RTU usage

Remote Terminal Unit usage. With configurations examples and Tips and Tricks.

- Protocols - configuration
  - IEC 61850
  - IEC 60870-5
  - IEC 60870-5-101
  - IEC 60870-5-103
  - IEC 60870-5-104
  - DNP 3.0
  - DLMS/COSEM
  - IEC 62056-21
  - Modbus
  - MQTT
  - Data Export
  - Event log

Protocols - configuration

You will find all information about configurations of protocols on WCCLite.

IEC 61850

Introduction

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

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

IEC 61850 Server

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

Commands

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

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

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

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

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

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

Configuring datapoints

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

IEC 61850 Server parameters for Devices tab:

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	No
description	string	Description of a device	No
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used ("**IEC 61850 Server**")	Yes
tls	string	Selecting if TLS should be used	No
bind_address	string (IP address format)	IP address of and interface to use with server (0.0.0.0 for any interface)	Yes
host	string (IP address format)	IP address list of allowed IPs (separated with spaces)	Yes
port	integer	TCP communication port	Yes
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)

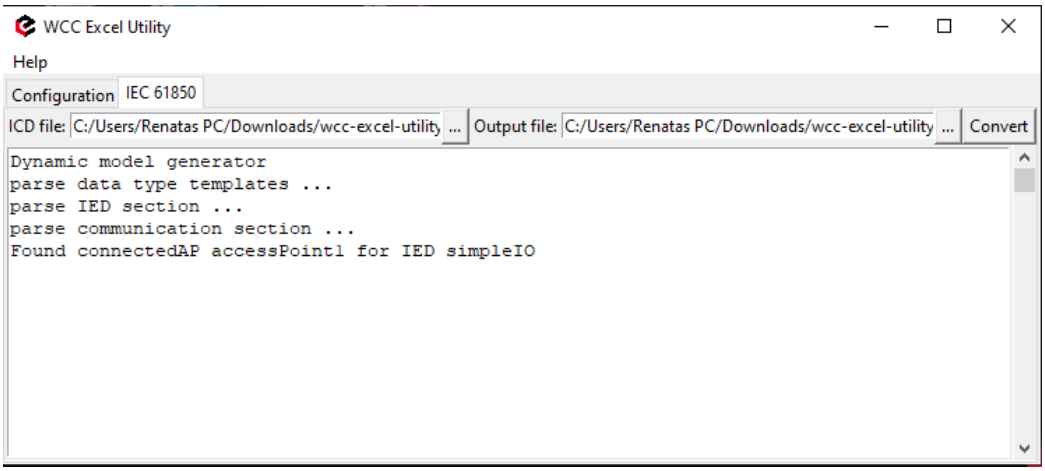
tls_private_key	string	File consisting of private key for TLS connection	Yes (for TLS)
event_history_size	integer	Event log size	No
ied_name	string	Name of an Intelligent Electronic Device	Yes
authorization	string	Authorization type (”password”)	No
password	string	Authorization password for server device	No
model_filename	string	Filename of data model uploaded to WCC (with or without file extension)	Yes
edition	string	Which IEC61850 edition to use: ”1”, ”2”, ”2.1” (Default: 2)	No
command_ack_timeout_ms	integer	Timeframe (ms) in which enhanced-security commands must be acknowledged (Default: 3000)	No
report_buffered_size	integer	Report control blocks buffer size in bytes (Default: 65536)	No
report_unbuffered_size	integer	Unbuffered report control blocks buffer size in bytes (Default: 65513)	No

IEC 61850 Server parameters for Signals tab:

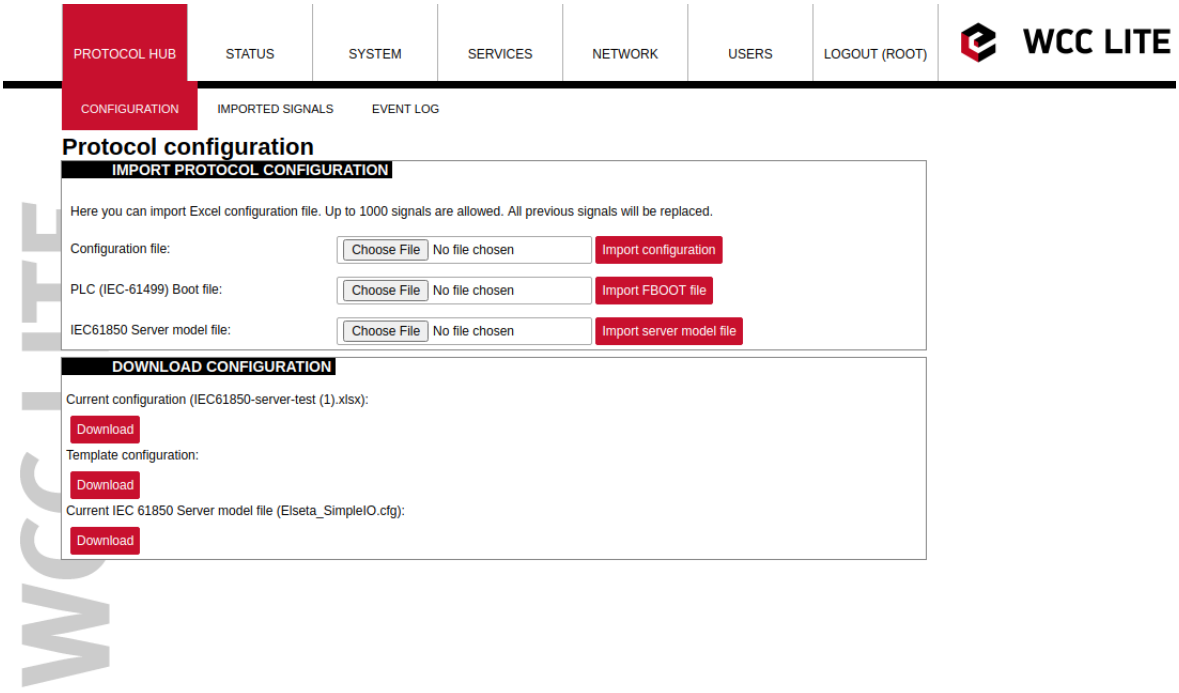
Parameter	Type	Description	Mandatory
signal_name	string	User-friendly signal name	Yes
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique alphanumeric name of the signal to be used	Yes
enable	boolean	Enabling/disabling of an individual signal	Yes
number_type	string	Number format type	Yes
ld_instance	string	Instance of a logical device	Yes
ln_class	string	Logical node class type	Yes
ln_instance	integer	Instance of a logical node	No
ln_prefix	string	Prefix of logical node string	No
cdc	string	Common Data Class (CDC) name	Yes
data_object	string	Name of data object in dataset	Yes
da_value	string	Name of a data attribute value node	Yes
da_time	string	Name of a data attribute time node	No
da_quality	string	Name of a data attribute quality node	No
da_fc	string	Functional constrain for data object	Yes
control_model	string	Model of output control	Yes (for commands)

Converting and uploading data model

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



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



Debugging a IEC 61850 server application

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

> ⚠ If IEC 61850 Server does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly.

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

`iec61850-server`

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

IEC 61850 Client

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

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

Acquiring data via report control blocks

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

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

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

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

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

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

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

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

IEC 61850 Client supported data attributes:

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

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

Controlling remote equipment via commands

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

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

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

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

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

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

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

Configuring datapoints

To use IEC 61850 Client in WCC Lite, it has to be configured via an Excel configuration. This configuration contains two Excel sheets where parameters have to be filled in - Devices and Signals tables.

Table IEC 61850 Client parameters for Devices tab:

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	No
description	string	Description of a device	No
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used ("**IEC 61850 Client**")	Yes
tls	string	Selecting if TLS should be used	Yes (for TLS)
host	string (IP address format)	IP address of server device	Yes
port	integer	TCP communication port (Default:102)	Yes
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	File consisting of private key for TLS connection	Yes (for TLS)
event_history_size	integer	Event log size	No
ied_name	string	Name of an Intelligent Electronic Device	Yes
authorization	string	Authorization type ("password")	No
password	string	Authorization password for server device	No
originator	string	Origin identificator for device	No

Table IEC 61850 Client parameters for Signals tab:

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly signal name	Yes
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique alphanumeric name of the signal to be used	Yes
enable	boolean	Enabling/disabling of an individual signal	Yes
number_type	string	Number format type	Yes
ld_instance	string	Instance of a logical device	Yes
ln_class	string	Logical node class type	Yes
ln_instance	integer	Instance of a logical node	No
ln_prefix	string	Prefix of logical node string	No
cdc	string	Common Data Class (CDC) name	Yes
data_object	string	Name of data object in dataset	Yes
da_value	string	Name of a data attribute value node	Yes
da_fc	string	Functional constrain for data object	Yes
control_model	string	Model of output control	No
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

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

IEC 61850 Client has an additional signal which can be configured to show communication status. It is used to indicate if the server device has disconnected from client (WCC Lite). To configure such signal, two columns should be filled with particular values. To a newly created additional signal one should make `job_todo` equal to device_status and `tag_job_todo` equal to communication_status. Communication error status is set after a disconnection of a server device.

Debugging a IEC 61850 Client application

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

IEC 61850 Client command line debugging options

`iec61850-client`

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

If IEC 61850 Client does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly.

To launch a debugging session, a user should stop `iec61850-client` process and run `iec61850-client` command with respective flags as was shown above.

IEC 60870-5

Introduction

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

- IEC 60870-5-1 Transmission Frame Formats
- IEC 60870-5-2 Data Link Transmission Services
- IEC 60870-5-3 General Structure of Application Data
- IEC 60870-5-4 Definition and Coding of Information Elements
- IEC 60870-5-5 Basic Application Functions
- IEC 60870-5-6 Guidelines for conformance testing for the IEC 60870-5 companion standards
- IEC TS 60870-5-7 Security extensions to IEC 60870-5-101 and IEC 60870-5-104 protocols (applying IEC 62351)

The IEC Technical Committee 57 has also generated companion standards:

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

IEC 60870-5-101/102/103/104 are companion standards generated for basic telecontrol tasks, transmission of integrated totals, data exchange from protection equipment & network access of IEC101 respectively.

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

IEC 60870-5-101

IEC 60870-5-103

IEC 60870-5-103

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 serial link (according EIA RS-485). Its full functionality list can be found in a IEC 60870-5-103 PID Interoperability List.

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

Devices parameters table:

Parameter	Type	Description	Mandetory
name	string	User-friendly name for a device	No
description	string	Description of a device	No
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Must be set "IEC 60870-5-103 master"	Yes
device	string	Communication port (PORT1 or PORT2)	Yes
baudrate	integer	Communication speed, bauds/s	Yes
databits	integer	Data bit count for communication	Yes
stopbits	integer	Stop bit count for communication	Yes
parity	string	Communication parity option (none/even/odd)	Yes
flowcontrol	string	Communication device’s flow control option. Available options (case insensitive) - ”no” or ”none”, ”sw” or ”software”, ”hw” or ”hardware”.	No
link address	integer	Address of device (link)	Yes
asdu_address	integer	Application Service Data Unit adress	Yes
time_sync_interval_sec	integer	Time frame between Time Synchronization requests in seconds	Yes
gi_interval_sec	integer	Time frame between General Interrogation requests in seconds	Yes
poll_interval_ms	integer	Polling interval in milliseconds. Time frame between two telegrams from master. Default - 100	No
event_history_size	integer	Maximum count of events in event log. Default - 0	No
poll_timeout_ms	integer	Timeout of waiting for incoming request	No
serial_delay	integer	Communication device’s serial delay in milliseconds. Time frame in which master station is not TX’ing after last RX byte. Default: 50	No

poll_retry_count	integer	Number of retries of failed requests before announcing that device is in Error state	No

Signals parameters table:

Parameter	Type	Description	Mandetory
signal_name	string	User-friendly name of a signal	No
device_alias	string	Device alias from a Devices tab	Yes
signal_alias	string	Unique signal name to be used	Yes
source_device_alias	string	device_alias of a source device	For commands
source_signal_alias	string	signal_alias of a source signal	For commands
enable	boolean	Enabling/disabling of a signal	
log_size	integer	Space for signal in event log	
gi	boolean	Including/excluding signal from General Interrogation. Default - 0 (exclude)	
common_address	integer	Address of a device	
function	integer	Function number	
info_address	integer	Information address	
info_number	integer	Information number	
data_type	integer	ASDU type identificator	
fleeting	boolean	Mark signal as fleeting type. Fleeting signals have go to DPI::OFF after defined time	
normalise_time_ms	integer	Time in milliseconds between station receiving DPI::ON and automatically switching to DPI::OFF. Default - 100.	If fleeting is used

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

Debugging a IEC 60870-5-103 Master aplication

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

If IEC 60870-5-103 does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command-line interface and find out why 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. There is two posibilities to run debugging mode:

- use WCC Utility tab Debug (introduced in version v.1.3 with WCCOS firmware version v1.5.0);
- use console to run command inside the device;

bellow described parameters for debugging is accesbile over console (SSH).

```
iec103-master parameters:
```

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

```
-V [--version] Show package version
```

```
-d< debug level > Set debugugging level
```

```
-c [--config] Config path
```

```
-r [--raw] Show raw telegram data
```

```
-f [--frame] Show frame data
```

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

IEC 60870-5-104

Introduction

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

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

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

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

IEC 60870-5-104 Master

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 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 cache, not sent and not confirmed by SCADA is transfered to new connection.

DNP 3.0

Introduction

Distributed Network Protocol 3 (**DNP3**) 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. Usage in other industries is not common. It was developed for communications between various types of data acquisition and control equipment. It plays a crucial role in SCADA systems, where it is used by SCADA Master Stations (a.k.a. Control Centers), Remote Terminal Units (RTUs), and Intelligent Electronic Devices (IEDs). It is primarily used for communications between a master station and RTUs or IEDs. ICCP, the Inter-Control Center Communications Protocol (a part of IEC 60870-6), is used for inter-master station communications.

Source: https://en.wikipedia.org/wiki/DNP3

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

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

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

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

DNP 3.0 Master

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

Table. Default command variations:

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

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. For more information check section Excel configuration.

Table. Default signal variations:

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

DLMS/COSEM

Introduction

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

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

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

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

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

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

DLMS Master

Overview

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

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

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

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

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

Configuration

Devices section

serialnumber, server_address and id define the meter addressing parameters. Either serialnumber (meter serial number) or a combination of server_address (physical server address) and id (logical server address) is used. If a serial number is provided, physical and logical server addresses are ignored.

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

master_address defines the client address. This usually depends on the authentication used. Most meters support 16 for no authentication.
type defines the object referencing. SN should be used for short name referencing and LN for logical name referencing.

mode defines the 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 connection is initiated only via IP.

Device configuration parameters for DLMS meters acquisition:

Parameter	Description	Type	Default value	Example
serialnumber	Meter serial number	unsigned long	0	1122334455
slave_address	Meter physical server address	unsigned long	0	1600
id	Meter logical server address	unsigned long	0	1
master_address	Client address	integer	16	1
type	Meter object referencing: SN - short referencing, LN - logical referencing	string	SN	LN
mode	Comms mode: IEC handshake (for serial only), DLMS-HDLC or DLMS-WRAPPER (for IP)	string	DLMS-HDLC	IEC
timeout_ms	Timeout in milliseconds	integer	2500	1500
auth	Authentication: None, Low, High, HighMd5, HighSha1, HighSha256, HighGmac HighEcdsa	string	None	Low
password	Password for authentication	string	n/a	MyPass123
ip	IP address	string	n/a	192.168.1.1
port	TCP port	integer	n/a	4059

Signals section

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

DLMS configuration parameters creating signals:

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

IEC 62056-21

Introduction

IEC 61107 or currently IEC 62056-21, was an international standard for a computerprotocol to read utility meters. It is designed to operate over any media, including the Internet. A meter sends ASCII (in modes A..D) or HDLC (mode E) data to a nearby hand-held unit (HHU) using a serial port. The physical media are usually either modulated light, sent with an LED and received with a photodiode, or a pair of wires, usually modulated by a 20mAcurrent 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.request will provide the HHU with the desired response. If the parameter required is in the low-security group, a password authentication of the HHU is required before information can be read.

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

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

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

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

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

Overview

The IEC 62056-21 standard defines protocol specifications for local meter data exchange.
Data is read out via serial port in modes A, B or C. The default initial serial port settings are 300 bps 7E1, as per standard, but can be user configured.

The driver implementation additionally allows for communication via TCP/IP, which is not described in the standard. In this case, baud rate acknowledgement is allowed however actual switchover between baud rates is not possible.

Mode A: data is requested and read out at the configured baud rate.

Mode B: data is requested at the configured baud rate and mutually switched to the baud rate proposed by the meter. Baud rate confirmation is absent.

Mode C: data is requested at the configured baud rate, new baud rate is proposed by the meter and, if acknowledged, data is read out at the proposed baud rate.

Currently data readout is supported in modes A, B and C.

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

Configuration

Device section

The serialnumber defines the serial number of the meter. 0 (zero) will result in a ’/?!’ handshake string and may cause issues if more than one meter is wired to the serial port.

The baudrate defines the initial connection baud rate. In modes B and C this will be switched to what ever baud rate is proposed by the meter.

The meter_model defines the meter profile. This is reserved for future use and should be set to 1. type defines the

connection mode. Modes A, B and C are supported.

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

IEC 62056-21 device configuration parameters:

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

Signals section

The tag_job defines the tag job. This is not used for this protocol and should be set to ”’1”. tag_job_todo defines the job sub-job. This field should contain the exact representation of the OBIS code as it is configured in the meter. E.g. if the parameter of interest is represented as

”1.8.0*24(0147238.4*kWh)”, the value of the configuration field should be ”1.8.0*24” (excluding quotation marks).

IEC 62056-21 tags configuration parameters:

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

ℹ 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.

Modbus

Introduction

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

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

WCC Lite supports both Modbus Master and Slave protocols. One can select between transmission over TCP/IP or serial connection (RS-485/RS232). Bytes to transmit can either be encoded according to both RTU and ASCII parts of standard.

Modbus Master

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

Configuring datapoints

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

Table of Modbus Master parameters for Devices tab

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	No
description	string	Description of a device	No
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used (”Modbus RTU” or ”Modbus TCP”)	Yes
host (ip)	string (IP address format)	Enabling/disabling of a device	Yes
port	integer	TCP communication port (Default: 502)	No (for TCP)
bind_address	string	IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”)	Yes (for TCP)
id	integer	Modbus Slave ID	Yes
ascii	boolean	Modbus ASCII mode (when Modbus RTU selected). (Default: 0)	No (for RTU/ASCII)
timeout_ms (timeout)	integer	Response timeout in milliseconds	Yes. ”timeout” parameter has a higher precedence
device	string	Communication port (”PORT1”/”PORT2”)	Yes (for RTU/ASCII)
baudrate	integer	Communication speed, baud/s	Yes (for RTU/ASCII)
databits	integer	Data bit count for communication	Yes (for RTU/ASCII)
stopbits	integer	Stop bit count for communication	Yes (for RTU/ASCII)
parity	string	Communication parity option (”none”/”even”/”odd”)	Yes (for RTU/ASCII)

flowcontrol	string	Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued	Yes (for RTU/ASCII)
scan_rate_ms	integer	If provided and positive - all jobs will have similar scan rate - all reads and writes will be executed within this timeframe (parameter scan_rate_ms in Signals tab will be ignored)	No
pool_retry_count	integer	Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued	No
poll_delay_ms	integer	RS485 delay between read and write operations in milliseconds (Default: 50)	No (for RTU/ASCII). ”serial_delay” parameter has a higher precedence
event_history_size	integer	Event log size	No
modbus_multi_write	boolean	Use 15/16 functions to write 1 register/coil (Default: 0)	No
comm_restart_delay	integer	Time delay between disconnecting from slave device and restarting connection (in milliseconds) (Default: 500)	No (for TCP)

Table. Modbus Master parameters for Signals tab

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

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

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

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

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

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

`tag_job_todo` equal to communication_status. Communication error status is set when a predefined count of messages (three by default, defined in `poll_retry_count` column) fail to be received or are considered invalid.

Debugging a Modbus Master application

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

Modbus Master command line debugging options

`modbus-master`

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

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

Modbus Slave

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

Configuring datapoints

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

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

Table. Modbus Slave parameters for Devices tab

Parameter	Type	Description	Mandatory
name	string	User-friendly name for a device	No
description	string	Description of a device	No
device_alias	string	Alphanumeric string to identify a device	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Protocol to be used (”Modbus serial Slave” or ”Modbus TCP Slave”)	Yes
host	string (IP address format)	Space separated host IP addresses of master devices	Yes (for TCP)
port	integer	TCP communication port (Default: 502)	No (for TCP)
bind_address	string	IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”)	Yes (for TCP)
mode	string	Choosing between RTU (”rtu”) and ASCII (”ascii”) modes	No (for RTU/ASCII)
device	string	Communication port (”PORT1”/”PORT2”)	Yes (for RTU/ASCII)
baudrate	integer	Communication speed, baud/s	Yes (for RTU/ASCII)
databits	integer	Data bit count for communication	Yes (for RTU/ASCII)

stopbits	integer	Stop bit count for communication	Yes (for RTU/ASCII)
parity	string	Communication parity option (”none”/”even”/”odd”)	Yes (for RTU/ASCII)
flowcontrol	string	Communication device’s flow control option. Available options (case insensitive) - ”no” or ”none”, ”sw” or ”software”, ”hw” or ”hardware”.	No (for RTU/ASCII)

Table. Modbus Slave parameters for Signals tab

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly signal name	Yes
device_alias	string	Alphanumeric string to identify a device	Yes
signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes
enable	boolean	Enabling/disabling of an individual signal	Yes
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, Yes etc.)	Yes
common_address	integer	Address of a device	Yes
function	integer	Modbus function number	Yes
info_address	integer	Register address	Yes
size	integer	Register/Coil size	Yes

Mapping values to registers

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

To find out how many register should be used for storing a values, how values can have their values swapped, a user should consult a section `number_type` (18.2.4).

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

Debugging a Modbus Slave application

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

Modbus Slave command line debugging options

`modbus-slave`

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

⚠ If Modbus Slave does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly.

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

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, managing and controlling devices.

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

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

The main advantages of MQTT broker are:

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

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

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

Using WCC Lite as MQTT Client

MQTT serves as an alternative for protocols conforming to IEC standards, for example, to send data to a cloud infrastructure that supports MQTT instead of IEC-60870-5-104.

WCC Lite supports MQTT messaging compatible with MQTT v3.1 standard (starting from version**v1.4.0**). Such messaging is possible via mapping of Redis and MQTT data therefore data can be transmitted from any protocol that is supported by WCC Lite.

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

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

Table. MQTT parameters for Devices tab:

Parameter	Type	Description	Mandatory
name	string	User-friendly device name	No
device_alias	string	Device alias to used in configuration	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Selection of protocol (MQTT)	Yes
host	string	MQTT broker IP address selection	Yes
port	integer	MQTT broker port selection (Default: 1883)	No

enable_threshold	boolean	A parameter to determine if identical values should not be sent multiple times in a row. (Default: true)	No
gi_interval_sec	integer	Parameter to determine how frequently should all values be sent at once. Disabled if equal to 0. (Default: 0)	No
mqtt_qos	integer	MQTT Quality of Service for message as in standard (Default: 0)	No
mqtt_retain	boolean	Selecting if MQTT broker should retain last received messages (Default: False)	No
user	string	MQTT user name	Yes
password	string	MQTT user password	Yes
use_last_will	boolean	Selecting if MQTT should use Last Will and Testament functionality (Default: False)	No
last_will_topic	string	Topic to which an MQTT message would be sent if the device abruptly disconnected message broker	Yes (If use_last_will=True)
last_will_message	string	Message to be sent over MQTT if the device abruptly disconnected message broker	No
last_will_qos	integer	MQTT Quality of Service selection as in standard (Default: 0)	No
last_will_retain	boolean	Selecting if MQTT broker should retain last will message (Default: False)	No

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

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

Table. MQTT parameters for Signals tab:

Parameter	Type	Description	Mandatory
signal_name	string	User-friendly signal name	No
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	Yes
topic	string	Topic name to override the value built by default	No

MQTT data format

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

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

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

Debugging a MQTT protocol

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

MQTT Client command line debugging options

`mqtt-client`

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

⚠ If MQTT Client does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly.

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

Data Export

General

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

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

Overview

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

Using WCC Lite for data export

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

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

Parameter	Type	Description	Mandatory
name	string	User-friendly device name	
device_alias	string	Device alias to be used in configuration	Yes
enable	boolean	Enabling/disabling of a device	Yes
protocol	string	Selection of protocol (Data Export)	Yes
timeout	string	Time frame during which transmission to remote server has to be completed (**5 seconds**)	
type	string	Selection of file format (**”csv-simple”**)	
host	string	A URL of remote server where files should be sent	Yes
upload_rate_sec	integer	Frequency of generated file uploads (**60 seconds**)	
records_buffer_size	integer	A maximum amount of data change entries to hold before initiating logging mechanism (**100**)	
logging_period_sec	integer	Describes how frequently data buffer of records_buffer_size is saved to file	
log_folder	string	A folder in WCC Lite file system to save generated file (**”/var/cache/data-export”**)	
timestamp	string	Selection of time format (”**unixtimestamp**”, ”iso8601”)	
compress	string	Selection of file compression mechanism (”**none**”, ”gz”, ”tar.gz”)	

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

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

• Signals for data export service are not seen in the*Imported Signals* tab;
• Signals for data export service are configured in different Excel sheet called DataExport

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

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

Parameter	Type	Description	Mandatory
device_alias	string	Device alias to be used in configuration Yes	Yes
device_name	string	User friendly device name as in Device sheet	Yes
tag_name	string	User friendly signal name	Yes
source_device_alias	string	device_alias of a source device	Yes
source_signal_alias	string	source_alias of a source signa	Yes
enable	boolean	Enabling/disabling of a measurement to be transmitted and logged	Yes
attribute	string	Additional attribute to be attached to a signal	

Debugging data export service

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

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

bellow described parameters for debugging is accesbile over console (SSH).

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

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

`-V [ –version ] Show version`

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

`-R [ –readyfile] Ready notification file`

`-p [ –http ] Show HTTP messages`

`-r [ –redis ] Show Redis output`

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

Host URL format rules

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

• *Protocol* - FTP or HTTP (encrypted and encrypted);

• *URL address* - both resolved and non-resolved;

• *Authentication* - pair of user and/or password;

• *Port* - useful when non-standard value is used;

• *Endpoint* - a place in server to which a call is made

The format for host parameter can be summarized as:

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

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

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

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

Format of exported data

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

⚠ As of software version 1.5.0, data format called ***csv-simple*** is supported.

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

```
### DUID:318110353
# device name, tag name, value , quality , timestamp , attribute

inv2 ,LAN0 Total RX,31.579,1,2020-10-13T17:51:32.718 , inv2 ,RAM usage,44.33,1,2020-10-13T17:51:32.473 ,
inv2 ,SMS sent ,12,1,2009-01-06T04:20:45.421 ,Pb
inv2 ,SMS sent ,12,1,2009-01-06T04:20:45.431 ,
```

Event log

General

Event log is a service that logs selected signal value changes to the web interface and archives them.

Event log functionality is available since firmware version v1.5.4, of WCC Lite.

Configuring Event log

To configure WCC Lite to use Event log user must mark specific signals with parameter **log** and value **1**. For older firmware, signals that have **log_size 1** or greater will also work.

Events log parameters for Signals tab table:

Parameter	Type	Description	Mandatory
log	boolean	Enabling/disabling of event log	Yes

signal_name	device_alias	signal_alias	source_device_alias	source_signal_alias	enable	tag_type	log	job_todo	tag_job_todo	number_type
Sample data #1	Sample device	sample_tag1			1	Normal	1	00|00|00	00|00|00|00|01	UNSIGNED16
Sample data #2	Sample device	sample_tag2			1	Normal	1	00|00|00	00|00|00|00|02	UNSIGNED16

Web interface

To access **event log** you need to configure your WCC Lite to log your specific values. After configuring it, you can see the values in the **Event log** tab. These values only appear if the value has changed.

There you can filter and sort your data by:

- Device name
- Signal alias
- Signal name
- Value
- Date

Also, you can download the full archive of events. It will generate a **.csv** file with all the same information which is displayed.

CONFIGURATION | IMPORTED SIGNALS | EVENT LOG | PROTOCOL CONNECTIONS

DEVICE EVENTS

Auto refresh ☑ Number of items: 50

Device	Signal alias	Signal name	Value	Timestamp
wcc	lan0_tx	LAN0 Total TX	2.076000	2021-07-16 17:22:51
wcc	lan0_rx	LAN0 Total RX	0.511000	2021-07-16 17:22:51
wcc	ram_usage	RAM usage	38.170000	2021-07-16 17:22:51
wcc	cpu_usage	CPU usage	100.000000	2021-07-16 17:22:51
wcc	lan0_tx	LAN0 Total TX	2.069000	2021-07-16 17:22:41
wcc	lan0_rx	LAN0 Total RX	0.508000	2021-07-16 17:22:41
wcc	ram_usage	RAM usage	36.210000	2021-07-16 17:22:41
wcc	cpu_usage	CPU usage	96.000000	2021-07-16 17:22:41
wcc	lan0_tx	LAN0 Total TX	2.063000	2021-07-16 17:22:31
wcc	lan0_rx	LAN0 Total RX	0.507000	2021-07-16 17:22:31
wcc	ram_usage	RAM usage	35.990000	2021-07-16 17:22:31
wcc	cpu_usage	CPU usage	30.000000	2021-07-16 17:22:31

Download Device Events .zip file:

Download