



ELSETA

WCC Lite

User manual

Elseta

2024/01/04

Doc version: 1.1.18

HW version: WCC Lite+

FW version: 1.1.18

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 has adequate engineering knowledge.

It is crucial to adhere to the laws and regulations of the jurisdiction where 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 the 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.

To prevent damage to the equipment, please carefully check the power supply, input and output ratings, and operating conditions. Failure to observe this information may render input, output, or the whole device inoperable and void the 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 disaster

WCC Lite+

This manual is for WCC Lite+ software version v1.1.18

Contents

COPYRIGHTSANDTRADEMARKS	2
SAFETY PRECAUTIONS	3
WARRANTY	3
1 Overview	8
2 Technical information	8
3 Installing the WCC Lite+	10
3.1 Hardware and software requirements	10
3.2 Mounting	10
3.3 Power	11
4 Interfaces	13
4.1 Serial port interfaces	13
5 Internal web page	13
5.1 Initial Setup	14
5.1.1 Static IP Address Setup on Windows	14
5.1.2 Connecting to an internal web page	15
5.2 Site layout	17
5.3 Protocol Hub	17
Configuration	17
Imported Signals	18
Event Log	18
Protocol Connections	19
Script-Runner	19
5.4 Status	19
Overview	19
Firewall	21
Routes	22
System Log	23
Kernel Log	24
Processes	24
Chronos	25
Realtime graph	25
VNSTAT Traffic monitor	26
5.5 System	27
System	27
Software	32
Startup	33
Scheduled tasks	34
Certificate storage	34
Time sync	35
Backup/flash firmware	35
Reboot	37
5.6 Services	37
Telemetry agent	38
1.1.18	4

IPsec	38
L2TP/IPsec	42
API	43
OpenVPN	44
ser2net	45
SNMP	45
5.7 Network	46
Interfaces	46
DHCP and DNS	50
Hostnames	53
Static routes	53
Diagnostics	54
Firewall	54
5.8 Users	59
Edit groups	59
Edit users	60
Password	60
5.9 Logout	61
6 Excel Configuration	61
6.1 Devices sheet	61
Optional settings	62
Serial port settings	62
TCP/IP settings	62
6.2 Optional parameters for signals	63
Required attributes	63
Optional attributes	63
Signal recalculation operation priority	64
Linking signals	66
Separators	66
6.3 Mathematical functions	66
Mathematical functions	67
Binary operations	68
Ternary operations	69
Examples	69
Signal mathematics	69
Command mathematics	72
Extra functions	76
6.4 Uploading configuration	76
Importing an Excel file	76
Generating .zip file	77
Uploading configuration remotely	77
6.5 Virtual device	77
General	77
Configuring Virtual device	77
7 WCC Lite+ internal signals	78
Overview	78
Configuration	78
Devices section	78
The signals section	78

8 Tags	80
Single point.....	80
Double point	80
9 API.....	80
Authentication	80
10 Certificates	82
11 Cyber security.....	83
User rights	83
Groups rights.....	86
Conformance to IEC 62351 standard	88
12 DNP 3.0	90
12.1 Introduction	90
12.2 DNP 3.0 Master.....	90
Configuring datapoints.....	90
Debugging the DNP3 Master service	95
12.3 DNP 3.0 Slave	96
DNP3 Slave parameters for Signals tab	98
Device status signals	99
Debugging the DNP3 Slave service	100
13 Modbus	101
13.1 Introduction	101
13.2 Modbus Master.....	101
13.3 Modbus Slave	105
14 IEC 60870-5-10X.....	108
14.1 Introduction	108
14.2 IEC 60870-5-101 Master.....	108
Configuring datapoints (master)	109
Debugging an IEC 60870-5-101 Master application	113
14.3 IEC 60870-5-101 Slave.....	113
Configuring datapoints (slave).....	113
Debugging an IEC 60870-5-101 slave application.....	116
14.4 IEC 60870-5-103 Master.....	116
IEC 60870-5-103 Master Configuring datapoints.....	117
Debugging an IEC 60870-5-103 Master application	120
14.5 IEC 60870-5-104 Master.....	120
Configuring IEC 104 Master data points	120
Debugging an IEC 60870-5-104 Master application	123
14.6 IEC 60870-5-104 Slave.....	123
Configuring IEC 104 Slave data points	124
Debugging an IEC 60870-5-104 Slave application	126
15 IEC 61850	127
15.1 Introduction	127
15.2 IEC 61850 Server.....	128
Commands	128
Configuring data points.....	128
Converting and uploading data model	131
Debugging an IEC 61850 server application	132
15.3 IEC 61850 Client.....	132
Acquiring data via report control blocks.....	132
1.1.18	

Controlling remote equipment via commands	134
16 Specific protocols	140
16.1 At command	141
16.2 Aurora	143
16.3 COMLYNX	144
Overview	144
16.4 Delta	146
16.5 GINLONG	148
Overview	148
16.6 Kaco	150
16.7 POWERONE	151
Overview	151
Configuration	151
16.8 SMA NET	153
Overview	153
Configuration	153
16.9 SOLPLUS	155
Configuration	155
16.10 VBUS	157
Configuration	157
16.11 VESTAS	159
Configuration	159
16.12 Windlog	161
16.13 M-Bus	162
16.14 KOSTAL	164
Configuration	164
17 Metering protocols +	166
17.1 DLMS/COSEM	166
Introduction	166
DLMS Master	166
17.2 IEC 62056-21	170
Introduction	170
Overview	170
Configuration	171
17.3 Elgama	173
18 SNMP	175
19 Data Export	175
General	175
Overview	175
Using WCC Lite+ for data export	176
Debugging data export service	178
Host URL format rules	178
Format of exported data	178
20 Programmable logic controller	180
20.1 IEC 61499	180
20.2 4Diac framework	181
20.3 Example project	181
20.4 Configuring data endpoints	187
20.5 Debugging an IEC 61499 application	189

20.6 Generating and uploading FORTE logic file	190
20.7 Distributed control application	191
21 MQTT.....	194
Introduction	194
Using WCC Lite+ as an MQTT Client	195
MQTT data format	197
Debugging a MQTT protocol.....	197
22 Lua script runner	198
Introduction	198
Overview	198
Configuration.....	201
Debugging the script runner service	202
Information about the equipment manufacturer	203

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 Technical information

System	
Processor	Intel ATOM CPU
Memory	128GB Storage memory/4GB RAM
Ethernet RJ45 connector up to 3 independent ports	3 x 2.5 GbE (2500Base-T / 1000Base-T / 100base-T)
Serial ports	2 x RS232 / 422 / 485
Power requirements	
Power supply	9 - 33 VDC
Dimensions	141 W x 111.2D x 71.9H mm
Weight	1.25kg
Mounting	Din rail
Environmental	
Operating temperature	-40°C to +85°C
Warranty	2 years
Software	

Routing	<ul style="list-style-type: none"> • Isolated LAN interface • Isolated LAN interface, but omitted to provide TCP / UDP ports or VPNmergers • Secure LAN data transfer via VPN • Secure LAN data transmission through VPN access to the Internet
Database	<ul style="list-style-type: none"> • File-based database • Data buffering in case of network outage
Data security	<ul style="list-style-type: none"> •Certificate for HTTPS connection via web interface •Cyber security according to industry standards: Compliance with IEC 62351-3, IEC 62351-5 and IEC 62351-8 •Syslog event to a remote server •Internal firewall to prevent intrusion and DoS attacks.
Supported protocols	<ul style="list-style-type: none"> •Modbus master/slave (RTU / ASCII / TCP) •IEC 60870-5-101 master/slave •IEC 60870-5-103 master •IEC 60870-5-104 master/slave •DNP3.0 TLS encrypted master/slave Level 3+ •IEC 61850 •DLMS Cosem •M-bus master (Serial/TCP) •IEC 62056-21 (Modes A, B, C) •Transparent tunnel from TCP to Serial •MQTT or Restful API as optional
Features	<ul style="list-style-type: none"> •Time sync with SNTP/NTP client and server, DNP3, IEC 60870-5101/104 •Terminal server (Serial to Ethernet) •SOE event management •Firewall rules-based •Network traffic monitoring and filtering •Remote User Authentication (RADIUS) •Role-based access control (RBAC) •Certificate storage (SSL/TLS and other X.509) •Signed Firmware

3 Installing the WCC Lite+

3.1 Hardware and software requirements

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 built-in or external Ethernet adapter is also required to connect to the WCC Lite+.

3.2 Mounting

To mount the device:

1. Secure the top of the mounting clip onto a DIN rail.
2. Press down on the tab to compress the release spring and move a device forward.
3. Release the tab to secure the clip in place.

To detach the device:

1. Press down on the tab to compress the release spring
2. Pull back the bottom of the device and dismount it

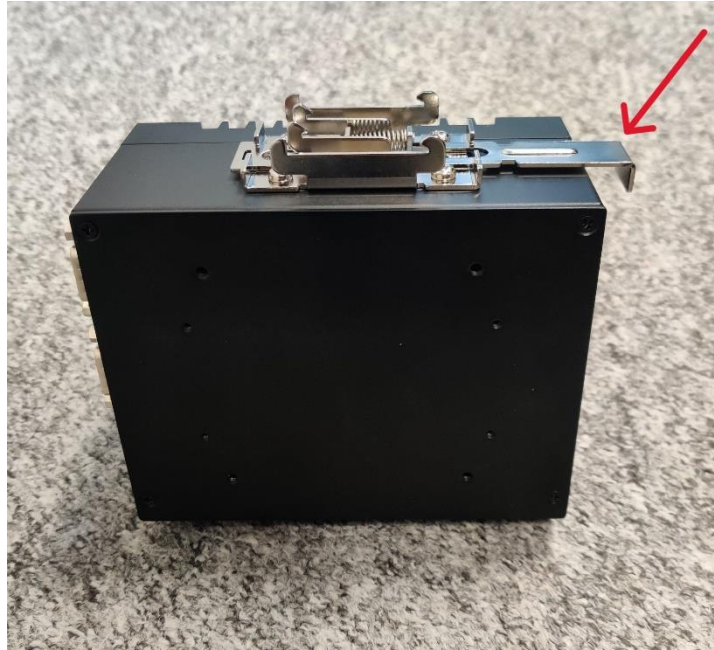


Figure 2: WCC Lite+ DIN rail mounting

3.3 Power

It is recommended to power WCC Lite+ from a 20W 12-24V DC power supply.



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.

4 Interfaces

WCC Lite+ supports various interfaces to acquire data and control external circuitry. That includes two serial port interfaces

4 Interfaces

4.1 Serial port interfaces

WCC Lite+ has 2 serial ports (Figure 7). Selectable RS485 (by default) or RS232. The interface can be changed in BIOS



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.

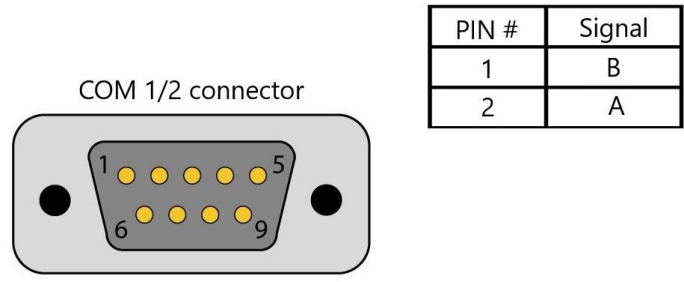


Figure 8: WCC Lite+ pinout diagram

Note: When connecting to the slave device, double-check if the A and B wires are not mixed up.

5 Internal web page

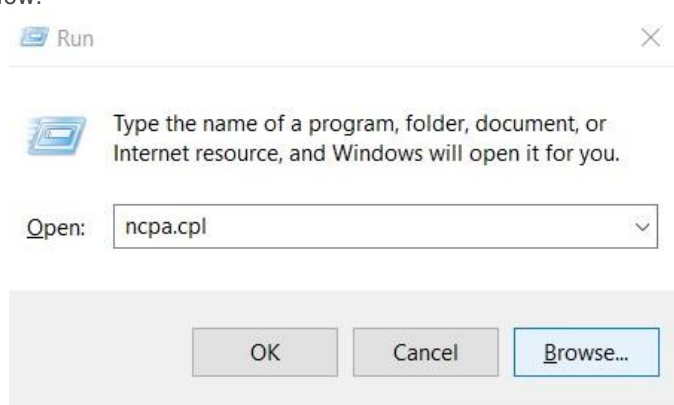
WCC Lite+ is configured via an internal web browser, so no additional software is required.

5.1 Initial Setup

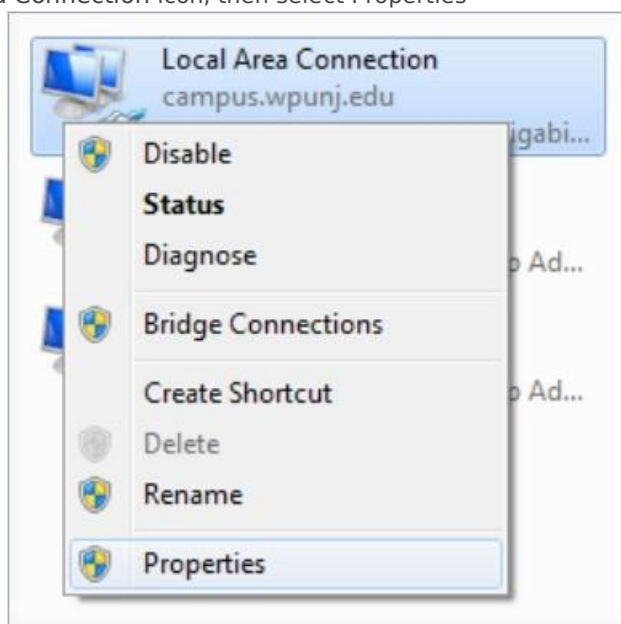
WCC Lite+ comes with a static network configuration with its IP set to 192.168.1.1. For initial setup set a static IP address on your computer and connect your network card to the WCC Lite+ with an ethernet cable.

5.1.1 Static IP Address Setup on Windows

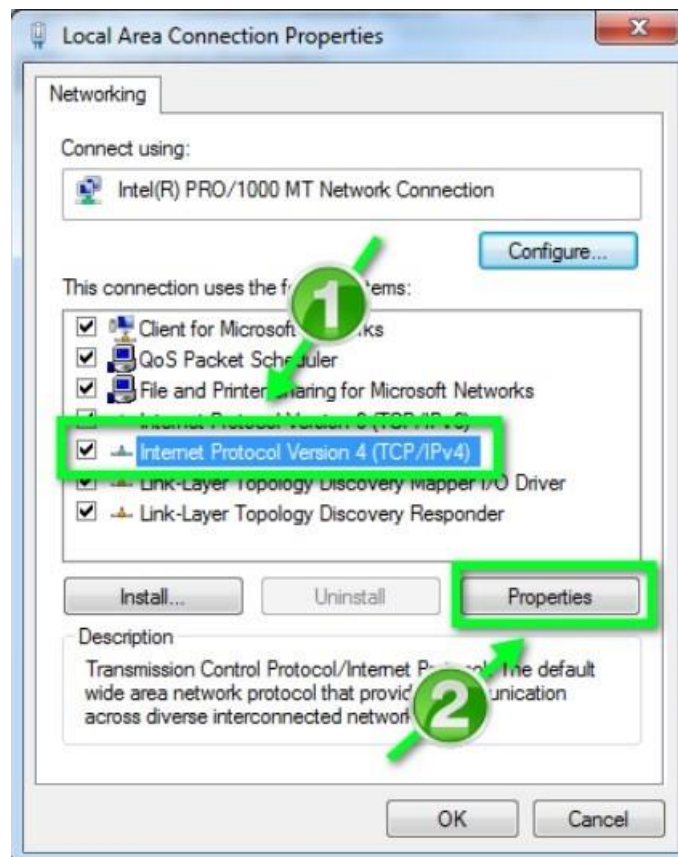
1. Press Win+R on your keyboard. This will open the run window. Enter ncpa.cpl and press OK. This will open the Network Connections window.



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

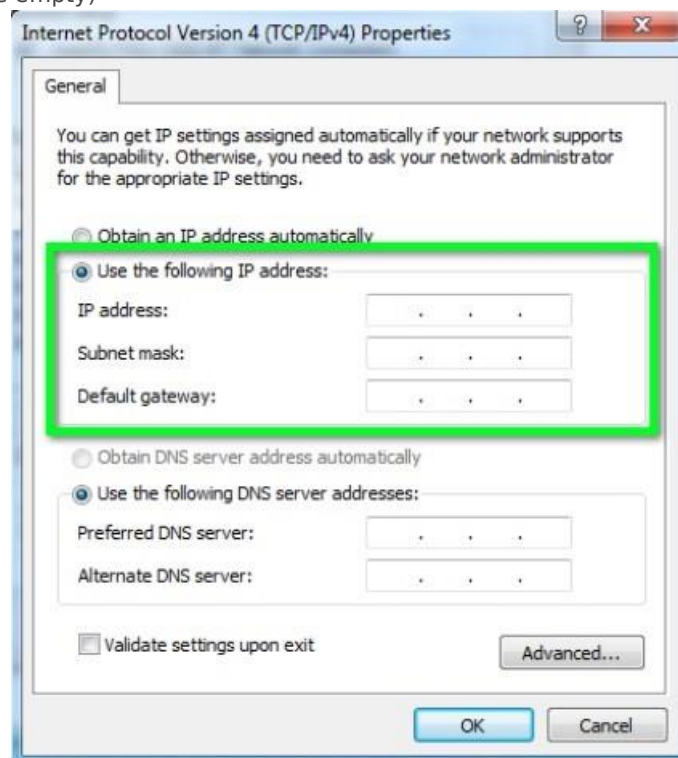


3. In the window that opens, click on Internet Protocol Version 4 (TCP/IPv4) (you may need to scroll down to find it). Next, click on the Properties button.



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

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



5.1.2 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

Authorization Required

Please enter your username and password.

Username	<input type="text"/>
Password	<input type="password"/>

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 into the local computer network, check if other devices have not already reserved such an address.

5.2 Site layout



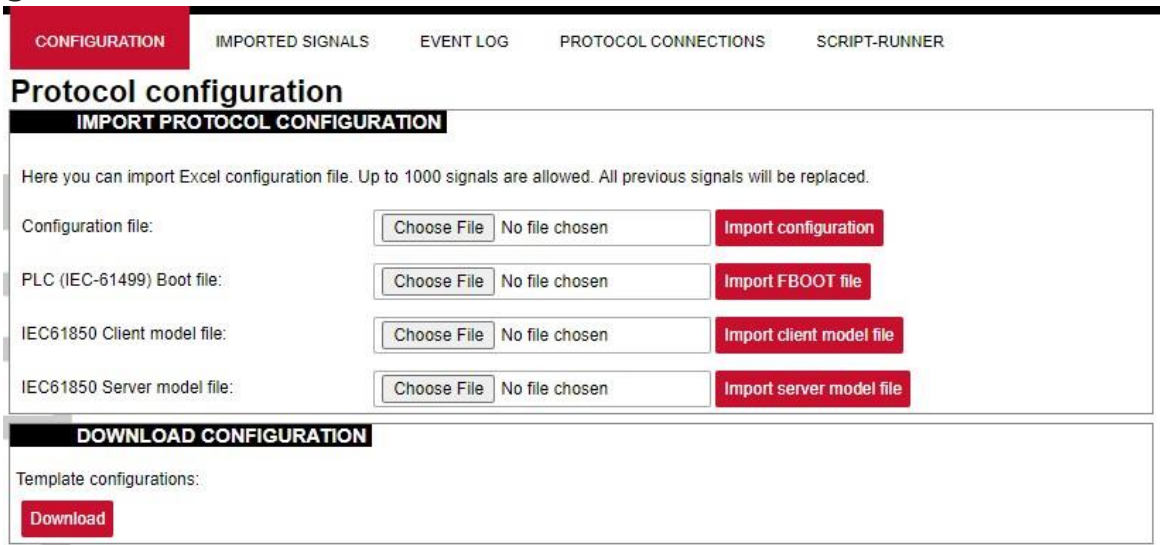
It provides the main navigation through the website. Contains the following sections:

- **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.

5.3 Protocol Hub

The Protocol HUB section stores the 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 the importing process will be stopped.
- Import .fboot file for PLC.
- Import the IEC61850 Server model file
- Import the IEC61850 Client model file
- Download the current configuration Excel file.
- Download a template configuration Excel file.

Imported Signals

IMPORTED SIGNALS						
Device	Signal	Value	Units	State	Attributes	Time
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Elseta IOMod16DI	DI1	0			asdu=1,cot=20,ioa=1,sin=1,type=dp	2024-03-18 11:51:51.47
Elseta IOMod16DI	DI2	0			asdu=1,cot=20,ioa=2,sin=1,type=dp	2024-03-18 11:51:51.77
Elseta IOMod16DI	DI3	0			asdu=1,cot=20,ioa=3,sin=1,type=dp	2024-03-18 11:51:52.74
Elseta IOMod16DI	DI4	0			asdu=1,cot=20,ioa=4,sin=1,type=dp	2024-03-18 11:51:52.37
Elseta IOMod16DI	DI5	0			asdu=1,cot=20,ioa=5,sin=1,type=dp	2024-03-18 11:51:52.68
Elseta IOMod16DI	DI6	0			asdu=1,cot=20,ioa=6,sin=1,type=dp	2024-03-18 11:51:52.98
Elseta IOMod16DI	DI7	0			asdu=1,cot=20,ioa=7,sin=1,type=dp	2024-03-18 11:51:53.28
Elseta IOMod16DI	DI8	0			asdu=1,cot=20,ioa=8,sin=1,type=dp	2024-03-18 11:51:53.58

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

Event Log

EVENT LOG						
DEVICE EVENTS						
Auto refresh <input checked="" type="checkbox"/>		Number of items: 50				
Device	Signal alias	Signal name	Value	Timestamp	Direction	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
IOMod16	DI16	DI16	0.000000	2024-03-18 09:51:55.177	M	
IOMod16	DI15	DI15	0.000000	2024-03-18 09:51:55.477	M	
IOMod16	DI14	DI14	0.000000	2024-03-18 09:51:55.177	M	
IOMod16	DI13	DI13	0.000000	2024-03-18 09:51:55.477	M	
IOMod16	DI12	DI12	0.000000	2024-03-18 09:51:54.177	M	
IOMod16	DI11	DI11	0.000000	2024-03-18 09:51:54.476	M	
IOMod16	DI10	DI10	0.000000	2024-03-18 09:51:54.176	M	
IOMod16	DI9	DI9	0.000000	2024-03-18	M	

Download events log archive:

[Download](#)

The event Log is the timestamped status data. It allows reviewing of the latest events and changes for the device's state changes in chronological order. The 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 WCC Lite+; 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 WCC Lite+ as an SOE point with an Excel template during configuration.

Each time a device changes state, the WCC lite+ will save it with a time tag 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

CONFIGURATIONIMPORTED SIGNALSEVENT LOG**PROTOCOL CONNECTIONS**SCRIPT-RUNNER

PROTOCOL CONNECTIONS				
Device	Protocol	Host	Status	Timestamp
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
DNP3_SCADA	DNP3 slave	192.168.1.2	Disconnected	2024-03-18 11:49:33
IOMod16	IEC 60870-5-103 master	PORT1	Connected	2024-03-18 11:51:50
Modbus_SCADA	Modbus TCP slave	192.168.1.2	Disconnected	2024-03-18 11:49:32
IEC104_SCADA	IEC 60870-5-104 slave	192.168.1.2	Disconnected	2024-03-18 11:49:31
IEC101_SCADA	IEC 60870-5-101 slave	PORT2	Disconnected	2024-03-18 11:49:30

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

Script-Runner

CONFIGURATIONIMPORTED SIGNALSEVENT LOG**PROTOCOL CONNECTIONS****SCRIPT-RUNNER**

Script-Runner

LUA SCRIPT INSTANCE CONTROL

Script Configuration	Script process	Status	Script File		
LUA	-	Stopped	No Script provided	Upload Script	Waiting for script

SAVED VALUE CLEARING

Clear all saved values

To upload the Lua script, the Lua device needs to be configured. In this tab, said script can be uploaded and executed.

5 Internal web page

5.4 Status

Overview

System

Status

SYSTEM	
Hostname	wcc-lite-plus
Model	WCC Lite+
Firmware Version	WCC+ wcc-lite+ v1.1.18
Kernel Version	4.4.14
Local Time	Mon Mar 18 10:39:09 2024
Uptime	1h 27m 56s
Load Average	0.02, 0.04, 0.00

The 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.

Memory

MEMORY	
Total Available	3979976 kB / 4048600 kB (98%)
Free	3836040 kB / 4048600 kB (94%)
Buffered	143936 kB / 4048600 kB (3%)



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

NETWORK	
IPv4 WAN Status	 eth1
IPv6 WAN Status	 ?
Active Connections	21 / 16384 (0%)

Type: dhcp
Address: 192.168.69.103
Netmask: 255.255.255.0
Gateway: 192.168.69.1
DNS 1: 192.168.69.1
Expires: 1h 59m 45s
Connected: 0h 0m 15s

Not connected

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
There are no active leases.			

DHCPv4 and DHCPv6 lease expiration time.

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

IPv4-Address: IPv4 address of network interface.

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

DUID: DHCP Unique Identifier of IPv6 network interface.

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

Board information

BOARD INFORMATION	
Hardware version	WCCLite v1.3
Serial number	318040040
SoC ID	c493000bf455

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

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	*	0.0.0.0/0	0.0.0.0/0	/* fw3 */
78	28.34 KB	input_rule	all	*	*	0.0.0.0/0	0.0.0.0/0	/* fw3: user chain for input */
73	28.10 KB	ACCEPT	all	*	*	0.0.0.0/0	0.0.0.0/0	ctstate RELATED,ESTABLISHED /* fw3 */
4	208.00 B	syn_flood	tcp	*	*	0.0.0.0/0	0.0.0.0/0	top flags:0x17/0x02 /* fw3 */
5	240.00 B	zone_lan_input	all	br-lan	*	0.0.0.0/0	0.0.0.0/0	/* fw3 */
0	0.00 B	zone_wan_input	all	eth1	*	0.0.0.0/0	0.0.0.0/0	/* fw3 */

Firewall rule list for IPv4 traffic.

Table: The four distinct tables that store rules regulating operations on the packet. Filter concerns filtering rules. NAT concerns the translation of source or destination addresses and ports of packages. The 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 are 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 is processed by the firewall.

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

Out: The network interface for the output chain is 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 */
130	10.87 KB	input_rule	all	*	*	::/0	::/0	/* fw3: user chain for input */
0	0.00 B	ACCEPT	all	*	*	::/0	::/0	ctstate RELATED,ESTABLISHED /* fw3 */
0	0.00 B	syn_flood	tcp	*	*	::/0	::/0	top flags:0x17/0x02 /* fw3 */
130	10.87 KB	zone_lan_input	all	br-lan	*	::/0	::/0	/* fw3 */
0	0.00 B	zone_wan_input	all	eth1	*	::/0	::/0	/* fw3 */

Firewall rule list for IPv6 traffic.

Table: The three distinct tables that store rules regulating operations on the packet. Filter concerns filtering rules. The 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 are 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 is processed by the firewall.

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

Out: The network interface for the output chain is 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

The following rules are currently active on this system.

ARP				
IPv4-Address	MAC-Address	Interface		
192.168.69.1	4c:02:89:1a:02:b2	eth1		
192.168.1.2	e4:b9:7a:37:5b:88	br-lan		

ACTIVE IPV4-ROUTES				
Network	Target	IPv4-Gateway	Metric	Table
wan	0.0.0.0/0	192.168.69.1	0	main
lan	192.168.1.0/24		0	main
wan	192.168.69.0/24		0	main
wan	192.168.69.1		0	main

ACTIVE IPV6-ROUTES				
Network	Target	Source	Metric	Table
lan	fd02:62d:1b2c::/64		1024	main
lan	ff00::/8		256	local
wan	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 that defines how the 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 to the subnet mask 255.255.255.0.

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

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

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

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

System Log

System Log

#	Time	Facility	Process	Priority	Message
2	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] Command line: BOOT=0x0000000000000000 block2mtd.block2mtd=PARTUUID=09c6d88d-02,65536,rootfs,5 root=/dev/mtdblock0 rootfstype=squashfs rootwait console=tty0 console=ttyS0,115200n8 noinitrd
3	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point registers'
4	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
5	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] x86/fpu: Enabled xstate features 0x3, context size is 576 bytes, using 'standard' format.
6	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] x86/fpu: Using 'eager' FPU context switches.
7	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] e820: BIOS-provided physical RAM map:
8	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000009fbfff] usable
9	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x0000000000009fc00-0x0000000000009ffff] reserved
10	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x000000000000f0000-0x000000000000ffff] reserved
11	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x00000000000100000-0x000000000000bffd9fff] usable
12	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x0000000000bffd9000-0x0000000000bfffffff] reserved
13	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x0000000000ffffffc000-0x0000000000ffffffe000] reserved
14	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x0000000000ffffffc0000-0x0000000000ffffffe0000] reserved
15	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] BIOS-e820: [mem 0x00000000000000000-0x0000000000000000013ffff] usable
16	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] NX (Execute Disable) protection: active
17	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] SMBIOS 3.0.0 present.
18	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] Hypervisor detected: KVM
19	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] e820: update [mem 0x000000000-0x00000fff] usable ==> reserved
20	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] e820: remove [mem 0x000a0000-0x000ffff] usable
21	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] e820: last_pfn = 0x140000 max_arch_pfn = 0x400000000
22	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] MTRR default type: write-back
23	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] MTRR fixed ranges enabled:
24	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 00000-9FFFF write-back
25	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] A0000-BFFFF uncachable
26	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] C0000-FFFFFF write-protect
27	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] MTRR variable ranges enabled:
28	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 0 base 00C0000000 mask 7FC0000000 uncachable
29	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 1 disabled
30	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 2 disabled
31	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 3 disabled
32	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 4 disabled
33	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 5 disabled
34	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 6 disabled
35	Thu Dec 21 22:56:08 2023	kern	kernel	debug	[0.000000] 7 disabled
36	Thu Dec 21 22:56:08 2023	kern	kernel	info	[0.000000] x86/PAT: Configuration [0-7]: WB WC UC- UC WB WC UC- WT

The 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 (arminius@ubuntu16) (gcc version 5.3.0 (OpenWrt GCC 5.3.0 v1.8.0-3-g264ba60) ) #22 SMP Fri Dec 8 12:18:47 UTC 2023
0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz block2mtd.block2mtd=PARTUUID=09c6d88d-02.65536.rootfs,5 root=/dev/mtdblock0 rootfstype=squashfs rootwait cons
0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point registers'
0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
0.000000] x86/fpu: Enabled xstate features 0x3, context size is 576 bytes, using 'standard' format.
0.000000] x86/fpu: Using 'eager' FPU context switches.
0.000000] e820: BIOS-provided physical RAM map:
0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
0.000000] BIOS-e820: [mem 0x00000000000a0000-0x00000000000affff] reserved
0.000000] BIOS-e820: [mem 0x00000000000b0000-0x00000000000bffff] usable
0.000000] BIOS-e820: [mem 0x00000000000c0000-0x00000000000cffff] reserved
0.000000] BIOS-e820: [mem 0x00000000000d0000-0x00000000000dffff] reserved
0.000000] BIOS-e820: [mem 0x00000000000e0000-0x00000000000effff] reserved
0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
0.000000] BIOS-e820: [mem 0x0000000000100000-0x000000000013ffff] usable
0.000000] NX (Execute Disable) protection: active
0.000000] SMBIOS 3.0.0 present
```

The 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

This list gives an overview over currently running system processes and their status.

PID	Owner	Command	CPU usage (%)	Memory usage (%)	Hang Up	Terminate	Kill
1	root	/sbin/procd	0%	1%	Hang Up	Terminate	Kill
2	root	[kthreadd]	0%	0%	Hang Up	Terminate	Kill
3	root	[ksoftirqd/0]	0%	0%	Hang Up	Terminate	Kill
4	root	[kworker/0:0]	0%	0%	Hang Up	Terminate	Kill
5	root	[kworker/0:0H]	0%	0%	Hang Up	Terminate	Kill
6	root	[kworker/u8:0]	0%	0%	Hang Up	Terminate	Kill
7	root	[rcu_sched]	0%	0%	Hang Up	Terminate	Kill
8	root	[rcu_bh]	0%	0%	Hang Up	Terminate	Kill
9	root	[migration/0]	0%	0%	Hang Up	Terminate	Kill
10	root	[migration/1]	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.

Chronos

Time synchronization status

Shows the source and status of the time synchronization service



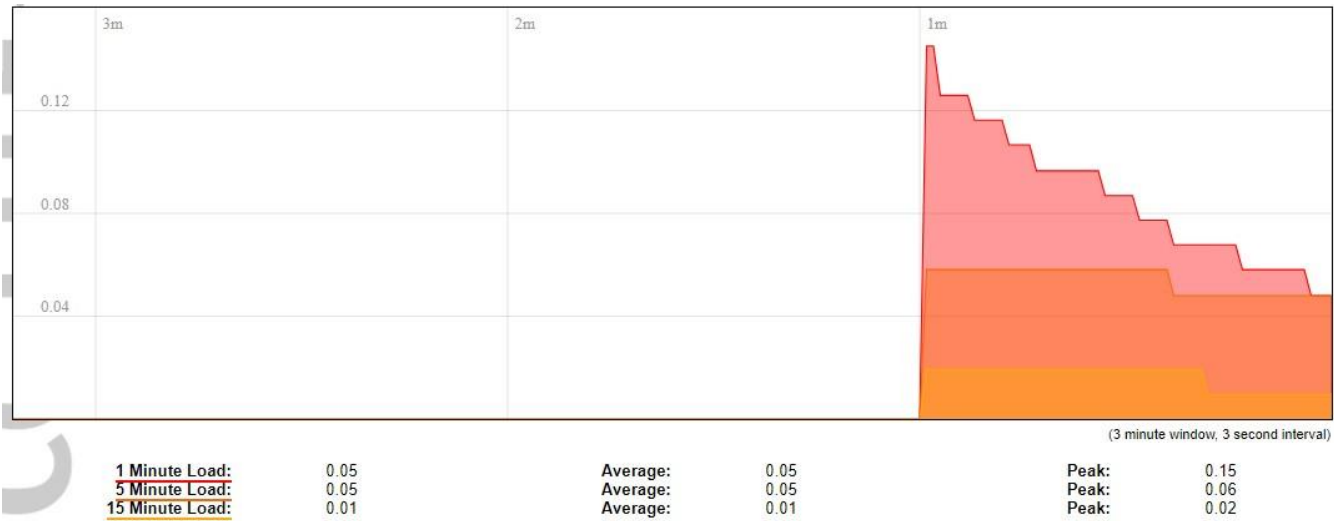
Shows the source and status of the time synchronization service.

Red - not synced

Green - synced

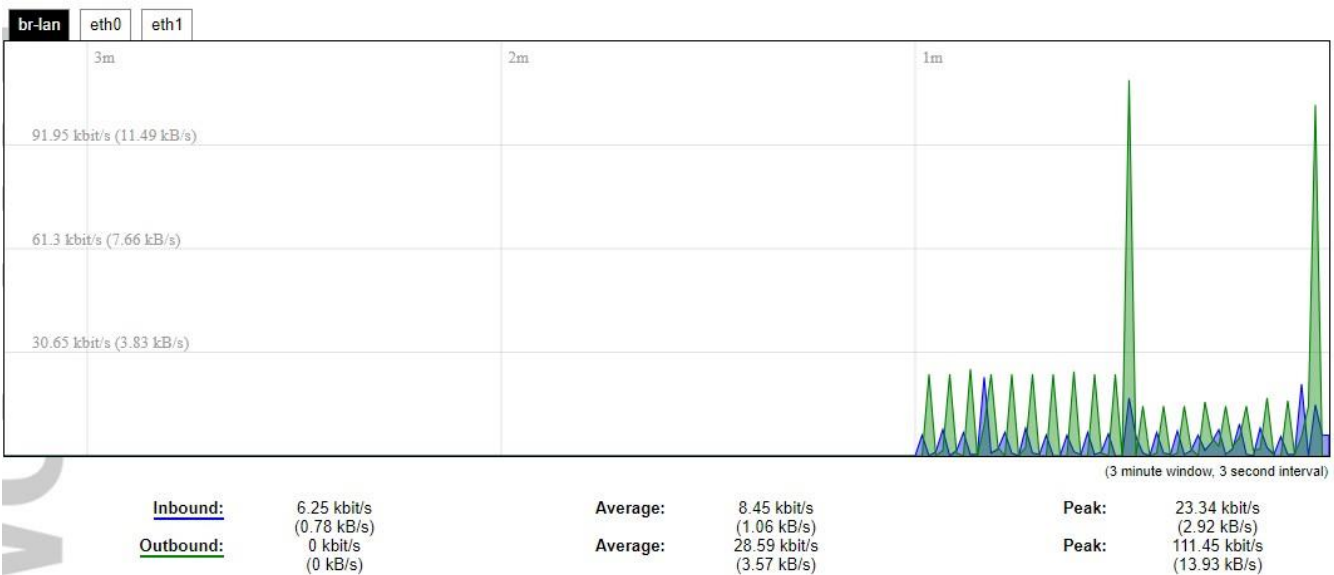
Realtime graph

Realtime Load



CPU utilization graph. A load of 0.5 means the CPU has been 50% utilized over the last period.

Realtime Traffic



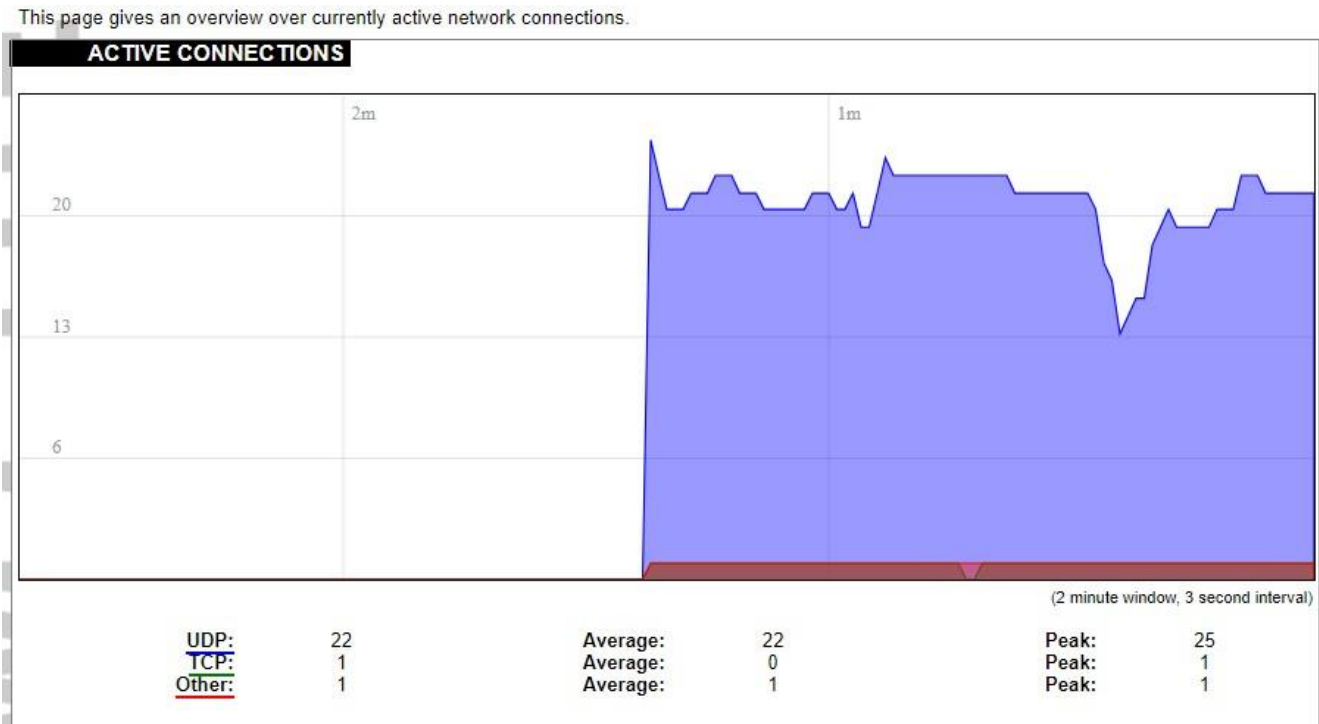
Graphs representing the status of the virtual and physical network interfaces of the device.

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

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

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

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.

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

Monitor selected interfaces

☒

☐

☒

Bridge: "br-lan" (lan)

Ethernet Adapter: "eth0"

Ethernet Adapter: "eth1" (wan, wan6)

Save & Apply

Save

Reset

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.

5 Internal web page

5.5 System System

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

- SYSTEM

ADMINISTRATION

SOFTWARE

STARTUP

SCHEDULED TASKS

CERTIFICATE STORAGE

BACKUP / FLASH FIRMWARE
- TIME SYNC

REBOOT

- **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.
- **CERTIFICATE STORAGE:** certificate management panel.
- **TIME SYNC:** time synchronization of WCC Lite+
- **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

System

Here you can configure the basic aspects of your device like its hostname or the timezone.

SYSTEM PROPERTIES

General Settings

Logging

Language and Style

Local Time

Thu Dec 21 15:24:18 2023

Sync with browser

*It is not advisable to synchronise time from web browser

Hostname

wcc-lite-plus

Timezone

UTC

The general settings of the WCC Lite+ device are defined as follows:
Local Time: Current local time.
Hostname: The label that is used to identify the device in the network.
Timezone: A region of the globe that observes a uniform standard time. The time zone number indicates the number of hours by which the time is shifted ahead of or behind UTC – Coordinated Universal Time. Some zones are, however, shifted by 30 or 45 minutes.

Logging

SYSTEM PROPERTIES

General Settings

Logging

Language and Style

System log buffer size

16

External system log server

0.0.0.0

External system log server port

514

External system log server protocol

UDP

Write system log to file

/root/syslog

Log output level

Debug

Cron Log Level

Normal

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

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.

TIME SYNCHRONIZATION

Enable NTP client

☒

Provide NTP server

☐

NTP server candidates

0.openwrt.pool.ntp.org

1.openwrt.pool.ntp.org


2.openwrt.pool.ntp.org

3.openwrt.pool.ntp.org

Save & Apply

Save

Reset

 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 using a 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.

Language and styles

SYSTEM PROPERTIES		
General Settings	Logging	Language and Style
Language	auto	
Design	Wcc	

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.

Administration

Administrator Password

PASSWORD INSTANCE	
Password	<input type="password"/>
Confirmation	<input type="password"/>

The 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.

Password policy

PASSWORD POLICY INSTANCE	
Enable Password Policy	<input type="checkbox"/>
Enable or disable password policy	
Minimum Password Length	6
Minimum Number of Upper Case Letters	0
Minimum Number of Lower Case Letters	0
Minimum Number of Digits	0
Minimum Number of Special Characters	0
Check for Similiar Characters	<input type="checkbox"/>
Enable or disable repeated character check in password	

For future password changes, users can configure a password policy to create a safer password. Here password requirements can be created such as minimum password length, minimum number of upper or lower case letters, digits, and special characters. By ticking the box for checking similar characters, a new password will be required to not have repeated characters.

SSH Access

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

DROPBEAR INSTANCE

Delete

Interface

☐ gsm:

☐ lan:

☐ wan:

☐ wan6:

☒ unspecified

☒ Listen only on the given interface or, if unspecified, on all

Port

22

☒ Specifies the listening port of this Dropbear instance

Password authentication

☒

☒ Allow SSH password authentication

Allow root logins with password

☒

☒ Allow the root user to login with password

Gateway ports

☐

☐ Allow remote hosts to connect to local SSH forwarded ports

Add

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 root logins with password: Allow the root user to log in with the password.

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

SSH-keys

SSH-KEYS

Here you can paste public SSH-Keys (one per line) for SSH public-key authentication.

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

RADIUS Client

RADIUS SERVER CONFIGURE

Add or Remove RADIUS client configuration

Enable	Hostname / IP	Timeout	Shared secret
This section contains no values yet			

Add

RADIUS client redirects user authorization to the remote server, which controls users and their access. A user can add multiple RADIUS clients by clicking add and entering the information required.

HTTPS certificate

Select a certificate from certificate storage to use for secure HTTP access

CERTIFICATE	
Certificate file	Default ▼

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

Users can use certificates uploaded to a certificate storage. It should be noted that only valid certificates with *.pem extension can be used. The certificate to be used is validated every time the device is restarted. If validation fails, a default certificate is used. This is done to prevent users from losing device access via a web interface.

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

Software

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

Actions	Configuration
No package lists available Update lists	
Free space: 99% (1004.31 MB) <div><div></div></div>	
Download and install package:	<input type="text"/> OK
Filter:	<input type="text"/> Find package

Status

Installed packages		Available packages
	Package name	Version
Remove	4diac-forte	1.11.0-2020-06-16
Remove	base-files	169-v1.8.1-7-g69fc33e

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.

OPKG-Configuration

General options for opkg						
<table><thead><tr><th>Actions</th><th>Configuration</th></tr></thead><tbody><tr><td colspan="2"><pre>dest root / dest ram /tmp lists_dir ext /var/opkg-lists option overlay_root /overlay option check_signature 1</pre></td></tr><tr><td>Submit</td><td>Reset</td></tr></tbody></table>	Actions	Configuration	<pre>dest root / dest ram /tmp lists_dir ext /var/opkg-lists option overlay_root /overlay option check_signature 1</pre>		Submit	Reset
Actions	Configuration					
<pre>dest root / dest ram /tmp lists_dir ext /var/opkg-lists option overlay_root /overlay option check_signature 1</pre>						
Submit	Reset					

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

Distribution feeds

Build/distribution specific feed definitions. This file will NOT be preserved in any sysupgrade.

```
src/gz wcclite+_base http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/base
src/gz wcclite+_kernel http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/kernel
src/gz wcclite+_telephony http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/telephony
src/gz wcclite+_elseta http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/elseta
src/gz wcclite+_packages http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/packages
src/gz wcclite+_routing http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/routing
src/gz wcclite+_luci http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/luci
src/gz wcclite+_management http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/management
# src/gz wcclite+_targets http://downloads.openwrt.org/snapshots/trunk/x86/64/packages/targets
```

Submit **Reset**

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

Custom feeds

Custom feed definitions, e.g. private feeds. This file can be preserved in a sysupgrade.

```
# add your custom package feeds here
#
# src/gz example_feed_name http://www.example.com/path/to/files
```

Submit **Reset**

Startup

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

You can enable or disable installed init scripts here. Changes will applied after a device reboot.
Warning: If you disable essential init scripts like "network", your device might become inaccessible!

Start priority	Initscript	Enable/Disable	Start	Restart	Stop
0	sysfixtime	Enabled	Start	Restart	Stop
2	wcclite	Disabled	Start	Restart	Stop
10	boot	Enabled	Start	Restart	Stop
10	system	Enabled	Start	Restart	Stop
11	sysctl	Enabled	Start	Restart	Stop
12	log	Enabled	Start	Restart	Stop
12	rpcd	Enabled	Start	Restart	Stop
19	firewall	Enabled	Start	Restart	Stop
20	network	Enabled	Start	Restart	Stop

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

This is the content of /etc/rc.local. Insert your own commands here (in front of 'exit 0') to execute them at the end of the boot process.

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

exit 0
```

Submit

Reset

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

Scheduled tasks

This is the system crontab in which scheduled tasks can be defined.

Note: you need to manually restart the cron service if the crontab file was empty before editing.

```
0 * * * * /usr/sbin/logrotate /etc/logrotate.conf
```

Submit

Reset

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.

The 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.

Certificate storage

Manage certificates used by various protocols

CERTIFICATES

Below is a list of succesfully uploaded certificates and their properties

File name

Valid from

Valid until

Issuer

Subject

This section contains no values yet

Choose file

No file chosen

Upload

Save & Apply

Save

Reset

This section is intended to upload certificate files and view information about them.

Time sync

TIMESYNC

Enable

☒

Timeout

NTP

Enable

☒

Priority

IEC101

Enable

☒

Priority

DNP3

Enable

☒

Priority

IEC104

Enable

☒

Priority

Save & Apply

Save


Reset

This service syncs WCC Lite+ time with the protocols shown. Here user can also select priority levels of protocols that sync with WCC Lite+.

Backup/flash firmware

Software update allows upgrading 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.



The 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 the operating system unstable or even unusable

Actions

Configuration

BACKUP / RESTORE

Click "Generate archive" to download a tar archive of the current configuration files. To reset the firmware to its initial state, click "Perform reset" (only possible with squashfs images).

Download backup:

Generate archive

Reset to defaults:

Perform reset

Get System Diagnostic Report:

Generate archive

To restore configuration files, you can upload a previously generated backup archive here.

Restore backup:

Choose file

No file chosen

Upload archive...

FLASH NEW FIRMWARE IMAGE

Upload a sysupgrade-compatible image here to replace the running firmware. Check "Keep settings" to retain the current configuration (requires a compatible firmware image).

Keep settings:

☐

Keep only network settings:

☐


Image:

Choose file

No file chosen

Flash image...

A user can choose to keep existing settings after an upgrade. Marking the 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 images, to preserve RAM, 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, the firmware file will be deleted and Protocol HUB processes will be restarted.

1.1.18

Elseta

36

Actions

Configuration

This is a list of shell glob patterns for matching files and directories to include during sysupgrade. Modified files in `/etc/config/` and certain other configurations are automatically preserved.

Show current backup file list

Open list...

```
## This file contains files and directories that should
## be preserved during an upgrade.

# /etc/example.conf
# /etc/openvpn/
```

Submit

Reset

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

Reboot

SYSTEM

ADMINISTRATION

SOFTWARE

STARTUP

SCHEDULED TASKS

CERTIFICATE STORAGE

BACKUP / FLASH FIRMWARE

TIME SYNC

REBOOT

Reboot

Reboots the operating system of your device

Perform reboot

This reboots the operating system of the device.

5 Internal web page

5.6 Services

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

TELEMETRY AGENT

IPSEC

API

OPENVPN

SER2NET

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

- **TELEMETRY AGENT:** device telemetry sent to a remote server;
- **IPSEC:** encrypted virtual private network (VPN) configuration.
- **API:** application programming interface configuration.
- **OPENVPN:** shows the open-source software application that implements a 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 a 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 a Telemetry agent a user should configure and enable it. Four options are available:

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

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

⚠ The telemetry agent doesn't start as a service if the Enable agent checkbox is unchecked.

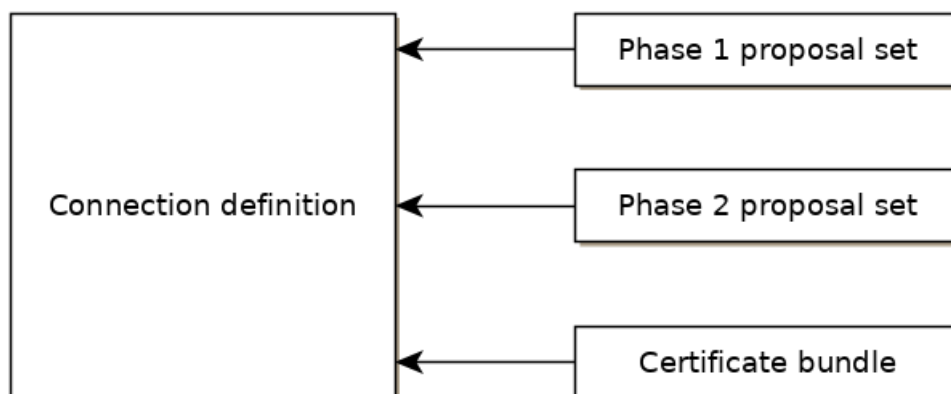
🔧 Enabling the agent and saving the configuration automatically starts the process with the new configuration.

IPsec

Background

WCC Lite+ supports ipsec VPN and thus can deliver data securely over encrypted links. To establish ipsec vpn, a connection definition must be created by entering the 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 at the time of creation. The following diagram shows the relations between connection and auxiliary sets.



Ipssec settings

Connection description

Options supported by WCC lite+ are 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 the form of 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 is set to Pre-shared key
Certificate set	selector	Available if Authentication is set to RSA certificate. Selectable from the 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: <ul style="list-style-type: none"> • none – default, disables sending of DPD messages. • clear – the connection closed with no action. • hold – keeps description, tries renegotiate connection on demand. • restart – will try to re-negotiate immediately.
DPD delay	string	Time interval in seconds between peer checks. Default 30.
DPD timeout	string	Time in seconds after which peers consider it 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 correspond to settings used in further configuration walk-through examples.

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: wcc-lite
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 the section “Phase 1 proposals”.
- Create a new record by assigning a 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.

Save

IPsec

PHASE 1 PROPOSALS

Below is a list of configured IPsec phase 1 proposals

	Encryption algorithm	Hash algorithm	DH exponentiation	
aes256_sha256_dh5	<div>aes256</div>	<div>sha256</div>	<div>modp3072 (15)</div>	<div>Delete</div>
<div></div>	<div>Add</div>			

Save & Apply

Save

Reset

Creation of Phase 2 proposal

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

Save

IPsec

PHASE 2 PROPOSALS

Below is a list of configured IPsec phase 2 proposals

	Encryption algorithm	Hash algorithm	DH exponentiation	
aes128_sha1	<div>aes128</div>	<div>sha1</div>	<div></div>	<div>Delete</div>
<div></div>	<div>Add</div>			

Save & Apply

Save

Reset

Creation of tunnel definition

Enter section connections

- Create a new record by assigning a new name (e.g. “demo0”) and clicking “Add”.
 - Call a detailed 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 the combo box “additional field”, select “Local ID”, then set the value to wcclite.
- Push “Save”.

Save

» CONNECTION "DEMO0"

Gateway	<input type="text" value="ipsec.vpn.net"/>
Type	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">Tunnel ▼</div>
Local subnet	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">10.10.12.0/24</div>
Remote subnet	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">10.10.10.0/24</div>
Authentication	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">Shared secret ▼</div>
Pre-shared key (PSK)	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">.....</div>
Phase 1 proposal (IKE)	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">aes256_sha256 ▼</div>
Phase 2 proposal (ESP)	<div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">aes128_sha1 ▼</div>
Local ID	<input type="text" value="wcclite"/>

-- Additional Fiel ▼

Add

Save & Apply

Save

Reset

Activating the tunnel

- Return to the section “connections”.
- Check the checkbox “Enabled”.
- Push the button “save & apply”.
- Examine the indicator “configured”, it should be “yes”, if not, review the 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 the button “start”.

Save

IPsec CONNECTIONS

Below is a list of configured IPsec connection instances and their current state

	Enabled	Configured	Established	Gateway	Start/Stop	
demo0	<input checked="" type="checkbox"/>	yes	yes	ipsec.vpn.net	<div style="background-color: red; color: white; padding: 2px 10px;">stop</div>	<div style="background-color: black; color: white; padding: 2px 10px;">Edit</div> <div style="background-color: red; color: white; padding: 2px 10px;">Delete</div>

Add

Save & Apply

Save

Reset

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 IETF RFC 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 an 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 that 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.

API

The firmware of the WCC Lite+ features a built-in API which is accessible via the web interface.

The API (as of version 1.2.0) implements API key authorization, that is generated on the first boot of the device. The authorization can be enabled or disabled and the API key can be changed via the web configuration interface at Services → API.

Authentication

API key authentication is enabled by default and it generates a random 12-character long key. The key is stored in the same uci configuration as API.

To change the API key via the interface, it must be equal to or longer than 12 characters.

When API authentication is enabled WCC Lite+ checks if the "apiKey" header exists in the request and then validates its value against the one in the configuration. If the value is correct it continues with the API request, if the apiKey is incorrect or doesn't exist it returns an error response 401 with the message.

```
{
  "error": "Unauthorized access."
}
```

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

 All endpoints are disabled by default.

Available API endpoints are shown in the table below.

Table. Available API endpoints:

Endpoint	Description
/api/version	Version of the API
/api/actions	List of available points
/api/syncVersion	A 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 a 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".

If API authorization is enabled, you must add a header "apiKey" with the correct key.

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 the WCC Lite+ Web interface or by uploading the OVPN file containing the necessary parameters. OpenVPN will automatically attempt to load all *.conf files placed in the /etc/openvpn folder. Several OpenVPN recipes are suggested containing the most used configurations that may only require minor changes. If a user intends to set up OpenVPN without an OVPN file, it is highly advised to use these recipes and tweak them up to individual needs.

OpenVPN

OPENVPN INSTANCES

Below is a list of configured OpenVPN instances and their current state

	Enabled	Started	Start/Stop	Port	Protocol	
sample_server	<input type="checkbox"/>	no	<button>start</button>	1194	udp	<button>Edit</button> <button>Delete</button>
sample_client	<input type="checkbox"/>	no	<button>start</button>	-	udp	<button>Edit</button> <button>Delete</button>

Template based configuration

Instance name

Select template ...

Add

OVPN configuration file upload

Instance name

No file chosen

Upload

Save & Apply

Save

Reset

The 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 the Edit button, configuration can be removed with the Delete button. Pressing Edit takes the user to the main configuration screen containing the options usually used

in particular OpenVPN recipes. To make 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 use OVPN files, however, if the 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.

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 an MIB (Management Information Base). WCC Lite+ supports SNMP service which is not added to the default build of firmware but can be installed as a module.

It enables users to collect data on various parameters of the 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 - the 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 - the 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 the

network; speed - the 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).

5.7 Network

INTERFACES

DHCP AND DNS

HOSTNAMES

STATIC ROUTES

FIREWALL




DIAGNOSTICS

The page shows information about the current interface status and its configurations, provides various interface, and 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 and covers the 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.

Interfaces

INTERFACE OVERVIEW

Network	Status	Actions
<div>LAN</div> <div>br-lan</div>	<div>Uptime: 1h 21m 6s</div> <div>MAC-Address: BC:24:11:AA:84:50</div> <div>RX: 1.12 MB (9774 Pkts.)</div> <div>TX: 2.61 MB (4555 Pkts.)</div> <div>IPv4: 192.168.1.1/24</div> <div>IPv6: fd02:62d:1b2c::1/60</div>	<div>Connect/Reconnect</div> <div>Stop</div> <div>Edit</div> <div>Delete</div>
<div>WAN</div> <div>eth1</div>	<div>Uptime: 1h 9m 57s</div> <div>MAC-Address: BC:24:11:7F:10:8F</div> <div>RX: 56.11 KB (594 Pkts.)</div> <div>TX: 133.63 KB (868 Pkts.)</div> <div>IPv4: 192.168.69.103/24</div>	<div>Connect/Reconnect</div> <div>Stop</div> <div>Edit</div> <div>Delete</div>
<div>WAN6</div> <div>eth1</div>	<div>Uptime: 0h 0m 0s</div> <div>MAC-Address: BC:24:11:7F:10:8F</div> <div>RX: 56.11 KB (594 Pkts.)</div> <div>TX: 133.63 KB (868 Pkts.)</div>	<div>Connect/Reconnect</div> <div>Stop</div> <div>Edit</div> <div>Delete</div>

Add new interface...

GLOBAL NETWORK OPTIONS

IPv6 ULA-Prefix

fd02:062d:1b2c::/48

Save & Apply

Save

Reset

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 the configured interface network if it does not do it automatically. If it is 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 a new Ethernet or wireless interface with the custom name, protocol, etc.


	eth0	eth1
Type	Static	DHCP
Address	192.168.1.1	
Subnet mask	255.255.255.0	
Gateway		


 Changes will only take effect after the device reboots.

Network interfaces can be configured on the common page, which can be accessed by adding a new interface or edit button.


Create Interface


Name of the new interface


 The allowed characters are: A-Z, a-z, 0-9 and _

Note: interface name length
 Maximum length of the name is 15 characters including the automatic protocol/bridge prefix (br-, 6in4-, pppoe- etc.)

Protocol of the new interface
Static address ▼
Create a bridge over multiple interfaces
☐
Cover the following interface

☐
 Ethernet Adapter: "eth0" (lan)

☐
 Ethernet Adapter: "eth1" (wan, wan6)

☐
 Custom Interface:

Back to Overview
Submit

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.

COMMON CONFIGURATION


General Setup

Advanced Settings

Physical Settings

Firewall Settings

Status

 br-1an

Uptime: 1h 23m 40s
 MAC-Address: BC:24:11:AA:84:50
 RX: 1.16 MB (10072 Pkts.)
 TX: 2.77 MB (4715 Pkts.)
 IPv4: 192.168.1.1/24
 IPv6: fd02:62d:1b2c::1/60

Protocol

Static address

IPv4 address

192.168.1.1

IPv4 netmask

255.255.255.0

IPv4 gateway

IPv4 broadcast

Use custom DNS servers

IPv6 assignment length

60

Assign a part of given length of every public IPv6-prefix to this interface

IPv6 assignment hint

Assign prefix parts using this hexadecimal subprefix ID for this interface.

IPv6 suffix

::1

Optional. Allowed values: 'eui64', 'random', fixed value like '::1' or '::1:2'. When IPv6 prefix (like 'a:b:c:d::') is received from a delegating server, use the suffix (like '::1') to form the IPv6 address ('a:b:c:d::1') for the interface.

General common interface setup panel.

General Setup

Advanced Settings

Physical Settings

Firewall Settings

Bring up on boot ☒

Use builtin IPv6-management ☒

Force link ☒

Set interface properties regardless of the link carrier (If set, carrier sense events do not invoke hotplug handlers).

Override MAC address

BC:24:11:AA:84:50

Override MTU

1500

Use gateway metric

0

Advanced common interface setup panel.

General Setup

Advanced Settings

Physical Settings

Firewall Settings

Bridge interfaces ☒

creates a bridge over specified interface(s)

Enable STP ☐

Enables the Spanning Tree Protocol on this bridge

Interface ☒

Ethernet Adapter: "eth0" (lan)

Ethernet Adapter: "eth1" (wan, wan6)

Custom Interface:

Physical common interface setup panel.

General Setup

Advanced Settings

Physical Settings

Firewall Settings

Create / Assign firewall-zone

☒

lan:

lan:

☐

wan:

wan:

wan6:

☐

unspecified -or- create:

Choose the firewall zone you want to assign to this interface. Select unspecified to remove the interface from the associated zone or fill out the create field to define a new zone and attach the interface to it.

Firewall common interface setup panel.

General Setup

Advanced Settings

IPv6 Settings

Ignore interface ☐

Disable DHCP for this interface.

Start

Lowest leased address as offset from the network address.

Limit

Maximum number of leased addresses.

Lease time

Expiry time of leased addresses, minimum is 2 minutes (2m).

DHCP server general setup panel.

General Setup

Advanced Settings

IPv6 Settings

Dynamic DHCP ☒

Dynamically allocate DHCP addresses for clients. If disabled, only clients having static leases will be served.

Force ☐

Force DHCP on this network even if another server is detected.





IPv4-Netmask

Override the netmask sent to clients. Normally it is calculated from the subnet that is served.

DHCP-Options

Define additional DHCP options, for example "6,192.168.2.1,192.168.2.2" which advertises different DNS servers to clients.

DHCP server advanced setup panel.

General Setup	Advanced Settings	IPv6 Settings
Router Advertisement-Service	server mode ▼	
DHCPv6-Service	server mode ▼	
NDP-Proxy	disabled ▼	
DHCPv6-Mode	stateless + stateful ▼	
 Default is stateless + stateful		
Always announce default router	<input type="checkbox"/>	
 Announce as default router even if no public prefix is available.		
Announced DNS servers	<input type="text"/> 	
Announced DNS domains	<input type="text"/> 	

DHCP server IPv6 settings setup panel.

DHCP and DNS

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

Dnsmasq is a combined DHCP-Server and DNS-Forwarder for NAT firewalls

SERVER SETTINGS

General Settings

Resolv and Hosts Files

TFTP Settings

Advanced Settings

Domain required

Don't forward DNS-Requests without DNS-Name

☒

Authoritative

This is the only DHCP in the local network

☒

Local server

/lan/

Local domain specification. Names matching this domain are never forwarded and are resolved from DHCP or hosts files only

Local domain

lan

Local domain suffix appended to DHCP names and hosts file entries

Log queries

Write received DNS requests to syslog

☐

DNS forwardings

/example.org/10.1.2.3

List of DNS servers to forward requests to

Rebind protection

Discard upstream RFC1918 responses

☒

Allow localhost

Allow upstream responses in the 127.0.0.0/8 range, e.g. for RBL services

☒

Domain whitelist

ihost.netflix.com

List of domains to allow RFC1918 responses for

Local Service Only

Limit DNS service to subnets interfaces on which we are serving DNS.

☒

Non-wildcard

Bind only to specific interfaces rather than wildcard address.

☐

General DHCP settings.

General Settings

Resolv and Hosts Files

TFTP Settings

Advanced Settings

Use /etc/ethers

Read /etc/ethers to configure the DHCP-Server

☒

Leasefile

/tmp/dhcp.leases

file where given DHCP-leases will be stored

Ignore resolve file

☐

Resolve file

/tmp/resolv.conf.auto

local DNS file

Ignore /etc/hosts

☐

Additional Hosts files

Resolve and host file settings.

General Settings	Resolv and Hosts Files	TFTP Settings	Advanced Settings
Enable TFTP server		<input checked="" type="checkbox"/>	
TFTP server root		<input type="text" value="/"/>	
<small>Root directory for files served via TFTP</small>			
Network boot image		<input type="text" value="pxelinux.0"/>	
<small>Filename of the boot image advertised to clients</small>			

TFTP server settings.

General Settings	Resolv and Hosts Files	TFTP Settings	Advanced Settings
Suppress logging		<input type="checkbox"/>	
<small>Suppress logging of the routine operation of these protocols</small>			
Allocate IP sequentially		<input type="checkbox"/>	
<small>Allocate IP addresses sequentially, starting from the lowest available address</small>			
Filter private		<input checked="" type="checkbox"/>	
<small>Do not forward reverse lookups for local networks</small>			
Filter useless		<input type="checkbox"/>	
<small>Do not forward requests that cannot be answered by public name servers</small>			
Localise queries		<input checked="" type="checkbox"/>	
<small>Localise hostname depending on the requesting subnet if multiple IPs are available</small>			
Expand hosts		<input checked="" type="checkbox"/>	
<small>Add local domain suffix to names served from hosts files</small>			
No negative cache		<input type="checkbox"/>	
<small>Do not cache negative replies, e.g. for not existing domains</small>			
Additional servers file		<input type="text"/>	
<small>This file may contain lines like 'server=/domain/1.2.3.4' or 'server=1.2.3.4' for domain-specific or full upstream DNS servers.</small>			
Strict order		<input type="checkbox"/>	
<small>DNS servers will be queried in the order of the resolvfile</small>			
Bogus NX Domain Override		<input type="text" value="67.215.65.132"/>	
<small>List of hosts that supply bogus NX domain results</small>			
DNS server port		<input type="text" value="53"/>	
<small>Listening port for inbound DNS queries</small>			
DNS query port		<input type="text" value="any"/>	
<small>Fixed source port for outbound DNS queries</small>			
Max. DHCP leases		<input type="text" value="unlimited"/>	
<small>Maximum allowed number of active DHCP leases</small>			
Max. EDNS0 packet size		<input type="text" value="1280"/>	
<small>Maximum allowed size of EDNS.0 UDP packets</small>			
Max. concurrent queries		<input type="text" value="150"/>	
<small>Maximum allowed number of concurrent DNS queries</small>			

Advanced settings.

ACTIVE DHCP LEASES

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

ACTIVE DHCPV6 LEASES

Host	IPv6-Address	DUID	Leasetime remaining
There are no active leases.			

STATIC LEASES

Static leases are used to assign fixed IP addresses and symbolic hostnames to DHCP clients. They are also required for non-dynamic interface configurations where only hosts with a corresponding lease are served. Use the Add Button to add a new lease entry. The MAC-Address identifies the host, the IPv4-Address specifies the fixed address to use, and the Hostname is assigned as a symbolic name to the requesting host. The optional Lease time can be used to set non-standard host-specific lease time, e.g. 12h, 3d or infinite.

Hostname	MAC-Address	IPv4-Address	Lease time	DUID	IPv6-Suffix (hex)
This section contains no values yet					

Add

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

HOST ENTRIES

Hostname	IP address	
<input type="text" value="Host1"/>	<input type="text" value="192.168.2.35"/>	<div>Delete</div>

Add

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.

STATIC IPV4 ROUTES

Interface	Target	IPv4-Netmask	IPv4-Gateway	Metric	MTU	Route type	
	Host-IP or Network	if target is a network					
<input type="text" value="lan"/>	<input type="text" value="192.168.0.254"/>	<input type="text" value="255.255.255.255"/>	<input type="text" value="192.168.0.254"/>	<input type="text" value="0"/>	<input type="text" value="1500"/>	<input type="text" value="unicast"/>	<div>Delete</div>

Add

STATIC IPV6 ROUTES

Interface	Target	IPv6-Gateway	Metric	MTU	Route type	
	IPv6-Address or Network (CIDR)					
<input type="text" value="lan"/>	<input type="text" value="0:0:0:0:ffff:c0a8:fe"/>	<input type="text" value="0:0:0:0:ffff:c0a8:fe"/>	<input type="text" value="0"/>	<input type="text" value="1500"/>	<input type="text" value="unicast"/>	<div>Delete</div>
<input type="text" value="lan"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="0"/>	<input type="text" value="1500"/>	<input type="text" value="unicast"/>	<div>Delete</div>

Add

Current IPv4 and IPv6 static routes configuration.
Interface: Let to choose 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, or dropped.

Diagnostics

NETWORK UTILITIES

openwrt.org

IPv4

Ping

openwrt.org

IPv4

Traceroute

openwrt.org

Nslookup

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

Firewall

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

General settings

GENERAL SETTINGS

Enable SYN-flood protection

☒

Drop invalid packets

☐

Input

accept

Output

accept

Forward

reject

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

ZONES

Zone => Forwardings

Input

Output

Forward

Masquerading

MSS clamping

lan:

lan: => wan

accept

accept

accept

☐

☐

Edit

Delete

wan:

wan: => wan6

reject

accept

reject

☐

☒

Edit

Delete

wan6:

wan6: => REJECT

Add


Additional zones for the firewall can be created, edited, or deleted.
Zone => Forwardings: Defines zones and their traffic flow.
Input: All incoming packets can be: accepted, rejected, or dropped.




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

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

Masquerading: Allows one or more devices in a zone 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.

General Settings	Advanced Settings
Name	<input type="text" value="lan"/>
Input	<input type="text" value="accept"/>
Output	<input type="text" value="accept"/>
Forward	<input type="text" value="accept"/>
Masquerading	<input type="checkbox"/>
MSS clamping	<input type="checkbox"/>
Covered networks	
<input checked="" type="checkbox"/> lan: 	
<input type="checkbox"/> wan: 	
<input type="checkbox"/> wan6: 	
<input type="checkbox"/>	
	create: <input type="text"/>

Common properties of newly created or edited zones can 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.

General Settings	Advanced Settings
Restrict to address family	<input type="text" value="IPv4 and IPv6"/>
Restrict Masquerading to given source subnets	<input type="text" value="0.0.0.0/0"/>
Restrict Masquerading to given destination subnets	<input type="text" value="0.0.0.0/0"/>
Force connection tracking	<input type="checkbox"/>
Enable logging on this zone	<input type="checkbox"/>

Advanced settings of newly created or edited zones. 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. Connection tracking and logging options enable additional information gathering on the zone.

Allow forward to destination zones:	<div>wan:   </div>
<input checked="" type="checkbox"/>	
Allow forward from source zones:	<div>wan:   </div>
<input type="checkbox"/>	

Controls 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 permission to forward from WAN to LAN as well.

Port forwards

PORT FORWARDS

Name	Match	Forward to	Enable	Sort	
4000	IPv4-tcp From any host in wan Via any router IP at port 4000	IP 192.168.2.1, port 4000 in lan	<input checked="" type="checkbox"/>	▲ ▼	<div>Edit</div> <div>Delete</div>
4001	IPv4-tcp, udp From any host in wan Via any router IP at port 4001	IP 192.168.2.1, port 4001 in lan	<input checked="" type="checkbox"/>	▲ ▼	<div>Edit</div> <div>Delete</div>

New port forward:

Name	Protocol	External zone	External port	Internal zone	Internal IP address	Internal port	
New port forward	TCP+UDP	wan		lan			<div>Add</div>

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

Name	Match	Action	Enable	Sort	
Allow-DHCP-Renew	IPv4-UDP From any host in wan To any router IP at port 68 on this device	Accept input	<input checked="" type="checkbox"/>	▲ ▼	<div>Edit</div> <div>Delete</div>
Allow-Ping	IPv4-ICMP with type echo-request From any host in wan To any router IP on this device	Accept input	<input checked="" type="checkbox"/>	▲ ▼	<div>Edit</div> <div>Delete</div>
Allow-IGMP	IPv4-IGMP From any host in wan To any router IP on this device	Accept input	<input checked="" type="checkbox"/>	▲ ▼	<div>Edit</div> <div>Delete</div>
Allow-DHCPv6	IPv6-UDP From IP range fc00::/6 in wan To IP range fc00::/6 at port 546 on this device	Accept input	<input checked="" type="checkbox"/>	▲ ▼	<div>Edit</div> <div>Delete</div>

Traffic rules 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. Every rule can be defined by 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.

Name	Match	Action	Enable	Sort
This section contains no values yet				
New source NAT:				
Name	Source zone	Destination zone	To source IP	To source port
<input type="text" value="New SNAT rule"/>	<input type="text" value="lan"/>	<input type="text" value="wan"/>	<input type="text" value="Do not rewrite"/>	<input type="text" value="Do not rewrite"/>
				<input type="button" value="Add and edit..."/>

Source NAT is a specific form of masquerading that allows fine-grained control over the source IP used for outgoing traffic, for 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

```
# This file is interpreted as shell script.
# Put your custom iptables rules here, they will
# be executed with each firewall (re-)start.

# Internal uci firewall chains are flushed and recreated on reload, so
# put custom rules into the root chains e.g. INPUT or FORWARD or into the
# special user chains, e.g. input_wan_rule or postrouting_lan_rule.
```

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

Setting up the L2TP interface

To create an L2TP tunnel following steps are required:

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

INTERFACES
DHCP AND DNS
HOSTNAMES
STATIC ROUTES
FIREWALL
DIAGNOSTICS

Interfaces

INTERFACE OVERVIEW


Network	Status	Actions
<div>LAN</div> <div>br-lan</div>	Uptime: 17h 24m 19s MAC-Address: BC:24:11:AA:84:50 RX: 5.60 MB (72138 Pkts.) TX: 7.03 MB (28304 Pkts.) IPv4: 192.168.1.1/24 IPv6: fd02:62d:1b2c::1/60	<div>Connect/Reconnect</div> <div>Stop</div> <div>Edit</div> <div>Delete</div>
<div>WAN</div> <div>eth1</div>	Uptime: 17h 13m 10s MAC-Address: BC:24:11:7F:10:8F RX: 498.54 KB (5179 Pkts.) TX: 580.48 KB (5962 Pkts.) IPv4: 192.168.69.103/24	<div>Connect/Reconnect</div> <div>Stop</div> <div>Edit</div> <div>Delete</div>
<div>WAN6</div> <div>eth1</div>	Uptime: 0h 0m 0s MAC-Address: BC:24:11:7F:10:8F RX: 498.54 KB (5179 Pkts.) TX: 580.48 KB (5962 Pkts.)	<div>Connect/Reconnect</div> <div>Stop</div> <div>Edit</div> <div>Delete</div>

Add new interface...

2. Enter the interface name and select L2TP protocol:

Create Interface

Name of the new interface

 The allowed characters are: A-Z, a-z, 0-9 and _

Note: interface name length

 Maximum length of the name is 15 characters including the automatic protocol/bridge prefix (br-, 6in4-, pppoe- etc.)

Protocol of the new interface

L2TP

▼

Back to Overview

Submit


3. Enter the server name and authorization parameters:

COMMON CONFIGURATION

General Setup

Advanced Settings

Firewall Settings

 **RX:** 0 B (0 Pkts.)
TX: 0 B (0 Pkts.)

i2tp-i2tp

Status

Protocol


L2TP Server

PAP/CHAP username

PAP/CHAP password

L2TP

▼



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

5.8 Users

Edit groups

GROUPS OVERVIEW

Groups	Status	Actions	
administrator	Authorization level: 11	Edit	Delete
engineer	Authorization level: 5	Edit	Delete
operator	Authorization level: 3	Edit	Delete
viewer	Authorization level: 1	Edit	Delete

Add New Group...

On this page, user groups can be edited, deleted, or added.

Groups: name of the user group

Status: shows authorization level set to a specific user group. The higher the lever, the higher the authorization requirements.

Actions: edit or delete a user group

Add new group

Group Name

Access level

?

Access level this group refers to. Use with external PAM module (e.g. RADIUS)

Enable Phub menus

View

Edit

Enable Users menus

View

Edit

Enable Services menus

View

Edit

Enable Status menus

View

Edit

Enable Network menus

View

Edit

Enable System menus

View

Edit

Back to Overview

Save & Apply

Save

Reset

Configuration window for the new group. After the group name is determined, access level and permissions can be set.

Edit users

Users

USERS OVERVIEW

Users	Status	Actions
<div>user</div> <div></div>	SSH Access: Enabled Group: viewer Date Added: Thu Dec 14 14:39:11 2023 Last Entry: undefined	<div>Edit</div> <div>Change Password</div> <div>Delete</div>

Add New User...

On the edit users window list of all the users is shown.

Users: user name

Status: shows if SSH access is enabled and which group the user belongs to.

Actions: edit, delete, or change the password for the user

Add new user

User Name

User Group

viewer

SSH Access

Enabled

Password



Back to Overview

Save & Apply

Save

Reset

Configuration window for new users. To create a new user, a name and password should be created and user group and SSH access should be set.

Password

Password



Confirmation



Save & Apply

Save

Reset

Changes the password of the device.

5.9 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 suspected of any deliberate damage made by unauthorized access.

6 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 the empty row are discarded.

6 Excel Configuration

6.1 Devices sheet

The device sheet contains all devices to be configured on the gateway. Each row represents one device and its settings. The 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 the 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 the 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.

6 Excel Configuration

6.2 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.

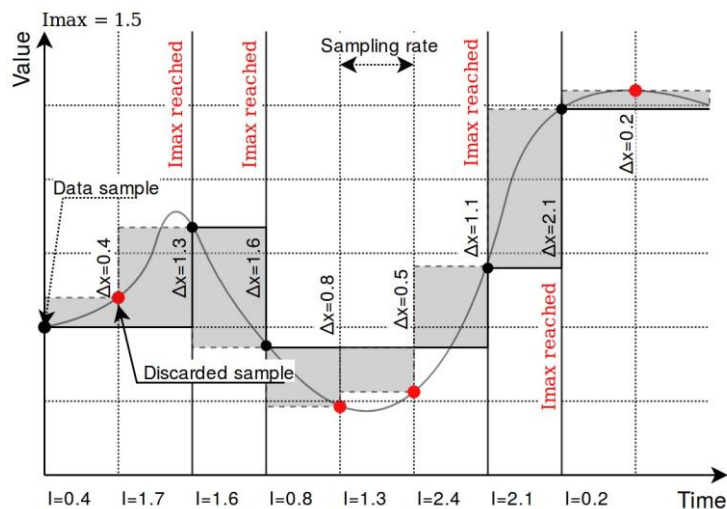
Optional attributes

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

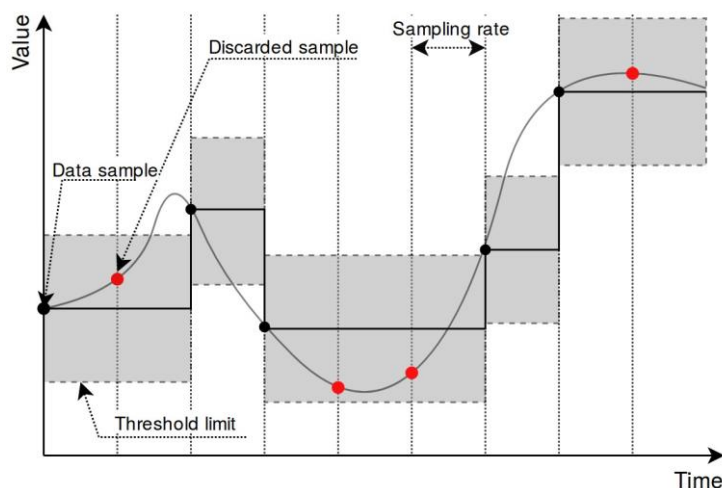
- **source_device_alias** - Alias of a source device defined in the Devices sheet. If a user intends to use several signals and combine them via mathematical or logical function, every alias should be separated by a newline symbol (in the same cell). An operation used must also be defined in an operation 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 measurement 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_ms** - 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 the 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**. The 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). An internal threshold is used to reduce value updates when the value doesn't change. 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 the **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 the **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 the 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 to 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 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**, and **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 the 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. The 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 5	Byte 4	Byte 7	Byte 6	Byte 1	Byte 0	Byte 3	Byte 2
SW8.SW16.SW32	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

❗ 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** to correctly parse a 32-bit signed integer in a little-endian format. The table 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 to the IEC 60870-5-104 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 the slave device signal to match **device_alias** and **signal_alias** on the master device's coil signal.

Example 2

To write a coil state to a Modbus device on a command from IEC 60870-5-104 station, 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 to 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)
- ","
- ";"

6 Excel Configuration

6.3 Mathematical functions

The signal value might require some recalculation or signal update before 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.

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 tangent function (rad)
sinh	1	hyperbolic sine function
cosh	1	hyperbolic cosine
tanh	1	hyperbolic tangent function
asinh	1	hyperbolic arcus sine function
acosh	1	hyperbolic arcus tangent function
atanh	1	hyperbolic arcus tangent 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	the square root of a value

sign	1	sign function -1 if $x < 0$; 1 if $x > 0$
rint	1	round to the nearest integer
abs	1	absolute value
min	variable	min of all arguments
max	variable	max of all arguments
sum	variable	the sum of all arguments
avg	variable	the 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 hyperbolic functions) only support arguments in radians. This means that arguments for this function have to be recalculated if the angle is defined in degrees.
- Value recalculation is only triggered on signal change of the preconfigured signal. That means that using other signals (via TagValue() call) does not trigger a value update.

Some mathematical expressions cannot be mathematically evaluated in some conditions, for example, a square root cannot be found for negative numbers. As complex numbers are not supported, the 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 the condition is true, the result after the question sign is selected. If the condition is false, the 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 equations by using the aforementioned operators and functions. These examples can be seen in the 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

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

The inner argument of TagValue() function has to be 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 the 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, the "out" keyword marks data sent out of the WCC Lite+ device, whereas the "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 is presented the way it was read by an individual protocol.

Signal mathematics


In this section, you will be shown how "math_expression" and other mathematical functions can be used in case of common signals. You can download the configuration to follow along [here](#). Signals which we are concerned with in this section are highlighted in green color.

Let us analyze what mathematical functions are configured for the signals. For both the second and third signals the same expression will be used: **"value + TagValue("tag/Master/RHR0/out")** ". The only difference is that for the second signal scale function "add" was used.

signal_name	device_alias	signal_alias	source_device_alias	source_signal_alias	enable	job_todo	tag_job_todo	number_type	multiply	add	bit_select	source_math_expression	math_expression
Read Holding Register 0	Master	RHR0			1	3;0;36	3;0;1	SIGNED16					
Write Single Register 1	Master	WSR1	Master	RHR0	1	6;0;36	6;1;1	SIGNED16					
Read Holding Register 2	Master	RHR2			1	3;0;36	3;2;1	SIGNED16		5			value + TagValue("tag/Master/RHR0/out")
Write Single Register 3	Master	WSR3	Master	RHR2	1	6;0;36	6;3;1	SIGNED16					
Read Holding Register 4	Master	RHR4			1	3;0;36	3;4;1	UNSIGNED16					value + TagValue("tag/Master/RHR0/out")

In this signal configuration, the value of the second signal is calculated by adding the current value of the second signal to the value of the first signal. Then the sum of two signals is going to be increased by 5. The third signal is going to be calculated in the same way except that 5 is not going to be added.

To see how it works let us start Modbus TCP Slave simulation in Vinci application. You can download the simulation [here](#).



Protocol:
Mode:

IP:
Port:

STOP

Settings
Console
Statistic

Slave	Function	Address	Value	FormattedValue	Name
1	3	0	1	1	Register0
1	3	1	1		
1	3	2	2	2	Register2
1	3	3	8		
1	3	4	3	3	Register4

In the picture above you can see 6 registers. However, our main focus is null, second, and fourth registers (Register0, Register2, Register4) since the first three signals of Modbus Master protocol (RHR0, RHR2, and RHR4) are reading the values of those registers (accordingly).

Let us go to the WCC Lite+ web interface to see how these signals are displayed there:

IMPORTED SIGNALS						
Device	Signal	Value	Units	State	Attributes	Time
Master Protocol	Read Holding Register 0	1				2023-07-26 12:07:09.35
Master Protocol	Write Single Register 1	1			cot=10	2023-07-26 12:07:10.73
Master Protocol	Read Holding Register 2	8				2023-07-26 12:07:09.58
Master Protocol	Write Single Register 3	8			cot=10	2023-07-26 12:07:10.74
Master Protocol	Read Holding Register 4	4				2023-07-26 12:07:10.21

As we can see the values of the third and fifth signals have been modified ($RHR2 = 2 + 1 + 5 = 8$; $RHR3 = 3 + 1 = 4$). However, the values of the signals that are displayed in the web interface are intermediate so to speak. All the math is done in the protocol services (Modbus TCP Master in this case). Then those values are transmitted to the REDIS service. The values that are displayed in the web interface are REDIS values. We are going to see why it is important in another example.

Now it was mentioned that the values of the third and fifth signals depend on the value of the first one. Let us see what will happen if we change the value of the Register 0. To do so we need to return to the VINCI application. Locate Register 0 and double-click on it. A menu with the register parameters will appear.

Protocol: Modbus TCP

Mode: Slave (Server) ▼

STOP

IP: 192.168.1.2

Port: 502

Settings

Console

Statistic

Slave	Function	Address	Value	FormattedValue	Name
1	3	0	1	1	Register0

▼ Tag

Name:

Type:

Slave:

Address:

Format:

Value:

Save

Cancel

▼ Tag

— □ ×

Now it would be expected that the value of the second and third signals would become 9 and 5 accordingly ($RHR2 = 2+2+5 = 9$; $RHR3 = 3+2 = 5$). However if one checks the WCC Lite+ web interface right after that, one could notice that the second and third signals remained unchanged:

CONFIGURATION

IMPORTED SIGNALS

EVENT LOG


PROTOCOL CONNECTIONS

SCRIPT-RUNNER

IMPORTED SIGNALS

Device	Signal	Value	Units	State	Attributes	Time
Master Protocol	Read Holding Register 0	2				2023-07-26 12:59:26.77
Master Protocol	Write Single Register 1	2			cot=10	2023-07-26 12:59:26.86
Master Protocol	Read Holding Register 2	8				2023-07-26 12:48:42.64
Master Protocol	Write Single Register 3	8			cot=10	2023-07-26 12:48:42.91
Master Protocol	Read Holding Register 4	4				2023-07-26 12:48:42.64
Master Protocol	Write Single Register 5	4			cot=10	2023-07-26 12:48:42.94

To explain this let us look again at the math expression of these signals. The equation (**value + TagValue("tag/Master/RHR0/out")**) consists of two operands "value" and "TagValue("tag/Master/RHR0/out")". Currently, the system is designed in such a way that only if the "value" operand has changed, only then there is going to be a change in a signal's value. So if the values of the second and fourth registers are changed (increased by one in this example) then the values of the third and fifth signals are going to change taking into account the previous change in the value of Register 0 ($RHR2 = 2+3+5 = 10$; $RHR3 = 2+4 = 6$).



Protocol:
 Mode:

IP:
 Port:

STOP

Settings
 Console
 Statistic

Slave	Function	Address	Value	FormattedValue	Name
1	3	0	2	2	Register0
1	3	1	2		
1	3	2	3	3	Register2
1	3	3	10		
1	3	4	4	4	Register4
1	3	5	6		

CONFIGURATION

IMPORTED SIGNALS

EVENT LOG

PROTOCOL CONNECTIONS

SCRIPT-RUNNER

IMPORTED SIGNALS

Device	Signal	Value	Units	State	Attributes	Time
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Master Protocol	Read Holding Register 0	2				2023-07-26 12:59:26.77
Master Protocol	Write Single Register 1	2			cot=10	2023-07-26 12:59:26.86
Master Protocol	Read Holding Register 2	10				2023-07-26 13:30:23.64
Master Protocol	Write Single Register 3	10			cot=10	2023-07-26 13:30:23.73
Master Protocol	Read Holding Register 4	6				2023-07-26 13:30:28.68
Master Protocol	Write Single Register 5	6			cot=10	2023-07-26 13:30:28.77


Command mathematics

In this section, you will be shown how "math_expression", and other mathematical functions can be used in case of command signals. You can download the configuration to follow along [here](#). Signals which we are concerned with in this section are highlighted in blue color.

Let us analyze what mathematical functions are configured for the signals.

signal_name	device_alias	signal_alias	source_device_alias	source_signal_alias	enable	job_todo	tag_job_todo	number_type	multiply	add	bit_select	source_math_expression	math_expression
Read Holding Register 0	Master	RHR0			1	3;0;36	3;0;1	SIGNED16	2				
Write Single Register 1	Master	WSR1	Master	RHR0	1	6;0;36	6;1;1	SIGNED16					
Read Holding Register 2	Master	RHR2			1	3;0;36	3;2;1	SIGNED16					
Write Single Register 3	Master	WSR3	Master	RHR2	1	6;0;36	6;3;1	SIGNED16	2				
Read Holding Register 4	Master	RHR4			1	3;0;36	3;4;1	SIGNED16					
Write Single Register 5	Master	WSR5	Master	RHR4	1	6;0;36	6;5;1	SIGNED16				value - TagValue("tag/Master/RHR0/out")	value + TagValue("tag/Master/RHR0/out")
Read Holding Register 6	Master	RHR6			1	3;0;36	3;6;1	SIGNED16					
Write Single Register 7	Master	WSR7	Master	RHR6	1	6;0;36	6;7;1	SIGNED16			3		

The first four signals are going to be used for scale function analysis. To see how it works let us start Modbus TCP Slave simulation in Vinci application. You can download the simulation [here](#).



Protocol:
Mode:

STOP

IP:
Port:

Settings
Console
Statistic

Slave	Function	Address	Value	FormattedValue	Name
1	3	0	4	4	Register0
1	3	1	8		
1	3	2	4	4	Register2
1	3	3	2		

Let us go to the WCC Lite+ web interface to see how these signals are displayed there:

CONFIGURATION

IMPORTED SIGNALS

EVENT LOG

PROTOCOL CONNECTIONS

SCRIPT-RUNNER

IMPORTED SIGNALS

Device	Signal	Value	Units	State	Attributes	Time
Master Protocol	Read Holding Register 0	8				2023-07-26 15:03:53.17
Master Protocol	Write Single Register 1	8			cot=10	2023-07-26 15:03:53.37
Master Protocol	Read Holding Register 2	4				2023-07-26 15:03:53.17
Master Protocol	Write Single Register 3	4			cot=10	2023-07-26 15:03:53.40

As one can notice the values displayed in the Vinci application differ from the values displayed in a web interface. To better understand the results let us see step by step how signals are modified.

The first signal RHR0 reads the value of Register 0 which is equal to 4. After the read operation, the value is multiplied by two in the master protocol service. Then a signal is transmitted to the REDIS service. REDIS value is presented in the web interface. Then the same value 8 is written to Register 1 by sending a WSR1 command signal. Finally, the value of the WSR1 signal is displayed in the Vinci simulation.

The third signal RHR2 behaves quite the same as the first RHR0 signal. It reads a value of Register 2 and without performing any scaling operations transmits value to REDIS. It again can be seen in the web interface. While still in REDIS service the value of Register 2 is passed to WSR3 signal. Then command signal WSR3 is transmitted back to the Modbus Master protocol service where the value of the signal is scaled. One could expect that the value of the signal would be multiplied by two but it is divided by two. Then the value of 2 is displayed in the Vinci Slave simulation. After changing the value of Register 3 WSR3 signal is sent back from the Master Protocol service to REDIS. On its way, the signal again passes the signal scaling place where it is again multiplied by two. This is why we see in the web interface that the value of the fourth signal WSR3 is 4.

All the moments when a signal passes a place where its value is scaled can be seen by turning on a debugger session. To do so one should connect to WCC Lite+ via Ubuntu terminal in Terminal application. Then Modbus Master protocol needs to be stopped by sending `"/etc/init.d/modbus-master stop"`. Then Modbus Master protocol needs to be started again with the -m flag (m for math) `"modbus-master -d7 -m -c /etc/modbus-master/modbus-tcp.json"`.

```
root@wcc-lite:~# /etc/init.d/modbus-master stop
root@wcc-lite:~# modbus-master -d7 -m -c /etc/modbus-master/modbus-tcp.json
```

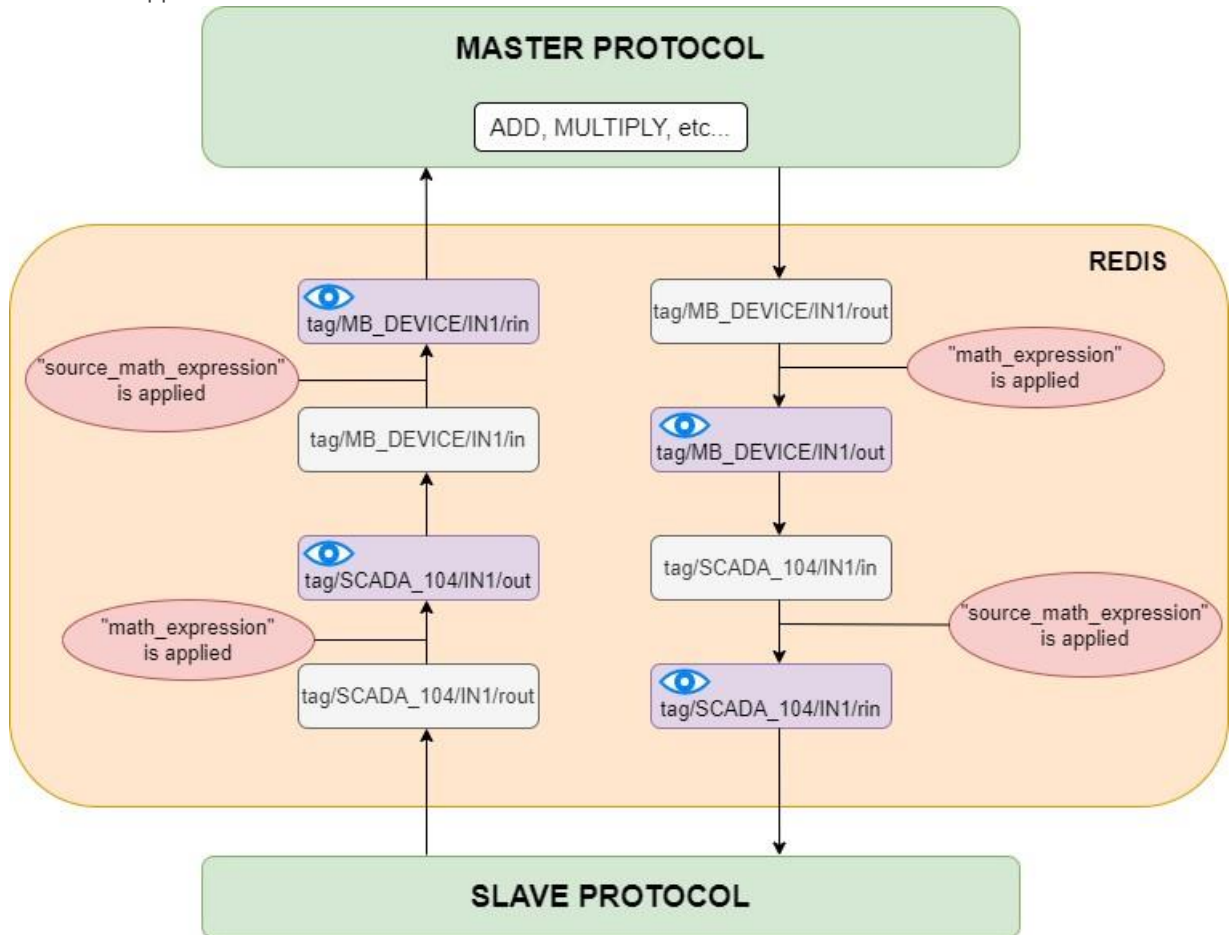

If you scroll up after starting the session you will be able to find how both RHR0 and WSR3 signals are scaled after passing the signal scaling place.

```
[DEBUG] SERVICE_MATH: Signal: tag/Master/RHR0/rout
[DEBUG] SERVICE_MATH: Value: 4
[DEBUG] SERVICE_MATH: value * 2 = 8

[DEBUG] SERVICE_MATH: Signal: tag/Master/WSR3/rin
[DEBUG] SERVICE_MATH: Value: 4
[DEBUG] SERVICE_MATH: (value) / 2 = 2
[DEBUG] SERVICE_MATH: Signal: tag/Master/WSR3/rout
[DEBUG] SERVICE_MATH: Value: 2
[DEBUG] SERVICE_MATH: value * 2 = 4
```

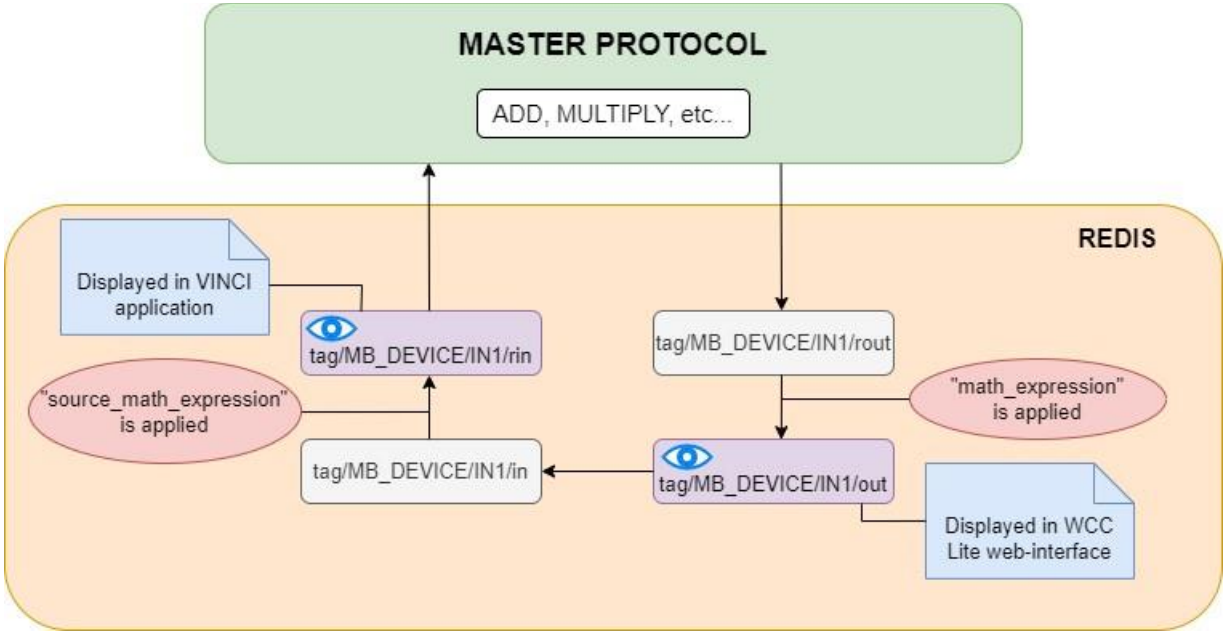
Why is it the case that the signal is divided instead of being multiplied by two? The answer to this question is that the scaling factor depends on the direction of the signal. When the RHR0 signal was traveling from Master Protocol to REDIS service and was passing the place where signals are scaled the signal was multiplied by the scaling factor. On the other hand, when the WSR3 signal was traveling from REDIS service to Master Protocol and was passing the same signal scaling place the signal was divided by the scaling factor.

Now let us analyze how "math_expression" and "source_math_expression" are applied to the WSR5 signal. To get a better insight let us look at how signals are transmitted and transformed inside WCC Lite and when mathematical expressions are applied.



Let us analyze the diagram above. As we can see all basic mathematical operations (such as add, multiply, etc.) are performed inside a Master protocol service. When signals are transmitted between protocols they travel through REDIS service. The period of existence of a signal inside the REDIS service can be divided into four stages. They can be denoted by their endings, namely: "rout", "out", "in" and "rin". At the "out" and "rin" stages signal values can be seen through certain interfaces which is why they are displayed in a purple color with blue eyes on them. At the "out" stage a signal is displayed on the WCC Lite+ web interface, at the "rin" stage signal value can be seen through the Vinci application. As one can notice "math_expression" is applied before the "out" stage and "source_math_expression" is performed before the "rin" stage.

However, in our particular example, we did not configure any Slave device. In this situation signal transportation inside REDIS service can be depicted as follows:



Let us return to our example and let us see what are the values of the WSR5 signal in the WCC Lite+ web interface and Vinci Slave simulation.

CONFIGURATION

IMPORTED SIGNALS

EVENT LOG

PROTOCOL CONNECTIONS

SCRIPT-RUNNER

IMPORTED SIGNALS

Device	Signal	Value	Units	State	Attributes	Time
Master Protocol	Read Holding Register 4	10				2023-07-27 15:51:16.56
Master Protocol	Write Single Register 5	10			col=10	2023-07-27 15:51:17.40

Protocol: Modbus TCP

Mode: Slave (Server)

STOP

IP: 192.168.1.2

Port: 502

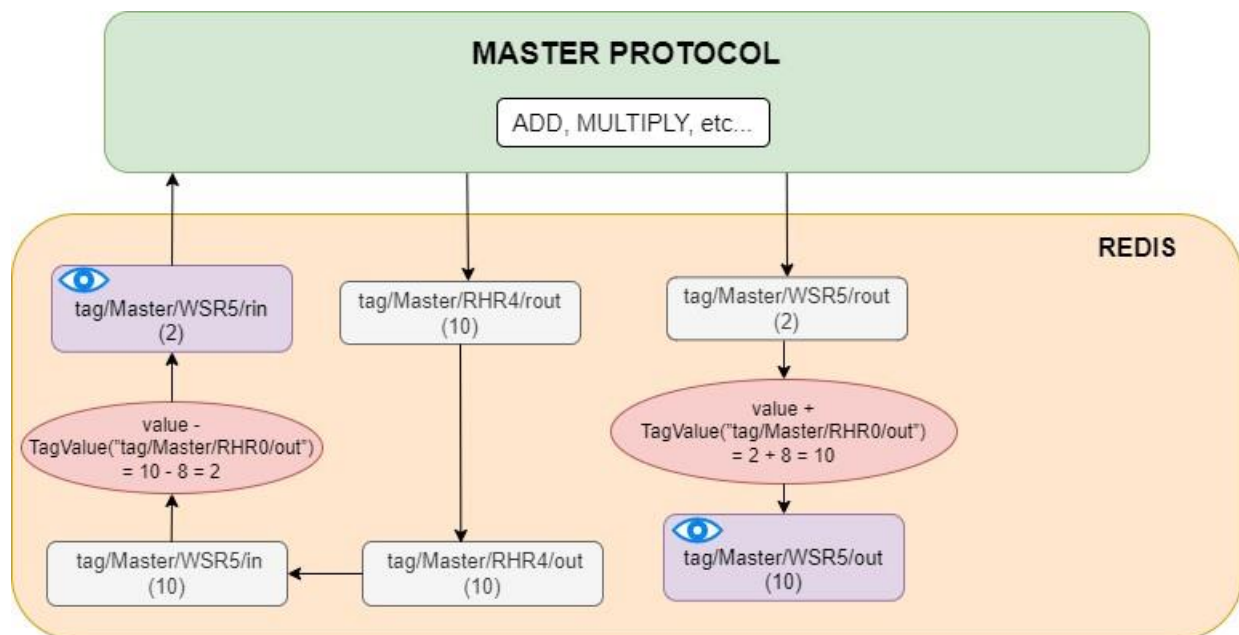
Settings

Console

Statistic

Slave	Function	Address	Value	FormattedValue	Name
1	3	4	10	10	Register4
1	3	5	2		

The diagram below explains how these values were calculated.



Extra functions

Several functions are defined to make tag operations possible:

- `TagValue(key)` - returns the last known value of the tag identified by the Redis key;
- `TagFlag(key)` - returns 1 if the tag flag exists. The name format is: "key flag". For example, to check if a tag is non-topical, the 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 the UNIX timestamp in milliseconds of the last known tag value.

6 Excel Configuration

6.4 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 in a file called phub.xlsx and later downloaded with a custom-built name reflecting the date of a download. The upgrade process from the 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, they must first download the configuration files and later re-upload 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 the configuration complexity
- Supported types of an Excel configuration: .xlsx, .xlsm, .xltm, .xltx

To upload an Excel file, open Protocol Hub --> Configuration screen in the 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 used for the import of an Excel file.

Uploading configuration remotely

As of the 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 that 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 the 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 upload process, the aforementioned button is disabled to prevent spanning multiple concurrent processes.

6 Excel Configuration

6.5 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.

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 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 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.

7 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 milliseconds 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 `board`, `netstat` and `process`. `tag_job_todo` defines the job sub job. This field should address the particular point of interest. There is also an optional extra trigger parameter. The default trigger is `value>0`. When a **trigger** column is added the trigger can be changed by entering i.e. `"value>10"`.

job_todo	Description	tag_job_todo	Description
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
netstat[[interface]	Network statistics	TX	Bytes transferred
		RX	Bytes received
process	Check if the process is running	[process name]	1 or 0 is returned
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

8 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 labels.

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. The INDETERMINATE state might, for example, mean that part of the equipment has been turned off or a mechanical part that 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

9 API

The firmware of the WCC Lite+ features a built-in API which is accessible via the web interface.

The API (as of version 1.2.0) implements API key authorization, that is generated on the first boot of the device. The authorization can be enabled or disabled and the API key can be changed via the web configuration interface at Services → API.

Authentication

API key authentication is enabled by default and it generates a random 12-character long key. The key is stored in the same uci configuration as API.

To change the API key via the interface, it must be equal to or longer than 12 characters.

When API authentication is enabled WCC Lite+ checks if the "apiKey" header exists in the request and then validates its value against the one in the configuration. If the value is correct it continues with the API request, if the apiKey is incorrect or doesn't exist it returns an error response 401 with the message.

```
{
  "error": "Unauthorized access."
}
```

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

All endpoints are disabled by default.

All endpoints are disabled by default. Available API endpoints are shown in the table below.

Path	Description	Request body (encoded as "multipart-formdata")	Response body example (json format)
------	-------------	---	--

/api/version	Version of the API		{ "api_version": "2", "api_patch": "1", "sync_version": "lite1" }
/api/actions	List of available points		
/api/syncVersion	A version of the sync service		
/api/sync	Protocol hub configuration upload (key name="file"). Returns excel-utility output)*	{ "file": "config.xlsx" }	
/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	Returns configured device information. (key name="device_alias")**	{ "device_alias": "INV1" }	{ "devices": [{ "device_alias": "INV1", "name": "INV1", "protocol": "modbus rtu" }] }
/api/device/tags	List of tags of a particular configured device. (key name="device_alias")**	{ "device_alias": "INV1" }	{ "total_signals": "7", "signals": [{ "signal_name": "Active power", "source_device_alias": "", "input_topic": "tag/INV1/P/rin", "output_topic": "tag/INV1/P/out", "signal_alias": "P" }], ... }
/api/device/tag/value	Returns tags value, name, timestamp, and flags of a particular device. (key name="device_alias", key name="signal_alias")**	{ "device_alias": "INV1", "signal_alias": "P" }	{ "total_signals": "1", "signal_values": [{ "signal_name": "Active power", "value": "5.9589999999999996", "timestamp": "1708000203824", "flags": "" }] }
/api/tags	List of configured tags		
/api/sysupgrade	Firmware upgrade (name="file")*		

* Endpoints accepting files (requires WCC Lite Firmware version 1.8 or newer) **

Endpoints accepting field data (requires WCC Lite firmware version 1.8 or newer)

The API accepts data and files as POST requests encoded as "multipart/formdata".

If API authorization is enabled, you must add a header "apiKey" with the correct key.

10 Certificates

Devices that send unencrypted data are susceptible to attacks that might cause deliberate damage to the user system. Therefore it is highly advised to use cryptography to secure 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, a 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 users and users with group level 15 permissions;
- By default, certificates are not added to the backup to avoid private key leakages; Private keys should never be revealed to the 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) certificates, certificates from peers (usually called Certificate Authority (CA)), and private keys. 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 a minimal understanding of the formats in which certificates are stored.

Certificates should conform to the X509 standard. The difference between local certificates 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 certificates 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 the public key enables 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 uses 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 suggests 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

File name	Valid from	Valid until	Issuer	Subject	
alice.crt	Apr 27 10:31:18 2012 GMT	Apr 25 10:31:18 2022 GMT	Freelan Sample Certificate Authority	alice	Delete
ca.crt	Apr 27 10:17:44 2012 GMT	May 27 10:17:44 2012 GMT	Freelan Sample Certificate Authority	Freelan Sample Certificate Authority	Delete
alice.key					Delete

Browse... No file selected.

Upload

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

To find out more about why certificates help keep devices secure please check the sectionCyber security or check X.509 and RFC 5755 standard.

11 Cyber security

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

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

The 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: the admin is generally entitled to make changes while the user does not have any editing permissions, and the relevant buttons are disabled. Users can be assigned to one of fifteen user groups that can access different amounts of device parameters. The highest (fifteenth) permission level grants the same permission as the root user has. User group rights can be edited to give more rights or restrictions, except for the 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 the remote RADIUS Server. Login into WCC Lite+ is available only if the RADIUS Server will grant authentication and will provide a user profile with user rights on that device (more detailed description below). This also means that a password for such a user cannot be changed remotely.
- Authentication via external RADIUS Server with fallback option. In this scenario, users will be authenticated via the RADIUS server. If a 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 an external RADIUS server, a user should first enable the RADIUS process and configure at least one server.

Locally stored credentials management

The device has predefined default users like root and user.

PROTOCOL HUBSTATUSSYSTEMSERVICESNETWORKUSERSLOGOUT (ROOT)

EDIT GROUPSE~~EDIT USERS~~PASSWORD

Users

USERS OVERVIEW

Users	Status	Actions
<div>user</div>	<div>SSH Access: Enabled</div> <div>Group: viewer</div> <div>Date Added: Thu Dec 14 14:39:11 2023</div> <div>Last Entry: undefined</div>	<div>Edit</div> <div>Change Password</div> <div>Delete</div>

Add New User...

Screen containing all users

PROTOCOL HUB	STATUS	SYSTEM	SERVICES	NETWORK	USERS	LOGOUT (ROOT)	
--------------	--------	--------	----------	---------	--------------	---------------	--

[EDIT GROUPS](#)
[EDIT USERS](#)
[PASSWORD](#)

Add New User

User Configuration

User Name

User Group

viewer

SSH Access

Enabled

Password

[Back to Overview](#)
[Save & Apply](#)
[Save](#)
[Reset](#)

Screen for new user configuration

root user has full permission set to connect to WCC Lite+ over the web interface and SSH or Telnet. This user is the 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 to attacks.

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

The system allows customers 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. The user should enter a user name, user groups to which the user should belong (the group must be preconfigured first), SSH access permission as well as password. When editing user settings, only User Group and SSH Access permission can be changed. To change a user password, the Change Password button should be pressed as seen in the figure above to lead the user to a screen seen in the figure below.

PROTOCOL HUB	STATUS	SYSTEM	SERVICES	NETWORK	USERS	LOGOUT (ROOT)	
--------------	--------	--------	----------	---------	--------------	---------------	--

[EDIT GROUPS](#)
[EDIT USERS](#)
[PASSWORD](#)

Router Password

Changes your password for accessing the device

Password

Confirmation

[Save & Apply](#)
[Save](#)
[Reset](#)

Changing user password

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

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

The following commands may be used in the command 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 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 a newly created user account via both commandline interface and web interface he should create and delete users via web interface and not use `adduser` and `deluser` commands as they don't create uci entries.

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

Authentication via external service

WCC Lite+ supports external authentication 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 WCC Lite+ web login form

This request includes access credentials, typically in the form of a username and password. Additionally, the request may contain other information that 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 the 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 the 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, the Radius Server returns Access-Reject and the login fails. If the RADIUS server is not available and the fallback option is disabled login into WCC Lite+ will be impossible. If the 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 log into the system.
- Device boot event, when the system was started.
- Device reboot/halt event.
- Configuration changes.
- Firmware changes.

Date and time changes in the system (excluding automatic system time updates over NTP or IEC 60870510x protocol)

 Enabling external system log server setup in System properties --> Logging is recommended. The system stores logs in RAM by default due to limited flash storage. Rebooting or powering off the device will result in a 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 a username and password on an unsecured connection (e.g. telnet)
3. ask for a username and password on an encrypted connection (e.g. SSH) (e.g. by following first login)
4. ask for a username and merely a signature instead of a password (e.g. SSH with signature.authentication)

If you ask for a 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 the 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 the external/WAN port by blocking incoming connections to those ports (TCP 22, 80, and 443 by default) in your firewall
- 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 the
 - 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 **Administration --> System > SSHkeys**. 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 the user is logged on via an external server, its authentication level is acquired. As no direct mapping to existing users is used, authentication levels are a way to grant proper permissions for external users. WCC Lite+ uses a CISCO authentication 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 to WCC Lite+ is made by the Dropbear software package. To extend the basic functionality, the Pluggable Authentication Module (PAM) for RADIUS is used. This enables a user to add his authentication modules as long as they are properly configured.

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

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

Protocol Hub

Status


System

Services

Network

Users

Logout (Root)


WCC LITE+

Edit Groups

Edit Users

Password

Groups

Groups Overview

Groups	Status	Actions	
administrator	Authorization level: 11	Edit	Delete
engineer	Authorization level: 5	Edit	Delete
operator	Authorization level: 3	Edit	Delete
viewer	Authorization level: 1	Edit	Delete

Add New Group...

administrator

Group Name

administrator

Access level

11

Access level this group refers to. Use with external PAM module (e.g. RADIUS)

<div>Enable Users menus</div> <div> <input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Edit </div>	<div> <input checked="" type="checkbox"/> Edit groups <input type="checkbox"/> Password </div>
<div>Enable Services menus</div> <div> <input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Edit </div>	
<div> <input checked="" type="checkbox"/> Telemetry agent <input checked="" type="checkbox"/> OpenVPN </div>	<div> <input checked="" type="checkbox"/> IPsec API <input checked="" type="checkbox"/> ser2net </div>
<div>Enable Status menus</div> <div> <input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Edit </div>	
<div> <input checked="" type="checkbox"/> Firewall <input checked="" type="checkbox"/> Kernel Log <input checked="" type="checkbox"/> Processes </div>	<div> <input checked="" type="checkbox"/> Routes <input type="checkbox"/> Chronos <input type="checkbox"/> GSM Status </div> <div> <input checked="" type="checkbox"/> System Log <input checked="" type="checkbox"/> Realtime Graphs <input checked="" type="checkbox"/> VnStat Traffic Monitor </div>
<div>Enable Network menus</div> <div> <input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Edit </div>	
<div> <input type="checkbox"/> Interfaces <input checked="" type="checkbox"/> Hostnames <input checked="" type="checkbox"/> Diagnostics </div>	<div> <input checked="" type="checkbox"/> Wireless <input checked="" type="checkbox"/> Static Routes <input type="checkbox"/> GSM </div> <div> <input checked="" type="checkbox"/> DHCP and DNS <input checked="" type="checkbox"/> Firewall </div>

-- Additional Field --

Add

Back to Overview

Save & Apply

Save

Reset

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 message only unconfigured levels are shown in a dropdown list when configuring. Other fields are dedicated to an individual menu configuration. To

add more firstlevel menus user should select from a dropdown list at the bottom named –Additional Field– and press Add.

Permissions for the web interface are split into two parts: View and Edit.

View permissions can be assigned to second-level menus meaning that only allowed subtabs are shown for a user. Selecting the View checkbox shows 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 successfully

Conformance to IEC 62351 standard

IEC 62351 is a standard developed by WG15 of IEC TC57. This is developed for handling the security of the TC 57 series of protocols including the 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 the IEC 62351 standard of WCC Lite+ devices is described in the 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 using 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 a 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 mandatory for GOOSE	No	
		RFC 2030 to be used for SNTP	No	
IEC 62351-7	Security through network and system management	Defines Management Information Bases (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 using role based access control (RBAC)	Yes	>=1.2.6
IEC 62351-9	Key Management	Describes the correct and safe usage of safety-critical parameters, e.g. passwords, encryption keys.	No	
		Covers the whole life cycle of cryptographic information (enrolment, creation, distribution, installation, usage, storage, and removal)	No	
		Methods for algorithms using asymmetric cryptography	No	
		A secure distribution mechanism based on GDOI and the IKEv2 protocol is presented for the usage of symmetric keys, e.g. session keys	No	
IEC 62351-10	Security Architecture	Explanation of security architectures for the entire IT infrastructure	No	
		Identifying critical points of the communication architecture, e.g. substation control center, substation automation	No	
		Appropriate mechanisms security requirements, e.g. data encryption, user authentication	No	
		Applicability of 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	

12 DNP 3.0


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 can 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 13 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 the protocol specification and/or check the 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 DNP 3.0

12.2 DNP 3.0 Master

Default groups and variations 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	
			TCP/ TLS	Serial		Min	Max
name	string	User-friendly device name	Yes	Yes			
description	string	Description of a device	No	No			

device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
protocol	string	Protocol to be used ("dnp3 serial"/"dnp3 tcp" (case insensitive))	Yes	Yes		DNP3 TCP, DNP3 serial	
mode	string	Choosing between TCP, TLS, and SERIAL modes. If the protocol is provided DNP3 TCP mode defaults to tcp and if DNP3 serial is provided mode defaults to serial	No	No	TCP (for DNP3 TCP) SERIAL (for DNP3 serial)	TCP, TLS (for DNP3 TCP) SERIAL (for DNP3 serial)	
host	string	The IP address of the TCP slave device	Yes	-			
bind_address	string	The IP address of the network adapter used to connect to the slave device	No	No	0.0.0.0		
port	integer	TCP communication port	No	No	20000		
device	integer	Communication port ("PORT1" or "PORT2")	-	Yes			
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	
tls	boolean	Enable/disable the 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	A file consisting of the private key for TLS connection	No (for TLS)	-			
max_rx_frag_size	integer	Maximum size of a received fragment.	No	No	2048	0	2048
destination_address	integer	Address of a master station	No	No	1	0	65535
source_address	integer	Address of a slave (local) station.	No	No	1	0	65535
unsol_classes	string	Defines which classes will have unsolicited actions on startup. (Example: "1,3,2")	No	No	no class	1	3
unsol_disable	bool	Disables unsolicited messages on startup. The parameter is going to be ignored if the unsol_classes parameter has any values assigned.	No	No	0	0	1
groups_scan_mask	integer	<p>Bitmask for enabling separate group scans with x06 qualifier (all objects). The parameter value is converted into a binary number where each bit stands for a separate group. Bits indexes and the groups that they represent:</p> <p>0 - Binary, 1 - Doublebit Binary, 2 - Binary Output Status, 3 - Counter, 4 - Frozen Counter, 5 - Analog, 6 - Analog Output Status, 7 - Octet String</p> <p>(Example: 115 (0111 0011) will trigger data polls for signals whose types are - Binary, Double bit Binary, Frozen Counter, Analog, Analog Output Status)</p>	No	No	0	0	255
groups_scan_interval	integer, string	The time between separate groups scans intervals in seconds. Set to 0 to disable.	No	No	0	0	

exception_scan_interval	integer, string	The time between exception scan (classes 1,2,3) intervals in seconds. Set to 0 to disable.	No	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	No	0	0	
timesync_mode	string	Will override the master default setting for choosing the time sync procedure	No	No	NON-LAN (for Serial) LAN (for tcp)	LAN, NON-LAN	
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 the slave. If < 0 - time syncs are disabled.	No	No			
select_ms	integer	Select command timeout. Valid for all signals.	No	No	10000		
timeout_ms	integer	Response timeout in milliseconds	No	No	2000		
keep_alive_timeout	integer	The time interval for sending a keep-alive packet in milliseconds.	No	-	60		

DNP3 Master parameters for the Signals tab

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

index	integer	Index of a signal.	Yes	Yes		0	65535
log	boolean	Enable logging in the event log	No	No	0	0	1
signal_type	string	DNP3 signal type. (case insensitive)	Yes	Yes		"binary", "doublebitbinary", "binaryoutputstatus", "binaryoutputcommand", "counter", "frozenscounter", "analog", "analogoutputstatus", "analogoutputcommand", "timeandinterval", "octetstring"	
command_variation	integer	DNP3 command variation. Supported variations depend on signal type and are provided in the table below	No	No	1	0	4
static_variation	integer	DNP3 command variation (). Supported variations depend on signal type and are provided in the table below.	No	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	No		0	8
control_code	string	DNP3 control model code of CROB signal. TripClose and Pulse control model requires PulseOn/off times to be set	Yes	Yes		LATCH, PULSE, TRIPCLOSE	
pulse_on_time_ms	integer	Pulse ON time in milliseconds, when using Pulse or TripClose control models must be set	Yes	Yes			
pulse_off_time_ms	integer	Pulse OFF time in milliseconds, when using Pulse or TripClose control models must be set	Yes	Yes			
class_num	integer	Class assignment of the signal.	No	No	0	0	3

operate_type	integer	Default command behavior. IF selected "-1" - DirectOperateNoAck (FC=6), "0" - DirectOperate (FC=5), "1" - SelectBeforeOperate (FC=3).	No	No	1	-1	1
job_todo	string	The device status signal can be configured by providing one of the given values.	No	No		communication_status, device_running, device_error, unknown_error	

Device status signals

To configure any device status signal for DNP3 protocol additional job_todo column is required. For DNP3 master required parameters for status signal will be: **signal_name**, **device_alias**, **signal_alias**, **index**, **signal_type**, **event_variation** (1,2 or 3) and **job_todo**. There are 4 possible signals: communication_status, device_running, device_error, unknown_error. Each signal has 4 possible values and is based on the same logic. If the signal returns the value of 0, it means an unknown error has appeared, 1 – the device or protocol connection is on and working properly, 2 – the device is off or protocol is disconnected, and 3 – the error or service is down.

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 Event variation (Group2) 1, 2, 3	Static variation 2 Event variation 1
Double Binary	Static variation (Group3) 2 Event variation (Group4) 1, 2, 3	Static variation 2 Event variation 1
Binary Output Status	Static variation (Group10) 1, 2 Event variation (Group11) 1, 2	Static variation 2 Event variation 1
Counter	Static variation (Group20) 1, 2, 5, 6 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 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 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 Event variation (Group111) 0	Static variation 0 Event variation 0

Debugging the DNP3 Master service

If the configuration for DNP3 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.

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 the command line interface and find out why the link is not functioning properly. To launch a debugging session, a user should stop the **dnp3-master** process and run the **dnp3-master** 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 wcc-lite: **/etc/init.d/dnp3-master stop**
- **Step 2:** After the service is stopped it must be started with the preferred configuration file (JSON files found in the/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 DNP 3.0

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	
			TCP/ TLS	RTU		Min	Max
name	string	User-friendly device name	Yes	Yes			

description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
protocol	string	Protocol to be used.	Yes	Yes		dnp3 tcp slave dnp3 serial slave	
mode	string	Choosing between TCP, TLS, and SERIAL modes. If the protocol is provided DNP3 TCP mode defaults to tcp and if DNP3 serial is provided mode defaults to SERIAL	No	No	TCP or SERIAL	TCP, SERIAL, TLS	
host	string	The IP address of the TCP slave device	Yes	-			
bind_address	string	The IP address of the network adapter used to connect to the slave device	No	-	0.0.0.0		
port	integer	TCP communication port	No	-	20000		
device	string	Communication port ("PORT1" or "PORT2")	-	Yes			
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	
tls	boolean	Enable/disable the 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	A file consisting of the private key for TLS connection	No (for TLS)	-			
max_tx_frag_size	integer	Maximum size of a received fragment.	No	No	2048	0	2048
destination_address	integer	Address of a master station	No	No	1	0	65535
source_address	integer	Address of a slave (local) station.	No	No	1	0	65535
unsol_classes	string	Defines which classes will have unsolicited actions on startup. (Example: "1,3,2")	No	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 the slave. If < 0 - time syncs are disabled.	No	No	0	0	
select_ms	integer	Select command timeout. Valid for all signals.	No	No	10000		
timeout_ms	integer	Response timeout in milliseconds	No	No	2000		
keep_alive_timeout	integer	The time interval for sending a keep-alive packet in milliseconds.	No	No	60		

DNP3 Slave parameters for Signals tab

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

signal_alias	string	Unique alphanumeric name of the signal to be used	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
index	integer	Index of a signal.	Yes	Yes		0	65535
log	boolean	Enable logging in the event log	No	No	0	0	
deadband	integer, string	Deadband for Analog, Analog Output Status, Counter, and Frozen Counter signals.	No	No	0		
signal_type	string	DNP3 signal type. (case insensitive)	Yes	Yes		"binary", "doublebitbinary", "binaryoutputstatus", "binaryoutputcommand", "counter", "frozenscounter", "analog", "analogoutputstatus", "analogoutputcommand", "timeandinterval", "octetstring"	
command_variation	integer	DNP3 command variation. Supported variations depend on signal type and are provided in the table below	No	No	1	0	4
static_variation	integer	Override default signal's static variation. Valid for Status mode signals.	No	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	No		0	8
control_code	string	DNP3 control model code of CROB signal. TripClose and Pulse control model requires PulseOn/off times to be set	Yes	Yes		LATCH, PULSE, TRIPCLOSE	
pulse_on_time_ms	integer	Pulse ON time in milliseconds, when using Pulse or TripClose control models must be set	Yes	Yes			
pulse_off_time_ms	integer	Pulse OFF time in milliseconds, when using Pulse or TripClose control models must be set	Yes	Yes			
class_num	integer	Class assignment of this signal.	No	No	0	0	3
operate_type	integer	Default command behavior. If selected: "-1" - DirectOperateNoAck (FC=6), "0" - DirectOperate (FC=5), "1" - SelectBeforeOperate (FC=3).	No	No	1	-1	1
job_todo	string	The device status signal can be configured by providing one of the given values.	No	No		communication_status, device_running, device_error, unknown_error	

Device status signals

To configure any device status signal for DNP3 protocol additional job_todo column is required. For DNP3 slave required parameters for status signal will be: **signal_name**, **device_alias**, **signal_alias**, **index**, **signal_type**, **event_variation** (1,2 or 3) and **job_todo**. There are 4 possible signals: communication_status, device_running, device_error, unknown_error. Each signal has 4 possible values and is based on the same logic. If the signal returns the value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

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 Event variation (Group2) 1, 2, 3	Static variation 2 Event variation 1
Double Binary	Static variation (Group3) 2 Event variation (Group4) 1, 2, 3	Static variation 2 Event variation 1
Binary Output Status	Static variation (Group10) 2 Event variation (Group11) 1, 2	Static variation 2 Event variation 1
Counter	Static variation (Group20) 1, 2, 5, 6 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 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 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 Event variation (Group111) 0	Static variation 0 Event variation 0

Debugging the DNP3 Slave service

If the configuration for DNP3 devices 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.

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 the command line interface and find out why the link is not functioning properly. To launch a debugging session, a user should stop the **dnp3-slave** process and run the **dnp3-slave** 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 wclite: **/etc/init.d/dnp3-slave stop**
- **Step 2:** After the service is stopped it must be started with the preferred configuration file (JSON files found in the /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 Modbus

13 Modbus

13.1 Introduction

Modbus is a communication 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 be encoded according to both RTU and ASCII parts of the standard.

13 Modbus

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, the master device requests a cluster of slave devices for data. If any device understands that this message is addressed to it – 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 the Devices tab

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU/ASCII		Min	Max
name	string	User-friendly name for a device	Yes	Yes			
description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
protocol	string	Protocol to be used	Yes	Yes		Modbus RTU, Modbus TCP	
ip	string	The IP address of the TCP slave device	Yes	-			
port	integer	TCP communication port	Yes	-	502		
bind_address	string	The IP address of the network adapter used to connect to the slave device (Default: "0.0.0.0")	No	No	0.0.0.0		
id	integer	Modbus Slave ID	Yes	Yes			
mode	string	Choosing between RTU ("rtu"), ASCII ("ascii") and TCP("tcp") modes. ASCII is the same as RTU, but with ASCII symbols.	No	No	TCP (for TCP) RTU (for Serial)	rtu, ascii, tcp	
timeout_ms	integer	Response timeout in milliseconds	Yes	Yes	10000		
device	string	Communication port ("PORT1"/"PORT2")	-	Yes		PORT1	PORT2
baudrate	integer	Communication speed, baud/s	-	Yes	9600	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	

databits	integer	Data bit count for communication	-	Yes	8	6	9
stopbits	integer	Stop bit count for communication	-	Yes	1	1	2
parity	string	Communication parity option	-	Yes	none	none, even, odd	
flowcontrol	string	Number of requests, before the link is considered lost (device status signals are changed) and reconnect attempt will be issued	-	Yes	none	none	
scan_rate_ms	integer	If provided and positive - all jobs will have similar scan rates - all reads and writes will be executed within this timeframe (parameter scan_rate_ms in the Signals tab will be ignored)	Yes	Yes	300		
retry_count	integer	Number of requests, before the link is considered lost (device status signals are changed) and reconnect attempt will be issued	No	No	3		
serial_delay	integer	RS485 delay between read and write operations in milliseconds	-	Yes	50		
keep_alive_timeout	integer	Time interval for sending a keepalive packet (in milliseconds)	No	No	60		
modbus_multi_write	boolean	Use 15/16 functions to write 1 register/coil (Default: 0)	No	No	0	0	1
comm_restart_delay	integer	Time delay between disconnecting from the slave device and restarting the connection (in milliseconds) (Default: 500)	No	-	500		
update	boolean	Enable to keep updating the tags even if they have the same value.	No	No	0	0	1

Modbus Master parameters for the Signals tab

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU/A SCII		Min	Max

signal_name	string	User-friendly signal name	Yes	Yes			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes	Yes			
enable	boolean	Enabling/disabling of an individual signal	No	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 a single request	Yes	Yes			
tag_job_todo	string	Similar format to the job_todo field. Address and length must be a subset of the job field. Defines the individual tag's register(s) or coil(s). Can be described in HEX or DEC formats	Yes	Yes			
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes	Yes			
log	integer	Size of this signal's log in the Event log.	No	No	0		
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	No	0		

Device status signals

Modbus Master has an additional signal that 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 signal for Modbus protocol, **job_todo** and **tag_job_todo** fields with string values are required. For Modbus master required parameters for the status signal will be **signal_name**, **device_alias**, **signal_alias**, **number_type**, **job_todo**, and **tag_job_todo**. **job_todo** value must be **device_status** and for **tag_job_todo** there are 4 variations: **communication_status**, **device_running**, **device_error**, **unknown_error**. Each signal has 4 possible values and is based on the same logic. If the signal returns a value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

Different device vendors can have different implementations of a Modbus protocol stack. A register table can be 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 two different formats - the user can select a more convenient way for him:

- hexadecimal format with every single byte separated by a | 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.

Debugging a Modbus Master application

If the configuration for Modbus Master 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.

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 the command line interface and find out why the link is not functioning properly. To launch a debugging session, a user should stop the modbus-master process and run the modbus-master command with respective flags as shown above.


13 Modbus

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 messages from Modbus Master with matching devices 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 transmission medium, only devices with IPs predefined in the 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	
			TCP	RTU/A SCII		Min	Max
name	string	User-friendly name for a device	Yes	Yes			
description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes	unknown		
enable	boolean	Enabling/disabling of a device	No	No	1	0	1

protocol	string	Protocol to be used	Yes	Yes		Modbus serial Slave, Modbus TCP Slave	
host	string	Space-separated host IP addresses of master device	Yes	-			
port	integer	TCP port to listen for incoming connections	Yes	-			
bind_address	string	The IP address of the network adapter used to connect to the slave device (Default: "0.0.0.0")	No	No	0.0.0.0		
keep_alive_timeout	integer	The minimum time a connection can be idle without being closed in milliseconds	No	No	60		
mode	string	Choosing between RTU ("rtu"), ASCII ("ascii") and TCP("tcp") modes. ASCII is the same as RTU, but with ASCII symbols.	No	No	TCP (for TCP) RTU (for Serial)	rtu, ascii, tcp	
device	string	Communication port ("PORT1"/"PORT2")	-	Yes		PORT1	PORT2
baudrate	integer	Communication speed, baud/s	-	Yes	9600	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	
databits	integer	Data bit count for communication	-	Yes	8	6	9
stopbits	integer	Stop bit count for communication	-	Yes	1	1	2
parity	string	Communication parity option	-	Yes	none	none, even, odd	
flowcontrol	string	The communication device's flow control option.	-	No	none	none	

Modbus Slave parameters for the Signals tab

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU/A SCII		Min	Max
signal_name	string	User-friendly signal name	Yes	Yes			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes	Yes			
enable	boolean	Enabling/disabling an individual signal	No	No	1	0	1

number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.). This defines the size that will be read.	Yes	Yes			
log	integer	Size of this signal's log in the Event log.	No	No	0		
slave_id	integer	Address of a slave device	Yes	Yes			
function	integer	Function number	Yes	Yes			
register_addresses	integer	Register address	Yes	Yes			


Device status signals

Modbus slave has an additional signal which can be configured to show communication status. It is used to indicate if the master device has disconnected from the slave (WCC Lite+). To configure such signal for Modbus protocol, `job_todo` and `tag_job_todo` fields with string values are required. For Modbus slave required parameters for status, the signal will be **signal_name**, **device_alias**, **signal_alias**, **number_type**, **slave_id**, **function**, **register_address**, **job_todo**, and **tag_job_todo**. `job_todo` value must be `device_status` and for `tag_job_todo` there are 4 variations: `communication_status`, `device_running`, `device_error`, `unknown_error`. Each signal has 4 possible values and is based on the same logic. If the signal returns a value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

Mapping values to registers

Internally stored values aren't organized in a register-like order, therefore mapping should be done by the user. This mapping includes setting the address of the device WCC Lite+ is simulating as well as the function number, register number, and how many 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 registers should be used for storing values, and how values can have their values swapped, a user should consult a section [number_type](#).

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


Debugging a Modbus Slave application


If the configuration for Modbus Slave 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.

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 the command line interface and find out why a link is not functioning properly.

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

14 IEC 60870-5-10X

14 IEC 60870-5-10X

14.1 Introduction

IEC 60870 part 5 is one of the IEC 60870 standards that 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, the transmission of integrated totals, data exchange from protection equipment & network access of IEC101 respectively.

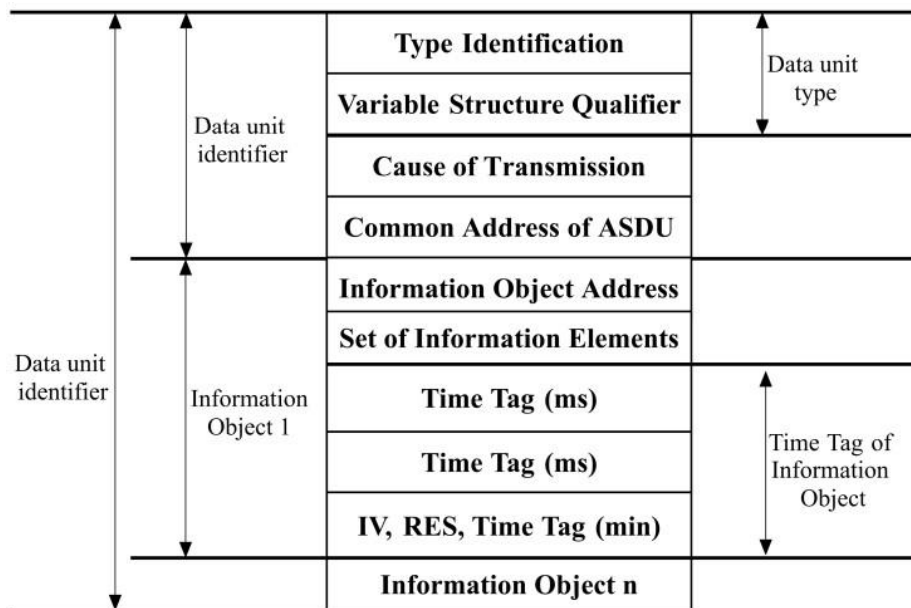
Source: https://en.wikipedia.org/wiki/IEC_60870-5

14 IEC 60870-5-10X

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. 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). ASDU there is an ASDU identifier (with the type of ASDU in it) and information objects.

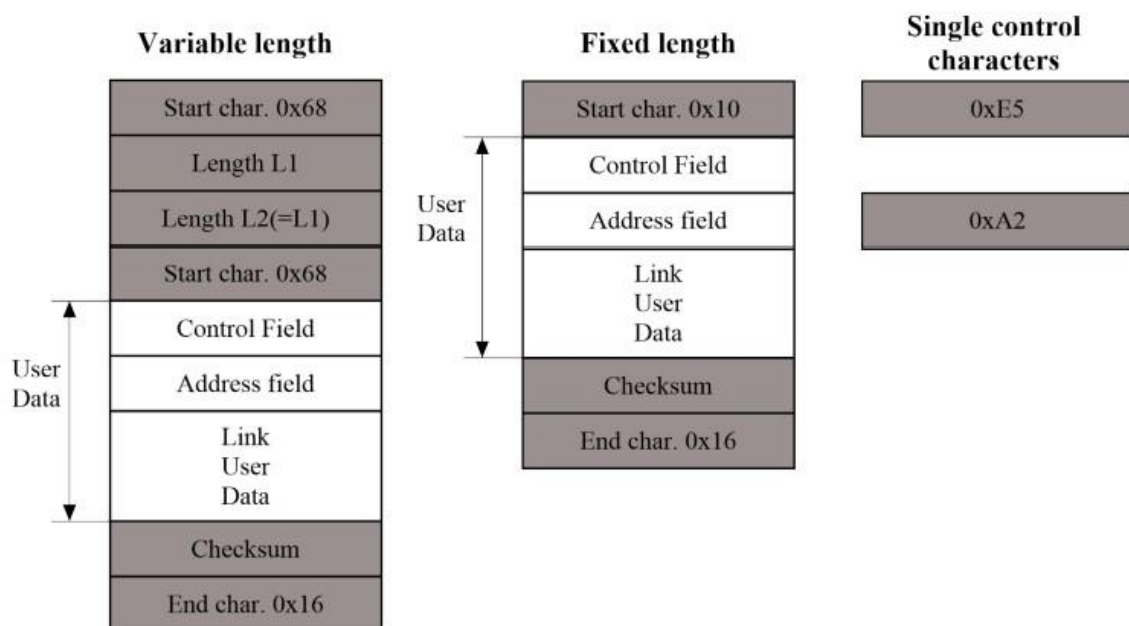


IEC 60870-5-101 ASDU structure

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 the monitor direction can be configured in the 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**



IEC 60870-5-101 ASDU structure

The control field provides information about the message direction, type of service, and checksum.

The 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 60870-5-101 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 (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	Number of requests, before the 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_address	integer	Application Service Data Unit address	Yes		0	65535
asdu_size	integer	Common address size in bytes	No	1	1	2
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_interval_sec	integer	Defines how often (in seconds) a slave will request time synchronization. If greater than 0 slave will request synchronizations and will reset the timer if the master did it earlier. If 0 slave won't request timesyncs, but will allow them. If 1 time syncs are not supported requests will be dropped.	No	60		
gi_interval_sec	integer	The time frame between General Interrogation requests in seconds if 0 requests are disabled	No	300		
scan_rate_ms	integer	Polling interval in milliseconds. The time frame between two telegrams from the master	Yes	100		
timeout_ms	integer	Response timeout in milliseconds	Yes	1000		
retry_count	integer	Number of retries of failed requests before announcing that the device is in Error state	Yes	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 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) signals from General Interrogation	No	0	0	1
common_address	integer	Address of a destination device	Yes	1		
info_address	integer	Information object address	Yes			
data_type	integer	ASDU type identifier	Yes		1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 30, 31, 32, 34, 35, 36, 45, 46, 47, 48, 49, 50	

Device status signals

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 the master (WCC Lite+). To configure such signal for IEC 60870-5-101 protocol, **job_todo** and **tag_job_todo** fields with string values are required. For IEC 60870-5-101 master required parameters for a status signal will be: **signal_name**, **device_alias**, **signal_alias**, **common_address**, **info_address**, **data_type**, **job_todo**, and **tag_job_todo**. **job_todo** value must be **device_status** and for **tag_job_todo** there are 4 variations: **communication_status**, **device_running**, **device_error**, **unknown_error**. Each signal has 4 possible values and is based on the same logic. If the signal returns a value of 0, it means an unknown error has appeared, 1 – the device or protocol connection is on and working properly, 2 – the device is off or protocol is disconnected, and 3 – the error or service is down.

Debugging an IEC 60870-5-101 Master application

If the configuration for IEC 60870-5-101 devices is set up, a handler for the protocol will start automatically. If the configuration is missing parameters 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 a link is not functioning properly. To launch a debugging session, a user should stop the iec101-master process and run the 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 WCC Lite+: **/etc/init.d/iec101-master stop**
- **Step 2:** After the 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 are 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 IEC 60870-5-10X

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 (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	Number of requests, before the 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	2
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	253	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	100	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 the connection tracker to display protocol status. This parameter does not affect protocol functionality and is only used to track its status in the 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 names	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_aliases	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) signals 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 identifier	Yes		1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 30, 31, 32, 34, 35, 36, 45, 46, 47, 48, 49, 50, 58, 59, 60, 61, 62, 63
-----------	---------	----------------------	-----	--	---

Device status signals

IEC 60870-5-101 has an additional signal which can be configured to show communication status. It is used to indicate if the master device has disconnected from the slave (WCC Lite+). To configure such signal for IEC 60870-5-101 protocol, `job_todo` and `tag_job_todo` fields with string values are required. For IEC 60870-5-101 slave required parameters for status, signal will be **signal_name device_alias, signal_alias, common_address, info_address, data_type, job_todo** and **tag_job_todo**. `job_todo` value must be `device_status` and for `tag_job_todo` there are 4 variations: `communication_status`, `device_running`, `device_error`, `unknown_error`. Each signal has 4 possible values and is based on the same logic. If the signal returns the value of 0, it means an unknown error has appeared, 1 – the device or protocol connection is on and working properly, 2 – the device is off or the protocol is disconnected, and 3 – the error or service is down.

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 parameters 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 WCC Lite+: **`/etc/init.d/iec101-slave stop`**
- **Step 2:** After the 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 are 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 IEC 60870-5-10X

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 (bauds/s)	No	9600	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 the 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	The time frame between Time Synchronization requests in seconds	No	60		
gi_interval_sec	integer	The time frame between General Interrogation requests in seconds if 0 requests are disabled	Yes	300		
scan_rate_ms	integer	Polling interval in milliseconds. The time frame between two telegrams from the 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 the master station is not TX'ing after the last RX byte	No	50		
retry_count	integer	Number of retries of failed requests before announcing that the 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_alias	string	device_alias of a source device	For commands			
source_signal_aliases	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) signals 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 identifier	No	0	0, 1, 2, 3, 4, 9, 20	
fleeting	boolean	Mark the signal as a fleeting type (1 or 0). Fleeting signals have to go to DPI::OFF after a defined time	No		0	1
normalise_time_ms	integer	Time in milliseconds between station receiving DPI::ON and automatically switching to DPI::OFF	If fleeting is used	100		

Device status signals

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 signal for IEC 60870-5-103 protocol, job_todo and tag_job_todo fields with string values are required. For IEC 60870-5-103 master required parameters for a status signal will be: **signal_name device_alias, signal_alias, common_address, info_address, info_number, job_todo, and tag_job_todo**. Job_todo value must be device_status and for tag_job_todo there are 4 variations: communication_status, device_running, device_error, unknown_error. Each signal has 4 possible values and is based on the same logic. If the signal returns a value of 0, it means an unknown error has appeared, 1 - the device or protocol connection is on and working properly, 2 - the device is off or protocol is disconnected, and 3 - the error or service is down.

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 parameters 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 WCC Lite+: **/etc/init.d/iec103-master stop**
- Step 2: After the 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 IEC 60870-5-10X

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 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 data points

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	2
time_sync_interval_sec	integer	The time frame between Time Synchronization requests in seconds	Yes	60		
gi_interval_sec	integer	The time frame between General Interrogation requests in seconds. If 0 requests are disabled	Yes	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 (k)	Yes			
rwt	integer	Receive window (w)	Yes			
cot_size	integer	Cause of transmission (COT) size in bytes	No	2	1	3
ip	string	Host IP address (ipv4)	Yes			
t1*	integer	Acknowledge timeout t1 (sec)	Yes	15	1	255
t2*	integer	Connection ACKRSN clock t2 (sec)	Yes	10	1	254
t3*	integer	Connection TESTFR clock t3 (sec)	Yes	20	1	172800

originator	integer	Provides a means for a controlling station to explicitly identify itself	No	0	0	255
------------	---------	--	----	---	---	-----

* - t1, t2 and t3 parameters must meet the inequality: $t2 < t1 < t3$.

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_alias	string	device_alias of a source device	For commands			
source_signal_aliases	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) signals 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 identifier	Yes		1, 3, 5, 9, 11, 13, 21, 30, 31, 32, 34, 35, 36, 45, 46, 47, 48, 49, 50, 58, 59, 60, 61, 62, 63	

select_ms	integer	Time limit in milliseconds for command execution. Command selection 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		
-----------	---------	---	----	---	--	--

Device status signals

IEC 60870-5-104 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 signal for IEC 60870-5-104 protocol, `job_todo` and `tag_job_todo` fields with string values are required. For IEC 60870-5-104 master required parameters for a status signal will be: **signal_name device_alias, signal_alias, common_address, info_address, data_type, job_todo, and tag_job_todo**. `job_todo` value must be `device_status` and for `tag_job_todo` there are 4 variations: `communication_status`, `device_running`, `device_error`, `unknown_error`. Each signal has 4 possible values and is based on the same logic. If the signal returns a value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

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 parameters 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 WCC Lite+: **`/etc/init.d/iec104-master stop`**
- Step 2: After the 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 IEC 60870-5-10X

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 data points

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	3
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)	Yes	15	1	255

t2	integer	Connection ACKRSN clock t2 (sec), t2 should be less than t1	Yes	10	1	254
t3	integer	Connection TESTFR clock t3 (sec)	Yes	20	1	172800
message_size	boolean	The maximum length of a message	Yes	253	0	255
cache_size	integer	Amount of data to be cached	Yes	100	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_certificate	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_alias	string	device_alias of a source device	For commands			
source_signal_aliases	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/excluding (1 or 0) signals 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		1, 3, 5, 9, 11, 13, 21, 30, 31, 32, 34, 35, 36, 45, 46, 47, 48, 49, 50, 58, 59, 60, 61, 62, 63	
select_ms	integer	Time limit in milliseconds for command execution. Command selection 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		

Device status signals

IEC 60870-5-104 has an additional signal which can be configured to show communication status. It is used to indicate if the master device has disconnected from the slave (WCC Lite+). To configure such signal for IEC 60870-5-104 protocol, `job_todo` and `tag_job_todo` fields with string values are required. For IEC 60870-5-104 slave required parameters for status, the signal will be **signal_name device_alias, signal_alias, common_address, info_address, data_type, job_todo**, and **tag_job_todo**. `job_todo` value must be `device_status` and for `tag_job_todo` there are 4 variations: `communication_status`, `device_running`, `device_error`, `unknown_error`. Each signal has 4 possible values and is based on the same logic. If the signal returns a value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

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 parameters 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 WCC Lite+: **`/etc/init.d/iec104-slave stop`**
- Step 2: After the 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 60870-5-104 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
```

15 IEC 61850

15 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 several protocols. Possible mappings in the standard can be MMS (Manufacturing Message Specification), GOOSE (Generic Object Oriented Substation Event), and 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 group setting services are not supported.

15.2 IEC 61850 Server

WCC Lite can act as an 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 the master protocol to either succeed or fail. If feedback is not received within the **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 the master protocol, and select logic is performed only in the IEC 61850 Server protocol.

Configuring data points

To use the IEC 61850 Server in WCC Lite, it has to be configured via an Excel configuration, and the data model must be uploaded. This configuration contains two Excel sheets where parameters have to be filled 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	
bind_address	string (IP address format)	The IP address of an interface to use with the 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	No	102		
auth	string	Authorization type	Yes		"NONE", "PASSWORD", "TLS"	
password	string	Authorization password for server device	Yes (for PASSWORD)			
tls_local_certificate	string	Local certificate for TLS connection	Yes (for TLS)			
tls_peer_certificate	string	Certificate authority file for TLS connection	Yes (for TLS)			
tls_private_key	string	A file consisting of the private key for TLS connection	Yes (for TLS)			
ied_name	string	Name of an Intelligent Electronic Device	Yes			
originator	string	Origin identification for the device	No			
model_filename	string	The filename of the server model, without the .server extension	Yes			
command_ack_timeout_ms	integer	Timeframe (ms) in which enhanced security commands must be acknowledged (Default: 3000)	No	3000		
report_buffered_size	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		BOOLEAN, INT8, INT16, INT32, INT64, INT128, INT8U, INT24U, INT32U, FLOAT32, FLOAT64, ENUMERATED, OCTET STRING 64, OCTET STRING 6, OCTET STRING 8, VISIBLE STRING 32, VISIBLE STRING 64, VISIBLE STRING 65, VISIBLE STRING 129, VISIBLE STRING 255, UNICODE STRING 255, TIMESTAMP, QUALITY, CHECK, CODEDENUM, GENERIC BITSTRING, CONSTRUCTED, ENTRY TIME, PHYCOMADDR, CURRENCY, OPTFLDS, TRGOPS	
Id_instance	string	An instance of a logical device	Yes			
In_class	string	Logical node class type	Yes			
In_instance	integer	An instance of a logical node	No			
In_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 a data object in the 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	Yes (for commands)	read-only	read-only, direct-with-normal-security, sbo-with-normal-security, direct-with-enhanced-security, sbo-withenhanced-security
---------------	--------	-------------------------	--------------------	-----------	--

Device status signals

IEC 61850 has an additional signal that can be configured to show communication status. It is used to indicate if the client device has disconnected from the server (WCC Lite). To configure such a signal for the IEC 61850 protocol, job_todo and tag_job_todo fields with string values are required. For the IEC 61850 server required parameters for the status signal will be **signal_name device_alias, signal_alias, number_type, job_todo, and tag_job_todo**. Job_todo value must be device_status and for tag_job_todo there are 4 variations: communication_status, device_running, device_error, unknown_error. Each signal has 4 possible values and is based on the same logic. If the signal returns the value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

Converting and uploading data model

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



The converted file can be uploaded in the WCC+ Lite web interface, Protocol Hub section. The current model can be also downloaded on the same page as shown in the picture below.

PROTOCOL HUB	STATUS	SYSTEM	SERVICES	NETWORK	USERS	LOGOUT (ROOT)
CONFIGURATION	IMPORTED SIGNALS	EVENT LOG	PROTOCOL CONNECTIONS	SCRIPT-RUNNER		

Protocol configuration

IMPORT PROTOCOL CONFIGURATION

Here you can import Excel configuration file. Up to 1000 signals are allowed. All previous signals will be replaced.

Configuration file: No file chosen

PLC (IEC-61499) Boot file: No file chosen

IEC61850 Client model file: No file chosen

IEC61850 Server model file: No file chosen

DOWNLOAD CONFIGURATION


Current configuration (WCC.xlsx):


Template configurations:

Current IEC 61850 Server model file (WCC.server):

Debugging an IEC 61850 server application

If the configuration for the 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 the 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 the command line interface and find out why the link is not functioning properly.

 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 IEC 61850

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 the 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 bandwidth. 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. The 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 behavior:

- 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 the 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 confusion and even financial losses. Unbuffered report control blocks are particularly useful for data that 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 objects, 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 Classes (CDC). Some of these types can have more than one data point of interest. The table below shows what data attributes are supported by 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 the correct value in da_fc column.

IEC 61850 Client-supported data attributes:

Common Data Class	Function Constraint	Data attributes
SPS DPS INS ENS	ST	stVal
ACT	ST	general phsA phsB phsC neut
ACD	ST	general dirGeneral phsA dirPhsA phsB dirPhsB phsC dirPhsC neut dirNeut
MV	MX	instMag mag
CMV	MX	instCVal cVal
SAV	MX	instMag
SPC DPC INC ENC	ST	stVal

BSC ISC	ST	valWTr
APC BAC	MX	mxVal

Some of the 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 the 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.

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) whose `ctlModel` DataAttribute is not set to status - only. Such data objects can be referred to as control objects. If controls are enabled in an IEC 61850 Server device the user can configure controls by filling the `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 behaviors 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 - and 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 v1.8 All issued commands are executed immediately.

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

If `ctlModel` is read-only, messages from the 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 the CDC of a signal does not have a means of control, the `ctlModel` parameter is ignored.

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

Configuring datapoints

To use the 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	
host	string (IP address format)	The IP address of the server device	Yes			
port	integer	TCP communication port	Yes	102		
auth	string	Authorization type	Yes		none, password, tls	
password	string	Authorization password for server device	Yes (for PASSWORD)			
tls_local_certificate	string	Local certificate for TLS connection	Yes (for TLS)			
tls_peer_certificate	string	Certificate authority file for TLS connection	Yes (for TLS)			
tls_private_key	string	A file consisting of the private key for TLS connection	Yes (for TLS)			
ied_name	string	Name of an Intelligent Electronic Device	Yes			
originator	string	Origin identifier for the device	No			
model_filename	string	The filename of the client model uploaded to WCC (must contain .client extension)	Yes			

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		BOOLEAN, INT8, INT16, INT32, INT64, INT128, INT8U, INT24U, INT32U, FLOAT32, FLOAT64, ENUMERATED, OCTETSTRING6, OCTETSTRING8, OCTETSTRING64, VISIBLESTRING32, VISIBLESTRING64, VISIBLESTRING65, VISIBLESTRING129, VISIBLESTRING255, UNICODESTRING255, TIMESTAMP, QUALITY, CHECK, CODEENUM, GENERICBITSTRING, CONSTRUCTED, ENTRYTIME, PHYCOMADDR, CURRENCY, OPTFLDS, TRGOPS	
ld_instance	string	An instance of a logical device	Yes			
ln_class	string	Logical node class type	Yes			
ln_instance	integer	An instance of a logical node	No			
ln_prefix	string, integer	Prefix of logical node string	No			
cdc	string	Common Data Class (CDC) name	Yes		SPS, DPS, INS, ENS, ACT, ACD, SEC, BCR, HST, VSS, MV, CMV, SAV, WYE, DEL, SEQ, HMV, HWYE, HDEL, SPC, DPC, INC, ENC, BSC, ISC, APC, BAC, SPG, ING, ENG, ORG, TSG, CUG, VSG, ASG, CURVE, CSG, DPL, LPL, CSD, UNDEF	
data_object	string	Name of a data object in the 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, SE	
control_model	string	Model of output control	No	read-only	read-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_block	string	Full object reference of a report control block	Yes			
intgPd	integer	Integrity period in milliseconds	No	0		

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

Device status signals

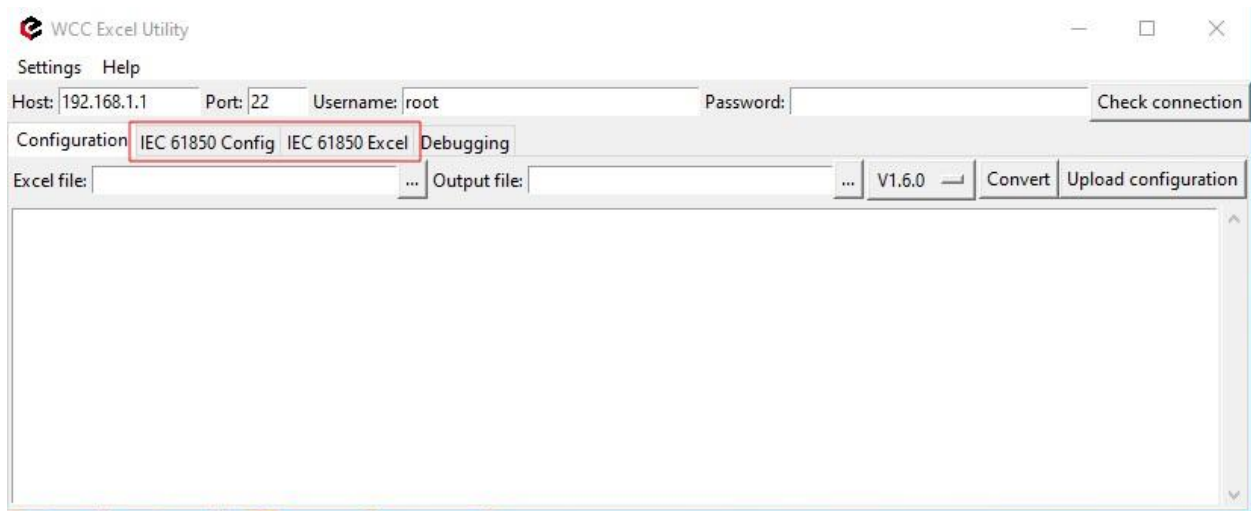
IEC 61850 has an additional signal which can be configured to show communication status. It is used to indicate if the server device has disconnected from the client (WCC Lite). To configure such a signal for the IEC 61850 protocol, job_todo and tag_job_todo fields with string values are required. For IEC 61850 client required parameters for the status signal will be **signal_name**, **device_alias**, **signal_alias**, **number_type**, **job_todo**, and **tag_job_todo**. Job_todo value must be device_status and for tag_job_todo there are 4 variations: communication_status, device_running, device_error, unknown_error. Each signal has 4 possible values and is based on the same logic. If the

signal returns the value of 0, it means an unknown error has appeared, 1 – device or protocol connection is on and working properly, 2 – device is off or protocol is disconnected, 3 – error or service is down.

Configuration

Configuration of IEC61850 Client for WCC Lite is done via WCC Utility. Elseta WCC Utility has two IEC 61850 selections IEC 61850 Config and IEC 61850 Excel:

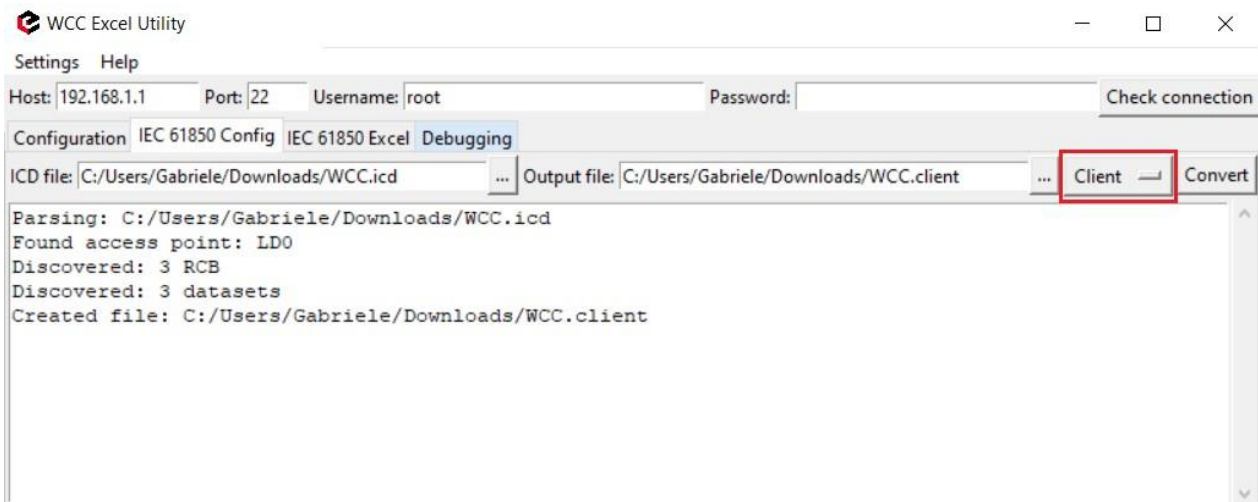
- **IEC 61850 Config** is used to create a configuration model file, which IEC 61850 Client service will use to parse reports from the server.
- **IEC 61850 Excel** is used to generate Excel configuration files which in turn will be used to generate configuration .json via excel-utility.



WCC Utility with IEC61850 selections

Generate model file

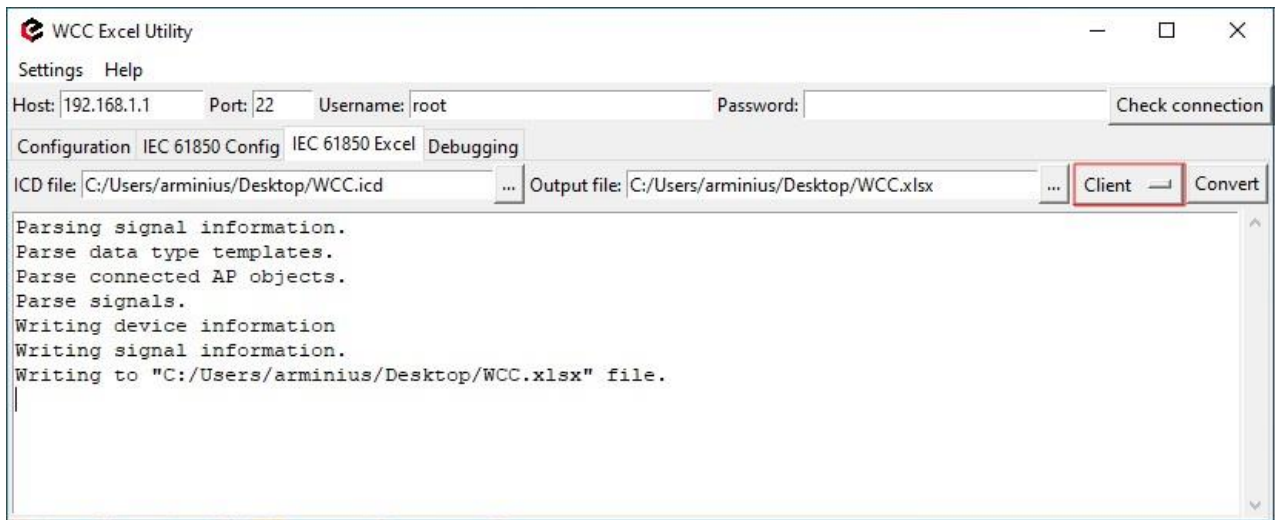
To generate the IEC 61850 Client model file select "Client" in the drop-down selection tab. Then select where to output the generated model and upload a file with extensions .icd, .scd, or .cid.



Generate IEC 61850 Client model file

Generate excel file

To generate the IEC 61850 Client Excel file select "Client" in the drop-down selection tab. Then select where to output the generated model upload file with extensions .icd, .scd, or .cid.



Generate IEC 61850 Client Excel file

After generating the Excel file additional configuration information must be written in the devices sheet:

- A valid host ip address must be provided.
- An authorization method must be provided (if it is a complex authorization method, additional parameters might be required).
- Model filename must be provided. The model filename must be the same as that was generated one step earlier (The model filename can include an extension, but it is not mandatory).

I	J	K
host	authorization	model_filename
192.168.122.146	none	WCC

Excel configuration (Devices sheet)

In the signals sheet, signals that are not used or needed can be removed. Their information might be modified as well.

	A	B	C	D	E	F	G	H	I	J	K
1	device alias	signal name	signal alias	ld_instance	ln_class	ln_instance	ln_prefix	cdc	data object	data_fc	number type
2	iec 61850 client	LD0_GGIO_1_SPS1_stVal	LD0_GGIO_1_SPS1_stVal	LD0	GGIO	1		SPS	SPS1	ST	BOOLEAN
3	iec 61850 client	LD0_GGIO_1_SPS2_stVal	LD0_GGIO_1_SPS2_stVal	LD0	GGIO	1		SPS	SPS2	ST	BOOLEAN
4	iec 61850 client	LD0_GGIO_2_SPC1_origin.orCat	LD0_GGIO_2_SPC1_origin.orCat	LD0	GGIO	2		SPC	SPC1	ST	INT8
5	iec 61850 client	LD0_GGIO_2_SPC1_origin.orient	LD0_GGIO_2_SPC1_origin.orient	LD0	GGIO	2		SPC	SPC1	ST	OCTET STRING 6
6	iec 61850 client	LD0_GGIO_2_SPC1_ctlNum	LD0_GGIO_2_SPC1_ctlNum	LD0	GGIO	2		SPC	SPC1	ST	INT8U
7	iec 61850 client	LD0_GGIO_2_SPC1_stSeld	LD0_GGIO_2_SPC1_stSeld	LD0	GGIO	2		SPC	SPC1	ST	BOOLEAN
8	iec 61850 client	LD0_GGIO_2_SPC1_stVal	LD0_GGIO_2_SPC1_stVal	LD0	GGIO	2		SPC	SPC1	ST	BOOLEAN
9	iec 61850 client	LD0_GGIO_2_SPC1_Oper.ctiVal	LD0_GGIO_2_SPC1_Oper.ctiVal	LD0	GGIO	2		SPC	SPC1	CO	BOOLEAN
10	iec 61850 client	LD0_GGIO_2_SPC2_origin.orCat	LD0_GGIO_2_SPC2_origin.orCat	LD0	GGIO	2		SPC	SPC2	ST	INT8
11	iec 61850 client	LD0_GGIO_2_SPC2_origin.orient	LD0_GGIO_2_SPC2_origin.orient	LD0	GGIO	2		SPC	SPC2	ST	OCTET STRING 6
12	iec 61850 client	LD0_GGIO_2_SPC2_ctlNum	LD0_GGIO_2_SPC2_ctlNum	LD0	GGIO	2		SPC	SPC2	ST	INT8U
13	iec 61850 client	LD0_GGIO_2_SPC2_stSeld	LD0_GGIO_2_SPC2_stSeld	LD0	GGIO	2		SPC	SPC2	ST	BOOLEAN
14	iec 61850 client	LD0_GGIO_2_SPC2_stVal	LD0_GGIO_2_SPC2_stVal	LD0	GGIO	2		SPC	SPC2	ST	BOOLEAN
15	iec 61850 client	LD0_GGIO_2_SPC2_Oper.ctiVal	LD0_GGIO_2_SPC2_Oper.ctiVal	LD0	GGIO	2		SPC	SPC2	CO	BOOLEAN
16	iec 61850 client	LD0_GGIO_3_DPS1_stVal	LD0_GGIO_3_DPS1_stVal	LD0	GGIO	3		DPS	DPS1	ST	CODEENUM
17	iec 61850 client	LD0_GGIO_3_DPS2_stVal	LD0_GGIO_3_DPS2_stVal	LD0	GGIO	3		DPS	DPS2	ST	CODEENUM
18	iec 61850 client	LD0_GGIO_4_DPC1_origin.orCat	LD0_GGIO_4_DPC1_origin.orCat	LD0	GGIO	4		DPC	DPC1	ST	INT8
19	iec 61850 client	LD0_GGIO_4_DPC1_origin.orient	LD0_GGIO_4_DPC1_origin.orient	LD0	GGIO	4		DPC	DPC1	ST	OCTET STRING 6
20	iec 61850 client	LD0_GGIO_4_DPC1_ctlNum	LD0_GGIO_4_DPC1_ctlNum	LD0	GGIO	4		DPC	DPC1	ST	INT8U
21	iec 61850 client	LD0_GGIO_4_DPC1_stSeld	LD0_GGIO_4_DPC1_stSeld	LD0	GGIO	4		DPC	DPC1	ST	BOOLEAN
22	iec 61850 client	LD0_GGIO_4_DPC1_stVal	LD0_GGIO_4_DPC1_stVal	LD0	GGIO	4		DPC	DPC1	ST	CODEENUM
23	iec 61850 client	LD0_GGIO_4_DPC1_Oper.ctiVal	LD0_GGIO_4_DPC1_Oper.ctiVal	LD0	GGIO	4		DPC	DPC1	CO	BOOLEAN
24	iec 61850 client	LD0_GGIO_4_DPC2_origin.orCat	LD0_GGIO_4_DPC2_origin.orCat	LD0	GGIO	4		DPC	DPC2	ST	INT8
25	iec 61850 client	LD0_GGIO_4_DPC2_origin.orient	LD0_GGIO_4_DPC2_origin.orient	LD0	GGIO	4		DPC	DPC2	ST	OCTET STRING 6
26	iec 61850 client	LD0_GGIO_4_DPC2_ctlNum	LD0_GGIO_4_DPC2_ctlNum	LD0	GGIO	4		DPC	DPC2	ST	INT8U
27	iec 61850 client	LD0_GGIO_4_DPC2_stSeld	LD0_GGIO_4_DPC2_stSeld	LD0	GGIO	4		DPC	DPC2	ST	BOOLEAN
28	iec 61850 client	LD0_GGIO_4_DPC2_stVal	LD0_GGIO_4_DPC2_stVal	LD0	GGIO	4		DPC	DPC2	ST	CODEENUM
29	iec 61850 client	LD0_GGIO_4_DPC2_Oper.ctiVal	LD0_GGIO_4_DPC2_Oper.ctiVal	LD0	GGIO	4		DPC	DPC2	CO	BOOLEAN
30	iec 61850 client	LD0_GGIO_5_AnIn1_mag.f	LD0_GGIO_5_AnIn1_mag.f	LD0	GGIO	5		MV	AnIn1	MX	FLOAT32
31	iec 61850 client	LD0_GGIO_5_AnIn2_mag.f	LD0_GGIO_5_AnIn2_mag.f	LD0	GGIO	5		MV	AnIn2	MX	FLOAT32

Signals sheet

Important! Information such as ld_instance and other data taken directly from SCD configuration files should not be modified, otherwise, access to info of these reports can be broken.

Uploading configuration

First, upload the model configuration file.

CONFIGURATION

IMPORTED SIGNALS

EVENT LOG

PROTOCOL CONNECTIONS

SCRIPT-RUNNER

Protocol configuration

IMPORT PROTOCOL CONFIGURATION

Here you can import Excel configuration file. Up to 1000 signals are allowed. All previous signals will be replaced.

Configuration file:	<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Import configuration"/>
PLC (IEC-61499) Boot file:	<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Import FBOOT file"/>
IEC61850 Client model file:	<input type="button" value="Choose File"/> WCC.client	<input type="button" value="Import client model file"/>
IEC61850 Server model file:	<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Import server model file"/>

DOWNLOAD CONFIGURATION

Template configurations:

Uploading model configuration file

After uploading the model configuration file it should appear under the DOWNLOAD CONFIGURATION tab.

DOWNLOAD CONFIGURATION

Template configurations:

Current IEC 61850 client model file (WCC.client):

Uploaded IEC 6180 Client configuration file

Then upload the Excel configuration (same as with every other protocol).

PROTOCOL HUB

STATUS

SYSTEM

SERVICES

NETWORK

USERS

LOGOUT (ROOT)

CONFIGURATION

IMPORTED SIGNALS

EVENT LOG

PROTOCOL CONNECTIONS

Protocol configuration

IMPORT PROTOCOL CONFIGURATION

Here you can import Excel configuration file. Up to 1000 signals are allowed. All previous signals will be replaced.

Configuration file:	<input type="button" value="Choose File"/> WCC.xlsx	<input type="button" value="Import configuration"/>
PLC (IEC-61499) Boot file:	<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Import FBOOT file"/>
IEC61850 Client model file:	<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Import client model file"/>
IEC61850 Server model file:	<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Import server model file"/>

Uploading excel configuration

After successful configuration upload, both configurations should appear under the DOWNLOAD CONFIGURATION tab. If any errors occur during Excel upload, fix them along Excel utility guidelines.

DOWNLOAD CONFIGURATION

Current configuration (WCC.xlsx):

Download

Template configurations:

Download

Current IEC 61850 client model file (WCC.json):

Download

Delete

Uploaded configurations

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 the 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 the command line interface and find out why the link is not functioning properly.

To launch a debugging session, a user should stop iec61850-client process by running `/etc/init.d/iec61850client stop` 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 (Soltronic 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 the 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 requests in milliseconds	Yes		0	60000
serial_close_delay	integer	Delay before closing serial port	No	400		

At command 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			

16.2 Aurora

Overview

The Aurora Protocol is a link layer communications protocol for use on point-to-point 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	None, Even, Odd	
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 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 in milliseconds	No	2500		
id	integer	Inverter ID	No	0		

device	string	Communication port	Yes		PORT1	PORT2
--------	--------	--------------------	-----	--	-------	-------

Aurora parameters for the 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 the 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	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a 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

16 Specific protocols

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 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 in milliseconds	Yes		0	60000

Comlynx parameters for the 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	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a 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 the 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

16 Specific protocols

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	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 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 in milliseconds	Yes		0	60000
id	integer	Inverter ID	Yes	0		
device	string	Communication port	Yes		PORT1	PORT2

Delta 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			
pulse_short_time_ms	integer	The time interval for a short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	0		

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, 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

16 Specific protocols

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 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 in milliseconds	No	2500		
id	integer	Inverter ID	Yes	0		
device	string	Communication port	Yes		PORT1	PORT2

GINLONG parameters for the 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	string	Define tagfunction	Yes			
tag_job_todo	string	Define tag action that depends on tag function	Yes			
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		

pulse_long_time_ms	integer	The time interval for a 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 the 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 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 incoming requests in milliseconds	Yes	2500	0	60000
subid	integer	Inverter serial number display	Yes	0		
ext_device	boolean	0 - The inverter is disconnected directly 1 - Inverter is connected via remote terminal	Yes	0	0	1

16 Specific protocols

id	integer	Inverter serial number	Yes			
device	string	Communication port	Yes		PORT1	PORT2

Kaco 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			
pulse_short_time_ms	integer	The time interval for a short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	0		

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 the 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		powerone	
serialnumber	integer	Inverter serial number	Yes			
type	string	Inverter type : <ul style="list-style-type: none"> • CU • • Collecting unit CB - Normal CB HID - HID with integrated CB 	No	CU	CU, CB, HID	
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 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 incoming requests in milliseconds	Yes	1000	0	60000

PowerOne parameters for Signals tab

Parameter	Type	Description	Required	Default value (when not specified)	Range	
					Min	Max

16 Specific protocols

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			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a 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 the 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 milliseconds	No	10000		
poll_delay_ms	integer		No	200		
timeout_ms	integer	Timeout of waiting for incoming requests in milliseconds	No	2500		
serial_number	string	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 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

16 Specific protocols

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			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	0		

16.9 SOLPLUS

Overview

Solplus protocol is used to download inverter data from Solplus inverters using an 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 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 incoming requests in milliseconds	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 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			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	0		

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 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 in milliseconds	No	2500	0	60000
------------	---------	-------------------------	----	------	---	-------

VBUS parameters for the 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	string	Define tagfunction	Yes			
tag_job_todo	string	Define tag action that depends on tag function	Yes			
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a 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

Vesta's parameters for the 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 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 in milliseconds	No	500	0	60000

Vesta's parameters for the 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	string	Define tagfunction	Yes			
tag_job_todo	string	Define tag action that depends on tag function	Yes			
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a 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 the 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 requests 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 WCC Lite RTU that is connected at periodic intervals (scan_rate_ms) to read all utility meters.

Configuration

M-Bus parameters for the Device tab

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU		Min	Max
name	string	User-friendly device name	Yes	Yes			
description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling a device	No	No	1	0	1
protocol	string	Protocol to be used.	Yes	Yes		mbus serial, mbus tcp	

16 Specific protocols

scan_rate_ms	integer	All reads and writes will be executed within the timeframe in milliseconds.	No	No	10000		
poll_delay_ms	integer	Minimum time delay in milliseconds to wait before sending any data on port.	No	No	200		
timeout_ms	integer	Timeout of waiting for an incoming response in milliseconds	Yes	Yes		0	60000
address	string	Device address	Yes	Yes			
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	
serial_close_delay	integer	Delay before closing the serial connection.	-	No	400		
ip	string	The IP address of the TCP slave device	Yes	-			
port	integer	TCP communication port	Yes	-		0	65535

M-Bus parameters for the Signals tab

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU		Min	Max
signal_name	string	User-friendly signal name	Yes	Yes			
device_alias	string	Device alias from a Devices tab	Yes	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be used	Yes	Yes			
enable	boolean	Enabling/disabling of an individual signal	No	No	1	0	1

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

16.14 KOSTAL

Overview

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

Configuration

Kostal parameters for the 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

16 Specific protocols

parity	string	Communication parity option	No	none	none, even, odd	
scan_rate_ms	integer	Delay before closing serial port 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 incoming requests in milliseconds	Yes		0	60000

Kostal 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			
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	0		

17 Metering protocols +

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

17 Metering protocols +

17.1 DLMS/COSEM

Introduction

IEC 62056 is a set of standards for electricity metering data exchange by the 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, and 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 using 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) ports 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.

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

Device configuration parameters for DLMS meters acquisition:

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU		Min	Max
name	string	User-friendly name for a device	Yes	Yes			
description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
protocol	string	Protocol to be used	Yes	Yes		DLMS	
serial_number	integer	Meter serial number	No	No	0		
physical_address	integer	Meter's physical server address	No	No	1600		
logical_address	integer	Meter logical server address	No	No	0		

address_size	integer	Meter address size in bytes	No	No	1	1	4
client_address	integer	Client address	Yes	Yes			
type	string	Meter object referencing: SN - short referencing, LN - logical referencing	No	No	SN	SN, LN	
mode	string	Initial handshake mode.	Yes	Yes	DLMS--HDLC	IEC, DLMS, DLMS-HDLC or DLMS-WRAPPER	
timeout_ms	integer	Timeout in milliseconds	No	No	2500		
auth	string	Authentication.	No	No	None	None, Low, High, HighMd5, HighSha1, HighSha256, HighGmac, HighEcdsa	
password	string	Password for authentication	No (when auth is None)	No (when auth is None)			
ip	string	IP address	Yes	-			
port	integer	TCP port	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	
flowcontrol	string	Communication device flow control option.	-	No	none	none	
retry_counter	integer	Number of requests, before the link is considered lost (device status signals are changed) and reconnect attempt will be issued	No	No	3		

scan_rate_ms	integer	If provided and positive all reads and writes will be executed within the timeframe in milliseconds	No	No	10000		
reconnect_time	integer	Defines how often (in milliseconds) the client will try to reestablish communication with the meter after an unsuccessful attempt.	No	No	1000		

Signals section

DLMS configuration parameters creating signals:

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU		Min	Max
signal_name	string	User-friendly signal name	Yes	Yes			
device_alias	string	Device alias from a Devices tab	Yes	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be used	Yes	Yes			
enable	boolean	Enabling/disabling of an individual signal	No	No	1	0	1
log	boolean	Enable logging in the event log	No	No	0	0	1
short_name	integer	Address of value to read (Short name).	No	No	0		
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	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 the dlms process and run the 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 WCC Lite+:
/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 Metering protocols +

17.2 IEC 62056-21

Introduction

IEC 61107 or currently IEC 62056-21, was an international standard for a computer protocol 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 20mA current loop. 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. the 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 the 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, liters 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, a 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 whatever 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	
			TCP	RTU		Min	Max
name	string	User-friendly name for a device	Yes	Yes			
description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
protocol	string	Protocol to be used	Yes	Yes		IEC 62056-21	
poll_delay_ms	integer	Minimum time delay in milliseconds to wait before sending any data on port.	No	No	200		
scan_rate_ms	integer		No	No	10000		
device	string	Communication port	-	No		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	
serialnumber	unsigned long	Meter serial number	Yes	Yes		1	
serial_close_delay	integer	Delay before closing serial port	-	No	400		
timeout_ms	integer	Timeout of waiting for incoming request	No	No	2500		
type	string	Defines a connection mode	No	No	C	A, B, C	
t2	integer	Time to wait before acknowledging the suggested baud rate in mode C	No	No	300	200	1500
ip	string	IP address for TCP connection	Yes	-			
port	integer	TCP port	Yes	-		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. 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	
			TCP	RTU		Min	Max
signal_name	string	User-friendly signal name	Yes	Yes			
device_alias	string	Device alias from a Devices tab	Yes	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be used	Yes	Yes			
enable	boolean		No	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	No	0	0	
number_type	string	Number format type	Yes	Yes			
tag_job_todo	string	Tag job in OBIS format	Yes	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 Metering protocols +

17.3 Elgama

Overview

Elgama protocol is used for communications with Elgama 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 requests in milliseconds	Yes			
id	integer	Meter serial number	Yes			
meter_model	integer	Meter type	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 the event log	No	0		
number_type	string	Type of number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
job_todo	string	Elgama protocolspecific data set values (see files below)	Yes			
tag_job_todo	string	Tag sub job	Yes			

Files

1. Example configuration file for Elgama protocol data mapping [download](#)

18 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 an MIB (Management Information Base).

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

19 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. SCADA) that have their 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**.

Overview

Data export service gathers information from other protocols and 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 be set up, for instance, some of the information can be sent to an FTP server, while others could be transmitted to the HTTP server which can handle POST requests.

Using WCC Lite+ for data export

To configure WCC Lite+ to use a data export server a user can fill in the needed parameters in Excel configuration. These parameters are shown in the two tables below. Default values are shown in 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	The time frame during which transmission to the remote server has to be completed (in seconds)	No	5		
type	string	Selection of file format	No	csv-simple	cvs-periodic csv-simple, json-simple, json-samples	
host	string	A URL of the remote server where files should be sent	Yes			
upload_rate_sec	integer	Frequency of generated file uploads (in seconds)	No	60		
records_buffer_size	integer	A maximum amount of data change entries to hold before initiating the logging mechanism	No	100		
logging_period_sec	integer	Describe how frequently the data buffer of records_buffer_size is saved to the file	No	10	1	3600
log_folder	string	A folder in the WCC Lite+ file system to save generated files (" /var/cache/data-export ")	No			
timestamp	string	Selection of time format	No	unixtimestamp	unixtimestamp, iso8601	

compress	string	Selection of file compression mechanism	No	none	none, gz, tar.gz
compress_password	string	Enable the feature of file password protection	No		yes, true
csv_field_separator	string	Columns separator in .csv file format	No	"," - (comma)	"," - (comma) ";" - (semicolon) "." - (dot) " " - (whitespace) " " - (pipe)
csv_decimal_separator	string	Decimal separator in values	No	"." - (dot)	"." - (dot) ", " - (comma)

i The 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.

The data generation rate may be going to be bigger than what the 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 the 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 a different Excel sheet called DataExportThe

parameters to be filled in the DataExport sheet are shown in the table below.


Data export (data-export) parameters for the DataExport tab

Parameter	Type	Description	Required	Default value (when not specified)	Range	
					Min	Max
device_alias	string	Device alias to be used in configuration Yes	Yes			
device_name	string	User-friendly device name as in the Device sheet	Yes			
tag_name	string	User-friendly signal name	Yes			
source_device_alias	string	device_alias of a source device	Yes			
source_signal_aliases	string	source_alias of a source signal	Yes			
enable	boolean	Enabling/disabling of a measurement to be transmitted and logged	No	1	0	1
attribute	string	An additional attribute to be attached to a signal	No			

Debugging data export service

If the configuration for the Data export service is set up, a handler for the protocol will start automatically. If the configuration is missing parameters or contains errors, the 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 the 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 the Data export service does not work properly (e.g. data is corrupted, etc.), a user can launch a debug session from the 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 a data-export command with respective flags as in the table above.

Host URL format rules

The 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 users and/or passwords;
- Port - useful when non-standard value is used;
- Endpoint - a place in the server to which a call is made

The format for the host parameter can be summarized as:

```
[ [ h t t p ( s ) / f t p ( s ) ] : / / [ [ u s e r ] : [ p a s s w o r d ] @ ] [ U R L ] [ : p o r t ] / [ e n d p o i n t ] ]
```

Options are printed in square brackets. A protocol has to be selected, otherwise HTTP will be used as a default. The user and password pair is optional, but if the user: password pair is used, it should proceed with the @ 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 the firewall on both the server and client side, otherwise, data will not be sent successfully.

Finally, a 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 has 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; attribute 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 }
    ]
  }
]
}

```

20 Programmable logic controller

+

A programmable logic controller (PLC) is a digital device adapted to control processes requiring 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 the IEC 61131 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 the IEC 61131 standard. WCC Lite+ supports PLC functionality while conforming to the specifications of the 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:

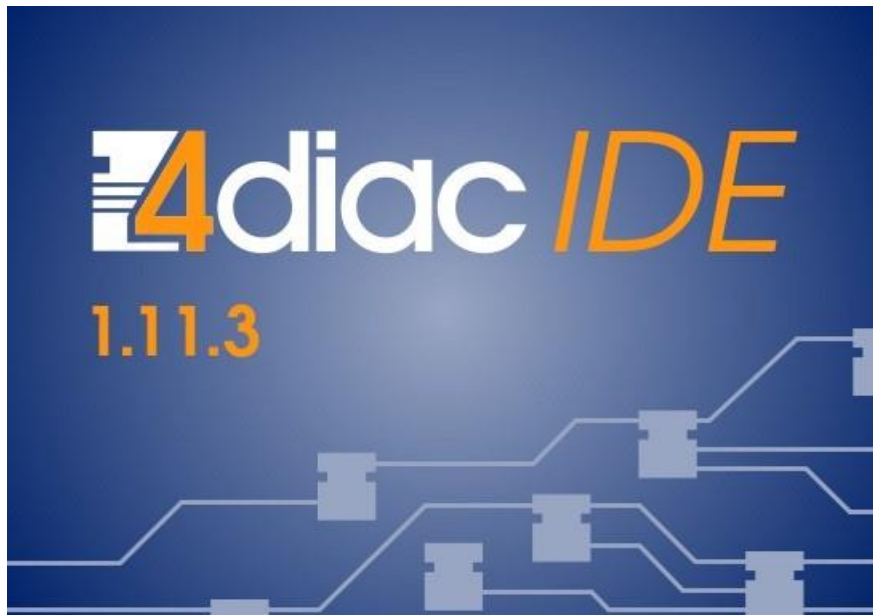
Service interface function block – SIFB: The source code is hidden and its functionality is only described by service sequences;

- Basic function block - BFB: Its functionality is described in terms of an Execution Control Chart (ECC), which is similar to a state diagram (UML). Every state can have several actions. Each action references one or zero algorithms and one or zero events. Algorithms can be implemented as defined in compliant standards.
- Composite function block - CFB: Its functionality is defined by a function block network.

- Adapter interfaces: An adapter interface is not a real function block. It combines several events and data connections within one connection and provides an interface concept to separate specification and implementation.
- Sub application: Its functionality is also defined as a function block network. In contrast to CFBs, sub-applications can be distributed.


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


20.2 4Diac framework



The PLC functionality in the WCC Lite+ is implemented using the 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 take a look at the 4diac manual - this document is only 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 a web interface.

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

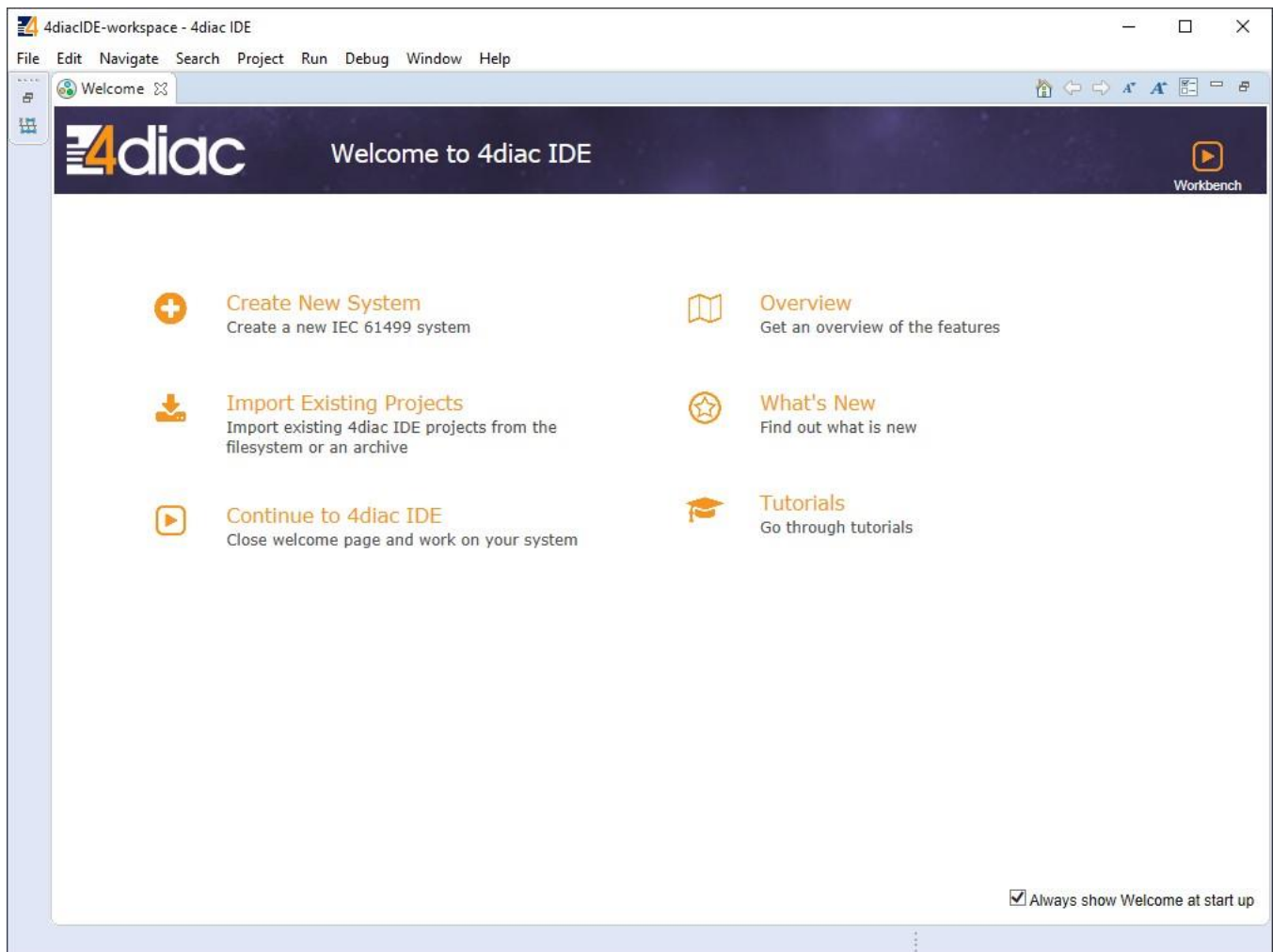
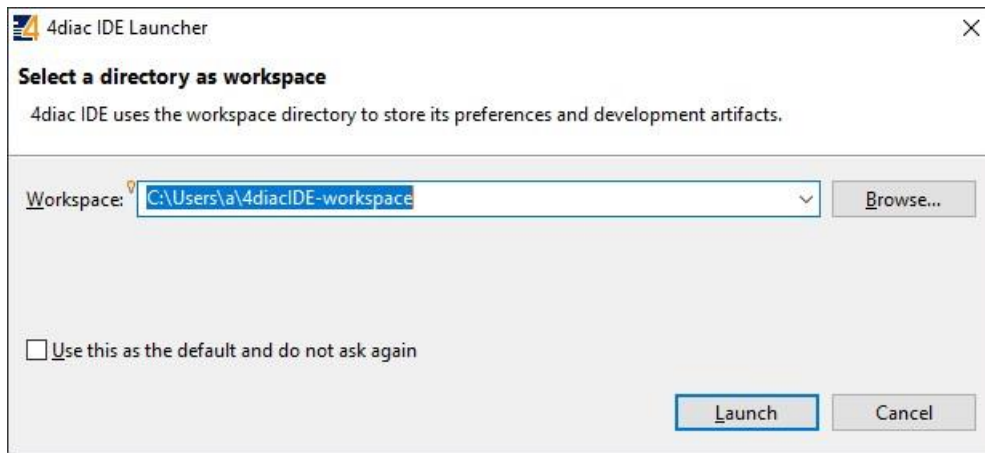
 It is possible to run multiple tasks at once. These tasks can either be implemented on the same screen or split into separate tasks. Please note, however, that all elements should have unique names, even between different tasks. As of 4Diac IDE 1.11.3, this is not enforced between separate apps, however,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

The 4diac FORTE runtime can 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 the 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 a definitive guide on how to use 4Diac IDE or how to interpret the 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.



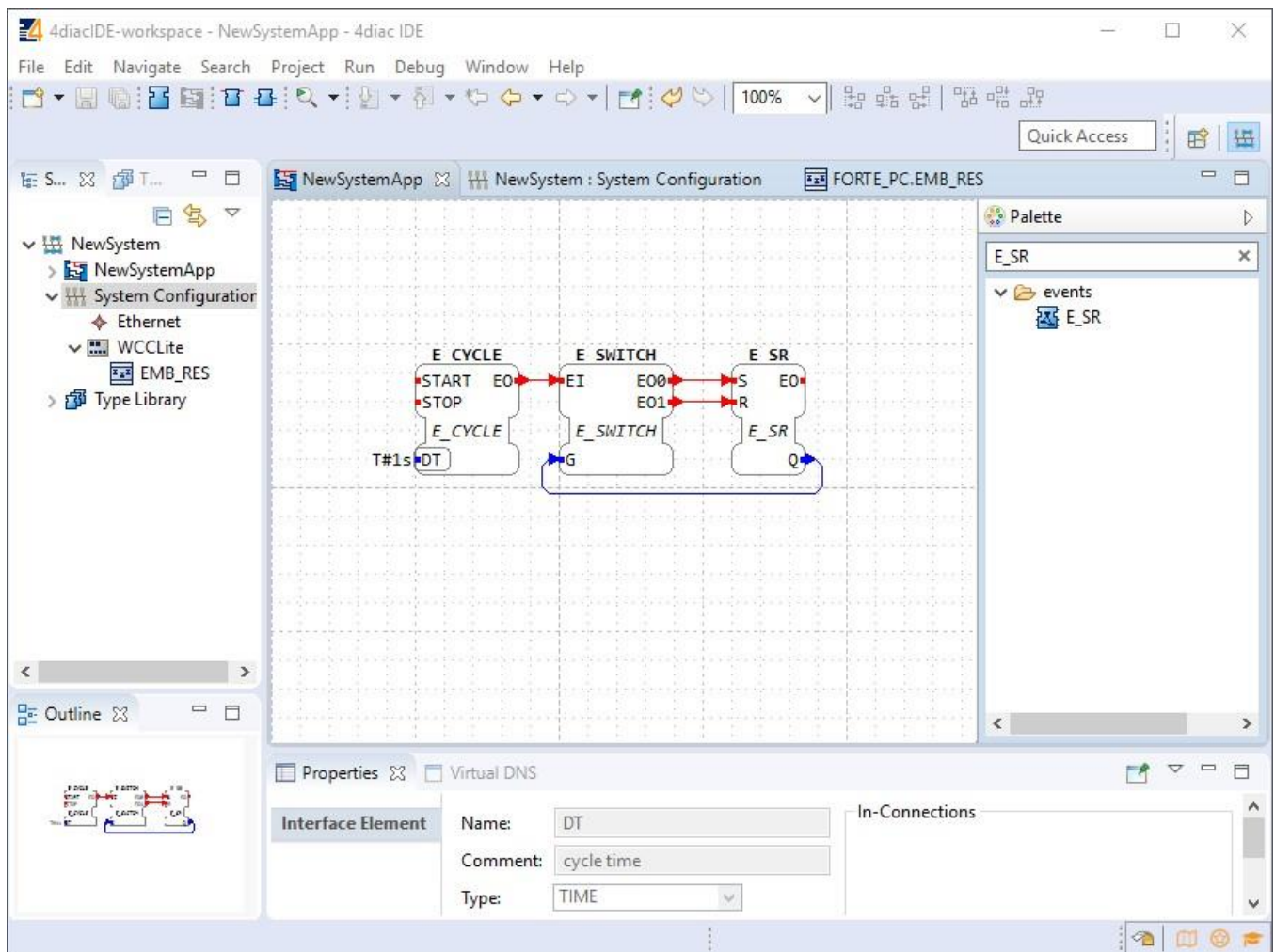
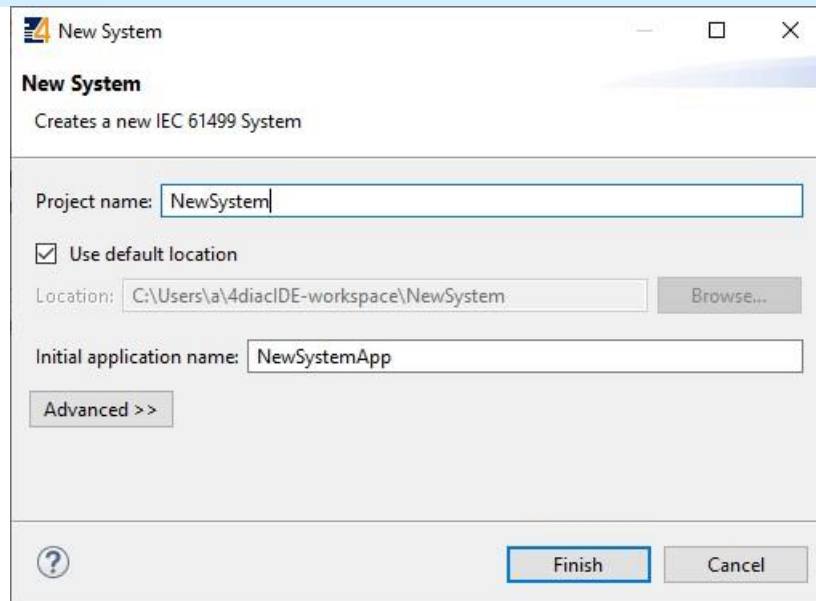
After that, a user should be met by the welcome window. If such a window is not shown, one can create a project by selecting File --> New --> Project and filling in the required fields.

Here is a library of function blocks, which are working with the newest 4Diac version: [Elseta Library.zi p](#). Upload the library to your project folder.

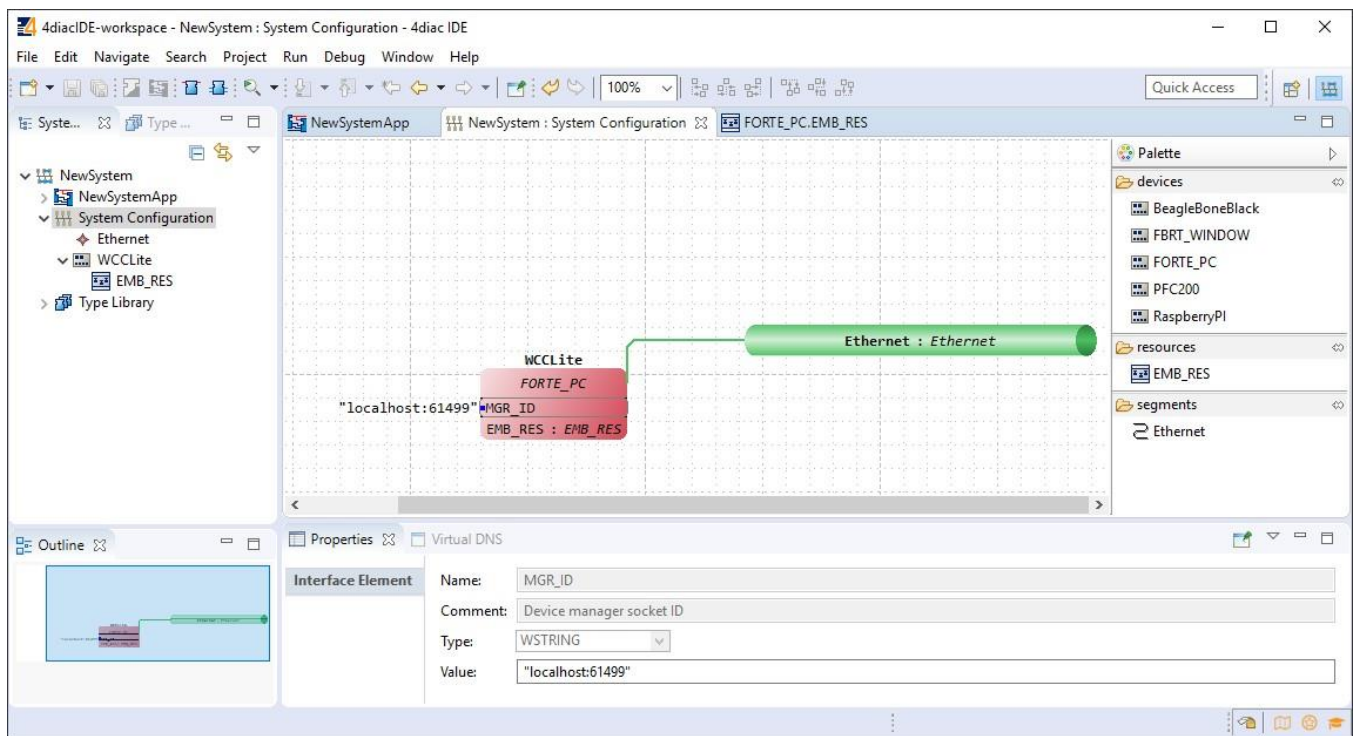
To create a simple application, simply drag and drop objects from the palette to the canvas and wire them accordingly.

Event trigger and data pathways cannot be connected. Displayed below is an example of a simple blinker application.

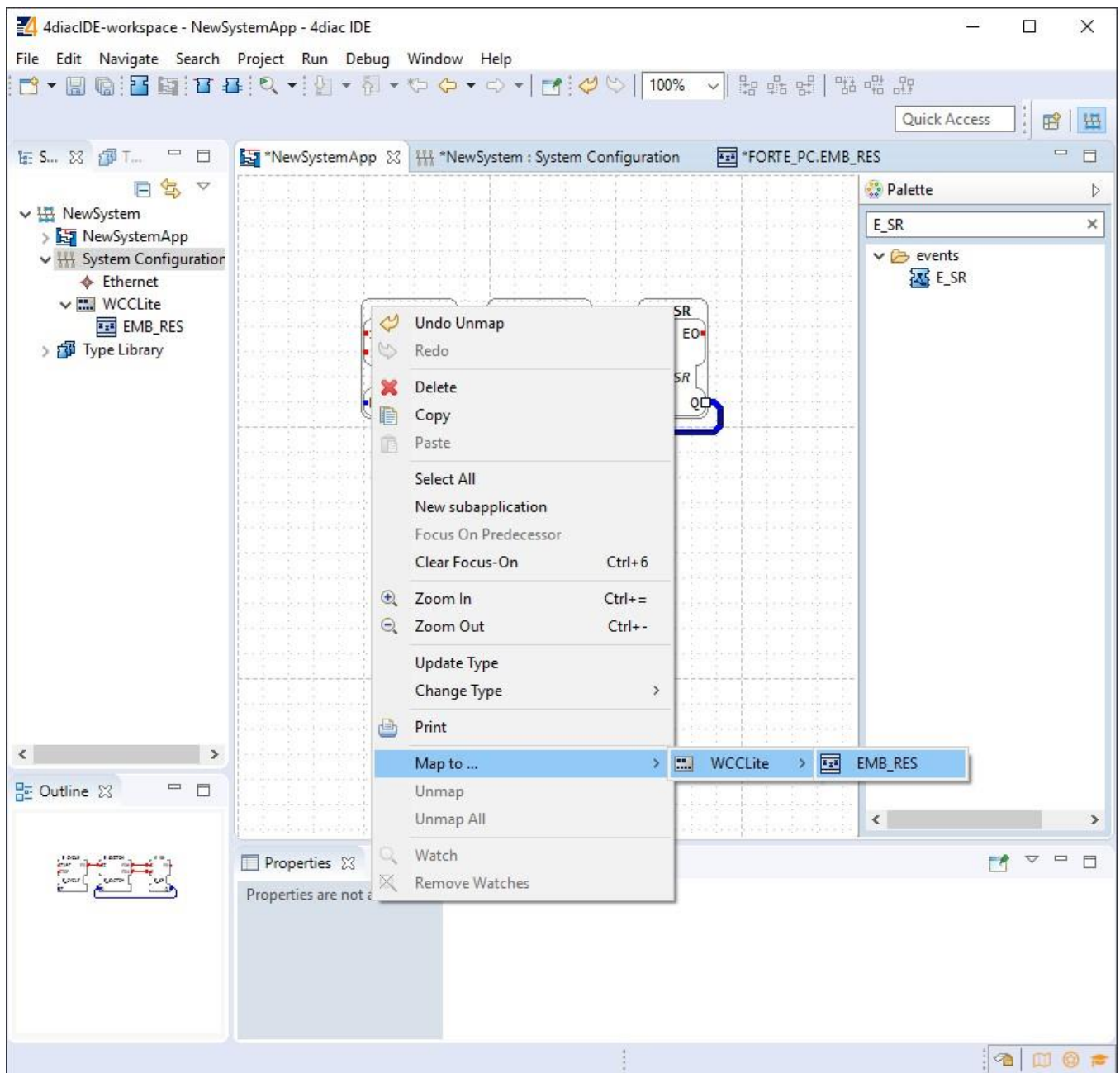
Having less wiring by connecting several signals to the same subnet as PCB designer (such as Altium Designer) as of 4Diac IDE 1.11.3 is not supported. However, if some parts are used frequently, it is highly advised to have less wiring by simply compiling several elements into a sub-application. For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



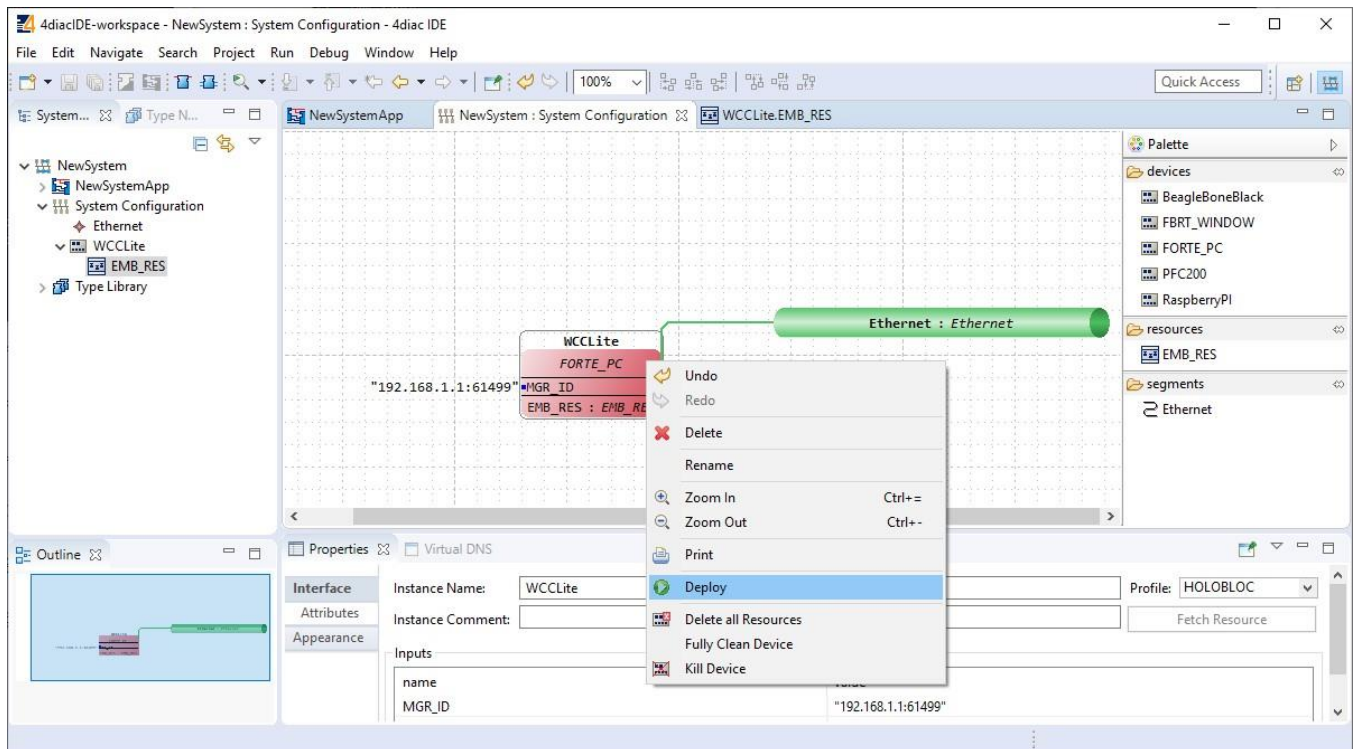
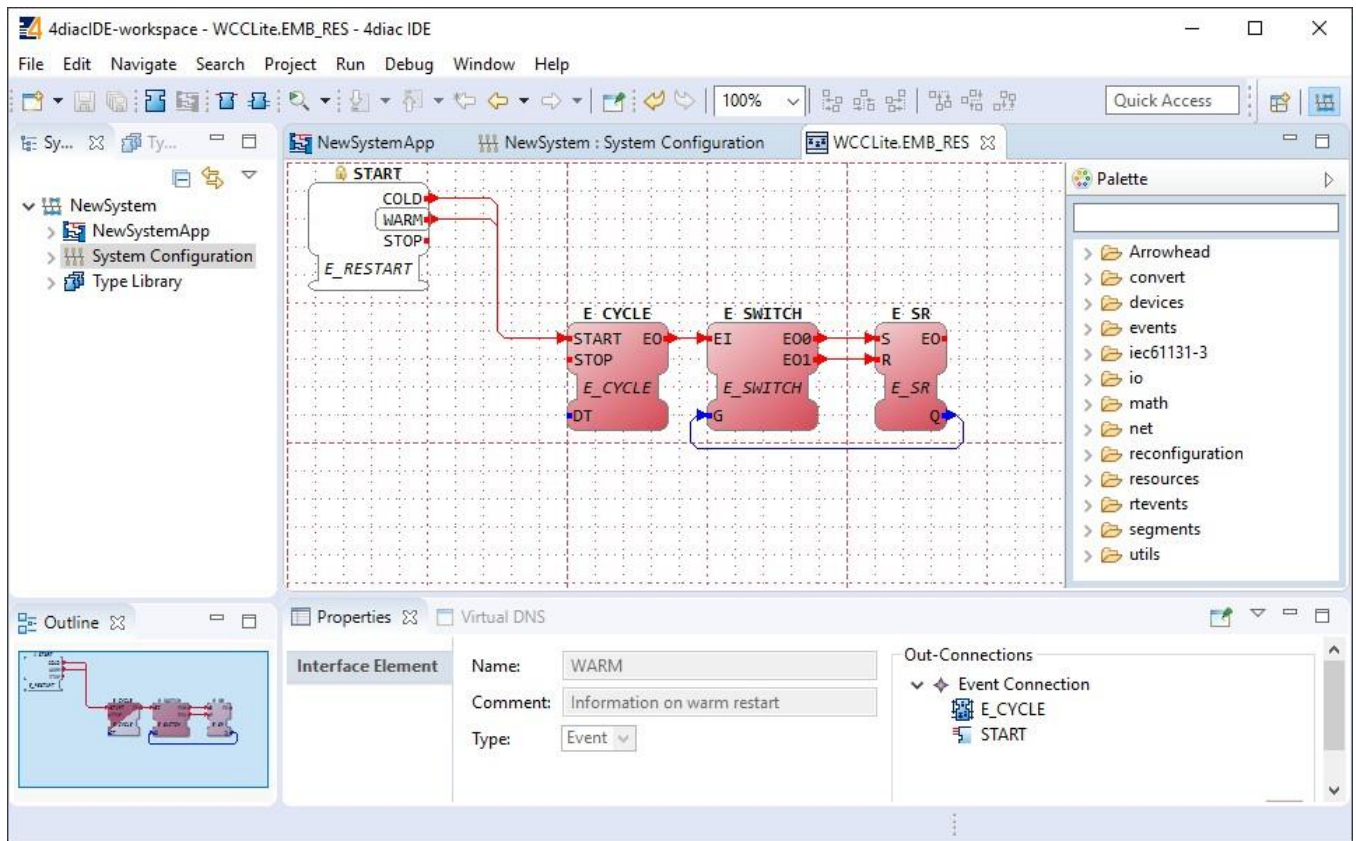
In the System Configuration section, drag and drop a FORTE_PC device, and an Ethernet segment and link them. For debugging in the local (PC) runtime, leave the address "localhost:61499". For testing on a WCC Lite+, enter the IP address of the device, along with the port number (which by default is 61499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



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:



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:



To deploy the application, go to the System Configuration tab and simply select "Deploy" from the right-click menu of the controller device. If a running application exists 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 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 has not been linked to a PLC program, all calls from the PLC runtime application to the 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, and 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 the Devices tab

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

Table. Mandatory parameters for the Signals tab

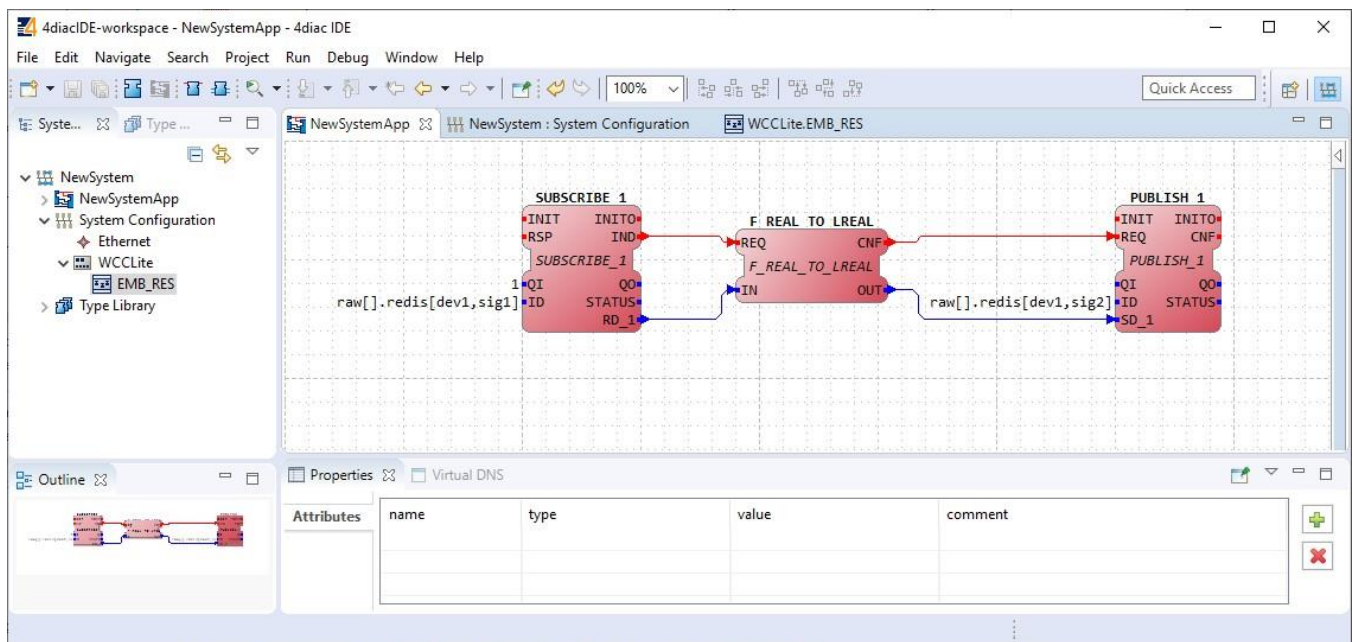
Parameter	Type	Description
signal_name	string	User-friendly signal name
device_alias	string	Device alias from a Devices tab
signal_alias	string	Unique signal name to be used
source_device_alias	string	device_alias of a source device
source_signal_alias	string	signal_alias of a source signal

If an upload consisting of configuration for IEC 61499 has been successful, 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 the correct firmware version installed and haven't made any typing errors.

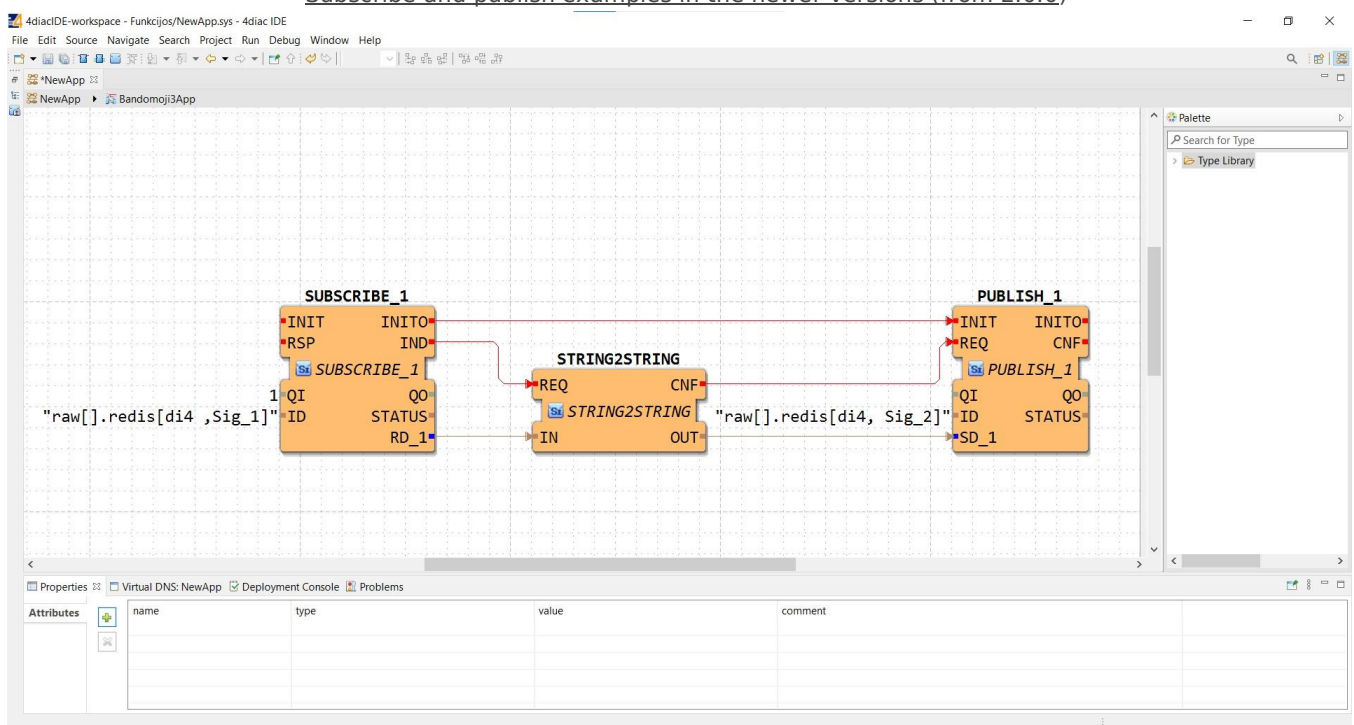
Parameters mentioned earlier, namely device_alias and signal_alias, are the only parameters one needs to fill to bind Excel configuration to the 4Diac framework. Two types of blocks are used for data transmission - PUBLISH blocks to write data to the REDIS database and SUBSCRIBE blocks to acquire data from the database as soon as it changes its value. Both of them have an ID connection. To connect a block to a data point, one should set this pin as raw[].redis[device_alias,signal_alias], e.g. raw[].redis[example_plc_device,example_plc_signal_alias]. As of version 2.0.0, the pin values must be surrounded by double commas.

Examples with SUBSCRIBE and PUBLISH function blocks are shown in the images below.

[Subscribe and publish examples in the older versions](#)



Subscribe and publish examples in the newer versions (from 2.0.0)



⚠ 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 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.

⚠ The functionality of the subscribe function is dependent on the presence of the "F_STRING_TO_" function block, for example (F_STRING_TO_REAL). Likewise, the publish function's proper operation relies on the availability of the "F_TO_STRING" FB, for example (F_REAL_TO_STRING).

If every step until now has been successful, the user can now proceed to debug the PLC application.

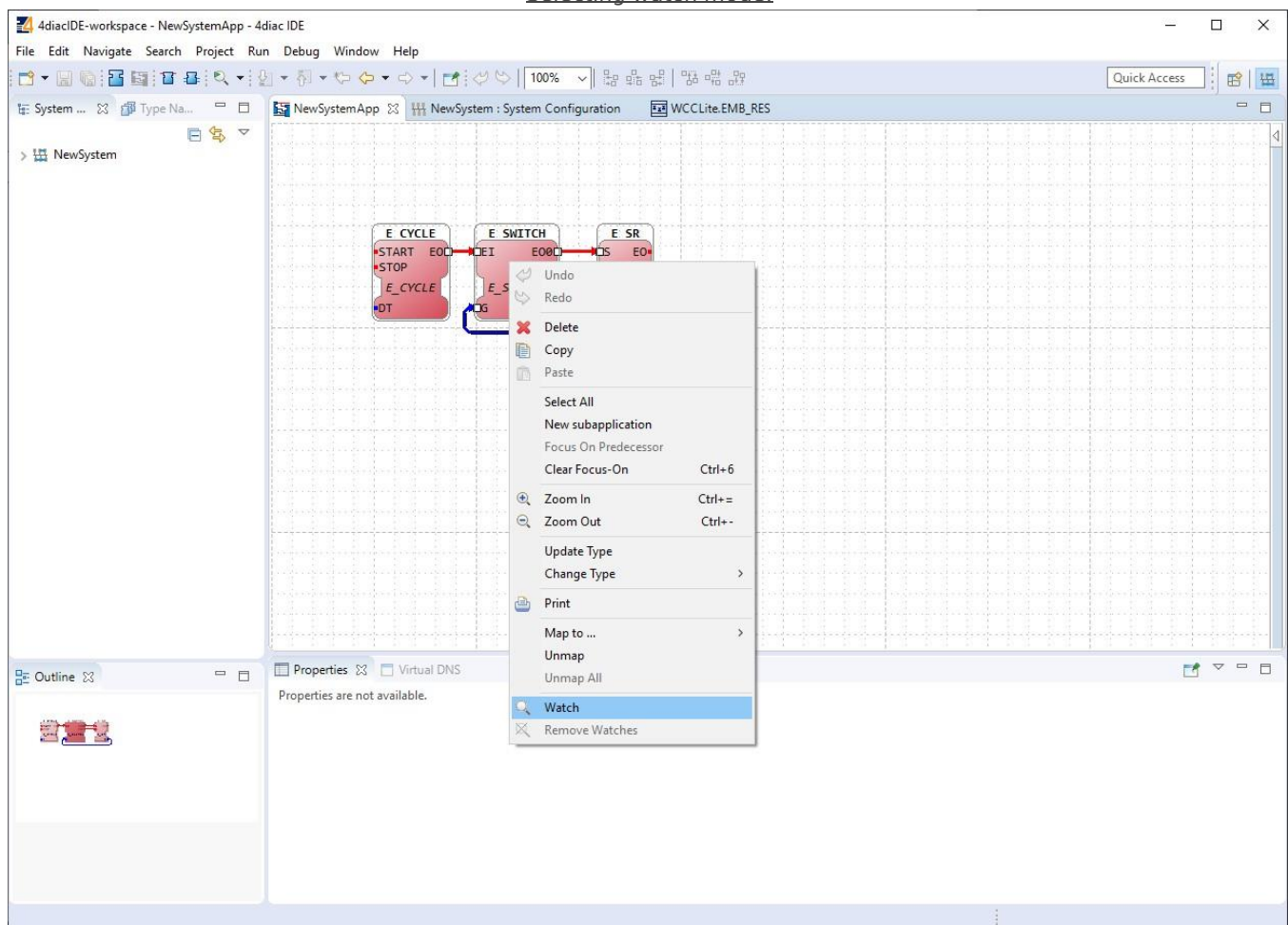
20.5 Debugging an IEC 61499 application

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

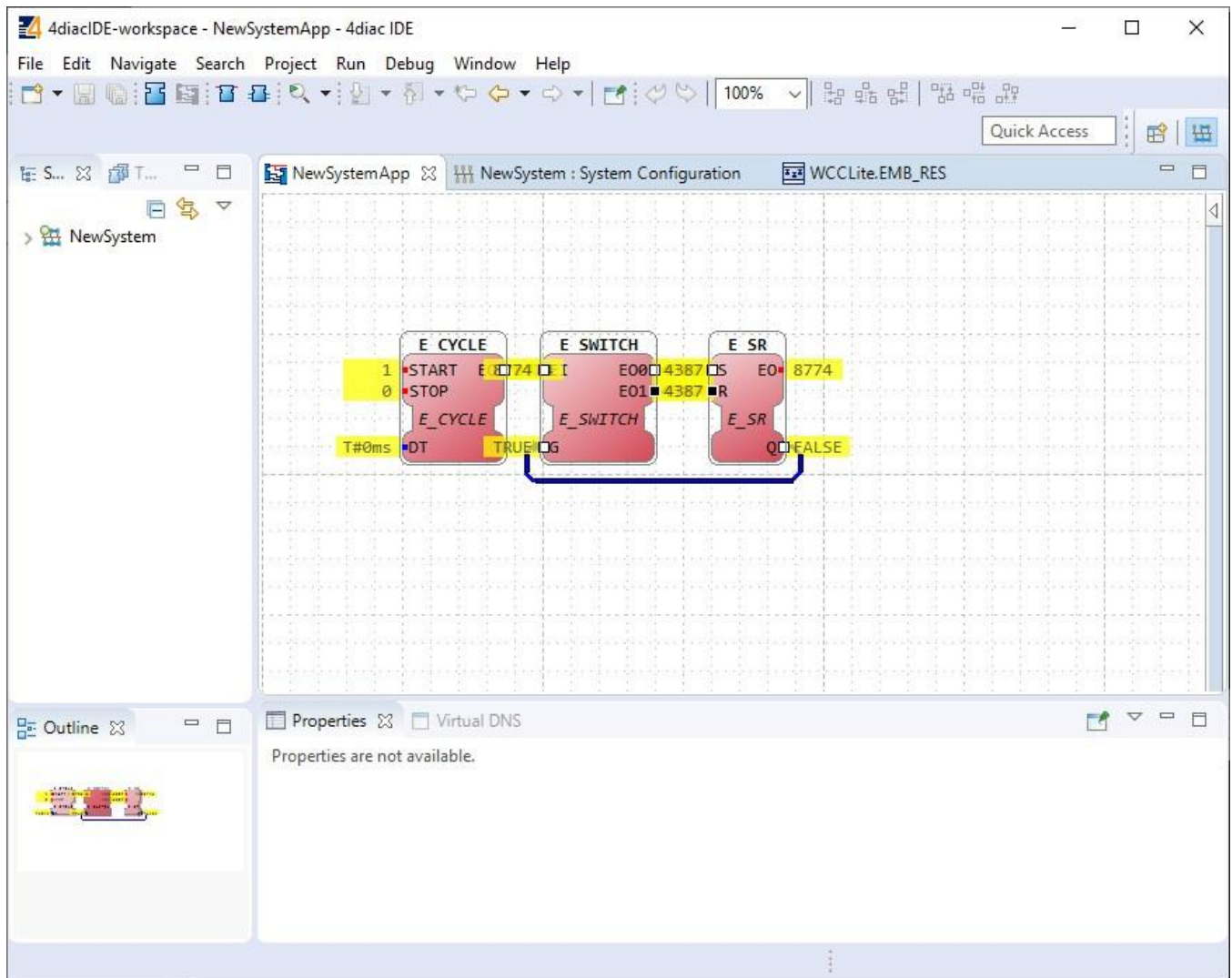
There is a possibility that 4Diac FORTE might not start as a process. It may happen if multiple faults occur and the process has stopped. The process is also programmed to not start if no Excel configuration file is found, therefore a user should make sure that Excel configuration is uploaded and ready 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 below). The application should be deployed to monitor the function blocks 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 accessing WCC Lite+ via a command-line interface. For example, in case of information not being updated one would want to ensure that 4Diac FORTE in WCC Lite+ is not filtering data out but sending it to an internal database (Redis). To run 4Diac FORTE debug from the command-line interface, a user should write forte and press Enter. All possible choices are shown by adding the -h flag. A debugging example would be: '/etc/init.d/forte stop' to stop any forte processes, 'forte -d7 -r -f /etc/forte/yourbootfilename.fboot' to launch the debug. More flags are shown in the Table below. Make sure to stop any running process that could use the address that the 4Diac framework is going to use.

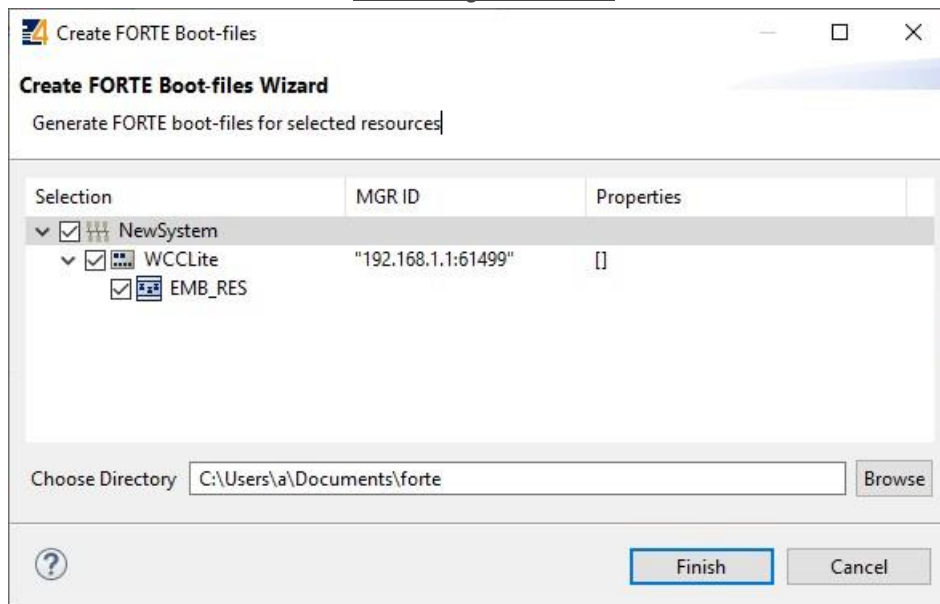
Table. 4Diac FORTE command line debugging options:

```
-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
```

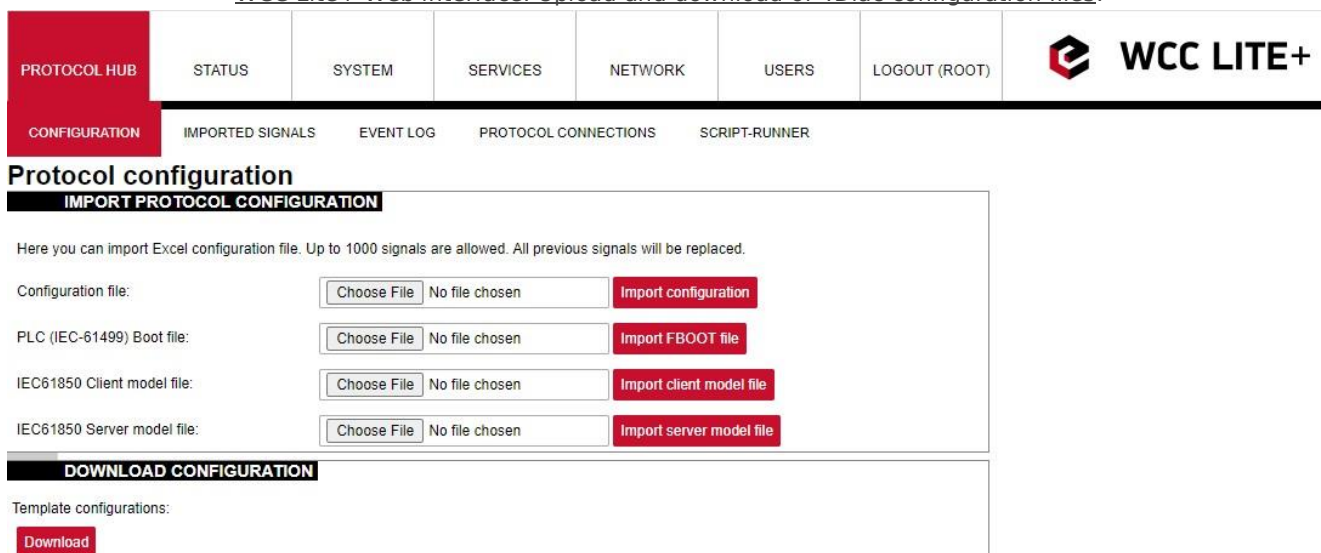
20.6 Generating and uploading FORTE logic file

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

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

Generating FBOOT file:

The upload button for the FORTE file in the web interface can be found in the Protocol Hub tab, Configuration screen (FORTE boot file upload supported for versions v1.4.0 and above). You should see a view in the 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 the *.fboot extension are allowed.

20.7 Distributed control application

IEC 64199 standard introduced requirements for 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:

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

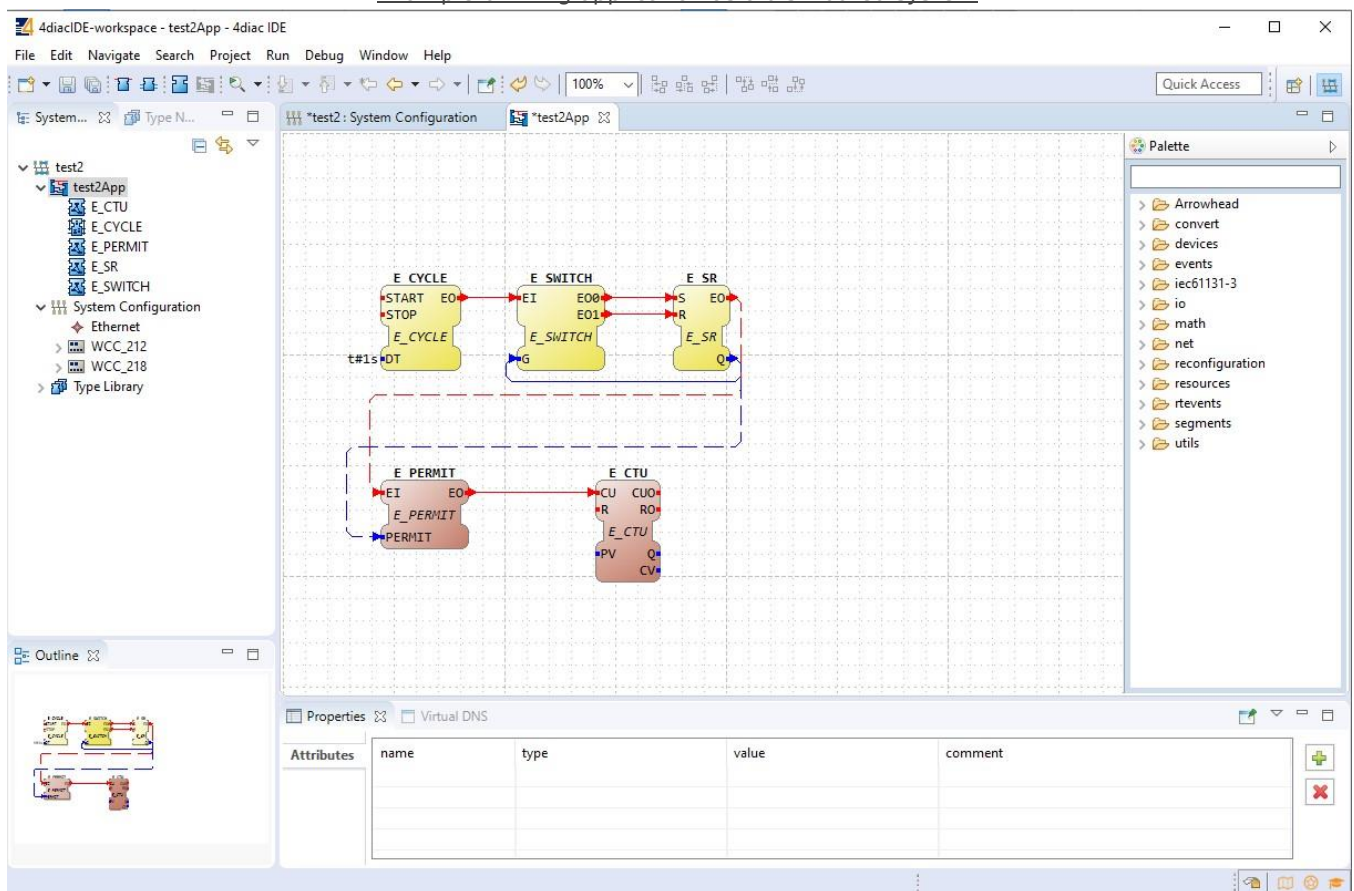
- Publish/Subscribe function blocks (via UDP packets);
- Client/Server function blocks (via TCP packets).

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 the multi-cast option.

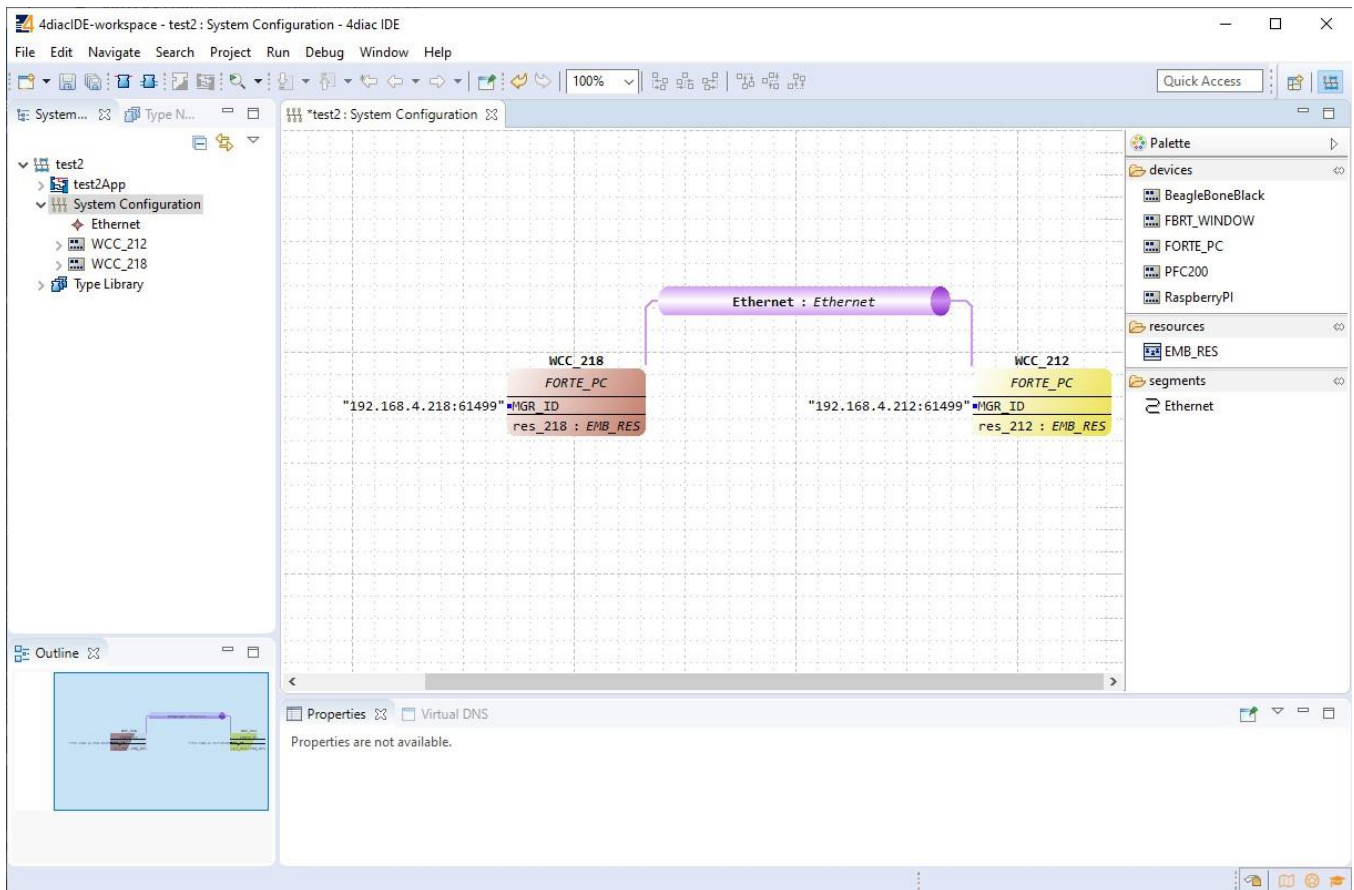
This user manual will only cover setting up point-to-point communication between devices via Publish/Subscribe blocks. For more information on communication between several IEC 61499 devices please check the documentation for Eclipse 4diac framework.

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 the application shown in the 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 a device and its properties by accessing the System Configuration screen as seen below. Yellow function blocks belong to the WCC_212 device which can be accessed through 192.168.4.212 (port number 61499) whereas brown function blocks belong to the WCC_218 device which can be accessed through 192.168.4.218 (port number 61499).

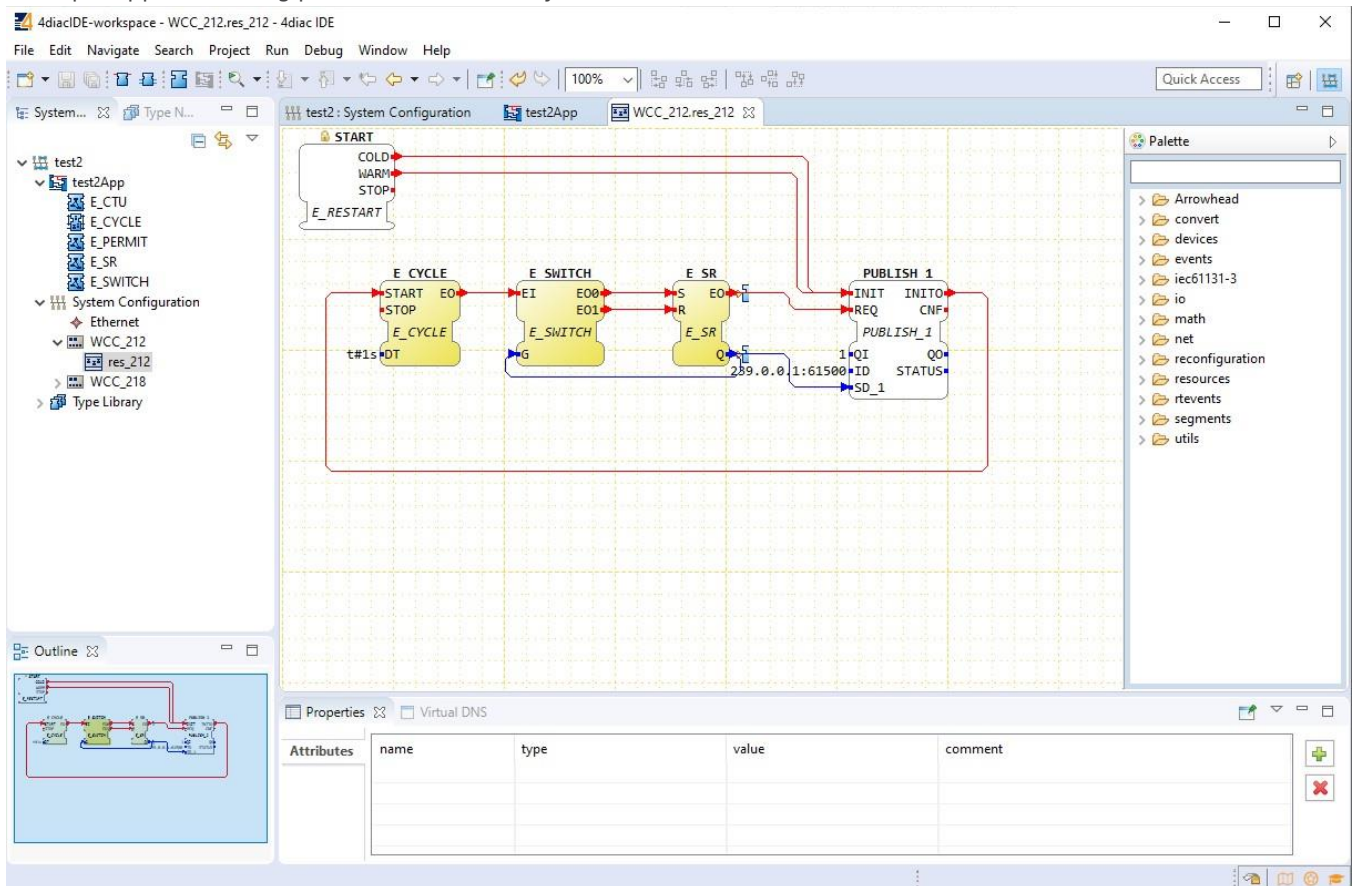
Example blinking application as a distributed system:



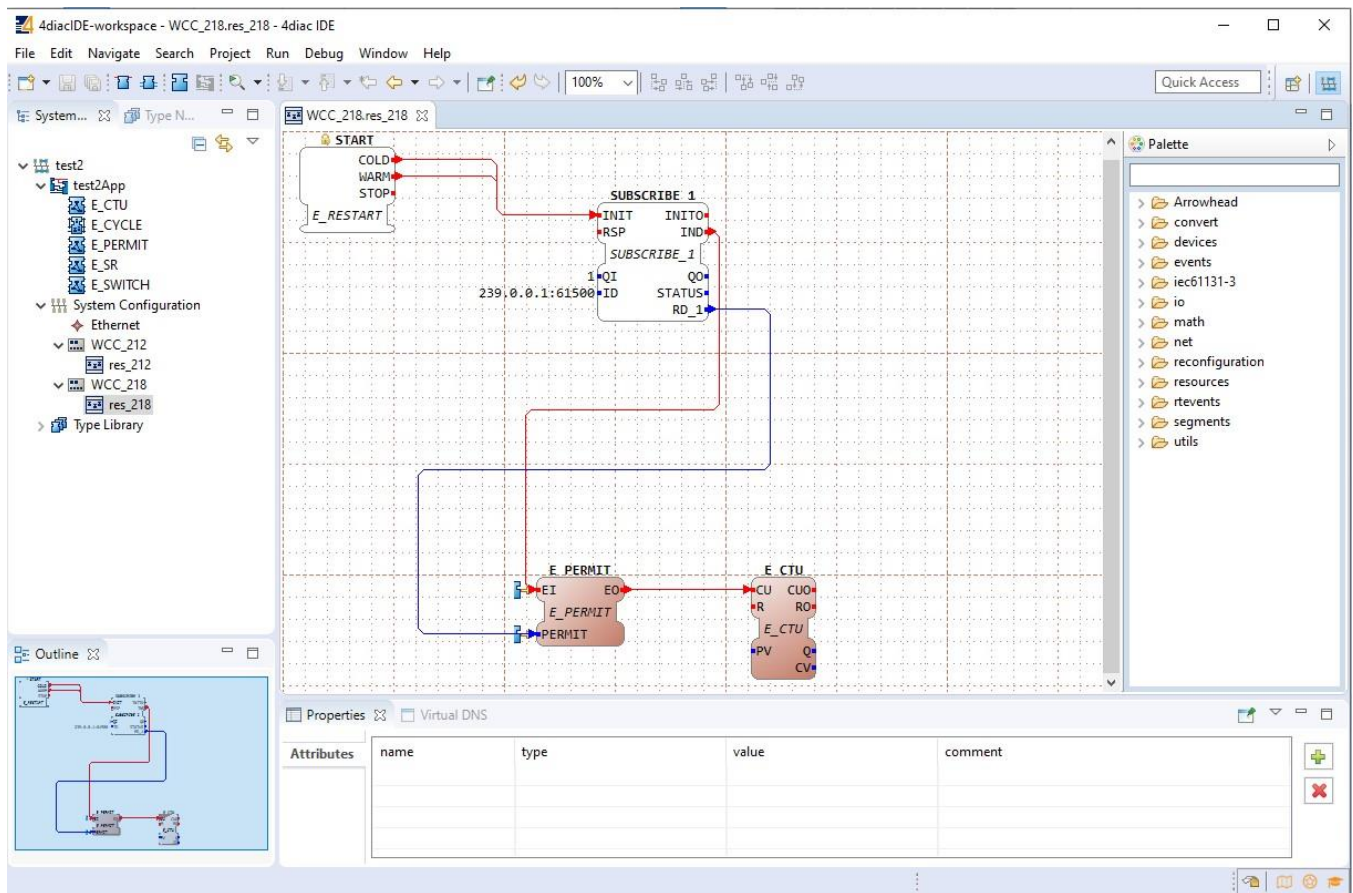
Example system configuration for a distributed system:



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 that are received by a 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, publish/subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP, although its variant, MQTT-SN, is used over other transports such as UDP or Bluetooth. It is designed for connections with remote locations where a small code footprint is required or the network bandwidth is limited.

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

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

With MQTT broker architecture the devices and applications become decoupled and more secure. MQTT might use Transport Layer Security (TLS) encryption with a user name, password-protected connections, and optional certifications that require clients to provide a certificate file that matches the server's. The clients are unaware of each other's 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 an 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 acknowledgment 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 an 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 the 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/Domain name 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
mqtt_qos	integer	MQTT Quality of Service for the message as in standard	No	0	0	2
mqtt_retain	boolean	Selecting if the MQTT broker should retain the last received messages	No	0	0	1
user	string	MQTT user name	Yes			
password	string	MQTT user password	Yes			
auth	string	Selecting if TLS should be used	No		tls	
ca_certificate	string	Certificate authority file for TLS connection	Yes (If auth=tls)			
client_certificate	string	Client certificate file for TLS connection	Yes (If auth=tls)			
client_key	string	The private key that corresponds to the client certificate for TLS connection	Yes (If auth=tls)			
use_last_will	boolean	Selecting if MQTT should use the last will 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 the 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	0	2
last_will_retain	boolean	Selecting if the MQTT broker should retain the last will message	No	0	0	1
client_id	string	User-friendly name for client ID	No			

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

Table. MQTT 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 signal name to be used	Yes			
source_device_alias	string	device_alias of a source device	Yes			
source_signal_alias	string	signal_alias of a source signal	Yes			
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, and quality (where value can be float, integer, or nan; timestamp - Unix timestamp in milliseconds; quality shows if a value is to be considered valid). Quality parts of a string are always equal to 1 except for Redis messages containing invalid (IV), nontopic (NT), and/or overflow (OV) flags.

As mentioned, the MQTT client acts as an adapter between Redis and MQTT, therefore data from the 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 the 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 within direction have their MQTT mappings.

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


Debugging a MQTT protocol

If the configuration for MQTT is set up, a handler for the protocol will start automatically. If the configuration is missing parameters or contains errors, the 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 the MQTT Client 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 `mqtt-client` process and run `mqtt-client` command with respective flags as was shown above.

22 Lua script runner

Introduction

Lua is a powerful, efficient, lightweight scripting language. It has been used in many industrial applications with an emphasis on embedded systems. Lua has a deserved reputation for performance. To claim to be "as fast as Lua" is an aspiration of other scripting languages. Several benchmarks show Lua as the fastest language in the realm of interpreted scripting languages. Lua is fast not only in fine-tuned benchmark programs but in real life too. Substantial fractions of large applications have been written in Lua.

In the WCC Lite+ system, Lua is used for extending the functionality of Excel configuration adding an interface to the existing signal-linking engine. Provided functions enable to recreate of PLC functionality and modify any value with ease.

Overview

Execution types:

Lua runner provides 3 execution modes: interval, date, and signal, which can be specified in **execution_type** e.

Interval: executes provided script based on provided time interval in **execution_parameter**. It uses milliseconds, meaning if a 500 value is provided, then the script is executed every 500 milliseconds.

Date: schedules a script execution based on the provided cron expression in **execution_parameter**. The format consists of 6-7 fields separated by space.

Name	Required	Allowed Values	Allowed Special Characters
Seconds	Y	0-59	, - * /
Minutes	Y	0-59	, - * /
Hours	Y	0-23	, - * /
Day of month	Y	1-31	, - * /
Month	Y	0-11 or JAN-DEC	, - * /
Day of week	Y	1-7 or SUN-SAT	, - * /
Year	N	empty or 1970-2099	, - * /

For example, `0 0 * * * *` will execute the script at every hour mark. There are a lot of online cron expression parsers or generators to convert this expression to a more understandable sentence. <https://crontab.cronhub.io/>

Signal: uses source signal provided in signal sheet to trigger script execution. Another non-Excel signal can be provided in the **execution_parameter** attribute. As an example, if a signal in Excel configuration has an attribute **execute** with value 1, and a source signal specified in **source_device_alias**, **source_signal_alias**, then if a value event happens to the source signal, the script will be executed. More than one signal can have an **execute** attribute. The signal that executes the script can be accessed through the **execution_signal** variable that has a table inside of it with variables: {tag = {device_alias = "script", signal_alias = "tag1"}, value = {value = 60, time = "123456789", attributes = "iv, nt, sb"}}

Additional functions:

To interface with existing signals and extend the available Lua functions some extra functions were added:

get, **set**, **publish**, and **subscribe** functions are used to get the values configured in Excel configuration or to communicate with other scripts.

A signal value in the WCC Lite+ system consists of 3 sub-values: **value**, **time**, and **attributes**. **get** function is used to get all the sub-values or **get_value**, **get_time**, and **get_attributes** to get only one of the sub-values. To execute this command, a signal has to be specified: **get(signal.value1)**. A signal is specified by a string "tag/<device_alias>/<signal_alias>" or a table that is created from an **iterator** parameter in Excel configuration. For example:

```
-- the default iterator is signals, that means there is a table 'signals' generated from excel signals
-- signals["tag1"] = "tag/device_alias/tag1" tag1 would be the signal_alias --
to get the value of this signal:
local variable = get_value(signals.tag1) local variable =
get_value(signals["tag1"]) -- both methods are viable
-- and the 'variable' will have the value of tag1 signal
-- get function will return the value of source signal that was sent to this signal
```

Function **publish** is used to send a value to other signals:

```
publish(signals.tag2, 90) -- this function will send 90 with current time to the signal tag2 publish(signals.tag2,
{value = 60, time = "123456789", attributes = "iv,nt,sb"})
-- above command is used when other sub values are needed to be specified
-- and the signal-linker will send it to another signal
-- if this signal was specified as source signal in another protocol signal
```

To provide different times and attributes to the publish function, a table of these 3 values has to be specified, time can be omitted. An example of the table is returned by the **get** function.

These two functions will be used the most, others are included for communication between different scripts in a more complex system. **set** function sets the value of the signal, without sending an event of change:

```
set(signals.tag3, 50) -- this command will set the value of the signal tag3
-- because it is a set function, other protocols will not see a change
-- and the value will not be accessible -- it is only used with
non excel signals set("signal1", 60) -- now the signal1 tag will
have a value -- and another script will be able to use get to
get this value
local value1 = get_value("signal1") -- value1 will be equal to 60 with current time
```

Function **subscribe** is used to wait (blocks code execution while waiting) for a value and get it. It is used the same as the **get** function (does not have separate functions for sub-values). It should only be used if more complex communication between scripts is needed.

 These functions are equivalent to REDIS functions.

Function **save** saves the specified value to flash memory for use after reboot. The same value sub-values apply to execution.

```
save(signals.tag1, 50) -- this will save a value to tag1
-- after reboot this value will be set to tag1
-- and will be accesible with get(signals.tag1)
-- this function will not set the value tag1 to 50
```

Function **time_ms** returns the current time in milliseconds and in UNIX format.

```
local t = time_ms()
-- t will be equal to 1665389490555
-- if the date right now is Mon Oct 10 2022 08:11:30:555 GMT+0000
```

Function **sleep** is the same as the Lua socket module function `socket.sleep`.

```
sleep(1) -- will wait for 1 second
```

Additional functions for debug information are **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, and **debug**. These functions correspond to levels from 0 to 7. The default level is 4, which means that the function from **emerg** to **warning** will be printed to syslog unless specified differently in file **/etc/lua-runner.conf**.

```

emerg("log message 0")
alert("log message 1")
crit("log message 2") err("log
message 3") warning("log
message 4") notice("log
message 5") info("log message
6") debug("log message 7")

```

⚠ These functions will require significant CPU resources even if the message log level is higher than default and no message is printed.

Web interface

The web interface is used to see what scripts are running and, if there is a script provided, to stop/start a script. After configuring a device with Lua runner as protocol, the script runner protocol hub tab will be populated with devices that were configured:

Script Configuration	Script process	Status	Script File
Device_1	-	Stopped	No Script provided
Device_2	-	Stopped	No Script provided
Device_3	-	Stopped	No Script provided
Device_4	-	Stopped	No Script provided

Then by pressing the Upload Script button, a script will be available to be selected (the name of the script will be changed to match device_alias). When uploaded the script will not be started automatically, pressing start will be necessary.

Script Configuration	Script process	Status	Script File
Device_1	-	Stopped	Device_1.lua
Device_2	-	Stopped	No Script provided
Device_3	-	Stopped	No Script provided
Device_4	-	Stopped	No Script provided

After pressing start, the script will be started, if there are errors it will try to start, but after a few attempts, it will stop.

Script-Runner

LUA SCRIPT INSTANCE CONTROL

Script Configuration	Script process	Status	Script File		
Device_1	12136	Running	Device_1.lua	Upload Script	Stop
Device_2	-	Stopped	No Script provided	Upload Script	Waiting for script
Device_3	-	Stopped	No Script provided	Upload Script	Waiting for script
Device_4	-	Stopped	No Script provided	Upload Script	Waiting for script

SAVED VALUE CLEARING

Clear all saved values

The clear all saved values button is used to clear the memory of saved values. Having a lot of values saved is not healthy for the SD card and faults can happen. Also, the script runner process initialization is slowed down when a lot of saved values are used.

Configuration

Device configuration parameters:

Parameter	Type	Description	Required	Default value (when not specified)	Range	
					Min	Max
name	string	User-friendly name for a device	Yes			
device_alias	string	Alphanumeric string to identify a device	Yes			
description	string	Description of a device	No			
enable	boolean	Enabling/disabling of a device	No	1	0	1
protocol	string	Protocol to be used	Yes		lua runner	
execution_type	string	Execution type to be used	Yes		interval, date, signal	
execution_parameter	int, string	Parameters for execution	Yes	NULL	interval time in ms, date in cron format, additional signal	

queue_max	int	Maximum execution queue jobs	No	3	0 to disable the queue
error_limit	int	Error limit before stopping	No	3	0 to disable
keep_alive_time_ms	int	Time to keep the script alive in milliseconds	No	600000	0 to disable

Signals configuration parameters:

Parameter	Type	Description	Required	Default value (when not specified)	Range	
					Min	Max
signal_name	string	User-friendly name for a signal	Yes			
device_alias	string	Alphanumeric string to identify a device	Yes		Must match device_alias in the device sheet	
signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes			
iterator	string	Lua table name to which signal is added	No	signals		
default_value	string	Default value for a signal	No			
execute	int	Enable signal update trigger to execute script (only available for signal execution mode)	No	0	0	1

Debugging the script runner service

If the configuration for the script runner is set up, the process will start automatically. If the configuration/script is missing or contains errors, the process will not start. It is done intentionally to decrease unnecessary memory usage.

Script runner runs a service called **lua-runner**. If the script doesn't start or does not work correctly, a user can launch a debug session from the command interface and find out what problem is causing it to not work. To launch a debugging session, a user should stop the script from the web interface and run the **lua-runner** command with respective flags and configurations as in the table given below.

Procedure for lua-runner service debugging:

- **Step 1:** The script must be stopped through a web interface.
- **Step 2:** After the script is stopped it must be started with the preferred configuration file (JSON files found in /etc/lua-runner, and the name corresponds to device_alias) and a debug level 7:
lua-runner -c /etc/lua-runner/device_alias.json -d7 -e.
- **Step 3:** Once the problem is diagnosed normal operations can be resumed by starting the script through a web interface.

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