YDOC.biz

ydocTerminal Manual

Title : ydocTerminal manual **Date** : Jan 2024

Version : 3.25

Author : Your Data Our Care

TABLE OF CONTENTS

1 INTRODUCTION

YDOC low power data loggers can be configured with a built-in Terminal application. A terminal application is not that fancy as a built-in Webserver, but a Webserver does not make any sense as YDOC low power data loggers are, to minimize power consumption, in deep sleep most of the time, not connected to the Internet and therefore not reachable by your browser.

You can use ydocTerminal to hands-on configure an YDOC logger locally by USB, wirelessly by Bluetooth LE or remotely by TCP.

You can also use any other serial terminal emulator instead (e.g. Hyperterminal). We however recommend to use ydocTerminal as it supports secure encrypted remote configuration and has some handy YDOC aware features.

1.1 Features

- Local terminal emulation over serial COM-port (data logger USB-port is recognized as virtual COM-port)
- Wireless terminal emulation over Bluetooth LE (When data logger is equipped with BLE)
- Remote secure terminal emulation over TCP (When data logger is equipped with WiFi or modem)
- yModem protocol for hands-on file transfer (e.g. configuration and firmware files).
- MQTT(S) protocol for off-line remote firmware upgrade, configuration update and to schedule remote hands-on connections over a secure TCP-tunnel.
- FTP(S) to schedule are remote hands-on connections over a secure TCP-tunnel.
- HTTP(S) to schedule are remote hands-on connections over a secure TCP-tunnel.
- Define function keys for often used serial commands.
- Auto terminal window refresh when a data logger outputs a new page or device-menu.
- Device navigation by clicking displayed command indicators, e.g.`<Ctrl-A><Shift-M><Ctrl-D>, <Ctrl-Z>, <Esc>`
- Device-menu navigation by clicking menu-option indicators between `[ ]` brackets.
- Select configuration options by clicking multiple choice indicators `(0=x, 1=y, 2=z)`
- Conforming questions by clicking yes or no `(yes/no)`.
- Position the edit cursor by clicking in the value to edit or by using left & right arrow keys.
- Display serial data as text (default), in hexadecimal notation `<00>` or combined.
- Supports an active terminal screen and a terminal history window with timestamped lines.

1.2 System requirements

- Microsoft Windows W10 or W11.
- Display resolution 1024x768 or higher.
- USB 2.0 port for local connections
- Network adapter for remote connections.
- Bluetooth LE 4.2 adapter for wireless connections

1.3 Version history

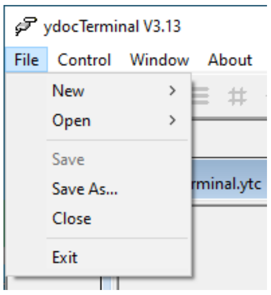
- 2024/01/xx (3.25): Custom payload formatter added.
- 2023/05/25 (3.22): Modbus-terminal & Equation evaluator added.
- 2022/08/02 (3.13): Bluetooth LE support added.
- 2021/06/20 (3.12): IPv6 support added.
- 2021/05/01 (3.11): Feature added to edit serial commands.

1.4 Installation

Download and run https://ydoc.biz/download/ydocterminalsetup.exe

2 Operating basics

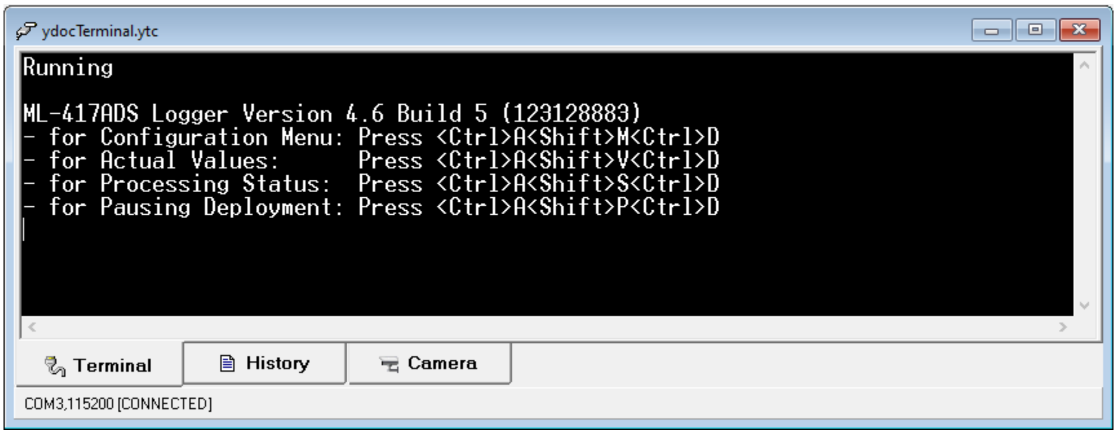
To connect to a data logger you have to create a new or open an existing connection configuration file. A configuration file stores settings like the COM-port used for local connections or the address of the TCP-server for remote connections.



2.1 Device navigation

When you have created a new or opened an existing configuration file ydocTerminal will pop-up a terminal screen.

When connected to an YDOC data logger the initial screen will look similar to the screen below:



```
Running

ML-417ADS Logger Version 4.6 Build 5 (123128883)
- for Configuration Menu: Press <Ctrl>A<Shift>M<Ctrl>D
- for Actual Values:      Press <Ctrl>A<Shift>V<Ctrl>D
- for Processing Status:  Press <Ctrl>A<Shift>S<Ctrl>D
- for Pausing Deployment: Press <Ctrl>A<Shift>P<Ctrl>D
```

COM3,115200 [CONNECTED]

The meaning of the contents of the screen itself is described in the manual of the data logger.

To navigate through the user interface of the connected device you have to perform various key strokes, e.g. pressing and holding the `Ctrl + A` key, the `Shift + M` key and the `Ctrl + D` key to acces the configuration settings.

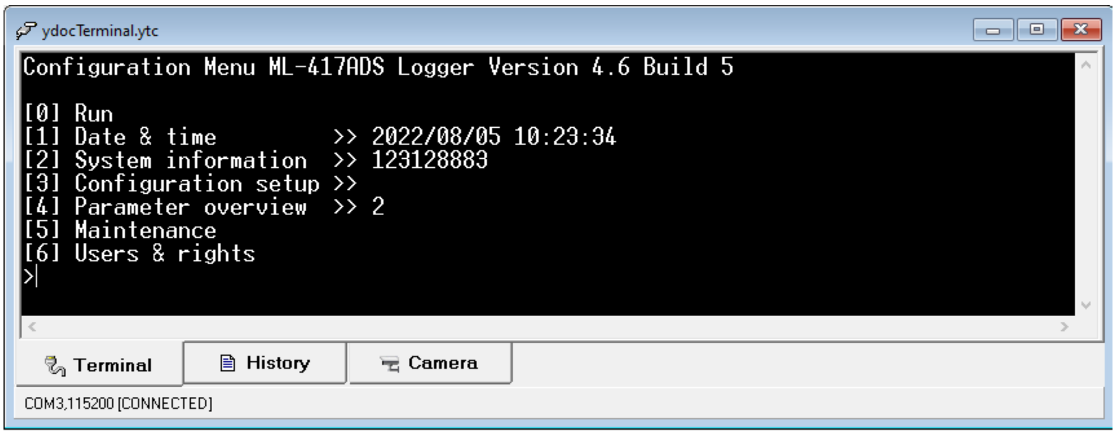
Instead of performing many key strokes you can also simple left click with your mouse somewhere in the displayed `<Ctrl>A<Shift>M<Ctrl>D` text.

Other commonly used clickable keystroke instructions are `<Ctrl-Z>` and `<Esc>` uses to terminate/abort some device action.

When somewhere the text '`press any key`' is displayed the effect of a random left mouse click is similar to pressing any key.

2.2 Device menu navigation

When the data logger displays a configuration menu it will look similar to the screen below:



```
Configuration Menu ML-417ADS Logger Version 4.6 Build 5

[0] Run
[1] Date & time          >> 2022/08/05 10:23:34
[2] System information   >> 123128883
[3] Configuration setup >>
[4] Parameter overview   >> 2
[5] Maintenance
[6] Users & rights
>
```

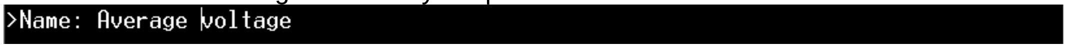
Each option in the menu is preceded with a number/letter between [ ] brackets, to choose an option you can either press a matching key or left mous click the concerning [ ] brackets.

The [0] option is always the first menu option and used to go back one (menu) level. When some configuration setting has changed the confirmation question `(yes/no)` will be dispalyed prior to going back one (menu) level. To confim you can either press the `Y` or `N` key or left mous click the `yes` or `no` text.

2.2.1 Editing device values

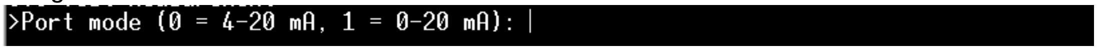
When a device value needs to be edited the data logger will display an editing prompt as the last line of a menu screen. The line starts with a great than token `>` followed by the name of the value to edit concluded with a colon **:**

It the value to edit is a text value, then the existing value will be displayed and you can left click your mouse or use the left/right arrow keys to position the cursor.



```
>Name: Average voltage
```

If the value has to be chosen from a limited list the data logger will display a line with options. To choose an option you can either press the key matching or left mous click the number displayed before the assigment token `=`.



```
>Port mode (0 = 4-20 mA, 1 = 0-20 mA): |
```

2.2.2 Speed buttons

Creates a new connection configuration file (COM-port, baud-rate, commands, etc.).

Opens an existing connection configuration file.

Connects to the concerned device.

Disconnects from the concerned device.

Activates the configuration menu of a running YDOC data logger.

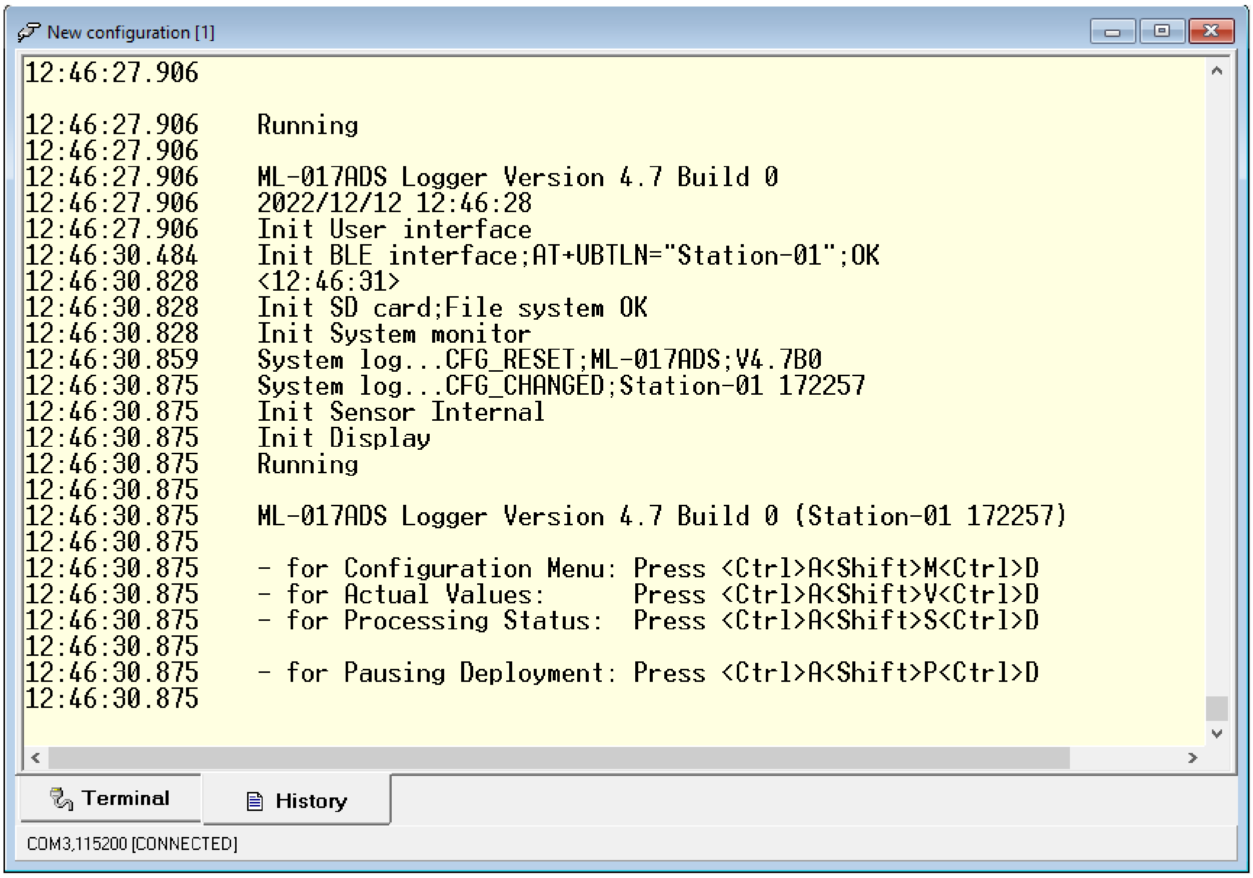
Shows the latest measured values of a running YDOC data logger.

Puts the terminal in passthrough mode (to connect external software over TCP to the device).

F1…F8, sends pre-programmed commands to the connected device.

2.3 Terminal history

The 'Terminal'-page us used to interact with the data loggers/serial device and the window will regularly be refreshed or cleared when the connected device sends new data. You can use the 'history'-page to analyse the communication history without the annoyance that your data of interest is cleared or scrolled out of view. Each line is timestamped, but it can be switched of thru the 'control'-menu.



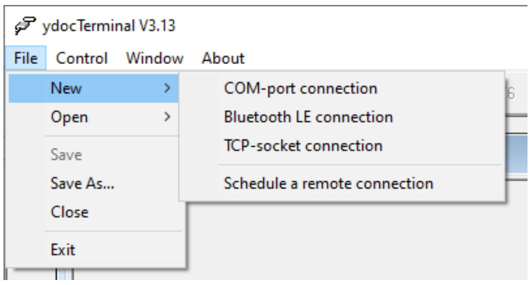
```
12:46:27.906

12:46:27.906    Running
12:46:27.906
12:46:27.906    ML-017ADS Logger Version 4.7 Build 0
12:46:27.906    2022/12/12 12:46:28
12:46:27.906    Init User interface
12:46:30.484    Init BLE interface;AT+UBTLN="Station-01";OK
12:46:30.828    <12:46:31>
12:46:30.828    Init SD card;File system OK
12:46:30.828    Init System monitor
12:46:30.859    System log...CFG_RESET;ML-017ADS;V4.7B0
12:46:30.875    System log...CFG_CHANGED;Station-01 172257
12:46:30.875    Init Sensor Internal
12:46:30.875    Init Display
12:46:30.875    Running
12:46:30.875
12:46:30.875    ML-017ADS Logger Version 4.7 Build 0 (Station-01 172257)
12:46:30.875
12:46:30.875    - for Configuration Menu: Press <Ctrl>A<Shift>M<Ctrl>D
12:46:30.875    - for Actual Values:      Press <Ctrl>A<Shift>V<Ctrl>D
12:46:30.875    - for Processing Status:  Press <Ctrl>A<Shift>S<Ctrl>D
12:46:30.875
12:46:30.875    - for Pausing Deployment: Press <Ctrl>A<Shift>P<Ctrl>D
12:46:30.875
```

COM3,115200 [CONNECTED]

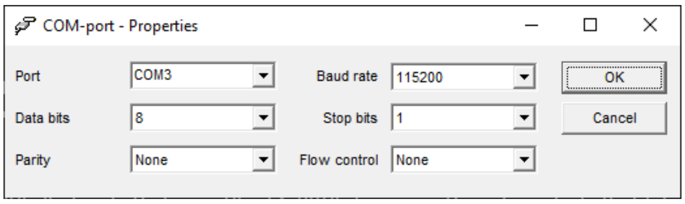
3 Connections

You can use ydocTerminal to hands-on configure an YDOC logger locally by USB, wirelessly by Bluetooth LE or remotely by TCP.



3.1 Local COM-port connection

An YDOC data logger is provisioned with an USB 2.0 port that represents itself to a Windows system as an asynchronous serial communication port (COM-port).



Port	COM3	Baud rate	115200
Data bits	8	Stop bits	1
Parity	None	Flow control	None

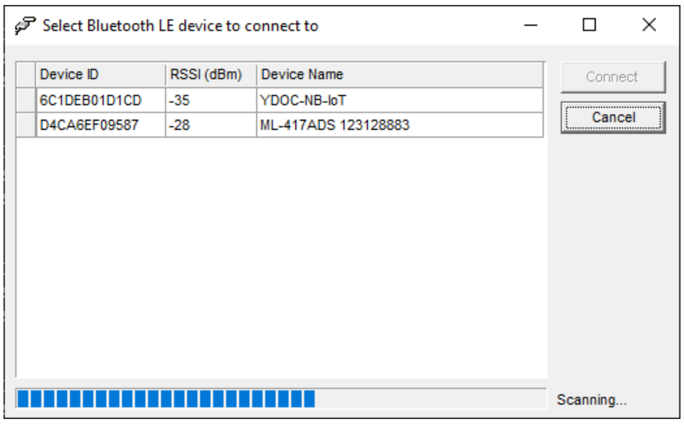
You only have to choose the connected port.

All other settings like baud rate are of interest only if you want to use ydocTerminal to communicate with some other non USB serial device.

3.2 Wireless Bluetooth LE connection

If you have Bluetooth LE (BLE) enabled on your Windows device, then you can use ydocTerminal to wirelessly connect to YDOC data loggers in range and equipped with BLE modules. BLE is chosen over traditional Bluetooth because BLE supports device wake-up over the air keeping power consumption low.

When you create or open an existing BLE connection ydocTerminal will pop-up a scanner scanning 30s for data loggers in range and once the concerned data logger is found you can immediately connect and no need to wait for scanning to finish.

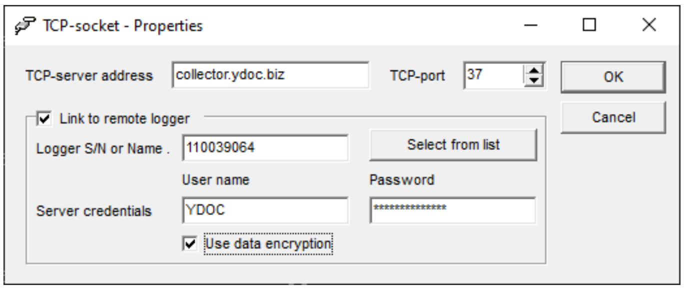


Device ID	RSSI (dBm)	Device Name
6C1DEB01D1CD	-35	YDOC-NB-IoT
D4CA6EF09587	-28	ML-417ADS 123128883

3.3 Remote TCP connection

When a data logger is configured to output log data to a TCP-server and this TCP-server, like our Java data collector, is fully compliant with our native TCP-protocol, then a remote connection to the data logger can be established when the data logger has performed its next data transfer. You might need a bit of patience if your data logger is set to a long data transmission interval.

You need to specify the domain name or IP-address of the TCP-server as well as the TCP-port it is listening at. You should also supply the serial number of the data logger you want to connect to and exactly the same credentials as used by the data logger. We also recommend to use data encryption, especially when you want to connect from a public network.



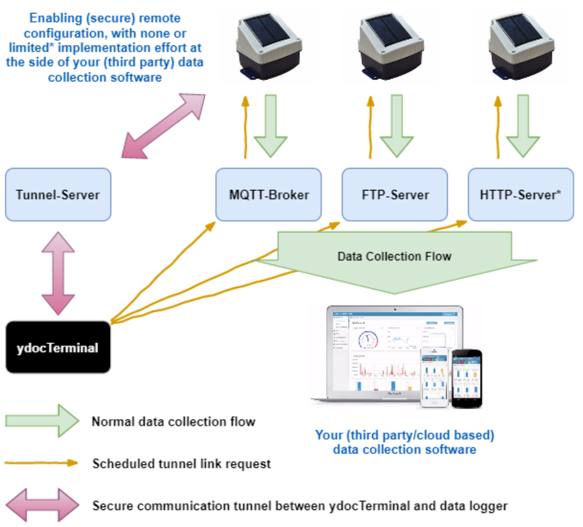
TCP-server address: collector.ydoc.biz
TCP-port: 37
☑ Link to remote logger
Logger S/N or Name: 110039064
Server credentials: User name: YDOC Password: **************
☑ Use data encryption

3.4 Scheduled remote TCP connection

Performing remote configuration of our data loggers with ydocTerminal in combination with our Java data collector is pretty easy, but obviously a lot of clients, which we encourage, want to use their own preferred (third party/cloud based) data collection software.

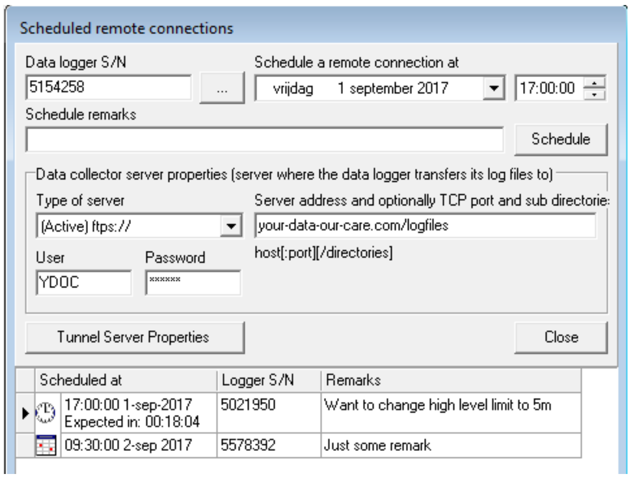
Commonly, data collection software is provisioned with several methods to collect data (e.g. by importing CSV files by FTP or JSON messages through MQTT). However, enabling interactive remote configuration through such systems is a "different cup of tea" and requires a lot of implementation effort if possible at all!

Luckily we have implemented a method to unburden the developers of such systems, so they can fully concentrate on what they need to do with all the collected data. The idea behind our method is to have a separate "side path" for configuration purposes, but a path that is only walked when there is a need to configure, not unnecessarily consuming precious battery power! Our method makes use of a "tunnel-server" servicing secure and "peer-to-peer" encrypted communication channels between data loggers and ydocTerminal.


Enabling (secure) remote configuration, with none or limited* implementation effort at the side of your (third party) data collection software
Tunnel-Server
MQTT-Broker
FTP-Server
HTTP-Server*
Data Collection Flow
ydocTerminal
Your (third party/cloud based) data collection software
Normal data collection flow
Scheduled tunnel link request
Secure communication tunnel between ydocTerminal and data logger

When a user wants to configure a data logger he can use ydocTerminal to schedule a "side path" connection by sending a request to the same MQTT-broker, FTP-server or HTTP-server as used by the data loggers and their data collection software. After a data logger has performed its regular scheduled data transfer, it will get/read the request from the broker/server.

Specify the S/N of the data logger you want to configure and when you think it’s a convenient moment to configure the logger. *Obviously: A tunnel cannot be established before the next scheduled data transfer interval of the data logger. A tunnel can only be established if ydocTerminal is running at the scheduled moment*.


Scheduled remote connections

Data logger S/N: 5154258
Schedule a remote connection at: vrijdag 1 september 2017 17:00:00
Schedule remarks:

Data collector server properties (server where the data logger transfers its log files to)

Type of server: (Active) ftps://
Server address and optionally TCP port and sub directorie: your-data-our-care.com/logfiles
host[:port][/directories]
User: YDOC
Password: ******

Scheduled at	Logger S/N	Remarks
17:00:00 1-sep-2017 Expected in: 00:18:04	5021950	Want to change high level limit to 5m
09:30:00 2-sep 2017	5578392	Just some remark

When running, ydocTerminal will pop-up a terminal window a few minutes before the scheduled moment.

3.4.1 Data transfer server properties

Supported server types: MQTT-broker, FTP(S)-server, HTTP(S)-server & TCP-server.

Server address: The same address, port and path (URL) as used by the data logger (e.g. your-data-our-care.com/logfiles)

User/Password: The same credentials as used by the data logger as they are a/o used in the encryption keys.

3.4.2 Tunnel server properties:

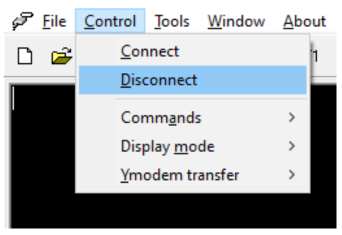
This is the Server servicing the secure and encrypted channel between data logger and ydocTerminal. Standard our server is used (tunnel.ydoc.biz), don’t worry we can’t eavesdrop your data as the communication is ‘Peer-to-Peer’ encrypted by security keys not know to the tunnel server. If having doubts, please feel free to run your own tunnel server (ydocDatacollector) on your own Java virtual machine.

3.4.3 Prerequisites

- **NTP**: ydocTerminal and data loggers don’t have to be in the same time zone, but should all be in sync with the world time clock.
- **FTP**: User should have read/write/delete rights and the FTP-server/collector should not auto delete any *.tlr files.
- **HTTP**: The HTTP-server should pass the request to the data logger in a custom HTTP response header (See: Data logger manual for details).

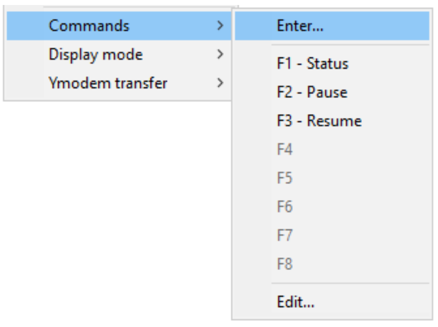
4 Control-menu

The options in the “Control”-menu are used to perform operations on the focused terminal window, like disconnecting from or (re)connecting to a device.



4.1 Commands

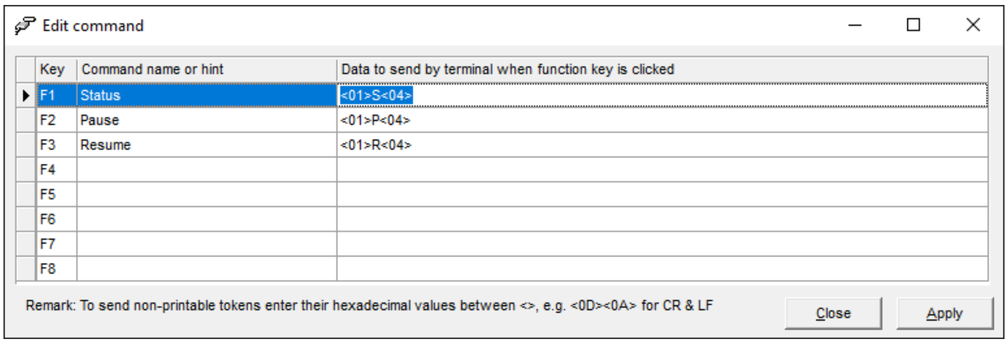
An YDOC data logger can be operated by clicking and typing in the terminal window, but when you are accessing another serial device (directly or indirectly passing thru an YDOC data logger), you might need to enter commands that are hard to type manually or recall when you need them.



If a command is only to be executed once you can use the option ‘Enter..’ to assemble the command and send it to the connected device.

If commands are to be executed more often you can build up a list of up to 8 pre-assembled commands, give them a convenient name and link them to keyboard function-keys F1 until F8 for quick access.

Configured commands can be stored in the active connection configuration file, so you can maintain various command sets.



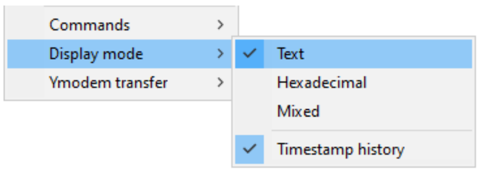
Edit command

Key	Command name or hint	Data to send by terminal when function key is clicked
F1	Status	<01>S<04>
F2	Pause	<01>P<04>
F3	Resume	<01>R<04>
F4		
F5		
F6		
F7		
F8		

Remark: To send non-printable tokens enter their hexadecimal values between <>, e.g. <0D><0A> for CR & LF

If a command contains non-printable tokens or tokens that can't be easily entered with a keyboard you can use the hexadecimal presentation of the token instead and surround them with greater and lower than signs. E.g. to enter a carriage return type <0D>, <0A> for line feed or <1B> for escape etc…

4.2 Display mode



The serial output of an YDOC-data logger is in readable ASCII text, but other devices might output unreadable tokens as well and to analyse the output you might want switch the display mode to 'Hexadecimal' in which case all received tokens will be represented in their hexadecimal values and surrounded by greater and lower than signs, e.g. <1B> for an escape character. If you choose 'Mixed' only non-ASCII and unprintable tokens will be represented in their hexadecimal notations.

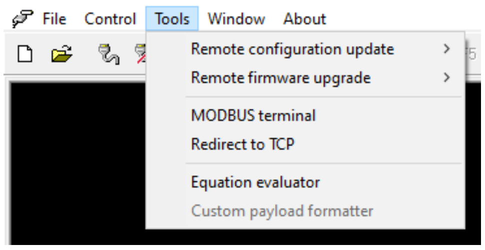
If you want to copy and use the accumulated raw data from the history window without having a generated timestamp in front of each line you have to uncheck the "Timestamp history"-option first.

4.3 yModem transfer

If a connected device supports the yModem transfer protocol, then you can use this option to send a file to or receive a file from the connected device. An YDOC data logger supports this protocol as well, but you don't have to initiate the transfers through these menu options manually as ydocTerminal detects automatically when an yModem transfer should be initiated.

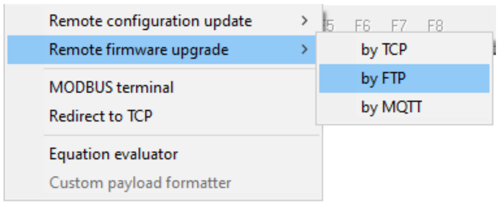
5 Tools-menu

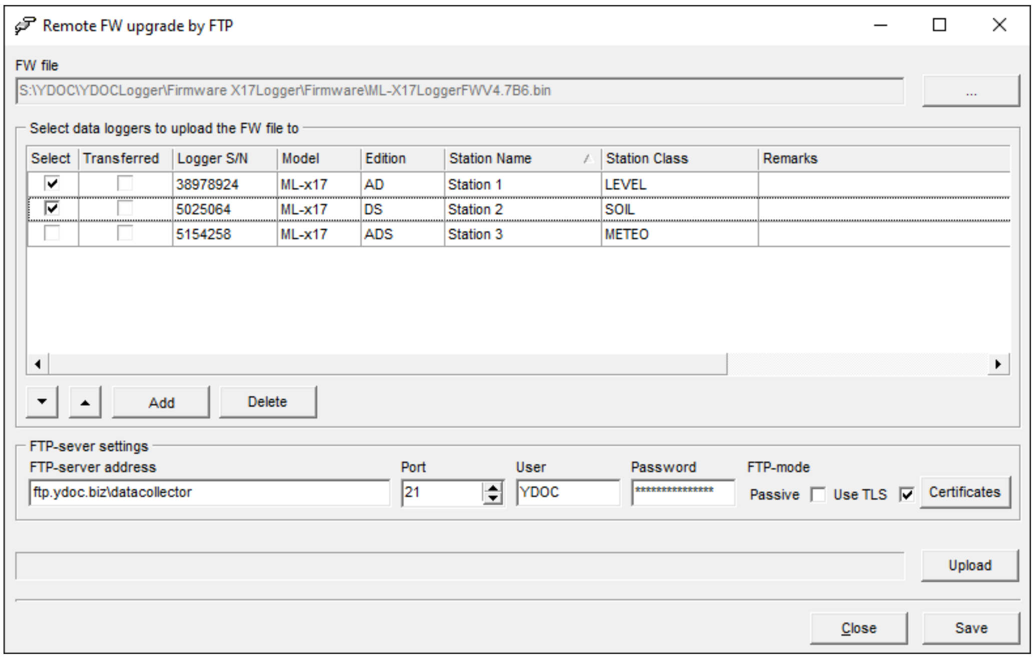
This menu contains various supportive tools, e.g. to upgrade or update your data loggers in the background.



5.1 Remote configuration update & firmware upgrade

When you have local, wireless or a remote connection with a data logger you can perform a configuration update of firmware upgrade manually. However, when you have many data loggers to service, this can become a time consuming task. When your data loggers are making use of FTP, MQTT or our native TCP protocol, then you can automate this task to be performed in the background by transfer configuration or firmware files to the same MQTT-broker, FTP-server or ydocDataCollector as used by your data loggers.




Remote FW upgrade by FTP

FW file
S:\YDOC\YDOCLogger\Firmware X17Logger\Firmware\ML-X17LoggerFWV4.7B6.bin

Select data loggers to upload the FW file to

Select	Transferred	Logger S/N	Model	Edition	Station Name	Station Class	Remarks
✓		38978924	ML-x17	AD	Station 1	LEVEL	
✓		5025064	ML-x17	DS	Station 2	SOIL	
		5154258	ML-x17	ADS	Station 3	METEO	

Add Delete

FTP-sever settings

FTP-server address	Port	User	Password	FTP-mode
ftp.ydoc.biz\datacollector	21	YDOC	***************	Passive [ ] Use TLS [✓]

Certificates

Upload

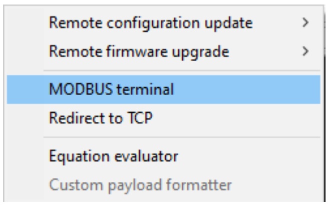
Close Save

You can build up a list of your data loggers and organize them in free definable configuration classes so when you want to update the configurations of a particular class of data loggers, sort them by the class column and select all the data loggers you want to update with the same configuration or upgrade to the same firmware. After clicking the ‘Upload’-button, the chosen file will be cloned into individual files per selected datalogger and transferred to the concerned server. When a data logger contacts the server for a (regular) log data transfer it will pull when ready the file from the server and perform the update or upgrade. Whether and update or upgrade is performed successfully can be determined by analysing the log data of the concerned data logger after the next data transfer if necessary.

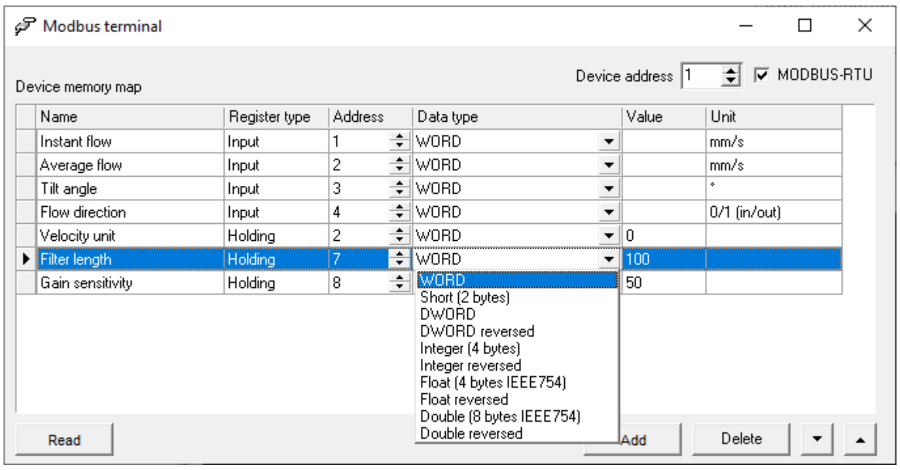
5.2 MODBUS Terminal

Reading measurements from connected MODBUS compatible devices/sensors is rather straight forward for various models and brands, a matter of specifying addresses and data types of the memory registers to be read. However, servicing such devices is a different cop of tea. MODBUS is a binary protocol its not do able to type servicing commands manually and to interpret their responses.

The “MODBUS Terminal”-tool can be used to service connected MODBUS devices more easily, whether connected directly to ydocTerminal or indirectly thru a local (USB), wireless (BLE) or remote (TCP) accessible YDOC data logger. **Note**: The data logger should be put in COM-terminal mode first when using indirect connections.



You can build up a memory-map that can be stored along with the active connection configuration file, so you can maintain various memory maps.



Device address 1 [✓ MODBUS-RTU]

Name	Register type	Address	Data type	Value	Unit
Instant flow	Input	1	WORD		mm/s
Average flow	Input	2	WORD		mm/s
Tilt angle	Input	3	WORD		°
Flow direction	Input	4	WORD		0/1 (in/out)
Velocity unit	Holding	2	WORD	0	
Filter length	Holding	7	WORD	100	
Gain sensitivity	Holding	8		50	

WORD
Short (2 bytes)
DWORD
DWORD reversed
Integer (4 bytes)
Integer reversed
Float (4 bytes IEEE754)
Float reversed
Double (8 bytes IEEE754)
Double reversed

Reading & Writing

When a connection to the device is active, clicking the ‘Read’-button will cause a read-out of all the configured registers. To write to a register, click the ‘Value’ field of a ‘Holding’-register, type the wished value, press enter or leave the ‘Value’-field.

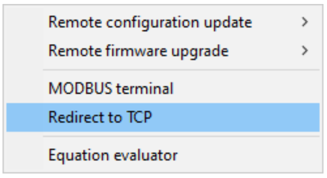
Device addressing: Specify the address of the MODBUS device, this is commonly 1 in case you have only one connected device on the same port/bus. MODBUS comes in two flavours MODBUS-RTU or MODBUS-ASCII. MODBUS-RTU is most common, if your device supports MODBUS-ASCII then uncheck the MODBUS-RTU box.

Register type: An ‘input’-register is by definition a read-only register used to store values acquired from connected sensors. A ‘holding’-register is similar and can also be read-only and be used to store the same input values, but can also be a read/write register to read or change variables (e.g. calibration values).

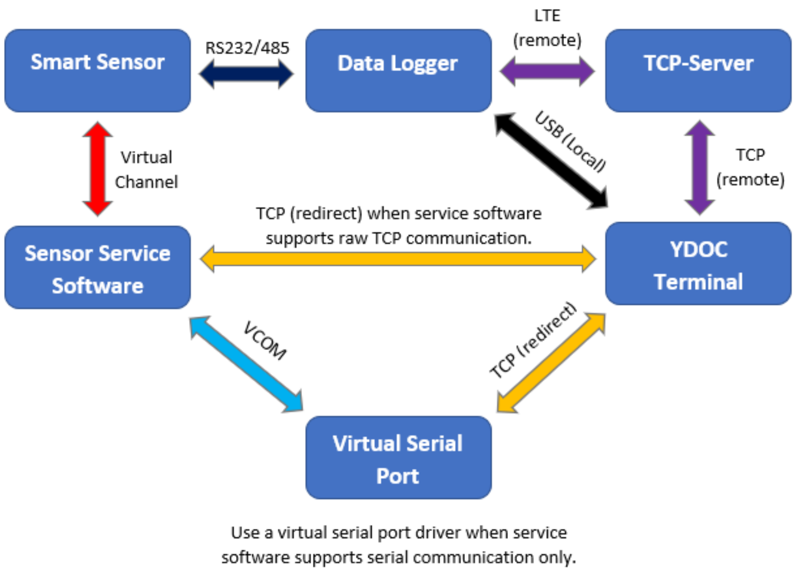
Data type: The data type specifies the format of the data contained by the register and this can be a WORD (an unsigned 16bits integer), a Short (a signed 16bits), a DWORD (an unsigned 32bits integer), a DWORD (a signed 32bits integer), a Float (a 4-bytes IEEE754 single precision floating point value) or a Double (an 8-bytes IEEE754 double precision floating point value). Some types are also available in reversed order (big endian vs small endian), if you don’t know just try.

Name & Unit: This are convenience fields to help you remember the meaning of the various registers.

5.3 TCP redirect server



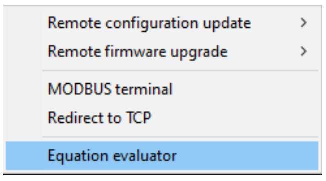
Reading measurements from connected devices/sensors is handled by the various standard input drivers of an YDOC data logger. However, servicing connected devices is always very specific and often requiring vendor specific software. The “TCP redirect server”-Tool is a TCP-server that you can run temporarily to transparently pass serial data between a connected serial device and a TCP-client. Where the TCP-client can be a virtual COM-port driver or vendor specific service software supporting serial communication over plain TCP. A serial device itself can be connected directly to ydocTerminal or indirectly thru a local (USB), wireless (BLE) or remote (TCP) accessible YDOC data logger.



Notes:

1) The data logger should be put in COM-terminal mode first when using indirect connections and
2) The TCP-communication between server and client is transparent, not encrypted and not secure. Its not recommend to use it over public internet.

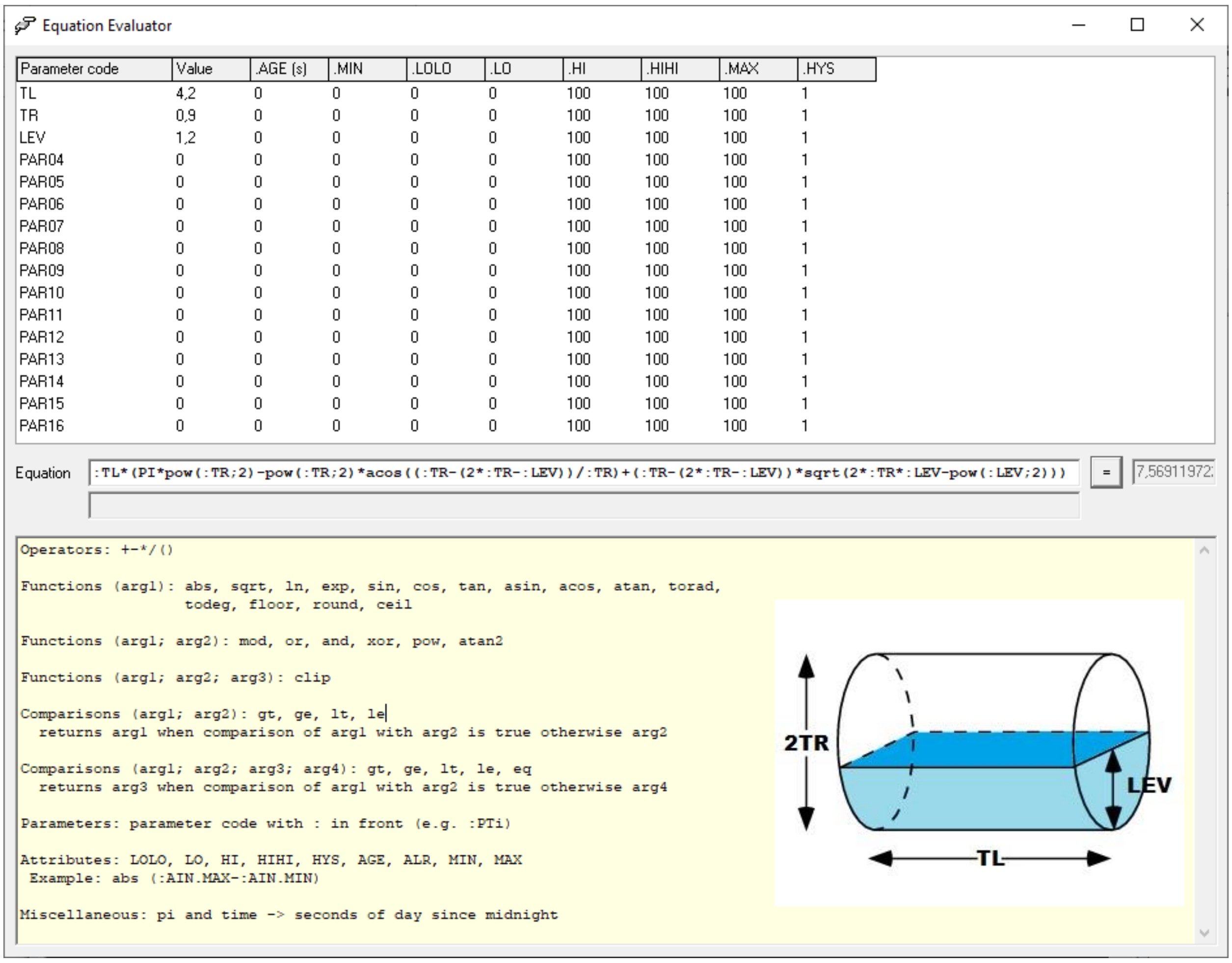
5.4 Equation evaluator



An YDOC data logger is equipped with several calculated channels to derive engineering values from sensed values using mathematical operators and functions (e.g. cos, sin, atan2, ln, sqrt).

The “Equation evaluator”-tool can be used to evaluate equations on your PC in case you don’t have a data logger at hand or in case you think its more convenient to you.

You can build up a list of up to 16 parameters of which their values and limits can be substituted as variables in your equations. **Note**: AGE will auto increment 1 per second when entering a value >0.



Parameter code	Value	.AGE (s)	.MIN	.LOLO	.LO	.HI	.HIHI	.MAX	.HYS
TL	4,2	0	0	0	0	100	100	100	1
TR	0,9	0	0	0	0	100	100	100	1
LEV	1,2	0	0	0	0	100	100	100	1
PAR04	0	0	0	0	0	100	100	100	1
PAR05	0	0	0	0	0	100	100	100	1
PAR06	0	0	0	0	0	100	100	100	1
PAR07	0	0	0	0	0	100	100	100	1
PAR08	0	0	0	0	0	100	100	100	1
PAR09	0	0	0	0	0	100	100	100	1
PAR10	0	0	0	0	0	100	100	100	1
PAR11	0	0	0	0	0	100	100	100	1
PAR12	0	0	0	0	0	100	100	100	1
PAR13	0	0	0	0	0	100	100	100	1
PAR14	0	0	0	0	0	100	100	100	1
PAR15	0	0	0	0	0	100	100	100	1
PAR16	0	0	0	0	0	100	100	100	1

Equation `:TL*(PI*pow(:TR;2)-pow(:TR;2)*acos((:TR-(2*:TR-:LEV))/:TR)+(:TR-(2*:TR-:LEV))*sqrt(2*:TR*:LEV-pow(:LEV;2)))` = 7,56911972

```
Operators: +-*/()

Functions (arg1): abs, sqrt, ln, exp, sin, cos, tan, asin, acos, atan, torad,
                  todeg, floor, round, ceil

Functions (arg1; arg2): mod, or, and, xor, pow, atan2

Functions (arg1; arg2; arg3): clip

Comparisons (arg1; arg2): gt, ge, lt, le
  returns arg1 when comparison of arg1 with arg2 is true otherwise arg2

Comparisons (arg1; arg2; arg3; arg4): gt, ge, lt, le, eq
  returns arg3 when comparison of arg1 with arg2 is true otherwise arg4

Parameters: parameter code with : in front (e.g. :PTi)

Attributes: LOLO, LO, HI, HIHI, HYS, AGE, ALR, MIN, MAX
 Example: abs (:AIN.MAX-:AIN.MIN)

Miscellaneous: pi and time -> seconds of day since midnight
```

In example to calculate the volume in m^3 of a horizontal placed cylindrical tank, enter the formula:

:TL*(PI*pow(:TR;2)-pow(:TR;2)*acos((:TR-(2*:TR-:LEV))/:TR)+(:TR-(2*:TR-:LEV))*sqrt(2*:TR*:LEV-pow(:LEV;2)))

Where **TL** is tank length (4.2m) , **TR** tank radius (0.9m) and **LEV** the measured tank level.

Measuring a tank level of 1.2m will evaluate to about 7.6 m^3 volumetric content.

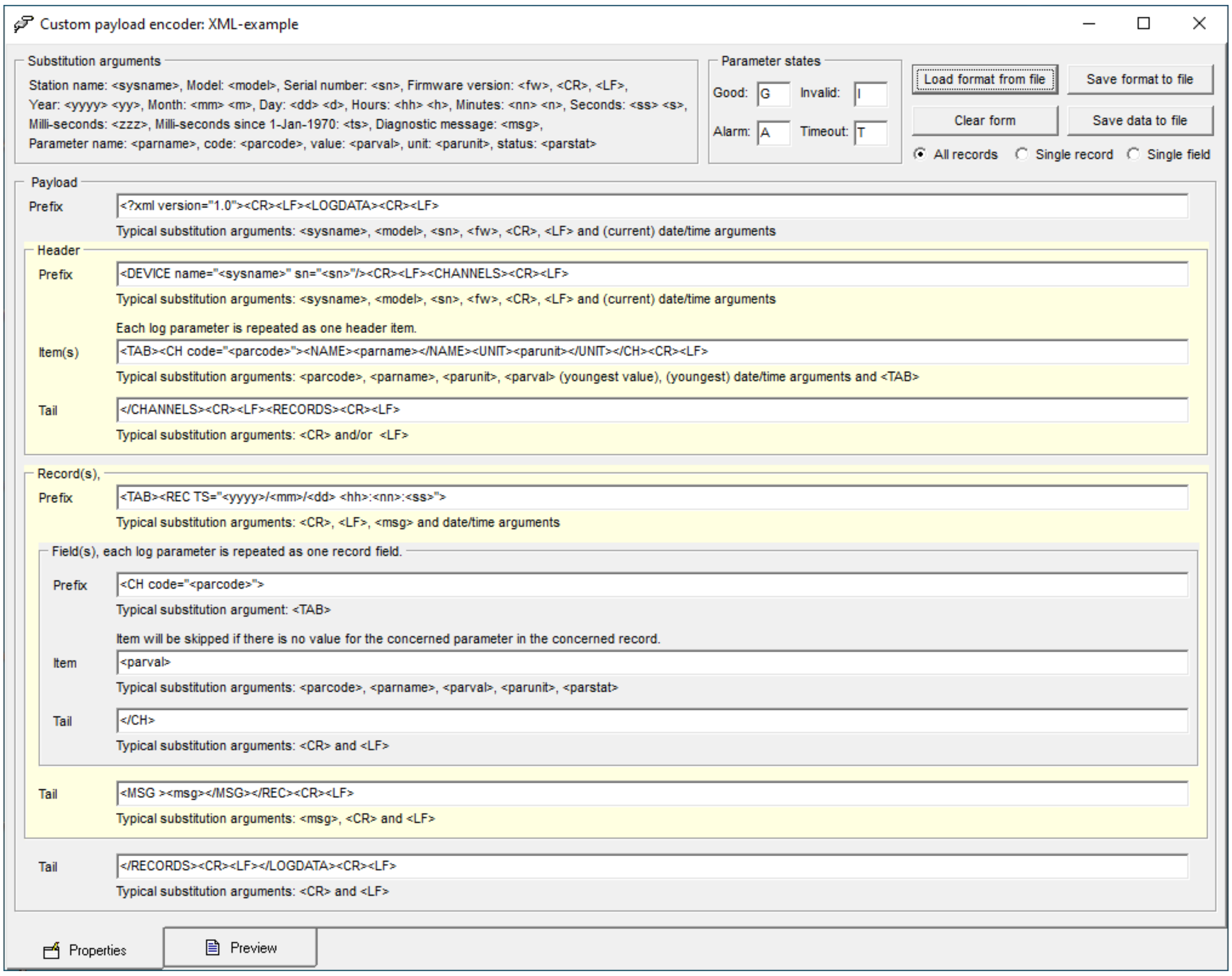
5.5 Custom payload encoder

We prefer the use of our native plain text and JSON payload formats as they are lightweight in order to minimize data transfer time and power consumption. JSON has achieved widespread adoption in the IoT industry with support from multiple platforms of which some have provisions to decode foreign JSON formats. To extend compatibility we also support CSV commonly used by legacy system and SparkplugB a rising format in SCADA environments. There a numerous alternate payload formats in use, including various XML formats and vendor specific plain text formats and its undoable to accommodate them all.

To further increase compatibility we have implemented a “Custom payload encoder” in our data logger firmware, which can be used to encode our native payload formats into a custom JSON, XML or any other text based payload format (within the operational capability of the encoder).

The 'Custom Payload Encoder'-tool allows users to define the encoding for a custom payload format. And once satisfied with the preview result, the definition can be saved as a configuration file for seamless upload into a data logger.

You can start the 'Custom Payload Encoder'-tool from the ‘Tools’-menu when you have first opened an YDOC native data log file or performed a data download from a data logger.



Substitution arguments

Station name: <sysname>, Model: <model>, Serial number: <sn>, Firmware version: <fw>, <CR>, <LF>,
Year: <yyyy> <yy>, Month: <mm> <m>, Day: <dd> <d>, Hours: <hh> <h>, Minutes: <nn> <n>, Seconds: <ss> <s>,
Milli-seconds: <zzz>, Milli-seconds since 1-Jan-1970: <ts>, Diagnostic message: <msg>,
Parameter name: <parname>, code: <parcode>, value: <parval>, unit: <parunit>, status: <parstat>

Parameter states

Good: G Invalid: I
Alarm: A Timeout: T

[✓ All records] Single record Single field

Payload

Prefix <?xml version="1.0"><CR><LF><LOGDATA><CR><LF>
Typical substitution arguments: <sysname>, <model>, <sn>, <fw>, <CR>, <LF> and (current) date/time arguments

Header

Prefix <DEVICE name="<sysname>" sn="<sn>"/><CR><LF><CHANNELS><CR><LF>
Typical substitution arguments: <sysname>, <model>, <sn>, <fw>, <CR>, <LF> and (current) date/time arguments

Each log parameter is repeated as one header item.
Item(s) <TAB><CH code="<parcode>"><NAME><parname></NAME><UNIT><parunit></UNIT></CH><CR><LF>
Typical substitution arguments: <parcode>, <parname>, <parunit>, <parval> (youngest value), (youngest) date/time arguments and <TAB>

Tail </CHANNELS><CR><LF><RECORDS><CR><LF>
Typical substitution arguments: <CR> and/or <LF>

Record(s),

Prefix <TAB><REC TS="<yyyy>/<mm>/<dd> <hh>:<nn>:<ss>">
Typical substitution arguments: <CR>, <LF>, <msg> and date/time arguments

Field(s), each log parameter is repeated as one record field.

Prefix <CH code="<parcode>">
Typical substitution argument: <TAB>

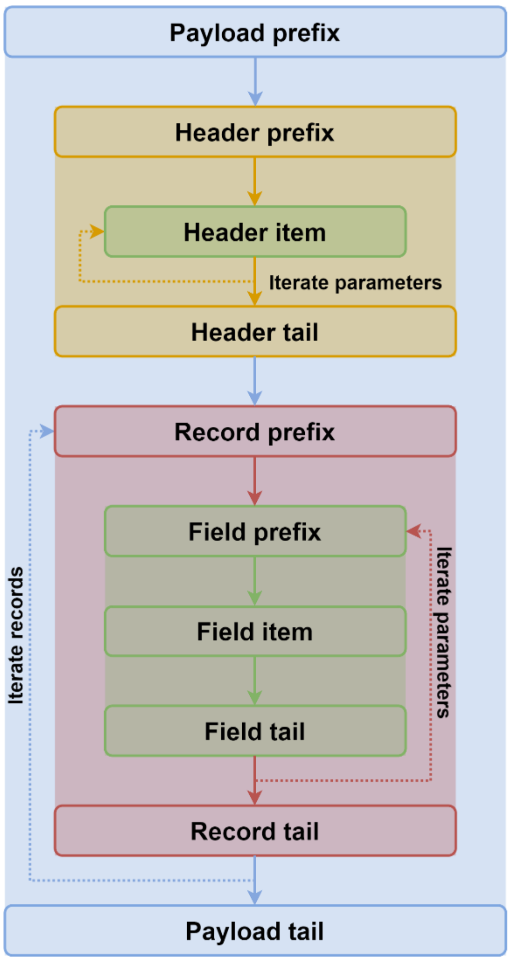
Item will be skipped if there is no value for the concerned parameter in the concerned record.
Item <parval>
Typical substitution arguments: <parcode>, <parname>, <parval>, <parunit>, <parstat>

Tail </CH>
Typical substitution arguments: <CR> and <LF>

Tail <MSG ><msg></MSG></REC><CR><LF>
Typical substitution arguments: <msg>, <CR> and <LF>

Tail </RECORDS><CR><LF></LOGDATA><CR><LF>
Typical substitution arguments: <CR> and <LF>

5.5.1 Process



The "encoder" will be fed by a data log file in native payload format and controlled by up to 10 user defined instructions. An instruction is a free editable text along with substitution arguments. An instruction appears at designated moments in the process flow and their substitution arguments will be substituted with data applicable at the designated moment.

Like our native data log file we assume that a custom payload starts with a prefix part and consequently concluded with tail part.

The payload prefix is followed by a header describing the channels/parameters used in the payload. The header starts with a prefix part and consequently concluded with a tail and in between an item part that is repeated for every available parameter used.

After the header, the records contained in the data log file will follow. Each record starts with a prefix part and consequently concluded with a tail and in between the fields contained in the record, the number and order of fields is equal to the number and order of parameters/channels in the header.

A field starts with a prefix part and consequently concluded with a tail and in between an item part commonly used to represent a measured value. The item part will be skipped if there is no value available at the give record timestamp or if the measured value is marked as invalid or timed-out and the item instruction does not contain a <parstat> substitution token.

5.5.2 Substitution arguments

Substitution arguments are placed between greater and lower than signs, the following arguments are possible:

- <sysname> system name of the data logger
- <model> model of the data logger, e.g. **ML-417ADS**
- <sn> serial number of the data logger
- <fw> firmware version of the data logger, e.g. **4.8B1**
- <CR>, <LF> & <TAB> to substitute a carriage return, line feed or tab character.
- <yyyy>, <yy> year of designated timestamp in process flow
- <mm>, <m> month of designated timestamp in process flow
- <dd>, <d> day of designated timestamp in process flow
- <hh>, <h> hour of designated timestamp in process flow
- <nn>, <n> minute of designated timestamp in process flow
- <ss>, <s> second of designated timestamp in process flow
- <zzz> thousand of second of designated timestamp in process flow
- <ts> milli-seconds since 1-jan-1970 of designated timestamp in process flow
- <msg> diagnostic message contained in the designated record in the process flow
- <parname> name of designated parameter in the process flow

- <parcode> code of designated parameter in the process flow
- <parval> value of designated parameter in the process flow
- <parstat> status of designated value in the process flow

Notes:

1) When a text between greater and lower than signs is not recognized as substitution argument the whole text including signs will be treated as literal text.
2) If you want to generate an XML-payload and a substitution argument e.g. <sn> should accidentally match with an XML-tag, then you could edit a space in the XML-tag to distinguish it from a substitution argument. An XML parser will ignore unnecessary spaces, e.g. <sn > is equal to <sn>
3) The total number of characters of all instructions combined may not exceed 246 bytes, a substitution argument counts as one byte.

5.5.3 Instructions

Payload prefix & Tail

The payload prefix is commonly used to include meta data including <sysname>, <sn>, <fw> and <model>. When including a timestamp argument it will be substituted with the current session time.

Header prefix & Tail

Can be used to start and conclude a list with channel/parameter info and can also be used to include meta data like <sysname>, <sn>, <fw> and <model>. When including a timestamp argument it will be substituted with the current session time.

Header item

The header item is used to provide info about individual channels/parameters and used to iterate <parname>, <parcode> and or <parunit>. If <parval>, <parstat> or timestamp arguments are included, then it will give the value, status and youngest known timestamp of the iterated parameter,

Record prefix and tail

Used to start and conclude a record the typical substitution arguments are timestamp of the current record and a diagnostic message <msg> to include in the record.

Field prefix and tail

Used to start and conclude a record field, it commonly contains literal text only, e.g. a comma or space.

Field item

A field should contain the current designated value <parval> of the current designated parameter and optionally status <parstat>. This field will be skipped if there is no value available in the current designated record or in case <parstat> is not included and <parvalue> is marked as invalid or timed-out.

5.5.4 Parameter states

You can use the <parstat> substitution token to indicate the status of a <parval> and define your own status indicators. An YDOC value can have for different states: 1) The value is good, 2) The value is outside configured limits (Alarm), 3) The value is invalid, 4) The value is not acquired in time (timeout).

5.5.5 Partial encoding

An YDOC data log file can be encoded to a custom payload in one session (all records at once) or split up in multiple sessions, a session per single record or per single record field.