authentication | No when auth is None | | | |
| ip | string | IP address | Yes (For TCP) | | | |
| port | integer | TCP port | Yes (For TCP) | | | |
| device | | Communication port | Yes (For Serial) | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No (For Serial) | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No (For Serial) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No (For Serial) | 1 | 1 | 2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| parity | string | Communication parity option | No (For Serial) | none | none, even, odd | |
| flowcontrol | string | Communication device flow control option. | No (For Serial) | none | none | |
| retry_counter | integer | Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued | No | 3 | | |
| scan_rate_ms | integer | If provided and positive all reads and writes will be executed within the timeframe in milliseconds | No | 10000 | | |
| poll_delay_ms | integer | Minimum time delay in milliseconds to wait before sending any data on port. | No | 200 | | |
| reconnect_time | integer | Defines how often (in ms) the client will try to reestablish communication with the meter after an unsuccessful attempt. | No | 1000 | | |

## Signals section

DLMS configuration parameters creating signals:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | boolean | Enable logging in event log | No | 0 | 0 | 1 |
| short_name | integer | Address of value to read (Short name). | No | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| obis_job | string | OBIS codes can be accompanied by an attribute index, eg.: 1.0.1.8.0.255:2. Objects of register and extended register types do not require indexes and the scalers are applied to values automatically (though they can still be used if attributes other than the value need to be read out). | Yes | | | |

## Debugging the DLMS service

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

If DLMS does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from the command-line interface and find out why the link is not functioning properly. To launch a debugging session, a user should stop dlms process and run dlms command with respective flags as in the table shown below.

Procedure for DLMS protocol service debugging:

- **Step 1**: Service must be stopped by entering the following command into the wcclite: **/etc/init.d/dlms stop**
- **Step 2**: After service is stopped it must be started with the preferred configuration file (JSON files found in /etc/dlms folder) and a debug level 7: **dlms -c /etc/dlms/dlms.json -d7 --dlms** Additional output forming options described in the table below.

- **Step 3**: Once the problem is diagnosed normal operations can be resumed with the following command: **/etc/init.d/dlms start**

DLMS command line debugging options

| Option | Description |
|---|---|
| -h [ —-help ] | Display help information |
| -V [ —-version ] | Show version |
| -p [ -—port ] | Show output for one port only |
| -d <debug level> | Set debugging level |
| -c [ —-config ] | Config path |



# 17.2 IEC 62056-21

## Introduction

**IEC 61107** or currently IEC 62056-21, was an international standard for a computerprotoco l to read utility meters. It is designed to operate over any media, including the Internet. A meter sends ASCII (in modes A..D) or HDLC (mode E) data to a nearby hand-held unit (HHU) using a serial port. The physical media are usually either modulated light, sent with an LED and received with a photodiode, or a pair of wires, usually modulated by a 20mAcurrent loo p. The protocol is usually half-duplex.

The following exchange usually takes a second or two, and occurs when a person from the utility company presses a meter-reading gun against a transparent faceplate on the meter, or plugs into the metering bus at the mailbox of an apartment building.

The general protocol consists of a "sign on" sequence, in which a handheld unit identifies itself to the metering unit. During sign-on, the handheld unit addresses a particular meter by number. The meter and hand-held unit negotiate various parameters such as the maximum frame length during transmission and reception, whether multiple frames can be sent without acknowledging individual frames (windowing), the fastest communication rate that they can both manage (only in case of mode E switching to HDLC) etc.

Next, the meter informs the handheld unit about the various parameters that are available with it in various security settings viz. the 'no-security logical group', ' the low-security logical groups' and ' the high-security logical groups'.

If the parameter required is in the no-security group, just a get.request will provide the HHU with the desired response. If the parameter required is in the low-security group, a password authentication of the HHU is required before information can be read.

In case of high-security parameters, the meter challenges the handheld unit with a cryptographic password. The handheld unit must return an encrypted password. If the password exchange is correct, the meter accepts the handheld unit: it is "signed on."

After signing on, the handheld unit generally reads a meter description. This describes some registers that describe the current count of metered units (i.e. kilowatt hours, megajoules, litres of gas or water) and the metering unit's reliability (is it still operating correctly?). Occasionally a manufacturer will define a new quantity to measure, and in this case, a new or different data type will appear in the meter definition. Most metering units have special modes for calibration and resetting meter registers. These modes are usually protected by anti-tampering features such as switches that sense if the meter enclosure has been opened.

The HHU may also be given limited rights to set or reset certain parameters in the meter.

The handheld unit then sends a sign-off message. If no sign-off message is sent, the meter automatically signs off after a previously negotiated time interval after the last message.

Source: https://en.wikipedia.org/wiki/IEC_62056#IEC_62056-21

# 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.

# Configuration

## Device section

The 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.

The 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.

The 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.

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

IEC 62056-21 device configuration parameters:

| Parameter | Type | Description | Required | Default value (when not specified) | Range |
|---|---|---|---|---|---|

| | | | | | Min | Max |
|---|---|---|---|---|---|---|
| name | string | User-friendly name for a device | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used | Yes | | IEC 62056-21 | |
| poll_delay_ms | integer | Minimum time delay in miliseconds to wait before sending any data on port. | No | 200 | | |
| scan_rate_ms | integer | | | 10000 | | |
| device | string | Communication port | No (for serial) | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No (for serial) | 6900 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | |
| databits | integer | Data bit count for communication | No (for serial) | 8 | 6 | 9 |
| stopbits | integer | Stop bit count for communication | No (for serial) | 1 | 1 | 2 |
| parity | string | Communication parity option | No (for serial) | NONE | NONE, EVEN, ODD | |
| flowcontrol | string | Communication device flow control option | No | | NONE | |
| serialnumber | unsigned long | Meter serial number | Yes | | 1 | |
| serial_close_delay | integer | Delay before closing serial port | No | 400 | | |
| timeout_ms | integer | Timeout of waiting for incoming request | No | 2500 | | |
| type | string | Defines a connection mode | No | C | A,B,C | |
| t2 | integer | Time to wait before acknowledging the suggested baudrate in mode C | No | 300 | 200 | 1500 |
| ip | string | IP address for TCP connection | Yes (for TCP) | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| port | integer | TCP port | Yes (for TCP) | | 0 | 65535 |

## Signals section

**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. For 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).

IEC 62056-21 tags configuration parameters:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 use | Yes | | | |
| enable | boolean | | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. If **log is 0 signal** will not be logged. If **log is more than 0** signal will be logged | No | 0 | 0 | |
| number_type | string | Number format type | Yes | | | |
| tag_job_todo | string | Tag job in OBIS format | Yes | | | |

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.



# 17.3 Elgama

## Overview

Elgama protocol is used for communications with Elagama elektronika electricity meters.

## Configuration

Available meter types (use number only):

- 0 EPQM/LZQM
- 1 EPQS
- 2 GAMA300
- 3 GAMA100

4 ITS cl

Elgama parameters for Device tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of a device | No | | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Protocol to be used. | Yes | | Elgama | |
| device | string | Communication port | Yes | | PORT1 | PORT2 |
| baudrate | integer | Communication speed, bauds/s | No | 9600 | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200 | |
| serial_close_delay | integer | Delay before closing serial port in milliseconds | No | 400 | | |
| timeout_ms | integer | Timeout of waiting for incoming request in milliseconds | Yes | | | |
| id | integer | Meter serial number | Yes | | | |
| meter_model | integer | Meter type (See 15.2) | Yes | | 0 | 4 |
| use_time | boolean | Use system/meter (0/1) time (Default: 0) | No | 0 | 0 | 1 |

Elgama parameters for the Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| 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 | No | 1 | 0 | 1 |
| log | integer | Enable logging in event log | No | 0 | | |
| number_type | string | Type of number (FLOAT, DOUBLE, DIGITAL, etc.) | Yes | | | |
| job_todo | string | Tag job as single or multiple comma separated OBIS codes | Yes | | | |
| tag_job_todo | string | Tag sub job | Yes | | | |

# 18 Excel Configuration

Protocol HUB uses the configuration in excel file format. Each sheet represents a specific part of the configuration: Devices contain device lists and protocol-related configurations. Signals contain a list of signals and their options. Firstline on each sheet is a header row that contains the 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 the end of the sheet. Any rows after empty row are discarded.

# 18.1 Devices sheet

Devices sheet contains all devices to be configured on the 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 of the device. Used for representation only. **device_alias** - A unique short name for the device. It is used for linking signals to a device.

⚠ An alias can only contain alphanumeric characters and dashes ( - and _ ). The alias must be unique for each device.

- **protocol** - Protocol type to use on the device. The exact values of protocols are written in every protocol documentation. Please look into the range of supported protocols:

**IEC 61850 MMS:**

– IEC 61850 Client (since FW 1.5.0) – IEC 61850 Server (since FW 1.5.0)

**IEC 60870-5:**

– 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

**DNP 3.0 Serial/LAN/WAN:**

- DNP3 Master

– DNP3 Slave

**Modbus Serial/TCP:**

- Modbus RTU/ASCII

– Modbus TCP

**Metering protocols:**

- DLMS/COSEM (since FW 1.3.0)

– IEC 62056-21 (since FW 1.2.13)

– MBus Serial

– MBus TCP

– Elgama (Meters based on IEC 62056-21 / 31 protocols)

**Industrial IOT protocols:**

- MQTT

- RESTful API

**Specific protocols:**

– Aurora (ABB PV inverters protocol)

– PowerOne (ABB PV inverters protocol)

– SMA Net (SMA PV inverters protocol)
– Kaco (Kaco PV inverters protocol)

– Ginlong (Ginlong PV inverters protocol)

– Solplus (Solutronic AG PV inverters protocol)

– ComLynx (Danfoss PV inverters protocol)

– Delta (Delta PV inverters protocol)

– Windlog (Wind sensors from RainWise Inc.)

– Vestas ( Wind turbines protocol)

– Internal data– VBus.

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

## Optional settings

- **enable** - Flag to enable or disable a device on the 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.

## 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)

## TCP/IP settings

Settings for any protocol that uses communication over TCP/IP. Note that all TLS certificates and keys are stored in a single folder therefore only the name and not the path should be filled in respective fields.

TLS fields are only supported for IEC 61850 Client and Server, IEC-60870-5-104 Slave, and DNP3 Master and Slave.

- **ip** - IP address for a master protocol to connect to;
- **bind_address** - one of the 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** - the name of local TLS certificate; **tls_peer_certificate** - the name of a certificate authority (CA) TLS
- certificate; **tls_private_key** - the name of a private key for making TLS connections.
- 18 Excel Configuration
-

# 18.2 Optional parameters for signals

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

# Required attributes

These attributes are mandatory for every configured signal. Every Excel configuration should have specified them in the first row of the Signals sheet:

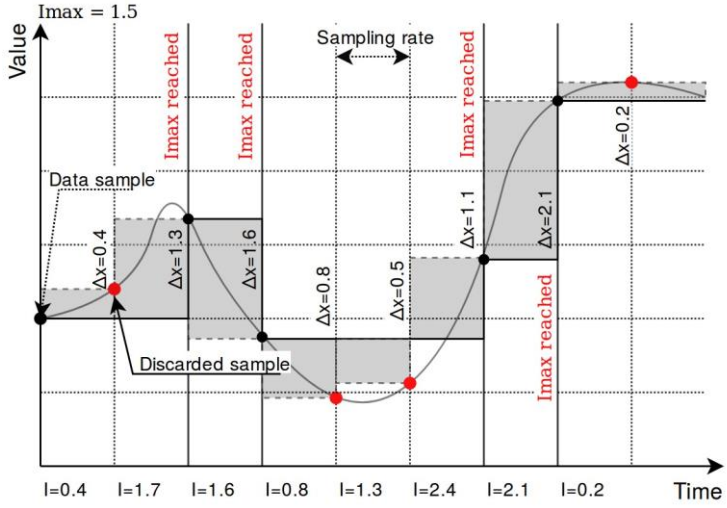
- **signal_name** - Name of the signal. Used for representation only. **device_alias** - Alias of a device defined in the Devices sheet. A signal is
- linked to a matching device. **signal_alias** - A unique short name for the signal. It is used for linking signals to other signals. The alias can
- only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

# Optional attributes

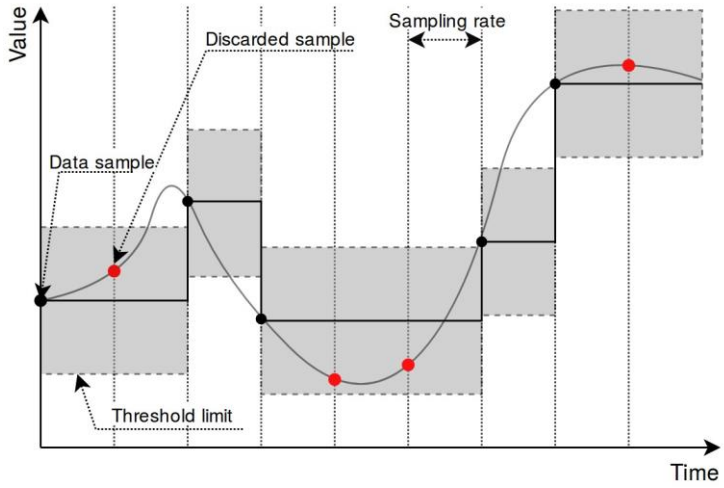
Optional attributes are required depending on the protocol in use and they can be used to extend protocol functionality:

- **source_device_alias** - Alias of a source device defined in the Devices sheet. If a user intends to use several signals and combine them via mathematical or logical function, every alias should be separated 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 the Signals sheet. If a user intends to use several signals and combine them via mathematical or logical function, every alias should be separated by a **separator** symbol (in the same cell). An operation used must also be defined in an operation column. Each 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 _ ). The device and signal alias combination must be unique. **enable** - Flag to enable or disable signal on the system. Can contain values 0 or 1.
- **tag_type** - Tag type. Simple signals are polled from the device. Virtual signals are computed internally. **units** - Signal value measurements units. **multiply** - Multiply the value by this number. **add** - Add this number to a value.
- **min_value** - Minimum expected value. If the result is lower than this value, the overflow flag is raised. **max_value** - Maximum expected value. If the result is higher than this value, the overflow flag is raised. **absolute_threshold** - Absolute threshold level.
- **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** - Maximum number of records for this tag to keep in events log.
- **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 which are separated using **separators**. 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 are intended for unsigned integers only. **bit_select** - selecting an individual bit of an integer number; bit numeration starts from zero.
- **math_expression** - a mathematical expression for master protocol monitor direction or slave command direction signals to be evaluated against. Explained in detail in **Mathematical expression document. source_math_expression** - a mathematical expression for master protocol command direction or slave monitor direction signals to be evaluated against. Explained in detail in **Mathematical expression document.**

Picture. Result of using an absolute threshold:




Picture. Result of using an integral threshold:




# 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 a 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. The 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, the value is approximated not be lower (or higher) than the limit. An additional overflow (OV) flag is added as frequently used in IEC-60870-5 protocols;
- Suppression 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 intermediaries 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;
- Threshold checking. If a new signal doesn't cross a threshold target value, the value is suppressed 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 be configured, missing operations are skipped and values are fed to a further stage of signal recalculation.

## 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 the byte order on the connected device side is different. In such a case, byte swap operations can be used. Adding swap prefixes to number types will set different byte orders 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. 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 4 | Byte 5 | Byte 6 | Byte 7 | Byte 1 | Byte 6 | 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 4 | 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 |

Where Byte x, means bit x position in the byte.

Add a dot-separated prefix to the number format to use byte swapping. Multiple swap operations can be used simultaneously. For example, use SW8.SW16.SIGNED32 it to correctly parse a 32-bit signed integer in a little-endian format. Table 35 shows in detail how bytes, words, or double-words can be swapped and how swapping functions can be combined to make different swapping patterns. The 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 the unavailability of some swapping operations ( SW32 for 32-bit and smaller numbers). Using such an unavailable operation might lead to undefined behavior.

# 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 the target signal. This way events polled from a Modbus device (e.g., Modbus, IEC 60870-5, etc.) can be delivered to an 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 toIEC 60870-5-10 4 station, the following steps may be taken:

1. Create a Modbus master configuration in the Devices sheet.
2. Create an IEC 60870-5-104 slave configuration in the Devices sheet.
3. Create a signal on the master device to read coil status (function 1).
4. Create a signal on the slave device with a 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 **signal_alias** on master device's coil signal.

## Example 2

To write a coil state to a Modbus device on a command fromIEC 60870-5-10 4 station, the following steps may be taken:

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

For additional information regarding the 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.

# Separators

These operators can be used when defining two or more values in a single cell. For example, source_signal_alias and source_device_alias from different signals have to be written in the same cell but separated by the separators listed below. This is useful when using the operation parameter when trying to do mathematical operations on more than one signal.

- “ “
- (newline)
- “;” “,”
- “’” “,”

# 18.3 Mathematical functions

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 implementation with many features
  - 25 predefined functions
- 18 predefined operators
  - Unit support

Use postfix operators as unit multipliers (3m -> 0.003)

## Mathematical functions

Table. 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 |
| floor | 1 | round down to the nearest integer |
| mod | variable | modulo operation |

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.

# Binary operations

Table. 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 39). 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.

# Ternary operations

Table. Supported ternary operators

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

# Examples

Users can construct their own equation by using the aforementioned operators and functions. These examples can be seen in Table below.

Table. 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 the value is greater than 5, the result should be equal to 1, otherwise equal to 0 |

Variable called value is generated or updated on every signal change and represents 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 raw, it was is presented the way it was read by an individual protocol.

# Extra functions

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/19xxxxxxx/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.

# 18.4 Uploading configuration

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

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

If a user intends to **downgrade** the 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.

## Importing an Excel file

Excel files 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 the validation of Excel configuration files.

**Generating configuration is a resource-intensive task.** It might take up to 10 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.

## Generating .zip file

To accelerate the task of generating configuration a computer can be used. For this users should download the WCC Excel Utility application. Upon opening an application, the user should search for a field called Excel file which lets to choose an Excel file for which a conversion should be made. The 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, the output file should contain the generated configuration, otherwise, an error message is shown to a user.

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

## Uploading configuration remotely

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

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

Configuration can only be uploaded if a port used for SSH connection is open for IP address filled in the 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 the upload process, the aforementioned button is disabled to prevent spanning multiple concurrent processes.

# 18.5 Virtual device

## General

The virtual device is a device that you can use to calculate additional math or keep a counter. It doesn't bind to any protocol and only works when its math expression is used.

Virtual device functionality is only available since firmware version v1.6.3, of WCC Lite.

## Configuring Virtual device

To configure WCC Lite to use the virtual device you must configure the device and signal sheets.

Virtual device parameters for Device tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| description | string | Description of the device | No | | | |
| device_alias | string | Device alias to be used in the configuration | Yes | | | |
| protocol | string | Selection of protocol | Yes | | **Virtual device** | |

Virtual device parameters for Signals tab:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |
| signal_alias | string | Unique signal name to be used | Yes | | | |
| math_expression | string | Field to calculate specific math. You must enter the signal you want to use. | Yes | | | |

The only field that is a must to use the virtual device is the math_expression field. Here you need to enter the signal which you want to associate it with. Some examples of what it can do:

- Hold a specific tag value.
- Calculate a specific math function with many signals, that you can, later on, pass to another device. Add tag value to the current value, to create a counter.

# 19 WCC Lite internal signals

## Overview

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

# Configuration

## 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.

### WCC Lite internal signals

| Parameter | Type | Description | Required | Default Value (when not specified) | Range |
|---|---|---|---|---|---|
| name | string | User-friendly device name | Yes | | |
| device_alias | string | Alphanumeric string to identify a device | Yes | | |
| protocol | | Protocol identifier Internal data | Yes | | **Internal data** |
| scan_rate_ms | integer | Update rate | No | 60000 | |
| poll_delay_ms | integer | Poll delay | No | 200 | |

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

## The signals section

tag_job defines the tag job. This can be set to gpio, board, netstat, gsm, led and process. tag_job_todo defines the job sub job. This field should address the particular point of interest. There is also an extra trigger parameter which is optional. It allows changing when the signal switches between on and off and is only applicable on LED and GPIO parameters that can be set. The default trigger is value>0. When a **trigger** column is added the trigger can be changed by entering i.e. "value>10". This is useful when mapping a source signal to for example trigger a relay. An example of how to use a trigger is in the example configuration which is attached to this page. Excel Configuration Example

⚠ Digital-input GPIO will only work with Hardware versions 1.4 and above.

⚠ Certain GSM parameters will only work if a sim card is inserted.

| job_todo | Description | tag_job_todo | Description |
|---|---|---|---|
| gpio | ReadOnly parameters | digital-input | If the value is 1 then the digital input pin is high. If it's 0 then the digital-input value is low. |
| | | rs232-enable | If the value is 1 then rs232 is enabled. If the value is 0 then rs485 is enabled. |
| | | sim-select | Switch between sim1 and sim2. If the **value** is 0 then sim1 is selected. If the value is 1 then sim2 is selected. |
| | Parameters that can be set. | | |

| | | | |
|---|---|---|---|
| | | modem-reset | Making this value equal to 1 will reset the modem. |
| | | relay | Making this value equal to 1 will activate the relay. |
| board | Board info | cpu-usage | CPU usage % |
| | | ram-usage | RAM usage % |
| | | mac-address | Device MAC address |
| | | uptime | Device uptime in seconds |
| | | fw-version | Firmware version |
| | | hw-version | Hardware version |
| | | modem-imei | Modem IMEI number |
| | | modem-type | Modem type: 0 - unknown<br>1 - single sim<br>2 - dual sim |
| netstat\|[interface] | Network statistics | TX | Bytes transferred |
| | | RX | Bytes received |
| led | LED status/control | blue-heartbeat | Heartbeat LED |
| | | blue-wlan | WLAN LED |

| | | | |
|---|---|---|---|
| | | green-eth0 | ETH0 LED |
| | | green-eth1 | ETH1 LED |
| | | green-signal1 | Signal 1 LED |
| | | green-signal2 | Signal 2 LED |
| | | green-signal3 | Signal 3 LED |
| | | red-fault | Fault LED |
| process | Check if the process is running | [process name] | 1 or 0 is returned |

| | | | |
|---|---|---|---|
| gsm | GSM information | rat-number | GSM RAT number |
| | | imsi-number | GSM IMSI number |
| | | internet-status | GSM Internet status |
| | | operator-number | GSM operator number |
| | | roaming-status | GSM roaming status |
| | | signal-quality | GSM signal quality |
| | | sim-status | SIM card status |
| date | Current set time values. | year | The current year set on the device |
| | | month | The current month set on the device |
| | | day | The current day set on the device |

| | | | |
|---|---|---|---|
| | | hour | The current hour set on the device |
| | | minute | The current minute set on the device |
| | | second | The current second set on the device |

# 20 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 realtime responses. Such functionality has long since replaced hardwired relays, timers and sequencers which would be required to complete various tasks. Programmable logic controllers usually had to conform to IEC 611313 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 611313 standard. WCC Lite supports PLC functionality while conforming to specifications of IEC 61499 standard.

# 20.1 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 similarto 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 connectionswithin 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 canbe 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.

# 20.2 4Diac framework




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.

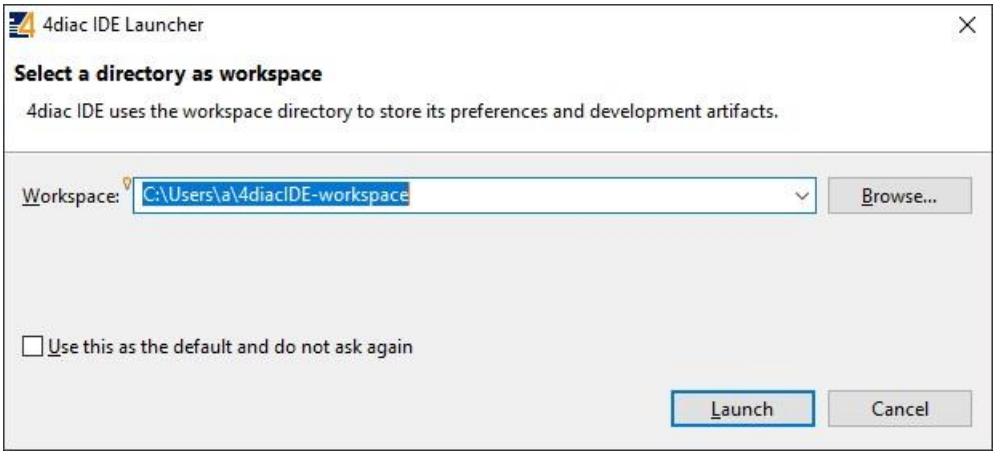
ℹ It is possible to run multiple tasks at once. These tasks can either be implemented in the same screen or split into separate tasks. Please note, however, that all elements should have unique names, even between different tasks. As of 4diac IDE 1.11.3 this is not enforced between separate apps, however,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

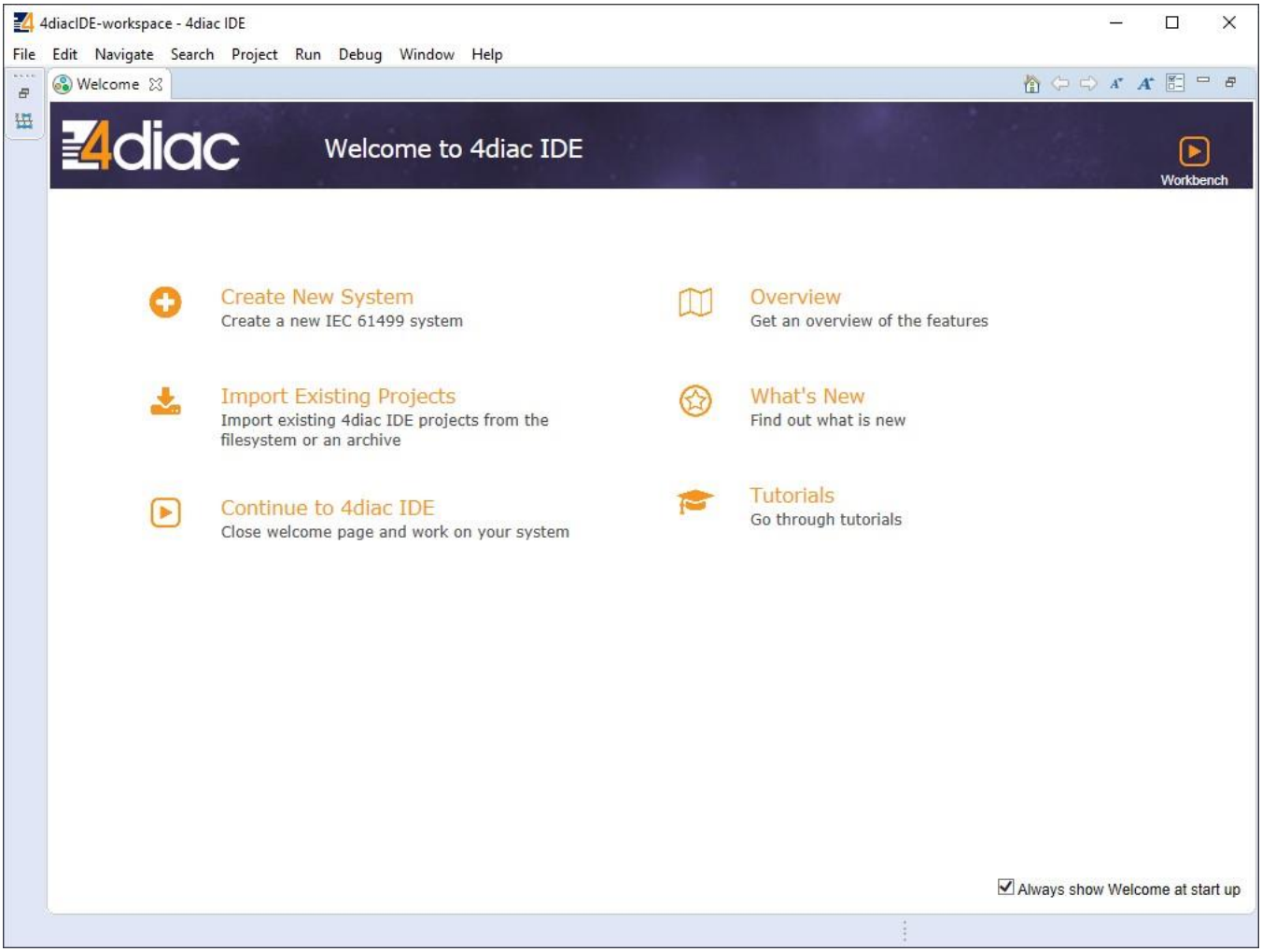
The 4diac FORTE runtime is able to execute the aforementioned fboot files containing the logic. The FORTE runtime can be run on both the WCC Lite and a PC for debugging purposes. The runtime is integrated to interact with the REDIS database.

# 20.3 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. Workspace is used to save files needed for projects.



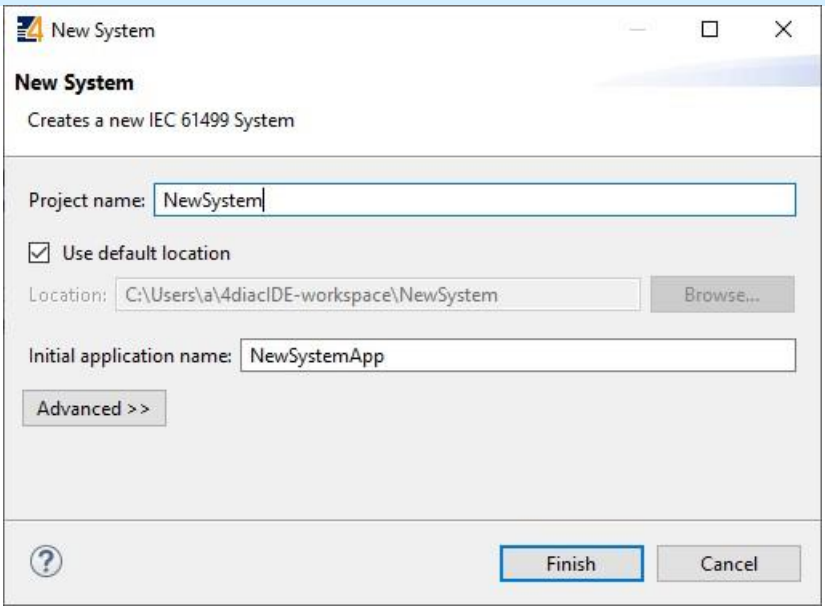


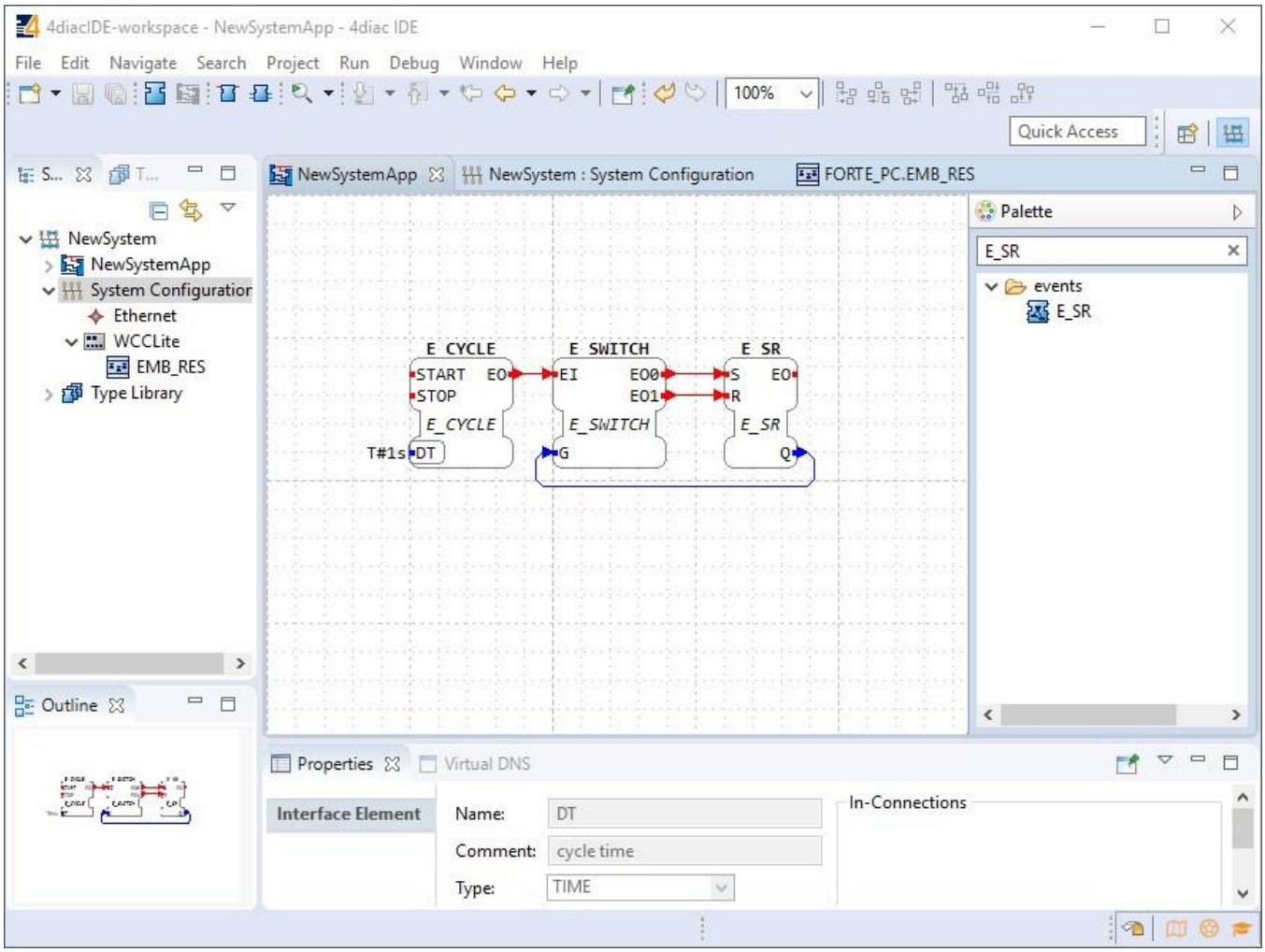

After that a user should be met by the welcome window as in Figure 20. If such window is not shown, one can create create project by selecting File->New->Project and filling in the required fields (figure 21).

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

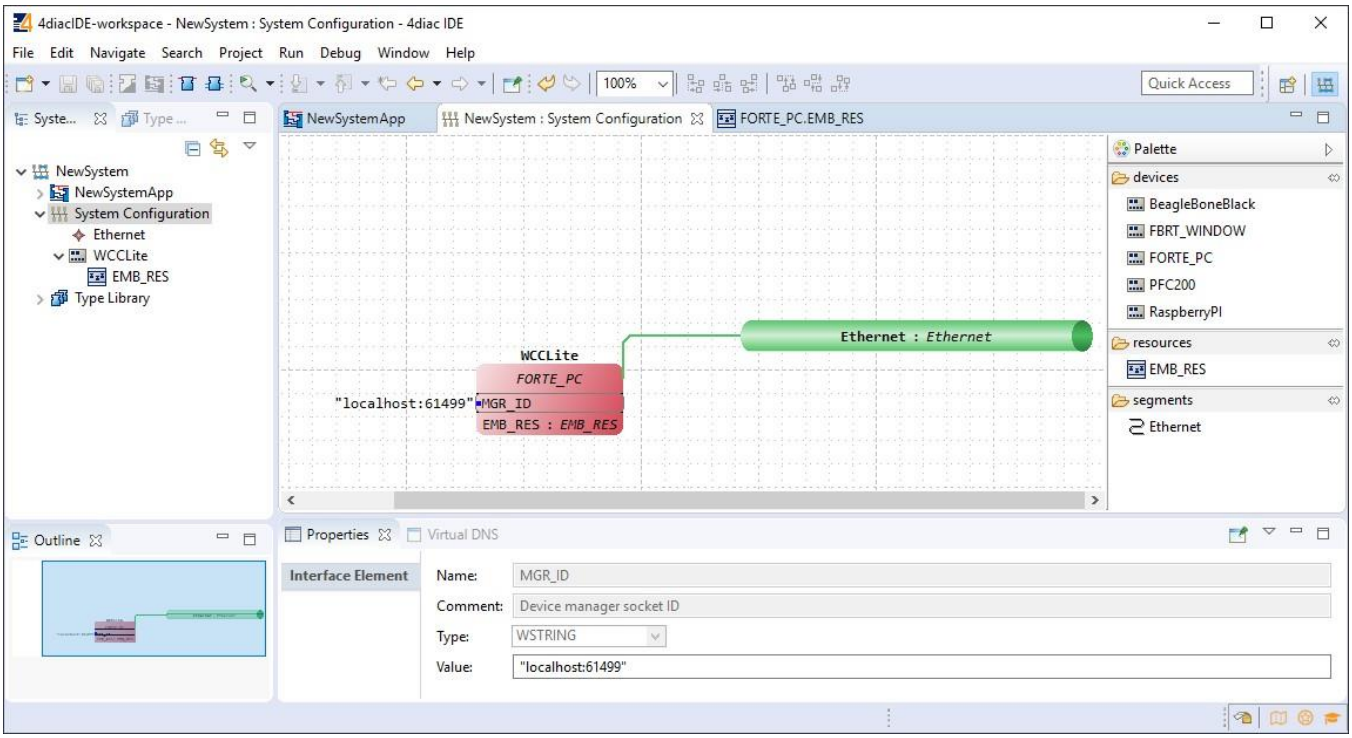
blinker application (figure 22).

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.

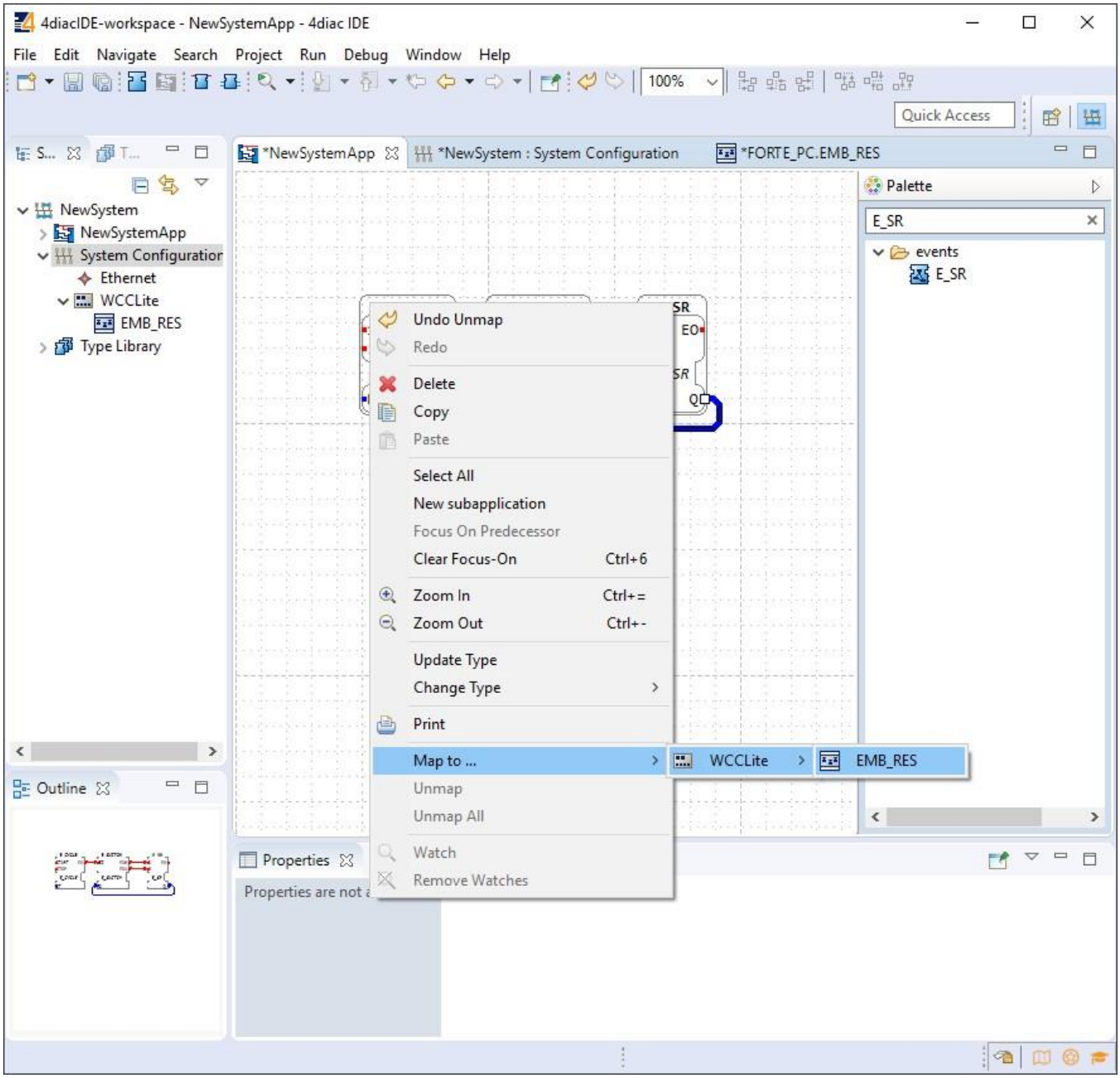


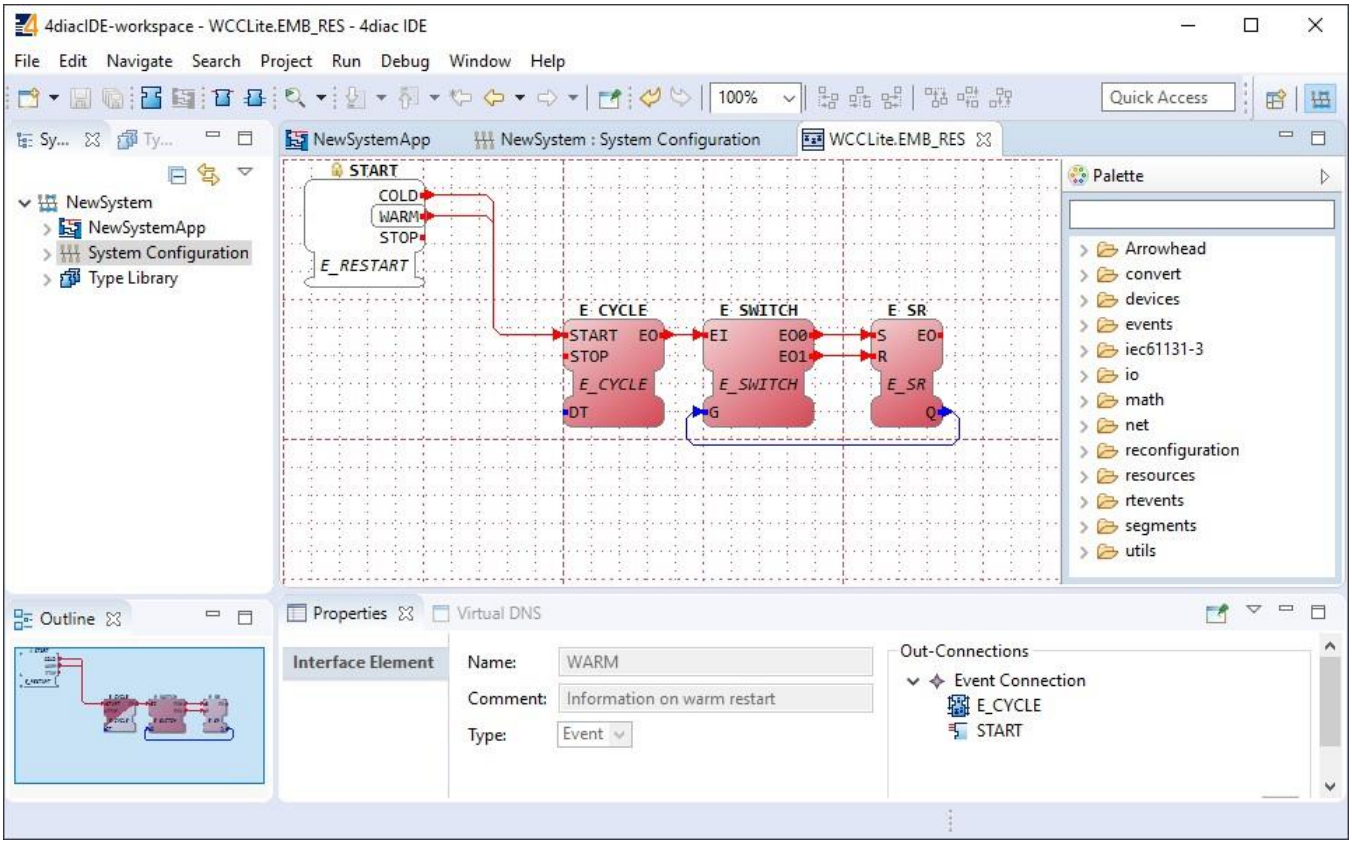
In the System Configuration section, drag and drop a FORTE_PC device, an Ethernet segment and link them (figure 23). For debugging in the local (PC) runtime, leave the address ”localhost:61499”. For testing on a WCC Lite, enter the IP address of the device, along with the port number (which by default is 61499 as well).

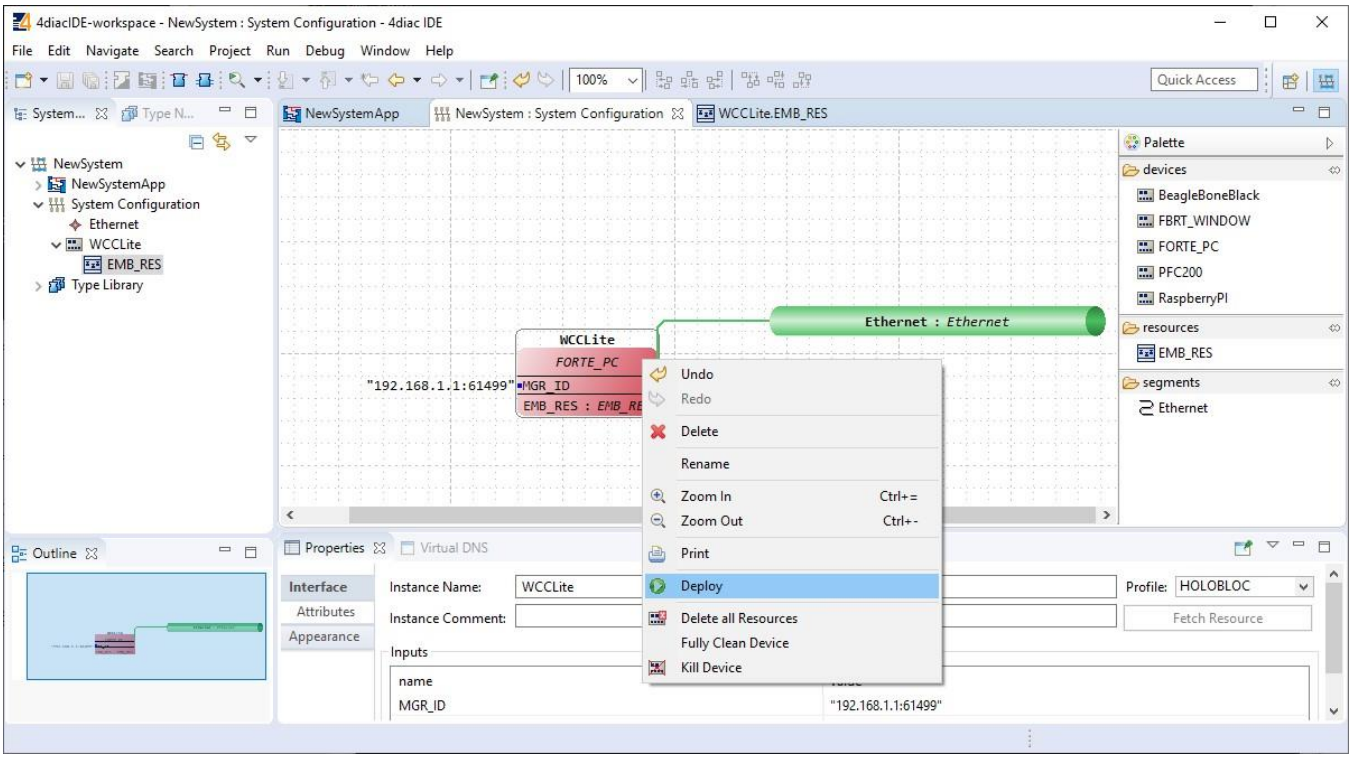



In order to deploy the application, the circuit needs to be mapped to the controller. For a non-distributed application (distributed application cases will not be discussed in this chapter), all the FBs of the application need to be selected and mapped to the configured controller as shown in figure 24.




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 25.




To deploy the application, go to the System Configuration tab and simply select ”Deploy” from the right-click menu of the controller device (figure 26). 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.

# 20.4 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. 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. Mandatory parameters for Signals tab

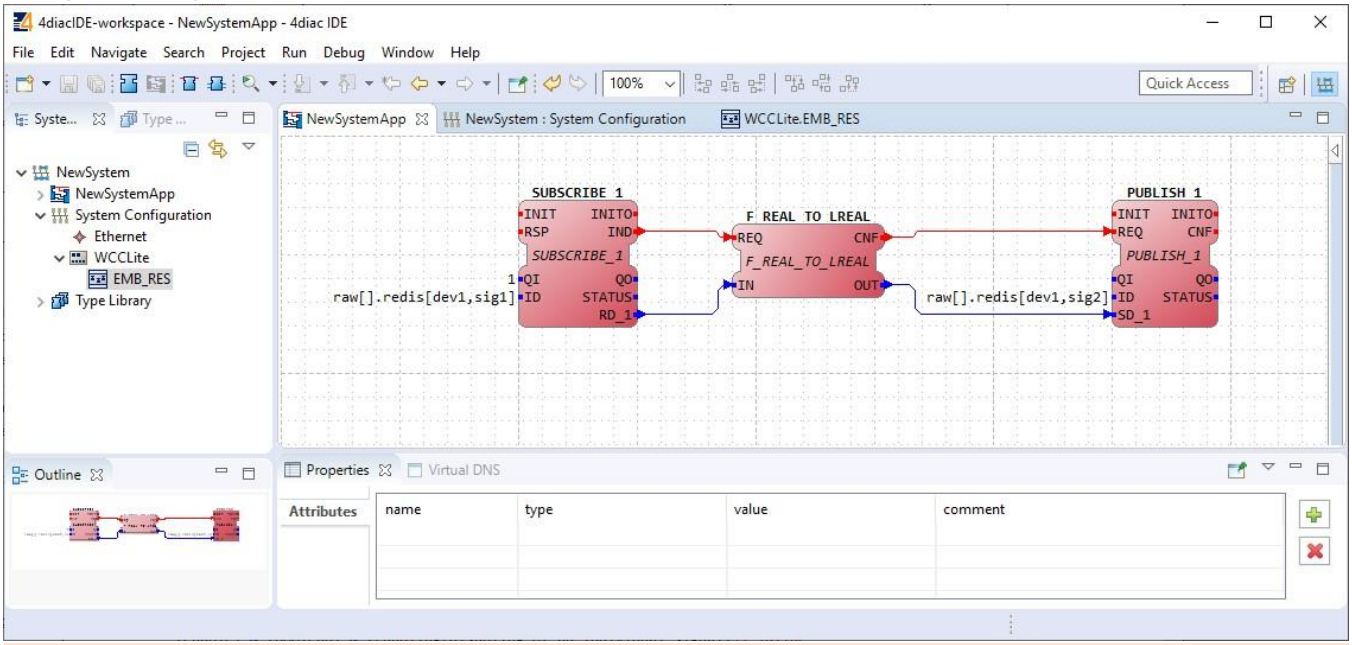
| Parameter | Type | Description |
|---|---|---|
| signal_name | string | User-friendly signal name |
| device_alias | string | Device alias from a Devices tab |
| signal_alias | string | Unique signal name to be used |
| source_device_alias | string | device_alias of a source device |
| source_signal_alias | string | signal_alias of a source signal |

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 image below.

Subscribe and publish examples




⚠ 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

⚠ No more than 20 tags should be published over a period of 5 seconds, as this may overfill the queue. A ”publish only on change” policy is advised.

⚠ Currently only PUBLISH_1 and SUBSCRIBE_1 function blocks are supported.

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

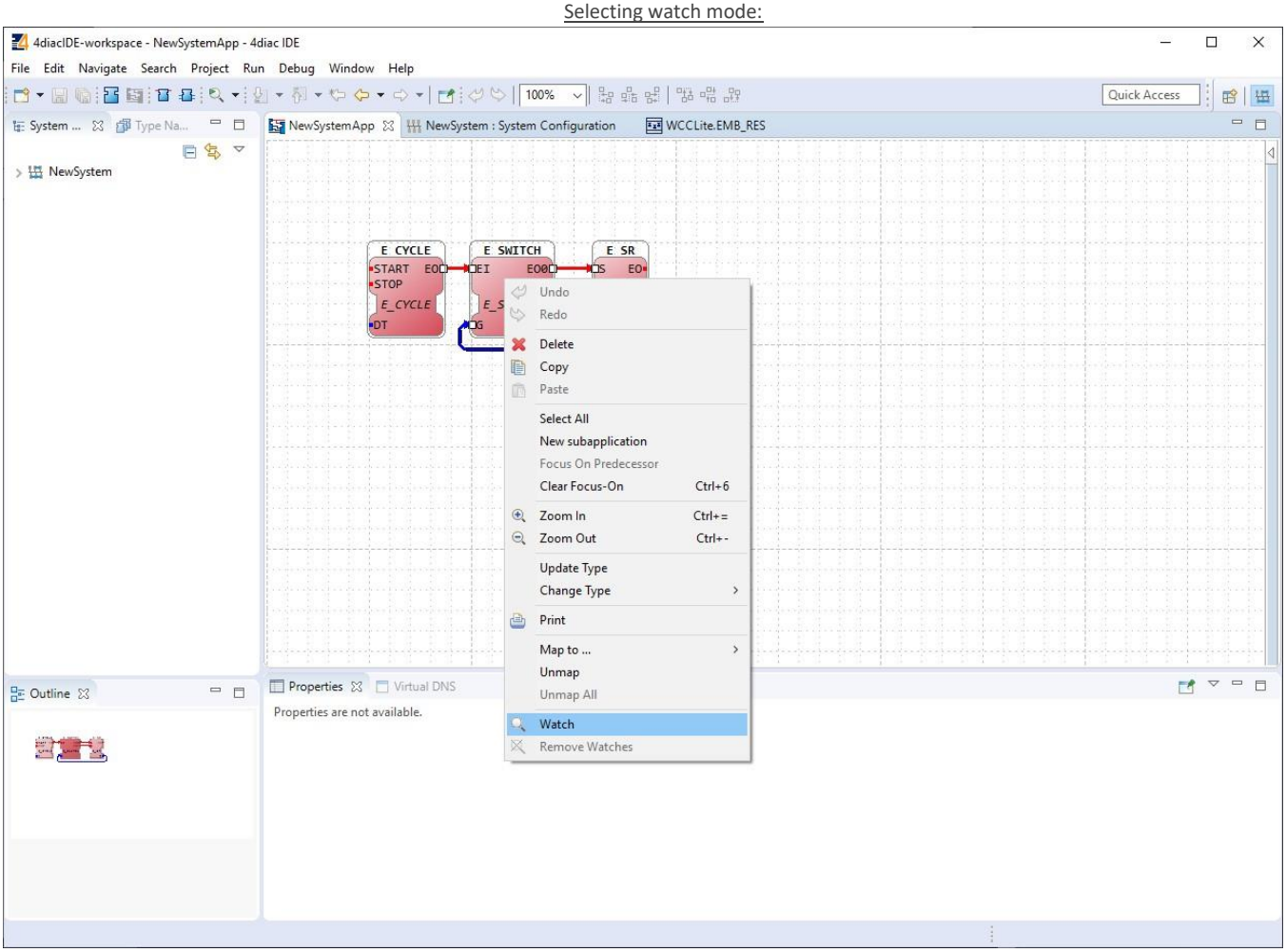
# 20.5 Debugging an IEC 61499 application

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.

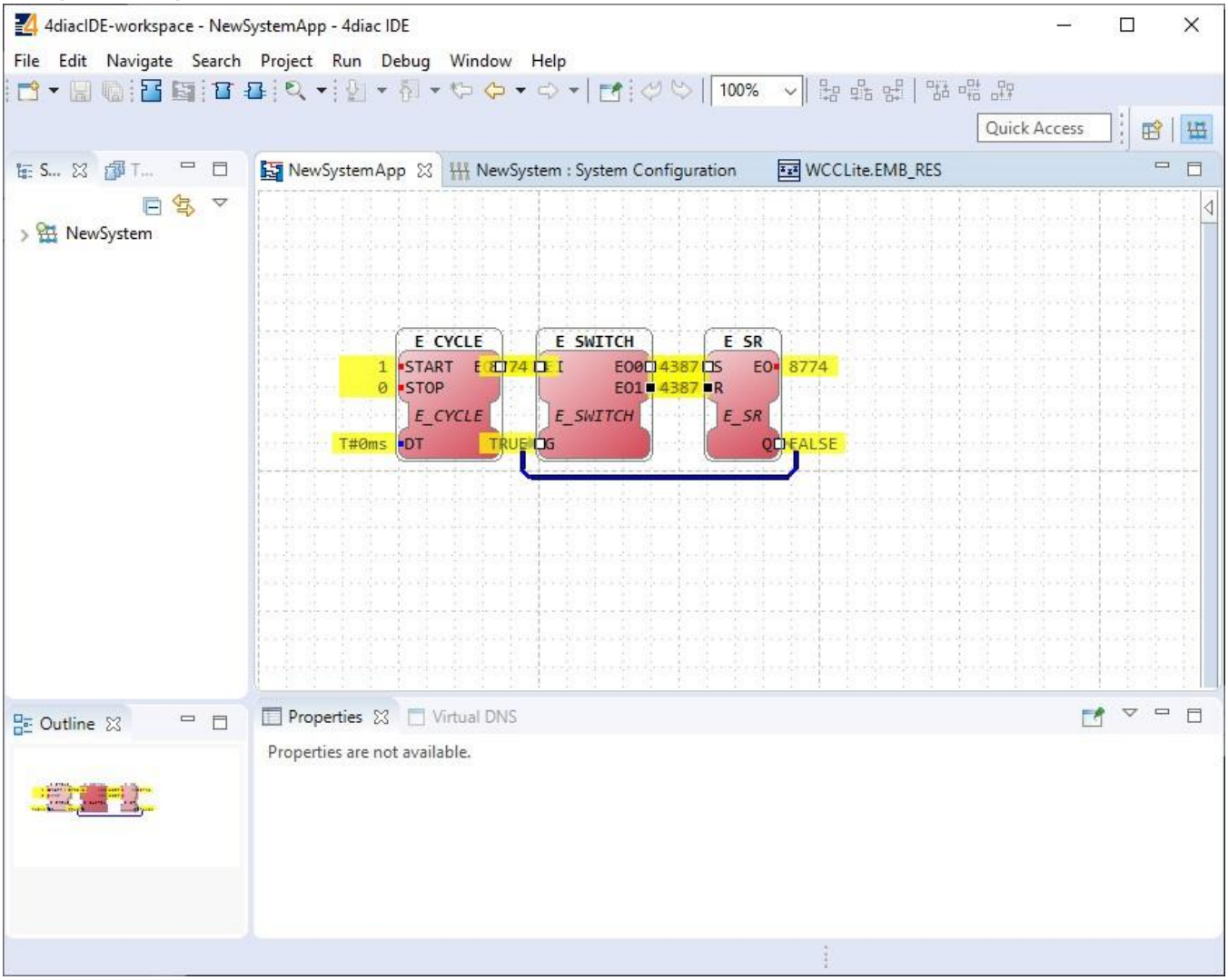
There is a possibility that 4Diac FORTE might not start as a process. It may happen if multiple faults occured and process has stopped. Process is also programmed to not start if no excel configuration file is found, therefore a user should make sure that Excel configuration is uploaded and ready for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

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

Selecting watch mode:




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 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 bellow. Make sure to stop any running process that could use the address that 4Diac framework is going to use.

Table. 4Diac FORTE command line debugging options:

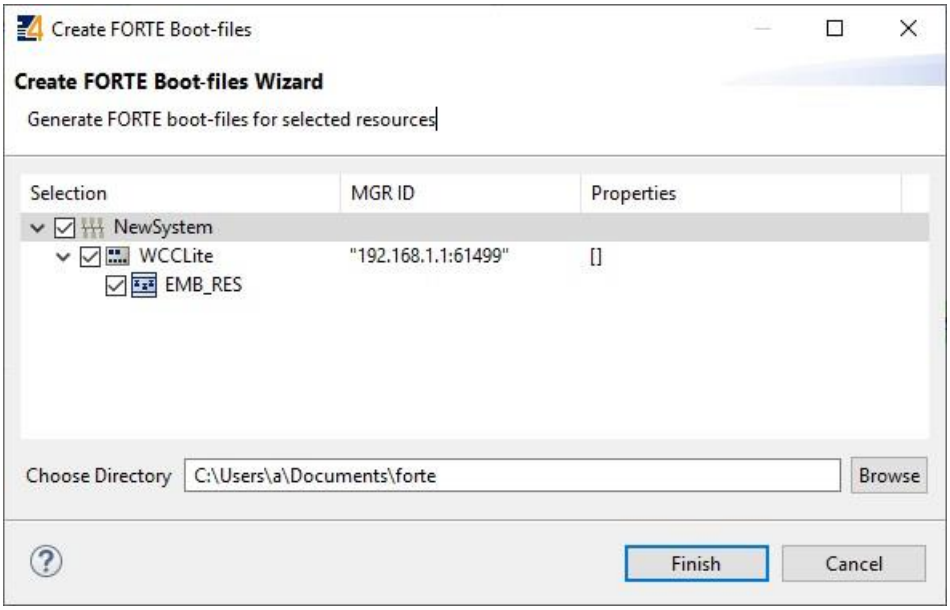
| |
|---|
| -h - Display help information |
| -c <IP>:<port> - Set the listening IP and port for the incoming connections |
| -r - Show redis messages |
| -d <debug level> - Set debugging level |
| -f - Set the boot-file where to read from to load the applications |

# 20.6 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.

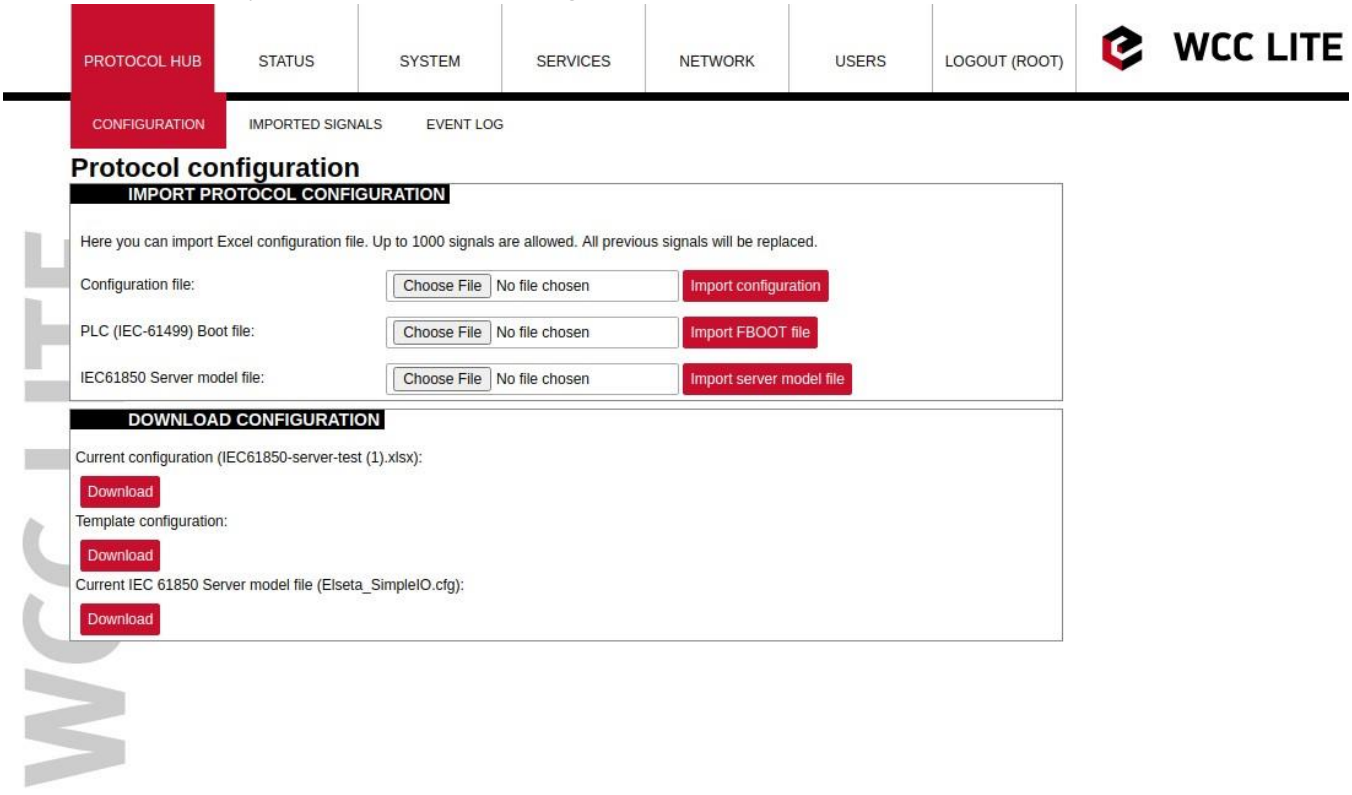
To generate FORTE boot-files a user should select Run->Create FORTE boot-file.... After that one should select devices which should have their boot files created as well as additional devices' properties and directory where these files should be stored as in picture bellow.

Generating FBOOT file:




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 picture below.

WCC Lite Web interface. 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 the picture before.

⚠ Please note that only files with *.fboot extension are allowed.

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

# 20.7 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:

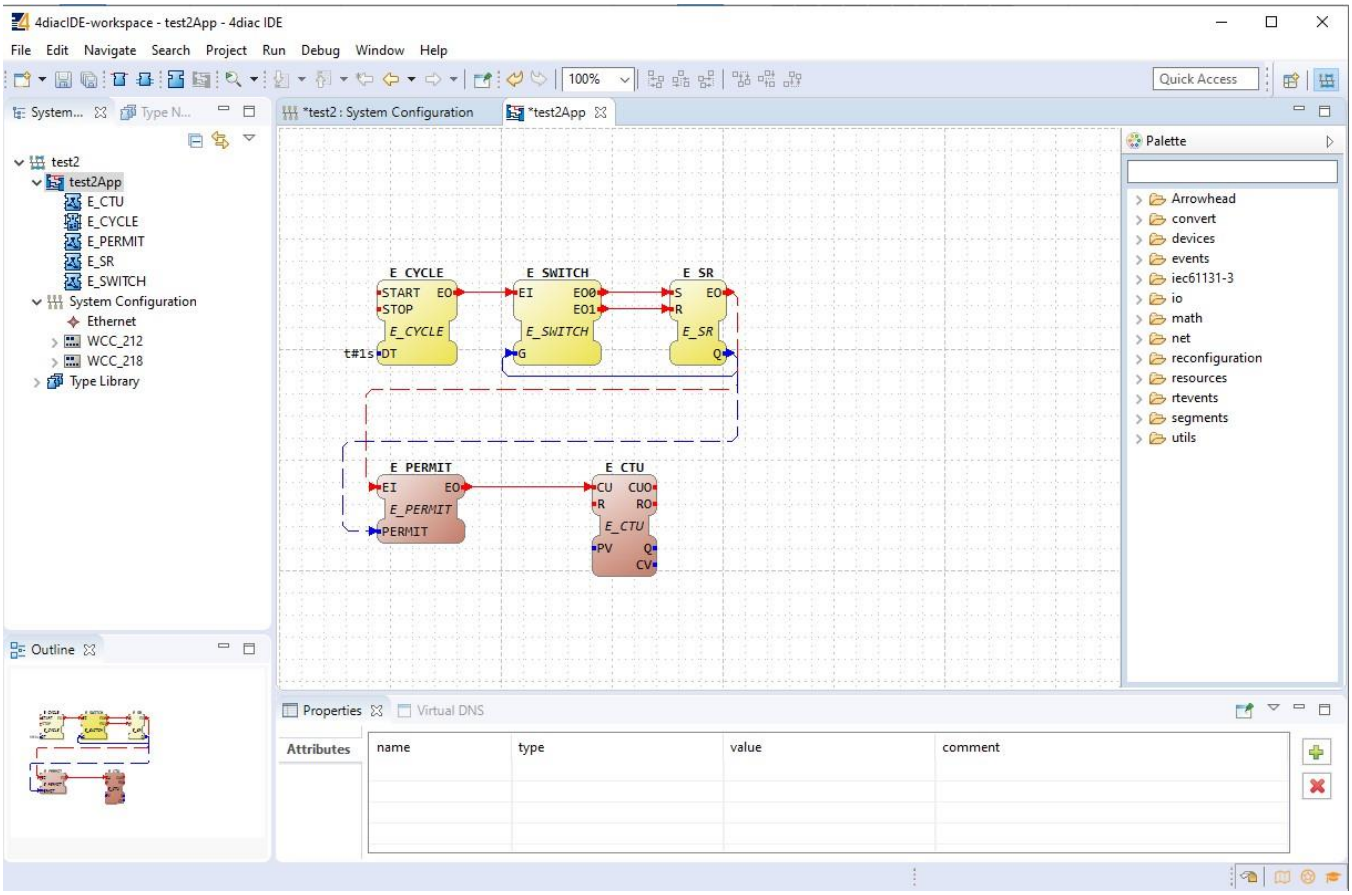
- 
- Publish/Subscribe function blocks (via UDP packets); Client/Server function blocks (via TCP packets).

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 multicast option.

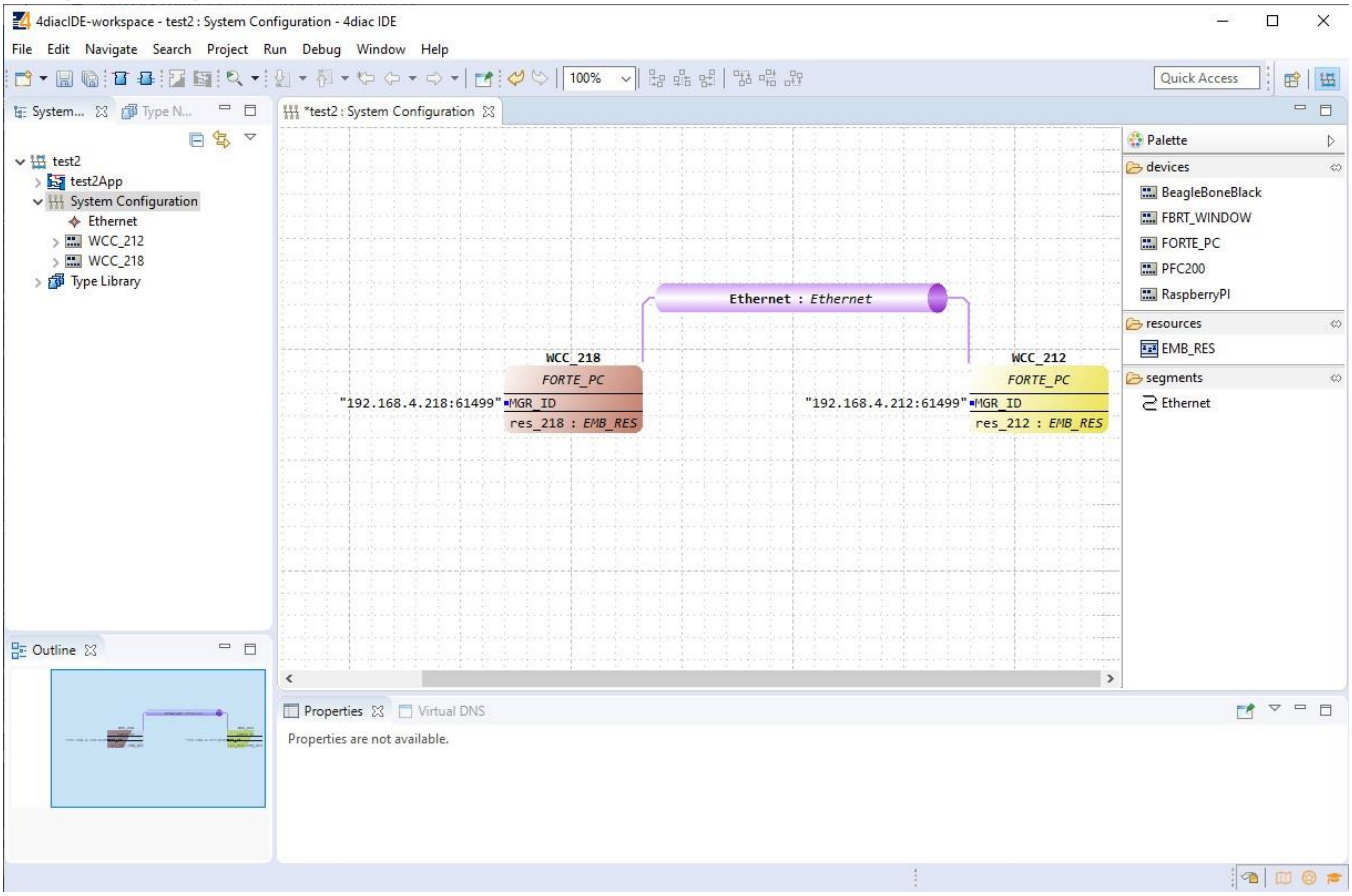
This user manual will only cover setting up point-to-point communication between devices via Publish/Subscribe blocks. For more information on communication between several IEC 61499 devices please check documentation for Eclipse 4diac framework.

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 picture below. 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. 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 below. 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).

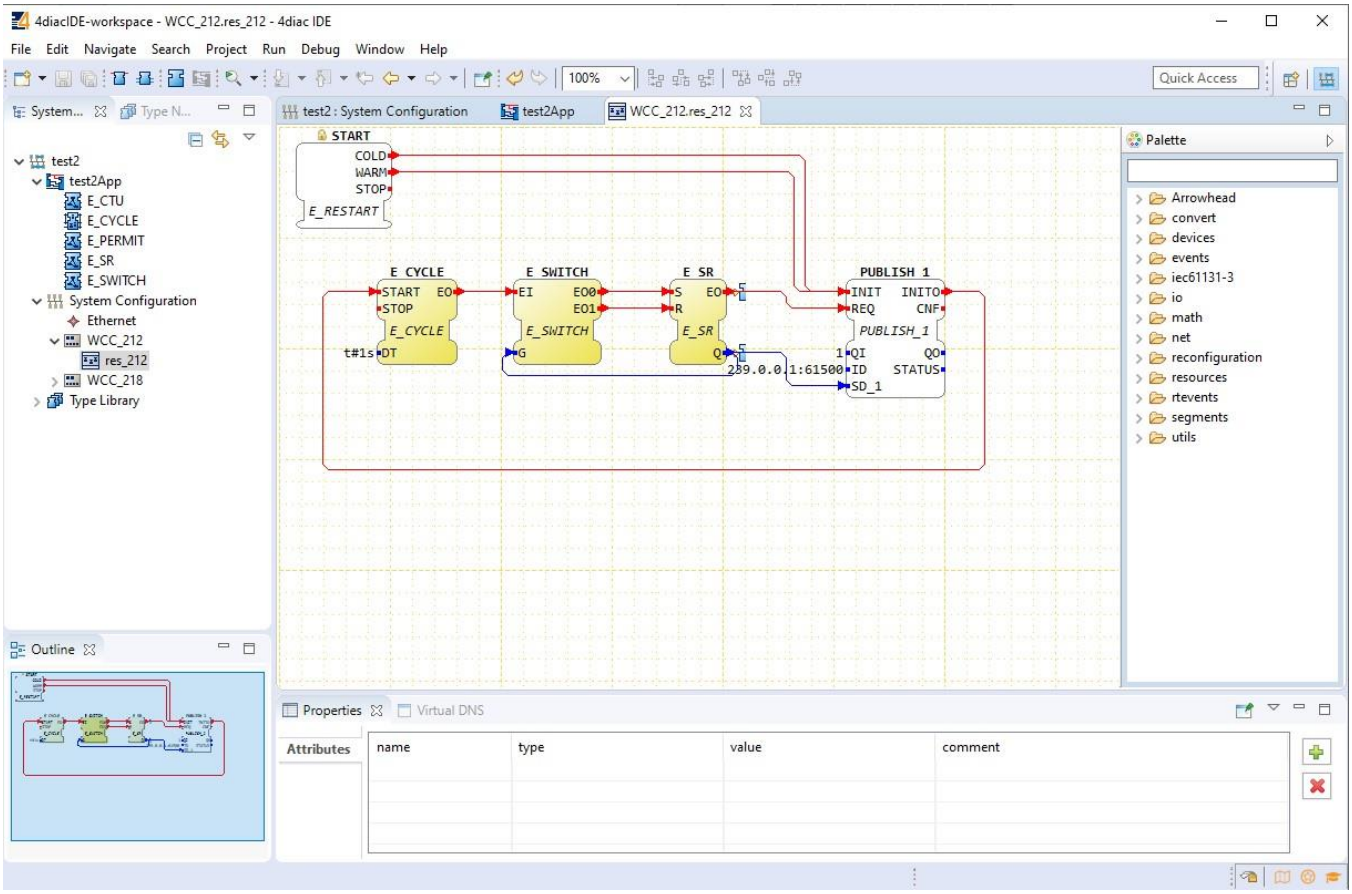
Example blinking application as a distributed system:



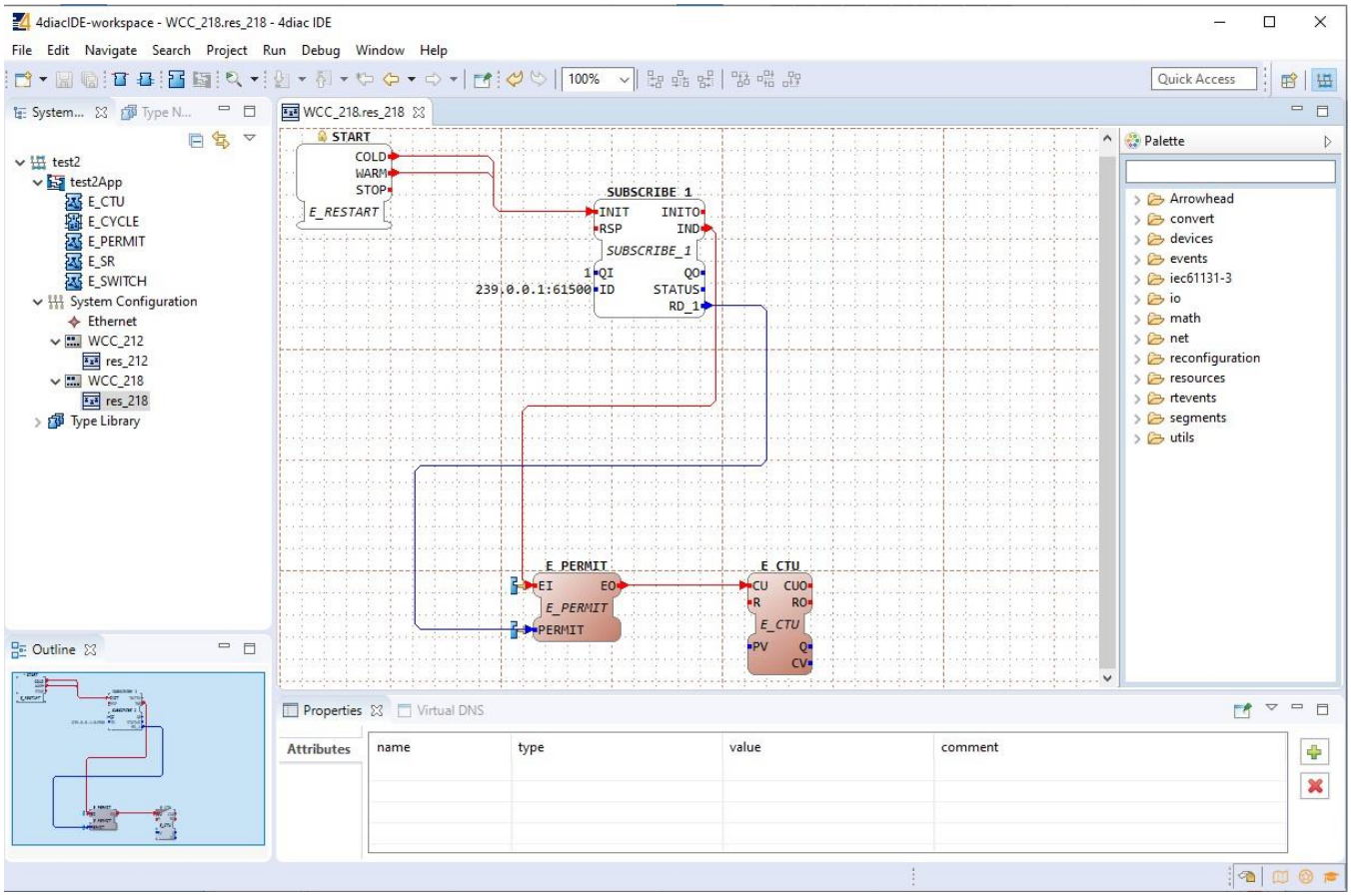

Example system configuration for a distributed system:




Example app for blinking part of a distributed system:




Example app for counting part of a distributed system:




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.

# 21 MQTT

## Introduction

MQTT (short for MQ Telemetry Transport) is an open OASIS and ISO standard (ISO/IEC PRF 20922) lightweight, publishsubscribe 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).

# Using WCC Lite as MQTT Client

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.

WCC Lite supports MQTT messaging compatible with MQTT v3.1 standard (starting from version **v1.4.0**). 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.

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 the two tables below.

Table. MQTT parameters for Devices tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| device_alias | string | Device alias to be used in the configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 0 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | MQTT | |
| ip | string | MQTT broker IP address selection | Yes | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| port | integer | MQTT broker port selection | No | 1883 | | |
| enable_threshold | boolean | A parameter to determine if identical values should not be sent multiple times in a row. | No | 1 | 0 | 1 |
| gi_interval_sec | integer | Parameter to determine how frequently should all values be sent at once. Disabled if equal to 0. | No | 0 | | |
| mqtt_qos | integer | MQTT Quality of Service for message as in standard | No | 0 | 0 | 2 |
| mqtt_retain | boolean | Selecting if MQTT broker should retain last received messages | No | 0 | 0 | 1 |
| 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) | No | 0 | 0 | 1 |
| last_will_topic | string | Topic to which an MQTT message would be sent if the device abruptly disconnected message broker | Yes (If use_last_will=True ) | | | |
| last_will_message | string | Message to be sent over MQTT if the device abruptly disconnected message broker | No | | | |
| last_will_qos | integer | MQTT Quality of Service selection as in standard | No | 0 | | |
| last_will_retain | boolean | Selecting if MQTT broker should retain last will message | No | 0 | 0 | 1 |

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 it

Table. MQTT parameters for Signals tab

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| signal_name | string | User-friendly signal name | Yes | | | |
| device_alias | string | Device alias from a Devices tab | Yes | | | |
| signal_alias | string | Unique signal name to be used | Yes | | | |
| source_device_ali as | string | device_alias of a source device | No | | | |
| source_signal_alia s | string | signal_alias of a source signal | No | | | |
| enable | boolean | Enabling/disabling of an individual signal | No | 1 | 0 | 1 |
| log | integer | Allow signal to be logged. Log signal with 1 and no logging with 0. | No | 0 | | |
| topic | string | Topic name to override the value built by default | No | | | |

# 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.

# Debugging a MQTT protocol

If configuration for MQTT 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.

MQTT Client command line debugging options

```
mqtt-client
```

```
-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
```

⚠ If MQTT Client 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.

ℹ To launch a debugging session, a user should stop mqtt-client process and run mqtt-client command with respective flags as was shown above.

# 22 Data Export

## General

Various protocols are made to transmit data points as they are generated. This is enough for a lot of systems (e.g. SCADAs) that have their own databases and devices only have to buffer fairly recent messages in case of connection or transmission errors. However, there is frequently a need to save and keep the data in files, grouped in batches, and later transmit these batches to a remote server via HTTP(S) or FTP(S). For this purpose a dedicated protocol has been created and called **Data export.**

Data export functionality is available since firmware version v1.5.0, of WCC Lite.

## Overview

Data export service gathers information from other protocols, puts it into files (optionally compressing them) after a timeout or when data buffers fill up; eventually periodically sending them to a server. HTTP(S) and FTP(S) servers with optional authentication are supported. A user can optionally choose between ISO8601 and UNIX timestamp time formats (the latter being the default value). More than one instance can set up, for instance, some of the information can be sent to an FTP server, while other could be transmitted to the HTTP server which is able to handle POST requests.

## Using WCC Lite for data export

To configure WCC Lite to use data export server a user can fill in the needed parameters in Excel configuration. These parameters are shown in two tables below. Default values are shown in a bold font.

Data export (data-export) parameters for Devices tab table:

| Parameter | Type | Description | Required | Default value (when not specified) | Range | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| name | string | User-friendly device name | Yes | | | |
| device_alias | string | Device alias to be used in configuration | Yes | | | |
| enable | boolean | Enabling/disabling of a device | No | 1 | 0 | 1 |
| protocol | string | Selection of protocol | Yes | | Data Export | |
| timeout | integer | Time frame during which transmission to remote server has to be completed (**in seconds**) | No | 5 | | |
| type | string | Selection of file format | No | csv-simple | csv-simple, json-simple, json-samples | |
| host | string | A URL of remote server where files should be sent | Yes | | | |
| upload_rate_sec | integer | Frequency of generated file uploads (**in seconds)** | No | 60 | | |
| records_buffer_siz e | integer | A maximum amount of data change entries to hold before initiating logging mechanism | No | 100 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| logging_period_se c | integer | Describes how frequently data buffer of records_buffer_siz e is saved to file | No | | 1 | |
| log_folder | string | A folder in WCC Lite file system to save generated file (**”/var/cache/dat a-export”**) | No | | | |
| timestamp | string | Selection of time format | No | | unixtimestamp, iso8601 | |
| compress | string | Selection of file compression mechanism | No | none | none, gz, tar.gz | |
| compress_passwo rd | string | Enable feature of file password protection | No | | yes, true | |
| csv_field_separato r | string | Columns separator in .csv file format | No | "," - (comma) | "," - (comma)<br>";" - (dot)<br>"." - (semicolon)<br>" " - whitespace)<br>"\|" - (pipe) | |
| csv_decimal_sepa rator | string | Decimal separator in values | No | "." - (dot) | "." - (dot) "," - (comma) | |

Same symbols cannot be selected for both csv_field_separator and csv_decimal_separator. In such case both of them will be set to default values ”.” and ”,” respectively.

It is possible that data generation rate is going to be bigger than what data buffer can hold (controlled by records_buffer_size and logging_period_sec). To make sure that no data loss occurs there’s an additional data logging call made in case data buffer reaches a records_buffer_size value.

Signals to be sent are configured differently than signals for most other protocols. As data export service only transmits signals and does no data processing, usual signal logic is not used for them. That means that:

- Signals for data export service are not seen in the Imported Signals tab;
- Signals for data export service are configured in different Excel sheet called DataExportParameters to be filled in the DataExport

  sheet are shown in a table below.

Data export (data-export) parameters for DataExport tab

| **Parameter** | **Type** | **Description** | **Required** | **Default value** (when not specified) | **Ra ıge** | |
|---|---|---|---|---|---|---|
| | | | | | Min | Max |
| device_alias | string | Device alias to be used in configuration Yes | Yes | | | |
| device_name | string | User friendly device name as in Device sheet | Yes | | | |
| tag_name | string | User friendly signal name | Yes | | | |
| source_device_ali as | string | device_alias of a source device | Yes | | | |
| source_signal_alia s | string | source_alias of a source signa | Yes | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| enable | boolean | Enabling/disabling of a measurement to be transmitted and logged | No | 1 | 0 | 1 |
| attribute | string | Additional attribute to be attached to a signal | No | | | |

# Debugging data export service

If configuration for Data export service 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.

Data export (data-export) command-line debugging options

The below-described parameters for debugging are accessible over console (SSH).

```
-h [--help] Display help information
```

```
-c [--config] Configuration file location
```

```
-V [ –version ] Show version
```

```
-d<debug level> [ –debug ] Set debugging level
```

```
-R [ –readyfile] Ready notification file
```

```
-p [ –http ] Show HTTP messages
```

```
-r [ –redis ] Show Redis output
```

If Data export service does not work properly (e.g. data is corrupted, etc.), a user can launch a debug session from command line interface and find out why it is not functioning properly. To launch a debugging session, a user should stop data-export processes and run data-export command with respective flags as in table above.

# Host URL format rules

Parameter host is highly configurable and might contain a considerable amount of information:

- Protocol - FTP or HTTP (encrypted and encrypted);
- URL address - both resolved and non-resolved;
- Authentication - pair of user and/or password;
- Port - useful when non-standard value is used;
- Endpoint - a place in server to which a call is made

The format for host parameter can be summarized as:

```
[http(s)/ftp(s)]://[[user]:[password]@][URL][:port]/[endpoin
]
```

Options are printed in square brackets. A protocol has to be selected, otherwise HTTP will be used as a default. User and password pair is optional, but if user:password pair is used, it should proceeded with @ sign.

HTTP and FTP use default or user assigned ports. By default HTTP uses port 80, while HTTPS uses port 443, FTP sends data over port 21, FTPS - over port 990. Make sure that these ports are open in firewall on both server and client side, otherwise data will not be sent succesfully.

Finally, POST request (for HTTP) or upload (for FTP) can be made to a specific place (endpoint). This endpoint should be described after a URL and port (if used).

# Format of exported data

For a server to interpret data, a set of rules for a file format have to be established.

**Csv-simple** format applies to all files by default and is used as in this example:

```
###DUID:3182121xx
#device name; tag name; value; quality; timestamp; atribute inv1;Ia;236.9,1;1599137189963;Pa
```

Example of additional format csv-periodic:

```
###DUID:318212xxx
##DEVICE:inv1
#Time;Upv1;Upv2;Upv3;Upv4;Upv5;Upv6;Ipv1;Ipv2;Ipv3;Ipv4;Ipv5;Ipv6;Status;Error;Temp;cos;fac;Pac;Qac;Eac;E-Day;E-
Total;Cycle Time
2020-09-
02T15:45:00Z;462.3;462.3;370.2;370.2;371.2;371.2;1.40;1.43;1.35;1.47;1.21;1.26;512;0;26.3;1.000;50.00;3.217;-
0.029;0.28;17.41;54284.53;5;
2020-09-
02T15:40:00Z;462.3;462.3;370.2;370.2;371.2;371.2;1.40;1.43;1.35;1.47;1.21;1.26;;512;0;26.3;1.000;50.00;3.217;-
0.029;0.28;17.41;54284.53;5;
##DEVICE:meter
#Time;Uab;Ubc;Uca;P;Q;S;F;eTOT;Cycle Time
2020-09-02T15:45:00Z;421.3;421.3;421.3;15000;100;15600;50;246894;5;
2020-09-02T15:40:00Z;421.3;421.3;421.3;15000;100;15600;50;246895;5;
```

Example of additional format json-simple:

```
{
 "metadata": {
  "duid": "318xxxxx",
  "name": "hostname",
  "loggingPeriod": "15min",
  "format": "json"
 },
 "data": [
  {
   "tag_name": "Ia",
   "device_name": "inv1",
   "attribute": "Pa",
   "last": { "value": 12.2, "timestamp": 1213123 },
   "min": { "value": 12, "timestamp": 1213123 },
   "max": { "value": 12, "timestamp": 1213123 },      "avg": { "value": 12, "timestamp": 1213123 }
  },
  {
   "tagName": "Ib",
   "deviceName": "inv1",
   "attribute": "Pb",
   "last": { "value": -12.3, "timestamp": 1213123 },
   "min": { "value": 12, "timestamp": 1213123 },
   "max": { "value": 12, "timestamp": 1213123 },      "avg": { "value": 12, "timestamp": 1213123 }
  },
 ]
}
```

Example of additional format json-sample:

```
{
  "metadata": {
   "duid": "318xxxxx",
   "name": "hostname",
   "loggingPeriod": "15min",
   "format": "json-samples"
  },
```

```
  "data": [
   {
    "tag_name": "Ia",
```

```
      "device_name": "inv1",
      "attribute": "Pa",
      "timestamp":{
         "first": 123123,
         "last": 123236
      },
      "first": { "value": 12.2, "timestamp": 1213123 },
      "last": { "value": 12.2, "timestamp": 1213123 },
      "min": { "value": -12, "timestamp": 1213123 },
      "max": { "value": 12, "timestamp": 1213123 },        "avg": { "value": 12, "timestamp":
1213123 },
      "samplesCount": 2,
      "samples": [
         {"value": 12, "timestamp": 1213123, "quality": true},
         {"value": -12.3, "timestamp": 1213123, "quality": true}
      ]
    }
   ]
  }
```

# 23 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 IEC60870-5-104 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.

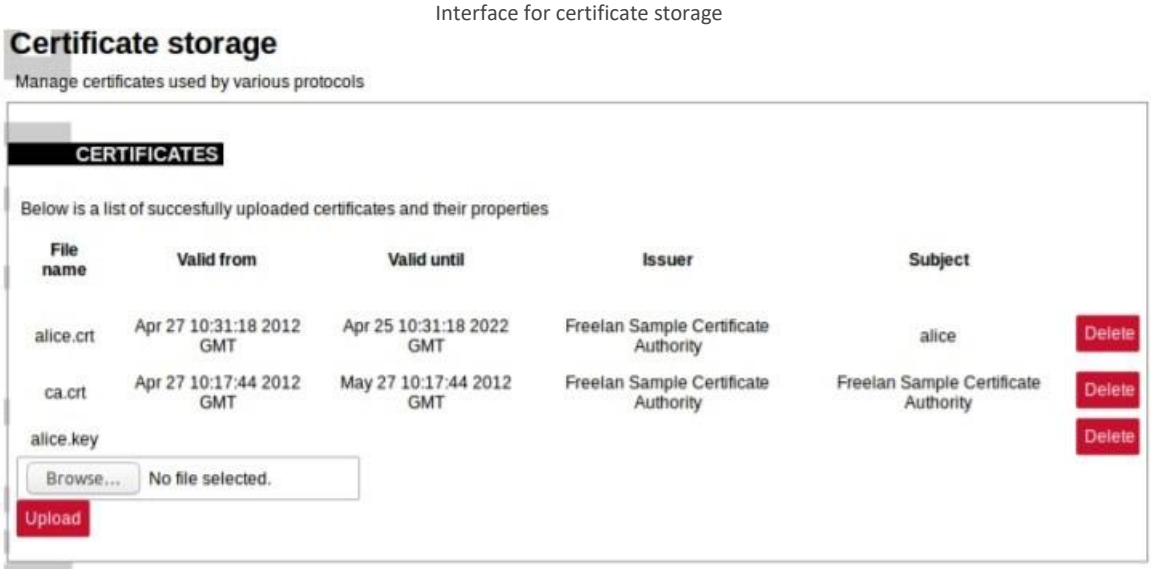
Certificates should conform to the X509 standard. The difference between local certificate and certificate authority certificates is that only certificate authority generates certificates for others. Therefore Issuer and Subject fields are always the same for certificate authority certificate whereas 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.

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

Interface for certificate storage




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.

# 24 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.

# 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.

## User management and rights authentication

WCC Lite provides different authentication mechanisms:

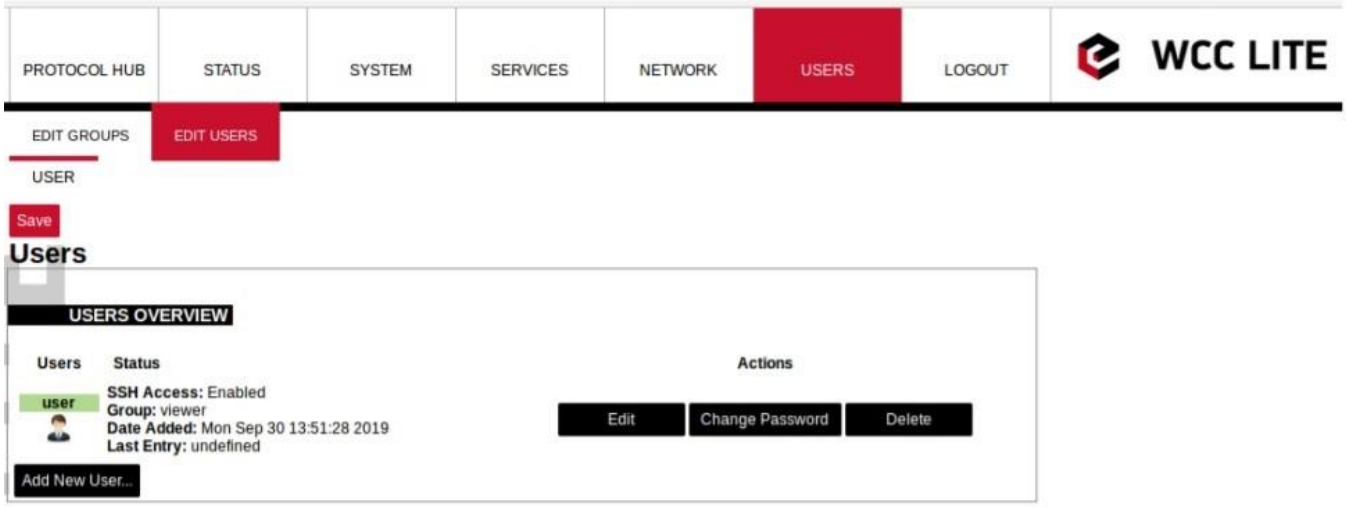
- Authentication via locally stored credentials. In this scenario all users, passwords and permissions are encrypted and stored in internal WCC Lite storage.
- Authentication via external RADIUS Server. In this scenario all users, passwords and permissions (profiles) are defined in remote RADIUS Server. 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.

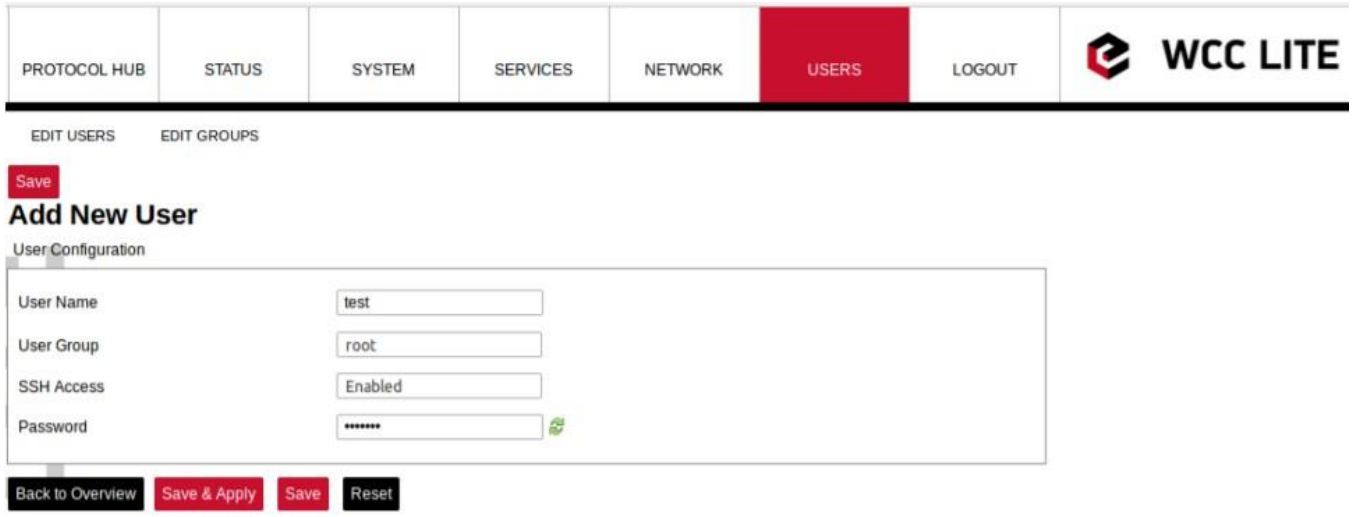
## Locally stored credentials management

Device has predefined default users like root and user.

Screen containing all users




Screen for new user configuration

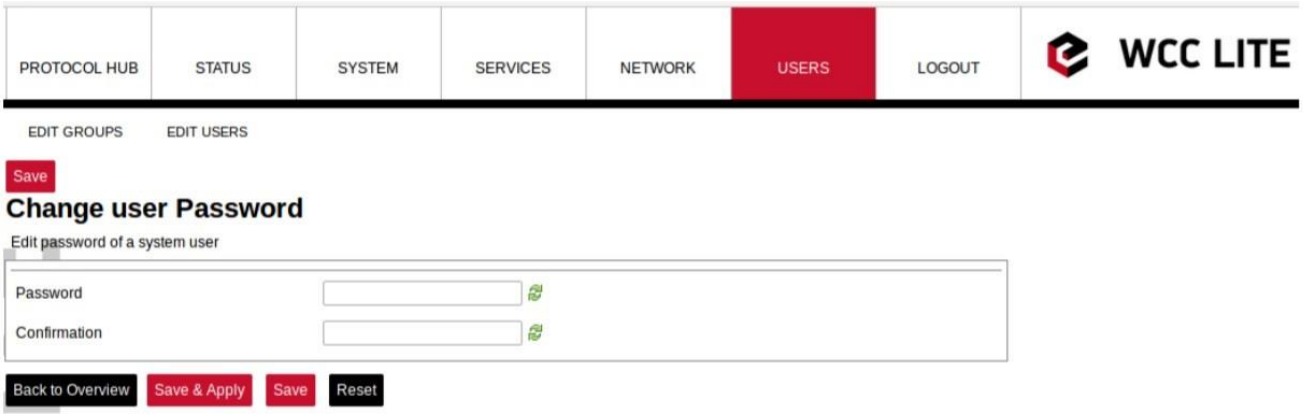



**root** user has full permission set to connect to WCC Lite over web interface and SSH or Telnet. This user is default user on WCC Lite and cannot be deleted. However, it is highly advised to change the default password to a different one less susceptible for attacks.

**user** is limited user on system and can't get root rights. A default password for access via commandline 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 the figures above. 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 figure above to lead user to a screen seen in the figure below.

Changing user password



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 commandline interface should be given by a root user via commandline interface.

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

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

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 commandline 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

## Authentication via external service

WCC Lite support external authentification via RADIUS service. Remote Authentication DialIn 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 backend 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.

## 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 60870510x 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.

## 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 bruteforce 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:

- not offer access from the Internet at all, or restrict it to certain IP addresses or IP address ranges by letting the SSH
  - server dropbear and the webServer lighttpd not listen on theexternal/WAN port by blocking incoming connections to
  - those ports (TCP 22, 80 and 443 by default) in yourfirewall
- make it more difficult to guess: don't
  - use the username root
  - don't use a weak password with 8 or less characters don't let the SSH server
- dropbear listen on the default port (22)
  - use the combination of:
  - username different than root tell dropbear to listen on a random port (should be >1024): **System > Administration**
  - **>Dropbear Instance > Port** 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.

# 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 CISCOlike 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

## 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.

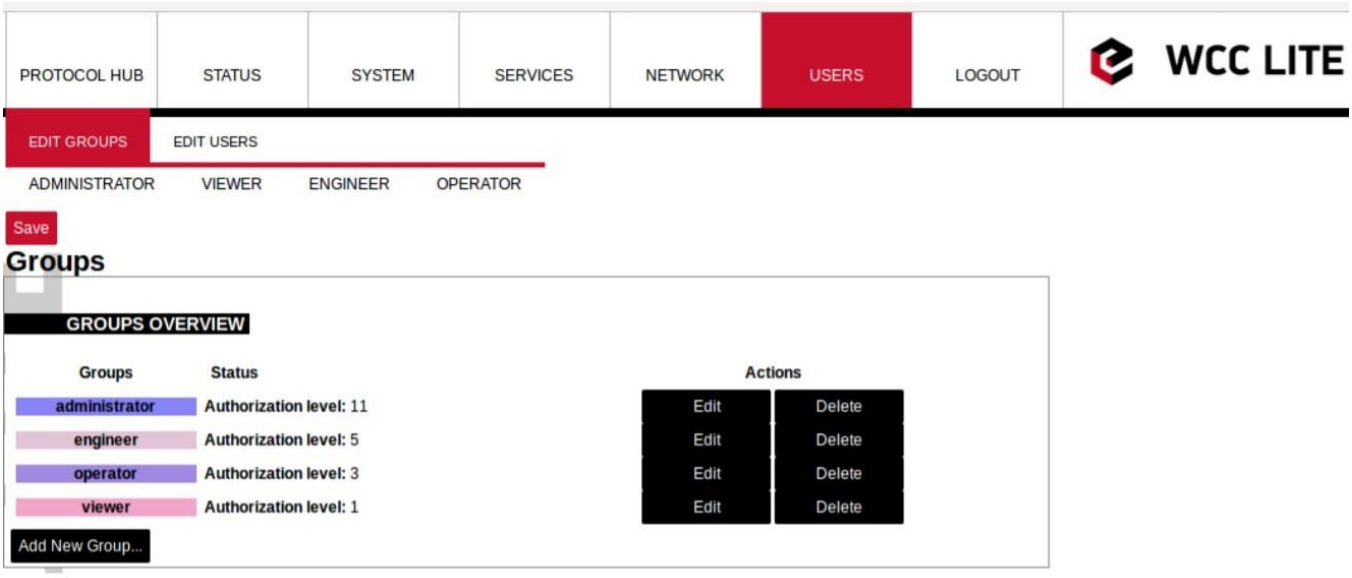
If a user intends to change permissions for user groups, it should be done via command line interfaces. It is only advised for advanced users.

## Web interface permissions

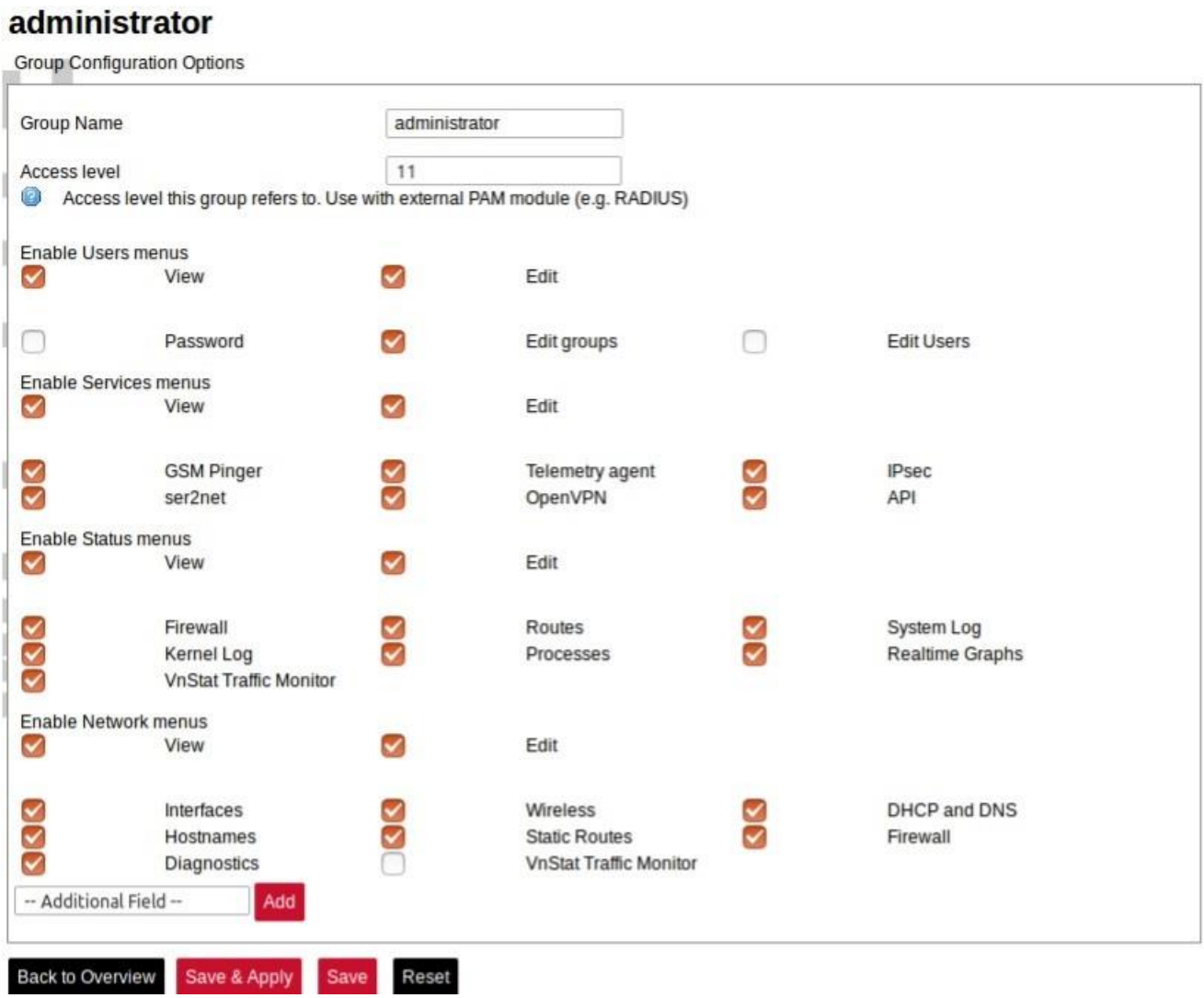
Fifteen levels of authorization permission are mapped for web interface access, meaning that user should be able to access web interface with credentials used to log into command line interface. User assigned to a highest authorization level group is able to access every possible screen therefore this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

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

Screen showing existing user groups




Screen for user group editing




Edit group screen for an individual group can be seen in Figure above. 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 groups' 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

# 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 608705 series, IEC 608706 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.

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 | Yes | >=1.5 |
| | | TLS (RFC 2246)is inserted between RFC 1006 & RFC 793 to provide transport layer security | Yes | >=1.5 |
| IEC 62351-5 | Security for any profiles including IEC 608705 | 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 | |
| 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 rolebased 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 wellproven 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 | |

# Information about the equipment manufacturer




**Office address:**
L. Zamenhofo g. 3
LT-06332 Vilnius
Lithuania
Tel.: +370 5 2032302
Email: info@elseta.com
Support email: support@elseta.com
In the web: elseta.com
Work hours: I-V 8:00 - 17:00