ENGLISH

Anybus® Communicator™ - EtherCAT to Modbus RTU/Serial

USER MANUAL

SCM-1202-208
Version 1.11
Publication date 2024-05-02

Important User Information

Disclaimer
The information in this document is for informational purposes only. Please inform HMS Networks of any inaccuracies or omissions found in this document. HMS Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Networks and is subject to change without notice. HMS Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the product. The effects caused by any direct or indirect use of such aspects of the product are undefined and may include e.g. compatibility issues and stability issues.

Contact Information
Postal address:
Box 4126
300 04 Halmstad, Sweden

E-Mail: info@hms.se

Table of Contents

1. Preface

1.1. About This Document

This document describes how to install and configure Anybus® Communicator™.

For additional documentation and software downloads, FAQs, troubleshooting guides and technical support, please visit www.anybus.com/support.

1.2. Document Conventions

Lists

Numbered lists indicate tasks that should be carried out in sequence:

1. First do this
2. Then do this

Bulleted lists are used for:

- Tasks that can be carried out in any order
- Itemized information

User Interaction Elements

User interaction elements (buttons etc.) are indicated with bold text.

Program Code and Scripts

```
Program code and script examples
```

Cross-References and Links

Cross-reference within this document: Document Conventions (page 1)

External link (URL): www.anybus.com

Safety Symbols

DANGER
Instructions that must be followed to avoid an imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING
Instructions that must be followed to avoid a potential hazardous situation that, if not avoided, could result in death or serious injury.

CAUTION
Instruction that must be followed to avoid a potential hazardous situation that, if not avoided, could result in minor or moderate injury.

IMPORTANT
Instruction that must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

Information Symbols

NOTE
Additional information which may facilitate installation and/or operation.

TIP
Helpful advice and suggestions.

1.3. Trademarks

Anybus® is a registered trademark of HMS Networks.

All other trademarks are the property of their respective holders.

1.4. About the EtherCAT Terminology

The EtherCAT® Technology Group has changed the terminology for Master and Slave.

Master is called **MainDevice**

Abbreviated: **MDevice**

Slave is called **SubordinateDevice**

Abbreviated: **SubDevice**

2. Safety

2.1. Intended Use

The intended use of this equipment is as a communication interface and gateway.

The equipment receives and transmits data on various physical layers and connection types.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

2.2. General Safety

CAUTION
Ensure that the power supply is turned off before connecting it to the equipment.

CAUTION
This equipment contains parts that can be damaged by electrostatic discharge (ESD). Use ESD prevention measures to avoid damage.

CAUTION
To avoid system damage, the equipment should be connected to ground.

IMPORTANT
Using the wrong type of power supply can damage the equipment. Ensure that the power supply is connected properly and of the recommended type.

3. Cybersecurity

3.1. General Cybersecurity

IMPORTANT
It is important to maintain the cybersecurity of the Communicator.

Before connecting the Communicator to a PLC, ensure the PLC is configured and installed in accordance with the PLC supplier hardening guidelines.

IMPORTANT
To physically secure networks and equipment and to prevent unauthorized access, it is recommended to install the equipment in a locked environment.

IMPORTANT
After completing the configuration of the Communicator, lock the security switch to prevent unauthorized access to the Communicator built-in web interface.

IMPORTANT
To avoid exposure of sensitive data, always perform a factory reset before decommissioning the equipment.

Factory reset will reset any on site made configuration changes and set the Communicator to the same state as leaving HMS production.

See Reset to Factory Settings (page 142).

3.2. Security Advisories

For cybersecurity reasons, stay informed about new vulnerabilities and follow the recommended actions.

HMS Networks Security Advisories includes information about our product vulnerabilities and available solutions.

You find our Safety Advisories at www.hms-networks.com/cybersecurity/security-advisories.

3.3. How to Report a Vulnerability

HMS Networks place the utmost importance on the security of our products and systems, however, despite all the measures we take, it cannot be excluded that vulnerabilities persist.

To report a potential vulnerability in an HMS product or service, please visit www.hms-networks.com/cybersecurity/report-a-vulnerability and follow the instructions.

3.4. Product Cybersecurity Context

3.4.1. Security Defense in Depth Strategy

The defense in depth strategy of the Communicator includes the following security measures:

- Secure Boot: Security standard used to ensure that the Communicator boots using only software that is trusted by HMS Networks.
- Signed firmware: HMS Networks delivers digitally signed firmware. Before the firmware file is imported into the Communicator, the firmware upgrade function performs a validation of the file, to ensure that is authentic.
- Security switch: Used to lock unauthorized access to the Communicator built-in web interface.
- The Communicator is intended to be installed in a Process Control Network (PCN) environment. See Level 1 in the Purdue Model (page 6).
- To physically secure networks and equipment and to prevent unauthorized access, the Communicator is intended to be installed in a locked environment.

3.4.2. Purdue Model

The Communicator is intended to be part of the process control network in Level 1 (E), to enable communication between PLCs or between a PLC and peripheral devices.

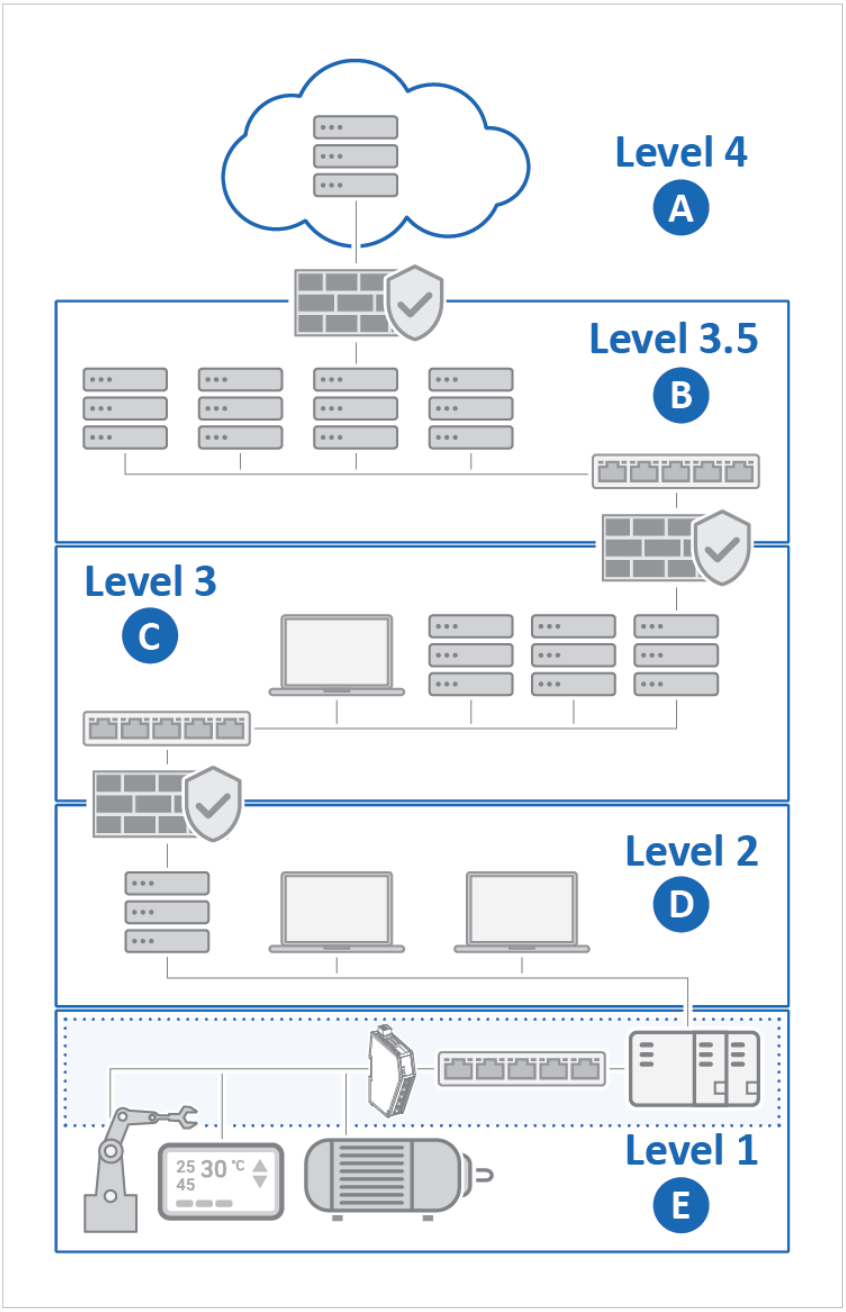


Figure 1. Purdue model, product security context

IT Network

A. **Level 4: Enterprise Network**
Example: Cloud solution, Business LAN (VPN)

B. **Level 3.5: Perimeter Network**
Example: Demilitarized Zone (DMZ)

OT Network

C. **Level 3: Advanced Control Network (ACN)**
Example: SCADA systems, Business control

D. **Level 2: Supervisory Control**
Example: Operator panels, Operator stations, Engineering stations

E. **Level 1: Process Control Network (PCN)**
Environment where the Communicator is installed
Example: Factory floor, Industrial product line

4. Preparation

4.1. Cabling

Have the following cables available:

- Power cable.
- Ethernet cable for configuration.
- Ethernet cable x 2 for connecting to the networks.
- 7-pin screw terminal block connector is included with the product.

4.2. System Requirements

4.2.1. Supported Operating Systems

Operating System	Description
Windows 7 SP1, 32-bit	Windows 7 32-bit with Service Pack 1
Windows 7 SP1, 64-bit	Windows 7 64-bit with Service Pack 1
Windows 10 64-bit	Windows 10 64-bit
Windows 11 64-bit	Windows 11 64-bit

4.2.2. Supported Web Browsers

The Communicator built-in web interface can be accessed from the following standard web browsers.

- Google Chrome
- Microsoft Edge
- Mozilla Firefox

4.3. Mechanical Tools and Equipment

Have the following tools available:

- Flat-head screwdriver, size 5.5 mm
  Needed when removing the Communicator from DIN-rail.
- Flat-head screwdriver, size 3 mm
  Needed when connecting the cables to the 7-pin connector.

4.4. Support and Resources

For additional documentation and software downloads, FAQs, troubleshooting guides and technical support, please visit www.anybus.com/support.

TIP
Have the product article number available, to search for the product specific support web page. You find the product article number on the product cover.

4.5. HMS Software Applications

Download the software installation files and user documentation from www.anybus.com/support.

HMS IPconfig

Use the software application HMS IPconfig and scan your network to discover and change the Communicator IP address and to access the Communicator built-in web interface.

NOTE
As an alternative, you can set a static IP address within the same IP address range as the Communicator IP address on the computer accessing the Communicator built-in web interface.

NOTE
HMS IPconfig is only available for Windows.

4.6. Third-Party Software Applications

Microsoft Excel

Microsoft Excel, or equivalent software application that supports the Office Open XML Workbook (xlsx) file format. Needed to open and read the **I/O data map** file and the **Event log** file.

4.7. Software License Information

For license agreements regarding the third-party software used in the Communicator, refer to the *LICENSE.txt* file(s) included in the Communicator firmware update package zip file.

To download the Communicator firmware update package zip file, please visit www.anybus.com/support.

TIP
Have the product article number available, to search for the product specific support web page. You find the product article number on the product cover.

5. About Anybus Communicator

5.1. Serial Protocol Communication

5.1.1. Serial Protocol Types

The gateway features three distinct modes of operation for the subnetwork communication, called Modbus RTU, Custom Request/Response and Custom Produce/Consume.

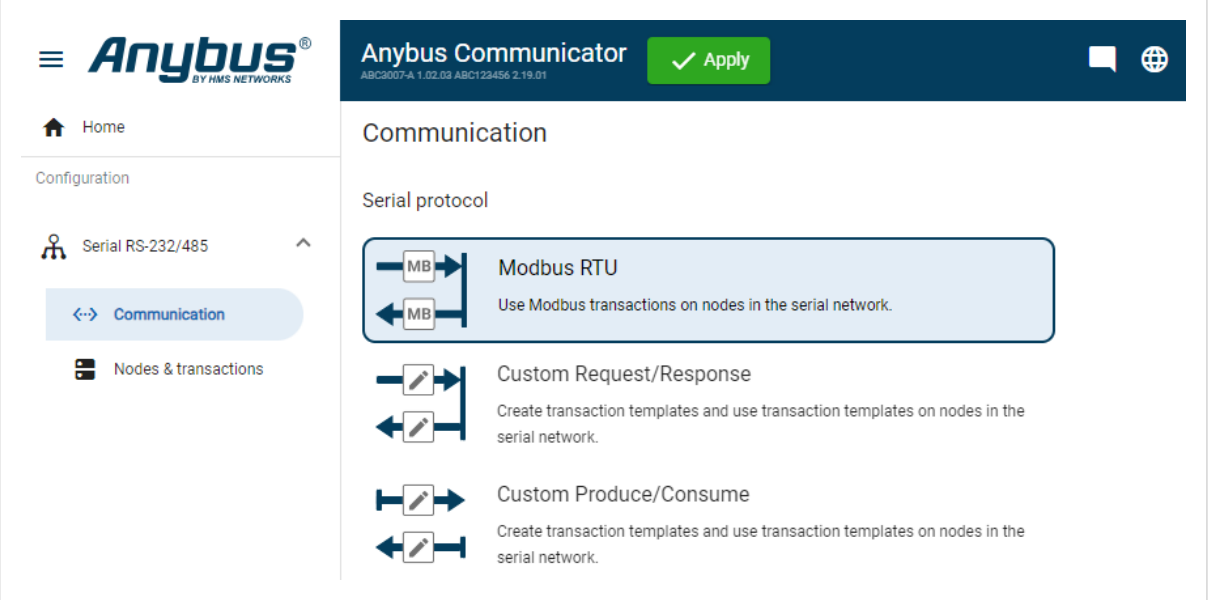


Figure 2. Communication, Serial protocol

Modbus RTU

By default the Communicator uses the Modbus RTU serial protocol.

The Communicator uses Modbus transactions defined by the Modbus standard.

The Communicator acts as a client on the subnetwork, and the serial communication takes place in a request/response fashion.

The nodes on the network are not permitted to issue messages unless they have been addressed by the Communicator first.

Custom Request/Response

In this mode, you can define your own serial transactions to handle a wide range of custom serial protocols.

The Communicator acts as a generic serial client on the subnetwork.

The serial communication takes place in a request/response fashion.

Custom Produce/Consume

In this mode, you can define your own serial transactions to handle a wide range of custom serial protocols.

The Communicator may consume and/or produce messages on the subnetwork.

There is no client-server relationship between the nodes on the network, messages are spontaneously produced or consumed when data is available.

5.1.2. Serial Protocol Building Blocks

The following building blocks are used to describe the subnetwork communication.

Frame Fields

The Frame editor is used to design custom transaction templates.

The Frame editor with Frame fields is available when either the Custom Request/Response or Custom Produce/Consume serial protocol is enabled.

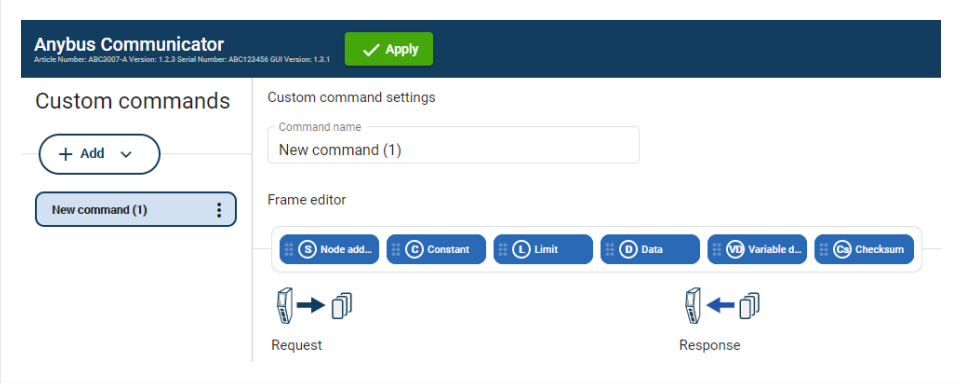


Figure 3. Frame editor

Frame fields are low level entities used to compose transactions.

A frame object can represent a:

- fixed value, a constant
- range of values, limit objects
- block of data or a calculated checksum

Transaction Templates

The Transaction templates are available when either the Custom Request/Response or Custom Produce/ Consume serial protocol is enabled.

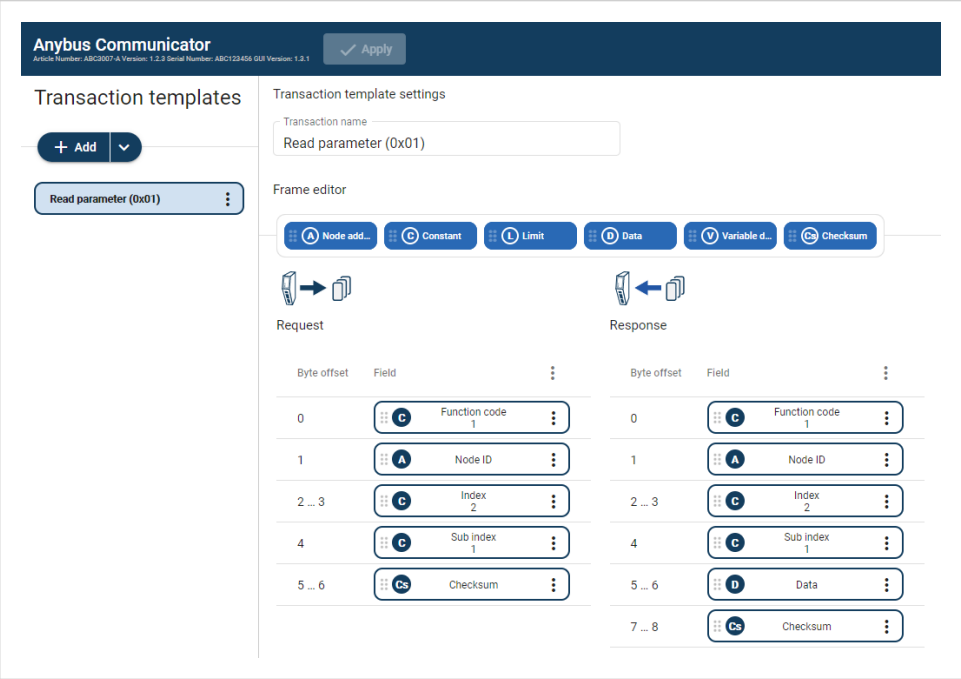


Figure 4. Frame editor, Request and Response

A transaction represents a complete serial telegram, and consists of a number of frame fields.

Each frame field is associated with a set of parameters controlling what is transmitted on the subnetwork.

The transaction templates are stored in the Communicator and can be reused multiple times.

Example 1. Common Read Transaction

If you have a common read transaction. Then you can create one single transaction template for the read transaction and reuse it multiple times times on your node(s).

If you have a function code in your protocol similar to a standard Modbus RTU transaction. Then you can create a transaction template based on the Modbus RTU transaction for the read operation. When you reuse the template on your node(s), you only have to change the function code each time it is used.

5.2. How the Communication Works

The Communicator enables communication, data exchange, between one or more server devices connected to a serial subnetwork and a client device connected to a high level network.

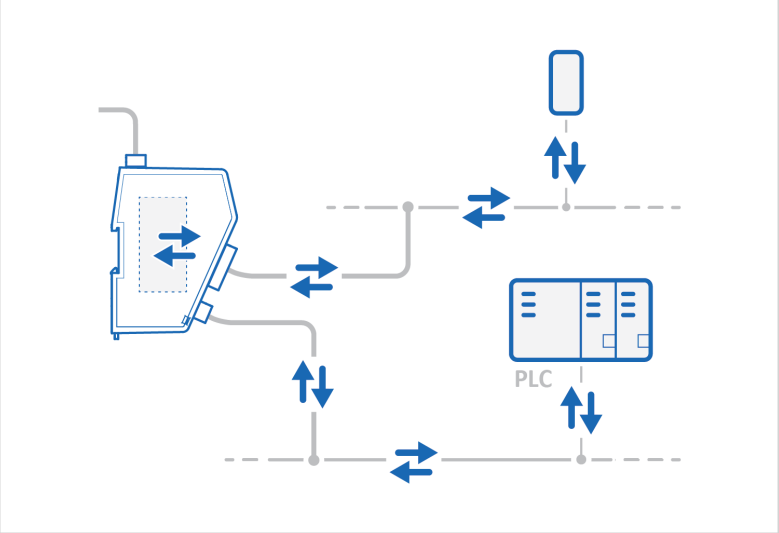


Figure 5. Process data traffic overview

For example:

- The client device can be a PLC controller or a PC.
- A server device can be a sensor, scanner, industrial robot, or sniffer.

The Communicator main task is to send the transactions that the server device(s) are configured to execute, in order to request and transfer process data.

Request Process Data

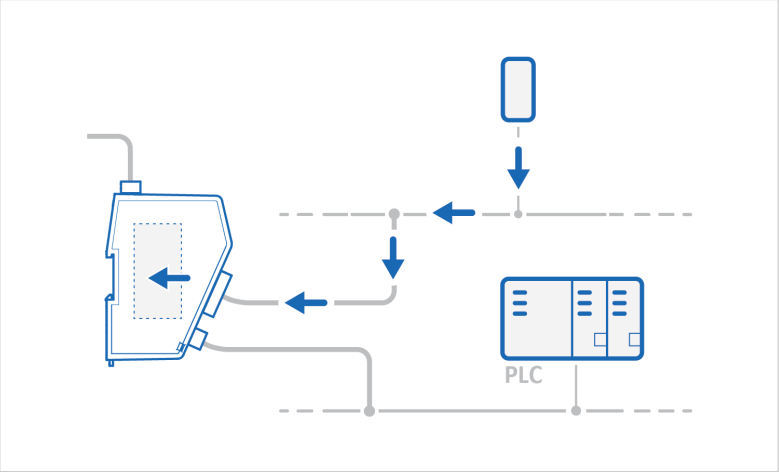


Figure 6. Process data traffic from nodes to client

Request process data from the serial subnetwork nodes, specified in the Communicator configuration, and make the process data available on the server interface and for the high level network client device.

Transfer Process Data

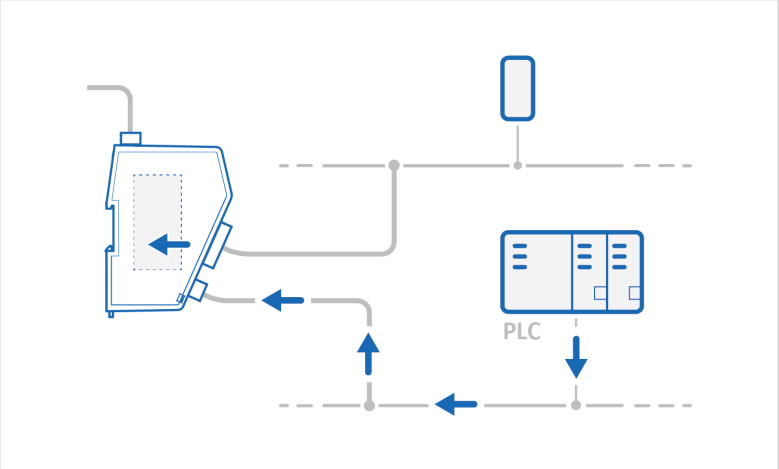


Figure 7. Process data traffic from client to nodes

Transfer process data from the high level network client device and make it available on the server interface and for the serial subnetwork nodes included in the configuration.

5.3. How the Data Exchange Works

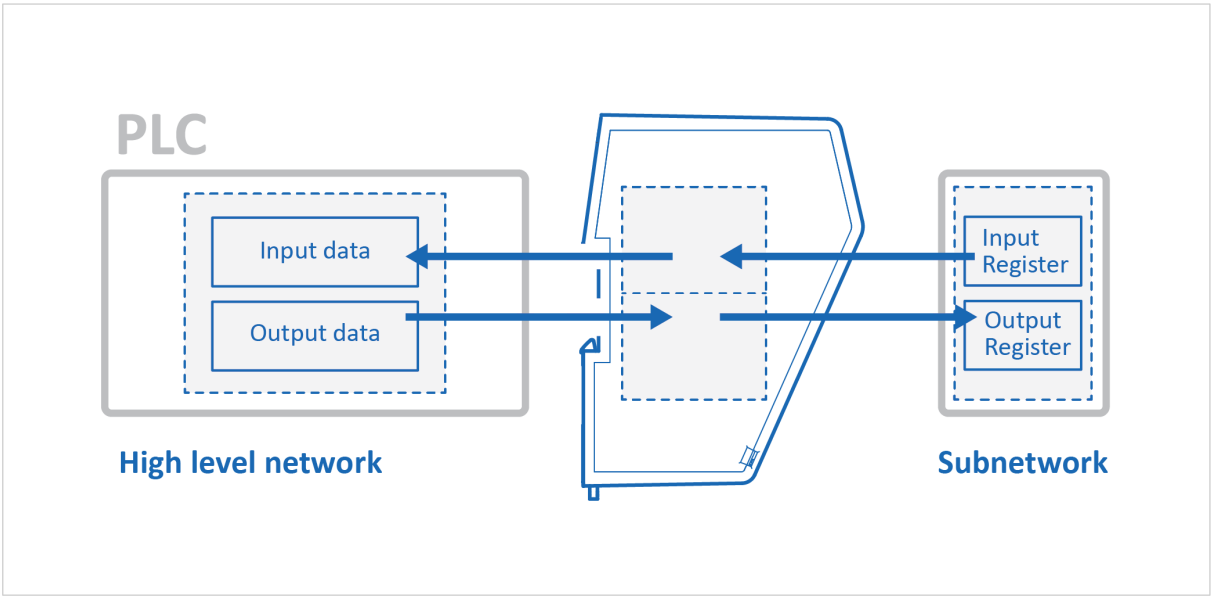


Figure 8. The Communicator internal memory areas

The data exchanged between the Communicator and the serial subnetwork and the high level network resides in the Communicator internal memory buffer.

To exchange data with the serial subnetwork, the high level network reads and writes data to the Communicator internal memory buffer.

The same memory locations are exchanged on the serial subnetwork.

The memory locations are specified when configuring the Communicator using the Communicator built-in web interface.

Input Data

The Input data area is read by the high level network.

Output Data

The Output data area is read/written by the high level network.

5.4. Data Integrity

A snapshot of the process data buffer between the Client and the server interface is used during the operation of executing all the transactions within one cycle.

When the cycle is completed, the process data available on the server interface is updated and a new snapshot is created for the next cycle.

6. Installation

6.1. External Parts

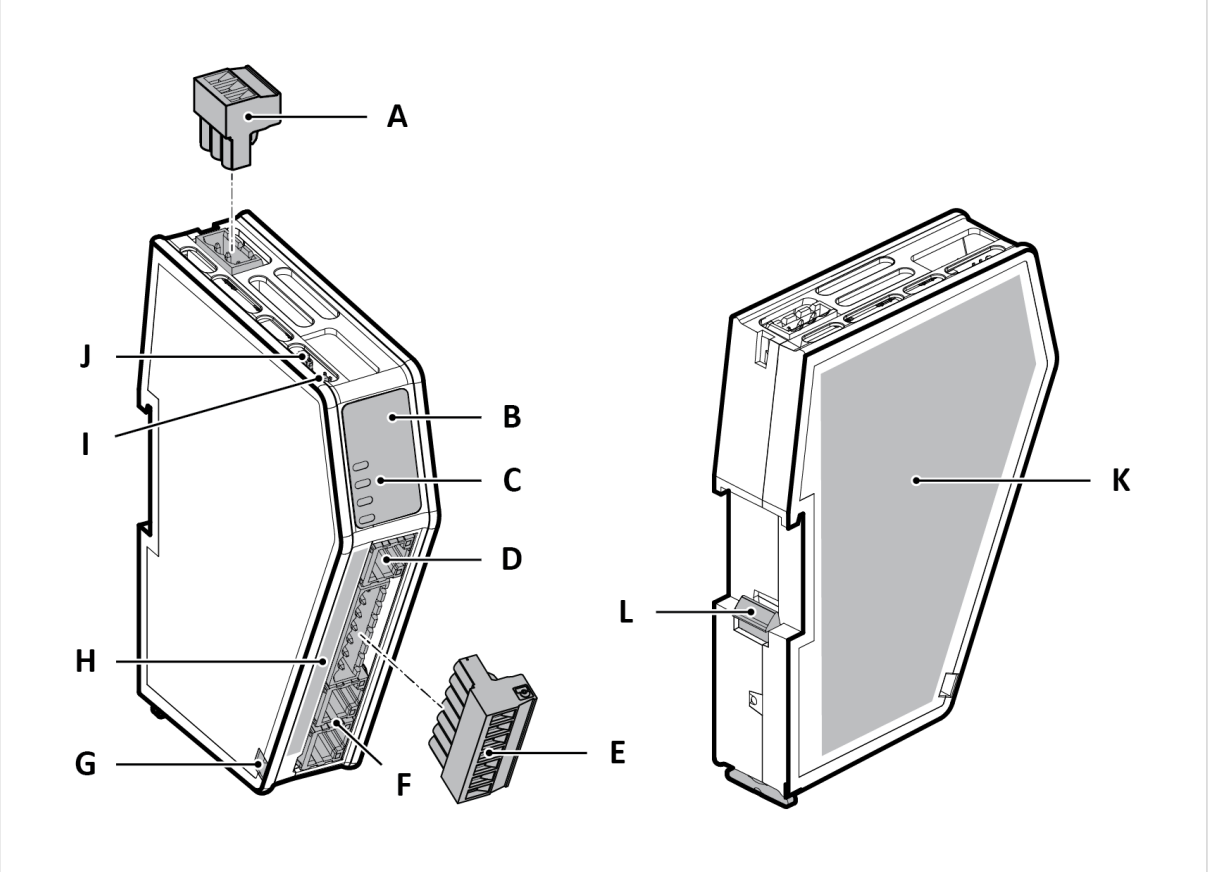


Figure 9. External parts

A. Power connector
B. Label with LED designation
C. Status LEDs
D. Configuration port
E. 7-pin connector
F. EtherCAT port x 2
G. Cable tie mount
H. Laser engraved connectors designation
I. Security switch
J. Factory reset button
K. Laser engraved label with product information
L. DIN rail locking mechanism

6.2. Connector Port Guide

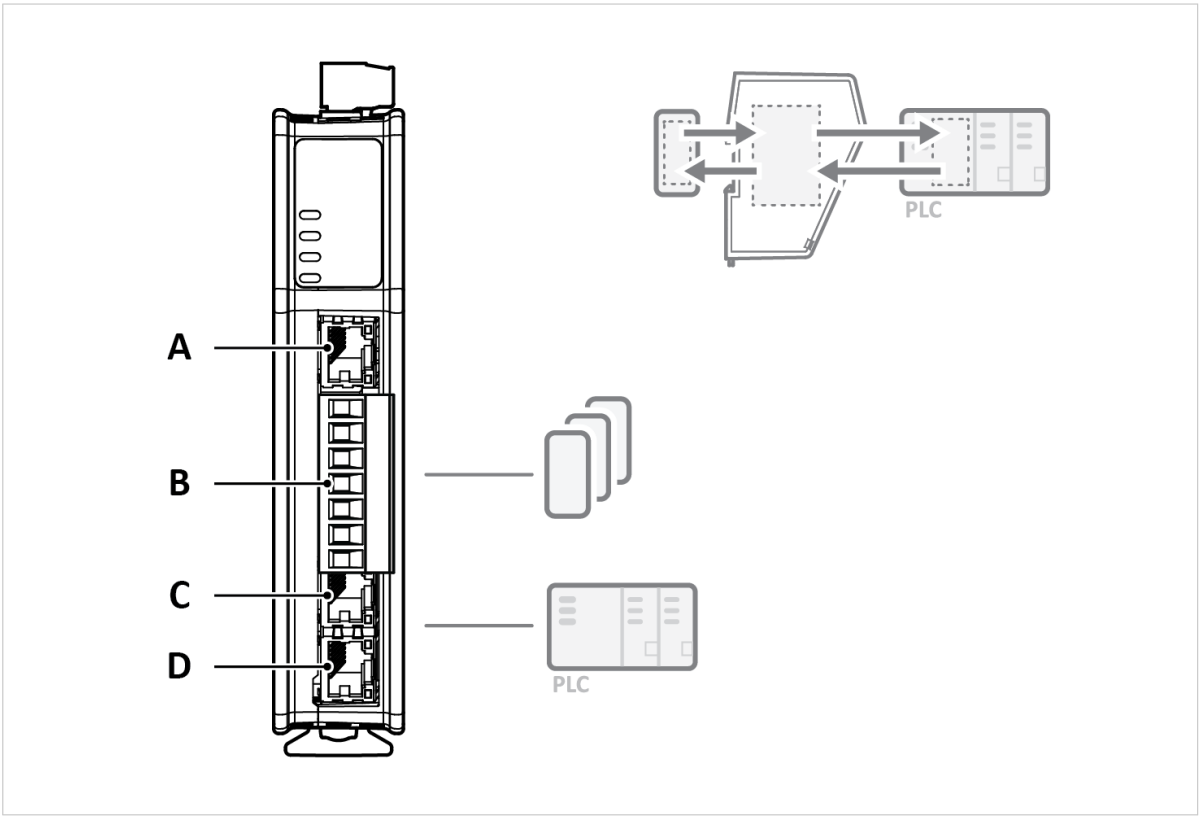


Figure 10. Communicator connector ports

Position	Port Number	Connector	Port Usage
A	X1	Ethernet	Configuration port
B	X3	Serial	Serial RS-232/485 Device
C	X2.1	Ethernet	EtherCAT network Input
D	X2.2	Ethernet	EtherCAT network Output

See also Connect to EtherCAT Network (page 18) and Connect to Serial RS232/RS485 Subnetwork (page 19).

6.3. DIN Rail Mounting

IMPORTANT
The equipment must be electrically grounded through the DIN rail for EMC compliance. Make sure that the equipment is correctly mounted on the rail and that the rail is properly grounded.

IMPORTANT
To physically secure networks and equipment and to prevent unauthorized access, it is recommended to install the equipment in a locked environment.

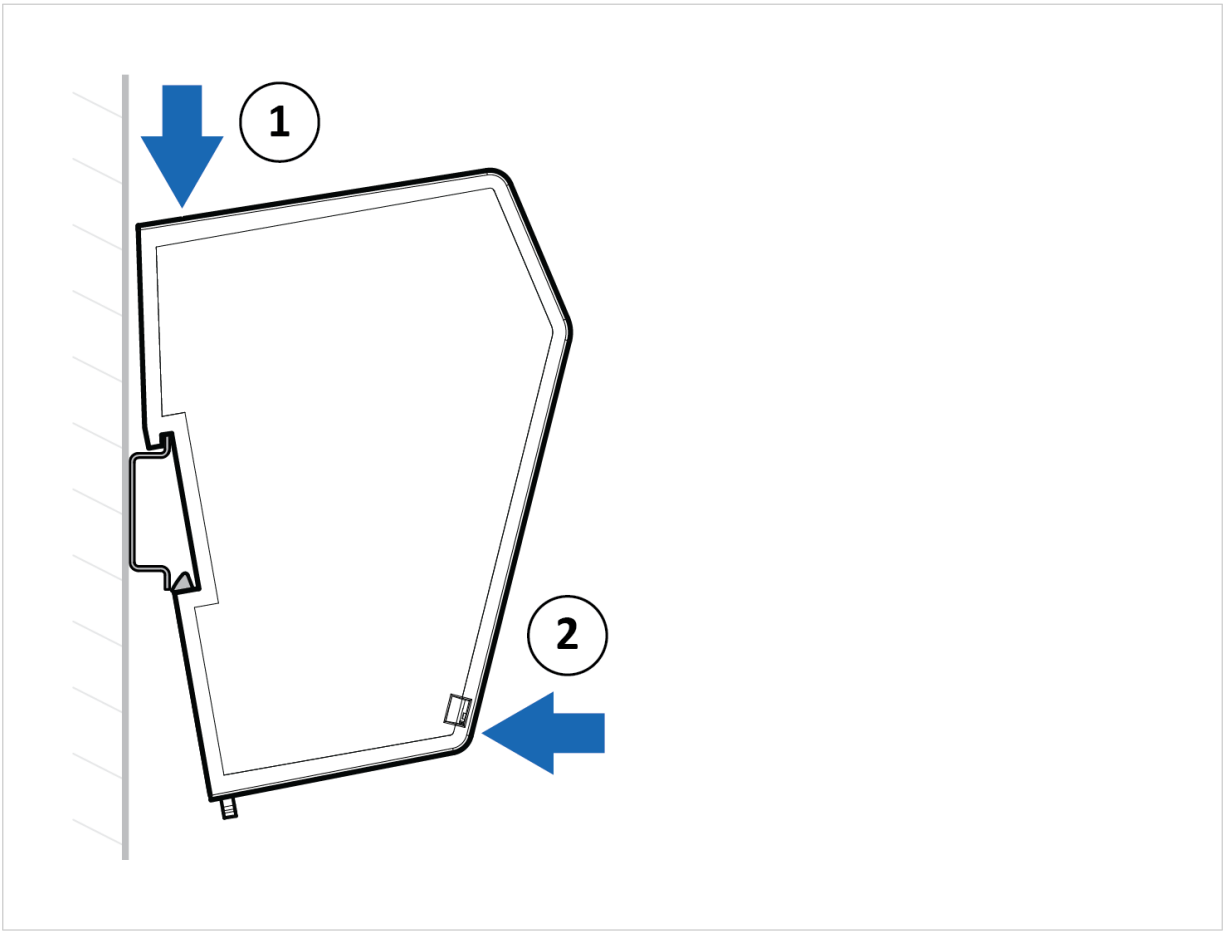


Figure 11. Attach the Communicator on the DIN rail

To attach the Communicator on the DIN rail:

1. Insert the upper end of the DIN rail clip into the DIN rail.
2. Push the bottom of the DIN rail clip into the DIN rail.

6.4. Connect to EtherCAT Network

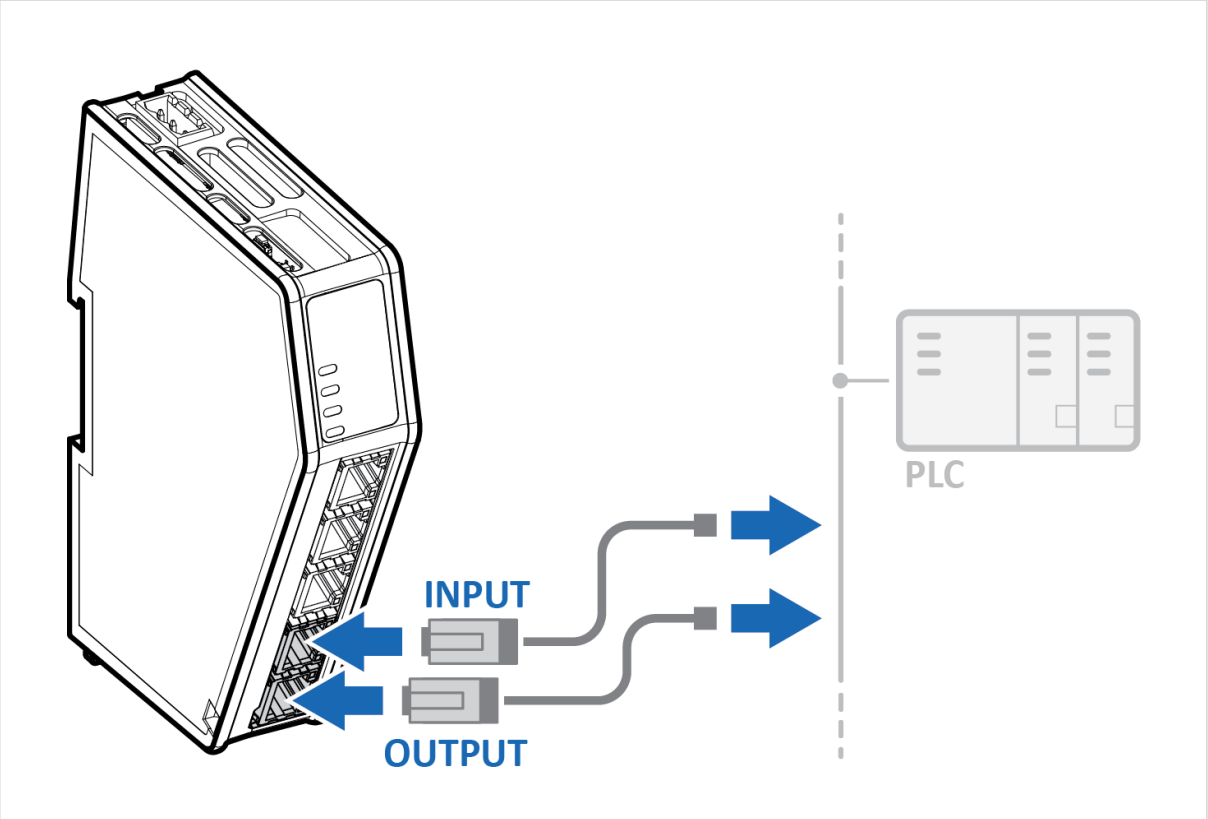
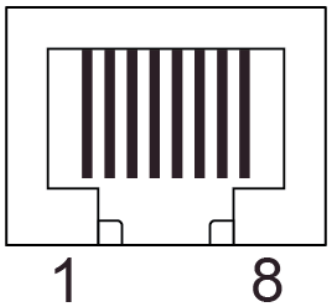


Figure 12. Connect to EtherCAT network

1. Connect the Communicator to your EtherCAT network.
   - Upper EtherCAT Connector is Input.
   - Lower EtherCAT Connector is Output.

RJ45 Connector	Pin	Description
	1	TD+
	2	TD-
	3	RD+
	4	Not used
	5	Not used
	6	RD-
	7	Not used
	8	Not used

To Do Next

Connect the Communicator to the serial subnetwork and to power.

Check LED status, refer to Communicator LED Indicators (page 108).

6.5. Connect to Serial RS232/RS485 Subnetwork

NOTE
Use minimum 90 oC copper (Cu) wire only.

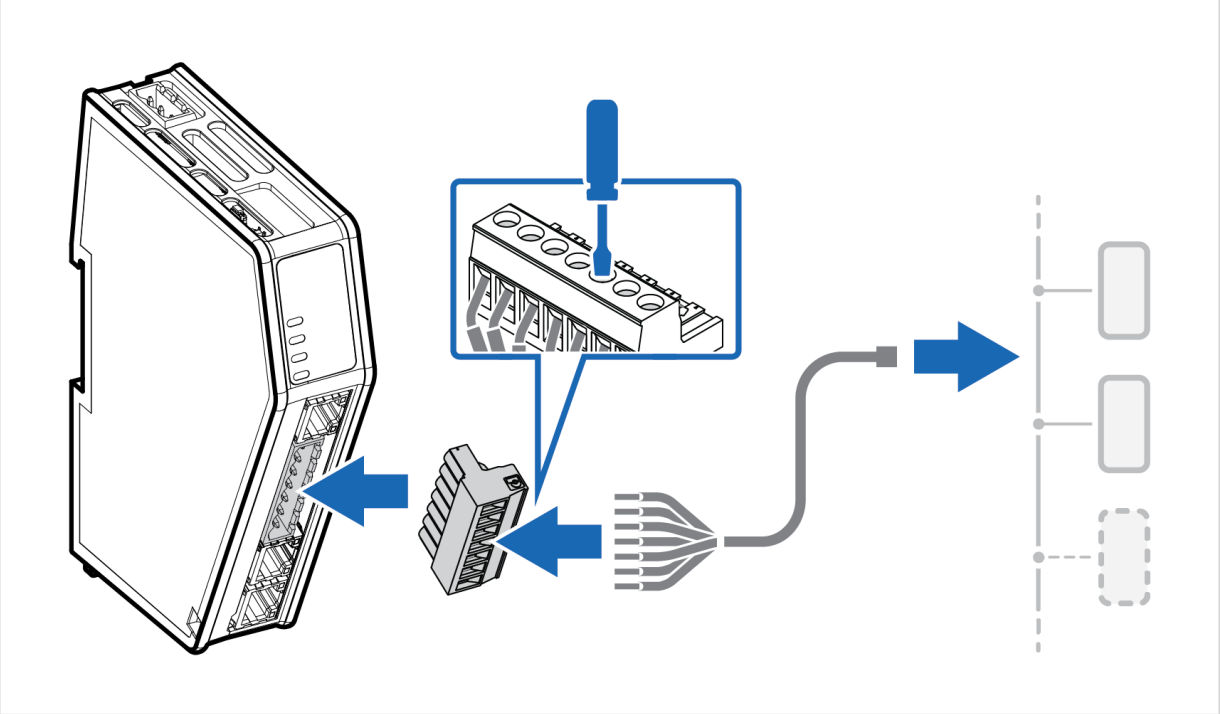
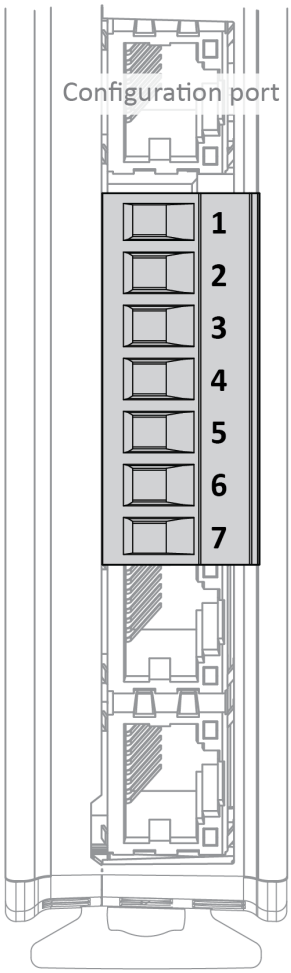


Figure 13. Connect to serial RS232/RS485 subnetwork

1. Insert the cable wires into the 7-pin connector and tighten the wire clamp screws.

7-pin connector	Pin	Signal
	1	+5 V OUT
	2	RS485+ A
	3	RS485- B
	4	Signal GND
	5	Functional Earth (FE)
	6	RS232 Tx Transmit Data
	7	RS232 Rx Receive Data

2. Connect the 7-pin connector to the Communicator.
3. Connect the Communicator to your serial subnetwork.

To Do Next

Connect the Communicator to the EtherCAT network and to power.

Check LED status, refer to Communicator LED Indicators (page 108).

6.6. Connect to Power

CAUTION
Ensure that the power supply is turned off before connecting it to the equipment.

IMPORTANT
Using the wrong type of power supply can damage the equipment. Ensure that the power supply is connected properly and of the recommended type.

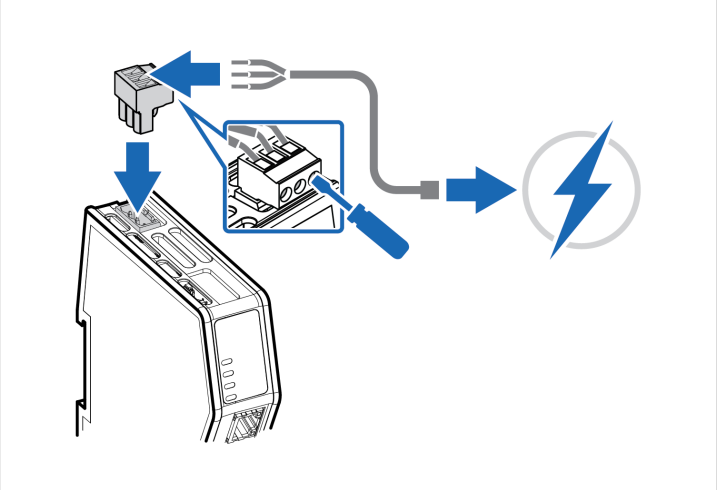
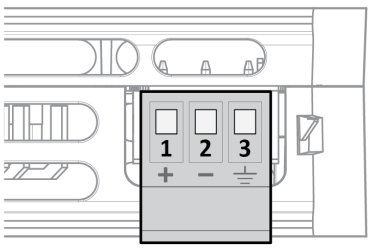


Figure 14. Connect to power

Power Connector Pinout

Power port	Pin	Description
	1	12-30 VDC Power Connector
	2	Ground (GND)
	3	Functional Earth (FE)

Procedure

1. Insert the cable wires to the terminal block and tighten the wire clamp screws.
2. Connect the terminal block to the Communicator.
3. Connect the Communicator to a power supply.
4. Turn on the power supply.

6.7. Security Switch

IMPORTANT
After completing the configuration of the Communicator, lock the security switch to prevent unauthorized access to the Communicator built-in web interface.

When the security switch is in its locked position, the Communicator built-in web interface cannot be accessed, and the Communicator cannot be configured using the built-in web interface. Network specific parameters, configured via the PLC is still available.

To Lock and Unlock the Security Switch

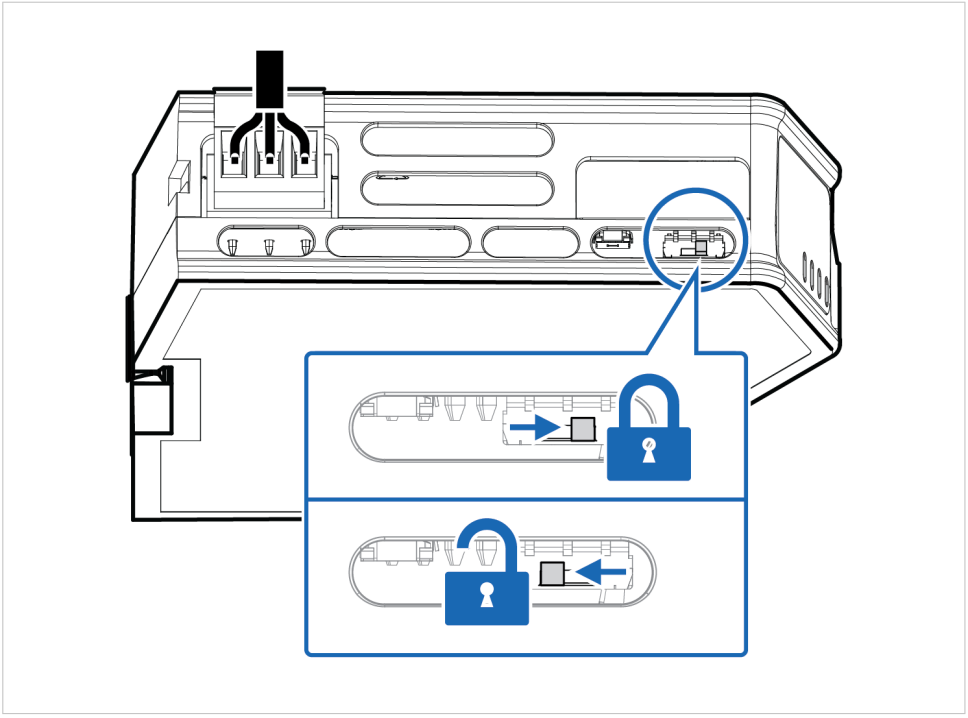


Figure 15. Security switch in locked and unlocked position

Use a pointed object, such as a ballpoint pen.

- To **lock** the security switch, push the toggle towards the **Communicator front**.
- To **unlock** the security switch, push the toggle towards the **Communicator back**.

Security Switch Status LED

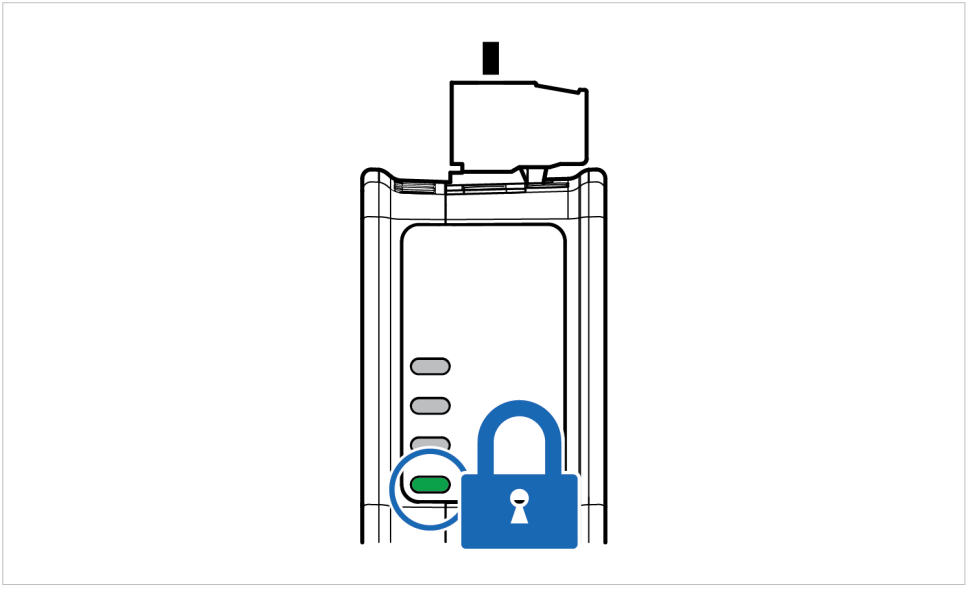


Figure 16. Security switch locked status LED

When the security switch is in its:

- locked position, the security switch status LED turn solid green.
- unlocked position, the security switch status LED is turned off.

6.8. Lock the Cables

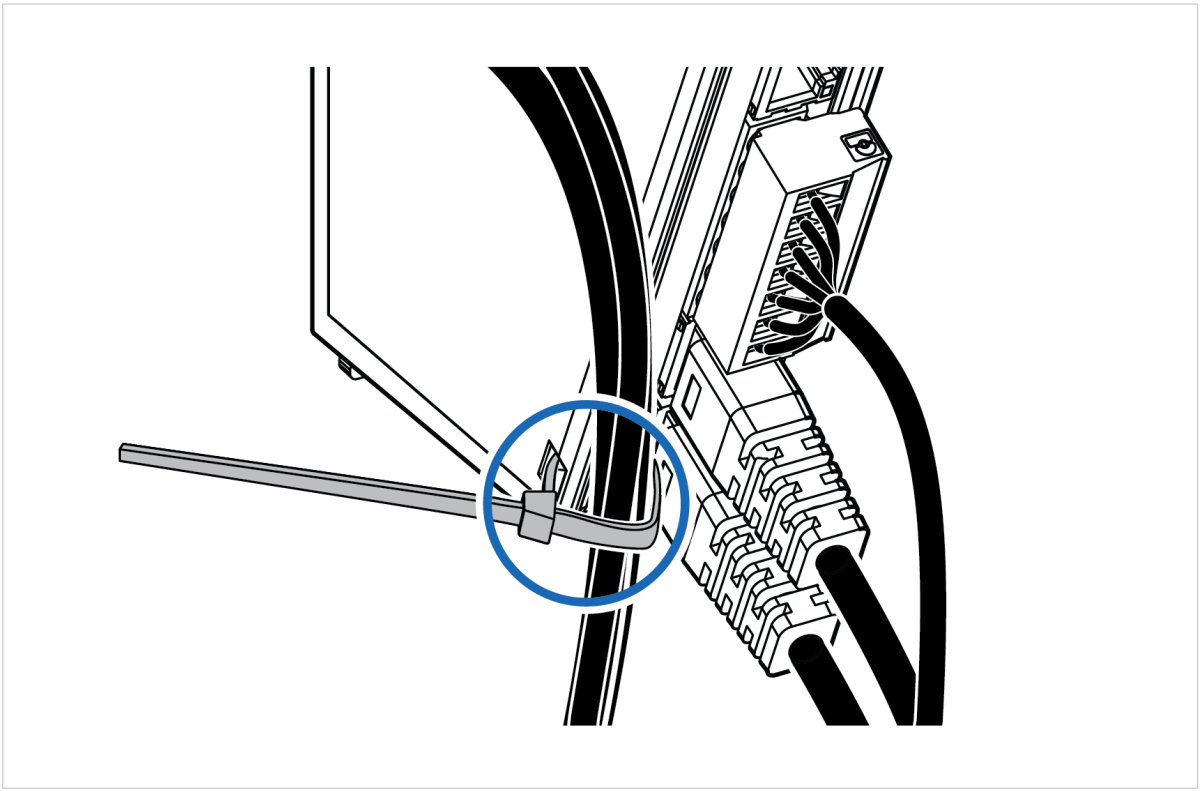


Figure 17. Lock the cables

To strain relieve the cables, place a cable tie in the holder and lock the cables.

6.9. DIN Rail Demount

Before You Begin

IMPORTANT
Be careful when removing the Communicator from the DIN-rail. If not removed properly, the DIN rail locking mechanism and the product cover can break.

Have a flat-blade screwdriver, size 5.5 mm, available.

Procedure

Remove the Communicator from the DIN rail:

1. Insert the screwdriver into the Communicator DIN rail locking mechanism.
2. To unlock the Communicator DIN rail locking mechanism, turn the screwdriver clockwise.

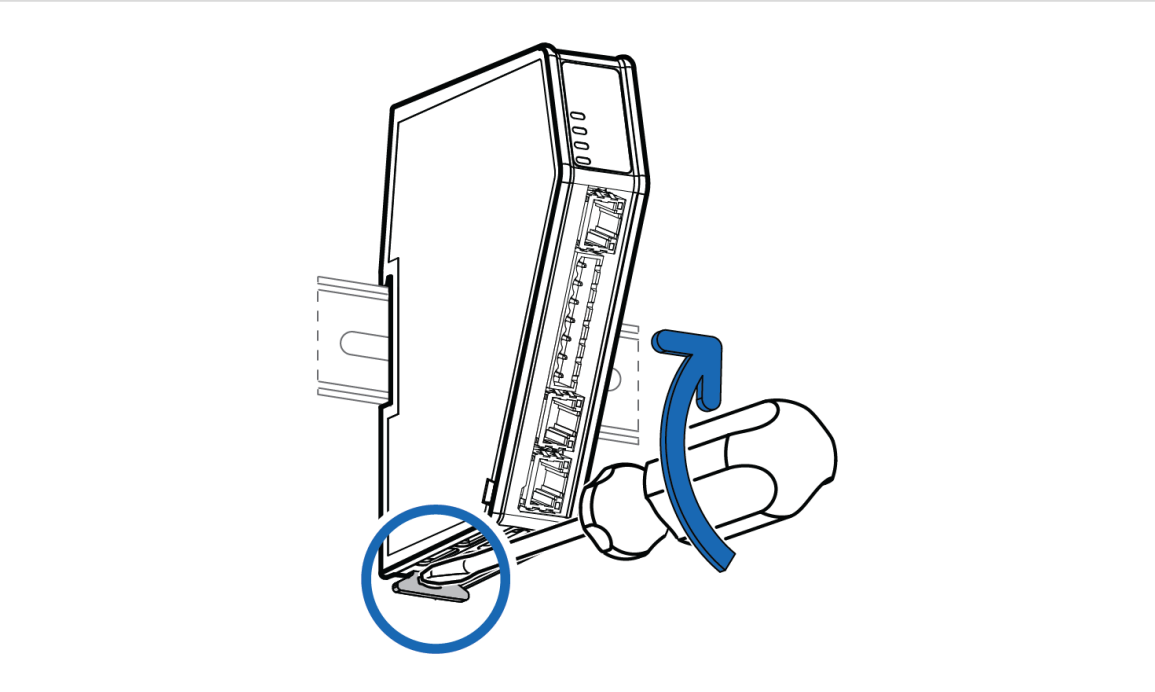


Figure 18. Unlock the Communicator

3. Hold the screwdriver in the DIN rail locking mechanism while you unhook the Communicator from the DIN rail.

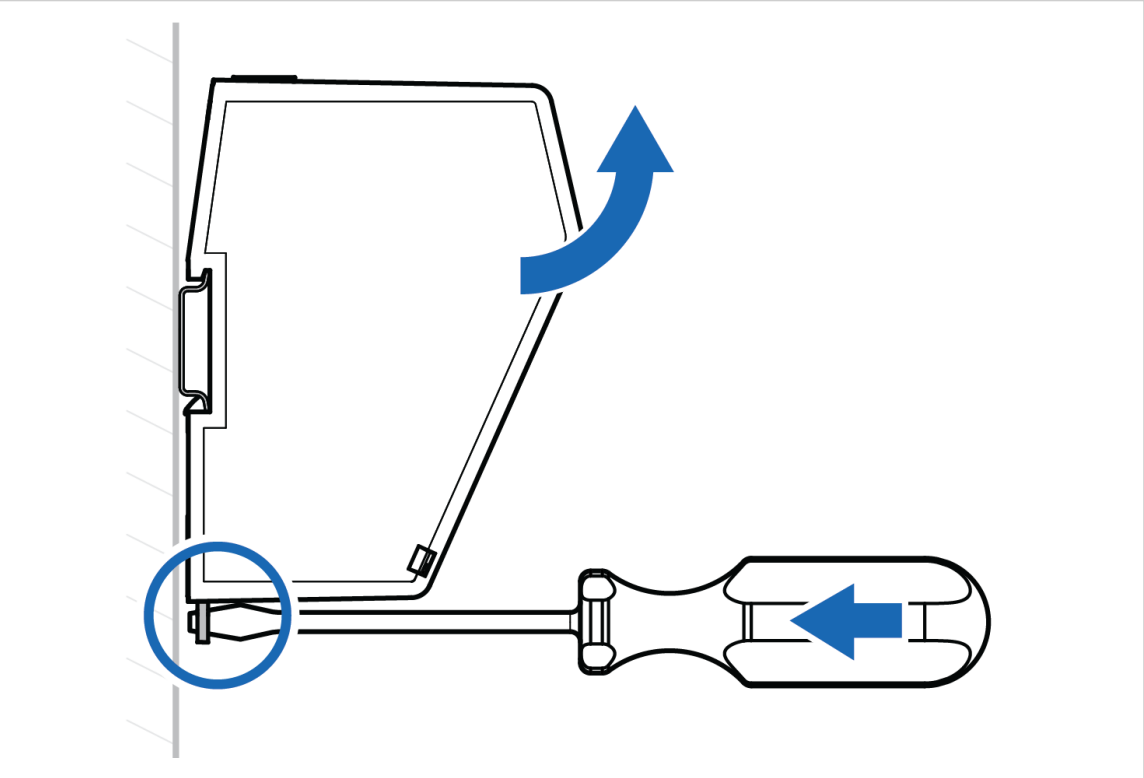


Figure 19. Unhook the Communicator

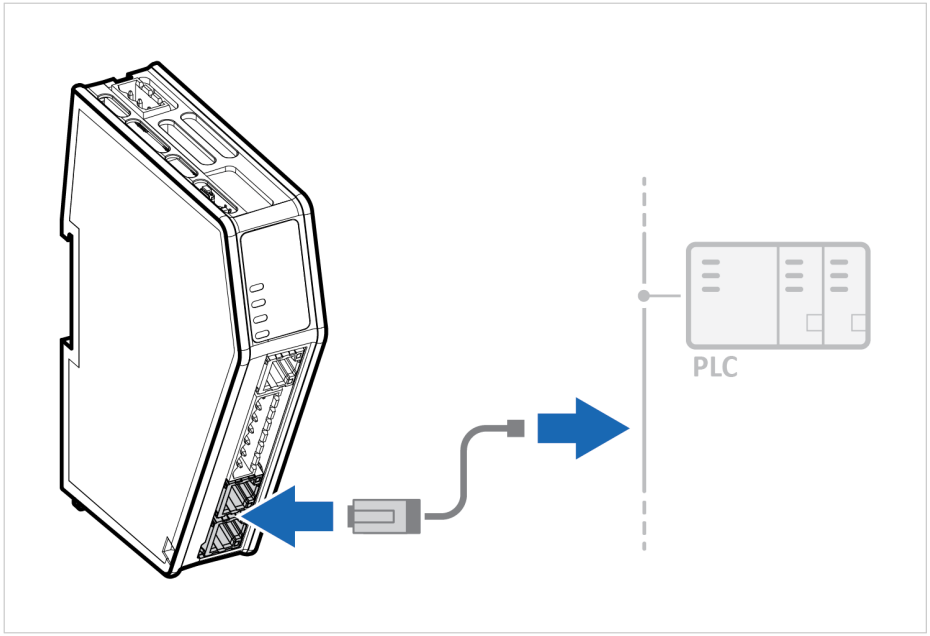
7. Configuration Quick Guide

This section is intended to give you a brief overview of the tasks you need to perform to configure the Communicator.

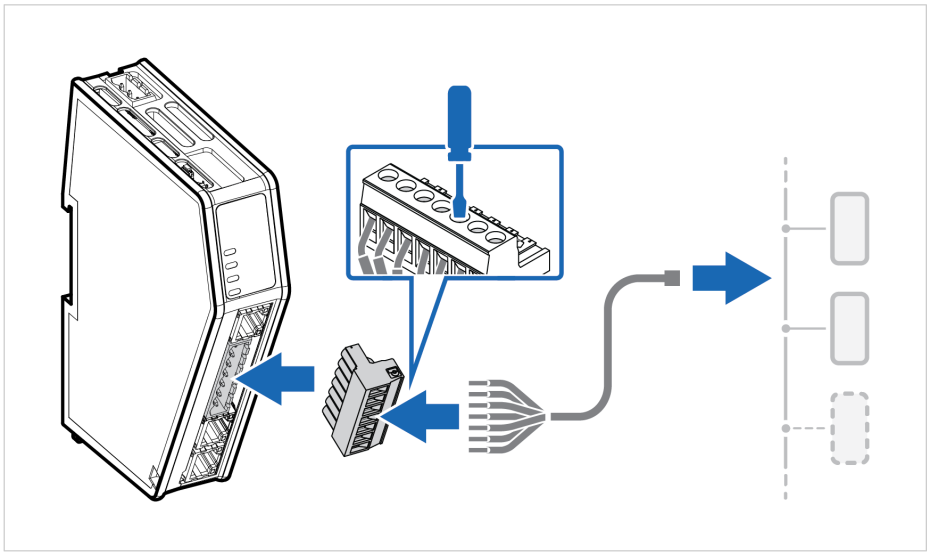
For detailed information, please refer to Communicator Configuration (page 37) .

7.1. Prepare Configuration

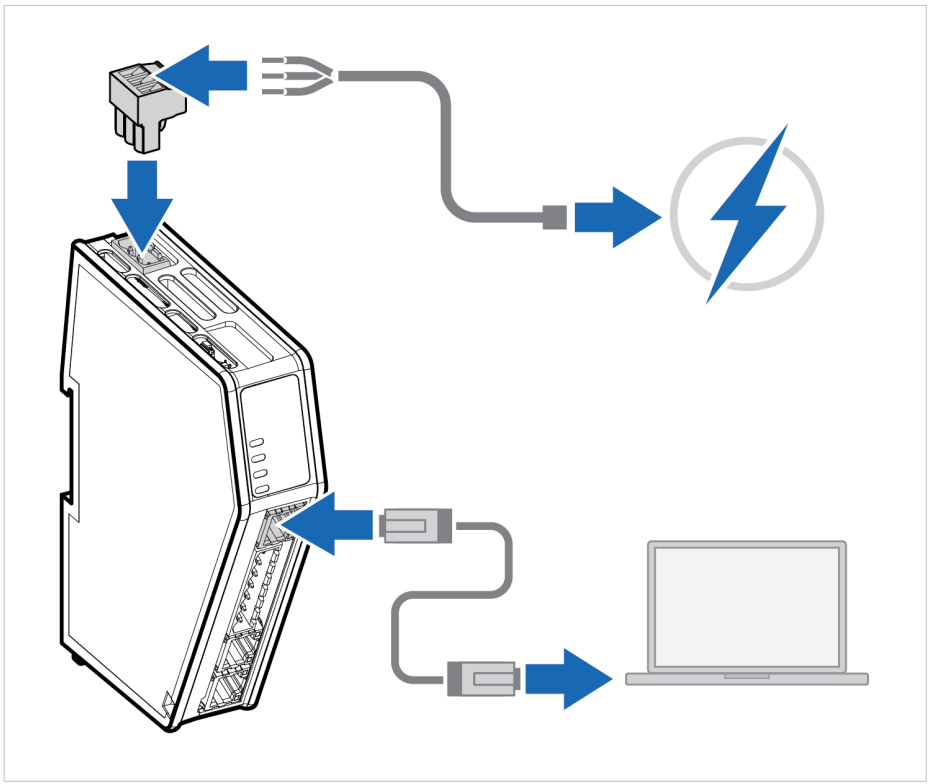
1. **Connect Communicator to the EtherCAT network.**



2. **Connect the Communicator to the serial RS232/RS485 subnetwork**



3. **Connect to PC and power**



a. Connect an Ethernet cable between the Communicator configuration port and your PC.

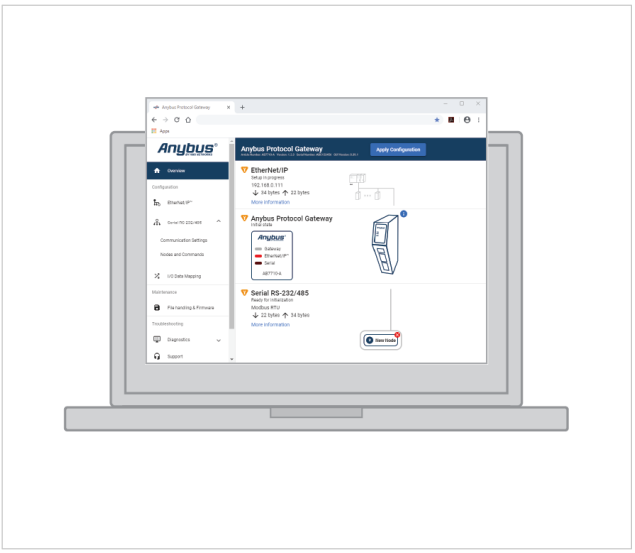
b. Connect the Communicator to a power supply.

4. **Find the Communicator on your PC**

The Communicator default IP address is 192.168.0.10.

Option 1	Option 2
 On the PC accessing the Communicator built-in web interface, set a static IP address within the same IP address range as the Communicator IP address.	 Change the IP address on the Communicator configuration port to one within the same IP address range as your PC. Use the software application HMS IPconfig to find the Communicator default IP address on your PC. Download the installation files and user documentation from www.anybus.com/support.

5. **Access the Communicator built-in web interface**



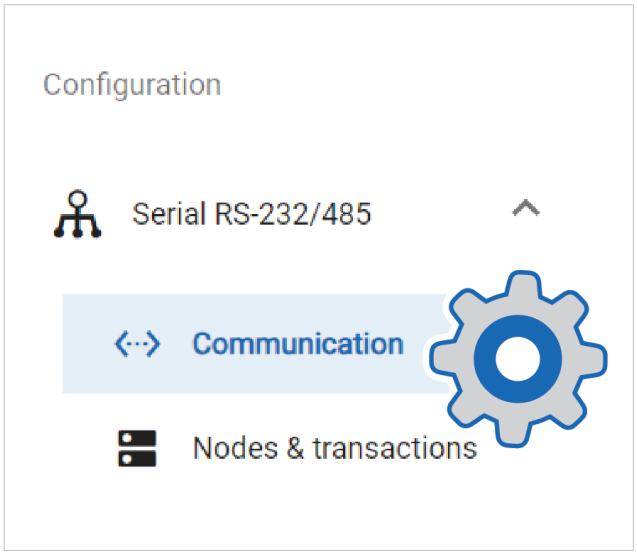
Open the Communicator built-in web interface in HMS IPconfig or enter the Communicator IP address in your web browser.

The Communicator built-in web interface overview page opens in your browser.

7.2. Setup New Configuration

Follow these steps to setup a new Communicator configuration.

1. **Subnetwork configuration**

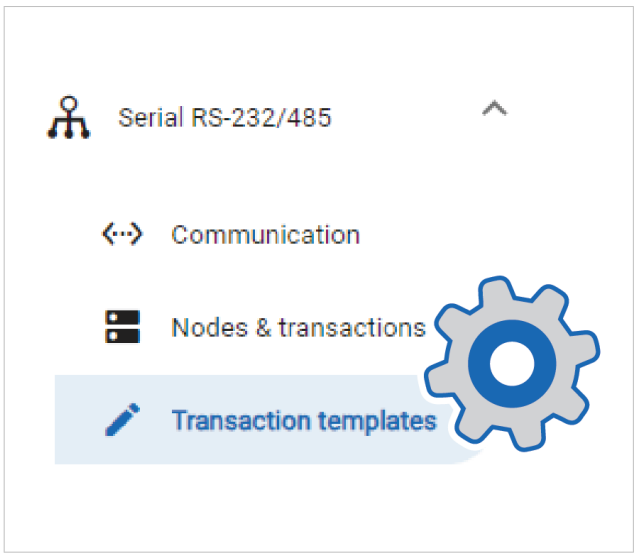


On the **Communication** page:

a. Select a serial protocol: **Modbus RTU** (default), **Custom Request/Response** or **Custom Produce/Consume**.
For information about the serial protocol types, see Serial Protocol Types (page 9).

b. Configure the basic settings Physical standard:, Baud rate, Data bits, Parity and Stop bits.

2. **Create Transaction Templates**

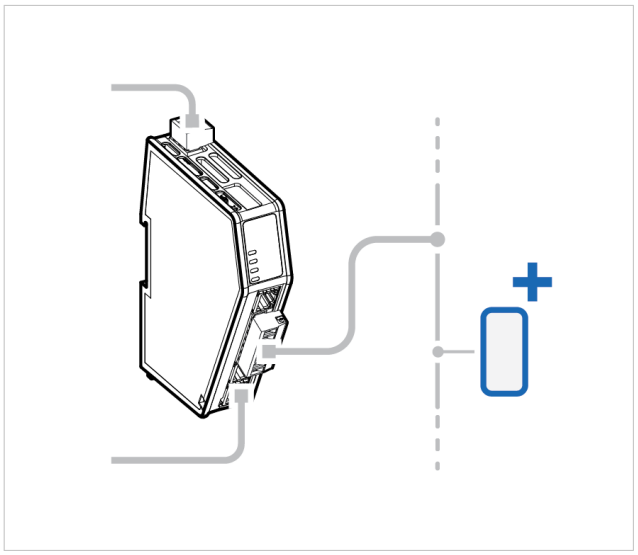


This step applies when the serial protocol **Custom request/response** is selected.

On the **Transaction templates** page: Add a transaction template and configure the template settings.

Repeat until you have added and configured all your transaction templates.

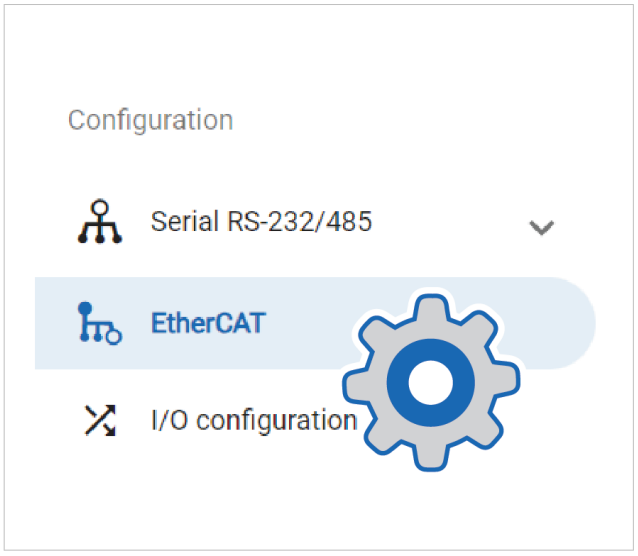
3. **Add Nodes and Transactions**



On the **Nodes & transactions** page:

a. Add a node and configure the node settings.
b. Add commands to the node and configure the command settings.
c. Repeat until you have added and configured all your nodes.

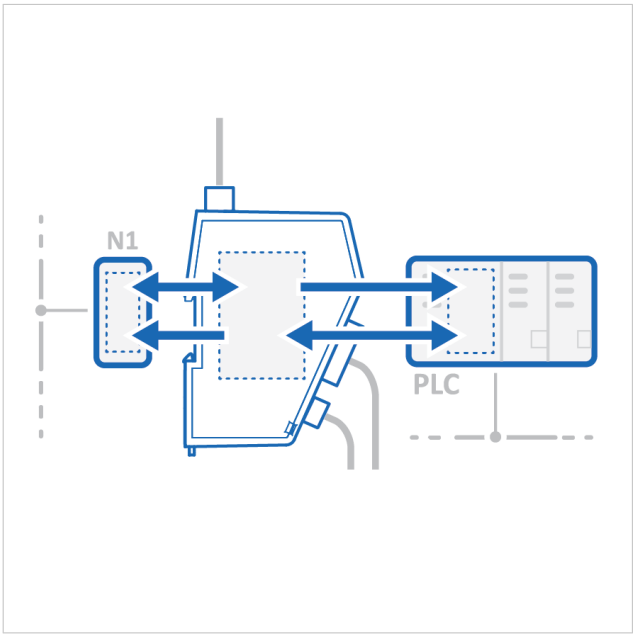
4. **High level network configuration**



On the **EtherCAT** page:

1. Use Automatic I/O sizes provided by the subnetwork or choose to set them manually.

5. **I/O Data Mapping**

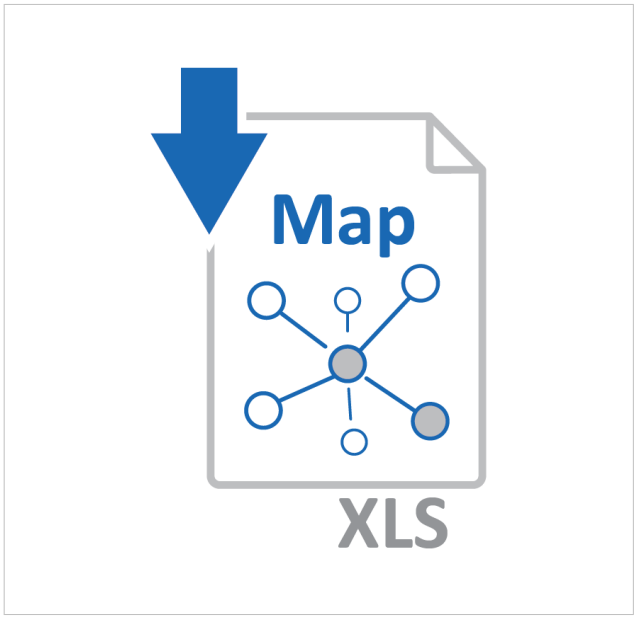


The commands you added to the nodes are automatically mapped to the Communicator internal memory area.
View the added nodes and commands on the **I/O configuration** page.

7.3. PLC Configuration

In the Communicator built-in web interface:

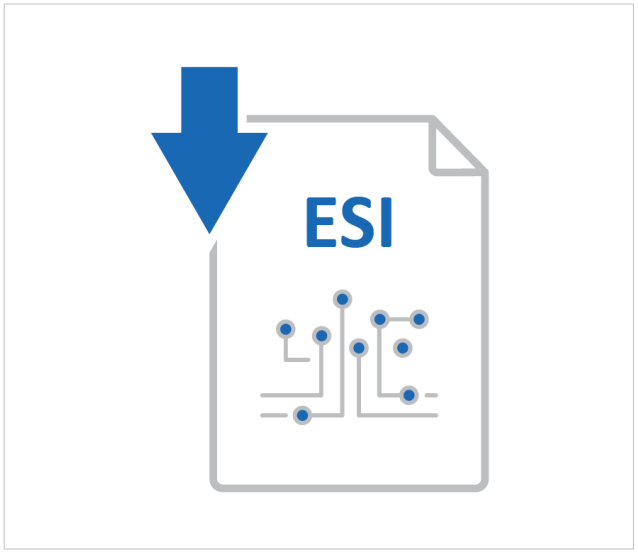
1. **Export I/O data map**



When you configure the communication between the Communicator and the PLC, you can use the I/O data map as a specification to ensure that the commands match.

On the **I/O configuration** page: You can download the I/O data mapping in a spreadsheet to your PC.

2. **Download ESI File**

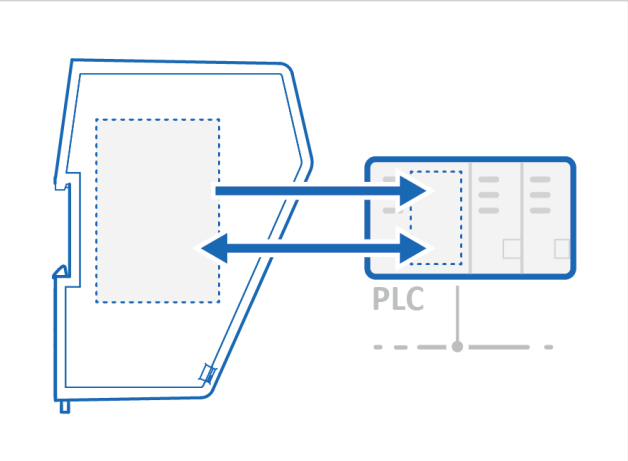


Option if the PLC program requires a ESI (EtherCAT SubDevice Information) file.

On the **EtherCAT** page: Download the ESI file to your PC.

In the PLC program:

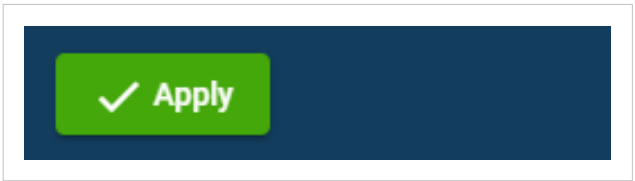
1. **Import product file**
   Option if the PLC program requires a ESI (EtherCAT SubDevice Information) file.
   Import the ESI file into your PLC project.
2. **Configure the communication**



Configure the PLC to communicate with the Communicator according to the I/O data map created in the Communicator.

7.4. Verify Operation

1. **Apply the configuration**



When you have completed and verified the configuration, click **Apply** for the settings to take effect.

2. **Verify status and LED indications**

On the **Home** page:
Monitor the Communicator, network and node status.
You can also view the Communicator LED indications remotely.

3. **Verify and monitor communication**

In **Diagnostics**, use the:

- **Serial RS-232/485** page to verify that the serial commands are sent and received by the Communicator.
- **I/O data** page to monitor how the data flow between the **Serial RS-232/485** side and the **EtherCAT** side, including any configured endian conversions.
- **Event log** page to detect failures and unexpected behavior over time.

8. Communicator Configuration

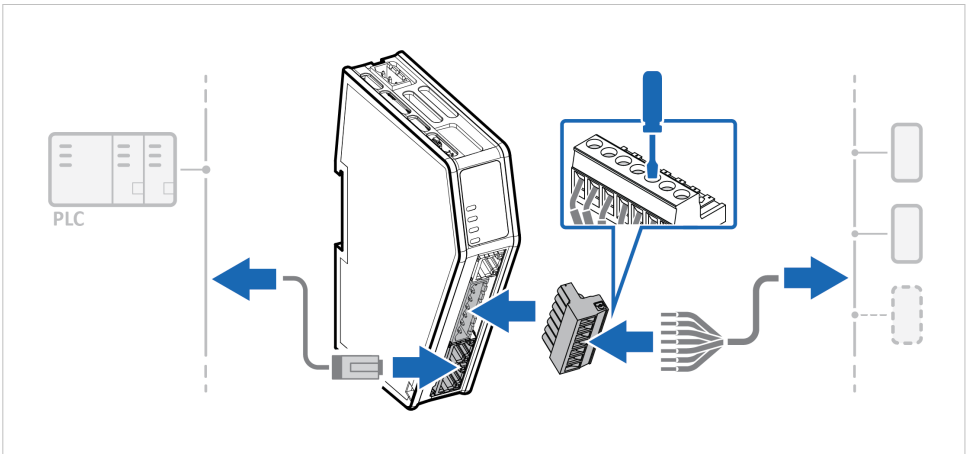
This section is intended to give you detailed information about the tasks you need to perform to setup a new Communicator configure.

For a more brief overview of the configuration steps, please refer to the Configuration Quick Guide (page 27).

8.1. Connect the Communicator

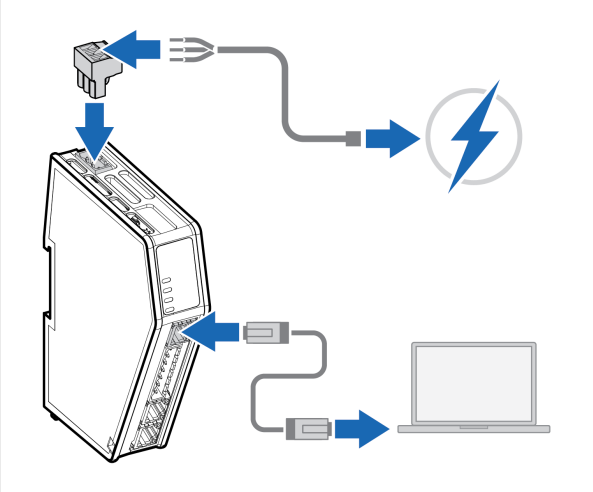
Procedure

Connect to Serial RS-232/485 network and EtherCAT network



1. Connect the Communicator to the high level network.
2. Connect the Communicator to the subnetwork.

Connect to PC and Power



1. Connect an Ethernet cable between the Communicator and your PC.
2. Connect the Communicator to a power supply.

8.2. Access the Built-In Web Interface from HMS IPconfig

Before You Begin

Download the software application HMS IPconfig installation files and user documentation from www.anybus.com/support.

NOTE
The Communicator default IP address is 192.168.0.10.

NOTE
To access the Communicator built-in web interface, ensure that Port 80 TCP is open in your Firewall. This applies to any Firewall between the web browser and the gateway.

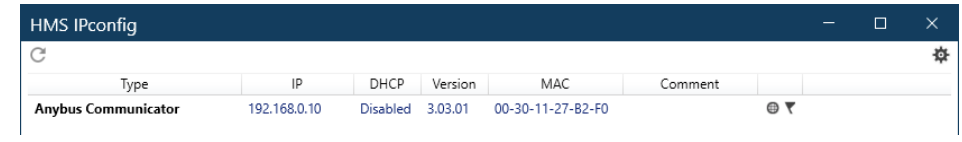
NOTE
To access the Communicator built-in web interface from HMS IPconfig, ensure that Port 3250 UDP is open in your PC Windows Firewall.

NOTE
Ensure that the security switch is unlocked. HMS IPconfig cannot configure the Communicator if the security switch is locked.

TIP
When you have accessed the Communicator built-in web interface, you can change the IP settings for the Communicator configuration port on the **System** > **Configuration port** page.

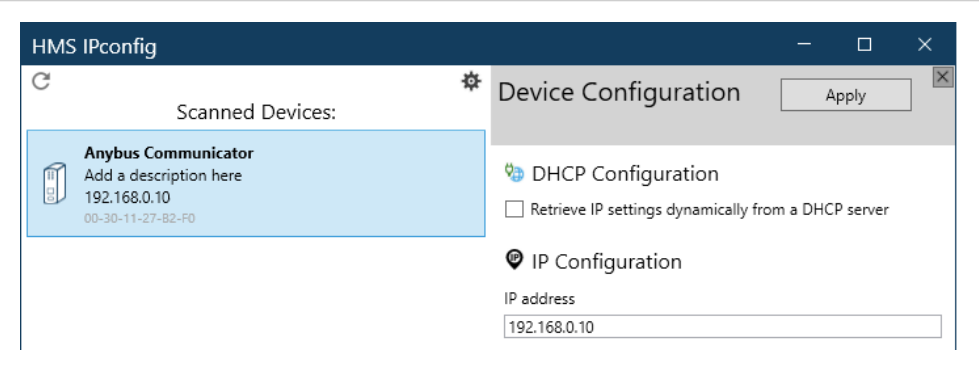
Procedure

1. Install HMS IPconfig on your PC.
2. Open HMS IPconfig.

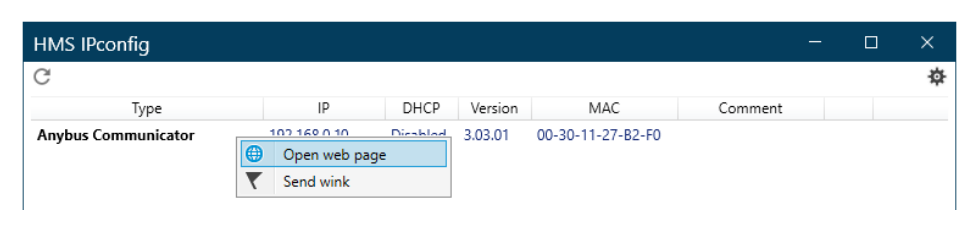


   - HMS IPconfig automatically starts scanning for compatible and active HMS devices.
   - Found HMS devices are added to the device list.
3. To open the settings pane, click on the Communicator in the device list.

4. Change the Communicator configuration port IP address to one within the same IP address range as your PC.

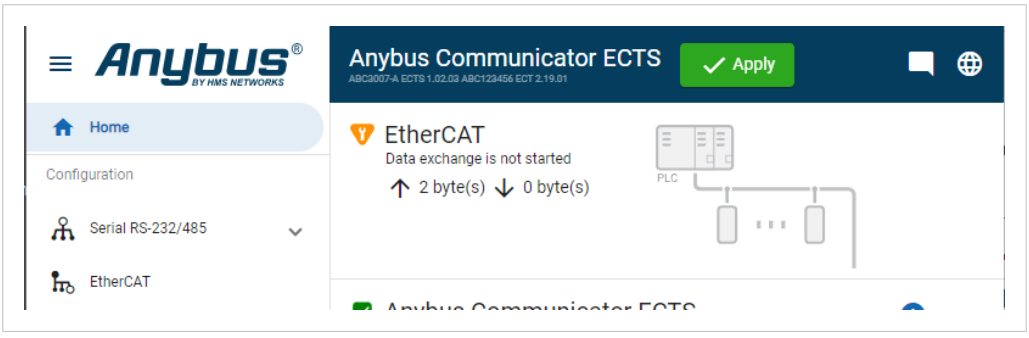


5. To open the **Open web page** built-in web interface, click Communicator.



Result

You are redirected to the Communicator built-in web interface **Home** page.



8.3. Access the Built-In Web Interface from a Web Browser

Before You Begin

NOTE
The Communicator configuration port default IP address is 192.168.0.10.

NOTE
To access the Communicator built-in web interface, ensure that Port 80 TCP is open in your Firewall. This applies to any Firewall between the web browser and the gateway.

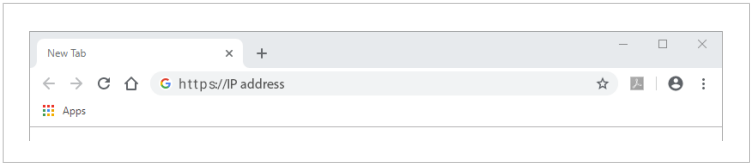
NOTE
When you change to a static IP address on your computer, internet access may be lost.

TIP
When you have accessed the Communicator built-in web interface, you can change the IP settings for the Communicator configuration port on the **System** > **Configuration port** page.

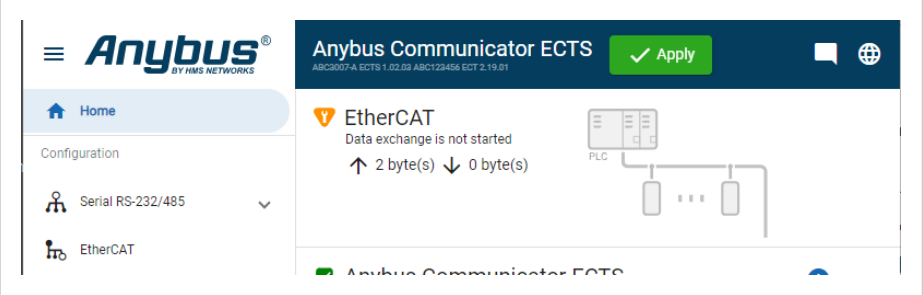
Procedure

1. On the PC accessing the Communicator built-in web interface, set a static IP address within the same IP address range as the Communicator IP address.

2. Open a web browser.
3. Click to select the **Address bar** and enter the Communicator IP address.



4. To open the built-in web interface **Home** page, press **Enter**.



8.4. Communicator Built-In Web Interface Overview

Use the Communicator built-in web interface to configure, maintain and troubleshoot the Communicator.

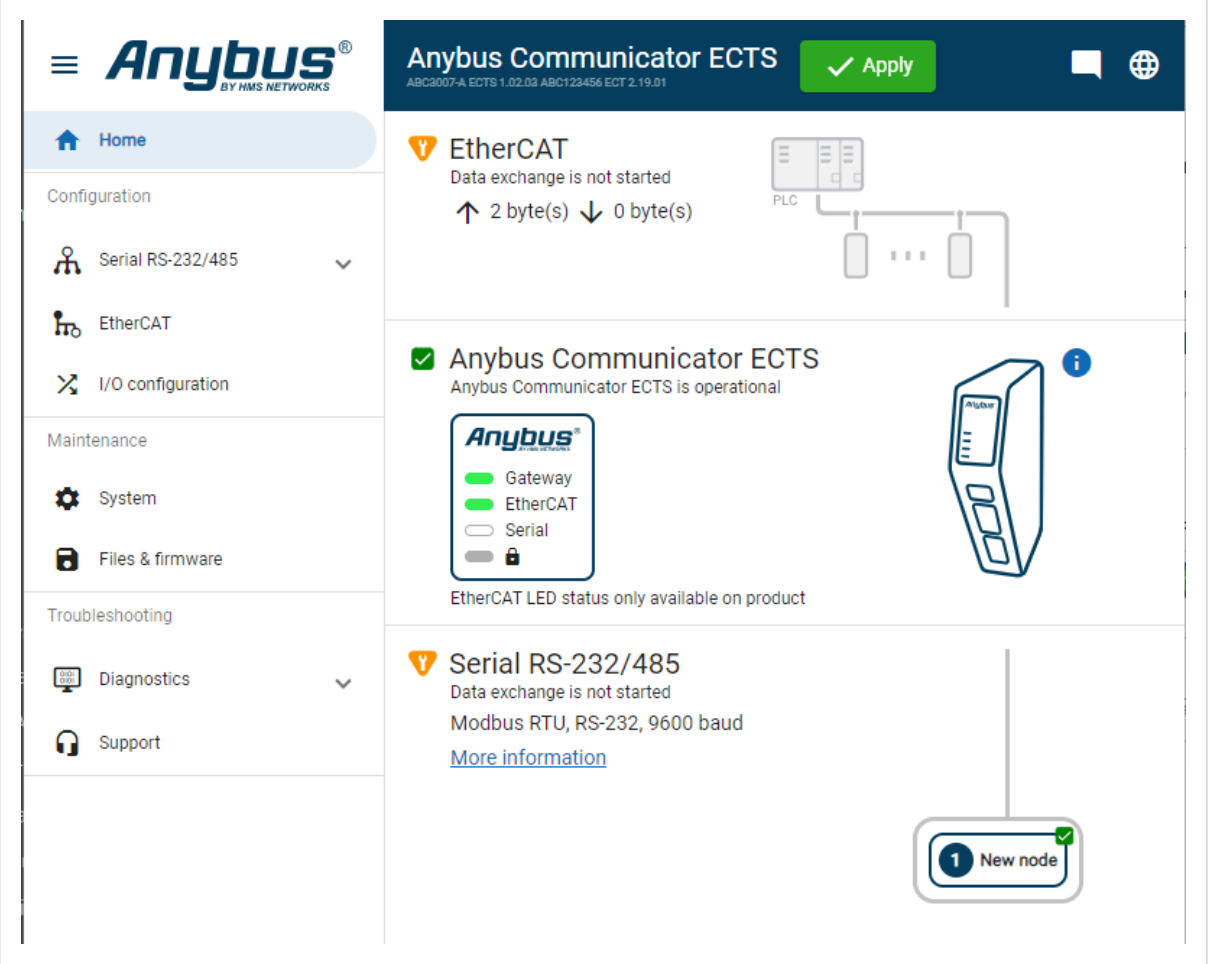


Figure 20. The Communicator built-in web interface Home page

Menu item	Description
Home	View the Communicator, network and node status.
Apply	After configuration changes are made and verified, press Apply to make the settings take effect.
Serial RS-232/485	Serial Subnetwork with Nodes. Select a Serial protocol, use Modbus RTU standard transactions or create your own transaction templates. Configure communication and add nodes and commands.
EtherCAT	High Level Network with Client. Configure the I/O Size.
I/O configuration	View the added commands mapped to the Communicator internal memory area.
System	Define how the device should behave if a serious error occurs. Configure the Communicator configuration port IP settings.
Files & firmware	Save settings in a configuration files, upload configuration files and upgrade firmware.
Diagnostics	Monitor and troubleshoot the Communicator.
Support	Contains Communicator product information, Anybus contact information, link to Anybus support website, and product file for download. Here you can generate a support package with product information, to send to your Anybus support technician.

8.5. General Subnetwork Settings

8.5.1. Communication Serial Protocol

Before You Begin

Before starting the configuration, select the Serial protocol you want to use:

- **Modbus RTU**, Default setting: Use for serial devices that conform to the Modbus communication specification.
- **Custom Request/Response**: Create your own custom request/response transactions.
  The transactions can be based on the Modbus communication specification or fully customized.
- **Custom Produce/Consume**: Create your own custom produce/consume transactions.

IMPORTANT

When changing the serial protocol, all settings are reset to default and all added nodes, transactions, and transaction templates are deleted.

Procedure

On the **Communication** page:

1. To choose a **Serial protocol**, select **Modbus RTU**, **Custom Request/Response** or **Custom Produce/Consume**.

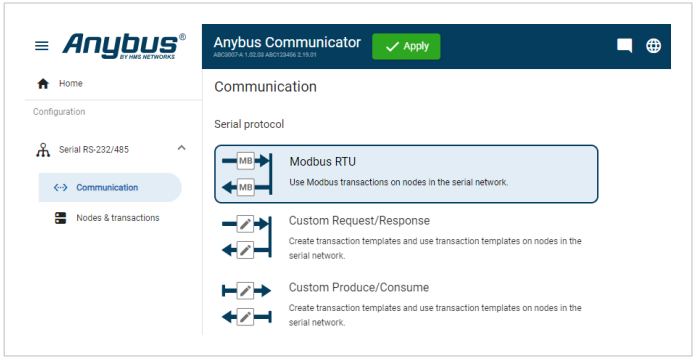


Figure 21. Communication, Serial protocol

2. To confirm the selected protocol, click **Change serial protocol**.

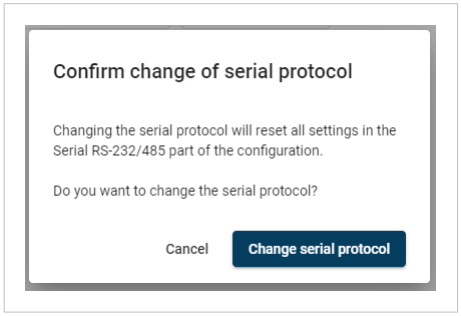


Figure 22. Confirm change of serial protocol

Apply Configuration

To apply the settings, click **Apply** in the built-in web interface header and follow the instructions.

8.5.2. Communication Basic Settings

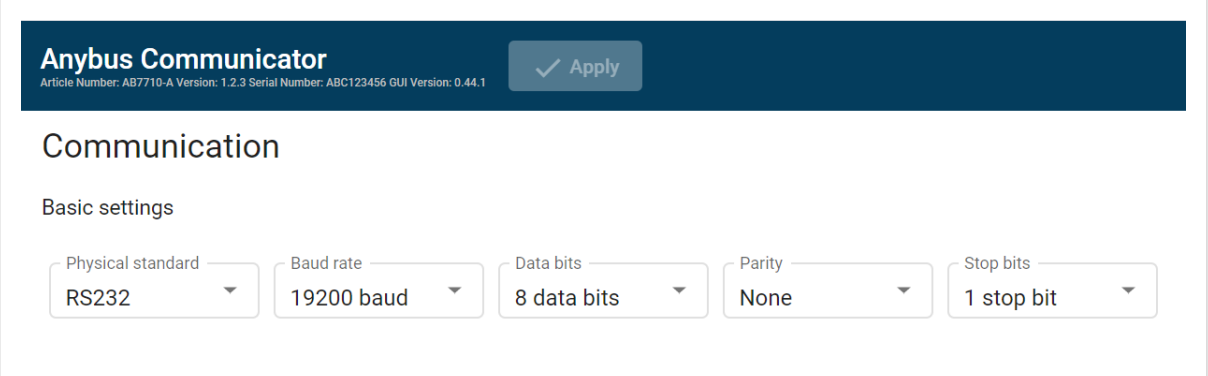
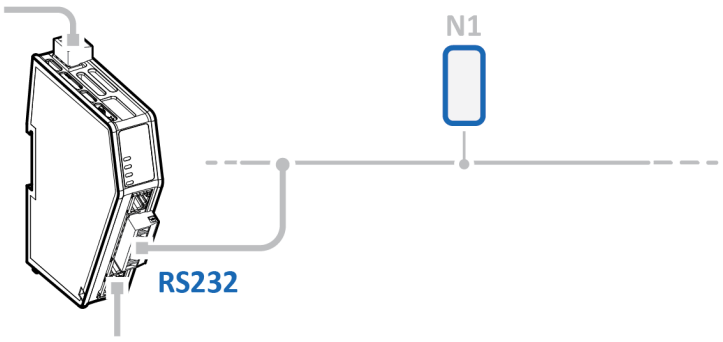
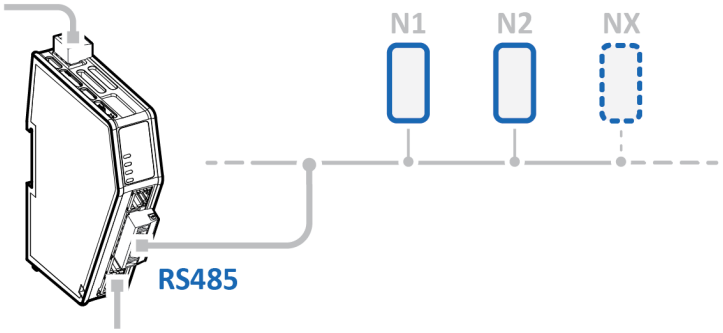


Figure 23. Communication, Basic settings

Physical standard

Specify the physical interface type for the device connected to the Communicator.

- Select a physical standard from the **Physical standard** drop-down menu.

Setting	Value	Description
Physical standard	RS-232, Default standard	Use RS-232 when one single node is connected to the subnetwork. 
	RS-485	Use RS-485 when multiple nodes are connected to the subnetwork. 

Baud rate

Specify the baud rate; the serial transfer speed, maximum bits per second.

Select a baud rate value from the **Baud rate** drop-down menu.

Setting	Value
Baud rate	1200 baud
	1800 baud
	2400 baud
	4800 baud
	9600 baud, Default value
	19200 baud
	35700 baud
	38400 baud
	57600 baud
	115200 baud
	128000 baud

Data bits
Data bits is the number of bits used in the data representation of characters in the telegrams.

The rate for Modbus RTU is 7 data bits or 8 data bits. The default setting is 8 data bits.

Parity
Specify if parity should be used to detect errors in the data.

Select parity value from the **Parity** drop-down menu.

Setting	Value	Description
Parity	None, Default value	No parity checking Parity bit is not transmitted
	Odd	Odd parity checking
	Even	Even parity checking

Stop bits
Specify the number of stop bits used to indicate the end of data transmission.

Select a stop bits value from the **Stop bits** drop-down menu.

Setting	Value
Stop bits	1 stop bit, Default value
	2 stop bit

Apply Configuration
To apply the settings, click **Apply** in the built-in web interface header and follow the instructions.

8.5.3. Communication Advanced Settings

Inter-Telegram Timeout Mode Settings

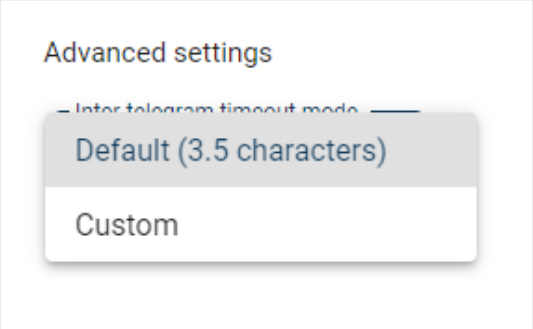


Figure 24. Advansed settings, Default (3.5 characters)

By default, Inter-telegram timeout mode Default (3.5 characters) is used.

This is according the Modbus RTU standard, which advocates the use of a silent period equivalent to 3.5 characters between each message. The silent period is used to find out where one message ends and the next begins.

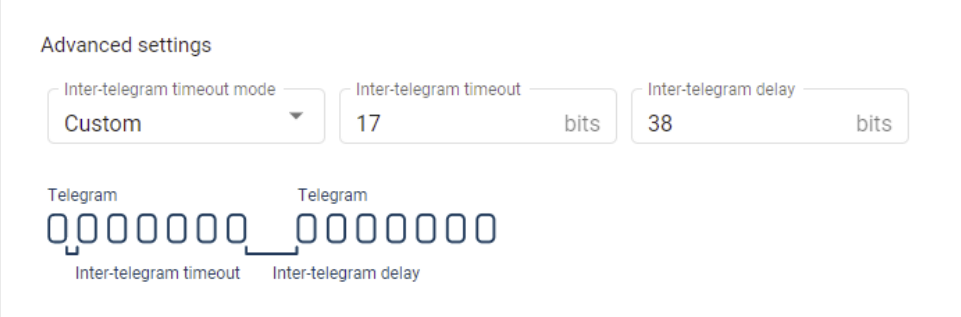


Figure 25. Custom settings, Inter-telegram timeout and Inter-telegram delay

You can use Custom settings to set the desired Inter-telegram timeout and Inter-telegram delay.

The following must be applied on all nodes:

- The time between two adjacent characters in the same telegram must be less than Intertelegram timeout.
- The time between two characters in two different telegrams the same or more than Intertelegram delay.

About Inter-Telegram Start and End Character

This topic describes scenarios for using Start character and End character.

Example 2. Both Start Character and End Character Disabled

Default setting, no **Start character** or **End character** is used.

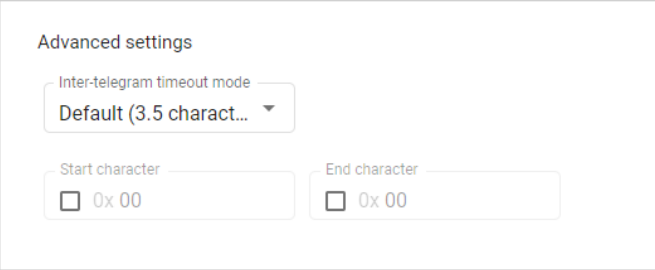


Figure 26. Start character and End character are Disabled

The size of the telegram is defined between two inter-telegram delays.

Standard **Inter-telegram delay**: 3.5 characters

Inter-telegram delay is set in the Advanced settings, see Inter-Telegram Timeout Mode Settings (page 45).

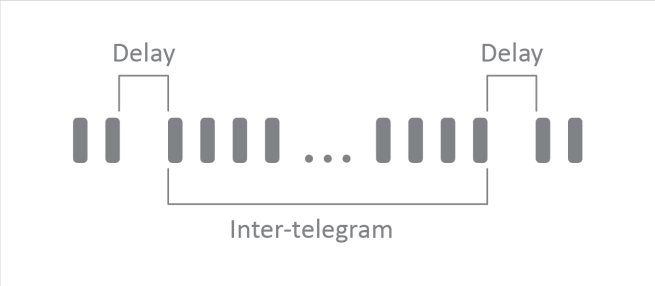


Figure 27. Data stream without Start and End characters

Example 3. Start Character Enabled and End Character Disabled

Start character is used.

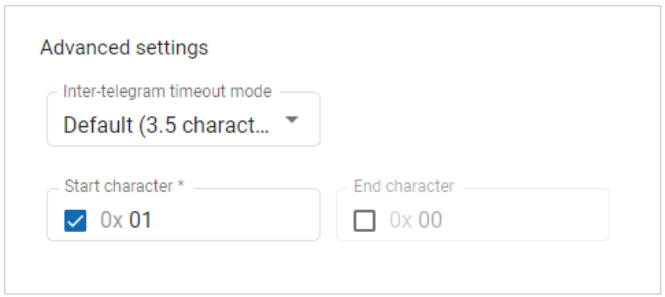


Figure 28. Example: Start character Enabled, Hex value 0x01

First, a start character is identified in the data stream, then the inter-telegram follows. The inter-telegram ends when an inter-telegram delay is identified in the data stream.

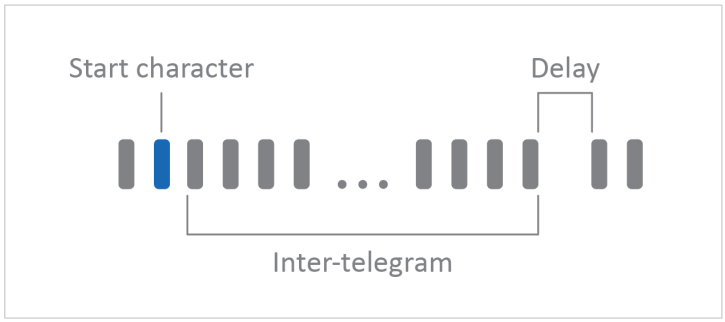


Figure 29. Data stream with Start character

Example 4. Start Character Disabled and End Character Enabled

End character is used.

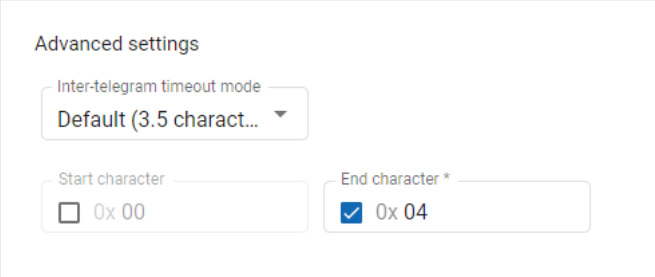


Figure 30. Example: End character Enabled, Hex value 0x04

First, an inter-telegram delay is identified in the data stream, then the inter-telegram follows. The inter-telegram ends when an end character is identified in the data stream.

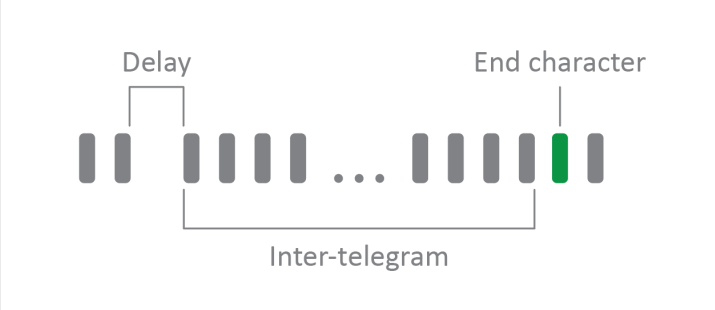


Figure 31. Data stream with End character

Example 5. Bothe Start Character and End Character Enabled

Both **Start character** and **End character** is used.

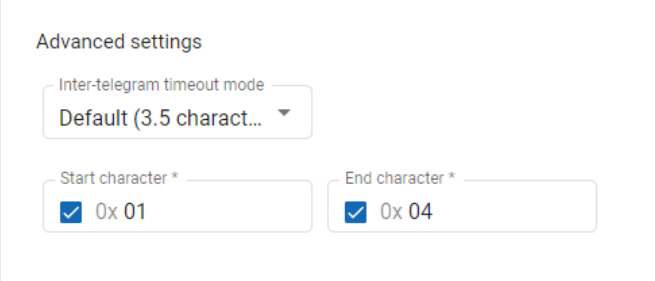


Figure 32. Example: Start character Enabled, Hex value 0x01 and End character Enabled, Hex value 0x04

First, a start character is identified in the data stream, then the inter-telegram follows. The inter-telegram ends when an end character is identified in the data stream.

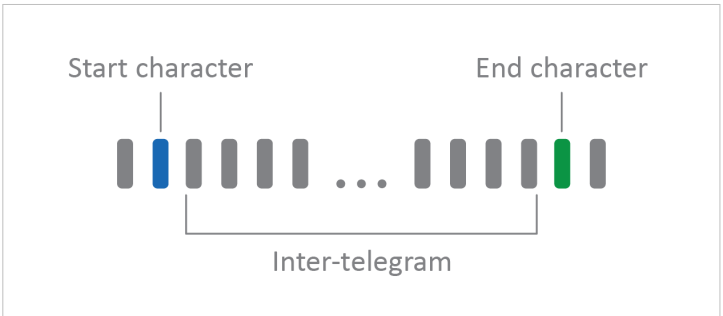


Figure 33. Data stream with Start character and End character

Inter-Telegram Start and End Character

Start character and **End character** is only valid for the **Custom Produce/Consume Serial protocol** mode; **Produce** or **Consume** transactions, see Communication Serial Protocol (page 42).

NOTE

Inter-telegram timeout mode Custom settings for **Inter-character timeout** and **Inter-telegram delay** still apply if **Start character** and/or **End character** is enabled.

NOTE

If a inter-telegram exceeds the maximum allowed transaction frame size of 1500 bytes, the telegram is invalid and discarded.

Start and End character Use Case Example

In serial channel data stream, there is a **Consume** transaction for which you want to obtain the content between two defined characters as an Inter-telegram.

Procedure

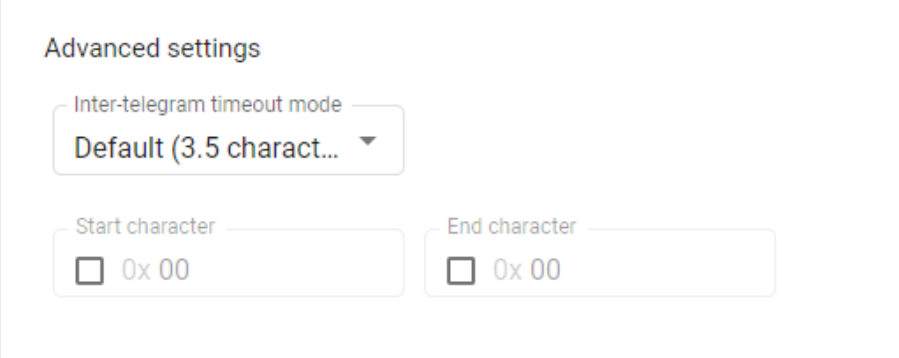


Figure 34. Start and End character settings

1. Navigate to the **Serial RS-232/485 Communication** page.
2. Select the **Custom Produce/Consume Serial protocol** mode.
3. To enable **Start character** and/or **End character**, select the checkbox(es).
4. Enter the decried character hexadecimal value(s) for the Inter-telegram.

Result

For **Produce** transactions, a start and/or end character is added to the inter-telegram when it is sent.

8.6. About Transaction Templates

This section applies when the Custom Request/Response or Custom Produce/Consume serial protocol is applied, refer to Communication Serial Protocol (page 42).

8.6.1. Transaction Template Example

Custom Request/Response

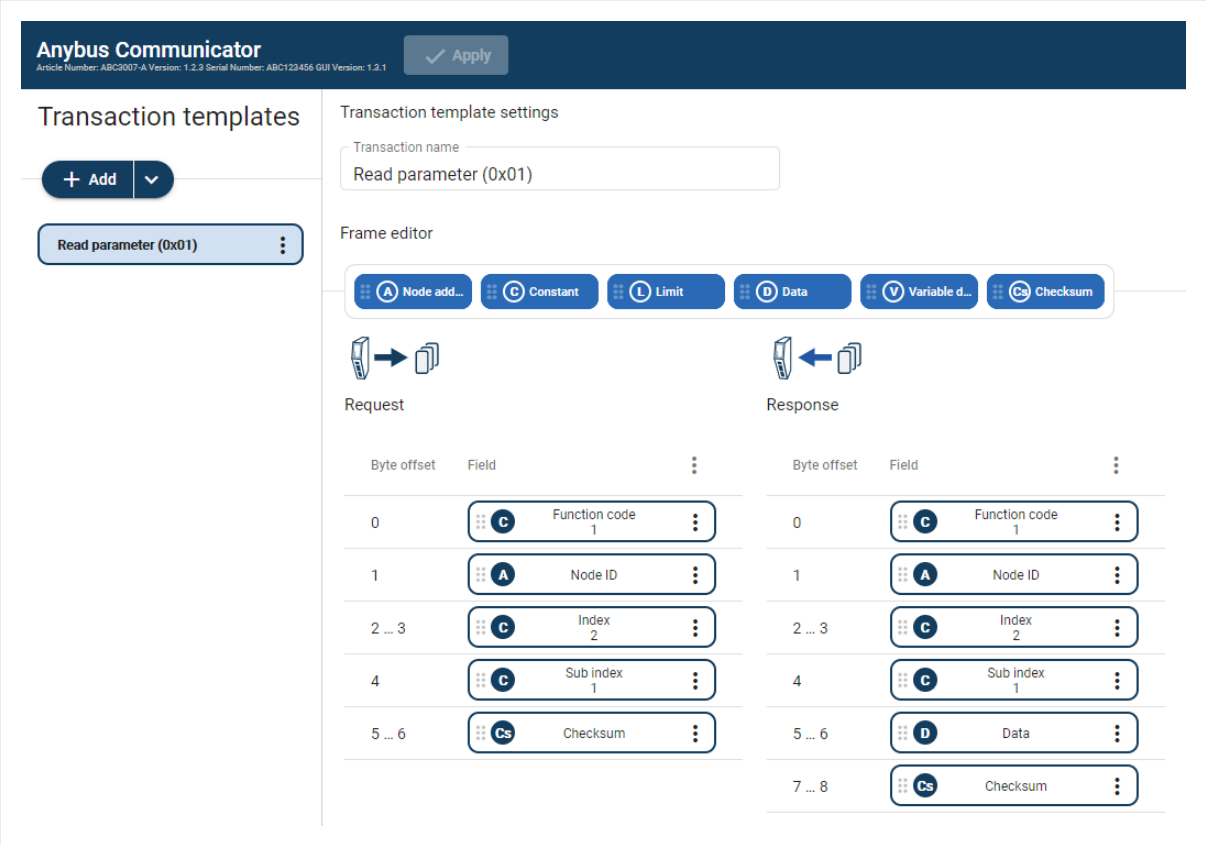


Figure 35. Request/Response transaction template example

The transaction named Read parameter (0x01) consists of a number of frame fields.

In the Request field there are three Constants, a Node address and a Checksum field.

In the Response field there are three Constants, a Node address, a Data field and a Checksum field.

Custom Produce/Consume

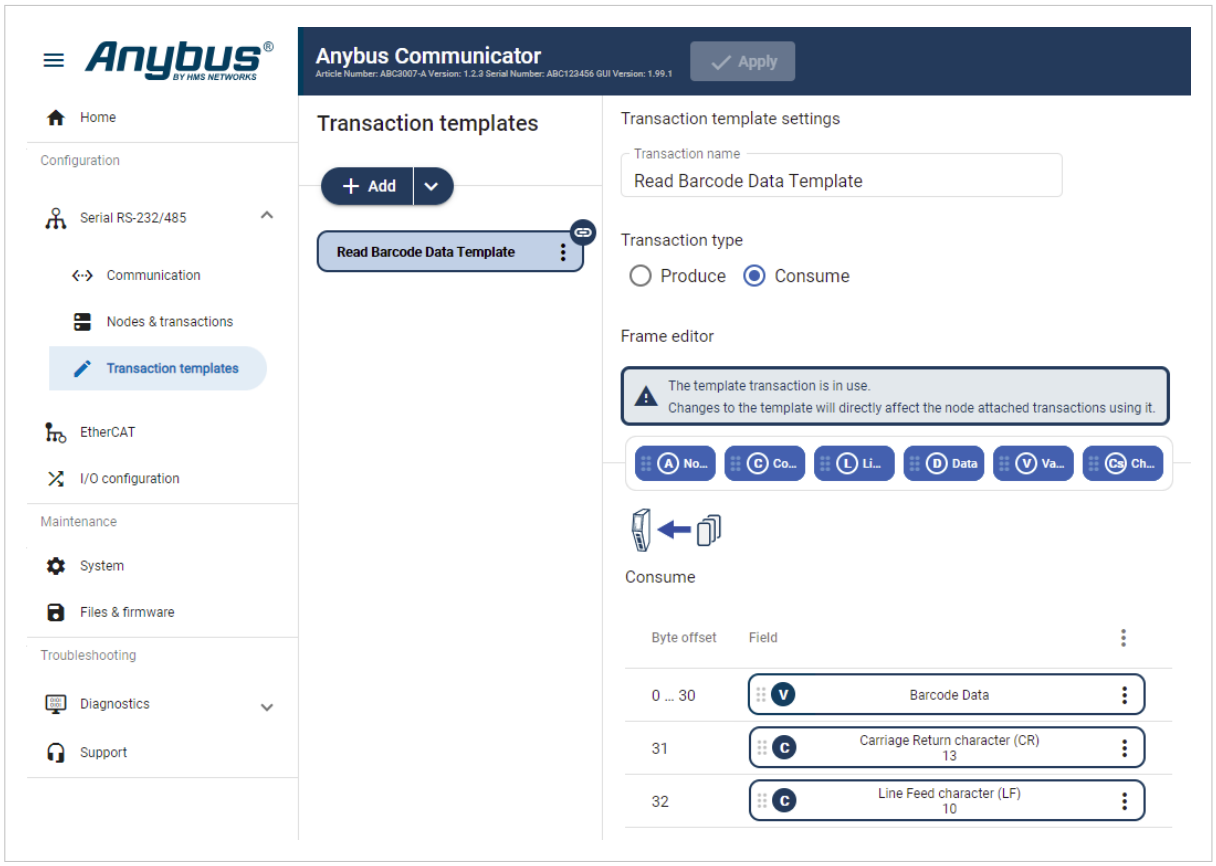


Figure 36. Produce transaction template example

The transaction named Read Barcode Data Template consists of a number of frame fields.

The Transaction type can be Produce or Consume. In this example the Transaction type Consume is selected.

In the frame field we have added one Variable data field and two Constant fields.

8.6.2. Transaction Template Types

There are two types of transaction templates, Empty template and Modbus template.

Empty Template

When using the **Empty template**, you start with an empty transaction and build a desired structure by adding and arranging frame fields.

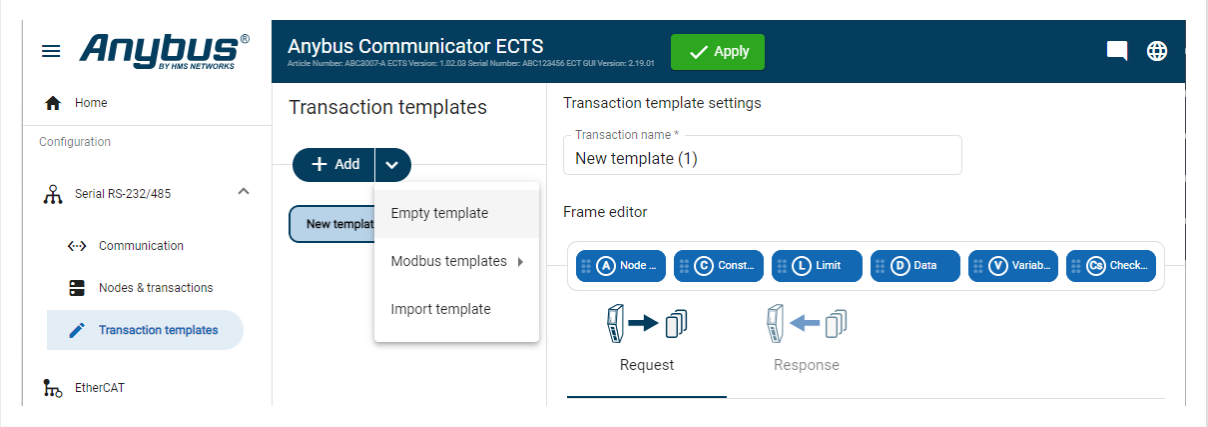


Figure 37. Transaction template, Empty template

For the produce/consume transactions you select; **Empty produce template** or **Empty consume template**.

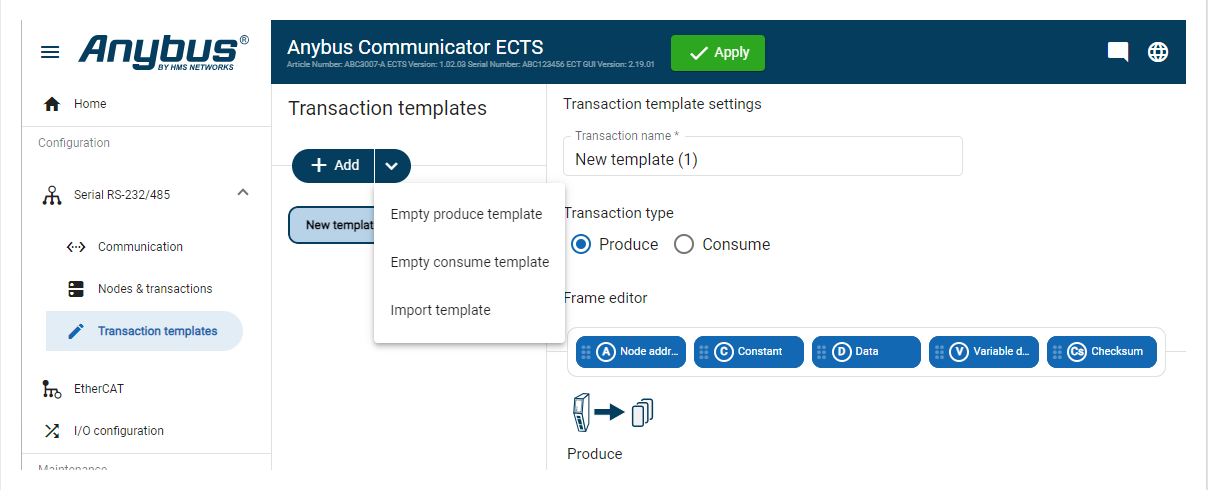


Figure 38. Transaction template, Empty Produce and Consume templates

Modbus Templates

Modbus templates are available for Custom Request/Response and Modbus RTU transactions.

When using the Modbus template, you first select the **Modbus template** from which you want to start.

You can then restructure the transaction by rearranging, adding or removing frame fields.

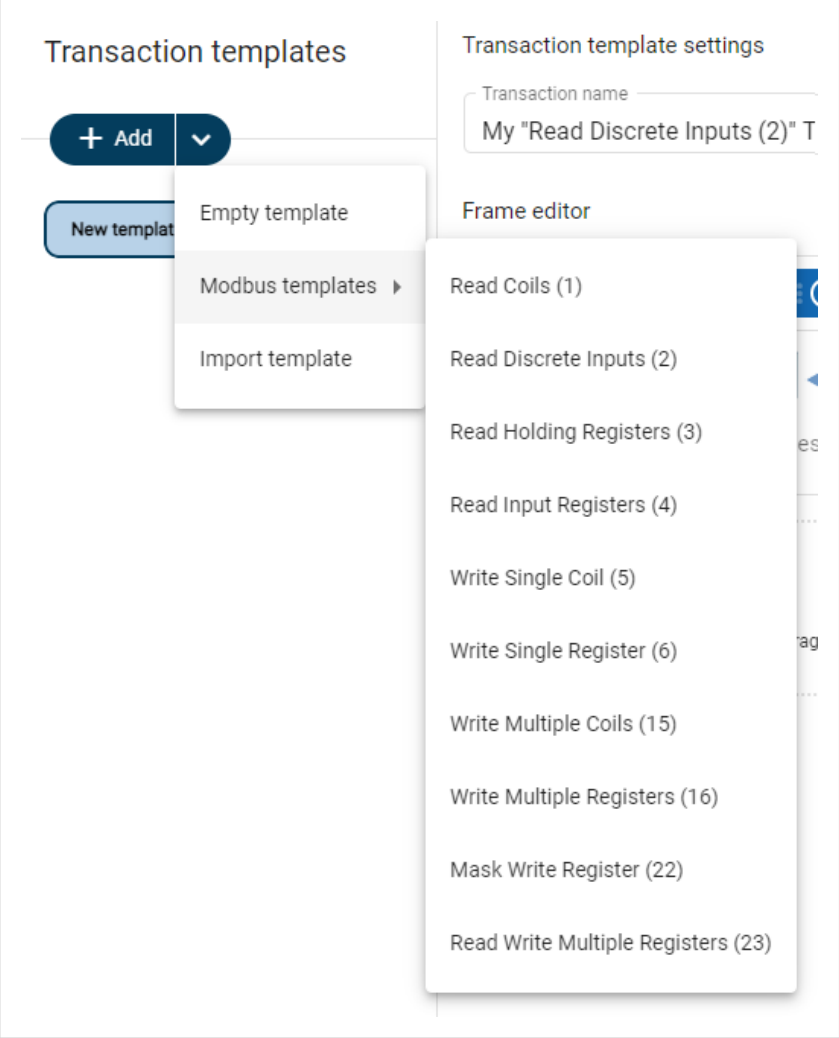
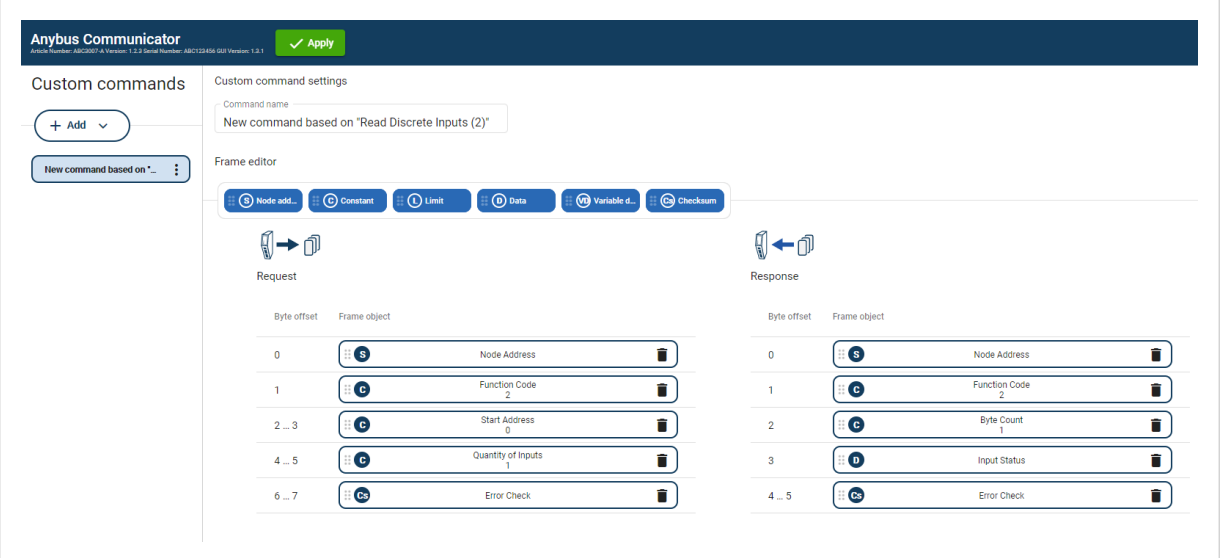


Figure 39. Transaction template, Modbus template

Example 6. New transaction template based on the Modbus template Read Discrete Inputs



8.6.3. Frame Field Types

Each transaction consists of frame fields which makes up the serial telegram frame.

Each frame field specifies how the Communicator shall interpret or generate a particular part of the telegram.

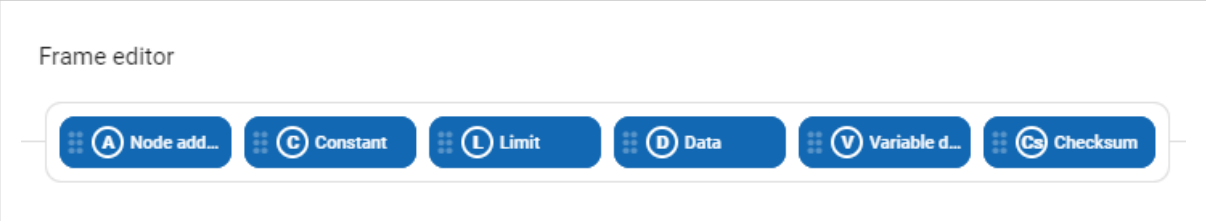


Figure 40. The following frame objects are available

Node address
Frame field representing the Node address of the Node.

A constant byte that holds a copy of the nodes address when the transaction is used by a node.

Constant
Constant frame fields are handled differently depending on the direction of the transaction.

- Produce/Request Transactions: The Communicator sends the value as it is without processing it.
- Consume/Response Transactions: The Communicator checks if the received byte/word/dword matches the specified value. If the message does not fit, it is discarded.

Limit

NOTE
Limit is not available for the Transaction Type Produce.

- Consume/Response Transactions: The Communicator checks if the received byte/word/dword fits inside the specified boundaries. If the message does not fit, it is discarded.

Data
Data frame fields are used to represent raw data as follows.

- Produce/Request Transactions: The specified data block is forwarded from the higher level network to the subnetwork.
- Consume/Response Transactions: The specified data block is forwarded from the sub-network to the high level network.

Variable data
Produce/Request Transactions:

- The specified data block will be forwarded from the higher level network to the sub-network.
- The control system must supply an End or Length character in order for the Communicator to know the size of the data block.
- The End- or Length-character itself may either be forwarded to the sub-network or discarded.

Consume/Response Transactions:

- The specified data block is forwarded from the sub-network to the higher level network.
- The End- or Length-character will be generated by the Communicator automatically (if applicable).

- The End- or Length-character itself may either be forwarded to the higher level network or discarded.

Checksum
Most serial protocols features some way of verifying that the data has not been corrupted during transfer.

The checksum frame field calculates and includes a checksum in a transaction.

8.7. Build Transaction Template

Before You Begin

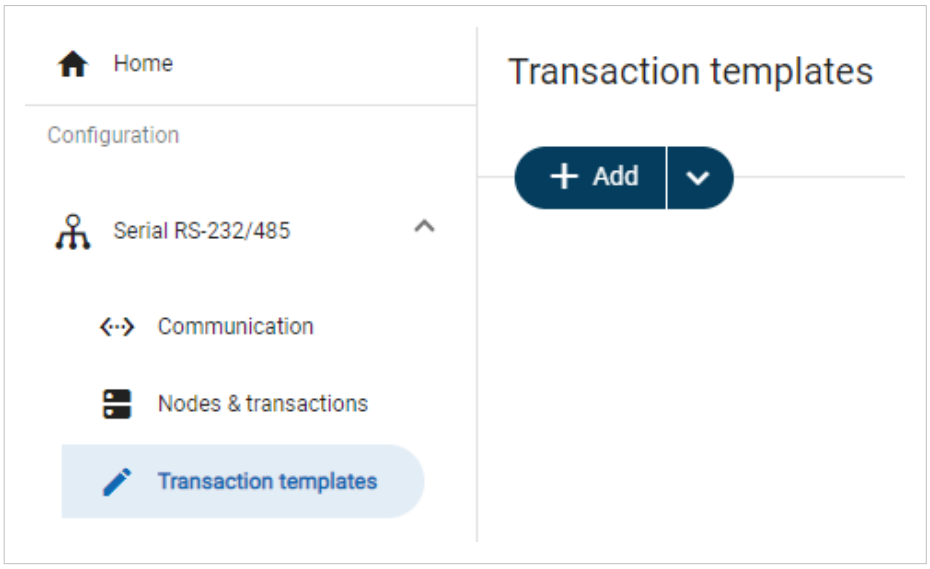
Ensure that you have applied the Custom Request/Response or Custom Produce/Consume serial protocol, refer to Communication Serial Protocol (page 42).

8.7.1. Add Transaction Template

Procedure

Add a transaction template:

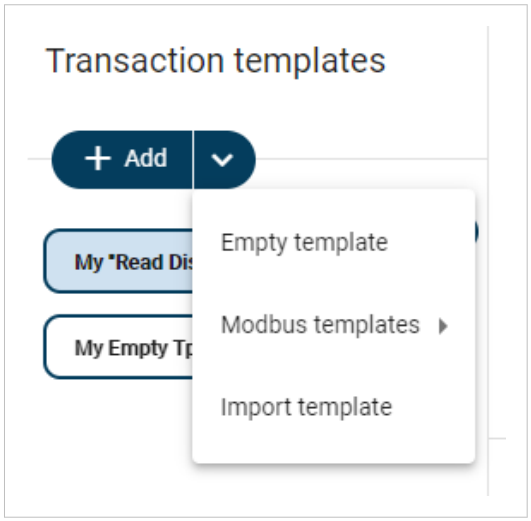
1. In the web-interface left sidebar menu, click **Transaction templates**.



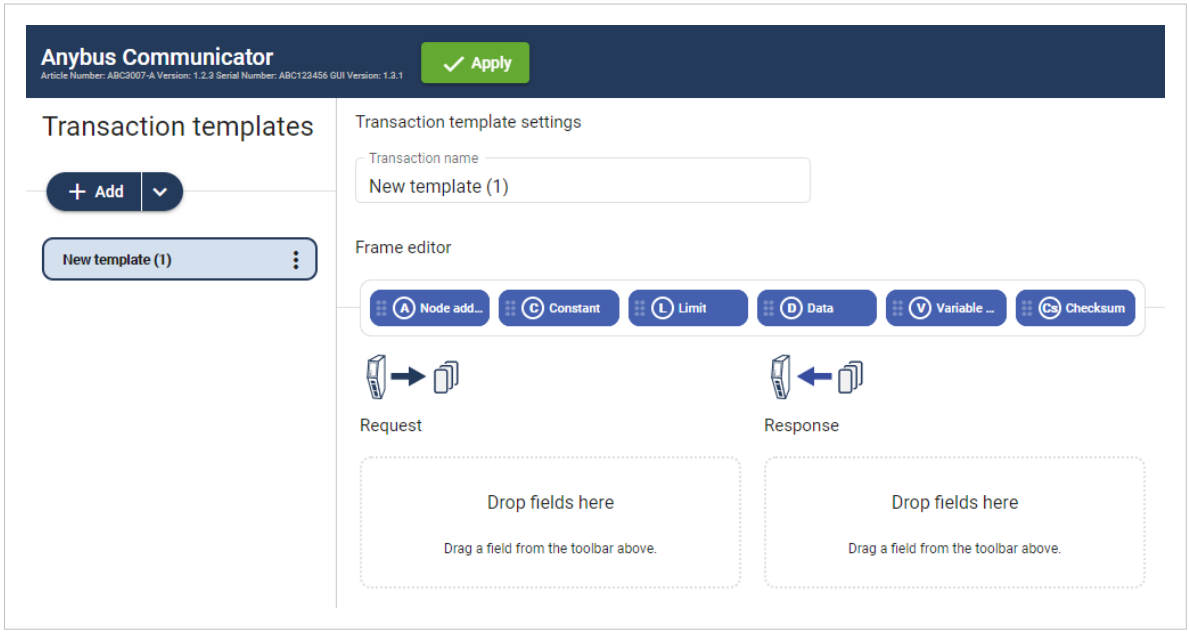
2. To select the template you want to use, click the **Add** drop-down button.

Options for the Custom Request/Response Protocol

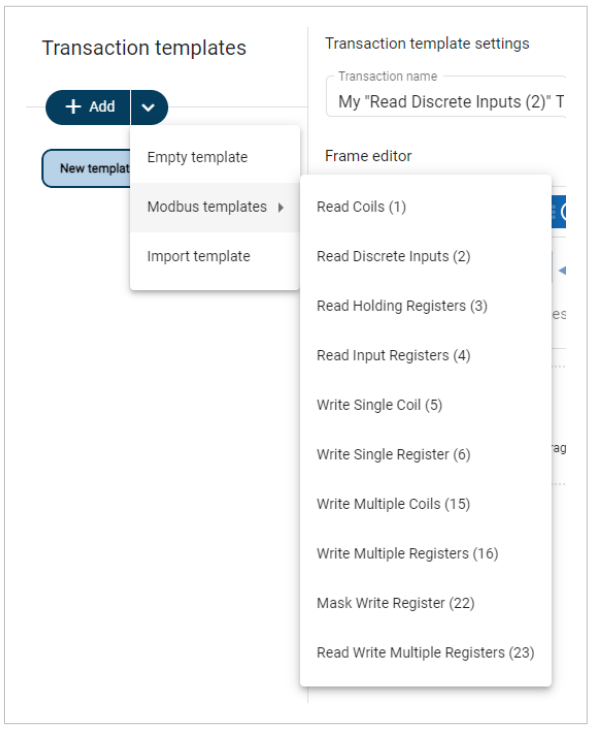
- To add a new empty template without any frame fields, select **Empty template**.



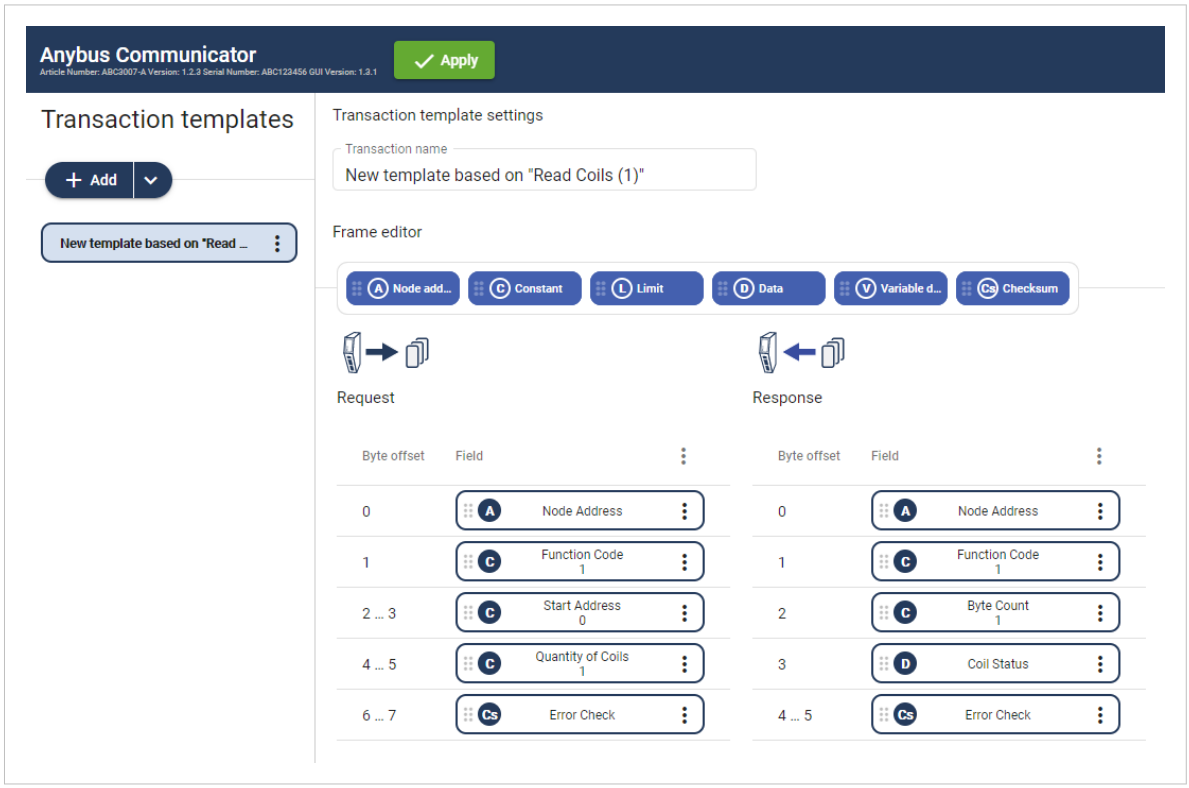
Example 7. A new empty request/response template is added to the transaction template list



- To add a new template based on a standard Modbus transaction, select **Modbus templates** and then the desired Modbus transaction.

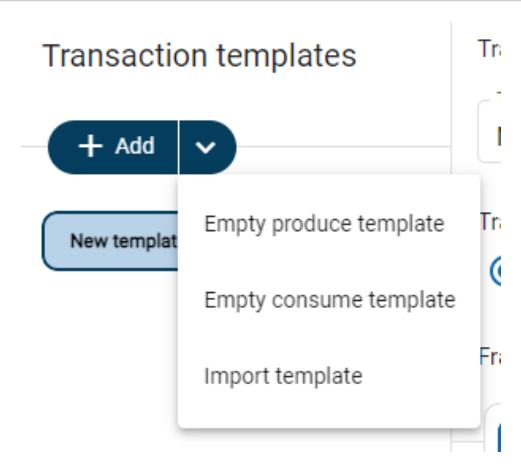


Example 8. A new request/response template based on “Read Coils (1)” is added to the transaction template list

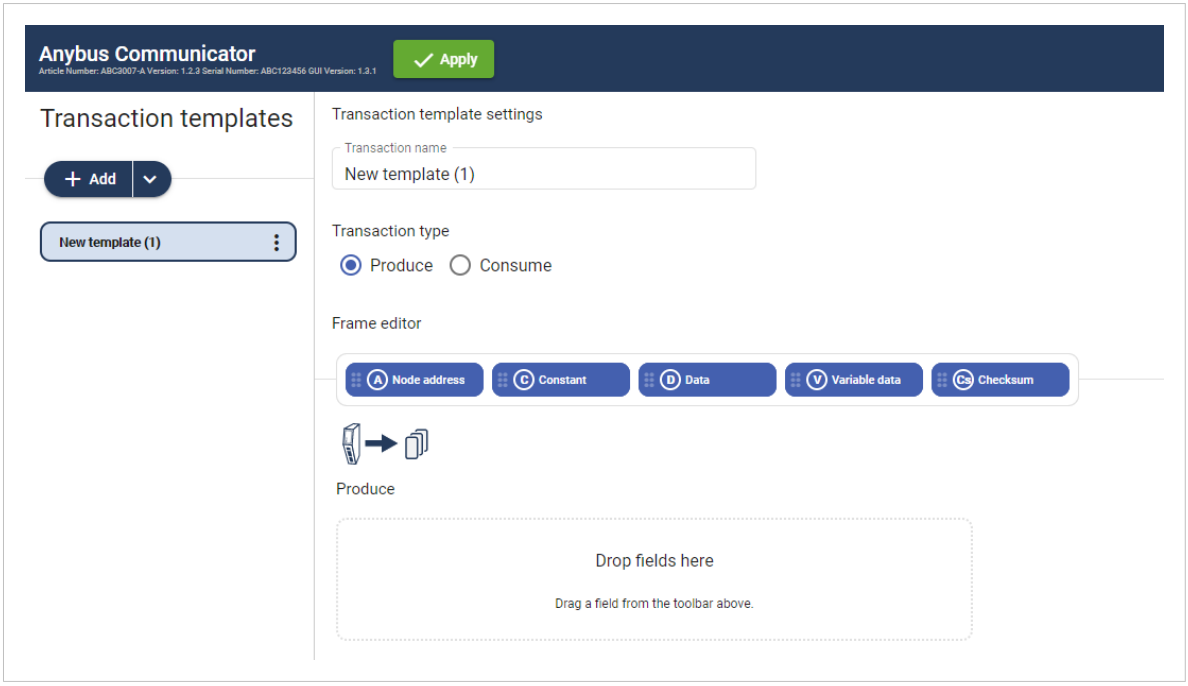


Option for the Custom Produce/Consume Protocol

- Select **Empty produce templates** or **Empty consume templates**.
  You can change the Transaction type after the transaction template is added.



Example 9. A new produce template is added to the transaction template list



3. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

To Do Next

- Add frame fields to the transaction template, refer to Add Frame Fields (page 61).

8.7.2. Add Frame Fields

Procedure

1. In the transaction template list, select a transaction template to add frame fields to.

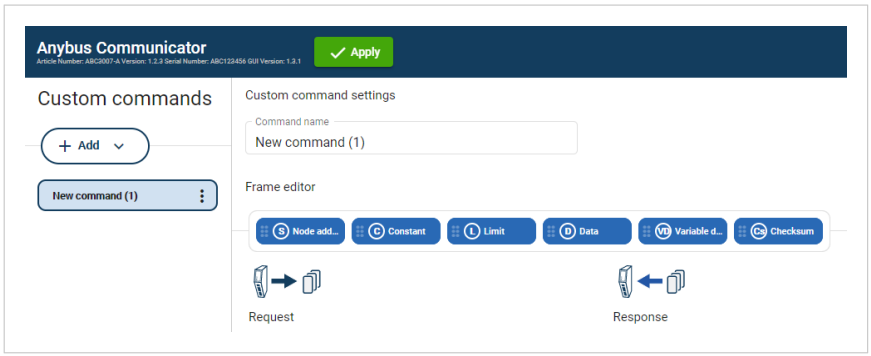


Figure 41. Select transaction template

2. Build the transactions.

- To add frame fields: In the Frame editor frame fields menu, drag and drop the desired frame fields into the drag and drop fields.

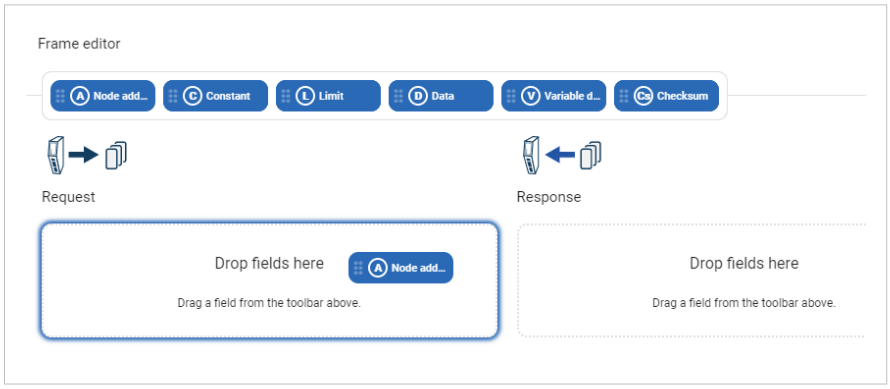


Figure 42. Add frame fields

- To duplicate a frame field: On the frame field that you want to duplicate, click the three dots icon and then click Duplicate.

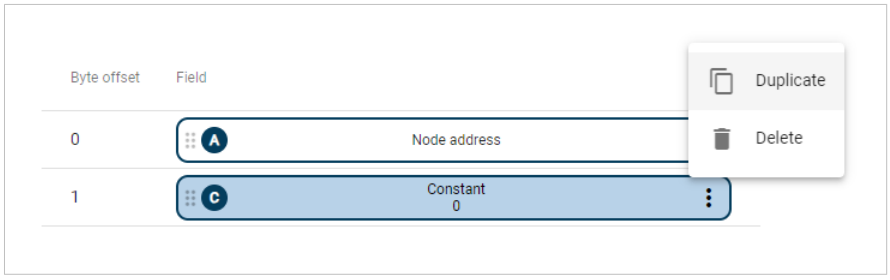


Figure 43. Duplicate frame fields

- To change the order of the frame fields: Drag and drop the frame fields in the list to change the order.

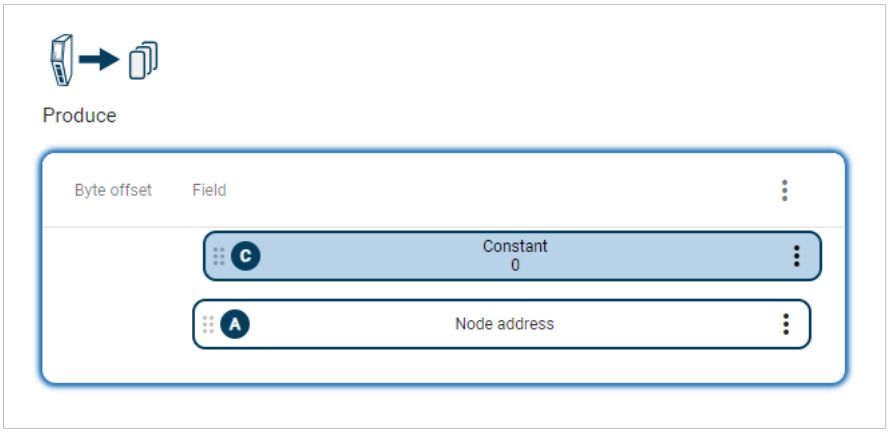


Figure 44. Change frame fields order

- To delete a frame field: On the frame field that you want to delete, click the **three dots icon**. Click **Delete** and then **Yes** to confirm.

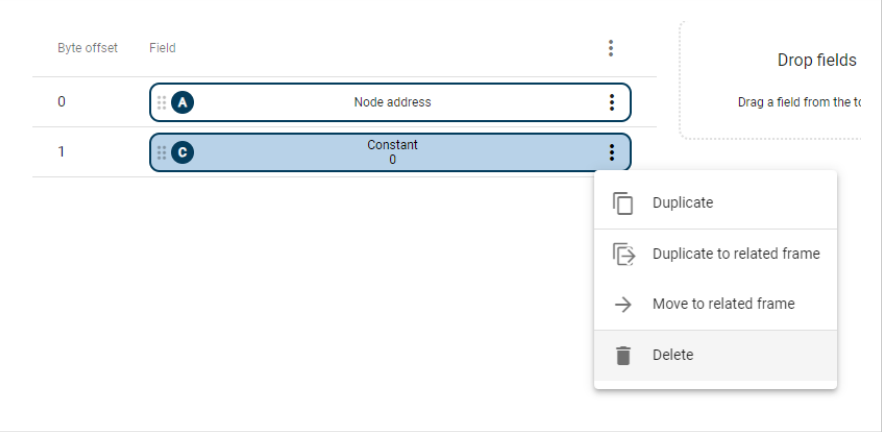


Figure 45. Delete frame field

3. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

To Do Next

- Configure the frame field settings, see Configure Frame Field Settings (page 63).

8.7.3. Configure Frame Field Settings

Procedure

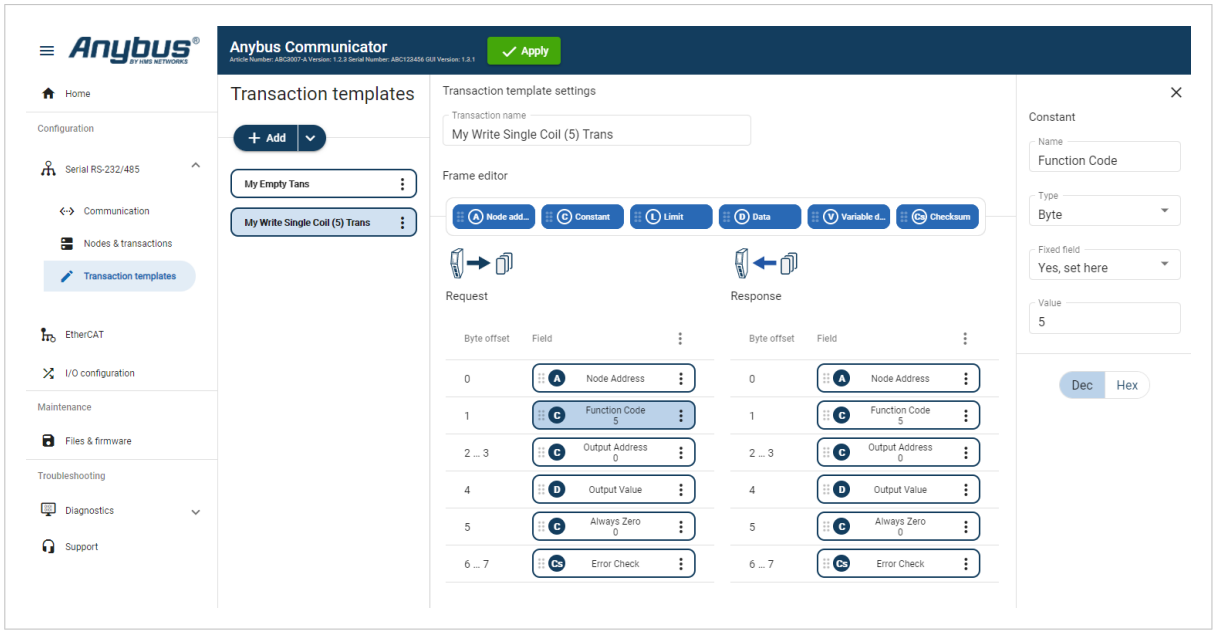


Figure 46. Frame Fields Settings

1. In the Transaction templates list, select a transaction template to configure.
2. In the Transaction template settings select a Field to configure.
   The Field sidebar opens, on the right side of the screen.
3. Configure the **Field** settings.

Node address

- Frame field representing the Node address of the node. A constant byte that holds a copy of the nodes address when the transaction is used by a node.
- When the transaction template is used by a node, the Node address field will automatically be replaced with the actual node address of the node.

Constant

- **Name**: You can name the Frame Field to make it easier to identify.
- **Type**: Specify the number of bytes in the frame.
  Select Byte (1 byte) (Default), Word (2 bytes), Double word (4 bytes), Array of bytes or String.
- **Endianess**: Select Big-endian (Default) or Little-endian.
- **Fixed field***: Select Yes, set here (Default) or No, set when used.
- **Length**: Valid for Array of bytes. Enter a byte offset value between 0 and 32 byte.
  Default value is 1 byte.
  Enter a Value for each Byte (0–31).
- **Value**: The value of the Constant in the frame.
  Enter a value between 0 (Default) and 255.
- **Min value**: Specify the minimum value that can be set when the template is used.
- **Max value**: Specify the maximum value that can be set when the template is used.
- **Default value**: Default value set when the template is used.

Limit

> **NOTE**
> Limit can only be added as a Response frame field.

- **Name**: You can name the Frame Field to make it easier to identify.
- **Type**: Specify the number of bytes in the frame. Select Byte (1 byte) (Default), Word (2 bytes), Double word (4 bytes).
- **Endianess**: Select Big-endian (Default) or Little-endian.
- **Min value**: The lowest value of the limit range.
- **Max value**: The highest value of the limit range.
- **Base number system**: Select Decimal Dec (Default) or Hexadecimal Hex.

Data

- **Name**: You can name the Frame Field to make it easier to identify.
- **Fixed field***: Select Yes, set here (Default) or No, set when used.
- **Length**: Enter a value between 1 (Default) and 512 bytes.
- **Min length**: Specify the minimum length that can be set when the template is used.
- **Max length**: Specify the maximum length that can be set when the template is used.
- **Default length**: Specify the default length that can be set when the template is used.

Variable data

- **Name**: You can name the Frame Field to make it easier to identify.
- **Fixed field***: Select Yes, set here (Default) or No, set when used.
- **Minimum payload length**: Specify the minimum payload length that can be set when the template is used.
- **Maximum payload length**: Specify the maximum payload length that can be set when the template is used.
- **Default max payload length**: Specify the default payload length that can be set when the template is used.
- **Data delimiter**: Specify how to detect/define the length of the variable data of the high level network. Select Byte counter, End pattern or None (Default).
  For information about End- and Length character, see Data Delimiter and Subnet Delimiter Options (page 66).
- **Subnet delimiter**: Specify how to detect/define the length of the variable data of the serial subnetwork. Select Byte counter, End pattern or None (Default).
  For information about End- and Length character, see Data Delimiter and Subnet Delimiter Options (page 66).
- **End pattern**: When a **Data delimiter** or **Subnet delimiter** is set to End pattern, specify the value defining the end of the payload.
- **Fill padding**: Fill up unused data mapped to the high level network or the general area with a field padding value.
  To deactivate/activate Fill padding, click the slide toggle. When Fill padding is activated, enter a Fill padding value between 0 and 255.
- **Base number system**: Select Decimal Dec (Default) or Hexadecimal Hex.

Checksum

- **Name**: You can name the Frame Field to make it easier to identify.
- **Checksum type**: Specify the algorithm used to calculate the checksum. Select CRC (CRC-16-IBM) (Default), LRC (ISO 1155:1978), XOR or ADD.
- **Start offset**: Specify the offset from where to start the checksum calculation. Enter a value between 0 (Default) and 511.

- **Error check type**: Specify how the checksum is converted. Select None (Default), One's complement or Select None, Two's complement.
- **Destination Representation**:
  Each byte represents two ASCII characters.
  Allowed characters: Digits 0-9 and Letters a-f
  The first two characters represent the 4 most significant bits of the byte. The second character represents the 4 least significant bits.
  Specify how the destination checksum is represented:
  - Binary (Default): Data is transmitted and received as-is, no pre- or post-processing is performed.
  - ASCII (Lower case): Received telegrams are case insensitive and sent telegrams are lowercase.
  - ASCII (Upper case): Received telegrams are case insensitive and sent telegrams are uppercase.

About Fixed field*

- **Yes, set here**: The Value set here is fixed and cannot be changed when the transaction is used on a node. The value must be updated in the transaction template.
- **No, set when used**: The Default value set here can be edited when the transaction is used on a node. The allowed range is the min/max values.

Total size including delimiters:

- High Level Network: 1 byte(s)
- Subnetwork: 1 byte(s).

4. Repeat step 1 to 3 until you have configured all the desired frame objects.

Apply Configuration
To apply the settings, click **Apply** in the built-in web interface header and follow the instructions.

8.7.4. Data Delimiter and Subnet Delimiter Options

In a variable data object, the length of the data field may vary depending on the type of data being read in a specific case.

In order to present the variable data correctly on the corresponding network, the length of the data field must be identified.

In a Variable data object, there are three ways to identify the data length; by length character, end character or length of message.

Data delimiter - Data is forwarded From the Communicator to the PLC

The Communicator can be configured to forward data as process data.

Different Data delimiter options can be used for data sent from the subnetwork to the Communicator and for data forwarded from the Communicator to the high level network, to fit the requirements in the PLC.

In most cases, when a stream of data is sent from the Communicator to the PLC the Byte counter (length character) or End pattern (end character) format is used.

Subnet delimiter - Incoming Data From a Serial Node to the Communicator

The Communicator can be configured to expect data from one of the three Subnet delimiter options; Byte counter, End pattern or None.

If the incoming data match the Subnet delimiter format the data is captured and the data section is forwarded to the high level network.

If the incoming data do not match the Subnet delimiter format, the data is ignored and will be matched with the next consume transaction.

Transaction Template Variable Data Settings

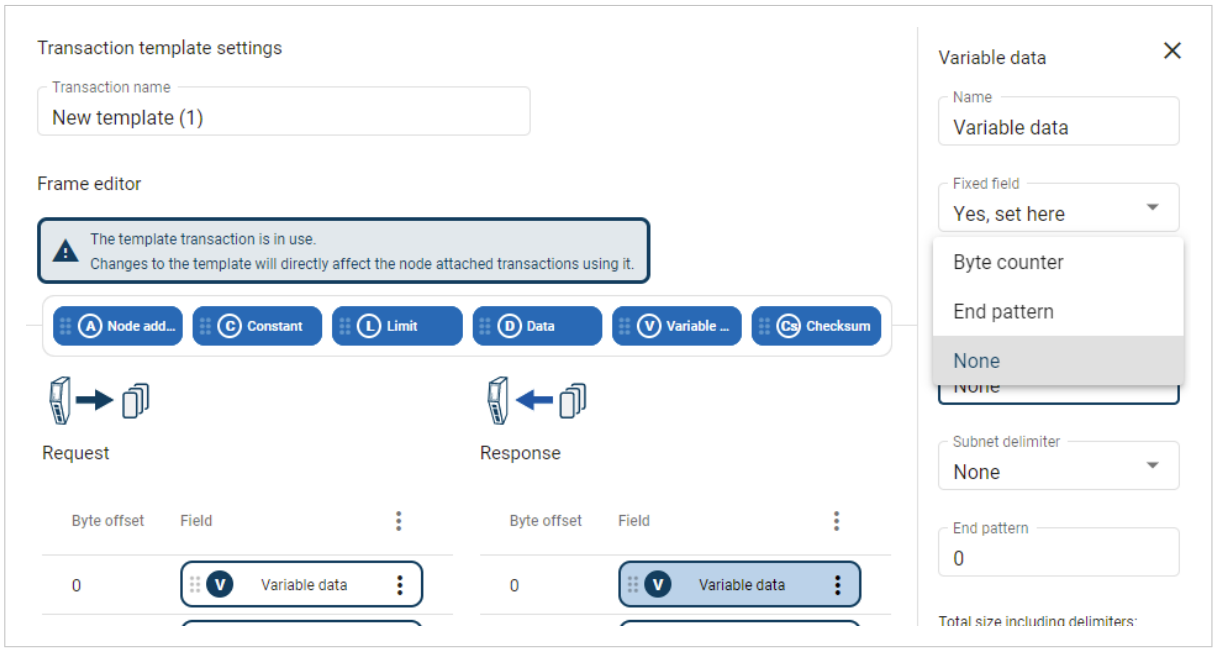
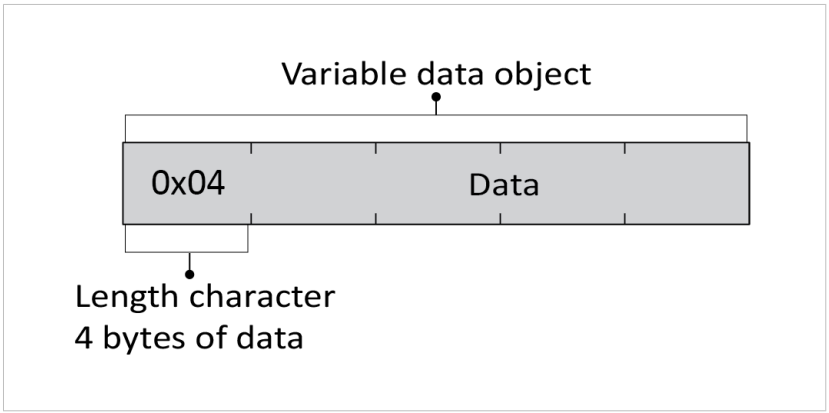


Figure 47. Transaction template variable data settings

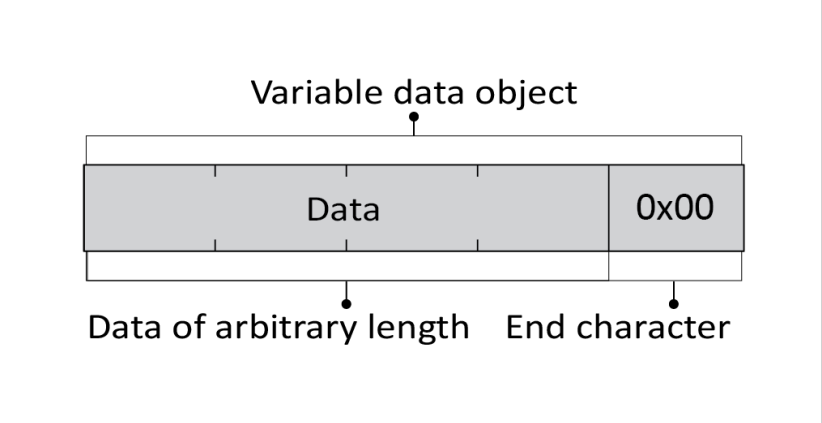
1. Select a desired **Variable data** object.
2. In the **Data delimiter** and/or **Subnet delimiter** drop down menu, select one of the following options.

Data delimiter and/or Subnet delimiter options

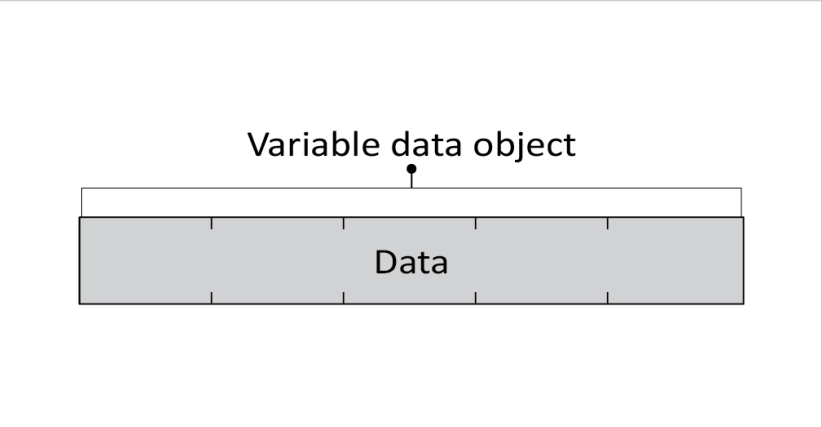
- Byte counter
  The data packet consists of a length character, indicating the length of the data section, followed by the variable data object itself. In order to copy the exact data size from the transaction message, the length of the variable data object is first identified.



- End pattern
  The package consists of a data section followed by an end character to indicate where the data section ends. End pattern is used to define whether the delimiter is an end character or an end pattern, which depends on whether the message is forwarded from the subnetwork or sent as process data.



- None (Default)
  The package contains only the data section. By measuring the total length of the message, the length of the data section can be calculated.



3. When a delimiter is set to End pattern: In the **End pattern** field, enter the value that will define the end of the payload.
   End pattern is used to define whether the delimiter is an end character or an end pattern, which depends on whether the message is forwarded from the subnetwork or sent as process data.

8.7.5. Store Transaction Templates

The transaction templates are stored on the **Transaction templates** page.

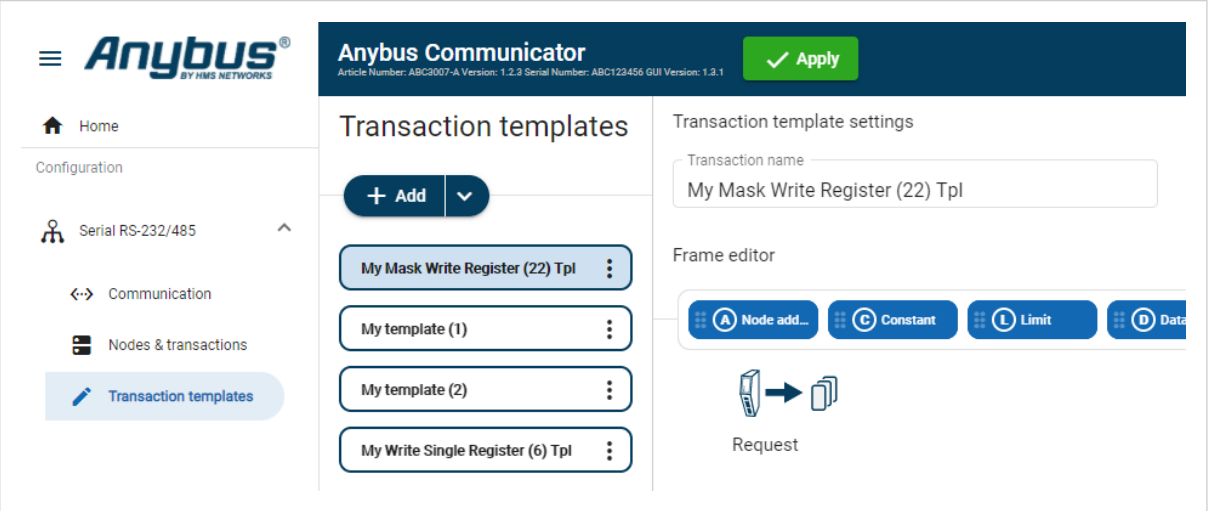


Figure 48. **Transaction templates** page

The transaction templates are available for use on the **Nodes & transaction** page, when you add transactions to a node.

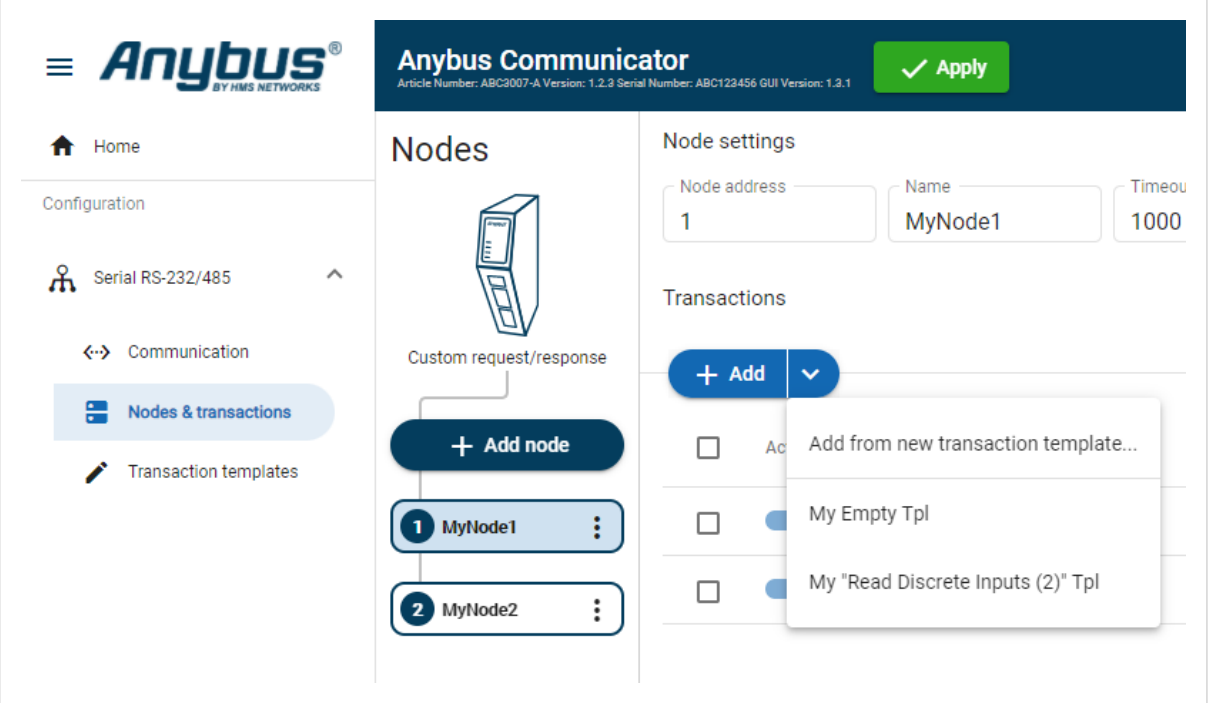


Figure 49. **Nodes & transaction** page

For information on how to add the transaction templates to the nodes, refer to Transaction Settings (page 83).

8.8. Nodes and Transactions

A node represents a single device on the serial subnetwork.

Add nodes and set up the communication between the nodes and the client.

Before You Begin

Obtain user documentation, from the manufacturers of the devices to communicate with, describing available registers and how to address them.

8.8.1. Node and Broadcast Node

You can add two types of nodes, Node and Broadcast Node.

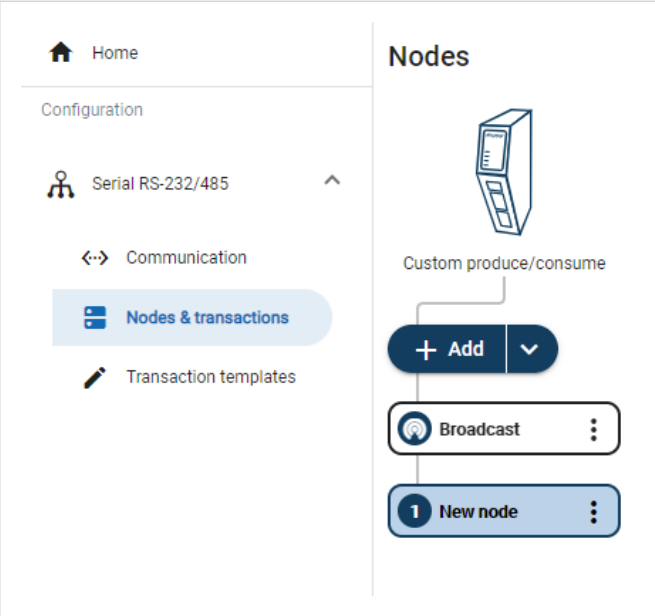


Figure 50. Add Node or Broadcast Node

Broadcast node

- You can add one single Broadcast node.
- The Broadcast node can only hold produce transactions.

Node

- You can add up to 31 Nodes.
- The type of transactions a node can hold depends on the serial protocol used, refer to About Transaction Templates (page 51).

8.8.2. Add Node

NOTE
You can add one single Broadcast node.

The maximum number of Nodes that can be added is 31.

1. In the web-interface left sidebar menu, click **Nodes & transactions**.
2. In the **Add** split button drop-down menu, select **Add broadcast node** or **Add node**.
   To **Import node**, see Import Node Settings From Other Communicator Unit (page 72).

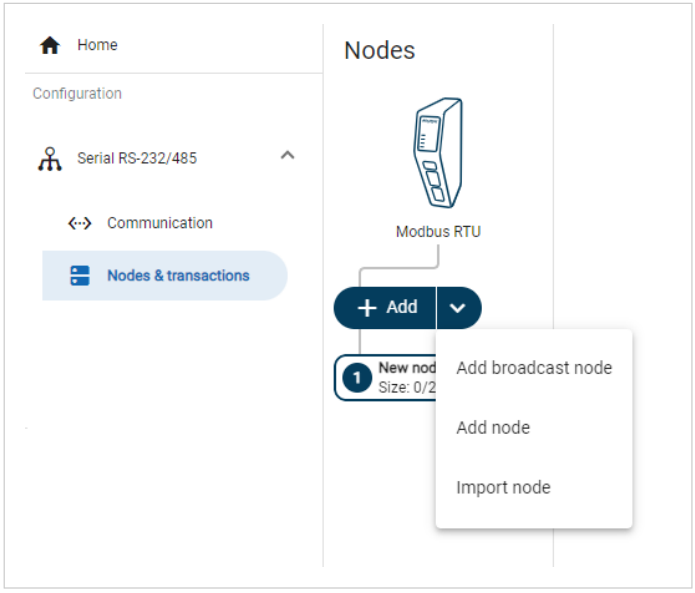


Figure 51. Add node

A new node/brodcats node is added to the nodes list.

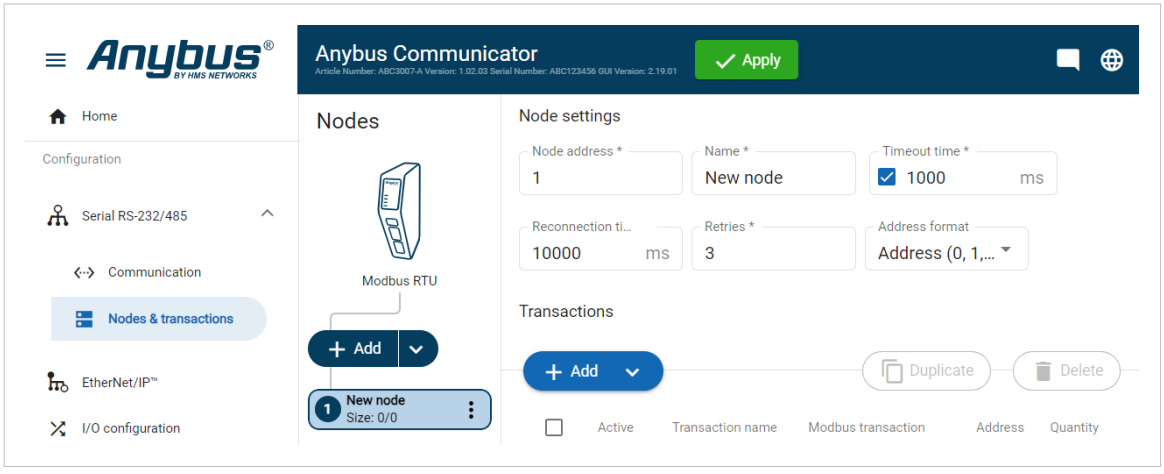


Figure 52. A new node is added to the node list

To Do Next
Configure the Node Settings, Node Settings (page 79).

8.8.3. Import Node Settings From Other Communicator Unit

When you have configured a node and want to use the same node for additional Communicator units, do the following.

Procedure

To Export the Node

In the built-in web-interface of the Communicator with the node you want to export:

1. In the web-interface left sidebar menu, click **Nodes & transactions**.
2. On the node to be exported, click the three dots icon and then click **Export node**.
   A *.node* file including the node settings and the related transaction templates is downloaded to your computer.

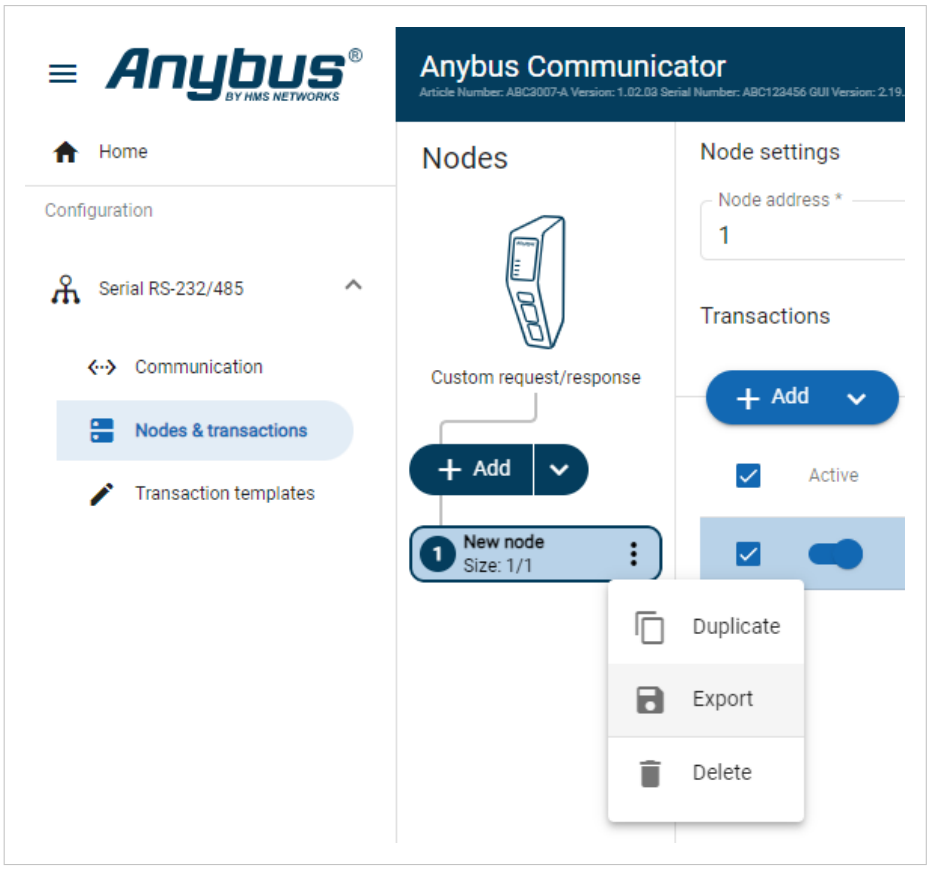


Figure 53. Export node

To Import the Node

In the built-in web-interface of the Communicator to which the node is to be imported:

3. In the web-interface left sidebar menu, click **Nodes & transactions**.

4. In the **Add** split button drop-down menu, select **Import node**.

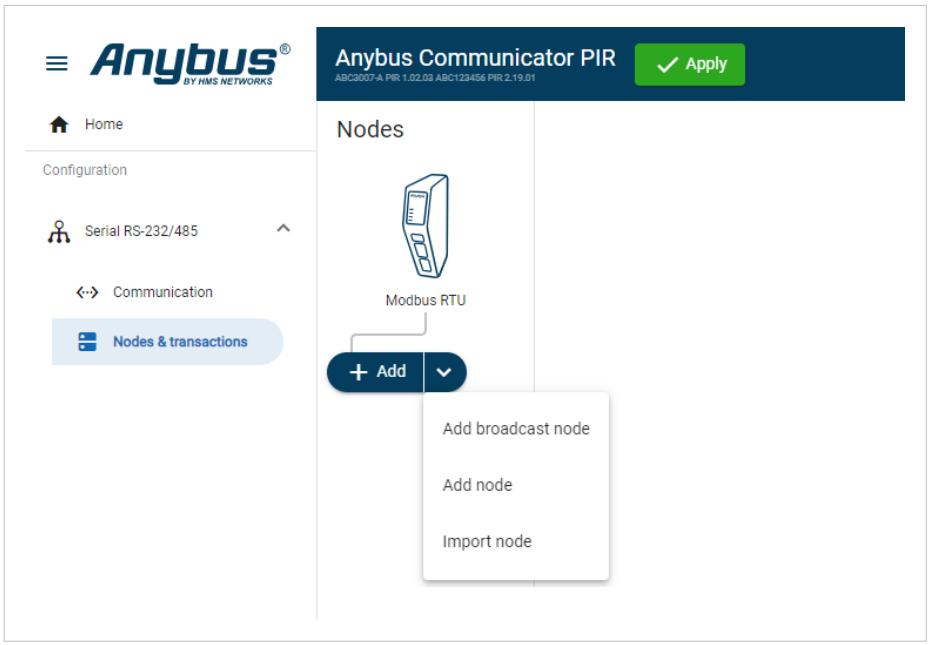


Figure 54. Import node

5. In the **Open** dialog box, browse to and select the *.node* file and click **Open**.
6. To import the *.node* file, click **Import**.

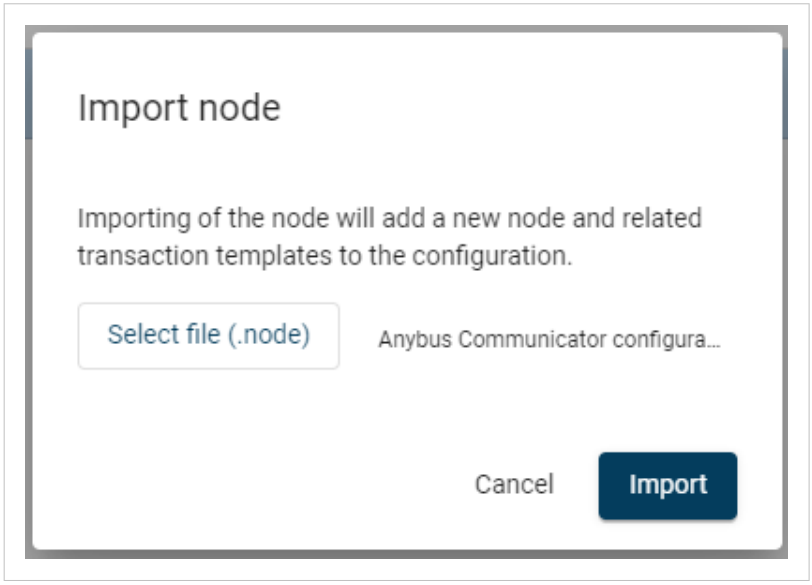


Figure 55. Example, selected .node file

Result

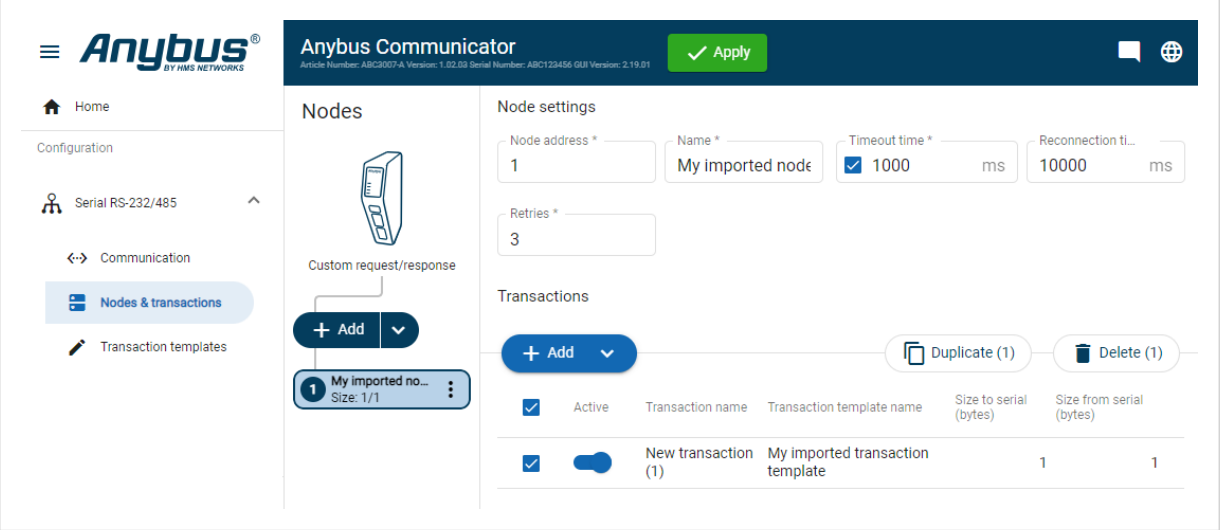


Figure 56. Example, imported node

The imported node is added to the **Nodes** list.

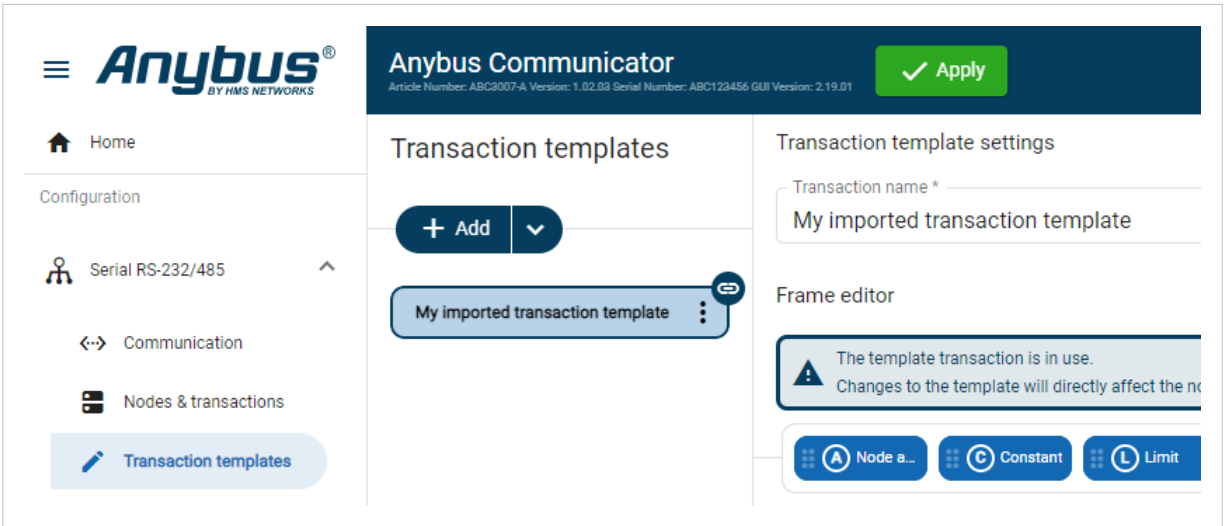


Figure 57. Example, imported transaction template

Any included transaction templates are added to the **Transaction templates** list on the **Transaction templates** page.

To Do Next

IMPORTANT

To identify the nodes and avoid node address conflicts on the subnetwork, each node (device) must have a unique node address number.

In the **Node settings** for the imported node, ensure that the **Node address** number is unique.

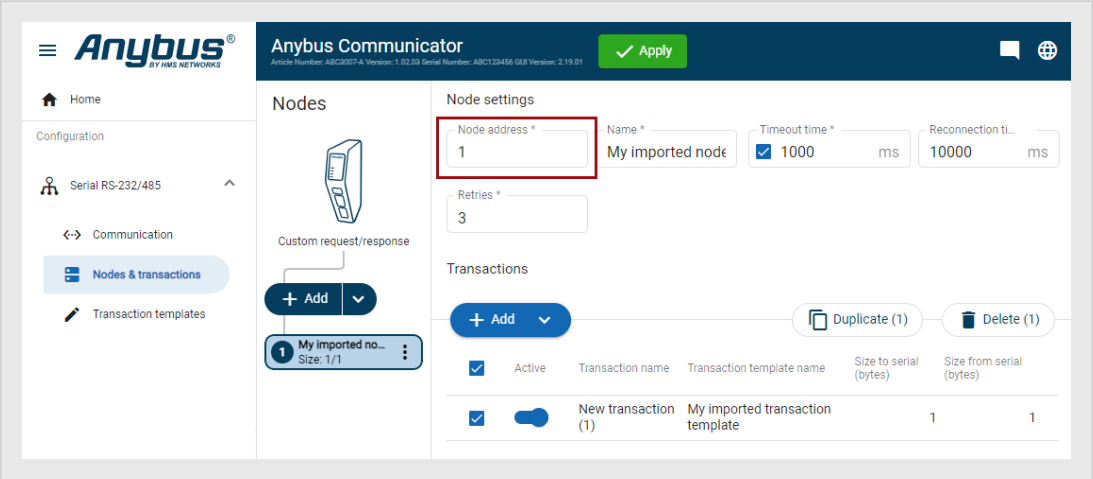


Figure 58. Node settings, Node address number

To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

8.8.4. Import Transaction Template From Other Communicator Unit

When you have configured a node and want to use the same node for additional Communicator units, do the following.

Procedure

To Export the Transaction Template

In the built-in web-interface of the Communicator with the transaction template you want to export:

1. In the web-interface left sidebar menu, click **Transaction templates**.
2. On the transaction template to be exported, click the three dots icon and then click **Export node**.
   A `.trans` file including the transaction template settings and frame fields is downloaded to your computer.

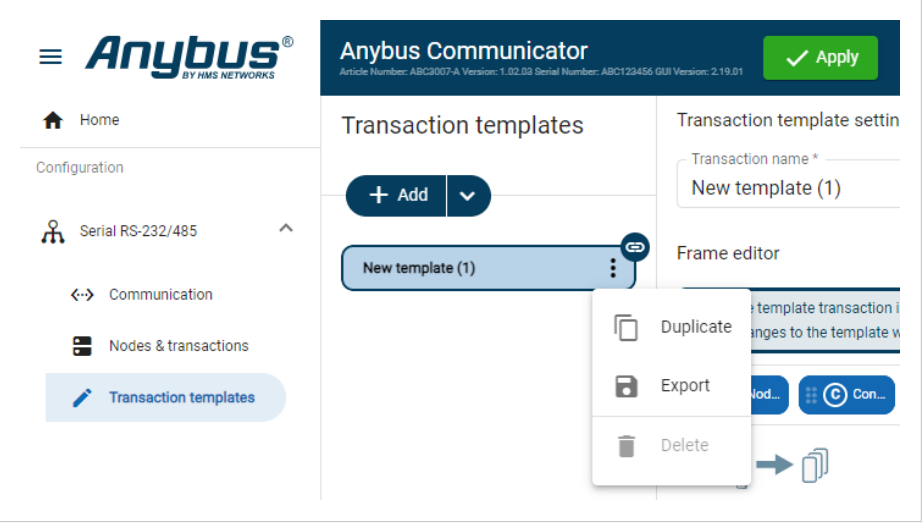


Figure 59. Export template

To Import the Transaction Template

In the built-in web-interface of the Communicator to which the transaction template is to be imported:

3. In the web-interface left sidebar menu, click **Transaction templates**.

4. In the **Add** split button drop-down menu, select **Import template**.

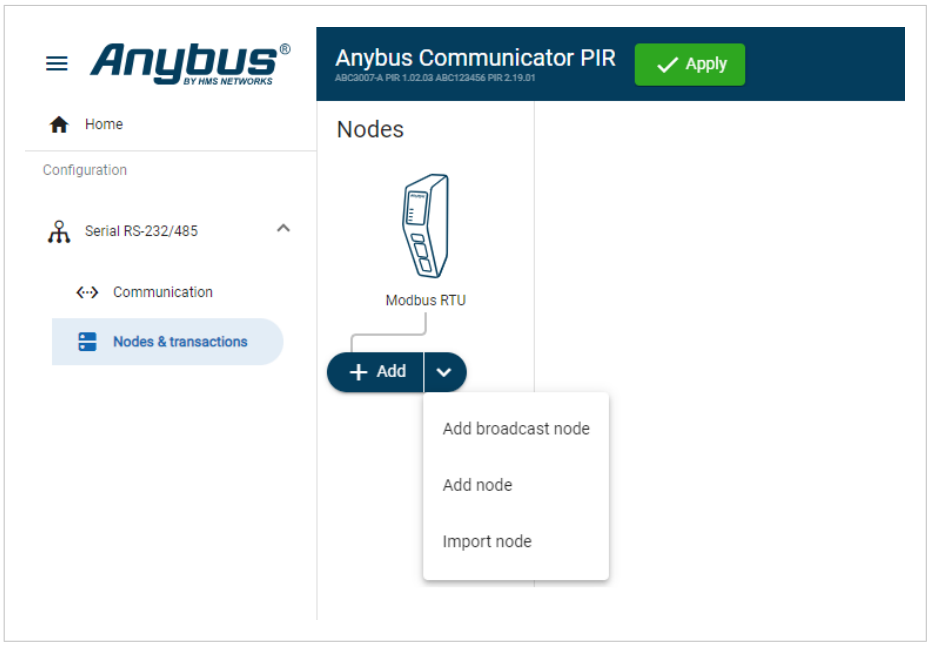


Figure 60. Import template

5. In the **Open** dialog box, browse to and select the `.trans` file and click **Open**.
6. To import the `.trans` file, click **Import**.

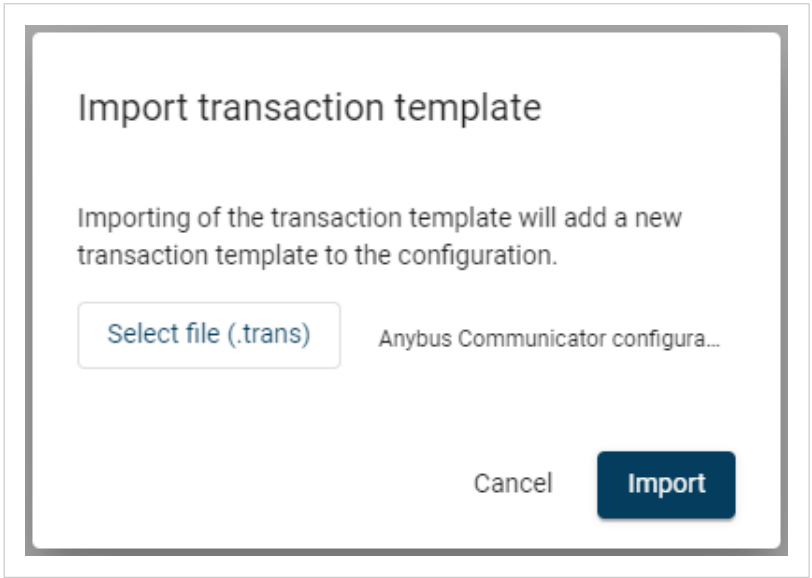


Figure 61. Example, selected .trans file

Result

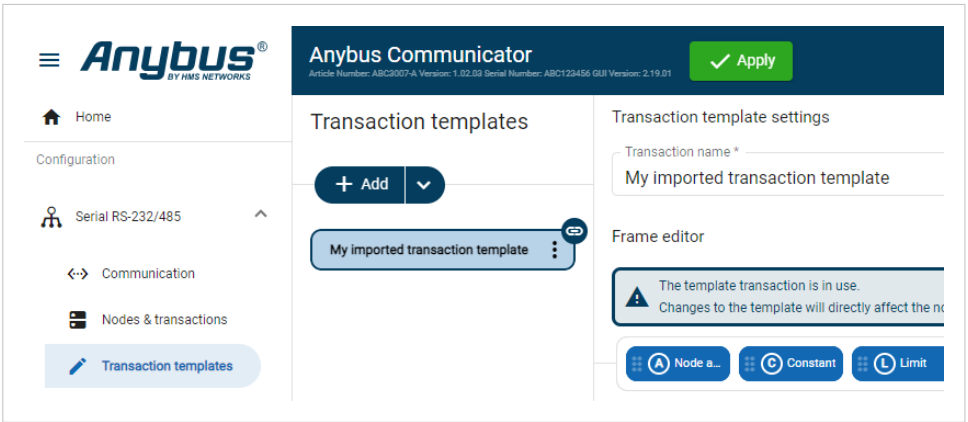


Figure 62. Example, imported transaction template

The imported transaction template is added to the **Transaction templates** list.

8.8.5. Node Settings

Before You Begin

Ensure that the Communicator Basic settings, on the **Communication** page, match the Node settings.

There are no Node settings for the Broadcast node, except Name.

Procedure

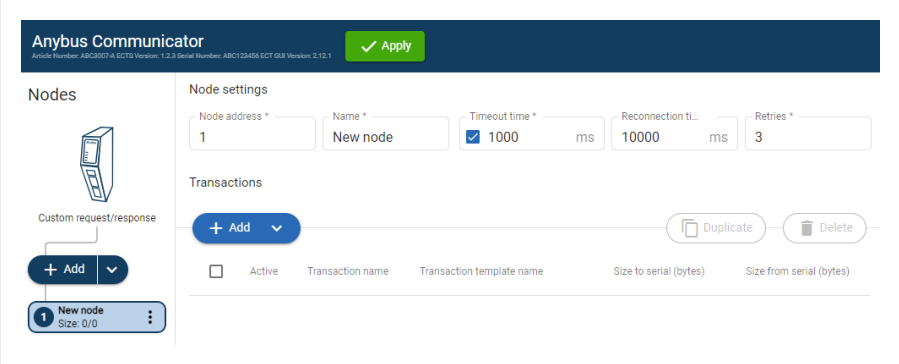


Figure 63. Nodes page, Node settings

1. In the node list, select a node to configure.
2. Configure the **Node settings**.

Setting	Value	Description
Node address	1 to 247	Node ID, also called node address, is the node's identity on the subnetwork. The node id is a number between 1 and 247. By default, the node is assigned the next available number. The same node id cannot be used on multiple nodes.
Name	N/A	By default, the node is assigned the name New node and the corresponding Server address. The node name can be changed.
Timeout time	10 ms to 10 000 ms Enabled by default Default 1000 ms	If a transaction in a transaction fulfills the specified timeout time value for all specified retries, the remaining transactions defined for the node will be skipped in the current cycle. The maximum addition to the cycle length is only one instance of the timout setting. If enabled, specify how long the Communicator should wait before sending the message again, when no response is received from the node. If the timeout time is exceeded, the Communicator continues to send the message until the maximum number of retries has been reached. If disabled, the Communicator immediately sends the message again, when no response is received from the node
Reconnection time	Min 10 ms Max 60 0000 ms Default 10 000 ms	Specify for how long the Communicator should wait before attempting to reconnect, if the node is disconnected. Reconnect time (10 ms) is not applicable for the broadcast node, that hold transactions destined to all nodes.
Retries	0 to 10 Default 3	Specify the number of attempts the Communicator should make, when no response is received from the node.
Address format	Default format: Address Register Modicon Modicon extended	Available for the Modbus RTU serial protocol. Specify the address format for the node. Address: 0, 1, 2, ... Register: 1, 2, 3, ... Modicon: 00001/10001/30001/40001 Modicon extended: 000001/100001/...

3. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

To Do Next

Add Transactions, Add Transactions (page 80).

8.8.6. Add Transactions

NOTE
The maximum number of transactions that can be added to a node is 150.

1. In the node list, select a node to configure.
2. In the transaction list, click **Add**.
3. Choose one of the following alternative:

When using the Modbus RTU Serial Protocol

- Click **Add** and select a transaction from the list of standard Modbus RTU transactions.

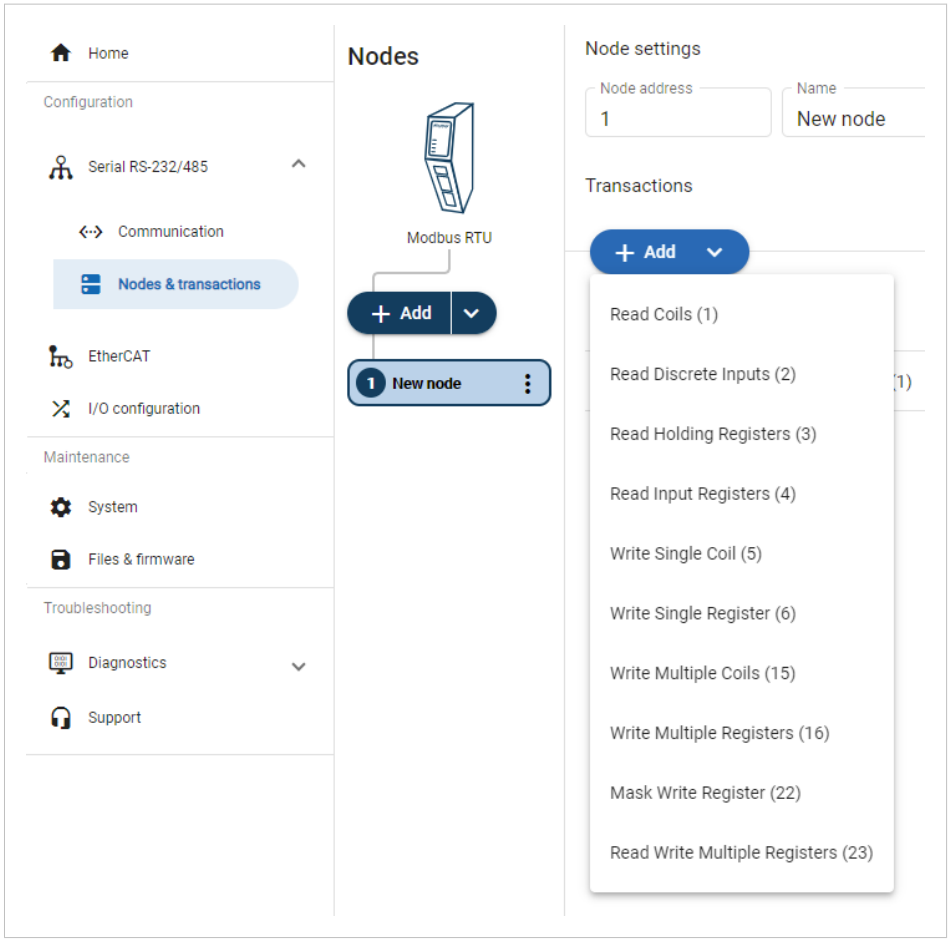


Figure 64. Add Modbus RTU transactions

When using the Request/Response or Produce/Consume Serial Protocol

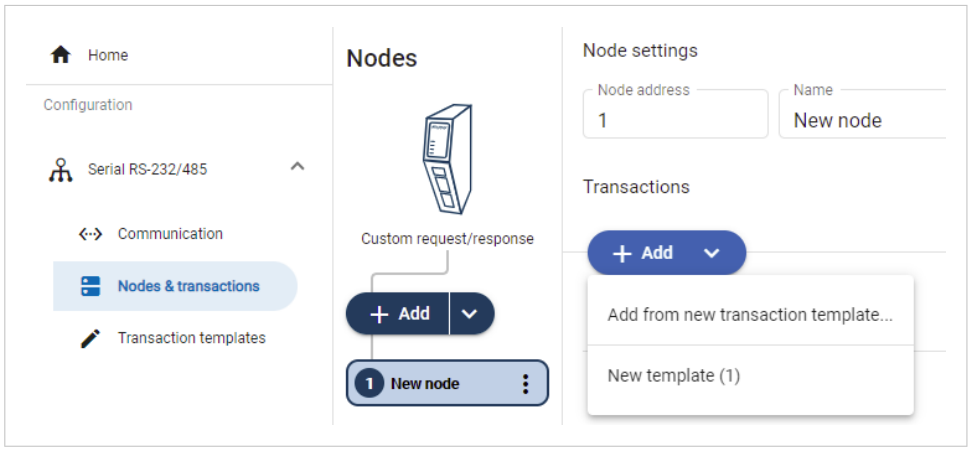


Figure 65. Add new empty template

- Click **Add** and select **Add from new transaction template**.
  You are redirected to the **Transaction template** page.
  A new empty template is added to the Transaction templates list.

> **NOTE**
> You must build the transactions before you can use the template, refer to Build Transaction Template (page 57).

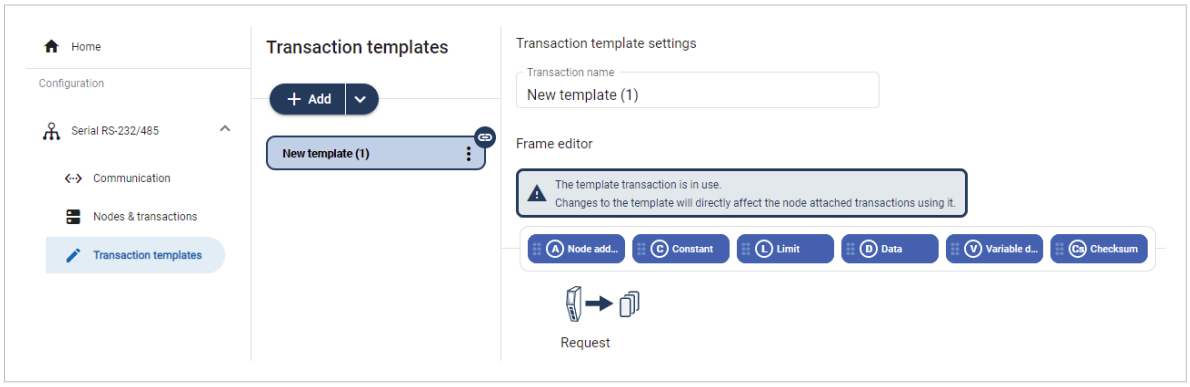


Figure 66. Add new empty template

- If you already have created Transaction templates, click **Add** and select the desired template from the list. A new transaction is added to the transactions list.

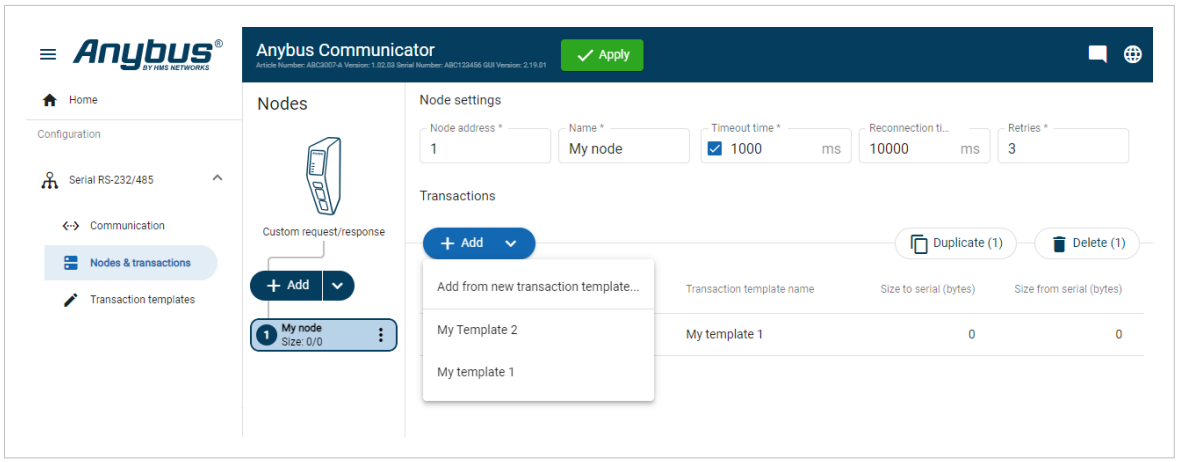


Figure 67. Add new transaction

To Do Next

Configure the Transactions settings, Transaction Settings (page 83).

8.8.7. Transaction Settings

Before You Begin

NOTE
When a custom transaction is selected, the custom transaction template is locked for editing.

For Modbus transaction reference guide, refer to Modbus Transactions (page 149).

Procedure

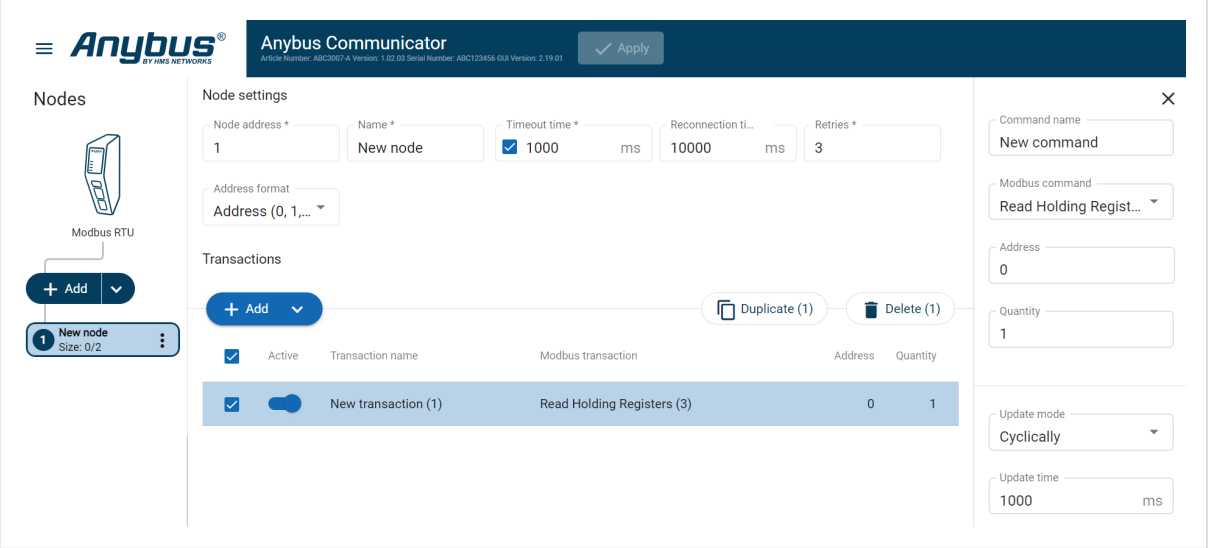


Figure 68. Modbus RTU serial protocol

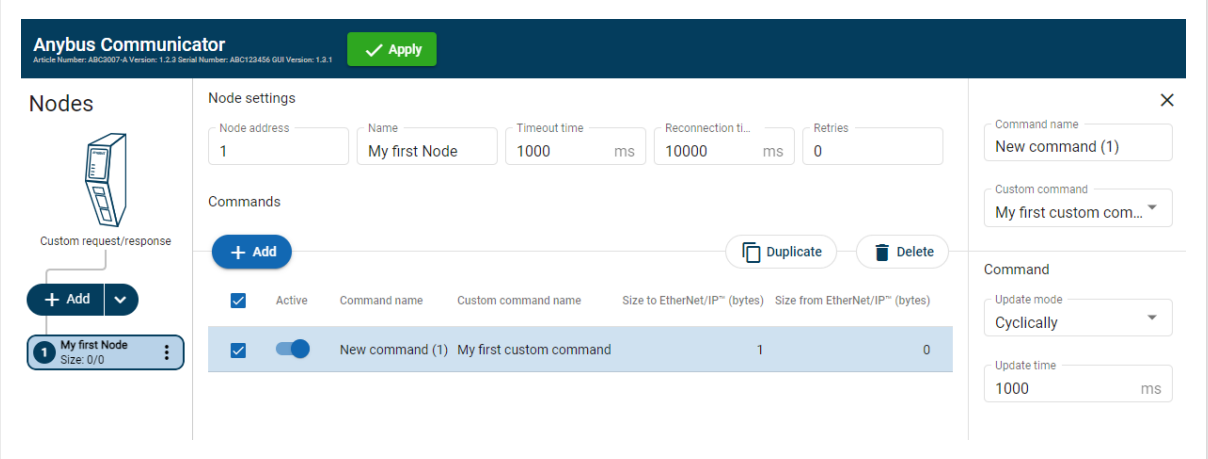


Figure 69. Custom request/response protocol

1. In the node list, select a node to configure.
2. In the transactions list, select a transaction to configure.
   The transactions sidebar opens, on the right side of the screen.
3. Enter a transaction name.
   By default, the node is assigned the name New transaction.
4. Select a transaction type from the **Modbus transaction/Custom transaction** drop-down menu.
   The transaction type defines what the node should perform when the transaction is executed.

5. Configure the Command settings.

Setting	Value	Description
Transaction name	N/A	You can name the transaction to make it easier to identify.
Read quantity	1 to 125	Specifies the number of registers to read to follow in the read data field. Appear when Modbus transaction Read Write Multiple Registers (23) is selected.
Address	0 to 65 535	Specify the start address for the read/write transaction. The address acts as an address to the data position, where the data is read from or written to. Modbus holding register addresses starts at 0. Modbus address 0 = Register 1
Write quantity	Read Write Multiple Registers (23), 1 to 123	Specifies the quantity of registers to follow in the write data field. Appear when Modbus transaction Read Write Multiple Registers (23) is selected.
Quantity	Read Holding Registers (3) Read Input Registers (4), 1 to 125 Write Multiple Coils (15), 1 to 1968 Write Multiple Registers (16), 1 to 123 Read Coils (1) Read Discrete Inputs (2), 1 to 2000	The Quantity parameter appear when you select a Modbus transaction that can address more than one data object. Example when Quantity is set: For the Modbus Transaction Read Input Registers (4) you need to set the Quantity in order to define the array of data. Example when no Quantity is set: For the Modbus Transaction Write Single Coil (5) you do not need to set the Quantity parameter because there can not be an array of data. The transaction is used to write a single output to either ON or OFF in a remote device. For Write Single Coil (5), Write Single Register (6) and Mask Write Register (22) Quantity cannot be set.
Constant	0 to 255	The value of the Constant in the frame.
Data	0 to 512	The length of the data field.
Variable data	Byte counter: 0 to 255 End pattern: 0 to 1499 None (Default): 0 to 1500	The maximum payload length of the variable data field.
Update mode	Cyclically On data change Single shot Change of state on trigger	Specify when a transaction shall be sent to the server. The transaction is issued cyclically, at the interval specified in the Update time parameter. Cyclically: The transaction is sent cyclically, at the interval specified in the Update time parameter. On data change: The transaction is sent when the data is changed. Single shot: The transaction is issued once at start up. Change of state on trigger: The transaction is triggered when the content of a specified byte changes. In the **I/O configuration**, the node will be marked with a flash icon.In the **I/O configuration** you can also configure the area map and the trigger byte address. See Trigger Byte (page 91).
Update time	3 ms to 60 000 ms	Update mode parameter must be set to Cyclically. The Update time parameter appear when Cyclically is select. Specify how often, in steps of 10 ms, the transaction are going to be issued.
Positive ack	N/A	When Positive Acknowledgement is enabled, the positive ack data byte in the **I/O configuration** is incremented each time this transaction succeeds.
Negative ack	N/A	When Negative Acknowledgement is enabled, the negative ack data byte in the **I/O configuration** is incremented each time this transaction fails.

6. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

8.8.8. Activate/Deactivate Transaction

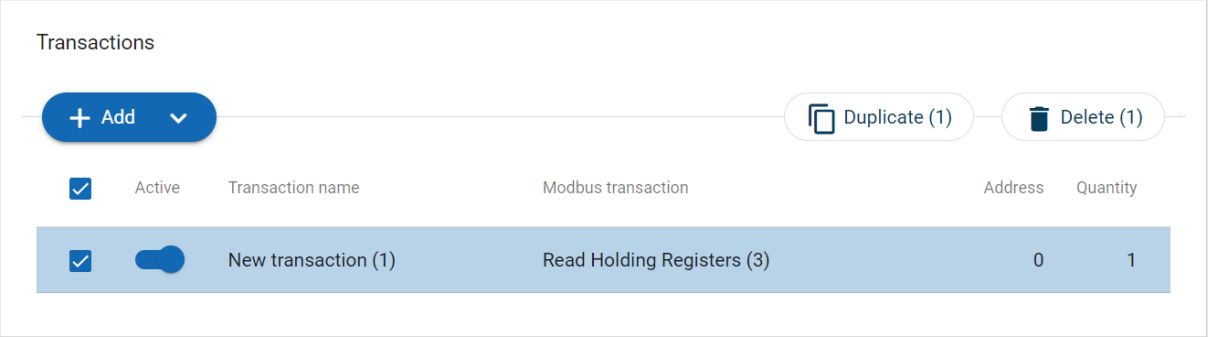


Figure 70. Activate/Deactivate Transaction

The transaction default status is **Active**.

To deactivate/activate a transaction, select the transaction and click the **slide toggle**.

8.8.9. Duplicate Transaction

When you duplicate a transaction, all settings are preserved.

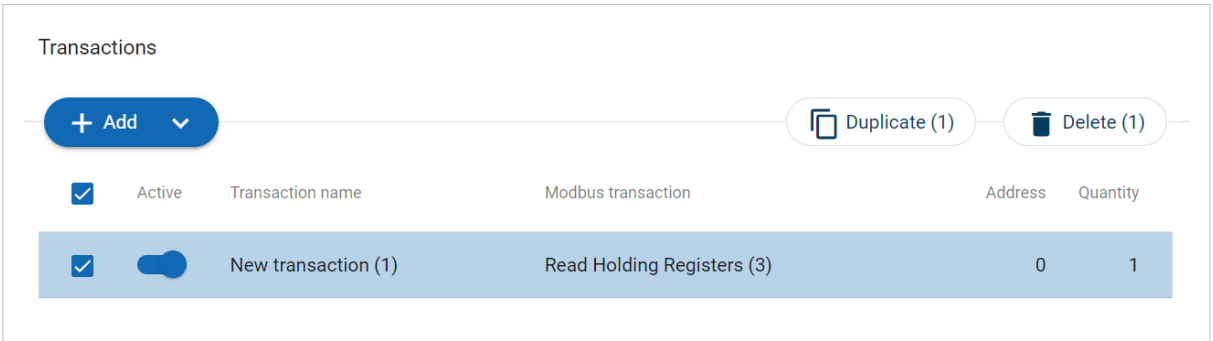


Figure 71. Duplicate transaction

To duplicate:

- One transaction, select the command and click **Duplicate**.
- Multiple transactions, select the checkbox in front of each transaction you want to duplicate and click **Duplicate**.
  The duplicated transaction is added at the bottom of the transactions list.

8.8.10. Delete Transaction

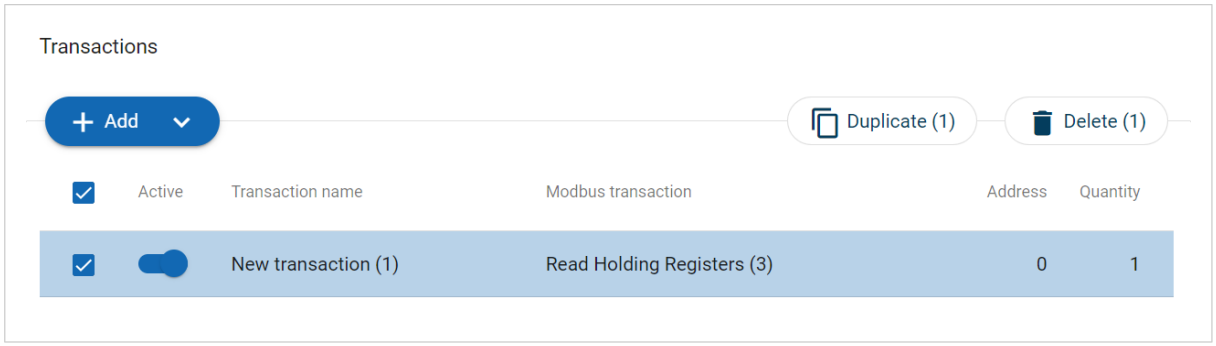


Figure 72. Delete transaction

1. To delete:
   - One transaction, select the transaction and click **Delete**.
   - Multiple transactions, select the checkbox in front of each transaction and click **Delete**.
2. To confirm, click **Yes**.

8.9. EtherCAT Network Settings

Configure the EtherCAT network settings.

8.9.1. To Use Automatic I/O Sizes

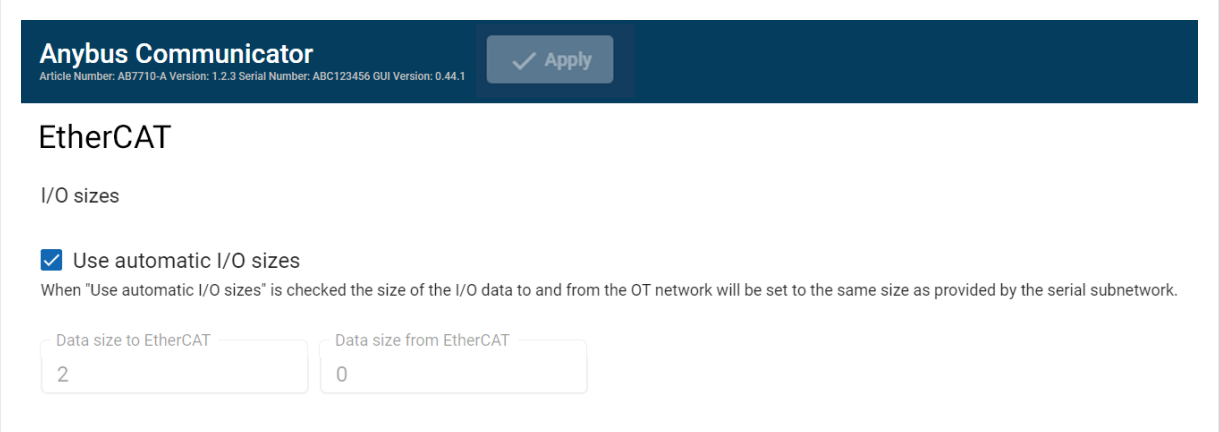


Figure 73. EtherCAT, I/O sizes

By default, the Communicator is set to use automatic I/O sizes.

The size of the input data, Data Size to EtherCAT, and the output data, Data Size from EtherCAT, is determined by the subnetwork configuration.

In the Communicator built-in web interface, the **Use Automatic I/O Sizes** checkbox is selected.

8.9.2. To Configure I/O Sizes Manually

NOTE
The maximum data size in each direction is 1486 bytes bytes.

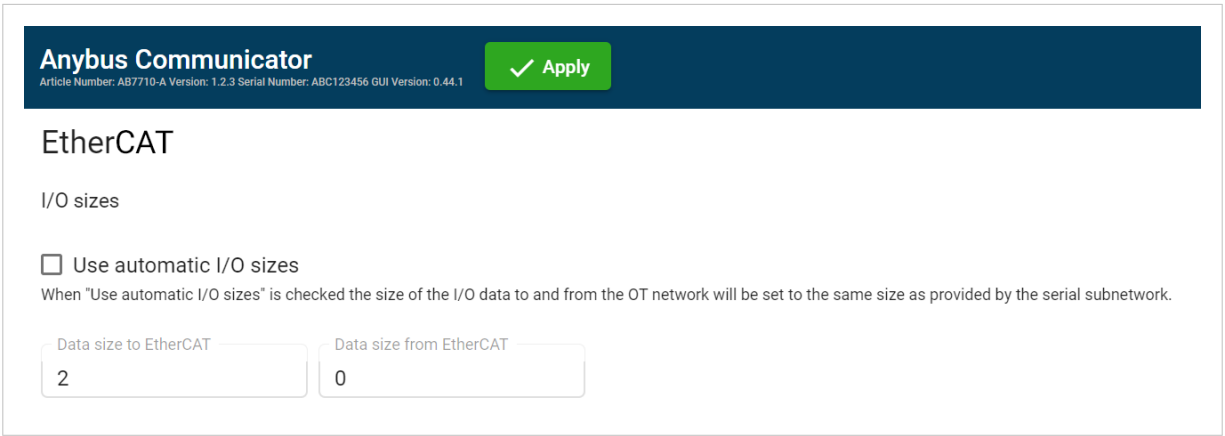


Figure 74. EtherCAT, I/O sizes

1. Deselect the **Use Automatic I/O Sizes** checkbox.
2. Enter a value for Data Size to EtherCAT and a value for Data Size from EtherCAT.

8.10. I/O Configuration

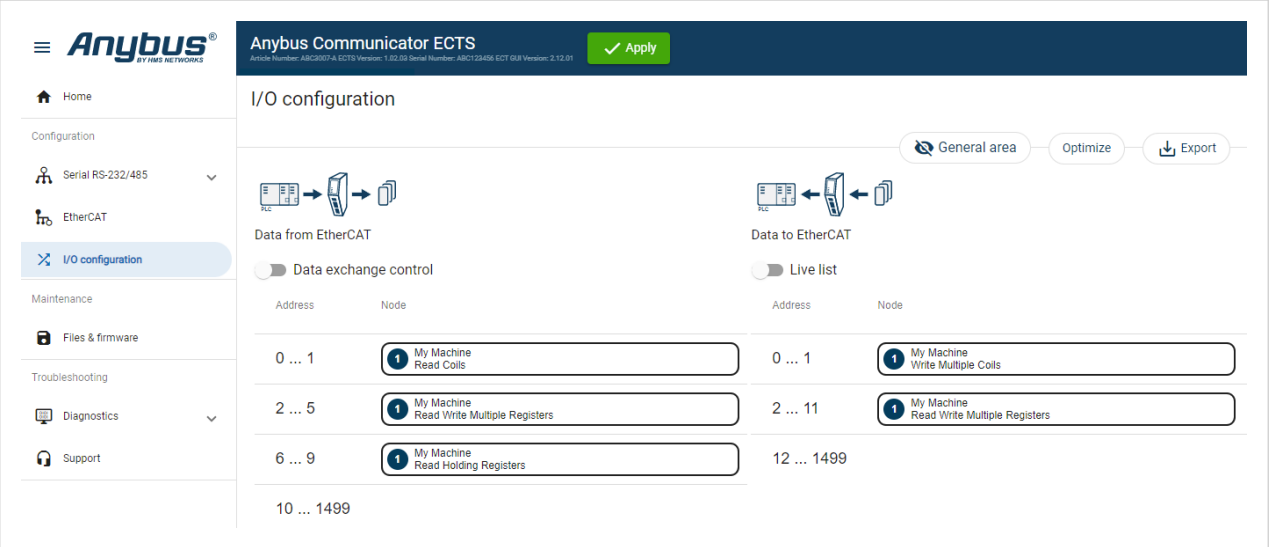


Figure 75. **I/O configuration** page

On the **I/O configuration** page the data communication between the subnetwork (Node) and the high level network (PLC) is mapped.

The allocated I/O area is auto-generated based on how the settings on the **Serial communication** page and the **Nodes and transactions** page are configured.

It is possible to set the I/O area manually, if you want to pro-actively allocate more I/O for future expansions without re-configuring the PLC. See To Configure I/O Sizes Manually (page 87).

There are three areas: **Data from EtherCAT**, **Data to EtherCAT** and **General area**. See Map Area (page 91).

8.10.1. Optimize the I/O Configuration

The optimize function is used to automatically remove gaps between the mapping.

IMPORTANT

Optimize remove gaps between the data objects in the map and should be used with care on already commissioned systems. Expected mapping in the PLC may change.

NOTE

If you optimize the I/O configuration, the current I/O configuration will be overwritten.

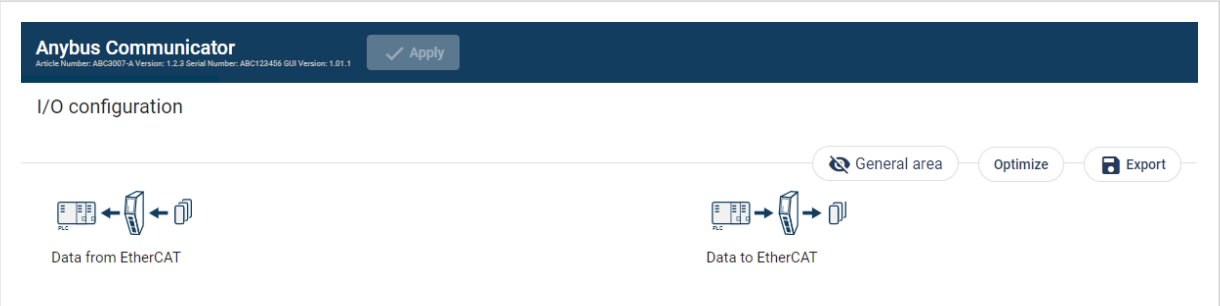


Figure 76. I/O configuration page, Optimize

To optimize the map:

1. Click **Optimize**.
2. To confirm, click **OK**.

8.10.2. Map Area Transactions Order

To change the order of the transactions in a map area, drag and drop the desired transaction to a new location.

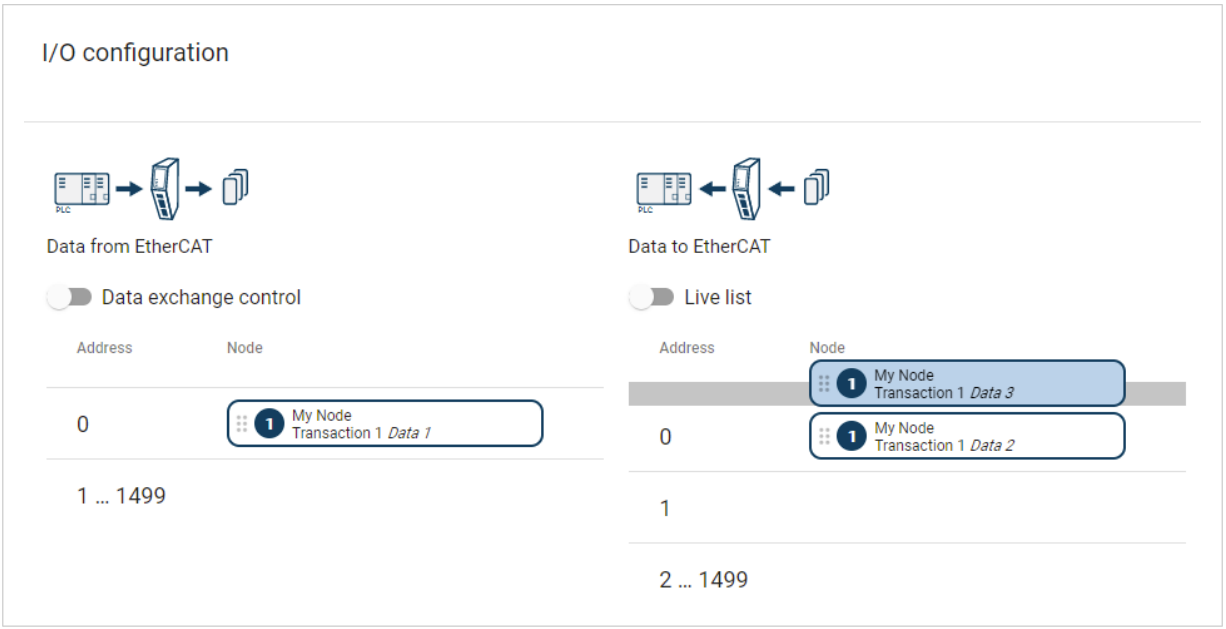


Figure 77. I/O configuration page, change the order of transactions

Transactions can not share the same I/O area.

If multiple transactions are placed in the same I/O area, the area is highlighted.

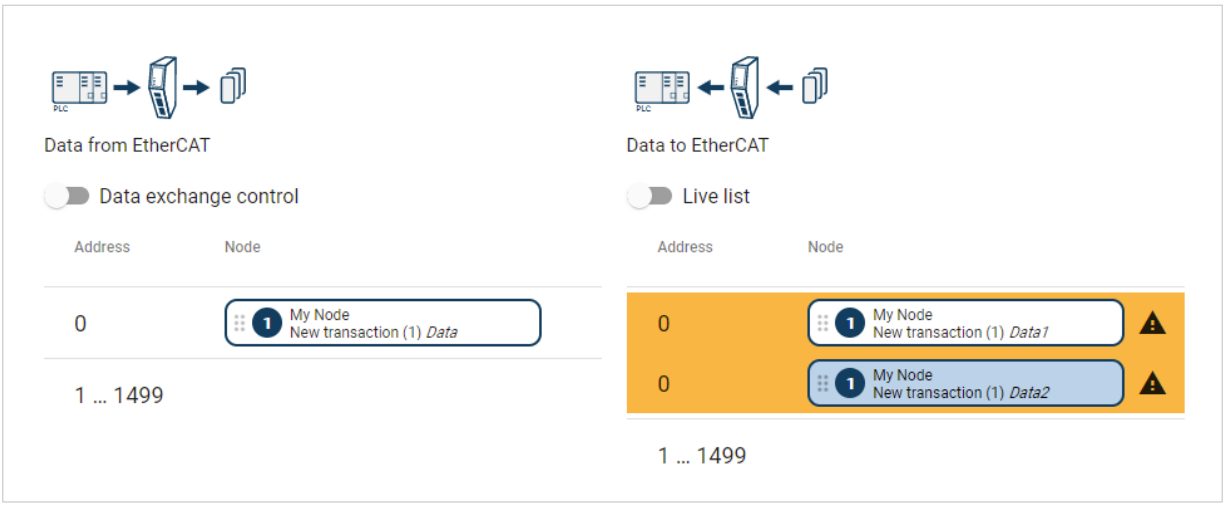


Figure 78. Highlighted I/O area

8.10.3. Map Area

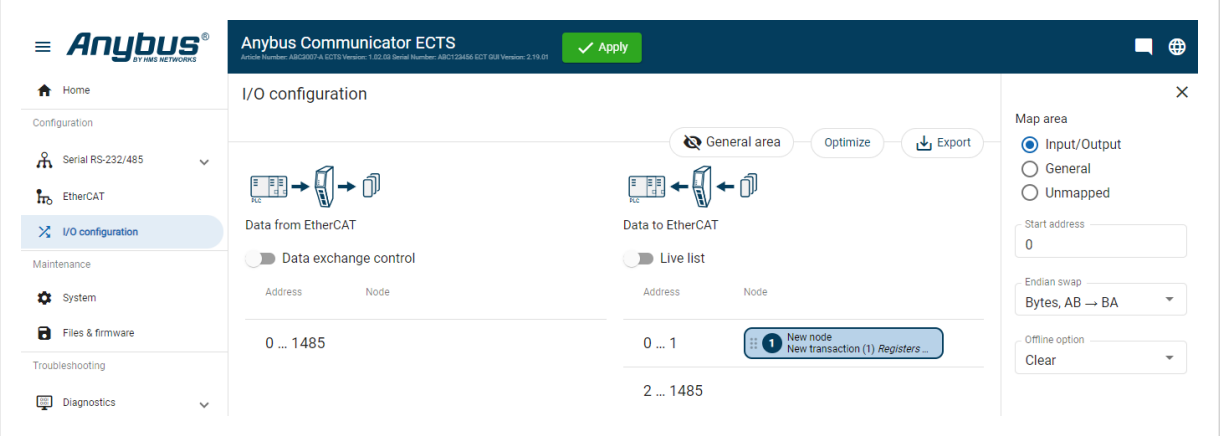


Figure 79. I/O configuration page, Map area options

Map area options
You must specify the map area to use for each transaction in the I/O configuration.

Select one of the following **Map area** options:

- **Input/Output**: The transaction data is sent/recieved to/from the high level network.
- **General**: This area is used for transferring transaction data between individual nodes on the subnetwork. When General is selected, the transaction data cannot be accessed from the high level network.
- **Unmapped**: The transaction data is not used.

Start address
For Input/Output and General, you can enter a start address for the transaction data.

8.10.4. Trigger Byte

Trigger byte is used to enable/disable the trigger functionality for the response.

When Trigger byte is enabled, the Communicator increases the trigger byte by one when the Communicator receives new data from the subnetwork.

The Trigger byte is stored in the **Data from EtherCAT** area or the **General** area.

The location of the trigger byte is specified by the address.

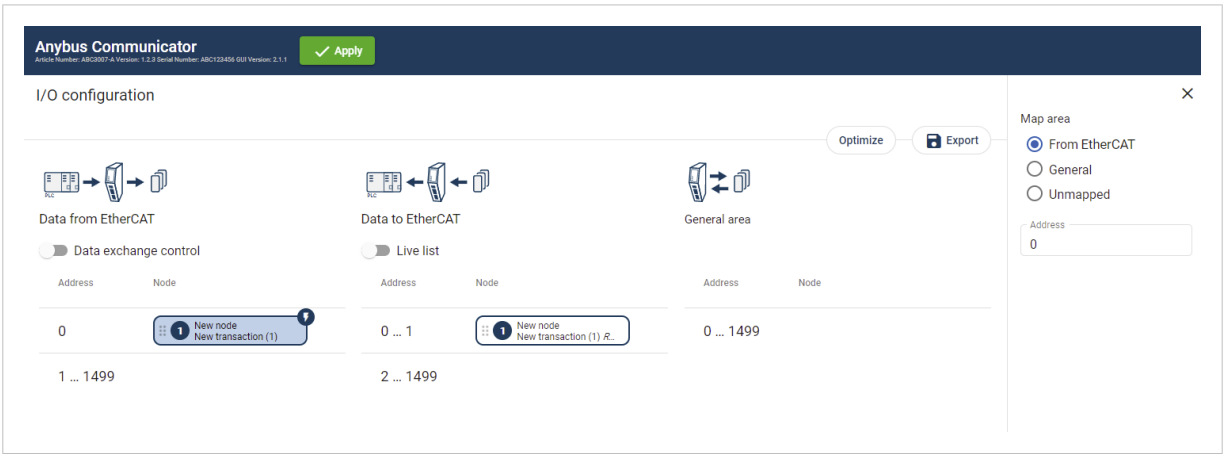


Figure 80. I/O configuration page

How to Enable Trigger Byte on a Node
Procedure

1. Navigate to the **Nodes & transactions** page.
2. Select the decried node and transaction.
3. In the transaction sidebar **Update mode** menu, select **Change of state on trigger**.
4. Navigate to the **I/O configuration** page.
5. The transaction with the trigger byte enabled is marked with a flash icon.
   To open the **Map area** sidebar, click on the flash icon.
6. In the **Map area** sidebar, specify the map area to use and the trigger byte address:
   Map area options
   - **From EtherCAT**: The trigger byte is stored in the I/O configuration **Data from EtherCAT** area.
   - **General**: The trigger byte is stored in the I/O configuration **General area**.
   - **Unmapped**: The transaction data is not used.

   Address
   - Enter an Address, the location in the specified Map area (From EtherCAT or General) where the trigger byte will be saved.
     Value: 0 (default) to 1499

8.10.5. Endian Swap

By default EtherCAT uses the little-endian format.

About Endianness

Big-endian (BE)
The big-endian format places the most significant byte of the data at the byte with the lowest memory address.

Little-endian (LE)
The little-endian format places the least significant byte of the data at the byte with the lowest memory address.

Convert Between Big-Endian and Little-Endian

To convert between big-endian and little-endian you must reverse the byte order.

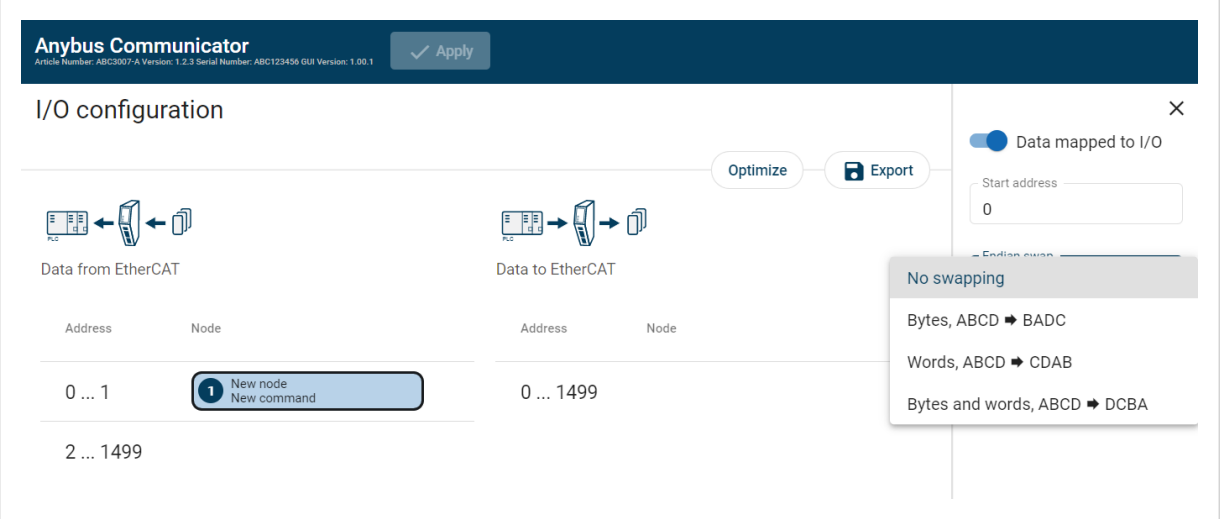


Figure 81. **I/O configuration** page, Endian swap

To reverse the byte order:

1. In the web-interface left sidebar menu, click **I/O configuration**.
2. In the data map, select the transaction for which you want to do swap the byte order.
3. Select the endian swap type from the **Endian swap** drop-down menu.

Setting	Description
No swapping	Default setting No swapping is performed on the data.
Bytes	Swap 2 bytes A B C D becomes B A D C
Words	Swap 4 bytes A B C D becomes C D A B
Bytes and words	A B C D becomes D C B A

4. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

8.10.6. Offline Option

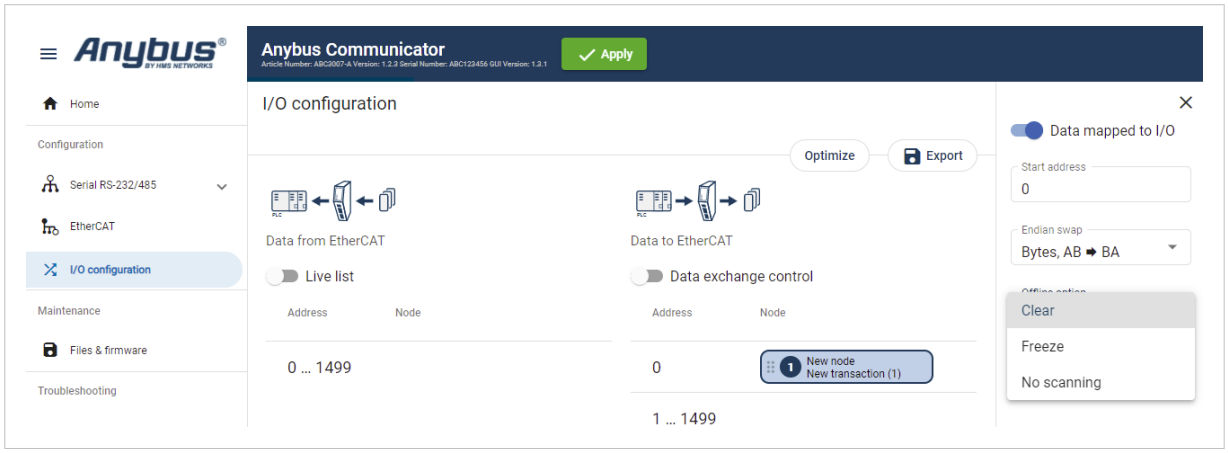


Figure 82. I/O configuration page, Offline options

Offline mode is used to define what data to send if the network connection or connection with a specific node is lost.

You must specify the offline mode to use for each transaction in the I/O configuration.

Select one of the following **Offline options**:

- **Clear (Default)**: The data is cleared and the value 0 is sent.
- **Freeze**: The Communicator holds the value until the connection is restored.
- **No scanning**: Stop sending this transaction on the sub-network. This option is only valid for produce and request transactions.

8.10.7. Live List

Figure 83. I/O configuration page, Live list enabled

By default, **Live list** is disabled.

About the Live List

- When **Live list** is enabled, the first four bytes of process data on the EtherCAT network contain the live list.
- Each bit in the **Live list** can hold the status for one node.

- The **Live list** holds 32 bits, a total of 32 nodes connected to the Communicator.
- The bit is 0 when the bit does not correspond to a configured node.
  For example, this occurs when the number of configured nodes is less than 32.
- Each bit is 1 when the corresponding nodes is online.

8.10.8. Data Exchange Control

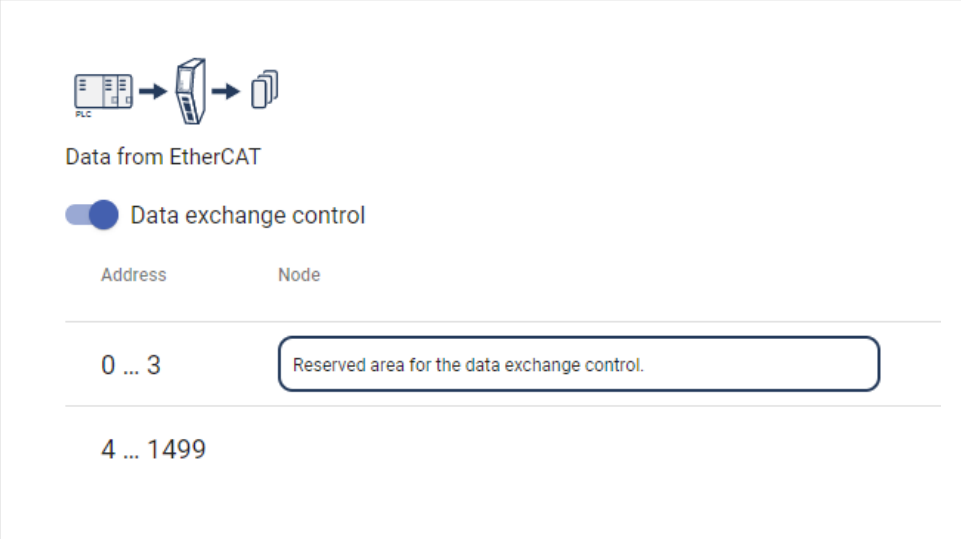


Figure 84. I/O configuration, **Data exchange control** enabled

By default **Data exchange control** is disabled.

When **Data exchange control** is enabled, the first four bytes of process data on the EtherCAT network contain the data exchange control.

The **Data exchange control** holds 32 bits.

Each bit in the **Data exchange control** can be used to enable/disable data exchange for individual node on the subnetwork.

If control bit does not correspond to a configured node, the control bit is ignored. For example, this occurs when the number of configured nodes is less than 32.

The node order in the **Data exchange control** 32 bit array always matches the Live List.

When data exchange is enabled for the corresponding node, the control bit is 1.

8.11. Configuration Notes

You can add notes to describe the Communicator configuration.

8.11.1. Add Configuration Note

Procedure

1. To open the **Configuration Notes** window, click on the **comments** icon .

Figure 85. Configuration note, comment icon

2. To add a new configuration note, click **Add**.

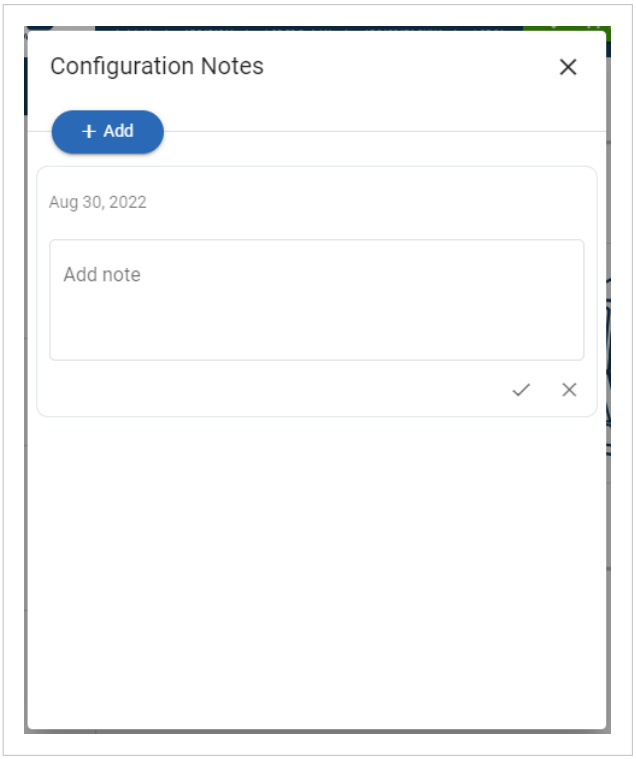


Figure 86. Add new configuration note

3. Write your configuration note and click **accept** ✓.

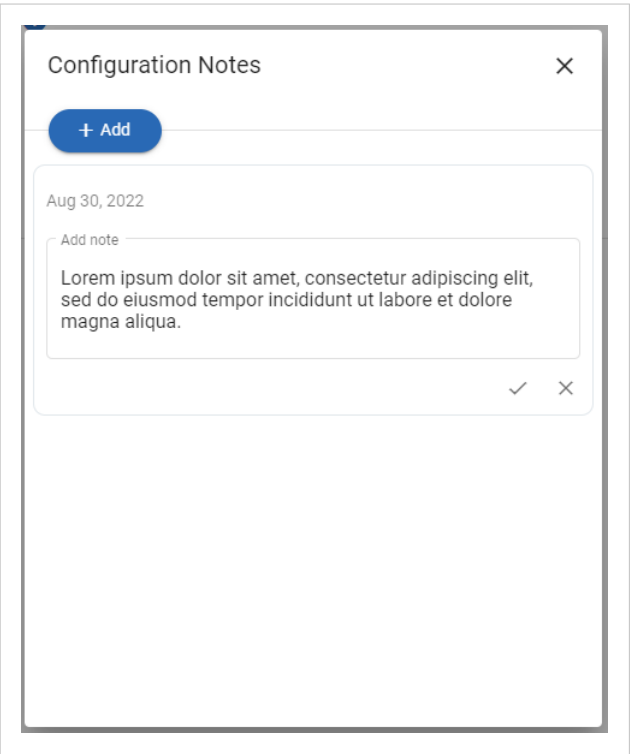


Figure 87. Write a configuration note

The configuration note is added to the list.

4. To close the window, click **close** ×.
5. To save the configuration note, click **Apply** in the web-interface header, and follow the instructions.

8.11.2. View and Edit Configuration Notes

To view and/or edit a note, click on the **comments** icon .

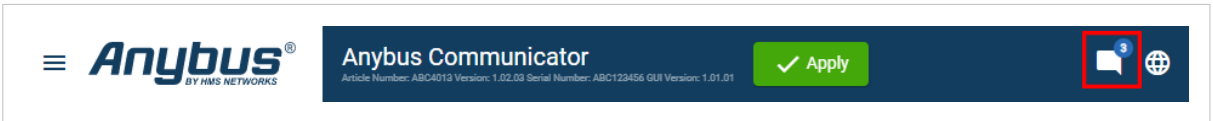


Figure 88. Example: The comment icon indicates that there are three added notes

The configuration notes are listed in the **Configuration Note** window.

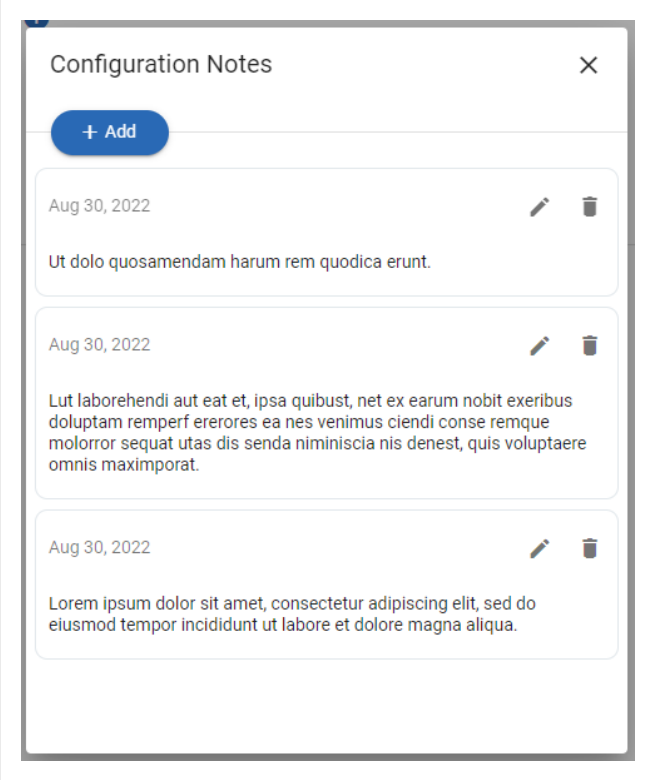


Figure 89. Example: The Configuration Notes window with added notes

8.12. Apply Configuration

Before You Begin

NOTE
When you apply the configuration, any existing configuration is overwritten.

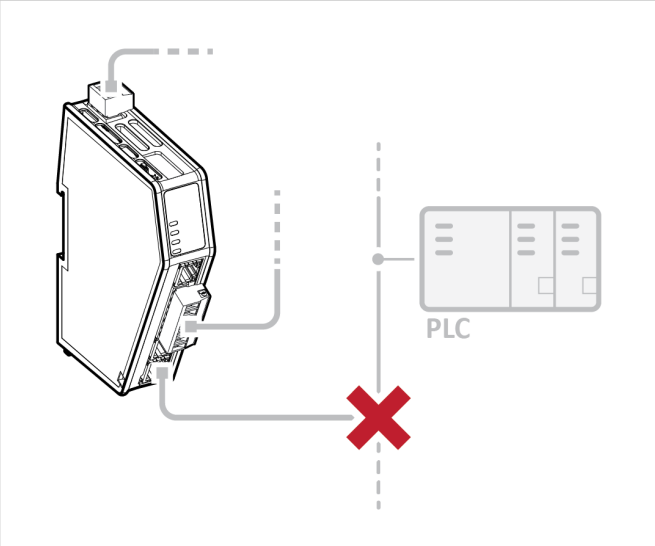


Figure 90. Disconnect the Communicator from the EtherCAT network

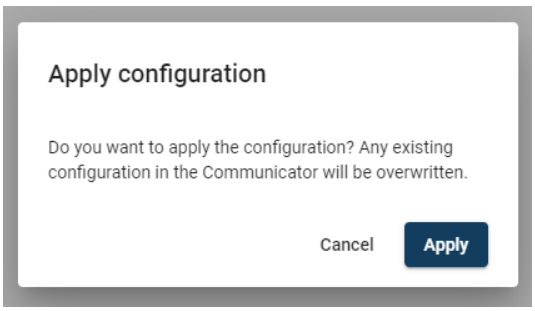
Before you can apply the configuration, ensure that there is no active communication on the EtherCAT network where the Communicator is connected.

Procedure

To make the settings take effect, download the configuration to the Communicator:

1. In the web-interface header, click **Apply**

2. To confirm download, click **Apply**.
   The configured settings are downloaded and applied to the system.



8.13. To Use an Existing Configuration

When you have configured a Communicator and want to use the same settings to configure additional Communicator, do the following.

Procedure

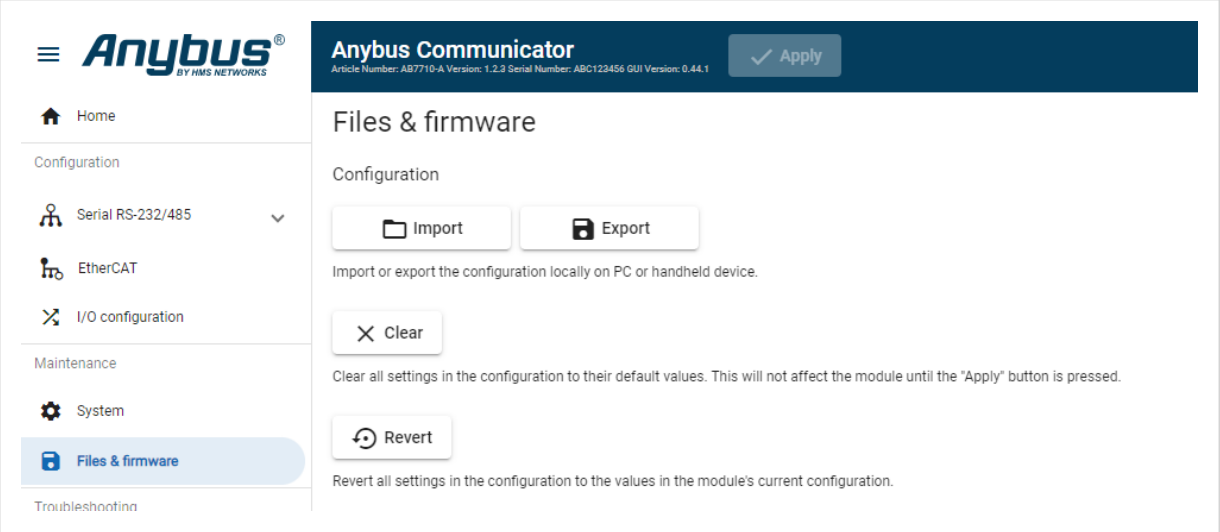


Figure 91. **Files & firmware** page

In the built-in web-interface of the Communicator with the configuration you want to use:

1. On the **Files & firmware** page, click **Export**
   The configuration is saved in a configuration file and downloaded to your PC.

In the built-in web-interface of the new Communicator to be configured:

2. On the **Files & firmware** page, click **Import**
3. In the Import configuration window, click **Select file (.conf)**.
4. In the Open dialog box, browse to and select the configuration file and click **Open**.
5. To import the configuration file, click **Import**.

Result

All the configuration settings are imported.

To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

8.14. To Use a Communicator Classic Configuration

Before You Begin

NOTE
Only the Communicator Classic serial configuration settings can be imported.

The I/O data map and high-level network settings are not supported and must be set manually in the Communicator built-in web interface.

Communicator Classic Classic configuration intended use

The intended use of the Communicator Classic configuration import is to:

- convert custom protocols.
- get a new Communicator unit up and running quickly and then complete the configuration in the Communicator built-in web interface.

Configuration files containing standard Modbus RTU commands

- It is not recommended to import Communicator Classic configuration files containing standard Modbus RTU commands if you need to be able to easily make change to the imported configuration.
- When standard Modbus RTU commands are imported, they are converted to Custom Request/Response transactions. See Communication Serial Protocol (page 42).
- The behavior of the imported standard Modbus RTU commands is preserved, but adding or changing Modbus RTU commands in Custom Request/Response mode is more difficult than in Modbus RTU mode.

Procedure

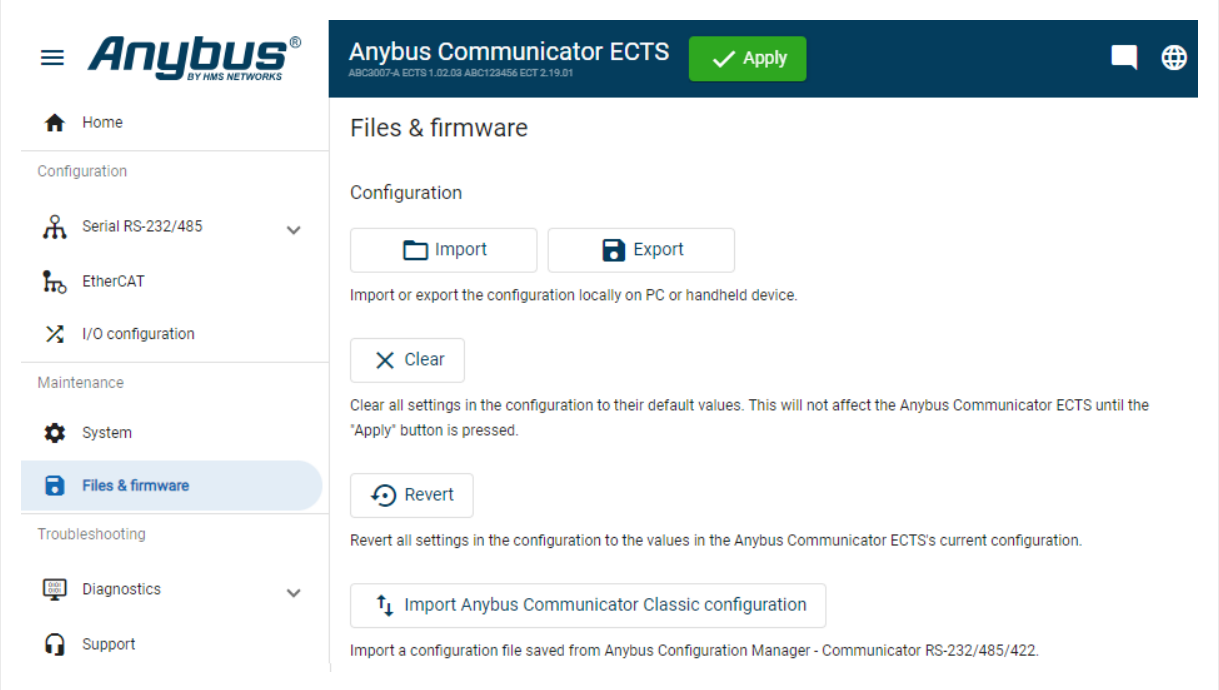


Figure 92. **Files & firmware** page

1. On the **Files & firmware** page, click **Import Anybus Communicator Classic configuration**
2. In the **Import Anybus Communicator Classic configuration** window, click **Select file (.cfg)**.

3. In the Open dialog box, browse to and select the configuration .cfg file and click **Open**.
4. If you want to import a name file, click **Select name file (.cfx)**.
5. In the Open dialog box, browse to and select the configuration .cfx file and click **Open**.
6. To import the configuration, click **Import**.

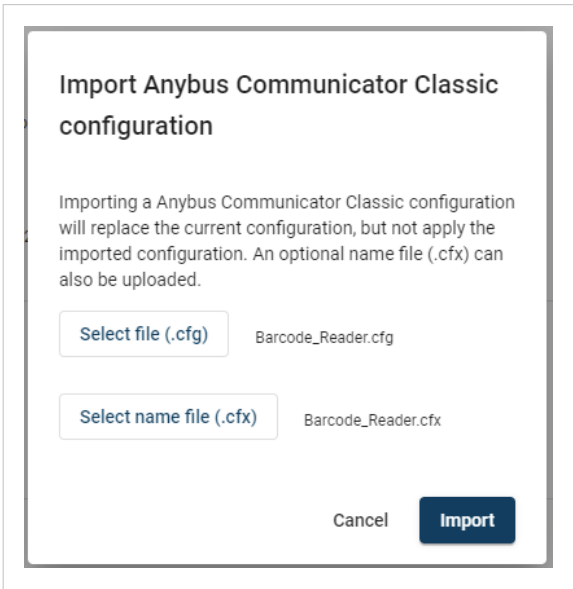
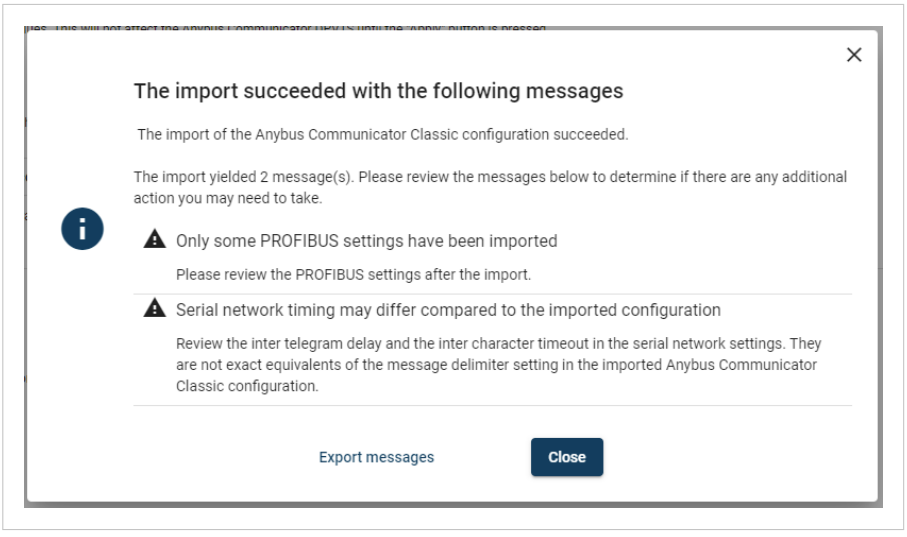


Figure 93. Example, selected .cfg and .cfx files

Result

The Communicator Classic serial configuration settings are imported.



A window with messages about the imported configuration appear.

In the list you can view the settings that are fully supported or adjusted to work with Communicator and which settings that are not supported and must be set manually in the built-in Communicator interface.

To export the messages in an Excel XLS file, click **Export Messages**.

To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

Figure 94. Example, list with messages about the import

9. PLC Configuration

9.1. PLC Device Security

IMPORTANT
It is important to maintain the cybersecurity of the Communicator.

Before connecting the Communicator to a PLC, ensure the PLC is configured and installed in accordance with the PLC supplier hardening guidelines.

9.2. Export I/O Configuration

When configuring the communication between the PLC and the nodes on the subnetwork, use the I/O data map as a specification to ensure that the transactions match.

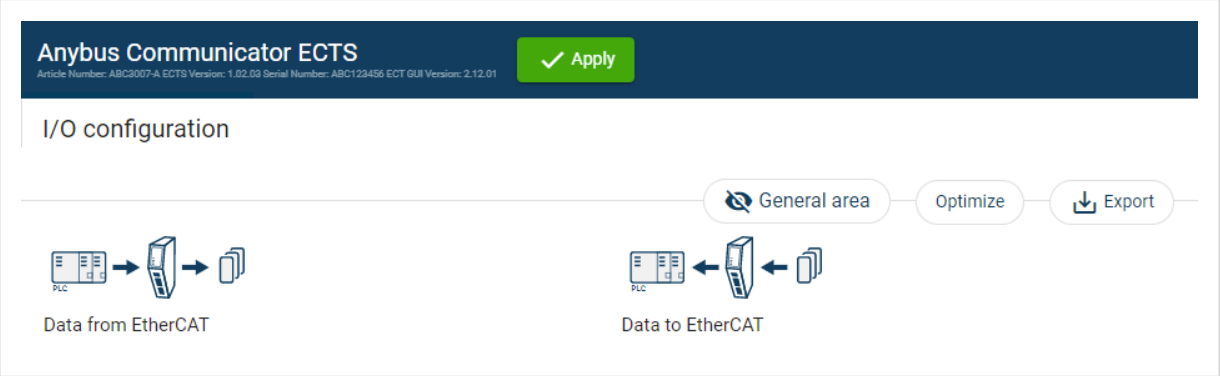


Figure 95. I/O configuration page

On the **I/O configuration** page you can export the I/O data map in an Excel XLS file, where all the nodes and transactions are listed.

To export the I/O data map:

1. Click **Export**.
   An Excel XLS file with the mapping is downloaded to your PC.

9.3. Export Product ESI File

Option for EtherCAT SubDevice.

Option if the PLC program requires a product file, ESI (EtherCAT SubDevice Information) file to configure the EtherCAT PLC to use the Communicator

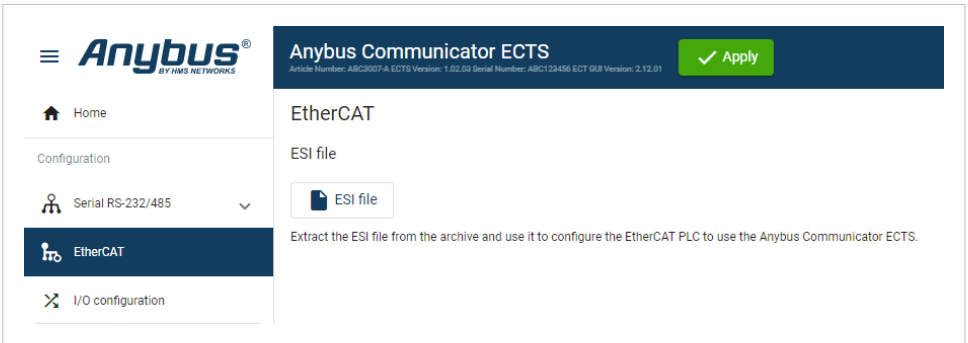


Figure 96. Export Product ESI File

You find the *EtherCAT ESI* file on the Communicator built-in web interface **EtherCAT** page, **Files & firmware** page and on the **Support** page.

To export the ESI file:

1. To save the configuration, click **Apply**.
   The **ESI file** button is activated.
2. Click **ESI file**.
   The ESI file is downloaded to your PC.

10. Verify Operation

Before You Begin

Ensure that the Communicator is connected to your PC, to a power supply and to the OT network.

See Installation (page 15).

10.1. Communicator Status Monitor

On the Home page, you can get a quick overview of the network and the Communicator operating status.

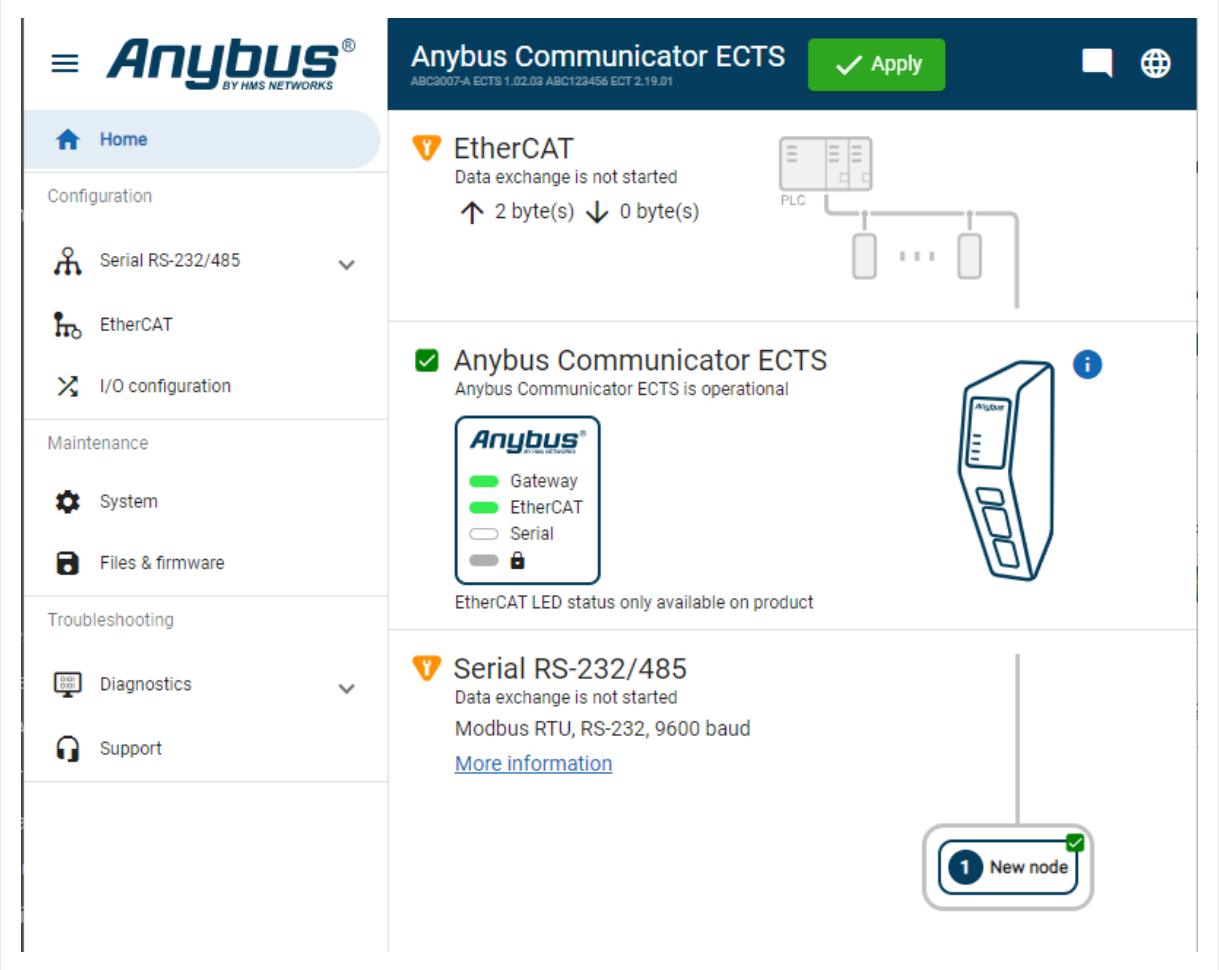


Figure 97. Home page

Gateway Status

Overview the Communicator LED indications remotely.

The Communicator EtherCAT LED status indication is only available on product.

Refer to Communicator LED Indicators (page 108).

Node Status

Overview the status for each node added to the subnetwork.

Network Status and Settings

Overview communication status and the current networks settings.

Status Symbols

Symbol	Description
	Internal error has occurred, and operation cannot be guaranteed.
	Out of Specification.
	Check Function: • Initial state where non network components are started and configured. • Network startup in progress. • Invalid configuration detected.
	Normal operation.

10.2. Communicator LED Indicators

NOTE
Before you can verify operation, you must configure the Communicator.

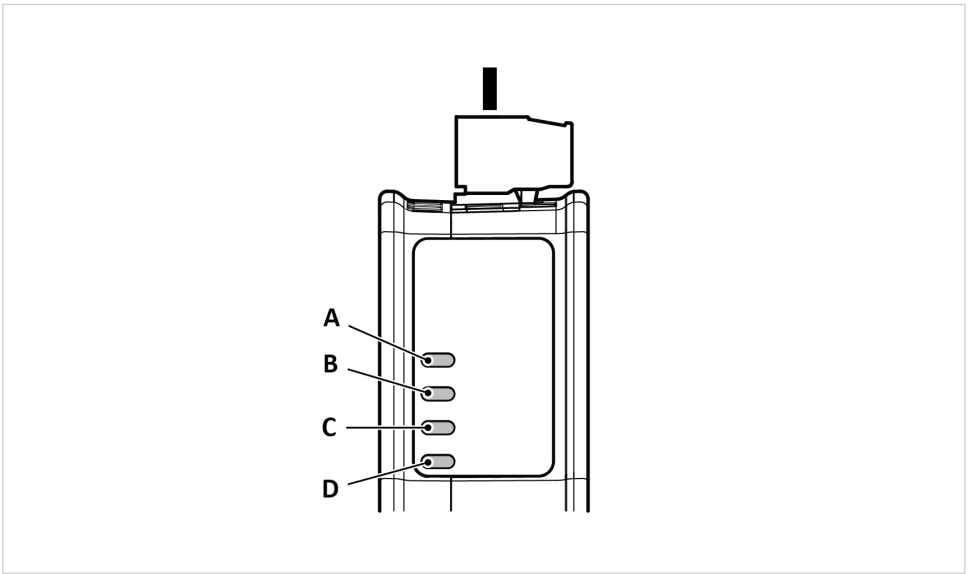


Figure 98. Communicator status (A), High level Network/Client (B), Subnetwork 2 (C) and (D) Security Switch

	LED A	LED B	LED C	LED D
Operation status	**Gateway status**	**EtherCAT**	**Subnetwork**	**Security switch**
Off	No power	No power/No IP address	No power/Subnetwork not running/Node is switched off via a control word	No power/Security switch is unlocked
Green, flashing	Startup phase	EtherCAT online, no connections established	Running, one or more nodes are offline	N/A
Green, solid	Operational	EtherCAT on	Running	Security switch is locked
Red, solid	N/A	N/A	N/A	N/A
Red, one flash	N/A	Unsolicited state change SubDevice application has changed the EtherCAT state autonomously	N/A	N/A
Red, two flash	N/A	Sync Manager watchdog timeout	N/A	N/A
Red, flashing	Invalid configuration	Invalid configuration	All nodes are offline	N/A
Green/Red, flashing	Power up self-test/ Firmware update/Firmware recovery	EtherCAT RUN (green) and ERROR (red) LED combined*	N/A	N/A

*The EtherCAT RUN (green) and ERROR (red) LED behaviors are combined in LED (B). This can cause LED (B) to alternate between red and green. The LED behavior still represents the states described in the table above.

Fatal Error and Exception Error

Fatal error: A fatal error causes the Communicator firmware application to crash in an uncontrolled manner.

Exception error: An exception error causes the Communicator to enter a controlled error state. The Communicator firmware application is still running.

LED	Fatal error	Exception error
A	Red, solid	Red, solid
B	Red, solid	Off
C	Red, solid	Off
D	Off	Off

10.3. EtherCAT LED Indicators

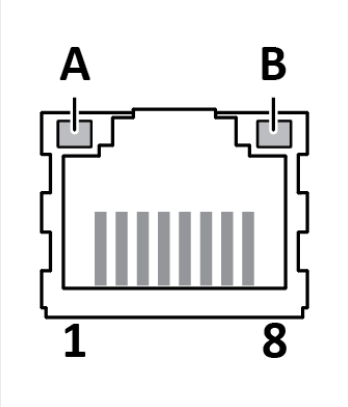


Figure 99. LED A. Activity LED B. Not used

LED A	Function
Off	No link (or no power)
Green	Link established
Green, flashing	Activity

LED B	Function
Off	Not used

11. Use Cases

11.1. Temperature Regulator - Modbus RTU Use Case

11.1.1. About the Use Case

The purpose of this use case is to explain how to use the **Modbus RTU** serial protocol.

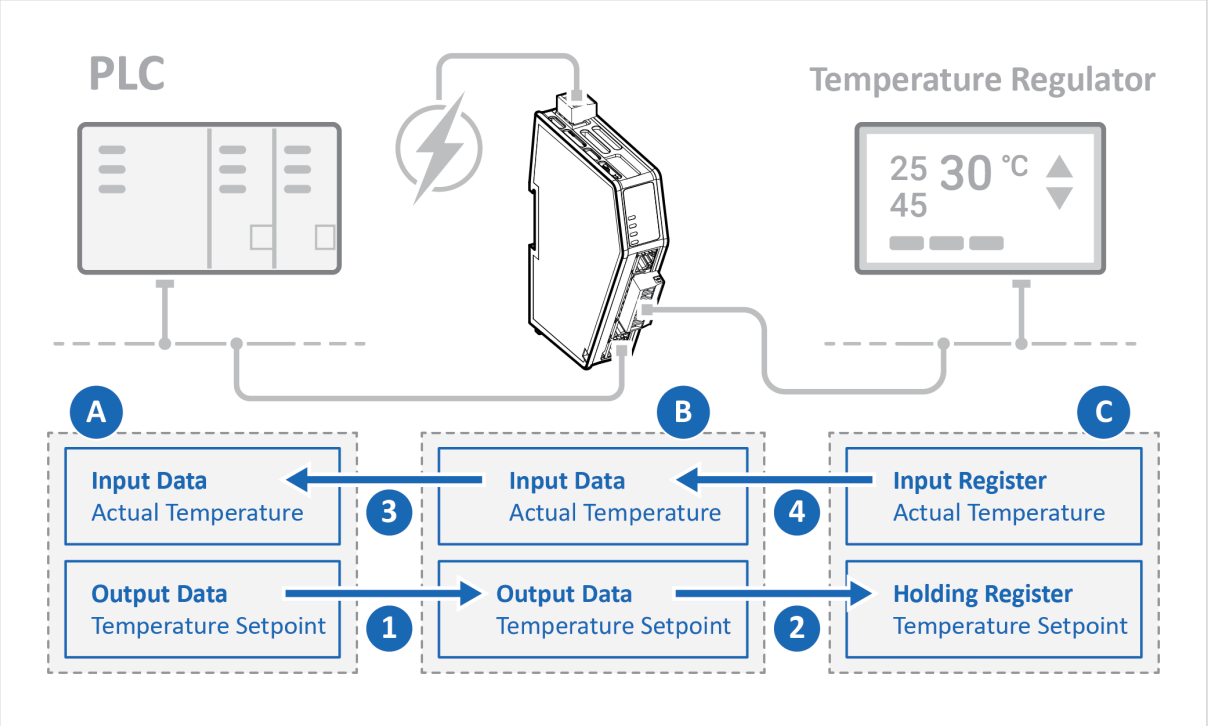


Figure 100. Temperature Regulator - Modbus RTU Use Case

In this use case we use the Communicator to enable data exchange between an Temperature Regulator and a PLC.

The use case describes how to map the communication in the Communicator.

The Temperature Regulator is connected to the serial subnetwork via a custom RS-232 protocol.

The PLC is connected to an EtherCAT network (high level network).

11.1.2. Before You Begin

- Connect the Communicator configuration port to your computer.
- Power on the Communicator.
- Ensure that your computer can find the Communicator IP address.
- Enter the Communicator built-in web interface of the.

For more information refer to Communicator Configuration (page 37).

11.1.3. Choose Serial Protocol Type

The Temperature Regulator is using a request/response protocol to access parameters addressed with index and sub index.

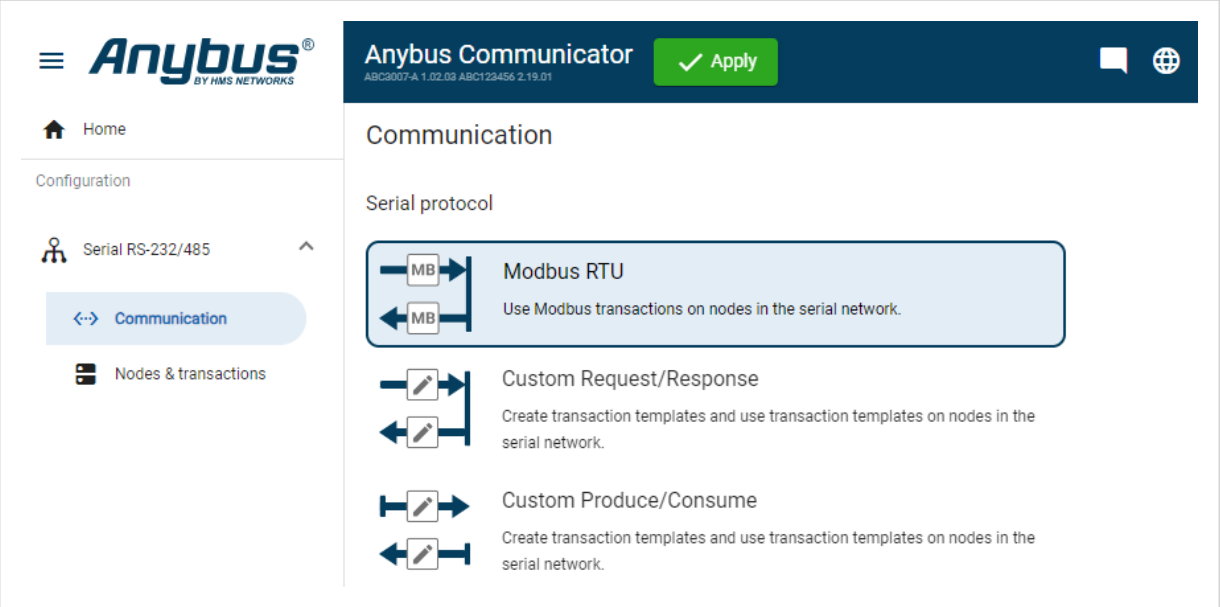


Figure 101. Communication page, **Modbus RTU**

On the **Serial RS232/485** page, select **Modbus RTU**.

11.1.4. Setup Serial Communication

Set up the communication between the Communicator and the Temperature Regulator.

In the **Serial RS232/485** page, configure the **Communication** settings.

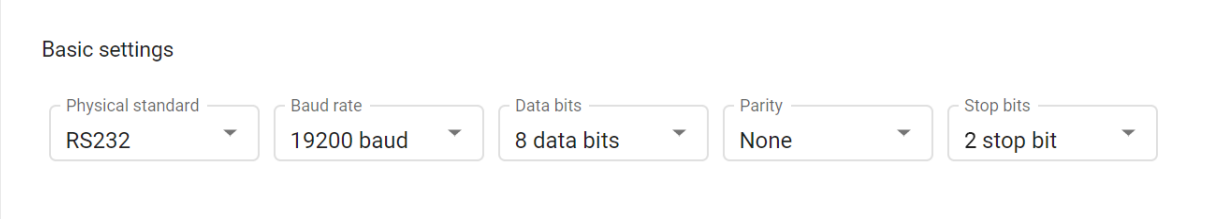


Figure 102. Serial RS232/485, Basic settings

Table 1. Used the following settings:

Frame objects	Value
Physical standard	RS-232
Baud rate	19200 baud
Data bits	8 bits
Parity	None
Stop bits	2 stop bit

11.1.5. Setup the Node

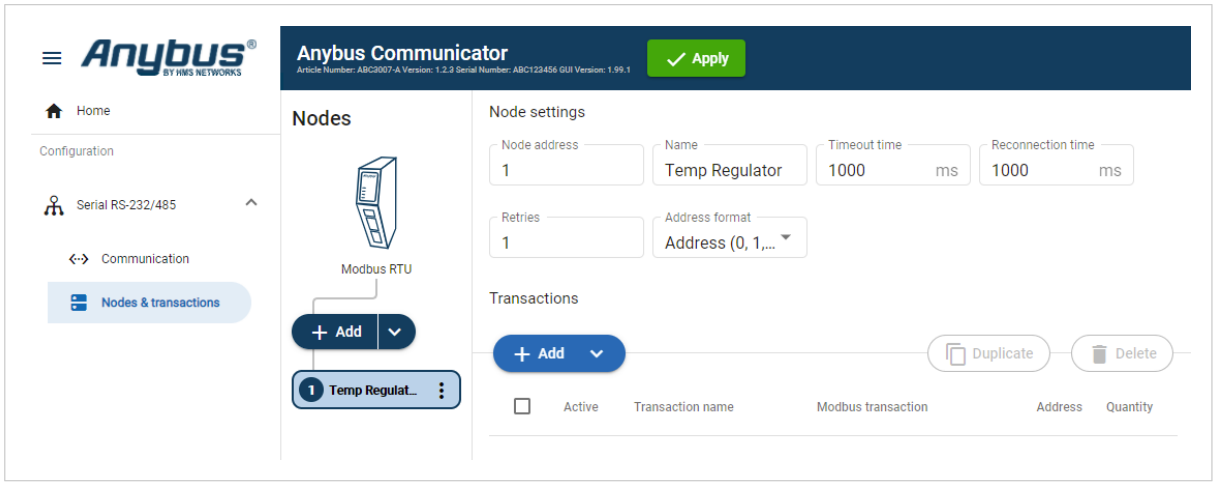


Figure 103. Add the Temperature Regulator node

1. Add a node and select it.
2. In Node settings, configure the node with the following settings:

Node settings	Value
SubDevice address	240
Name	Temp Regulator
Timeout time	1000 ms
Reconnection time	1000 ms
Retries	1
Address format	Register

11.1.6. Setup the Transactions

Set up the communication between the node and the master.

In this example, the communication between the Temperature Regulator and the PLC.

The Temperature Regulator has two Modbus transactions:

- One registers holding the setpoint temperature.
- One registers holding the actual temperature.

Configure the temperature setpoint transaction

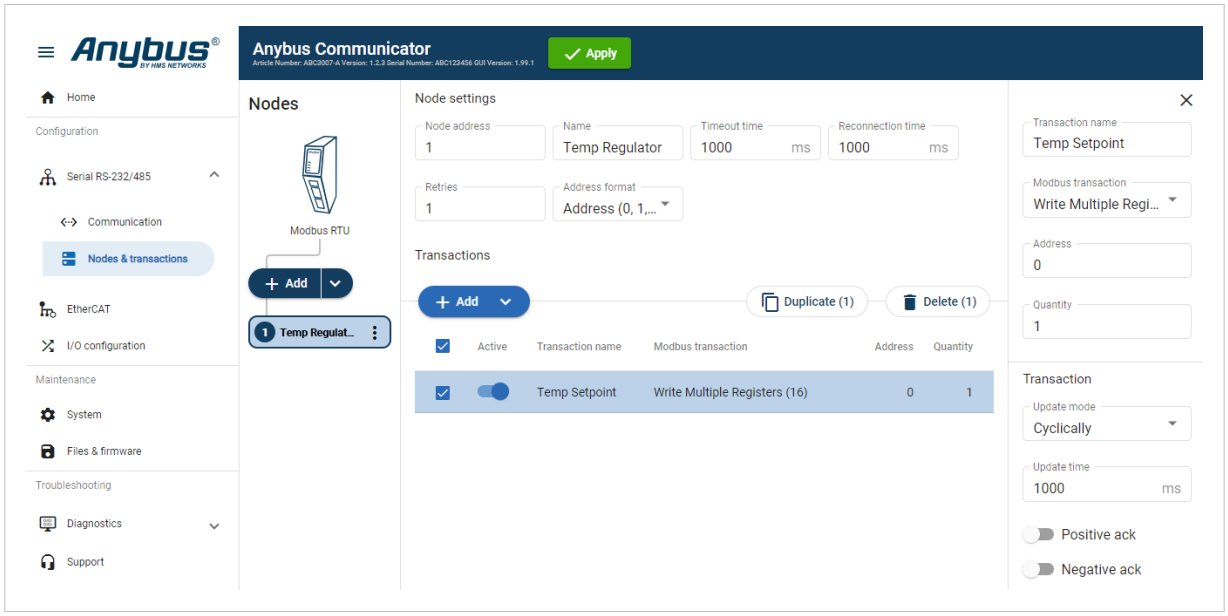


Figure 104. Temperature setpoint transaction

1. Select the **Temp Regulator** node.
2. To add a transaction, click **Add**.
3. Select the transaction to configure.
4. In the transaction sidebar, on the right side of the screen.
   Enter values for the transaction settings.

   Table 2. Setpoint temperature transaction settings

Setting	Value	Description
Transaction name	Temp Setpoint	Give the transaction a name.
Modbus transaction	Write Multiple Registers (16)	The PLC writes a block of contiguous registers to the temperature regulator.
Address/ Register	Address: 0 Register: 1	Address 0 is Register 1.
Quantity	1	The transaction will address one data object.
Update mode	Cyclically	The temperature regulator sends a new message cyclically, every 1000 ms.
Update time	1000 ms	The update cycle is 1000 ms.

Configure the actual temperature transaction

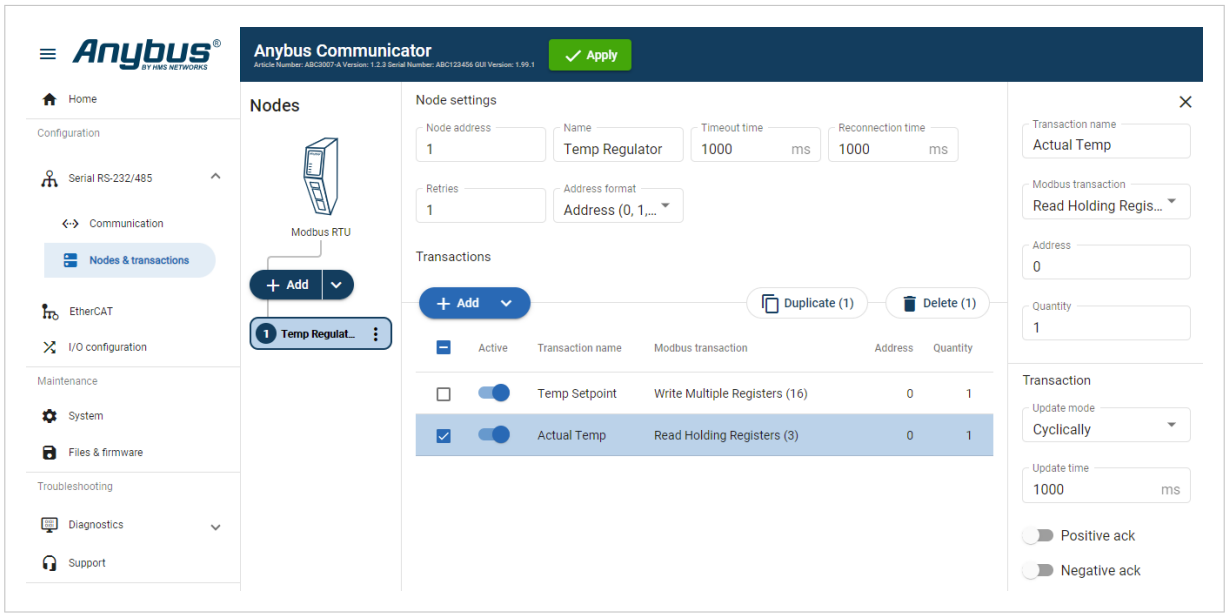


Figure 105. Actual temperature transaction

1. To add a second transaction, click **Add**.
2. Select the transaction to configure.
3. In the transaction sidebar, on the right side of the screen.
   Enter values for the transaction settings.

   Table 3. Actual temperature transaction settings:

Setting	Value	Description
Transactio name	Actual Temp	Give the transaction a name.
Modbus transaction	Read Holding Registers (3)	This register read the actual temperature from the temperature regulator to the PLC.
Address	Address: 0 Register: 1	Address 0 is Register 1.
Quantity	1	The transaction will address one data object.
Update mode	Cyclically	Default value, can not be changed.
Update time	1000 ms	The update cycle is 1000 ms.

11.1.7. Check the I/O Configuration

The transactions to and from the Temperature Regulator are mapped as follows in the **I/O configuration** page.

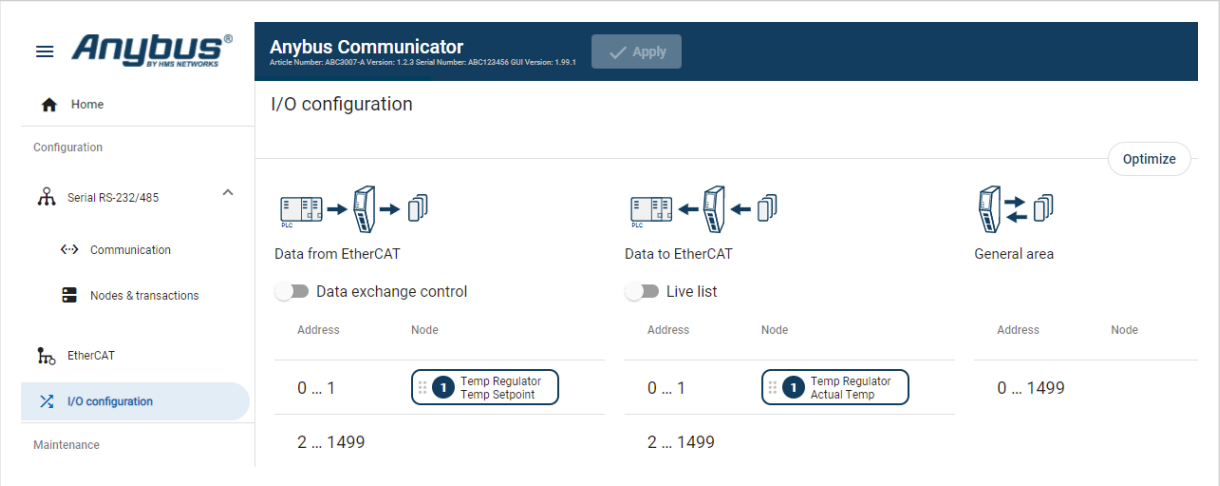


Figure 106. I/O configuration page

Address	Data to EtherCAT
0-1	Setpoint temperature from EtherCAT to the Temperature Regulator.

Address	Data from EtherCAT
0-1	Actual temperature from the Temperature Regulator to EtherCAT.

11.2. AC Motor Drive - Custom Request/Response Use Case

11.2.1. About the Use Case

The purpose of this use case is to explain how to use the **Custom Request/Response** serial protocol.

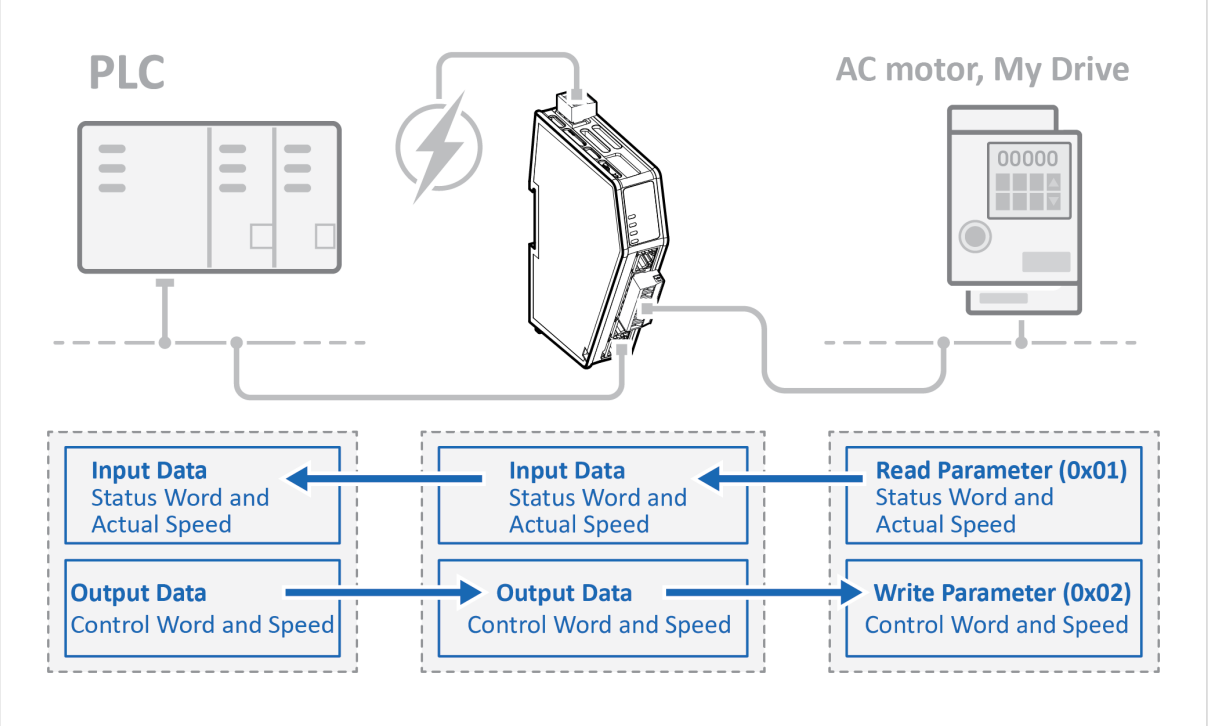


Figure 107. AC Motor Drive - Custom Request/Response Use Case

In this use case we use the Communicator to enable data exchange between an AC motor, of the type My Drive, and a PLC.

The use case describes how to map the communication in the Communicator.

My Drive is connected to the serial subnetwork via a custom RS-485 protocol.

The PLC is connected to an EtherCAT network (high level network).

We use the Custom Request/Response serial protocol and create customized transaction template to map up:

- Status word and actual speed from My Drive to the EtherCAT network.
- Control word and speed from the EtherCAT network to My Drive.

11.2.2. Before You Begin

- Connect the Communicator configuration port to your computer.
- Power on the Communicator.
- Ensure that your computer can find the Communicator IP address.
- Enter the Communicator built-in web interface of the.

For more information refer to Communicator Configuration (page 37).

11.2.3. Choose Serial Protocol Type

My Drive is using a request/response protocol to access parameters addressed with index and sub index.

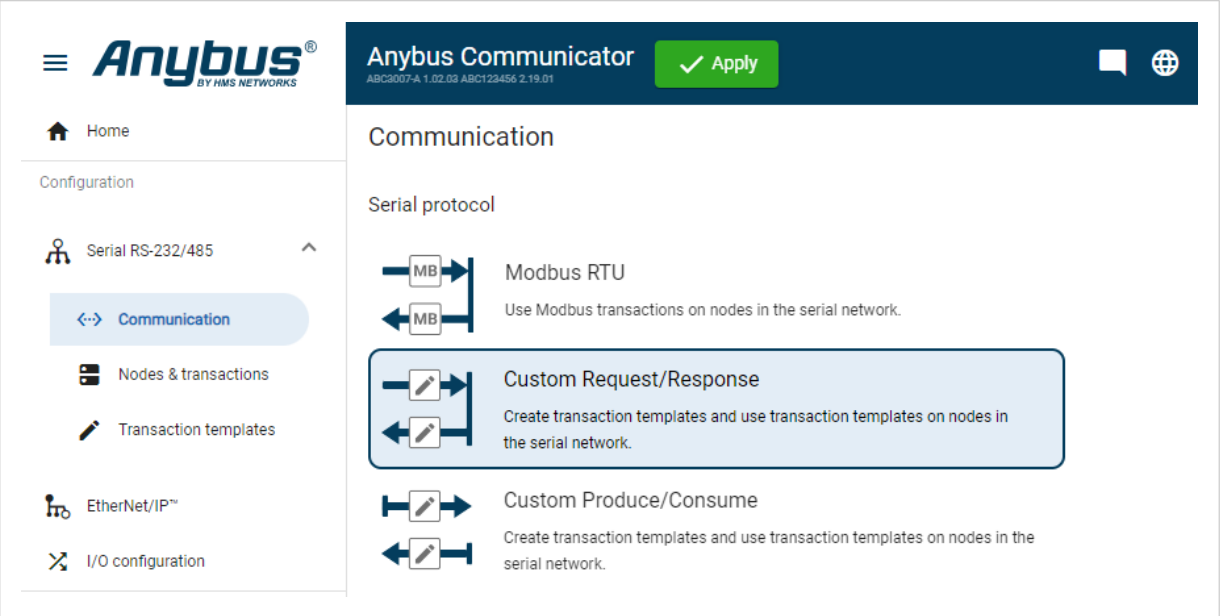


Figure 108. Communication page, **Custom Request/Response**

On the **Serial RS232/485** page, select **Custom Request/Response**.

11.2.4. Setup Serial Communication

Set up the communication between the Communicator and My Drive.

In the **Serial RS232/485** page, configure the **Communication** settings.

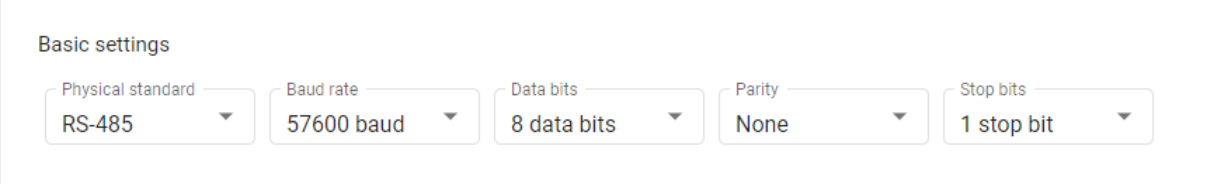


Figure 109. Serial RS232/485, Basic settings

Table 4. Used the following settings:

Frame objects	Value
Physical standard	RS-485
Baud rate	57600 baud
Data bits	8 bits
Parity	None
Stop bits	1 stop bit

11.2.5. Create Transaction Templates

All frames are verified using a CRC-16-IBM checksum.

My Drive is using a request/response protocol to access parameters addressed with index and sub index.

Map up control word, speed from EtherCAT to My Drive and status word and actual speed from the drive to EtherCAT.

Create Read Parameter (0x01)

The Communicator reads values delivered from to the My Drive node on to the PLC.

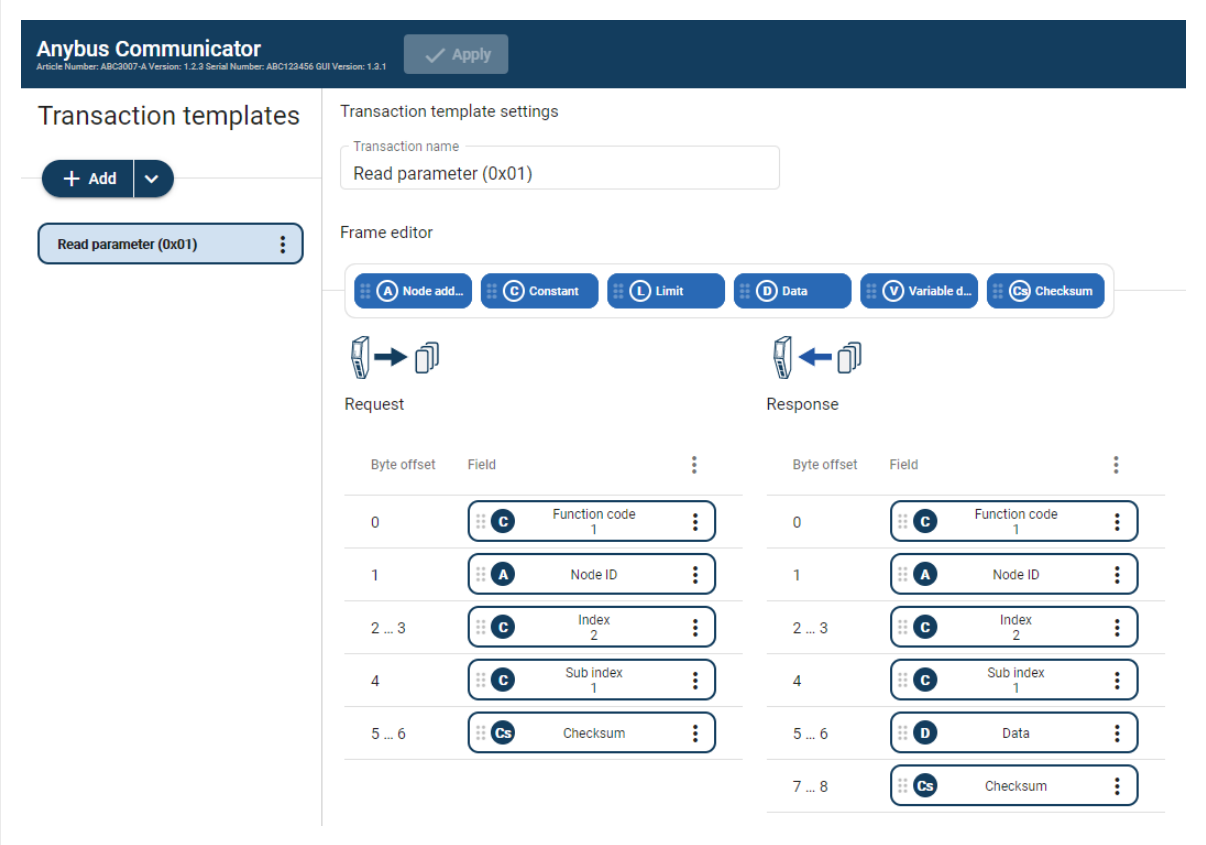


Figure 110. Read Parameter (0x01)

1. Add an **Empty template** and select it.
2. Name the template **Read parameter (0x01)**.
3. In the Frame editor **Request** area, add five **frame objects** with the following settings:

Table 5. Request frame objects

Frame objects	Name	Bytes/Length	Type/Checksum type	Endianess	Fixed field	Value
Constant	Function code	1	Byte	N/A	Yes	N/A
Node address	Node ID	1	Byte	N/A	N/A	N/A
Constant	Index	2	Word (two bytes)	Big-endian	No	Min 0 Max 1000
Constant	Sub index	1	Byte	N/A	No	Min 0 Max 255
Checksum	Checksum	2	CRC	N/A	N/A	N/A

4. In the Frame editor **Response** area, add six **frame objects** with the following settings:

Table 6. Response frame objects

Frame object	Name	Bytes/Length	Type/Checksum type	Endianess	Fixed field	Value
Constant	Function code	1	Byte	N/A	Yes	N/A
Node address	Node ID	1	Byte	N/A	N/A	N/A
Constant	Index	2	Word (two bytes)	Big-endian	No	Min 0 Max 1000
Constant	Sub index	1	Byte	N/A	No	Min 0 Max 255
Data	Data	2	Byte	N/A	Yes	N/A
Checksum	Checksum	2	CRC	N/A	N/A	N/A

Create Write Parameter (0x02)

The Communicator writes values delivered from the PLC to the My Drive node.

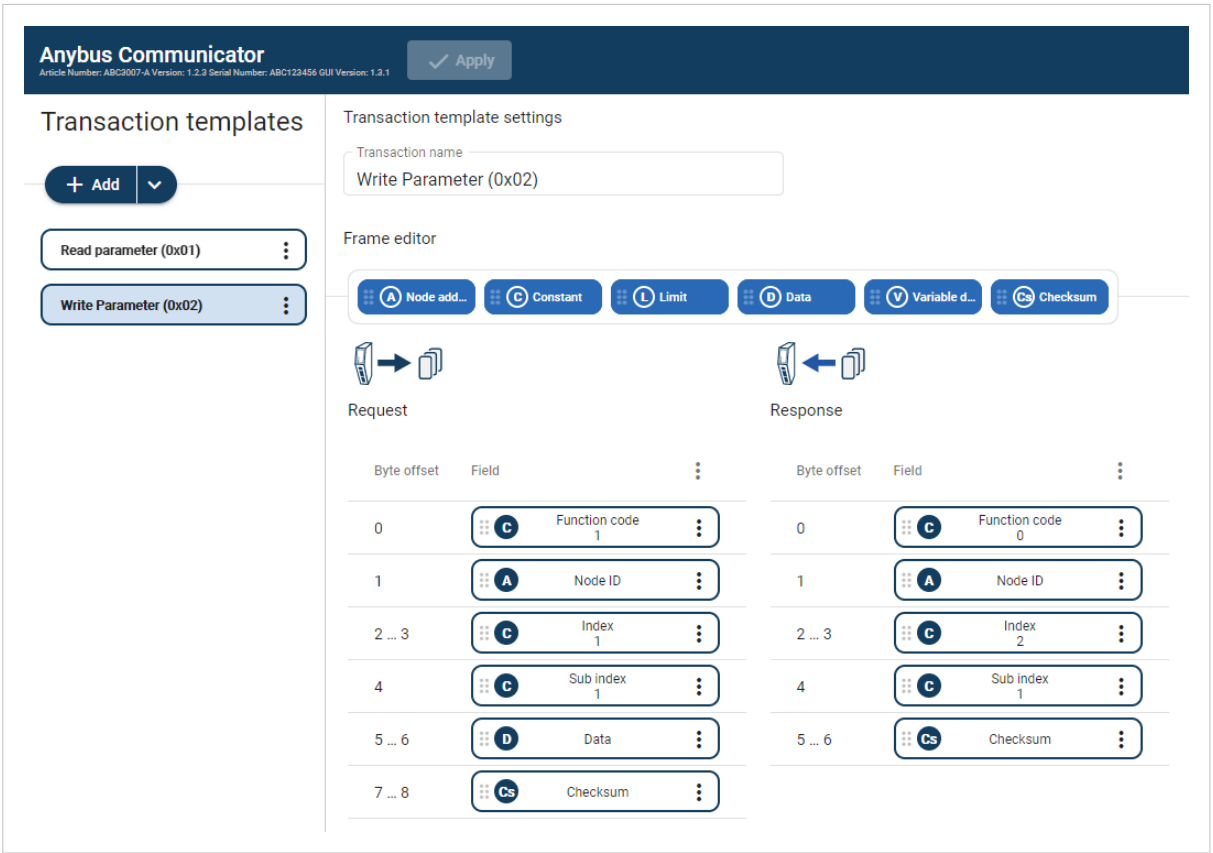


Figure 111. Write Parameter (0x02)

1. Add an **Empty template** and select it.
2. Name the template **Write parameter (0x02)**.
3. In the Frame editor **Request** area, add six **frame objects** with the following settings:

Table 7. Request frame objects

Frame object	Name	Bytes/Length	Type/Checksum type	Endianess	Fixed field	Value (Hex)
Constant	Function code	1	Byte	N/A	Yes	N/A
Node address	Node ID	1	Byte	N/A	N/A	N/A
Constant	Index	2	Word (two bytes)	Big-endian	No	Min 0 Max 1000
Constant	Sub index	1	Byte	N/A	No	Min 0 Max 255
Data	Data	2	Byte	N/A	Yes	N/A
Checksum	Checksum	2	CRC	N/A	N/A	N/A

4. In the Frame editor **Response** area, add five **frame objects** with the following settings:

Table 8. Response frame objects

Frame objects	Name	Bytes	Type/Checksum type	Endianess	Fixed field	Value (Hex)
Constant	Function code	1	Byte	N/A	Yes	N/A
Node address	Node ID	1	Byte	N/A	N/A	N/A
Constant	Index	2	Word (two bytes)	N/A	No	Min 0 Max 1000
Constant	Sub index	1	Byte	N/A	No	Min 0 Max 255
Checksum	Checksum	2	CRC	N/A	N/A	N/A

11.2.6. Setup Node and Transactions

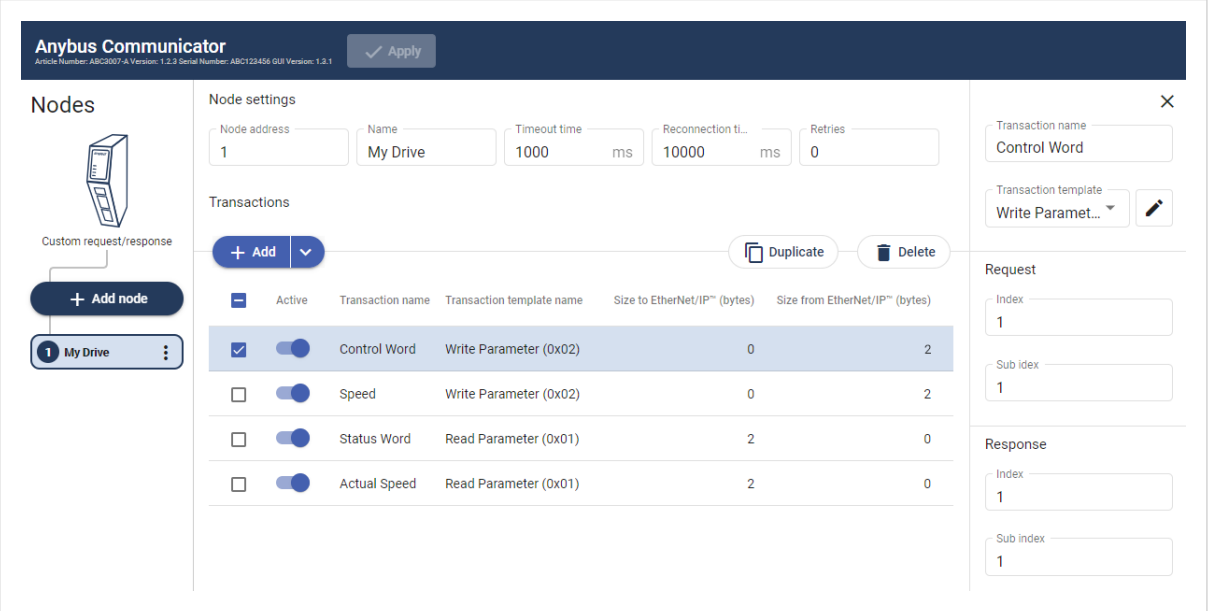


Figure 112. My Drive node with transactions

1. Add a node and select it.
2. In Node settings configure the node with the following settings:

Node settings	Value
Node address	1 My Drive is set up as a node with Node address 1.
Name	My Drive
Timeout time	1000 ms (default)
Reconnecting time	1000 ms (default)
Retries	0 (default)

3. Add four transactions to the My Drive node and configure them with the following settings:

Table 9. My Drive contains the following parameters

Transaction name	Transaction template	Index	Sub index
Control Word	Write Parameter (0x02)	1	1
Speed	Write Parameter (0x02)	1	2
Status Word	Read Parameter (0x01)	2	1
Actual Speed	Read Parameter (0x01)	2	2

11.2.7. Check the I/O Configuration

The control word, speed from EtherCAT to My Drive and status word and actual speed from My Drive to EtherCAT are mapped as follows in the **I/O configuration** page.

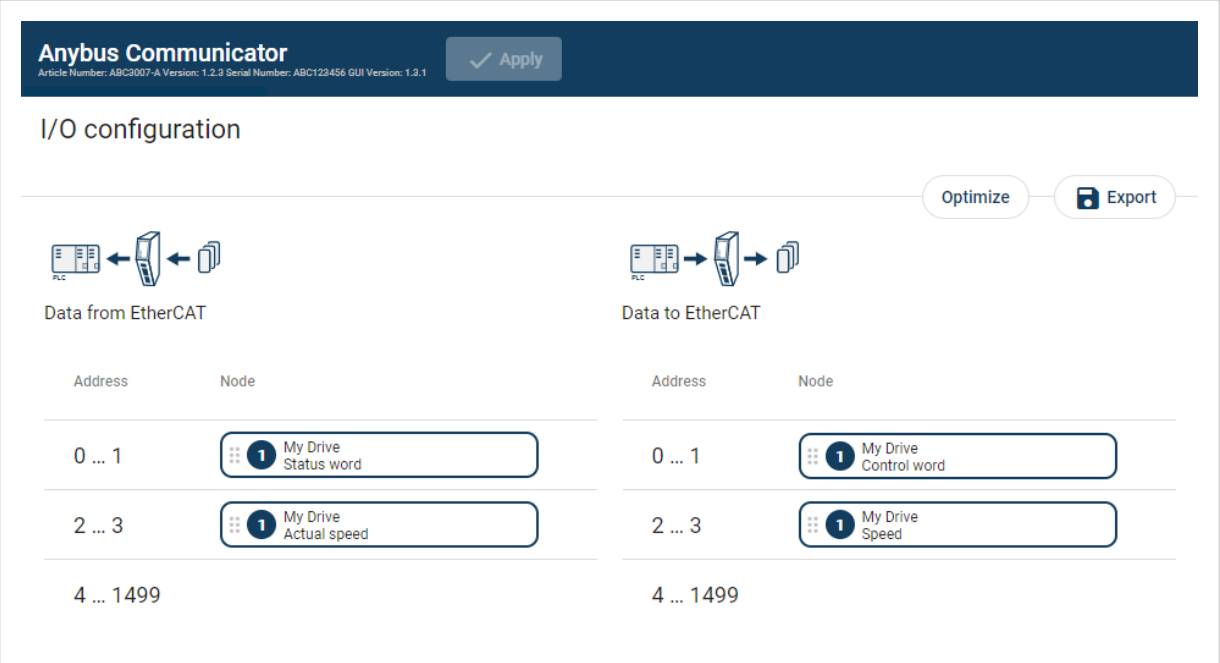


Figure 113. I/O configuration page

Table 10. Control word and speed from EtherCAT to My Drive

Address	Drive Parameter
0-1	Control Word
2-3	Speed

Table 11. Status word and actual speed from My Drive to EtherCAT

Address	Drive Parameter
0-1	Control Word
2-3	Speed

11.3. Barcode Scanner - Custom Produce/Consume Use Case

11.3.1. About the Use Case

The purpose of this use case is to explain how to use the **Custom Produce/Consume** serial protocol.

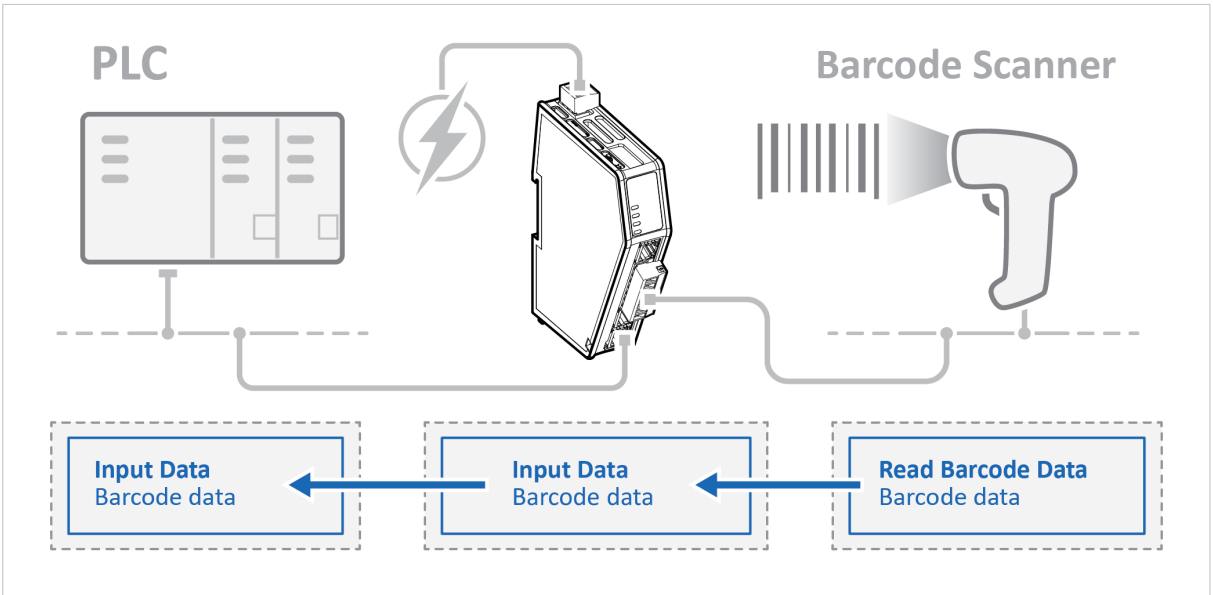


Figure 114. Barcode Scanner - Custom Produce/Consume Use Case

In this use case we use the Communicator to enable data exchange between an Barcode Scanner and a PLC.

The use case describes how to map the communication in the Communicator.

The Barcode Scanner is connected to the serial subnetwork via a custom RS-232 protocol.

The PLC is connected to an EtherCAT network (high level network).

We use the Custom Produce/Consume serial protocol and create a customized transaction template.

11.3.2. Before You Begin

- Connect the Communicator configuration port to your computer.
- Power on the Communicator.
- Ensure that your computer can find the Communicator IP address.
- Enter the Communicator built-in web interface of the.

For more information refer to Communicator Configuration (page 37).

11.3.3. Choose Serial Protocol Type

The purpose of this use case is to explain how to use the **Custom Produce/Consume** serial protocol.

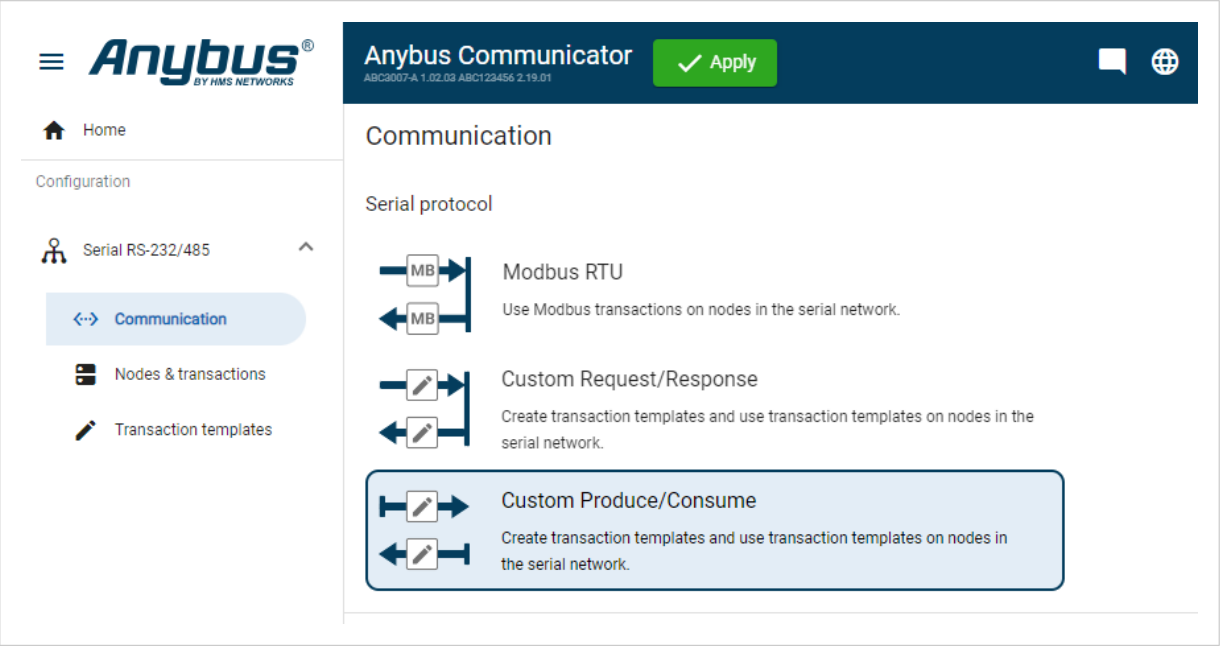


Figure 115. Communication page, **Custom Produce/Consume**

On the **Serial RS232/485** page, select **Custom Produce/Consume**.

11.3.4. Setup Serial Communication

Set up the communication between the Communicator and Barcode Scanner.

In the **Serial RS232/485** page, configure the **Communication** settings.

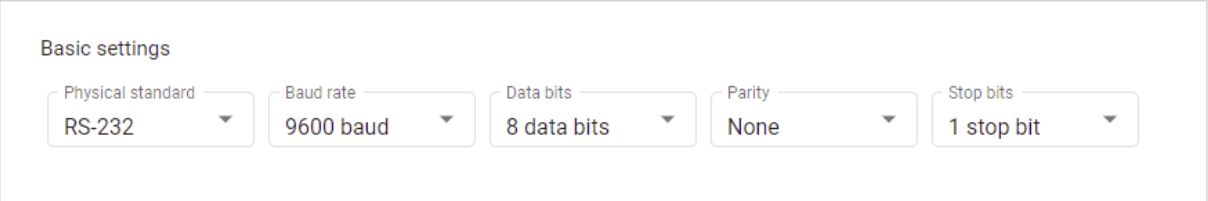


Figure 116. Serial RS232/485, Basic settings

Table 12. Used the following settings:

Frame objects	Value
Physical standard	RS-232
Baud rate	9600 baud
Data bits	8 bits
Parity	None
Stop bits	1 stop bit

11.3.5. Create Transaction Templates

Create Read Barcode Data Parameter

Before You Begin

The Communicator reads values delivered from to the Barcode Scanner node on to the PLC.

The Barcode Scanner sends data whenever it is available, without any request or handshake from the Communicator.

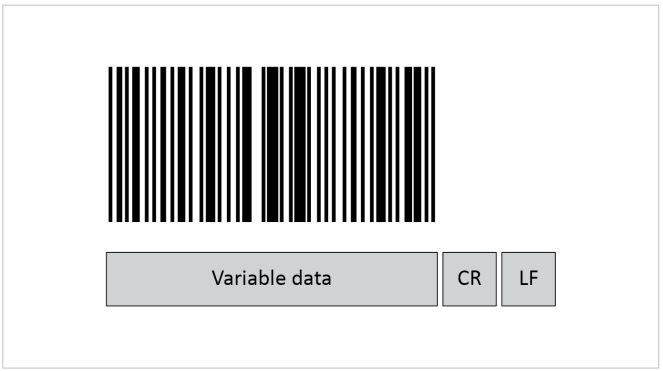


Figure 117. Barcode Variable data, CR and LF

In this example we have added three frame fields for the barcode data transaction:

- One Variable data frame for the length of the barcode.
  We use a fixed value.
  The maximum payload length is 31 (ASCII).
- The Barcode Scanner is configured to append:
  - One Carriage Return character (CR) to the barcode.
    So we create one Constant frame with the Value 13 (ASCII).
  - One Line Feed character (LF) to the barcode.
    So we create one Constant frame with the Value 10 (ASCII).

Procedure

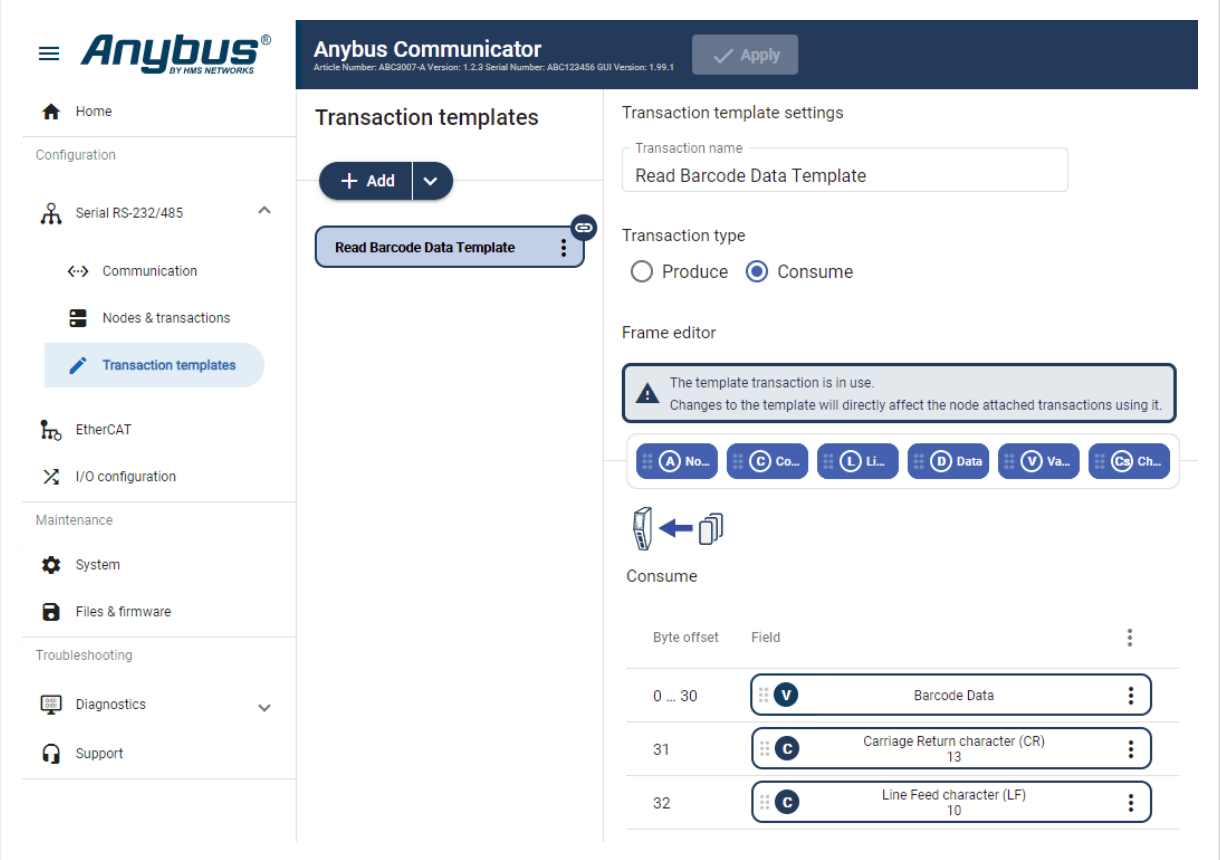


Figure 118. Read Barcode Data Parameter

1. Add an **Empty consume template** and select it.
2. Name the template **Read Barcode Data**.
3. In the **Frame editor**, add four frame field with the following settings:

Table 13. Consume frame fields

Frame fields	Name	Type	Value	Fixed field	Maximum payload length	Process data delimiter	Subnet delimiter	End pattern
Variable data	Barcode Data	N/A	N/A	Yes, set here	31 bytes	None	None	0
Constant	Carriage Return character (CR)	Byte (1 byte)	13	Yes, set here	N/A	N/A	N/A	N/A
Constant	Line Feed character (LF)	Byte (1 byte)	10	Yes, set here	N/A	N/A	N/A	N/A

11.3.6. Setup Node and Transactions

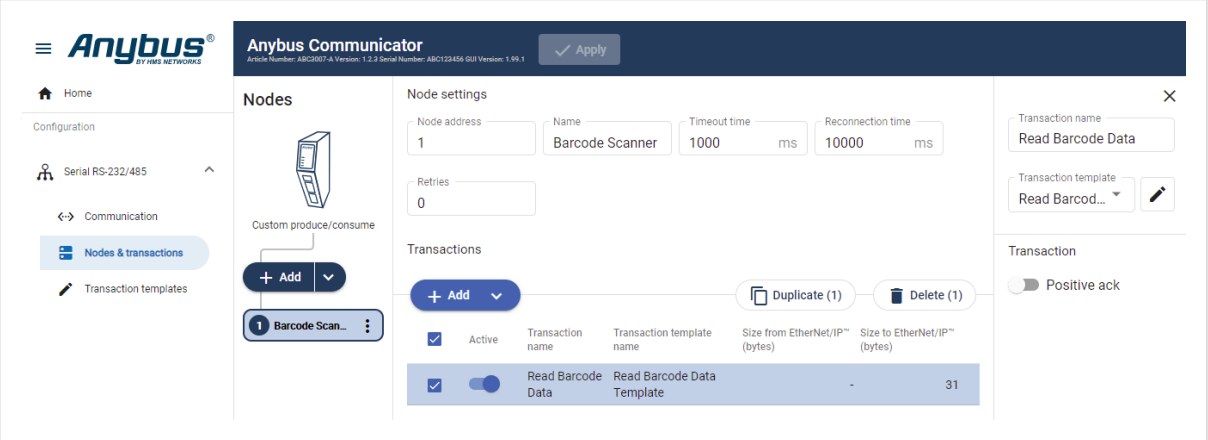


Figure 119. Node settings

1. Add a node and select it.
2. In **Node settings** configure the node with the following settings:

Node settings	Value
Node address	The Barcode Scanner is set up as a node with Node address 1.
Name	Barcode Scanner
Timeout time	1000 ms (default)
Reconnecting time	1000 ms (default)
Retries	0 (default)

3. Add one transactions to the Barcode Scanner node and configure it with the following settings:

Table 14. The Barcode Scanner contains the following parameters

Transaction name	Transaction template
Read Barcode Data	Read Barcode Data Template

11.3.7. Check the I/O Configuration

The transactions from the Barcode Scanner is mapped as follows in the **I/O configuration** page.

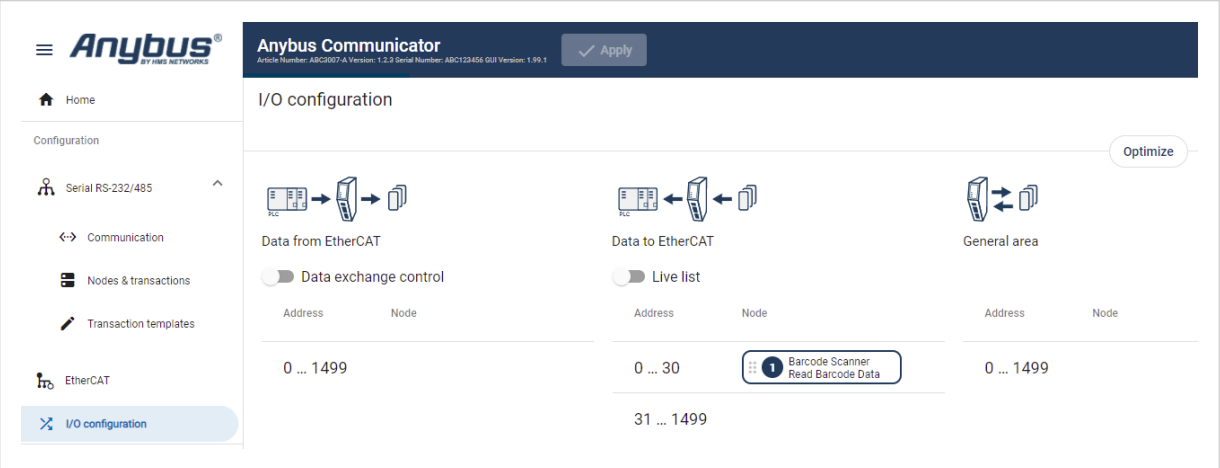


Figure 120. I/O configuration page

Table 15. Status word and actual speed from My Drive to EtherCAT

Address	Barcode Scanner Parameter
0-31	The variable data, 31 bytes, are forwarded from the Barcode Scanner to the PLC.

12. Maintenance

12.1. Action on Fatal Error

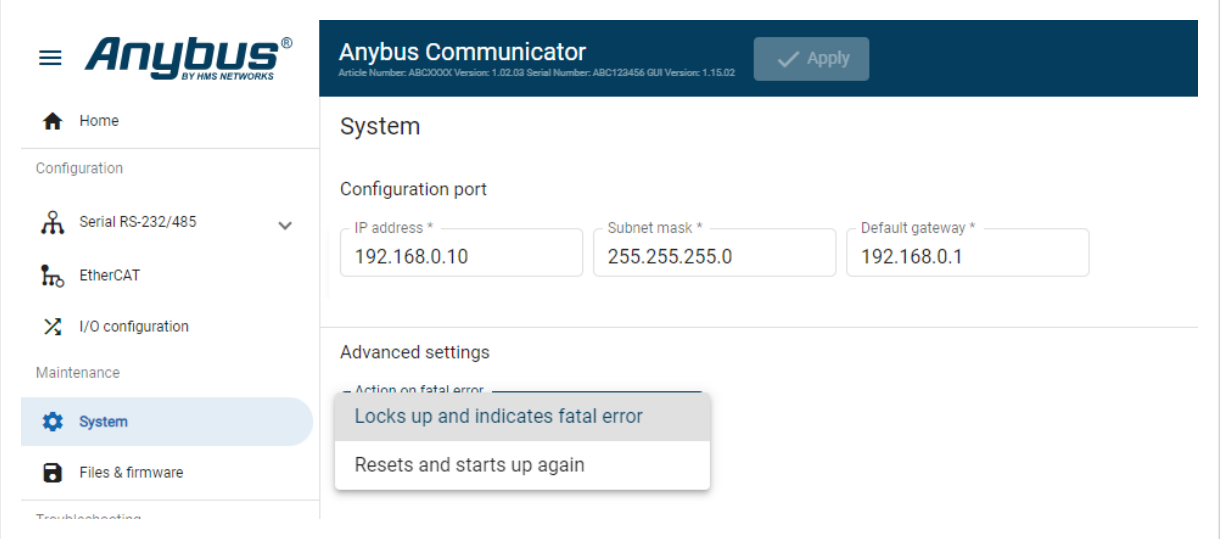


Figure 121. System page, Action on fatal error menu

A fatal error causes the Communicator firmware application to crash in an uncontrolled manner.

You can configure how the Communicator should behave if a fatal error occurs.

In the **Action on fatal error** menu, select one of the following settings:

- **Locks up and indicates fatal error**: Default setting, the Communicator locks up and the LED indicators indicates a fatal error.
- **Resets and starts up again**: The Communicator is rebooted to reset the system and return to normal operation.

12.2. Configuration Port IP Settings

On the **System** page you can change the IP address of the Communicator configuration port.

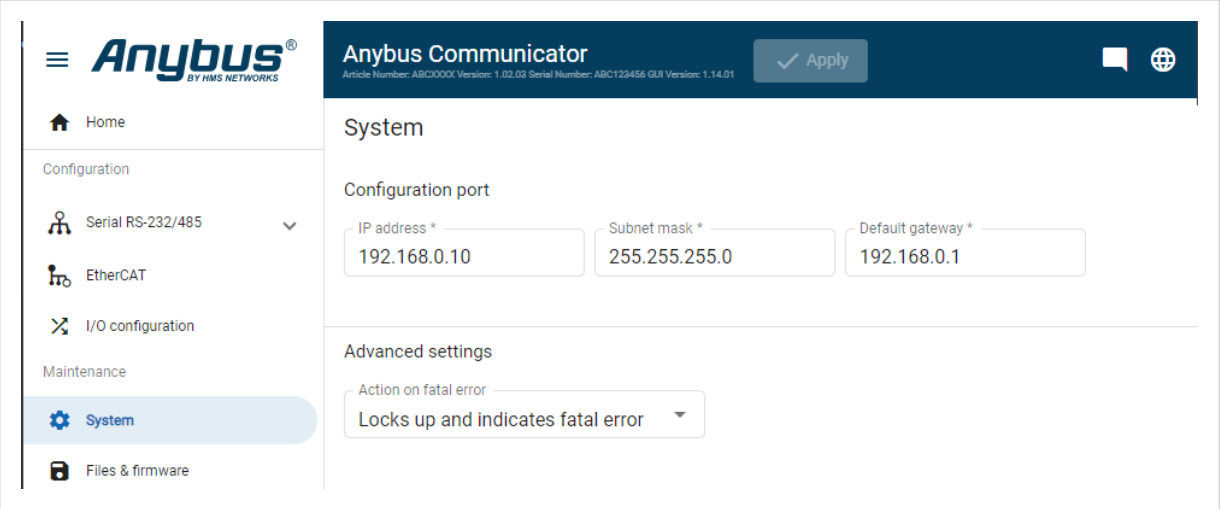


Figure 122. System page, Configuration port settings

Default Configuration Port IP settings

Setting	Default value
IP address	192.168.0.10
Subnet mask	255.255.255.0
Gateway	There is no default Gateway address.

12.3. Configuration File Handling

12.3.1. Export Configuration

You can export the current configuration, to import and use the same settings to configure additional Communicator.

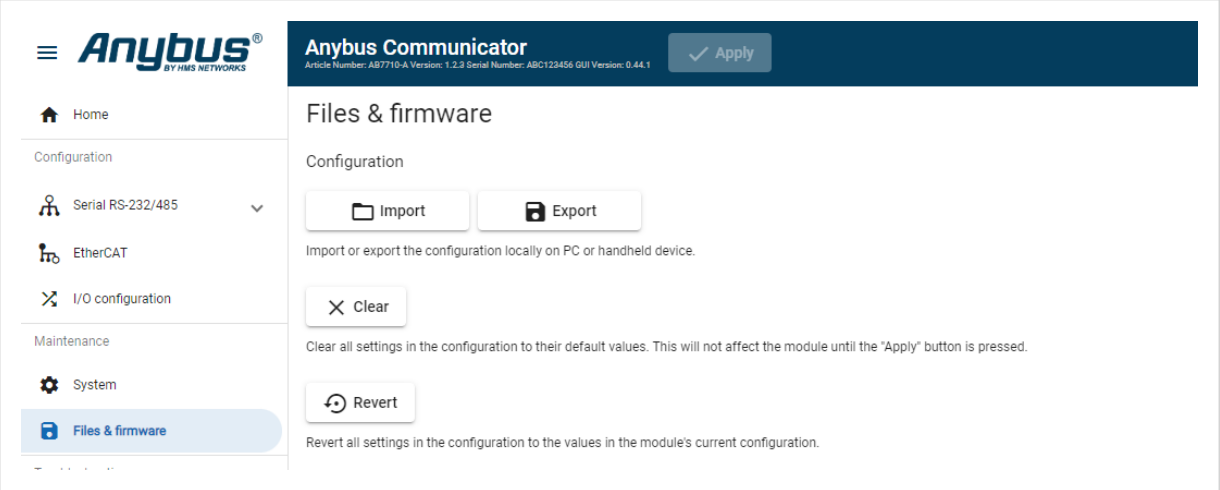


Figure 123. Files & firmware page

To export a configuration file:

In **Files & firmware**, click **Export**.

The configuration settings are stored in a .conf file and downloaded to your PC.

12.3.2. Import Configuration

To easily configure multiple Communicator with the same settings, you can import a configuration file.

Before You Begin

NOTE
Importing a configuration replaces the current applied configuration.

The supported file format is .conf.

Procedure

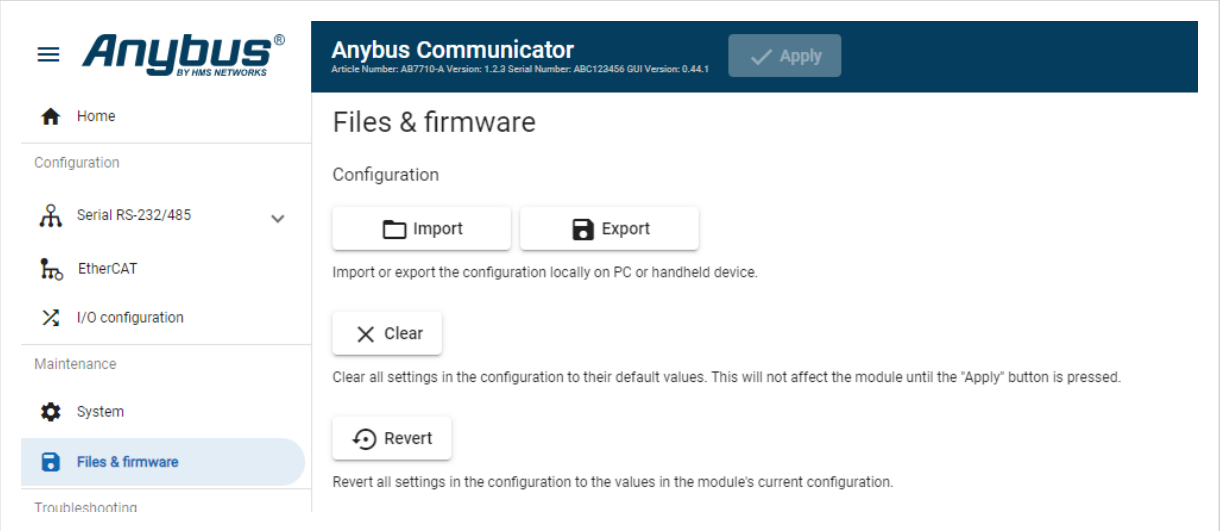


Figure 124. Files & firmware page

Import configuration file:

1. On the **Files & firmware** page, click **Import**.
2. In the Import configuration window, click **Select file (.conf)**.
3. In the Open dialog box, browse to and select the configuration file and click **Open**.
4. In the Import configuration window, click **Import**.
5. In the Communicator address settings window:
   - To import IP settings from the selected configuration file, click **Imported settings**.
     All configuration settings are imported.
   - To continue using the current IP settings, click **Configured settings**.
     All configuration settings except the IP settings are imported.
6. The configuration file is parsed.
   - If the configuration is compatible, the settings are imported.
   - If any compatibility mismatches occur, a message about the mismatch appears.
7. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

12.4. Clear and Revert Configuration

You can restore all settings in a configuration to the default settings.

Procedure

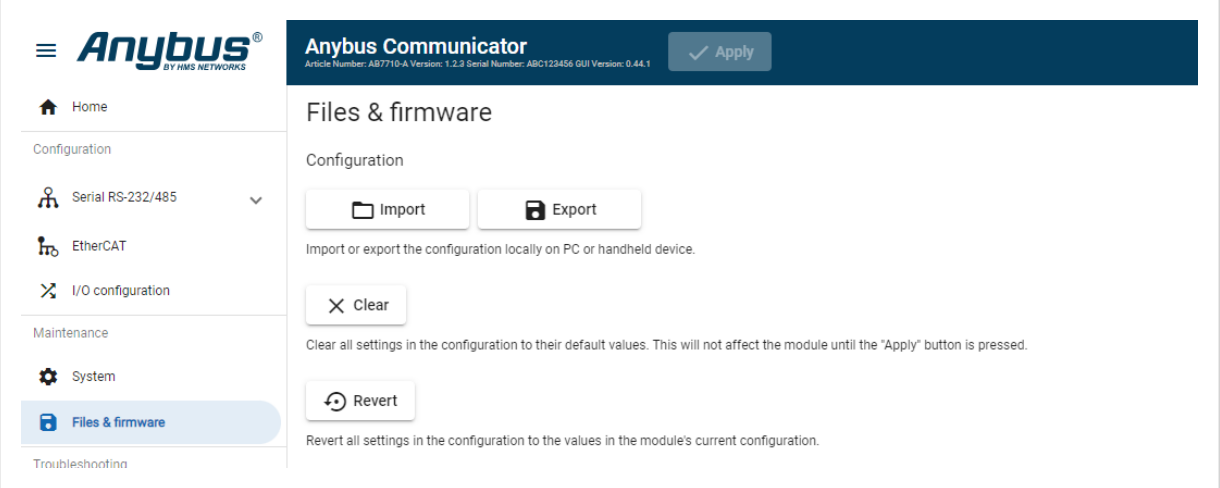


Figure 125. Files & firmware page

To Clear the Configuration

When you want to clear a configuration and return to the default settings.

1. On the **Files & firmware** page, click **Clear**.
2. In the Confirm clear window, click **Clear**.
3. To apply the change, click **Apply** in the web-interface header, and follow the instructions.

To Revert the Configuration

When you want to remove any configuration made in a current session and re-load the configuration from the gateway.

1. On the Files & firmware page, click **Revert**.
2. In the Confirm revert window, click **Revert**.
3. To apply the change, click **Apply** in the web-interface header, and follow the instructions.

12.5. Firmware Management

12.5.1. View the Firmware Version

On the **Support** page, you can view the current applied firmware version.

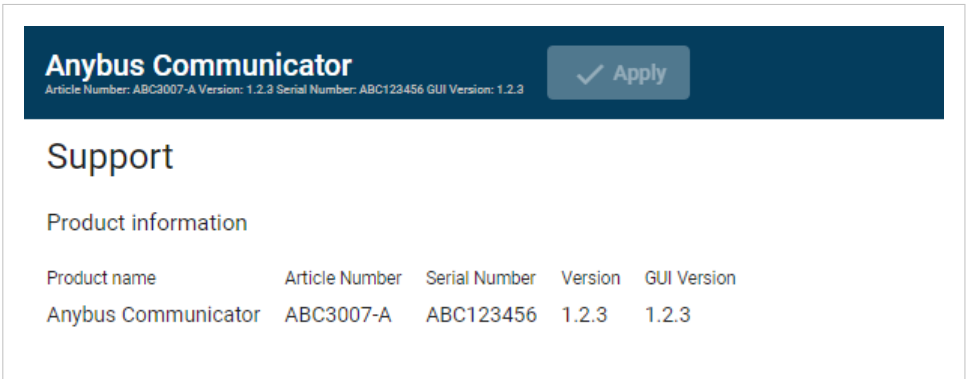


Figure 126. Support page, Product information example

12.5.2. Firmware and Configuration Compatibility

Compatibility after firmware upgrade

Current configuration is still compatible after upgrading the firmware.

Compatibility after firmware downgrade

IMPORTANT

Compatibility after a firmware downgrade cannot be guaranteed.

The current configuration may use features not available in the older firmware version.

12.5.3. Firmware File Validation

Before the firmware file is imported into the system, the firmware upgrade function performs a validation of the file, to ensure that:

- the firmware is compatible with the Communicator hardware
- the firmware is suited for the product
- the officially HMS software signatures are valid
- that the firmware file is not corrupt or damaged

If the firmware file does not pass the validation, the firmware file is rejected and an error message appear.

12.5.4. Update Firmware

Before You Begin

IMPORTANT
To eliminate the risk of interference with plant operation, firmware update is only available when the Communicator is disconnected from the OT networks.

Ensure to disconnect the Communicator from the OT networks.

Procedure

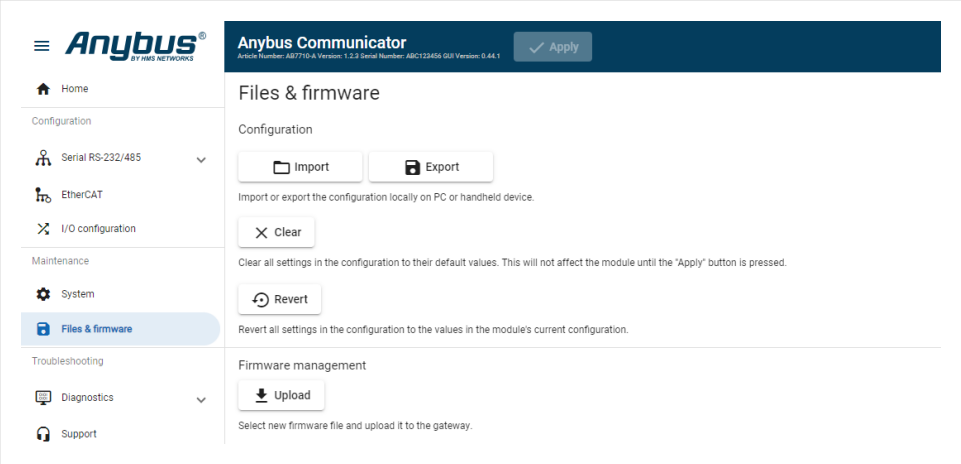


Figure 127. Files & firmware page

To update the firmware:

1. On the **Files & firmware** page, click **Upload**.
2. In the Upload Firmware window, click **Select firmware (.hiff)**.
3. In the Open dialog box, browse to and select the firmware file and click **Open**.
4. To start the firmware upgrade, click **Update firmware**.
   The firmware file is validated and transferred.

Result

- If the firmware file passes the validation: The firmware is upgraded and then the Communicator automatically reboots, for the upgrade to take effect.
- If the firmware file is rejected: An error message appears.

12.6. Change Language

Default language is **English**.

To change the language of the Communicator built-in web interface:

1. In the Communicator built-in web-interface header, click the **Language** icon .

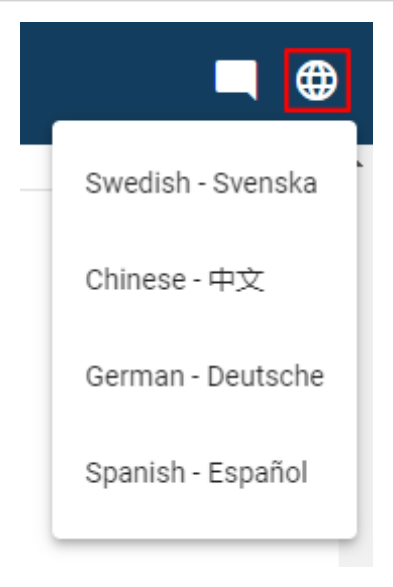


Figure 128. Language menu

2. Select a new language from the list.

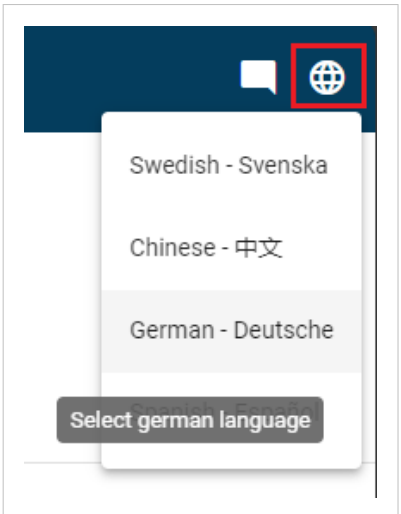


Figure 129. Example: Change language to German

The language change takes effect immediately.

13. Troubleshooting

13.1. Diagnostics

13.1.1. Serial RS-232/485 Data Monitor

On the Serial RS-232/485 page you can monitor how the data flow between the nodes and the gateway changes over time.

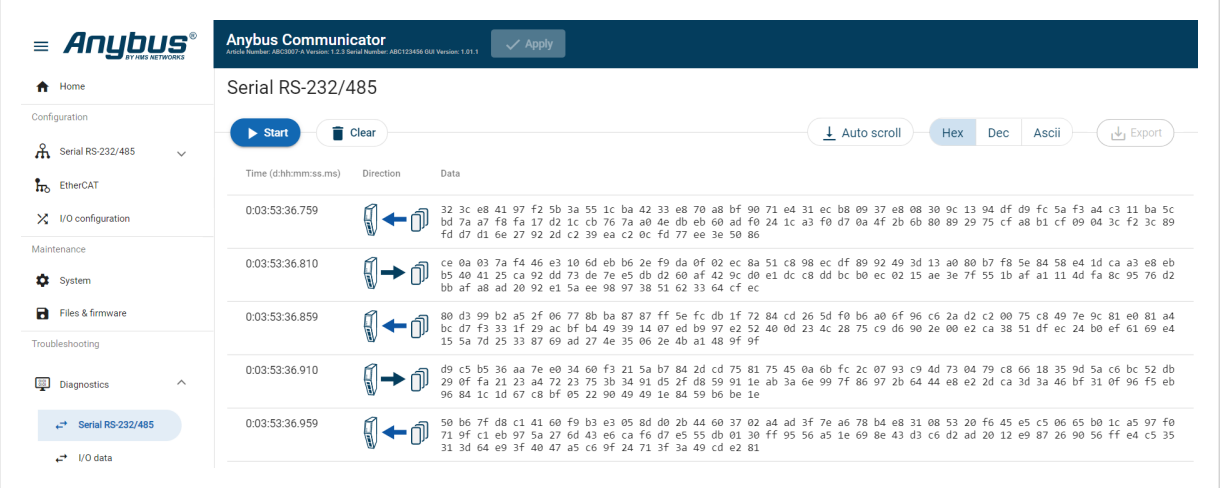


Figure 130. Serial RS-232/485 page

The table can contain at most 10000 messages. When the limit is reached, the oldest messages are discarded when new messages are added.

Choose How Data is Displayed

To choose if the data should be displayed in Hexadecimal, Decimal or ASCII, click **Hex**, **Dec** or **Ascii**.

Start and Stop Data Flow

To start the data flow, click **Start**.

To end the data flow, click **Stop**.

Export Data Flow

To export the data flow, click **Export**.

An Excel file with the data flow is downloaded to your PC.

13.1.2. I/O Data

On the **Diagnostics**, **I/O data** page you can monitor how the data flow between the **Serial RS-232/485** side and the **EtherCAT** side, including any configured endian conversions.

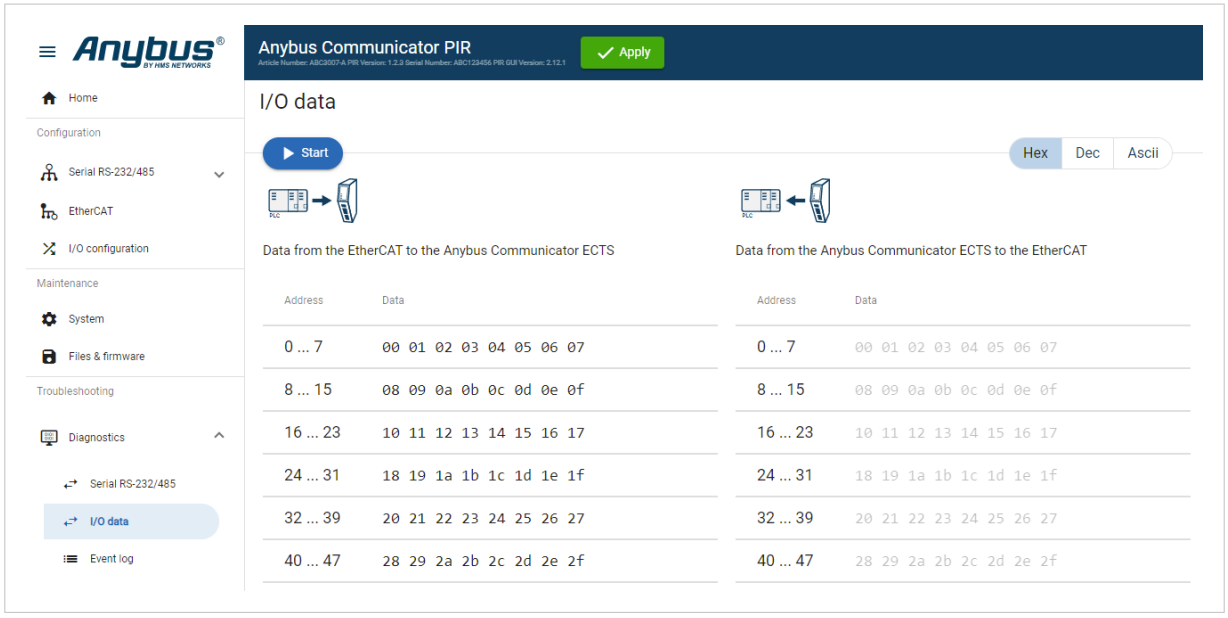


Figure 131. I/O data

I/O data is updated twice every second.

Select how data is displayed

To choose if the data should be displayed in Hexadecimal, Decimal or ASCII, click **Hex**, **Dec** or **Ascii**.

Start and Stop Data flow

- To start the data flow, click **Start**.
- To end the data flow, click **Stop**.

13.1.3. Event Log

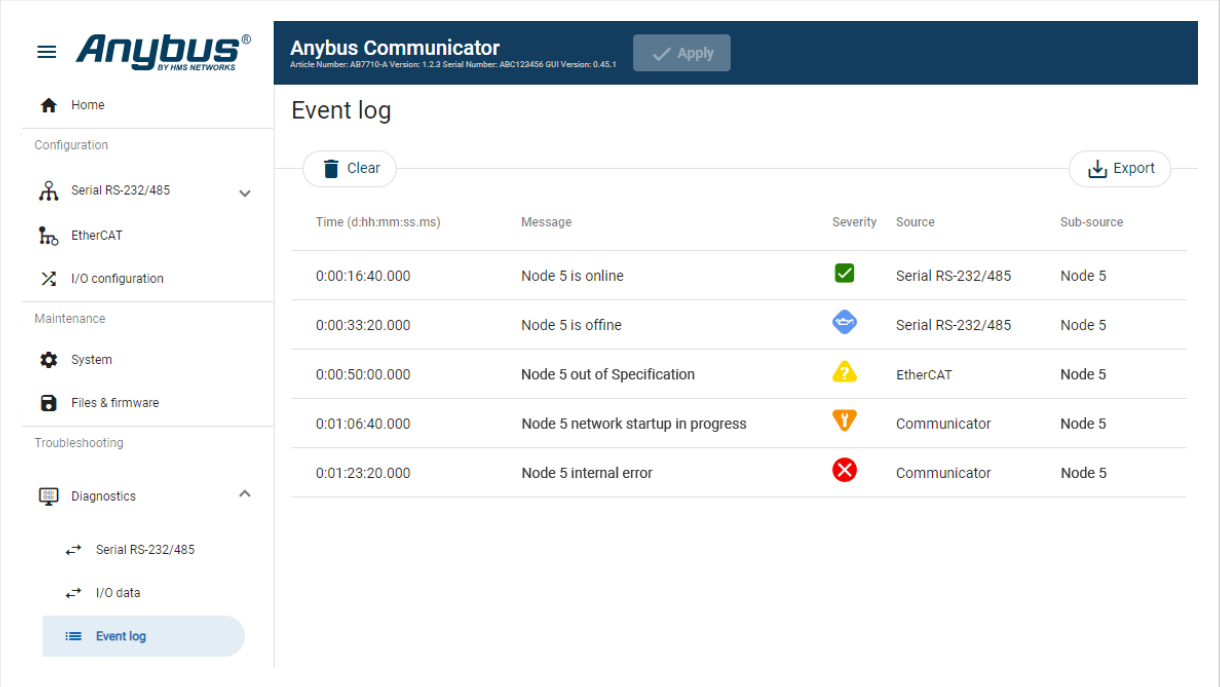


Figure 132. Event log page example

How To Analyze the Information

The log follows the FIFO principle, first in and first out. The oldest (first) value is processed first.

Time (d:hh:mm:ss.ms)	The date and time when the event occurred.	
Message	A brief description of the event.	
Severity	The severity of the event occurred. For description of the symbols, see Communicator Status Monitor (page 106).	
Source	0	Communicator
	1	High level network, EtherCAT
	2	Subnetwork, Serial RS-232/485
Sub-source	The nodes connected to the subnetwork and the PLC connected to the high level network. If there is a problem with a node the node name is displayed in the Sub-source column. Example 10. Sub-source number If the node name is 5, number 5 is displayed in the Sub-source column.	

To clear the current log, click **Clear**.

13.1.4. LED Status

On the Home page, you can remotely monitor the Communicator LED status.

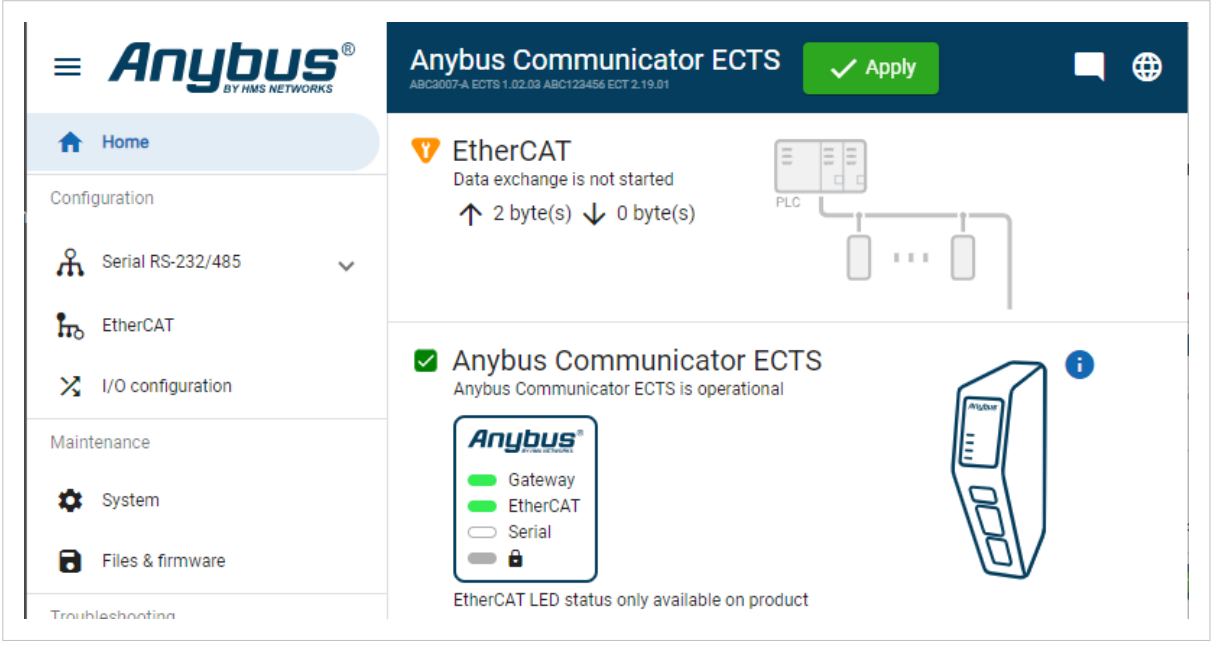


Figure 133. Home page

For information about the LED indication, see Communicator LED Indicators (page 108).

13.2. Reset to Factory Settings

Before You Begin

Factory reset will reset any on site made configuration changes and set the Communicator to the same state as leaving HMS production.

When the Firmware has been updated, factory reset will revert the Communicator configuration to initial state after the update.

Procedure

To reset the Communicator:

1. Disconnect the Communicator from power.

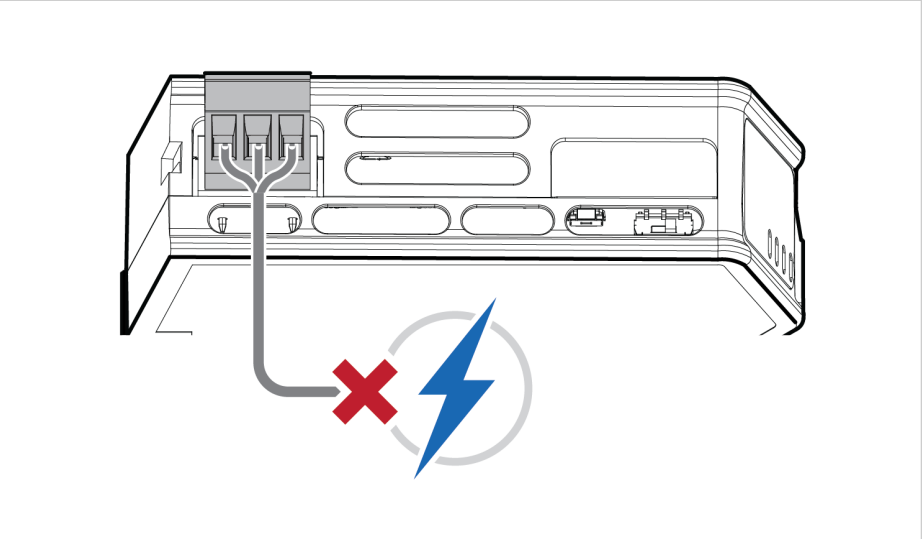


Figure 134. Disconnect power

2. Use a pointed object, such as a ballpoint pen to press and hold the **Reset** button.

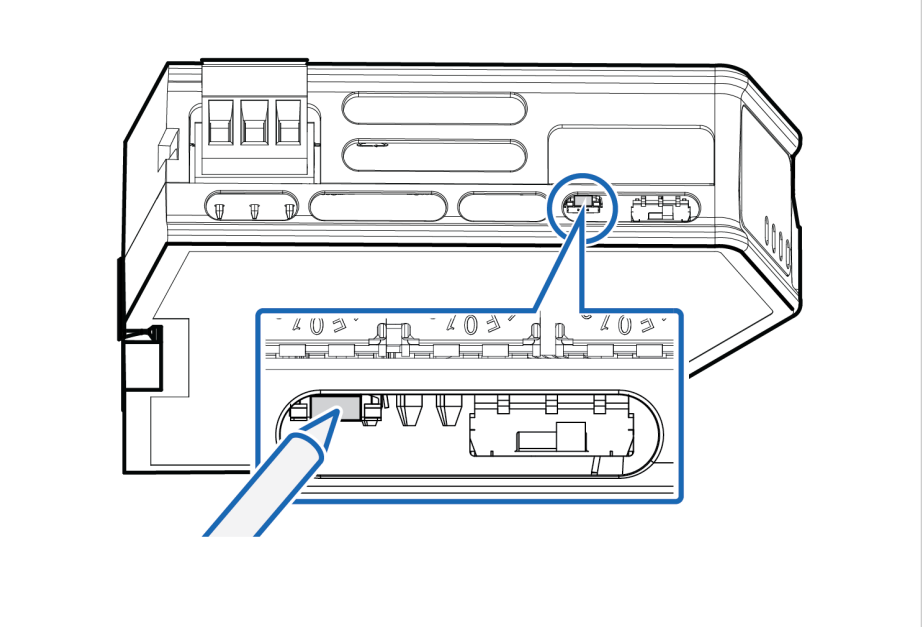


Figure 135. Press and hold **Reset** button

3. While holding the **reset** button, reconnect the Communicator to power.

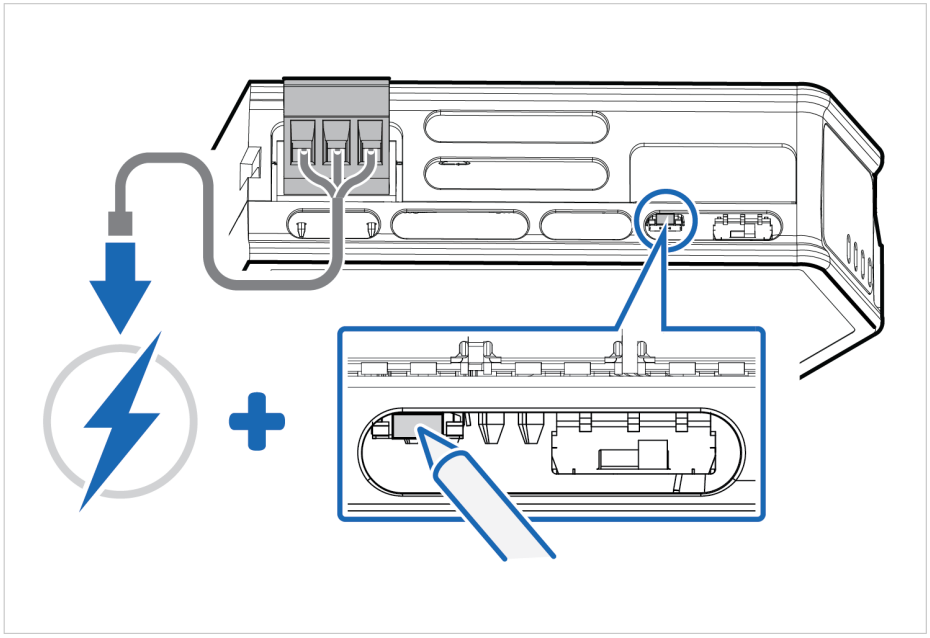


Figure 136. Hold **Reset** button and reconnect power

4. Release the **reset** button.
   The Communicator enters exception state.

5. Reboot the Communicator.

Result

When the Communicator has successfully rebooted, the Communicator configuration is reset to the factory default configuration or the current configuration after firmware upgrade.

To Do Next

To ensure that the Communicator built-in web-interface is synchronized.

1. Open the Communicator built-in web interface.
2. Navigate to the **Files & firmware** page and click **Revert**.

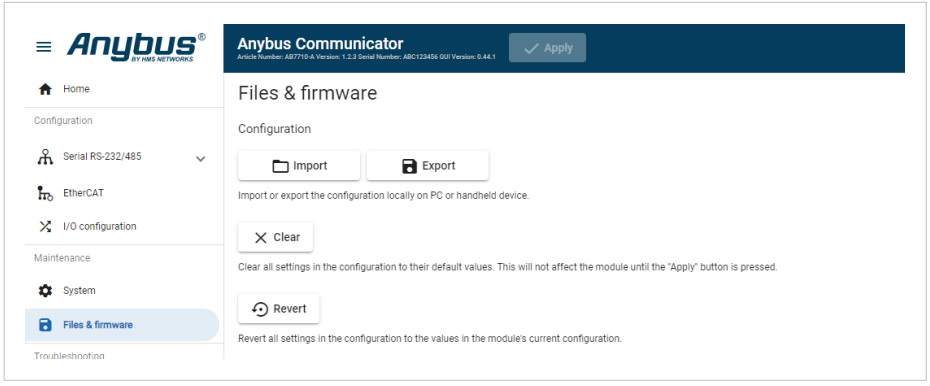


Figure 137. Files & firmware, Revert

13.3. Firmware Upgrade Error Management

Before You Begin

If the firmware update process is interrupted or if the power is lost during the update process, the Communicator goes into fallback mode.

The last working firmware is still available on the flash, but it is not active.

Procedure

To complete the interrupted firmware update:

1. Disconnect the Communicator from power.

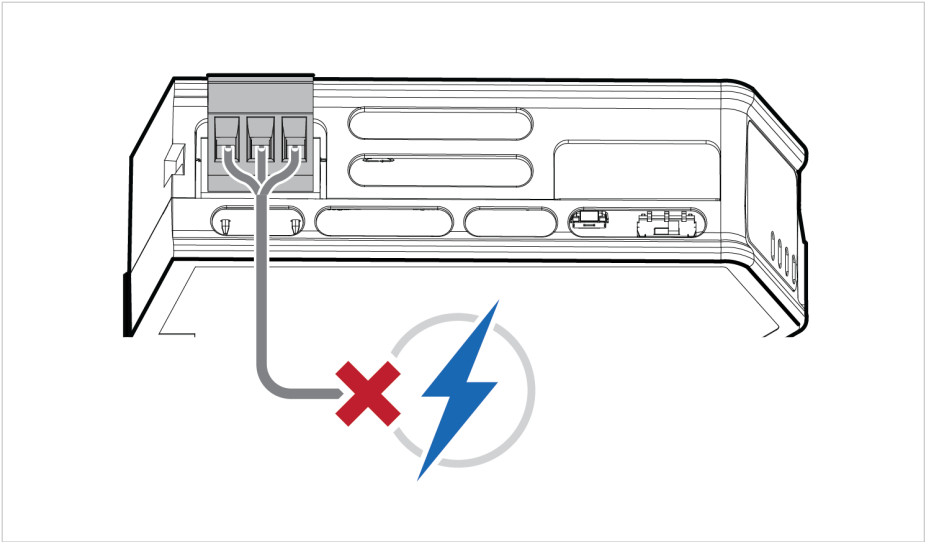


Figure 138. Disconnect power

2. Reconnect the Communicator to power.

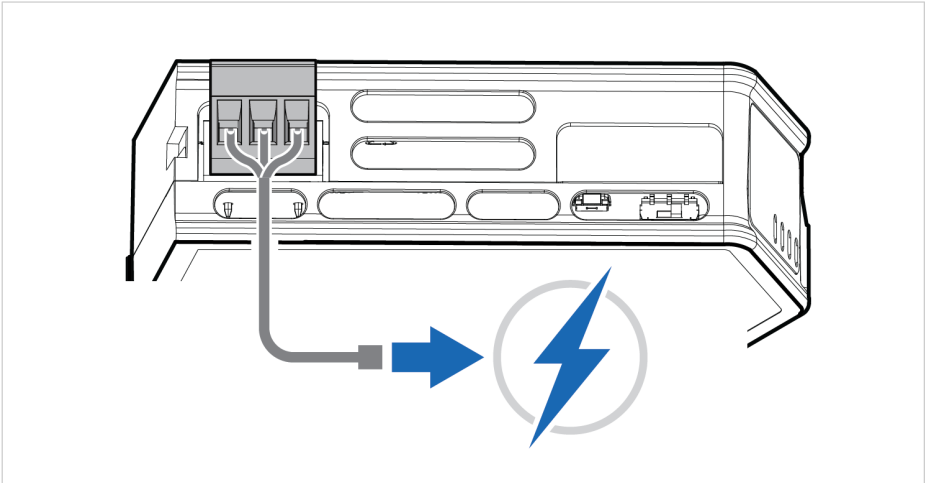


Figure 139. Reconnect power

3. Leave the Communicator for 10 minutes.
   The Gateway status led indicator flashes red and green until the firmware upgrade is completed.

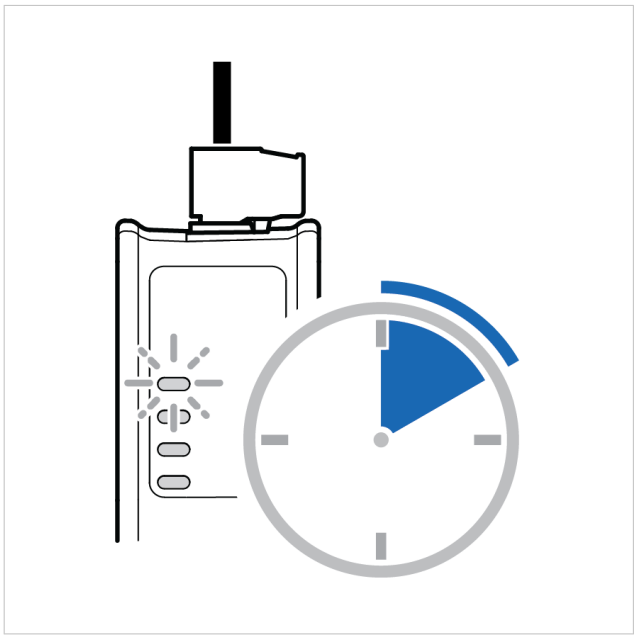


Figure 140. Firmware upgrade LED indication

Result

The Communicator recover and return to normal operation.

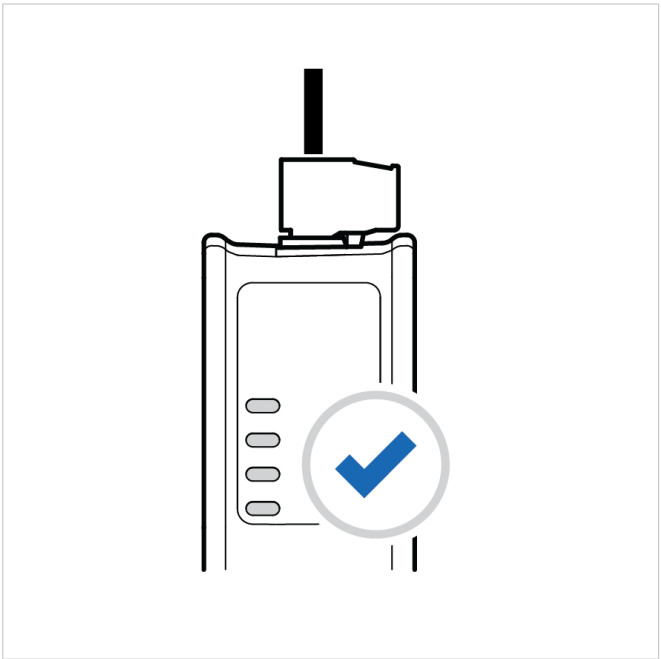


Figure 141. Recover and return to normal operation

To Do Next

To check LED status, refer to Communicator LED Indicators (page 108).

13.4. Support

13.4.1. Support Package

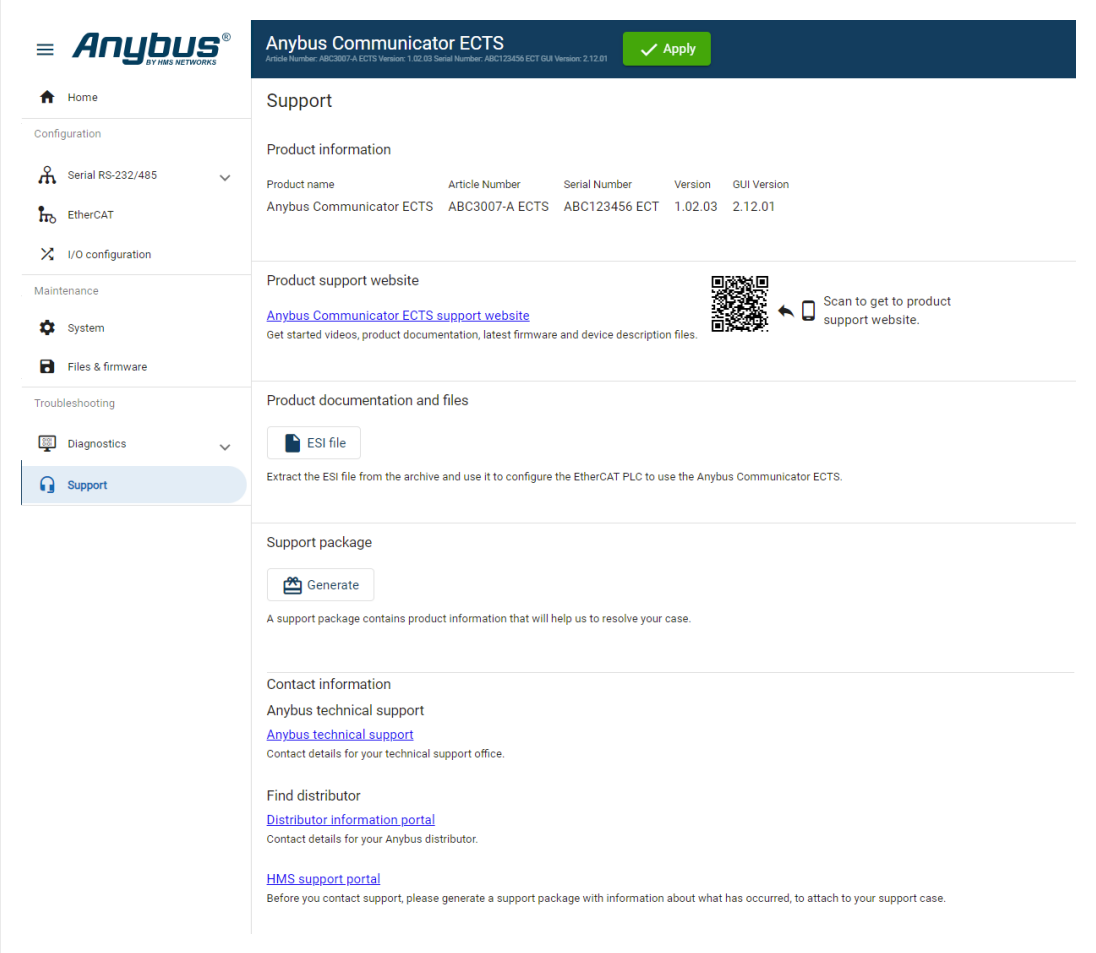


Figure 142. Support page example

Before you create a ticket for technical support, generate a support package.

The support package contains information about what has occurred and will help the Anybus technical support team resolve the support case as quickly and efficiently as possible.

Support Package Content
The information in the support package is available to open and read, the files are not locked or encrypted.

Generate Support Package
On the **Support** page, click **Generate**.

A zip file with the support files is downloaded to your PC.

Create a Support Ticket

1. On the **Anybus Technical Support** page, navigate to the **Support Center** page and click **HMS Support Portal**.
2. In the **HMS Support Portal**, create a support ticket and upload the support package.

14. Technical Data

For complete technical specifications and regulatory compliance information, please visit www.anybus.com.

14.1. Technical Specifications

Article identification	ABC3061
Configuration connector	RJ45
Communication connector	RJ45 x 2
Serial connector	7-pin screw connector
Power connector	3-pin screw connector
Power supply	12-30 VDC, Reverse voltage protection and short circuit protection
Power consumption	Typical: 90 mA @ 24 V (2.2 W) Max: 3 W
Storage temperature	-40 to +85 °C
Operating temperature	-25 to +70 °C
Humidity	EN 600068-2-78: Damp heat, +40°C, 93% humidity for 4 days EN 60068-2-30: Damp heat, +25°C – +55°C, 95% RH, 2 cycles
Vibration	See datasheet
Housing material	Plastic, See datasheet for details
Protection class	IP20
Product weight	150 g
Dimensions	27 x 144 x 98 mm (W x H x D) with connectors included
Mounting	DIN-rail

15. End Product Life Cycle

15.1. Secure Data Disposal

IMPORTANT
To avoid exposure of sensitive data, always perform a factory reset before decommissioning the equipment.

Factory reset will reset any on site made configuration changes and set the Communicator to the same state as leaving HMS production.

See Reset to Factory Settings (page 142).

16. Reference Guides

16.1. About Input Registers and Holding Registers

Modbus data is most often read and written as registers which are 16-bit pieces of data.

Holding registers and Input registers are both 16-bit registers.

Input Registers

Input registers can only be read.

Holding Registers

Holding registers can be read or written.

These registers can be used for a variety of things such as inputs, outputs, configuration data, or other requirement for holding data.

16.2. Modbus Data Model

Discretes Input	Single bit	Read-Only	Data can be provided by the I/O system.
Coils	Single bit	Read-Write	Data can be alterable by the application program.
Input Registers	16-bit word	Read-Only	Data can be provided by the I/O system
Holding Registers	16-bit word	Read-Write	Data can be alterable by the application program.

Reference: MODBUS Application Protocol Specification V1.1b3, April 26 2012

For more information refer to the Modbus organization website.

16.3. Modbus Transactions

Reference: MODBUS Application Protocol Specification V1.1b3, April 26 2012

For more information refer to the Modbus organization website.

Nr	Transaction	Function Code	Description
1	Read Coils	0x01	Read from 1 to 2000 contiguous status of coils in a remote device.
2	Read Discrete Inputs	0x02	Read from 1 to 2000 contiguous status of discrete inputs in a remote device.
3	Read Holding Registers	0x03	Read the contents of a contiguous block of holding registers in a remote device.
4	Read Input Registers	0x04	Read from 1 to 125 contiguous input registers in a remote device.
5	Write Single Coil	0x05	Write a single output to ON or OFF in a remote device.
6	Write Single Register	0x06	Write a single holding register in a remote device.
15	Write Multiple Coils	0x0F	In a sequence of coils, force each coil to either ON or OFF in a remote device.
16	Write Multiple Registers	0x10	Write a block of contiguous registers in a remote device.
22	Mask Write Register	0x16	In a single transaction, modify the contents of a specified holding register using a combination of an AND mask, an OR mask, and the register's current contents. Can be used to set or clear individual bits in the register.
23	Read/Write Multiple Registers	0x17	Performs a combination of one read operation and one write operation. The write operation is performed before the read.

16.4. Modus Exception Codes

Exception Code	Name	Description
01	Illegal Function	The server does not recognize or permit the function code.
02	Illegal Data Address	The data address (register, discrete input or coil number) is not an permitted address for the server. If multiple registers were requested, at least one was not permitted.

Reference: MODBUS Application Protocol Specification V1.1b3, April 26 2012

For more information refer to the Modbus organization website.

16.5. CANopen over EtherCAT (CoE) Objects

TIP
"XXXX" in the table Name column is the process data offset value.

Table 16. Product specific CANopen objects

Index	Obj.type	Subindex	Type	Access	PDO Mappable	Name	Value
0x2000	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0001	Send process data, byte 1
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 2 - 127
		128	UNSIGNED8	R	TxPDO	Input byte 0128	Send process data, byte 128
0x2001	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0129	Send process data, byte 129
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 130 - 255
		128	UNSIGNED8	R	TxPDO	Input byte 0256	Send process data, byte 256
0x2002	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0257	Send process data, byte 257
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 258 - 383
		128	UNSIGNED8	R	TxPDO	Input byte 0384	Send process data, byte 384
0x2003	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0385	Send process data, byte 385
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 386 - 511
		128	UNSIGNED8	R	TxPDO	Input byte 0512	Send process data, byte 512
0x2004	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0513	Send process data, byte 513
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 514 - 639
		128	UNSIGNED8	R	TxPDO	Input byte 0640	Send process data, byte 640
0x2005	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0641	Send process data, byte 641
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 642 - 767
		128	UNSIGNED8	R	TxPDO	Input byte 0768	Send process data, byte 768
0x2006	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0769	Send process data, byte 769
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 770 - 895
		128	UNSIGNED8	R	TxPDO	Input byte 0896	Send process data, byte 896
0x2007	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 0897	Send process data, byte 897

Index	Obj.type	Subindex	Type	Access	PDO Mappable	Name	Value
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 898 - 1023
		128	UNSIGNED8	R	TxPDO	Input byte 1024	Send process data, byte 1024
0x2008	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 1025	Send process data, byte 1025
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 1026 - 1151
		128	UNSIGNED8	R	TxPDO	Input byte 1152	Send process data, byte 1152
0x2009	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 1153	Send process data, byte 1153
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 1154 - 1279
		128	UNSIGNED8	R	TxPDO	Input byte 1280	Send process data, byte 1280
0x200A	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	TxPDO	Input byte 1281	Send process data, byte 1281
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 1282 - 1407
		128	UNSIGNED8	R	TxPDO	Input byte 1408	Send process data, byte 1408
0x200B	RECORD	N/A	UNSIGNED8	N/A	N/A	Inputs	N/A
		0	UNSIGNED8	R	No	Number of entries	78
		1	UNSIGNED8	R	TxPDO	Input byte 1409	Send process data, byte 1409
		…	UNSIGNED8	R	TxPDO	Input byte XXXX	Send process data, bytes 1410 - 1487
		78	UNSIGNED8	R	TxPDO	Input byte 1486	Send process data, byte 1486
0x2100	RECORD		UNSIGNED8			Outputs	
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0001	Receive process data, byte 1
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 2 - 127
		128	UNSIGNED8	R	RxPDO	Output byte 0128	Receive process data, byte 128
0x2101	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0129	Receive process data, byte 129
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 130 - 255
		128	UNSIGNED8	R	RxPDO	Output byte 0256	Receive process data, byte 256
0x2102	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0257	Receive process data, byte 257
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 258 - 383
		128	UNSIGNED8	R	RxPDO	Output byte 0384	Receive process data, byte 384
0x2103	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0385	Receive process data, byte 385
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 386 - 511
		128	UNSIGNED8	R	RxPDO	Output byte 0512	Receive process data, byte 512
0x2104	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A

Index	Obj.type	Subindex	Type	Access	PDO Mappable	Name	Value
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0513	Receive process data, byte 513
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 514 - 639
		128	UNSIGNED8	R	RxPDO	Output byte 0640	Receive process data, byte 640
0x2105	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0641	Receive process data, byte 641
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 642 - 767
		128	UNSIGNED8	R	RxPDO	Output byte 0768	Receive process data, byte 768
0x2106	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0769	Receive process data, byte 769
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 770 - 895
		128	UNSIGNED8	R	RxPDO	Output byte 0896	Receive process data, byte 896
0x2107	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 0897	Receive process data, byte 897
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 898 - 1023
		128	UNSIGNED8	R	RxPDO	Output byte 1024	Receive process data, byte 1024
0x2108	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 1025	Receive process data, byte 1025
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 1026 - 1151
		128	UNSIGNED8	R	RxPDO	Output byte 1152	Receive process data, byte 1152
0x2109	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 1153	Receive process data, byte 1153
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 1154 - 1279
		128	UNSIGNED8	R	RxPDO	Output byte 1280	Receive process data, byte 1280
0x210A	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	128
		1	UNSIGNED8	R	RxPDO	Output byte 1281	Receive process data, byte 1281
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 1282 - 1407
		128	UNSIGNED8	R	RxPDO	Output byte 1408	Receive process data, byte 1408
0x210B	RECORD	N/A	UNSIGNED8	N/A	N/A	Outputs	N/A
		0	UNSIGNED8	R	No	Number of entries	78
		1	UNSIGNED8	R	RxPDO	Output byte 1409	Receive process data, byte 1409
		…	UNSIGNED8	R	RxPDO	Output byte XXXX	Receive process data, bytes 1410 - 1487
		78	UNSIGNED8	R	RxPDO	Output byte 1486	Receive process data, byte 1486

16.6. ASCII Table

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1x	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2x	(sp) 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6x	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

16.7. RS232/RS485 Electrical Connection

16.7.1. RS485 Typical Connection

IMPORTANT

The Communicator has a typical RS485 connection and there are no internal terminations.

To ensure signal integrity and prevent signal reflections on the bus, add termination resistors externally at both ends of the RS485 transmission line.

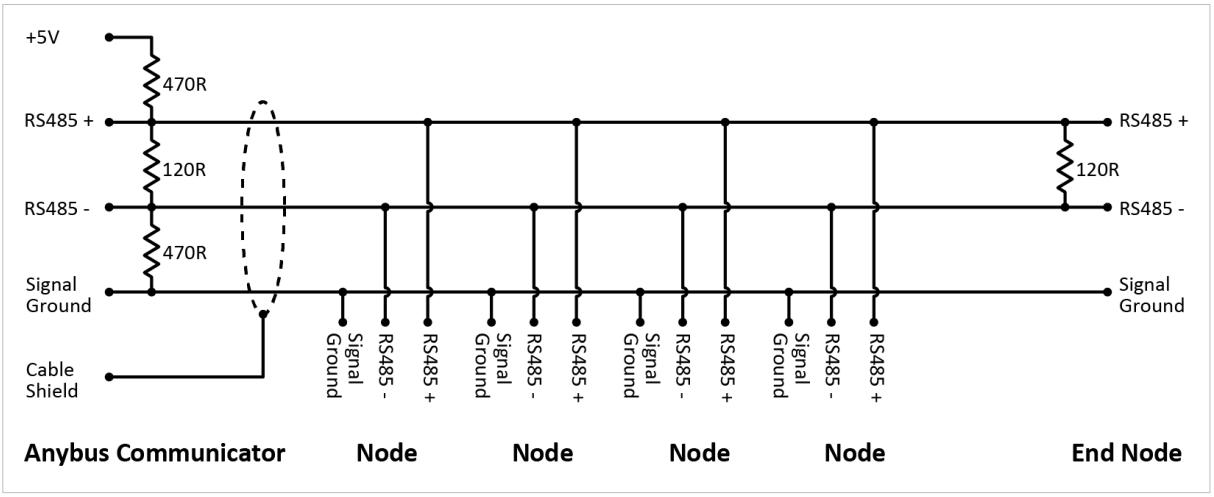


Figure 143. RS485

16.7.2. RS232 Typical Connection

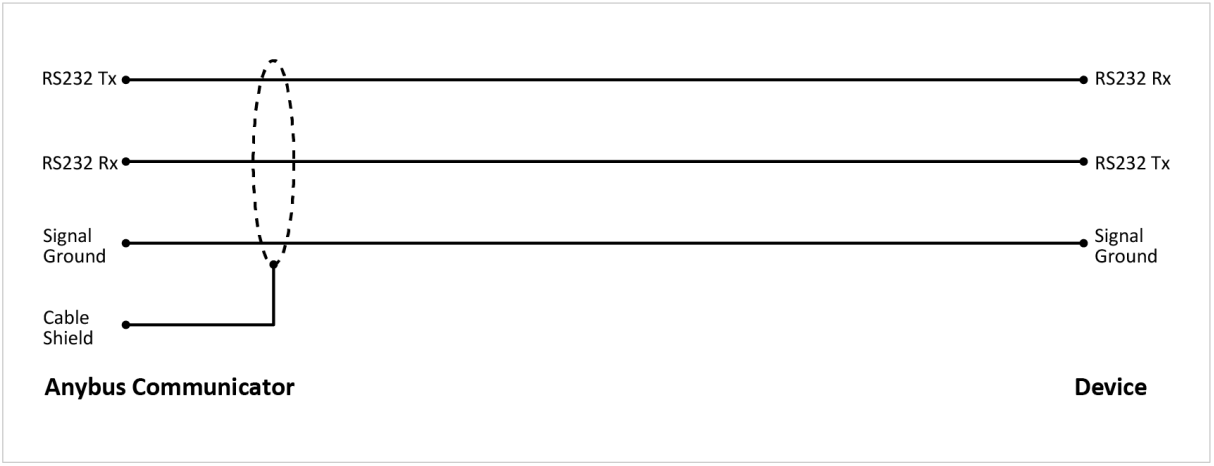


Figure 144. RS232

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

HMS Networks:

ABC3061-A