

# Anybus<sup>®</sup> Communicator<sup>™</sup>

EtherNet/IP<sup>™</sup> to Modbus RTU/Serial

## USER MANUAL

SCM-1202-152 1.10 en-US ENGLISH



---

# 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.

---

# Table of Contents

Page

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Preface .....</b>                              | <b>5</b>  |
| 1.1      | About This Document .....                         | 5         |
| 1.2      | Document Conventions .....                        | 5         |
| 1.3      | Trademarks .....                                  | 5         |
| <b>2</b> | <b>Safety .....</b>                               | <b>6</b>  |
| 2.1      | Intended Use .....                                | 6         |
| 2.2      | General Safety .....                              | 6         |
| <b>3</b> | <b>Preparation .....</b>                          | <b>7</b>  |
| 3.1      | Cabling .....                                     | 7         |
| 3.2      | System Requirements .....                         | 7         |
| 3.3      | Tools .....                                       | 7         |
| 3.4      | Support and Resources .....                       | 7         |
| 3.5      | HMS Software Applications .....                   | 8         |
| 3.6      | Third-Party Software Applications .....           | 8         |
| <b>4</b> | <b>About Anybus Communicator .....</b>            | <b>9</b>  |
| 4.1      | Serial Protocol Communication .....               | 9         |
| 4.2      | How the Communication Works .....                 | 13        |
| 4.3      | How the Data Exchange Works .....                 | 15        |
| 4.4      | Data Integrity .....                              | 15        |
| <b>5</b> | <b>Installation .....</b>                         | <b>16</b> |
| 5.1      | External Parts .....                              | 16        |
| 5.2      | DIN Rail Mounting .....                           | 17        |
| 5.3      | Connecting to EtherNet/IP Network .....           | 18        |
| 5.4      | Connecting to Serial RS232/RS485 Subnetwork ..... | 19        |
| 5.5      | Connecting to Power .....                         | 20        |
| 5.6      | Security Switch .....                             | 21        |
| 5.7      | Locking the Cables .....                          | 22        |
| 5.8      | DIN Rail Demount .....                            | 23        |
| <b>6</b> | <b>Configuration Quick Guide .....</b>            | <b>24</b> |
| 6.1      | Prepare Configuration .....                       | 24        |
| 6.2      | Setup New Configuration .....                     | 26        |
| 6.3      | PLC Configuration .....                           | 28        |
| 6.4      | Verify Operation .....                            | 30        |

---

|           |  |            |
|-----------|--|------------|
| <b>7</b>  | <b>Communicator Configuration .....</b>                    | <b>31</b>  |
| 7.1       | Connecting the Communicator .....                          | 31         |
| 7.2       | Access the Built-In Web Interface From HMS IPconfig.....   | 32         |
| 7.3       | Access the Built-In Web Interface From a Web Browser ..... | 34         |
| 7.4       | Communicator Built-In Web Interface Overview .....         | 35         |
| 7.5       | General Subnetwork Settings .....                          | 36         |
| 7.6       | About Transaction Templates.....                           | 40         |
| 7.7       | Build Transaction Templates .....                          | 46         |
| 7.8       | Nodes and Transactions .....                               | 58         |
| 7.9       | High Level Network Settings.....                           | 67         |
| 7.10      | I/O Data Map .....   | 70         |
| 7.11      | Apply Configuration.....                                   | 79         |
| 7.12      | Use an Existing Configuration .....                        | 81         |
| <b>8</b>  | <b>PLC Configuration .....</b>                             | <b>82</b>  |
| 8.1       | Export I/O Data Map .....                                  | 82         |
| 8.2       | Export Product EDS File.....                               | 82         |
| 8.3       | CIP Objects .....  | 83         |
| <b>9</b>  | <b>Verify Operation.....</b>                               | <b>84</b>  |
| 9.1       | Communicator Status Monitor .....                          | 84         |
| 9.2       | Communicator LED Indicators .....                          | 86         |
| 9.3       | Ethernet LED Indicators.....                               | 87         |
| <b>10</b> | <b>Use Cases .....</b>                                     | <b>88</b>  |
| 10.1      | Temperature Regulator - Modbus RTU Use Case.....           | 88         |
| 10.2      | AC Motor Drive - Custom Request/Response Use Case .....    | 94         |
| 10.3      | Barcode Scanner - Custom Produce/Consume Use Case.....     | 101        |
| <b>11</b> | <b>Maintenance.....</b>                                    | <b>106</b> |
| 11.1      | Configuration File Handling.....                           | 106        |
| 11.2      | Clear and Revert Configuration.....                        | 108        |
| 11.3      | Firmware Management.....                                   | 108        |
| <b>12</b> | <b>Troubleshooting .....</b>                               | <b>111</b> |
| 12.1      | Diagnostics .....  | 111        |
| 12.2      | Reset to Factory Settings.....                             | 114        |
| 12.3      | Firmware Upgrade Error Management.....                     | 116        |
| 12.4      | Support .....  | 118        |
| <b>13</b> | <b>Technical Data .....</b>                                | <b>119</b> |
| 13.1      | Technical Specifications.....                              | 119        |

---

|          |   |            |
|----------|---|------------|
| <b>A</b> | <b>Reference Guides .....</b>                     | <b>121</b> |
| A.1      | About Input Registers and Holding Registers ..... | 121        |
| A.2      | Modbus Data Model.....                            | 121        |
| A.3      | Modbus Transactions.....                          | 121        |
| A.4      | Modbus Exception Codes .....                      | 122        |
| A.5      | ASCII Table.....                                  | 122        |
| A.6      | RS485/RS232 Electrical Connection .....           | 123        |

**This page intentionally left blank**

# 1 Preface

## 1.1 About This Document

This manual describes the installation and configuration of the Anybus Communicator.

For additional documentation and software downloads, FAQs, troubleshooting guides and technical support, please visit [www.anybus.com/support](http://www.anybus.com/support).

## 1.2 Document Conventions

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
- ▶ An action
  - and a result

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

```
Program code and script examples
```

Cross-reference within this document: [Document Conventions, p. 5](#)

External link (URL): [www.hms-networks.com](http://www.hms-networks.com)



### **WARNING**

Instruction that must be followed to avoid a risk of death or serious injury.



### **Caution**

Instruction that must be followed to avoid a risk of personal injury.



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.



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

## 1.3 Trademarks

Anybus® is a registered trademark of HMS Networks AB.

All other trademarks are the property of their respective holders.

## 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.



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 Preparation

### 3.1 Cabling

Have the following cables available:

- **Ethernet cable for configuration**
- **Ethernet cable for connecting to the high level network**
- **Power cable**

### 3.2 System Requirements

#### 3.2.1 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                    |

#### 3.2.2 Supported Web Browsers

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

- Google Chrome
- Microsoft Edge
- Mozilla Firefox

### 3.3 Tools

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*.

### 3.4 Support and Resources

For additional documentation and software downloads, FAQs, troubleshooting guides and technical support, please visit [www.anybus.com/support](http://www.anybus.com/support).



*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.*

## 3.5 HMS Software Applications

Download the software installation files and user documentation from [www.anybus.com/support](http://www.anybus.com/support).

### IPconfig

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



*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.*

---



*IPconfig is only available for Windows.*

---

## 3.6 Third-Party Software Applications

Microsoft Excel, or equivalent software application that supports the Office Open XML Workbook (xlsx) file format.

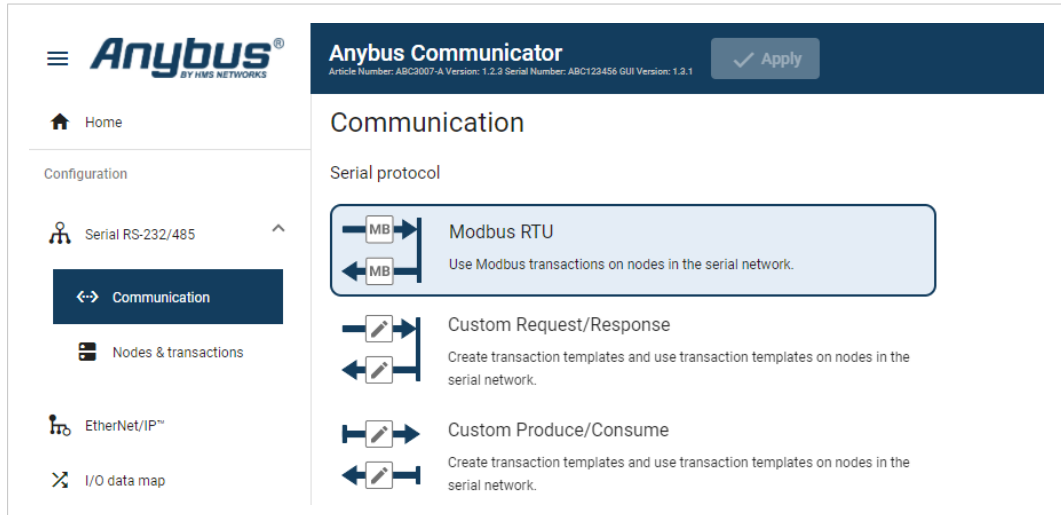
Needed to open and read the I/O data mapping file.

## 4 About Anybus Communicator

### 4.1 Serial Protocol Communication

#### 4.1.1 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**.



#### 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.

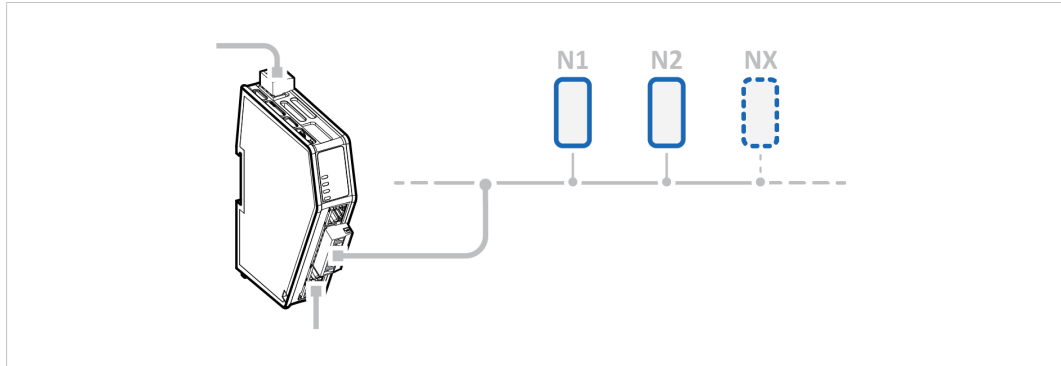
The Communicator may consume and/or produce messages on the subnetwork.

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

### 4.1.2 Serial Protocol Building Blocks

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

#### Node



A node represents a single device on the subnetwork.

Each node can be associated with a number of transactions.

#### Nodes and Transactions

**Anybus Communicator**
✓ Apply

**Nodes**

Custom request/response

+ Add node

1 My Drive
⋮

**Node settings**

|                                |                                       |                                      |                                       |                                |
|--------------------------------|---------------------------------------|--------------------------------------|---------------------------------------|--------------------------------|
| Node address                   | Name                                  | Timeout time                         | Reconnection ti...                    | Retries                        |
| <input type="text" value="1"/> | <input type="text" value="My Drive"/> | <input type="text" value="1000"/> ms | <input type="text" value="10000"/> ms | <input type="text" value="0"/> |

**Transactions**

+ Add
⌵
Duplicate
Delete

|                                     | Active                              | Transaction name | Transaction template name | Size to EtherNet/IP™ (bytes) | Size from EtherNet/IP™ (bytes) |
|-------------------------------------|-------------------------------------|------------------|---------------------------|------------------------------|--------------------------------|
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Control Word     | Write Parameter (0x02)    | 0                            | 2                              |
| <input type="checkbox"/>            | <input checked="" type="checkbox"/> | Speed            | Write Parameter (0x02)    | 0                            | 2                              |
| <input type="checkbox"/>            | <input checked="" type="checkbox"/> | Status Word      | Read Parameter (0x01)     | 2                            | 0                              |
| <input type="checkbox"/>            | <input checked="" type="checkbox"/> | Actual Speed     | Read Parameter (0x01)     | 2                            | 0                              |

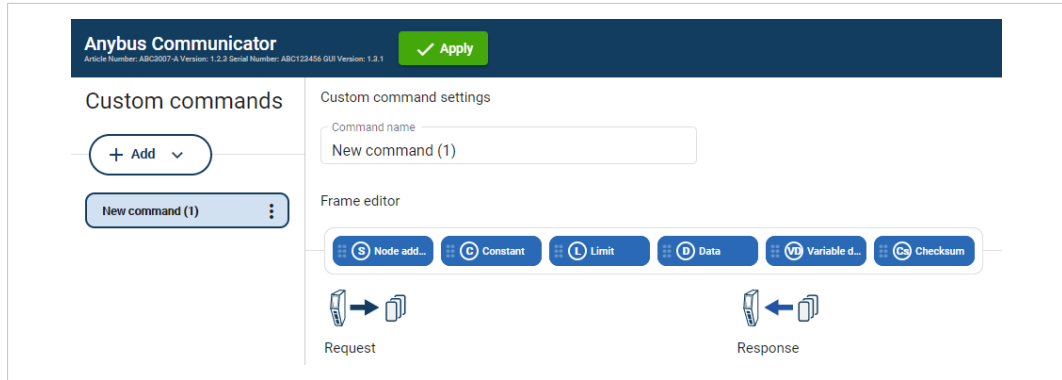
Transactions are based on standard Modbus RTU transactions (Modbus RTU serial protocol) or transactions templates (Custom Request/Response or Produce/Consume serial protocol) and define the data to be sent or received.

Each transaction has a number of parameters that need to be configured to define how data is to be sent / received.

#### 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.



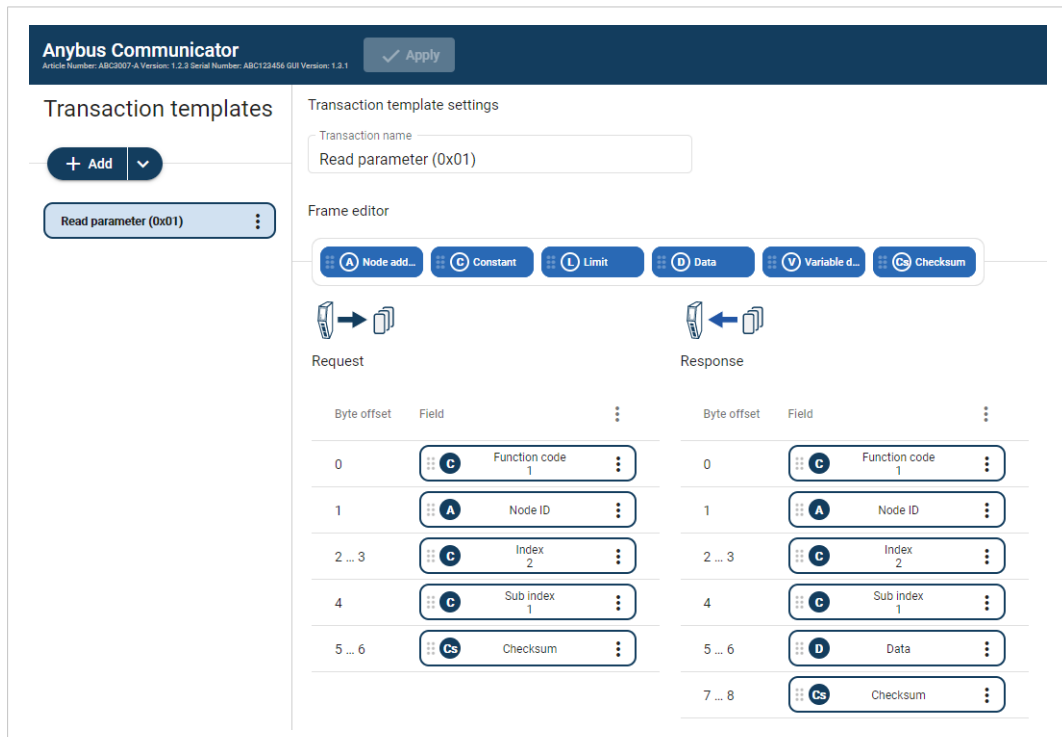
Frame fields are low level entities used to compose transactions.

A frame field 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.



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.

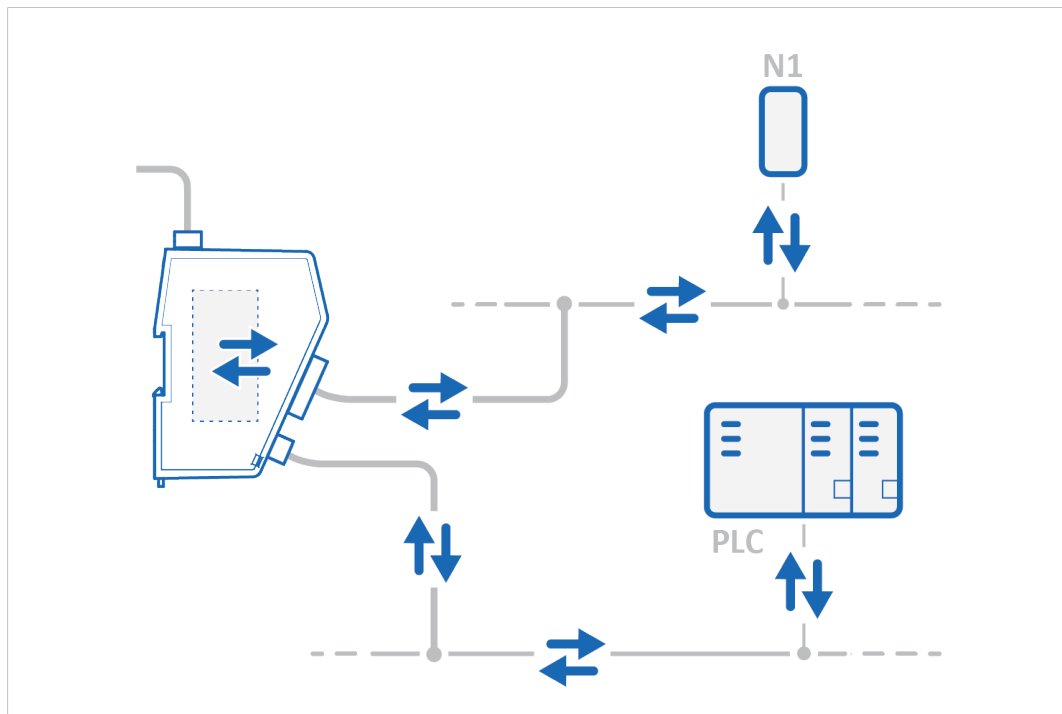
The transaction templates are stored in the Communicator and can be reused multiple times.

**Examples:**

If you have a common read transaction. Then you can create one single transaction template for the read transaction and reuse it multiple 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.

## 4.2 How the Communication Works



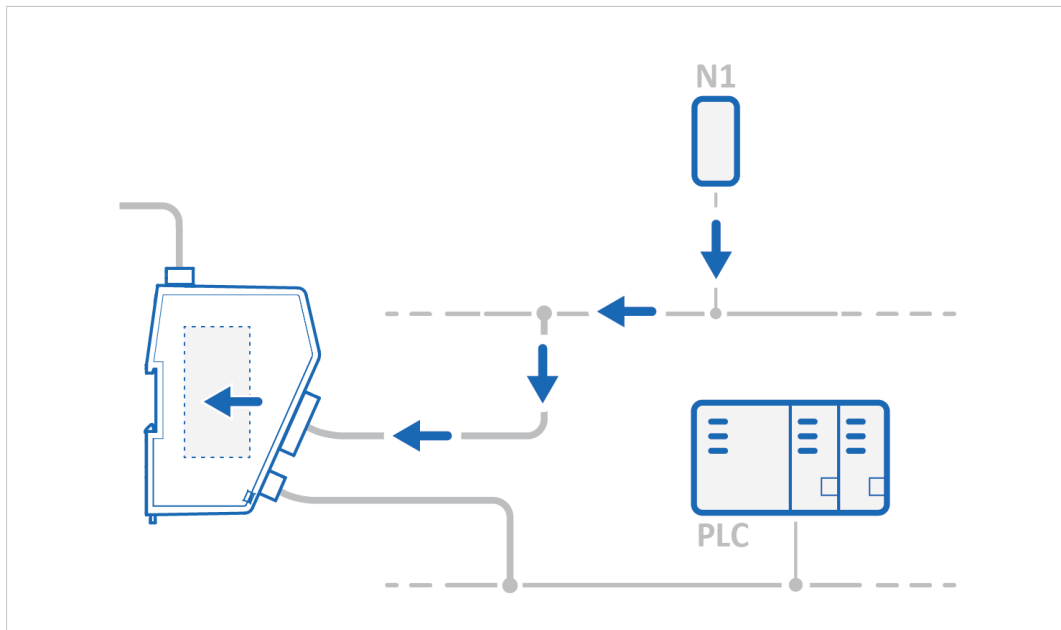
The Communicator enables communication, data exchange, between one or more server devices connected to a *serial subnetwork* and a client device connected to a *high level network*.

For example:

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

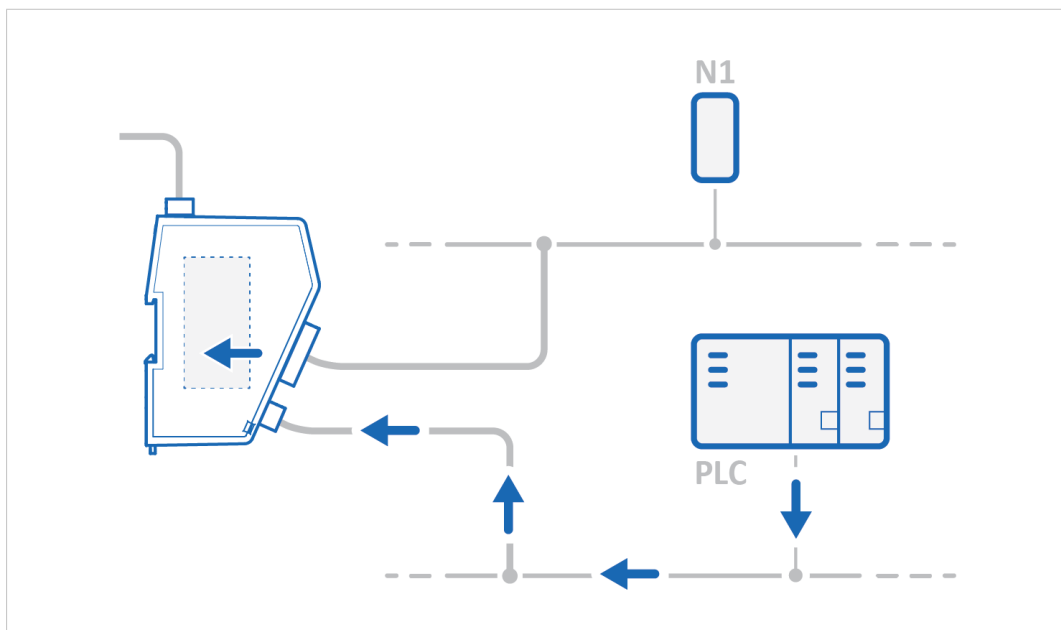
The Communicator main task is to cyclically send the transactions that the server(s) are configured to execute, in order to request and transfer process data.

**Request process data**



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.

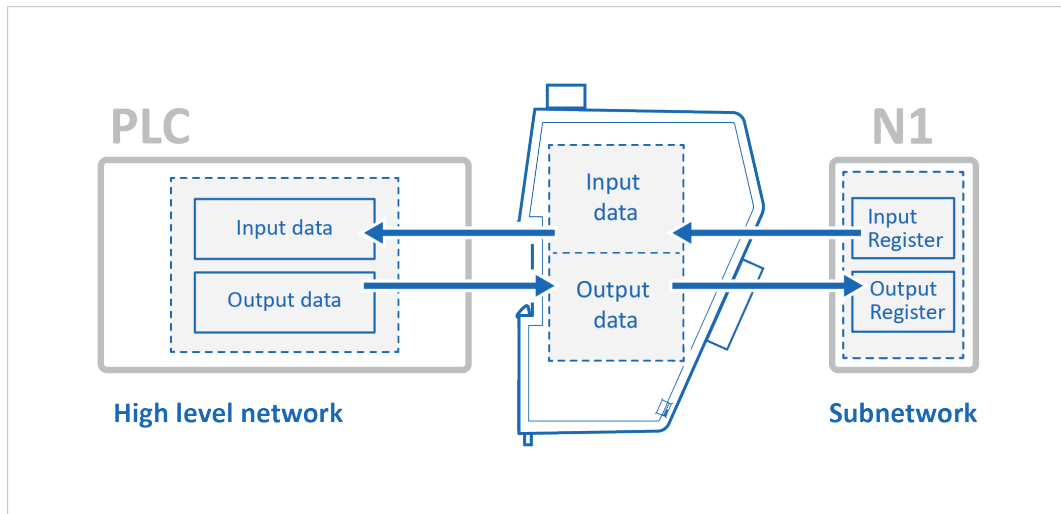
**Transfer process data**



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.



### 4.3 How the Data Exchange Works



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.

The Communicator internal memory buffer is divided into two areas: Input data and Output data.

#### Input Data

This *Input data area* is read by the *high level network*.

The Communicator can handle up to 1500 bytes input data.

#### Output Data

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

The Communicator can handle up to 1500 bytes output data.

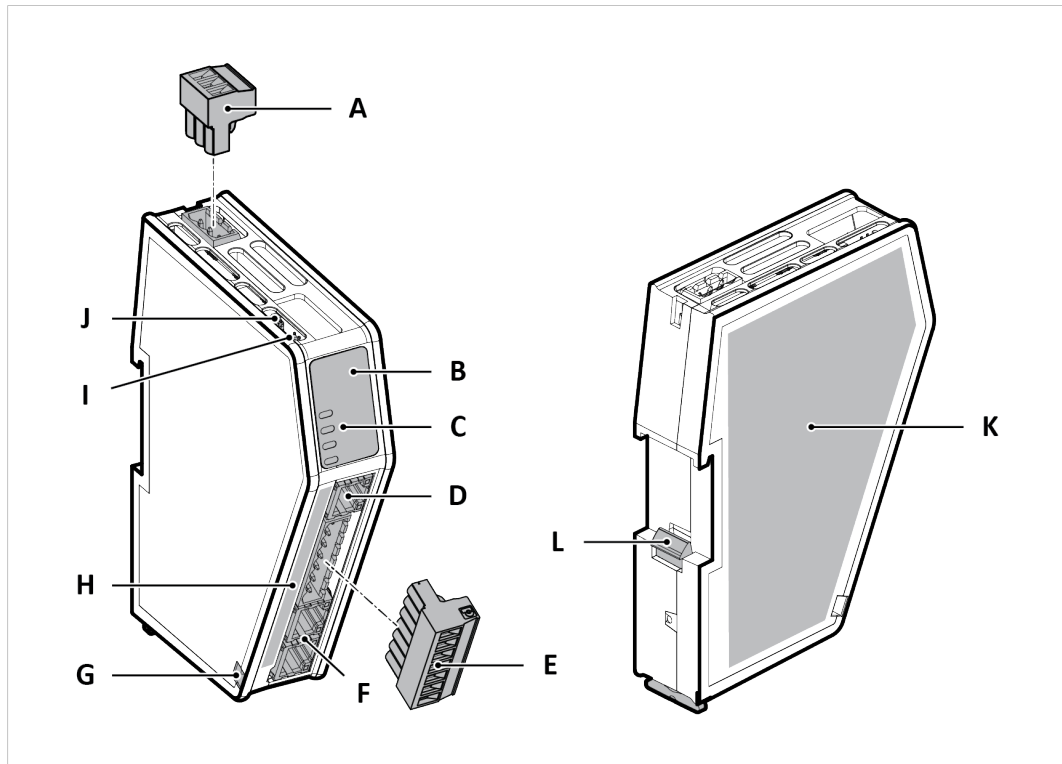
### 4.4 Data Integrity

A snapshot of the process data buffer between the Modbus 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.

## 5 Installation

### 5.1 External Parts

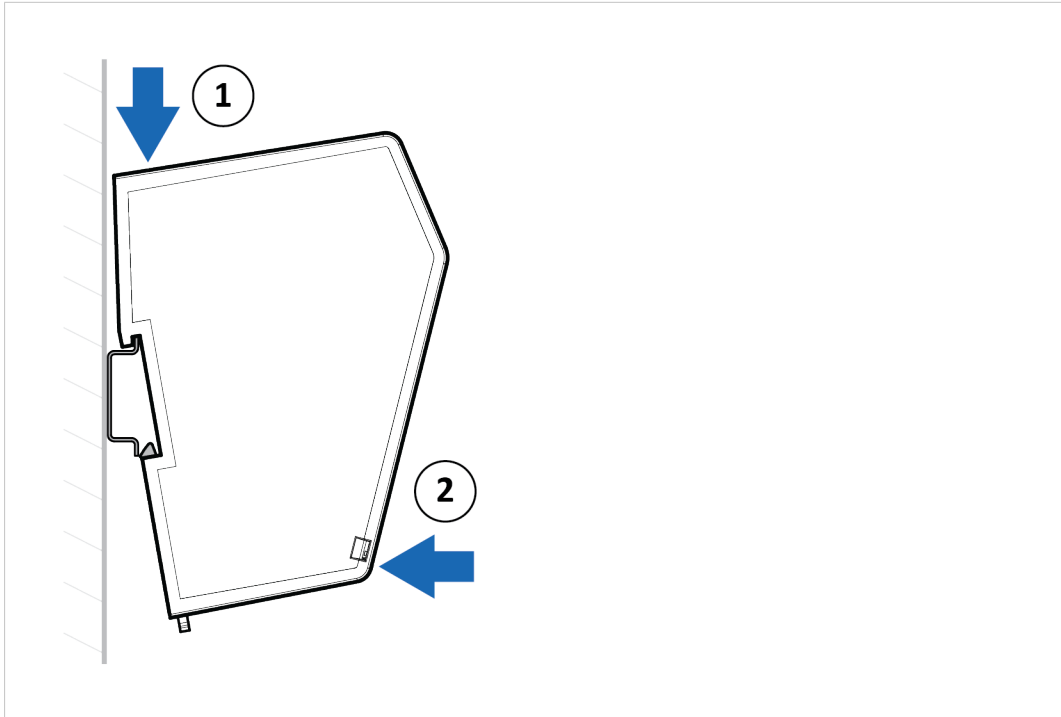


- |                               |  |  |
|-------------------------------|--|--|
| A. Power connector            | E. 7-pin connector                       | I. Security switch                               |
| B. Label with LED designation | F. Ethernet port x 2                     | J. Factory reset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

## 5.2 DIN Rail Mounting



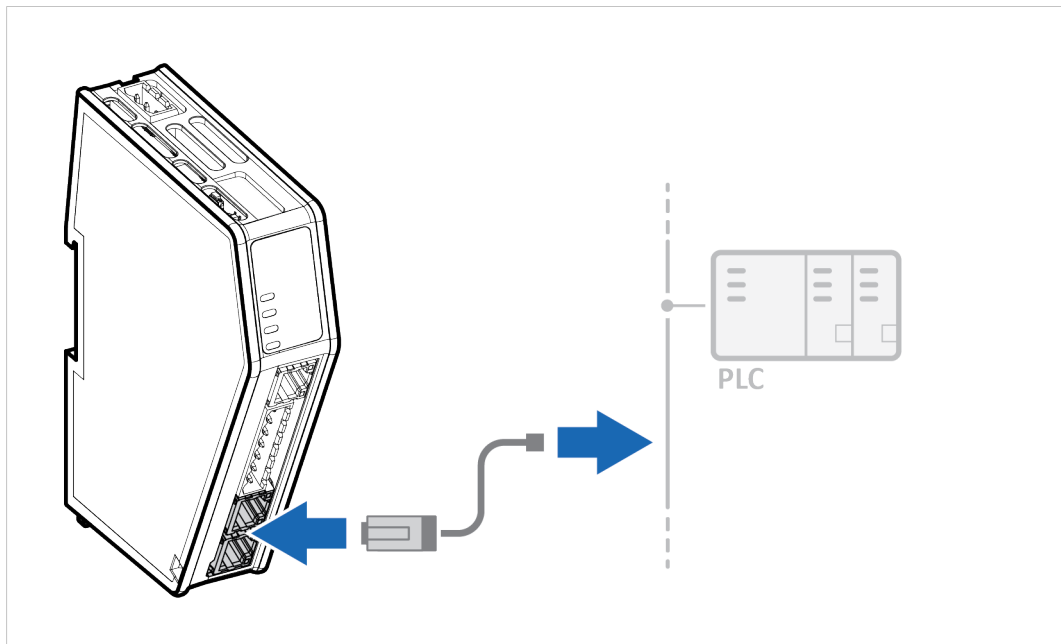
The equipment must be electrically grounded through the DIN rail for EMC compliance. Make sure that the equipment is correctly mounted on the rail and that the rail is properly grounded.



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.

### 5.3 Connecting to EtherNet/IP Network



1. Connect the Communicator to your EtherNet/IP network.

| EtherNet/IP Connector |     |             |
|-----------------------|-----|-------------|
|                       | Pin | Description |
|                       | 1   | TD+         |
|                       | 2   | TD-         |
|                       | 3   | RD+         |
|                       | 4   |             |
|                       | 5   |             |
|                       | 6   | RD-         |
|                       | 7   |             |
|                       | 8   |             |

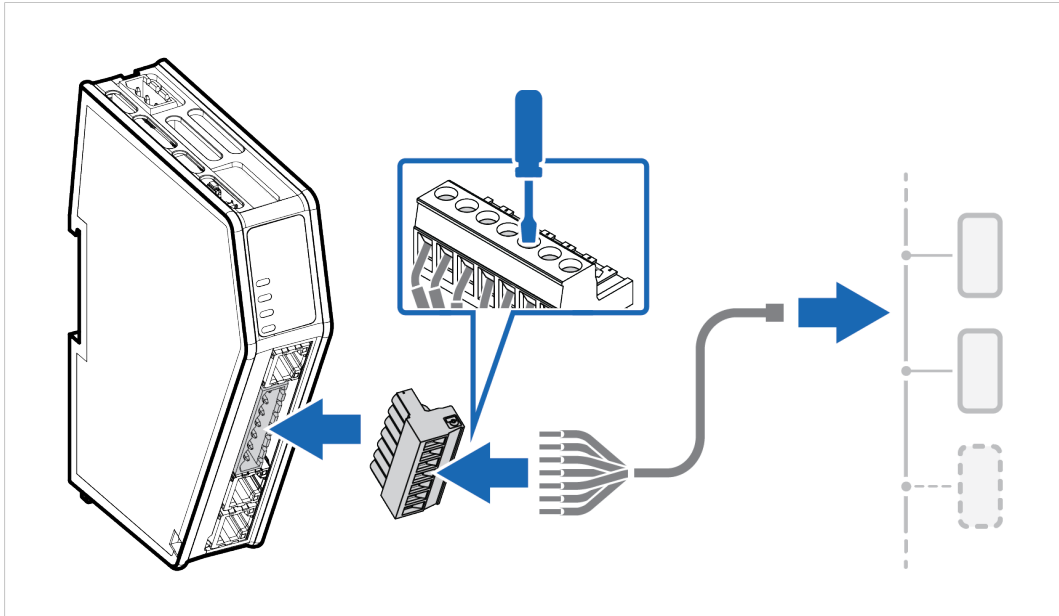
**To Do Next**

Connect the Communicator to the serial subnetwork and to power.

Check LED status, refer to [Communicator LED Indicators, p. 86](#).

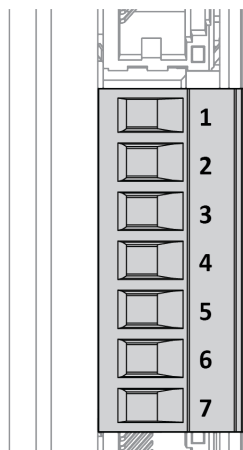
## 5.4 Connecting to Serial RS232/RS485 Subnetwork

**i** Use minimum 90 oC copper (Cu) wire only.



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 Output       |
| 7   | RS232 Rx Input        |

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 EtherNet/IP network and to power.

Check LED status, refer to [Communicator LED Indicators, p. 86](#).

## 5.5 Connecting to Power

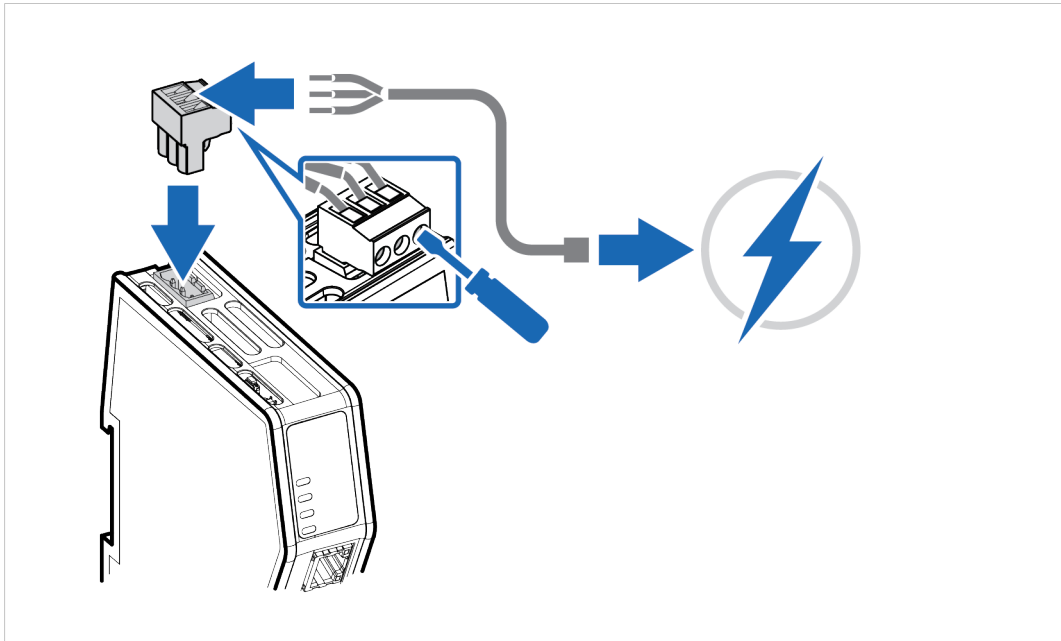


### Caution

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



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



1. Insert the cable wires to the terminal block and tighten the wire clamp screws.

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

2. Connect the terminal block to the Communicator.
3. Connect the Communicator to a power supply.
4. Turn on the power supply.

### To Do Next

Connect the Communicator to the EtherNet/IP and serial subnetwork.

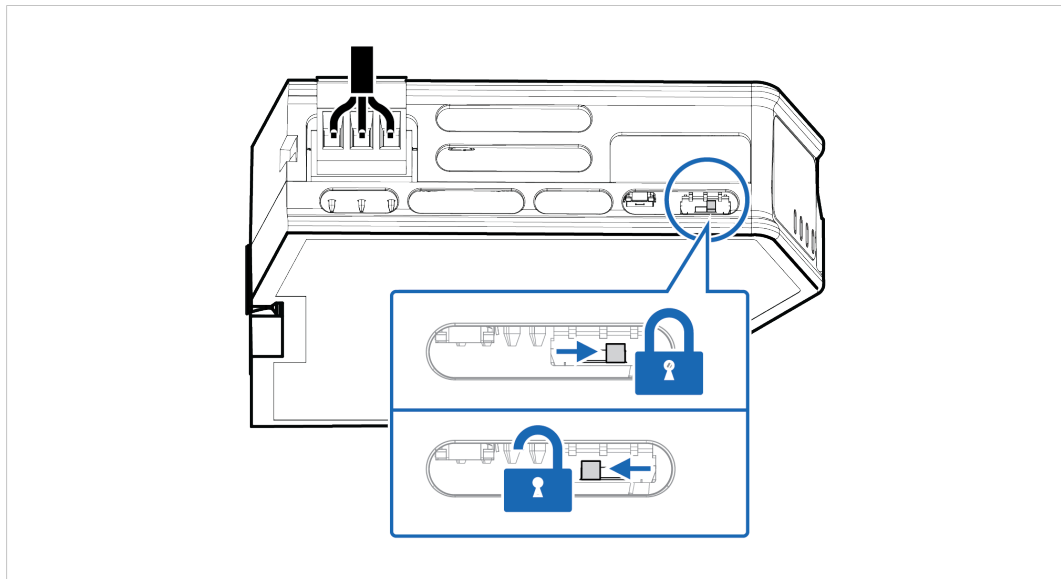
Check LED status, refer to [Communicator LED Indicators, p. 86](#).

## 5.6 Security Switch

**i** 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 can not be accessed and the Communicator can not be configured. Network specific parameters, configured via the PLC is still available.

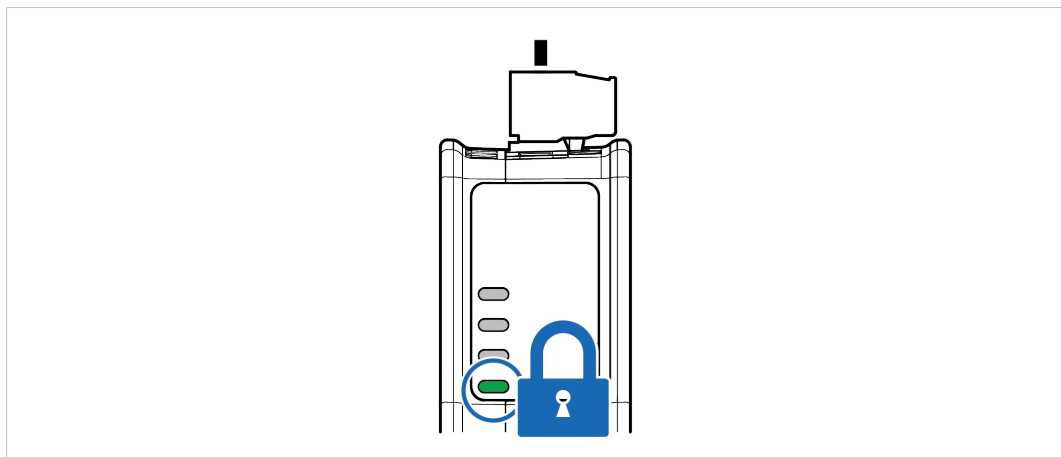
### To Lock and Unlock the Security Switch



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**.

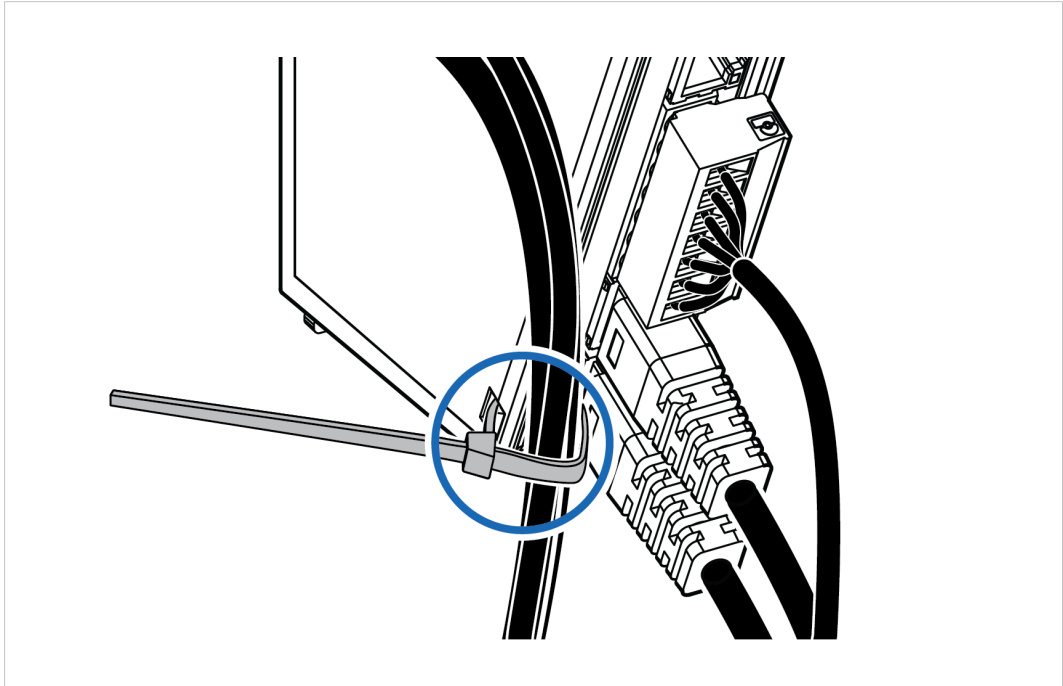
### Security Switch Status LED



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.

## 5.7 Locking the Cables



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



## 5.8 DIN Rail Demount

### Before You Begin



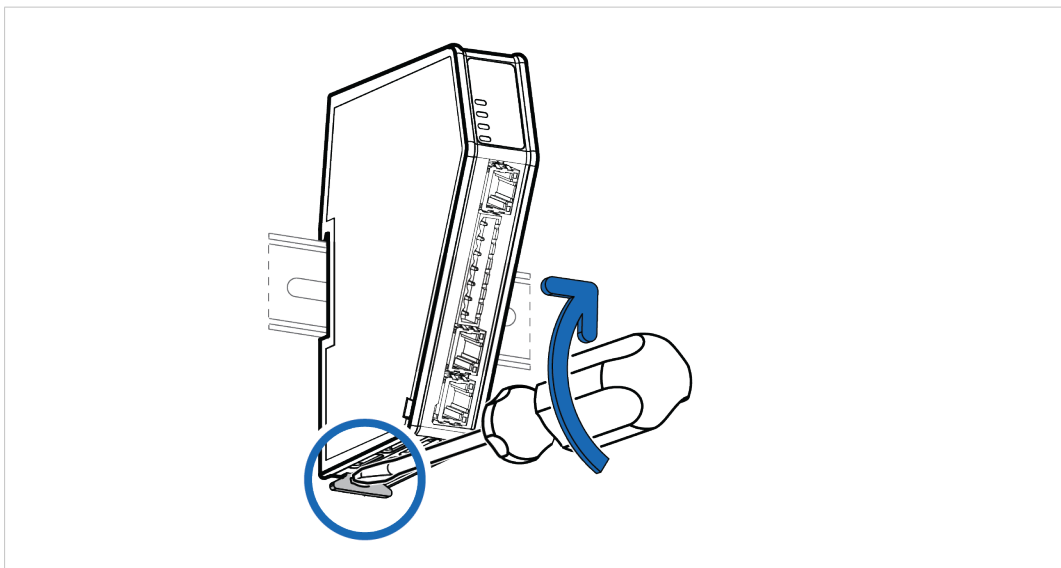
Be careful when removing the Communicator from the DIN-rail. If not removed properly, the DIN rail locking mechanism and the product cover can break.

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

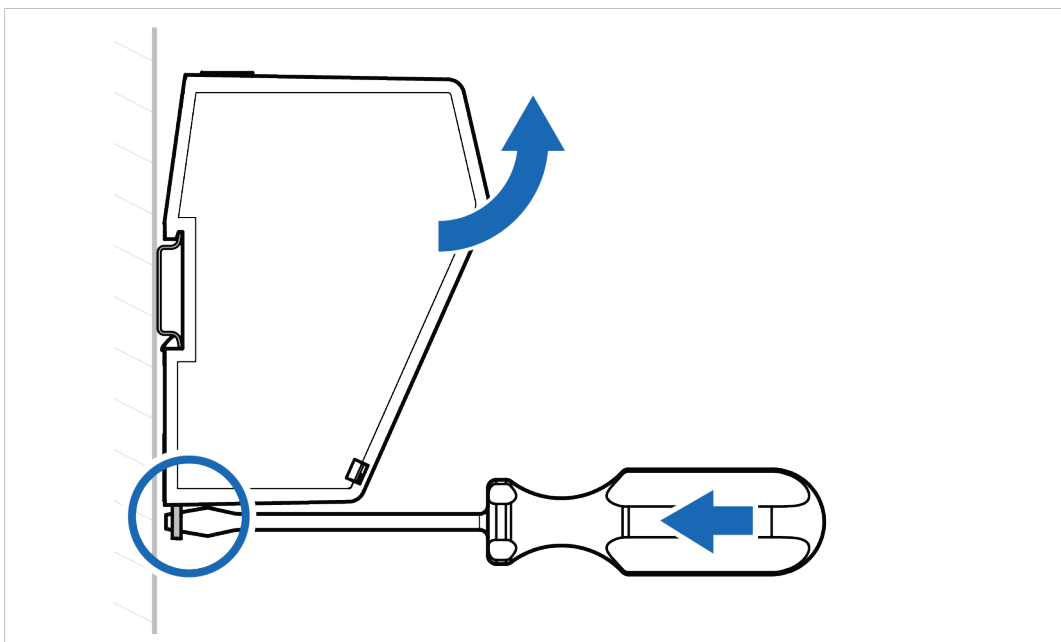
### Procedure

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.



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



## 6 Configuration Quick Guide

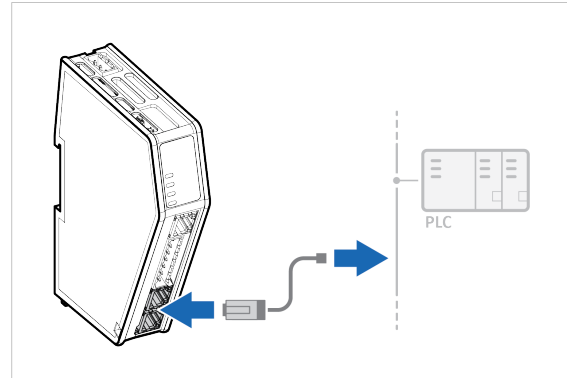
This section is intended to give you a short overview of the tasks you need to perform to configure the Communicator.

For detailed information, please refer to [Communicator Configuration, p. 31](#).

### 6.1 Prepare Configuration

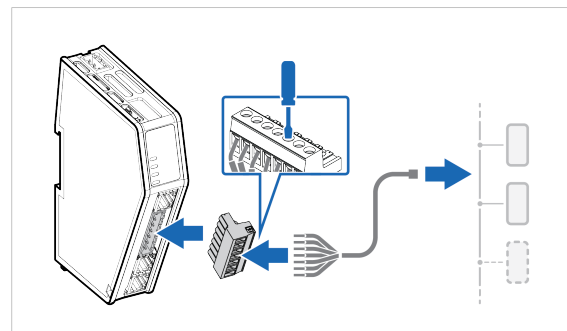
#### 1. Connecting to the high level network

Connect the Communicator to the EtherNet/IP high level network.



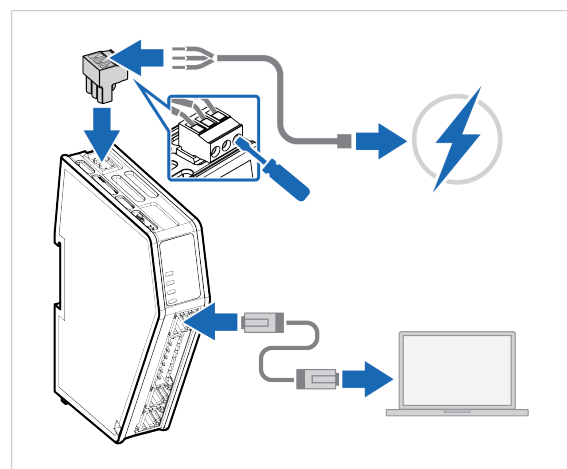
#### 2. Connecting the Communicator to the subnetwork

Connect the Communicator to the serial RS232/RS485 subnetwork.



#### 3. Connecting to PC and power

- Connect an *Ethernet cable* between the *Communicator configuration port* and your PC.
- Connect the Communicator to a power supply.



#### 4. Finding the Communicator on your PC

The Communicator default IP address is **192.168.0.10**.

##### Option 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.

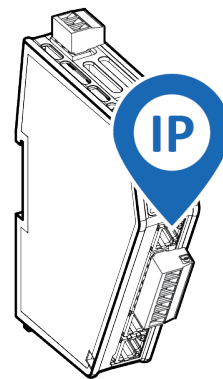


##### Option 2

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.

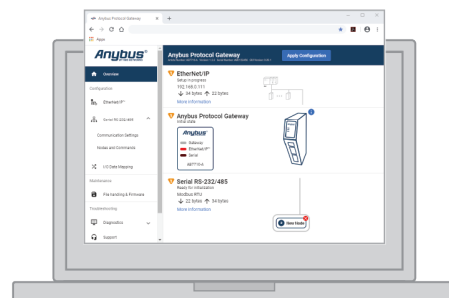
Download the installation files and user documentation from [www.anybus.com/support](http://www.anybus.com/support).



#### 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.



## 6.2 Setup New Configuration

Follow these steps to setup a new Communicator configuration.

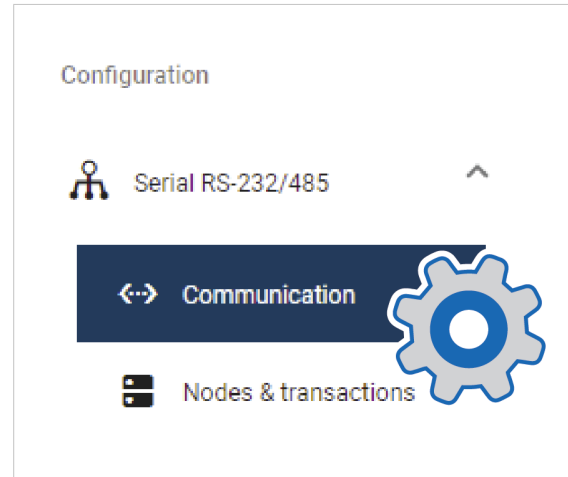
### 1. Subnetwork configuration

On the **Communication** page:

- a. Select a serial protocol:
  - **Modbus RTU** (default)
  - **Custom Request/Response**
  - **Custom Produce/Consume**

For information about the serial protocol types, refer to [Serial Protocol Types, p. 9](#).

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

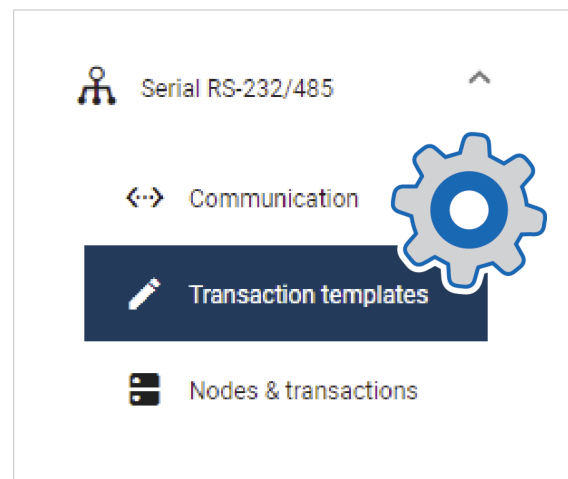


### 2. Create Transaction Templates

This step applies when the serial protocol **Custom Request/Response** or **Custom Produce/Consume** is selected.

On the **Transaction templates** page: Add a transaction template for each way information can be requested/received or produced/consumed.

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

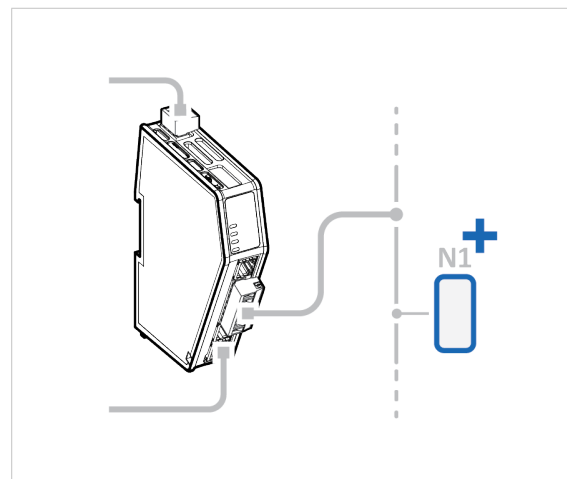


### 3. Add Nodes and Transactions

On the **Nodes & transactions** page:

- a. Add a node and configure the Node settings.
- b. Add transactions to request/receive data or produce/consume data and configure the transaction settings.

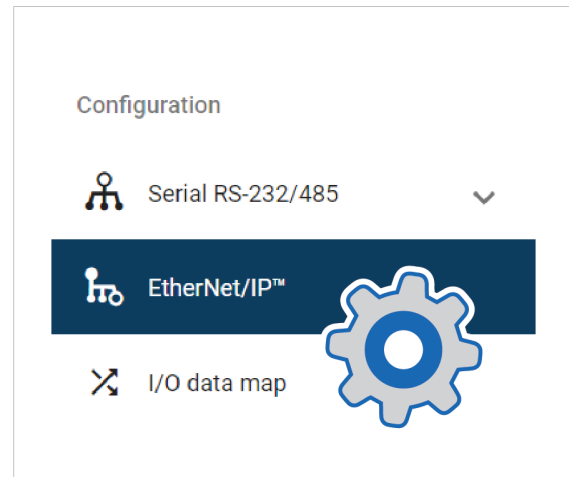
Repeat until you have added and configured all your nodes.



#### 4. High level network configuration

On the **EtherNet/IP™** page:

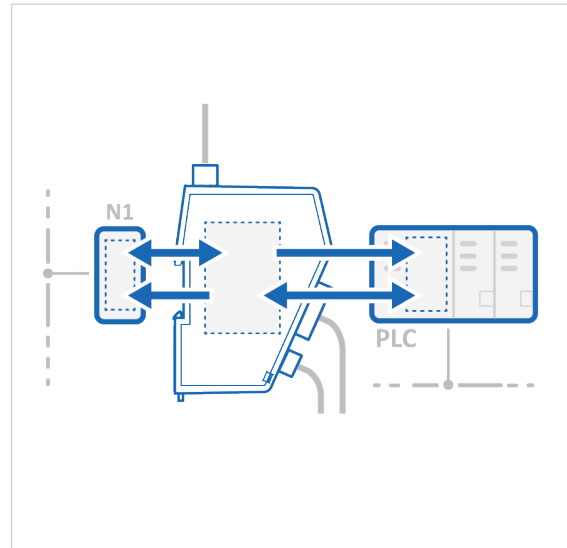
- a. Use Automatic I/O sizes provided by the subnetwork or choose to set them manually.
- b. Enable DHCP server or choose to set a specific IP address.
- c. Apply the IP settings.



#### 5. I/O Data Mapping

The transactions you added to the nodes are automatically mapped to the Communicator internal memory area.

View the added nodes and transactions on the **I/O data map** page.



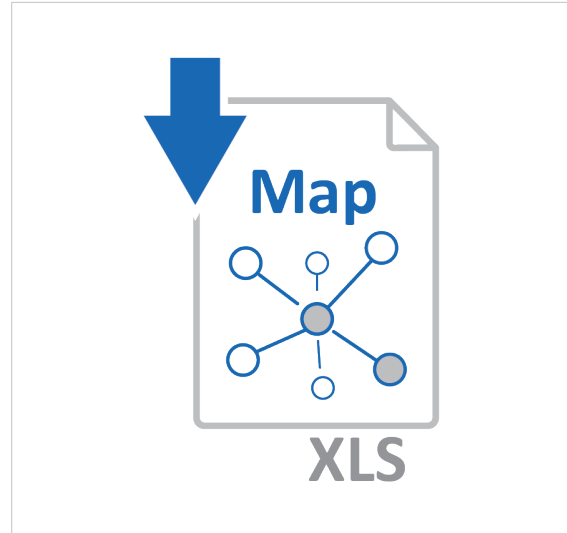
## 6.3 PLC Configuration

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 transactions match.

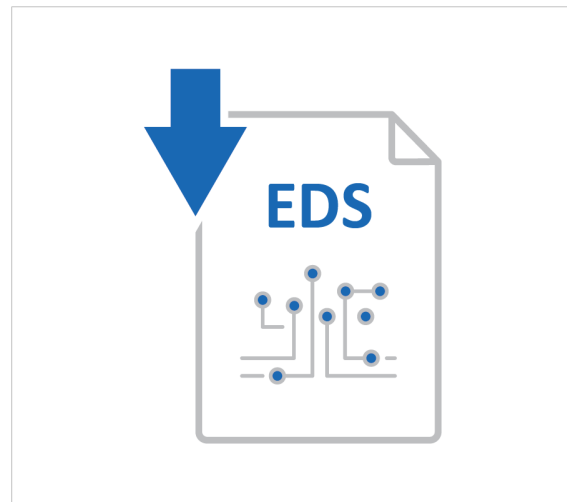
On the **I/O data map** page: You can download the I/O data mapping in a spreadsheet to your PC.



### 2. Download EDS File

Option if the PLC program requires a EDS (Electronic Data Sheet) file.

On the **EtherNet/IP™** page: Download the EDS file to your PC.



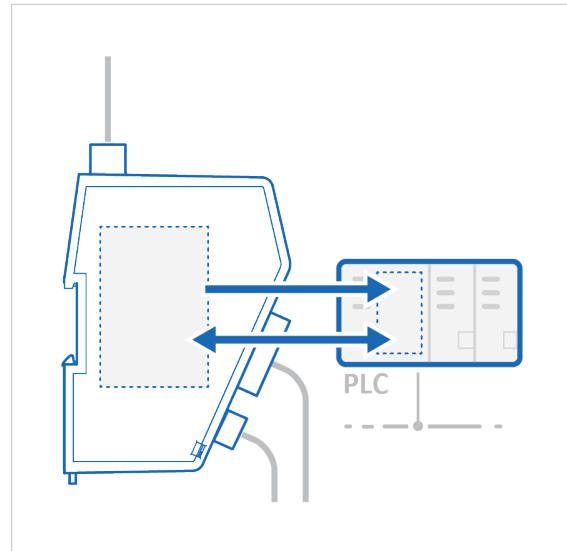
**In the PLC program:****3. Import product file**

Option if the PLC program requires a EDS (Electronic Data Sheet) file.

Import the EDS file into your PLC project.

**4. Configure the communication**

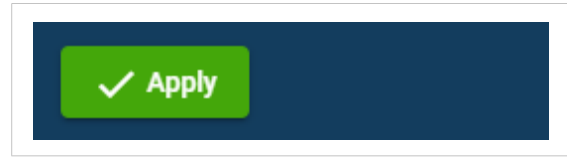
Configure the PLC to communicate with the Communicator according to the I/O data map created in the Communicator.



## 6.4 Verify Operation

### 1. Apply the configuration

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

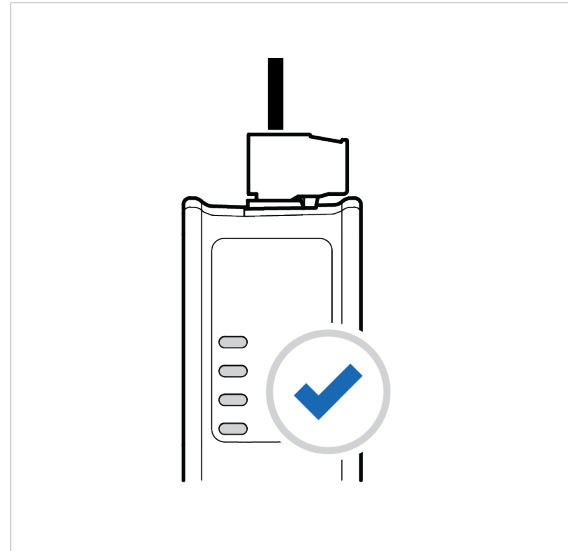


### 2. Verify status and LED indications

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 transactions are sent and received by the Communicator.
- **Event log** page to detect failures and unexpected behavior over time.





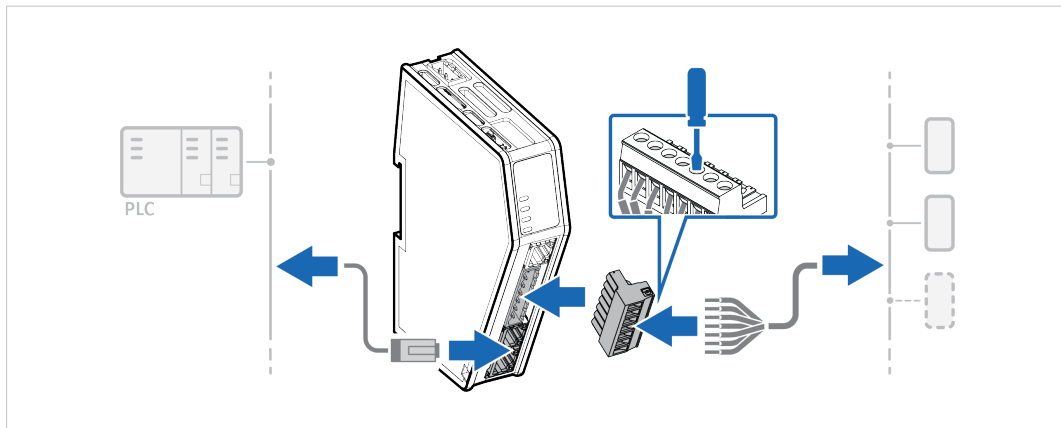
## 7 Communicator Configuration

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

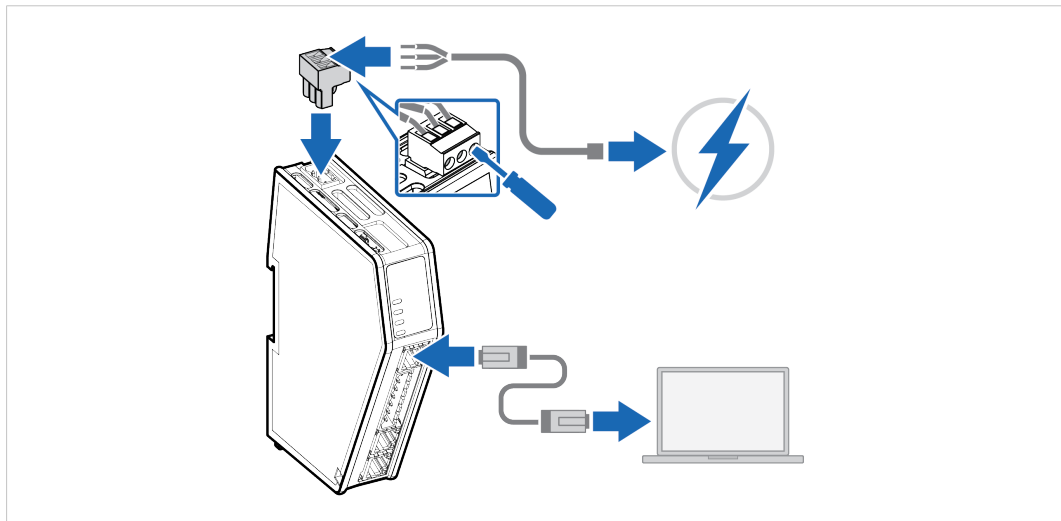
For a more brief overview of the configuration steps, please refer to [Configuration Quick Guide, p. 24](#).

### 7.1 Connecting the Communicator

#### Procedure



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





3. Connect an *Ethernet* cable between the *Communicator configuration port* and your PC.
4. Connect the Communicator to a power supply.


## 7.2 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](http://www.anybus.com/support).

 The Communicator default IP address is **192.168.0.10**.

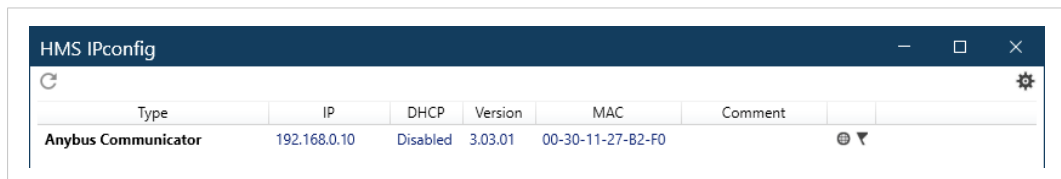
 To access the Communicator built-in web interface, ensure that Port 80 TCP is open in your Firewall. This applies to any Firewall between the web browser and the gateway.

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

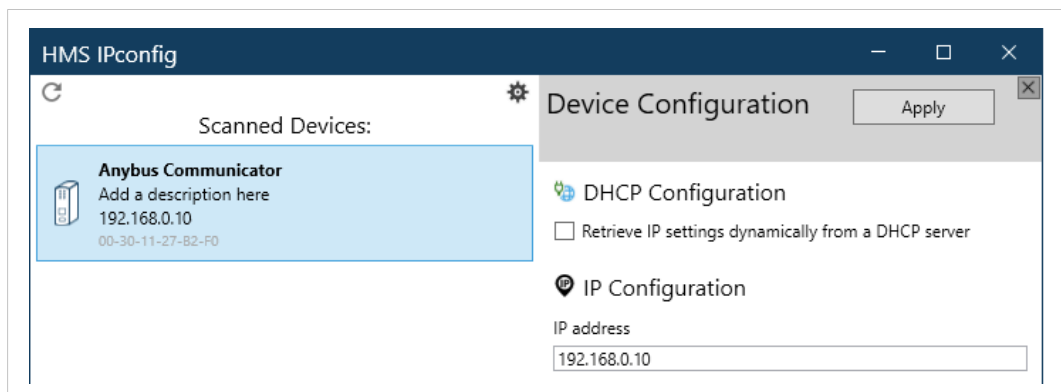
 Make sure the security switch is unlocked. HMS IPconfig cannot configure the Communicator if the security switch is locked.

### Procedure

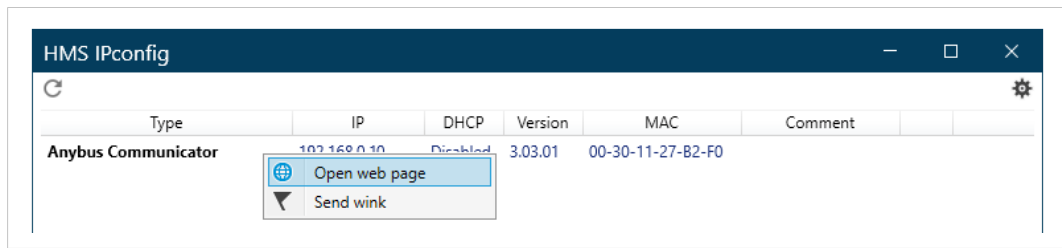
1. Install HMS IPconfig on your PC.
2. Open HMS IPconfig.
  - HMS IPconfig automatically starts scanning for compatible and active HMS devices.
  - Found HMS devices are added to the device list.



3. To open the settings pane, click on the Communicator in the device list.
4. Change the Communicator IP address to one within the same IP address range as your PC.

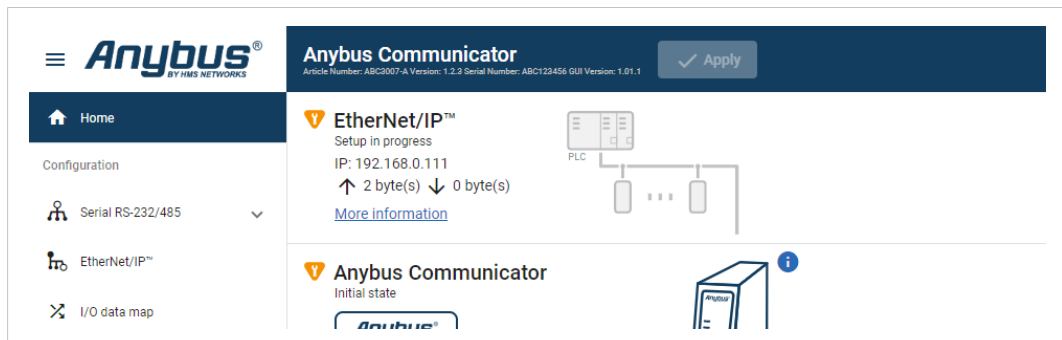


- To open the Communicator built-in web interface, click **Open web page**.



**Result**


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




## 7.3 Access the Built-In Web Interface From a Web Browser

### Before You Begin

 The Communicator default IP address is **192.168.0.10**.

 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.

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

### Procedure

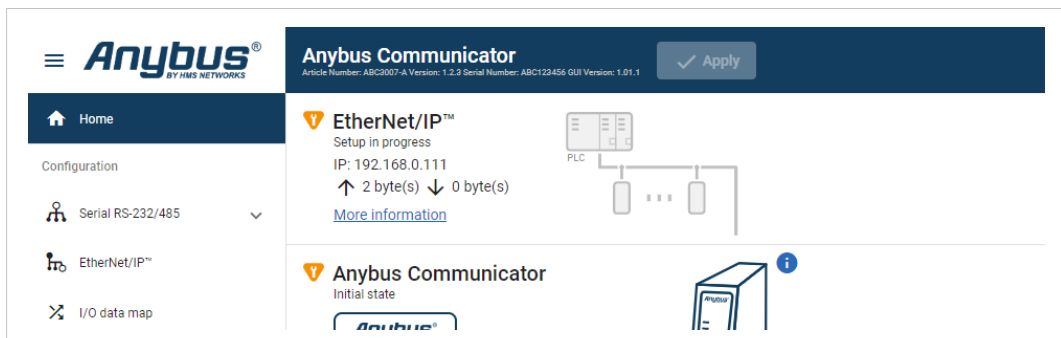
1. On the PC accessing the Communicator built-in web interface, set a static IP address within the same IP address range as the Communicator IP address.



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

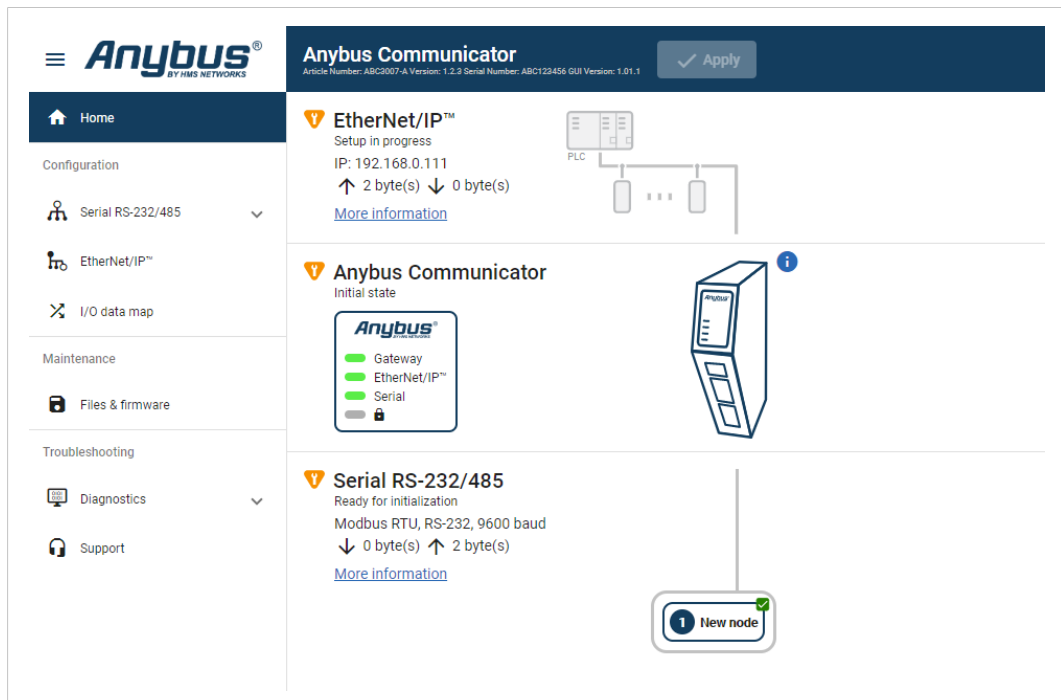


4. To open the built-in web interface Home page, press **Enter**.



## 7.4 Communicator Built-In Web Interface Overview

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



|                             |   |
|-----------------------------|---|
| <b>Home</b>                 | View the Communicator, network and node status.   |
| <b>Apply</b>                | After configuration changes are made and verified, press Apply to make the settings take effect.  |
| <b>Serial RS-232/485</b>    | Serial Subnetwork with Nodes.<br>Select a Serial protocol, use Modbus RTU standard transactions or create your own transaction templates.<br>Configure communication and add nodes and transactions.  |
| <b>EtherNet/IP™</b>         | High Level Network with Client. Configure I/O Size and IP settings.   |
| <b>I/O data map</b>         | View the added transactions mapped to the Communicator internal memory area.  |
| <b>Files &amp; firmware</b> | Save settings in a configuration files, upload configuration files and upgrade firmware.  |
| <b>Diagnostics</b>          | Monitor and troubleshoot the Communicator.  |
| <b>Support</b>              | Contains Communicator product information, Anybus contact information, link to Anybus support website, and product file for download.<br>Here you can generate a support package with product information, to send to your Anybus support technician. |

## 7.5 General Subnetwork Settings

### 7.5.1 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.

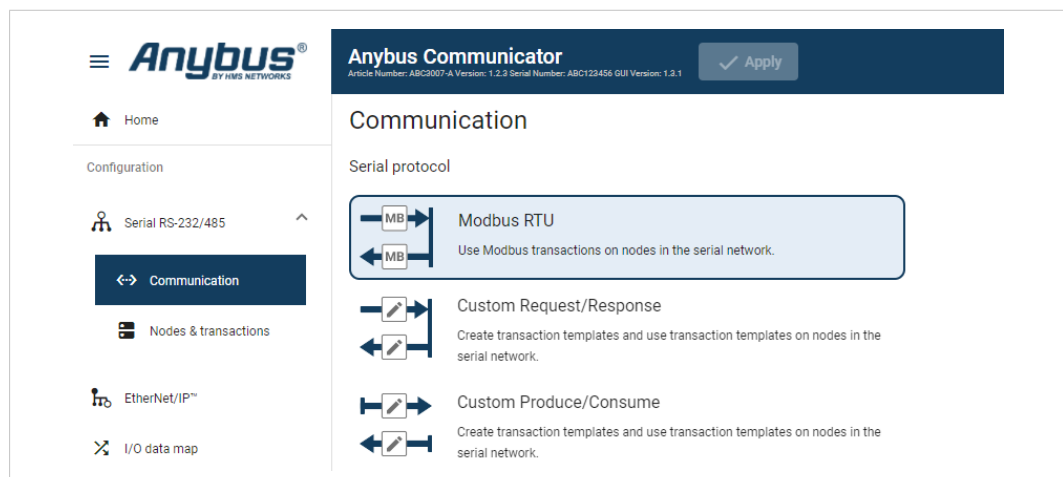


When changing the serial protocol, all settings are reset to default and all added nodes, transactions, and transaction templates are deleted.

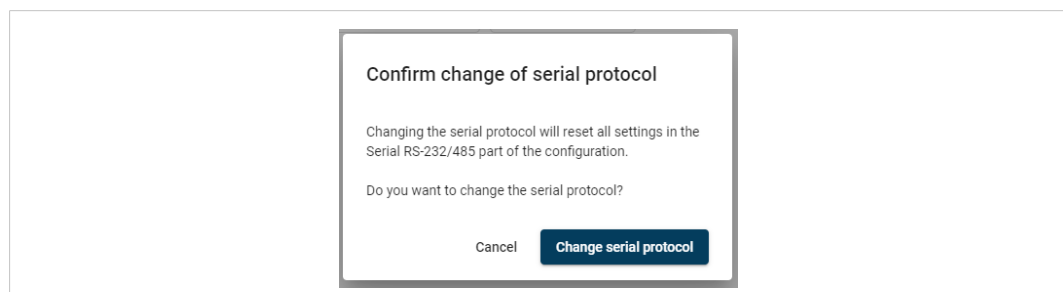
#### Procedure

On the **Communication** page, Serial protocol:

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



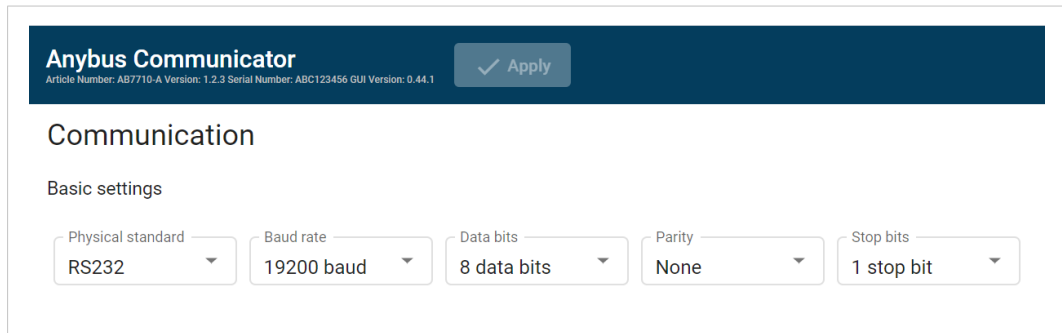
2. To confirm the selected protocol, click **Change serial protocol**.



#### Apply configuration

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

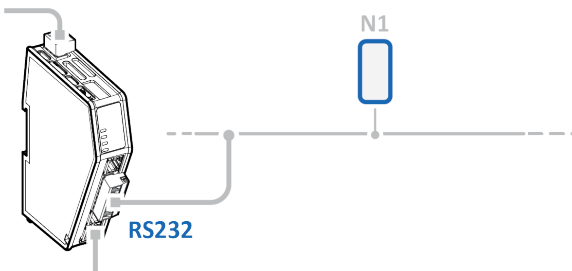
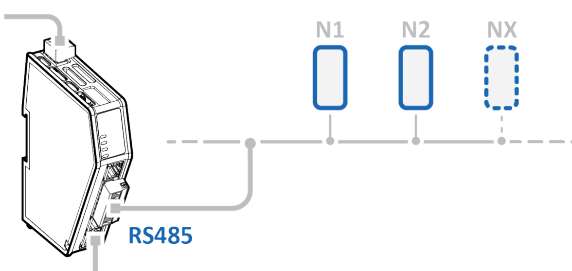
### 7.5.2 Communication Basic Settings



#### Physical standard

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

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

| Setting           | Value                      | Description  |
|-------------------|----------------------------|--|
| Physical standard | RS-232<br>Default standard | Use RS-232 when one <i>single node</i> is connected to the subnetwork.<br>  |
|                   | RS-485                     | Use RS-485 when <i>multiple nodes</i> are connected to the subnetwork.<br> |

#### Baud rate

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

2. 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                |

| Setting | Value       |
|---------|-------------|
|         | 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 8 data bits and can not be changed.

### Parity

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

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

| Setting | Value                | Description   |
|---------|----------------------|---|
| Parity  | None   Default value | No parity checking<br>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.

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

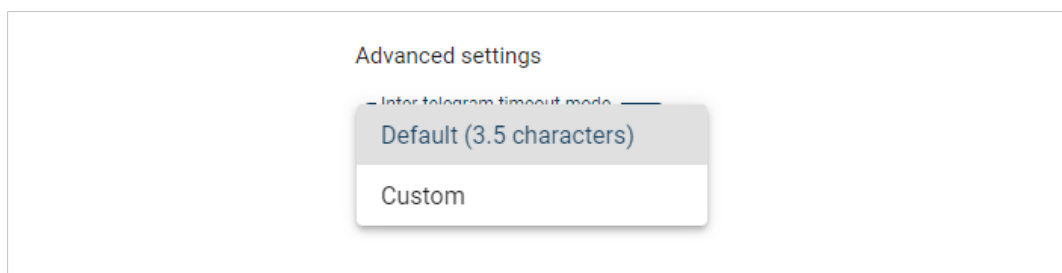
| Setting   | Value                      |
|-----------|----------------------------|
| Stop bits | 1 stop bit   Default value |
|           | 2 stop bit                 |

### Apply configuration

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

## 7.5.3 Communication Advanced Settings

### Inter-Telegram Timeout Mode Settings



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.



Advanced settings

Inter-telegram timeout mode: Custom

Inter-telegram timeout: 17 bits

Inter-telegram delay: 38 bits

Telegram: 00000000

Inter-telegram timeout

Telegram: 00000000

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 Inter-telegram timeout.
- The time between two characters in two different telegrams the same or more than Inter-telegram delay.

## 7.6 About Transaction Templates

This section applies when the **Custom Request/Response** or **Custom Produce/Consume** serial protocol is applied, refer to [Communication Serial Protocol, p. 36](#)

### 7.6.1 Transaction Template Example

#### Custom Request/Response

Request/Response transaction template example:

The screenshot shows the 'Anybus Communicator' interface. On the left, under 'Transaction templates', a template named 'Read parameter (0x01)' is selected. The main area is divided into 'Transaction template settings' and 'Frame editor'. The settings show the transaction name as 'Read parameter (0x01)'. The frame editor has a toolbar with icons for 'Node address', 'Constant', 'Limit', 'Data', 'Variable', and 'Checksum'. Below the toolbar are two tables: 'Request' and 'Response'. Each table has columns for 'Byte offset' and 'Field'. The 'Request' table contains five rows: Function code (Constant, offset 0), Node ID (Node address, offset 1), Index (Constant, offset 2...3), Sub index (Constant, offset 4), and Checksum (Checksum, offset 5...6). The 'Response' table contains six rows: Function code (Constant, offset 0), Node ID (Node address, offset 1), Index (Constant, offset 2...3), Sub index (Constant, offset 4), Data (Data, offset 5...6), and Checksum (Checksum, offset 7...8).

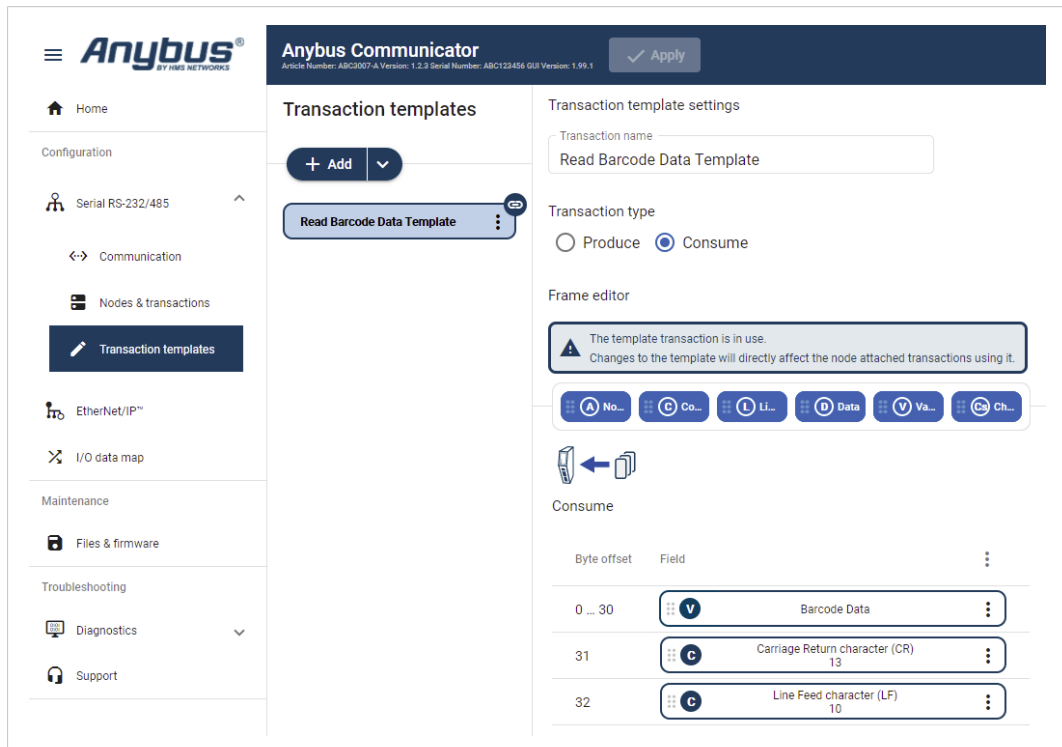
The transaction named Read parameter (0x01) consists of a number of frame fields.

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

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.

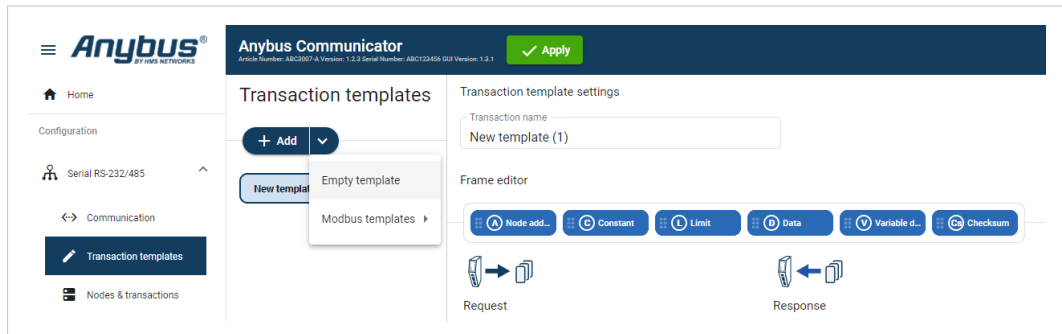
In the frame field we have added one Variable data field and two Constant fields.

### 7.6.2 Transaction Template Types

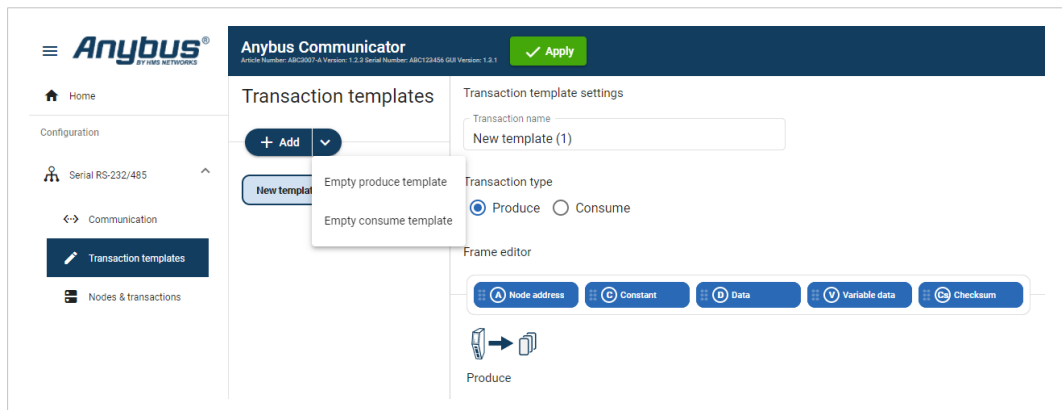
There are two types of transaction templates, Empty template and Modbus template.

#### Empty template

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



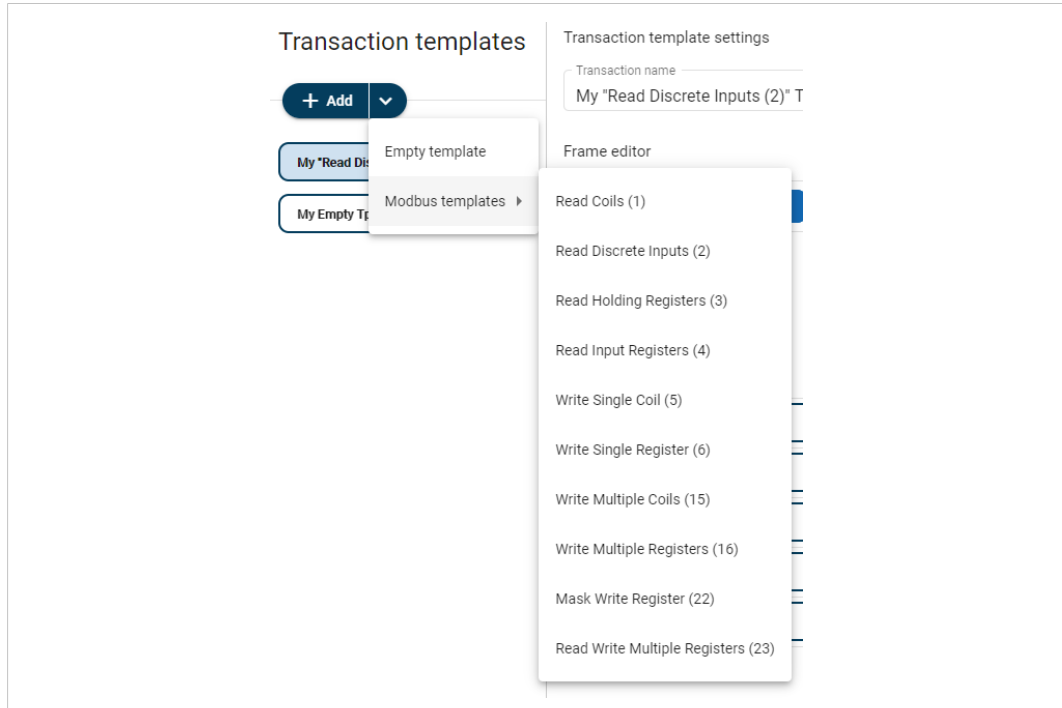
For the produce/consume transactions you select; **Empty produce template** or **Empty consume template**.



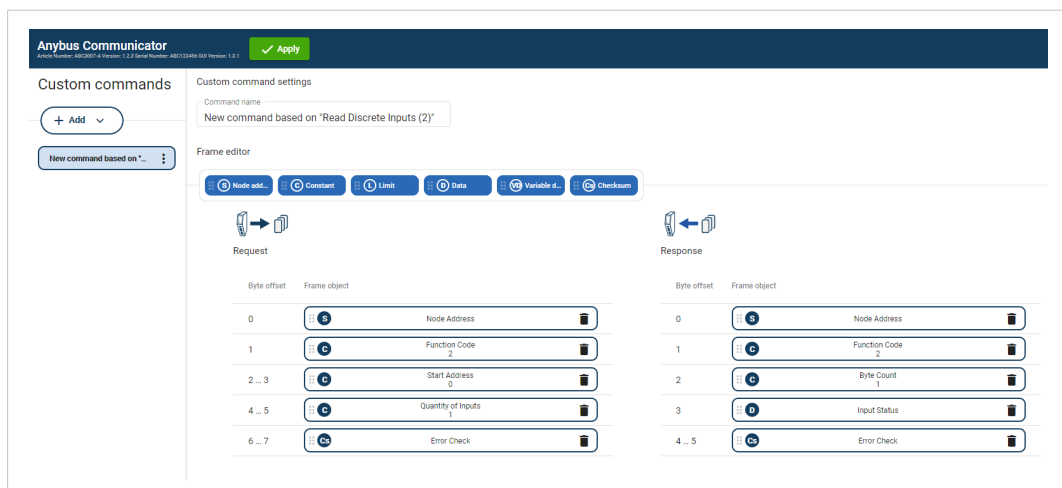
### Modbus templates

Modbus templates are available for request/response and 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.



Example, new transaction template based on the Modbus template Read Discrete Inputs:

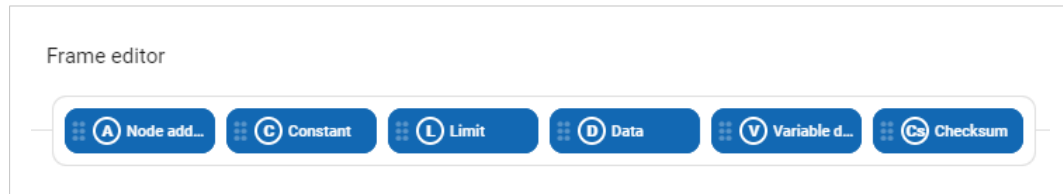


### 7.6.3 Frame Field Types

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.

The following frame fields 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



*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 sub-network. |
| 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.

## 7.7 Build Transaction Templates

### Before You Begin

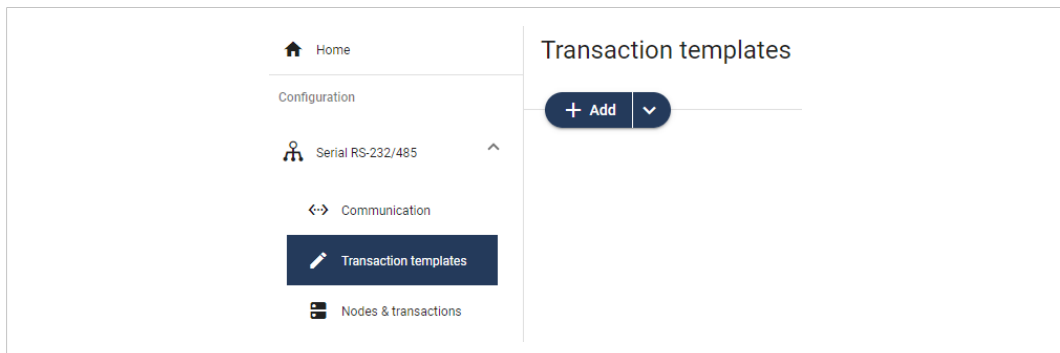
Ensure that you have applied the **Custom Request/Response** or **Custom Produce/Consume** serial protocol, refer to [Communication Serial Protocol, p. 36](#).

### 7.7.1 Add Transaction Template

#### Procedure

Add a transaction template:

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

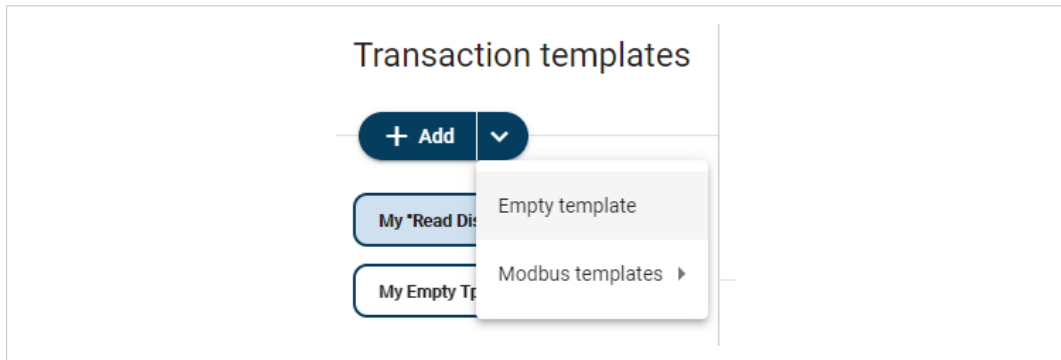


2. To select the template you want to use, click the **Add** drop-down button.

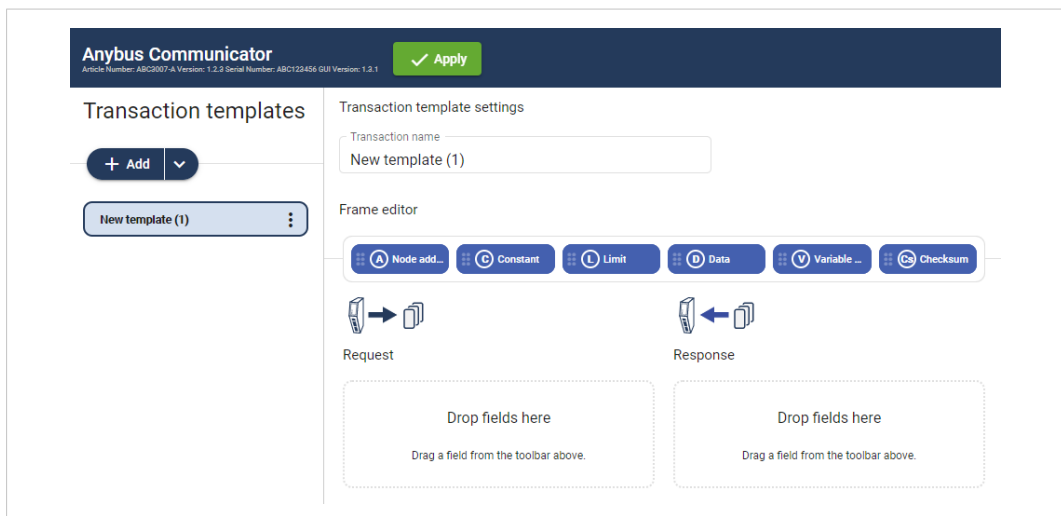
Options for the **Custom Request/Response** Protocol:



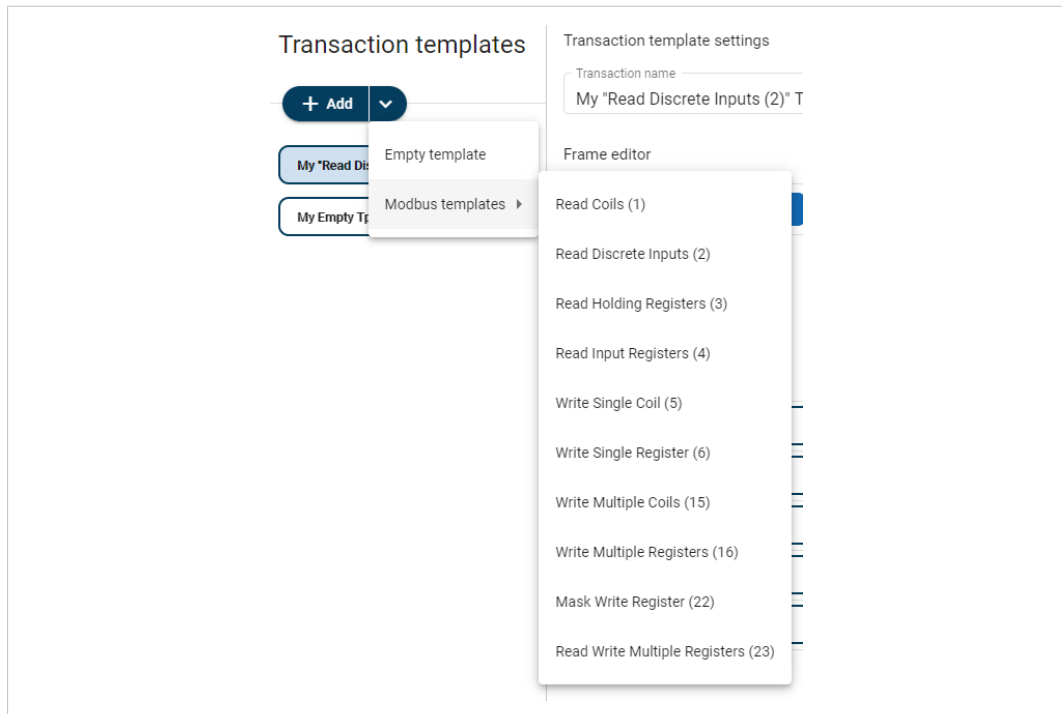
- ▶ To add a new empty template without any frame fields, select **Empty template**.



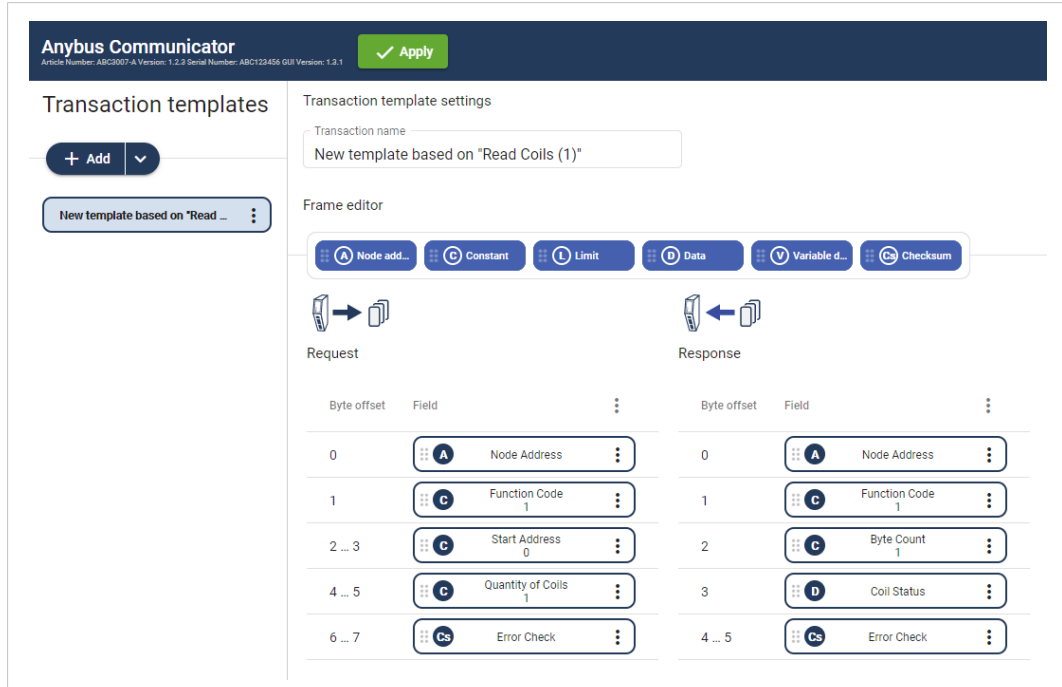
Example, a new empty request/response template is added to the transaction template list:



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



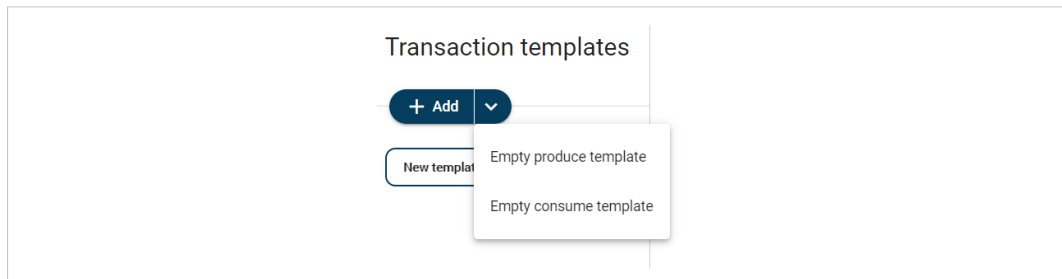
Example, a new request/response template based on “Read Coils (1)” is added to the transaction template list:



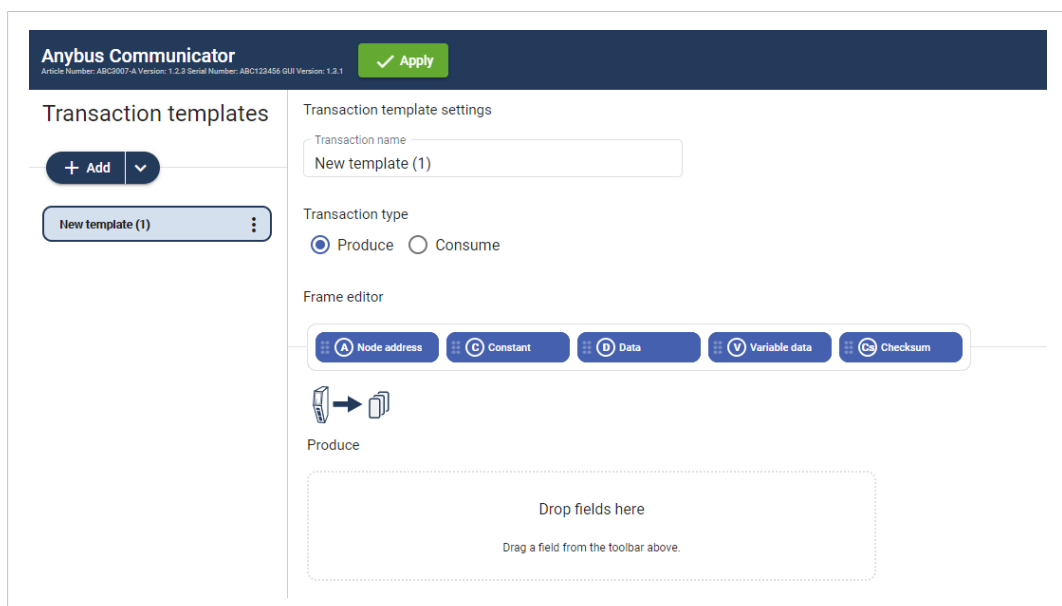
Option for the **Custom Produce/Consume** Protocol:

- ▶ Select **Empty produce template** or **Empty consume template**.

You can change the Transaction type after the transaction template is added.



Example, 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.

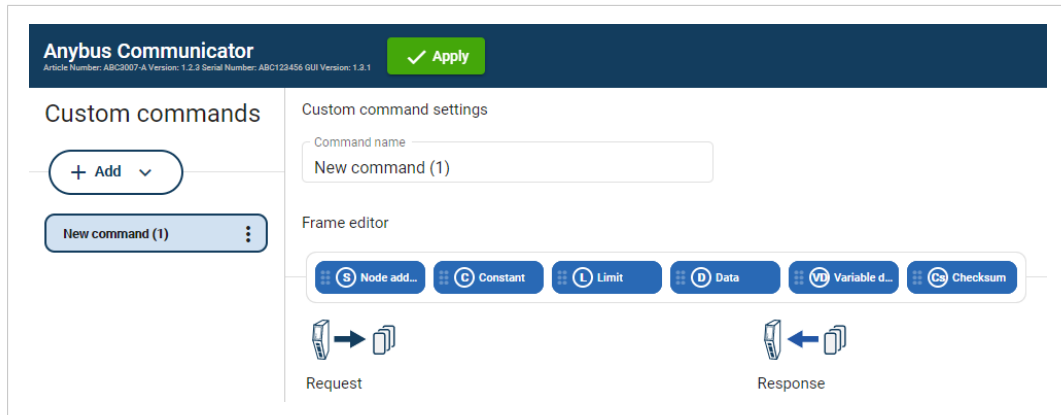
#### To Do Next

- ▶ Add frame fields to the transaction template, refer to [Add Frame Fields, p. 50](#).

## 7.7.2 Add Frame Fields

### Procedure

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

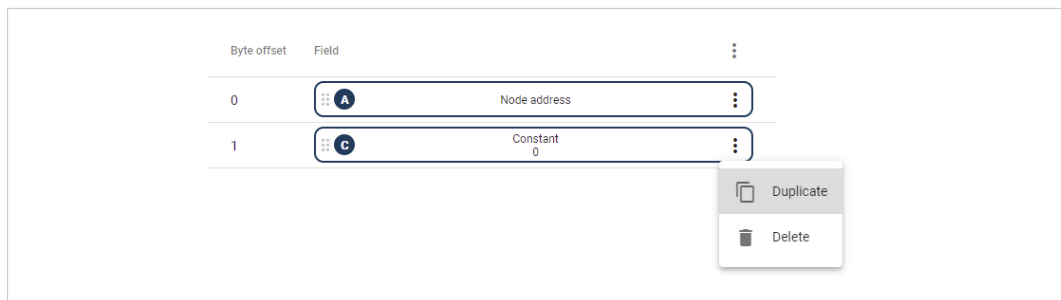


2. Build the transactions.

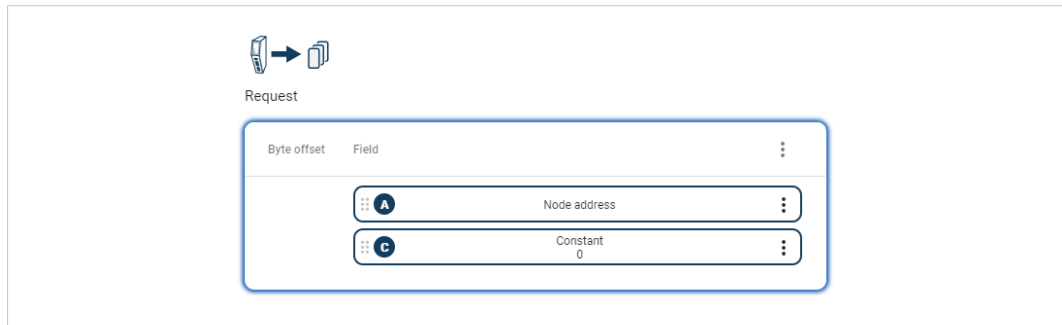
- ▶ To add frame fields: In the **Frame editor** frame fields menu, drag and drop the desired frame fields into the drag and drop fields.



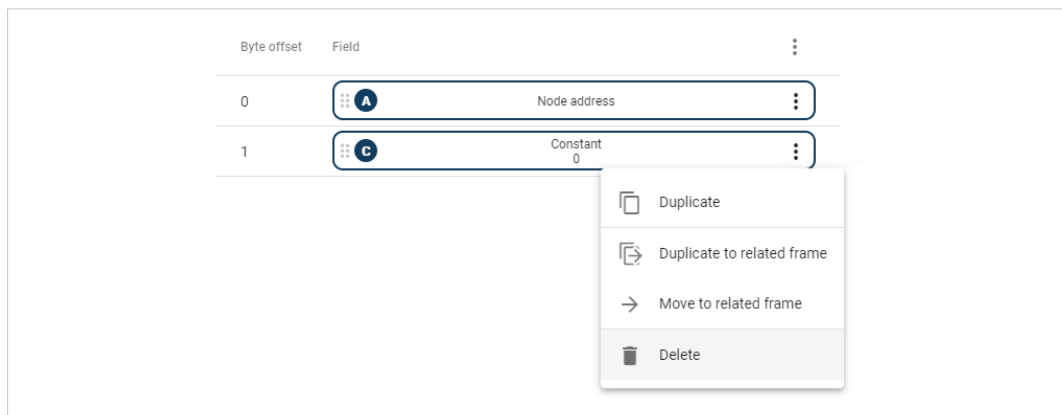
- ▶ To duplicate a frame field: On the frame field that you want to delete, click the **three dots icon** and then click **Duplicate**.



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



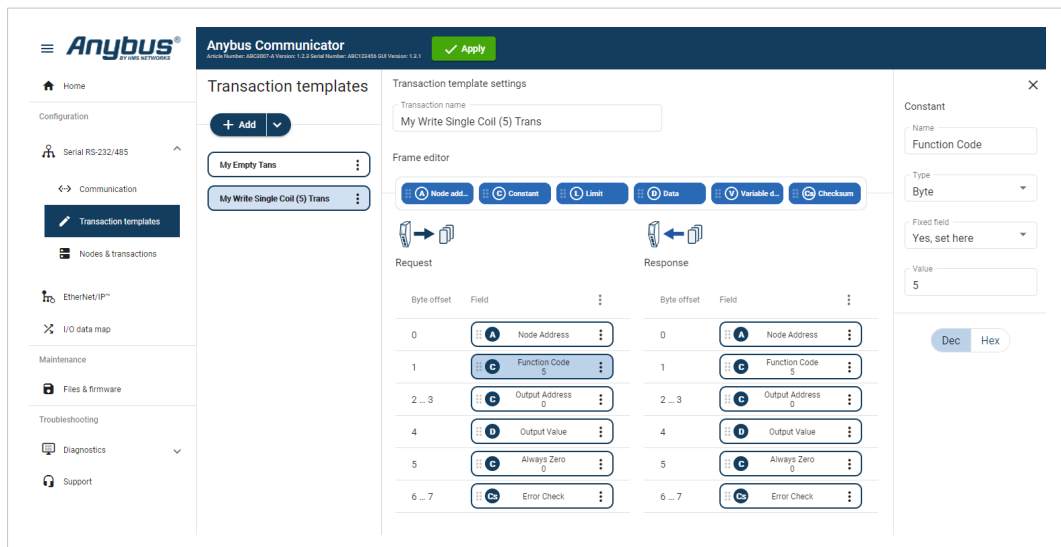
3. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

#### To Do Next


- ▶ Configure the frame field settings, refer to [Configure Frame Field Settings, p. 52](#).

## 7.7.3 Configure Frame Field Settings

### Procedure



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.

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

#### 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

- **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).
- **Endianness:** 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, refer to [Data Delimiter and Subnet Delimiter Options, p. 55](#).
- **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, refer to [Data Delimiter and Subnet Delimiter Options, p. 55](#).
- **End pattern:** Specify the value defining the end of the payload, when a delimiter is set to end pattern.
- **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.
- **Representation:** Specify how the checksum is represented. Select Binary (Default) or ASCII.

**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 fields.

**Apply configuration**

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



### 7.7.4 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

The screenshot shows the 'Transaction template settings' window. The 'Transaction name' is 'New template (1)'. The 'Frame editor' contains a warning: 'The template transaction is in use. Changes to the template will directly affect the node attached transactions using it.' Below this are buttons for 'Node add...', 'Constant', 'Limit', 'Data', 'Variable ...', and 'Checksum'. There are two sections: 'Request' and 'Response'. Each section has a table with 'Byte offset' and 'Field'. In the 'Request' section, the fields are 'Variable data' at offset 0, 'Constant 0' at offset 1, and 'Data' at offset 2. In the 'Response' section, the fields are 'Variable data' at offset 0, 'Limit 0-255' at offset 1, and 'Data' at offset 2. On the right, the 'Variable data' settings panel is open, showing 'Name: Variable data', 'Fixed field: Yes, set here', and a dropdown menu with 'Byte counter', 'End pattern', and 'None' selected. Below this, 'Subnet delimiter' is set to 'None' and 'End pattern' is '0'. At the bottom, 'Total size including delimiters' is shown as 1 byte for both EtherNet/IP and Serial subnetwork.

1. Select a desired **Variable data** object.

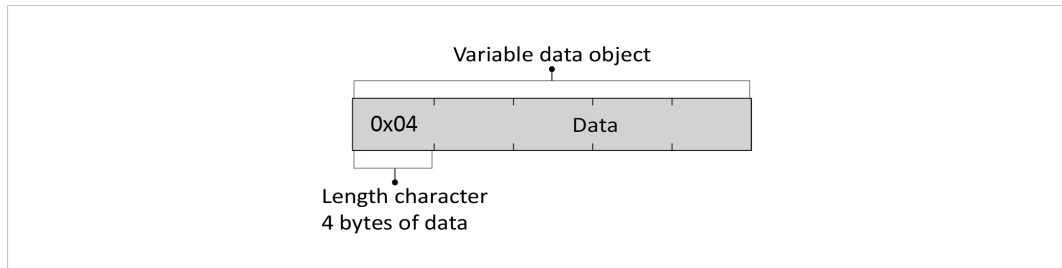
2. In the **Data delimiter** and/or **Subnet delimiter** drop down menu, select one of the following options:

– **Byte counter**

The data packet consists of a length character, indicating the length of the data section, followed by the variable data object itself.

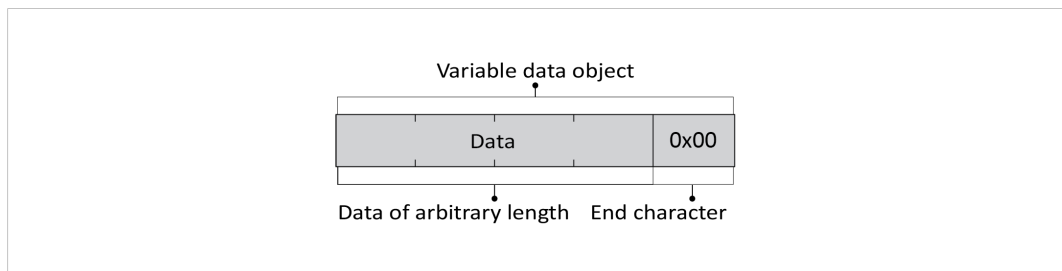
In order to copy the exact data size from the transaction message, the length of the variable data object is first identified.

In this example we have added a length character with 4 bytes of data.



– **End pattern**

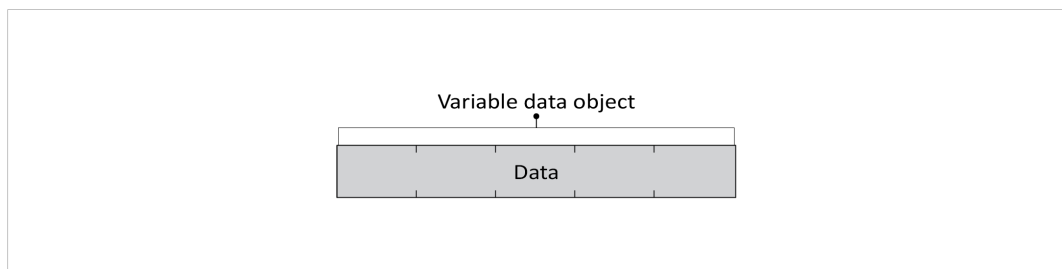
The package consists of a data section followed by an end character to indicate where the data section ends.



– **None**

The package contains only the data section.

By measuring the total length of the message, the length of the data section can be calculated.

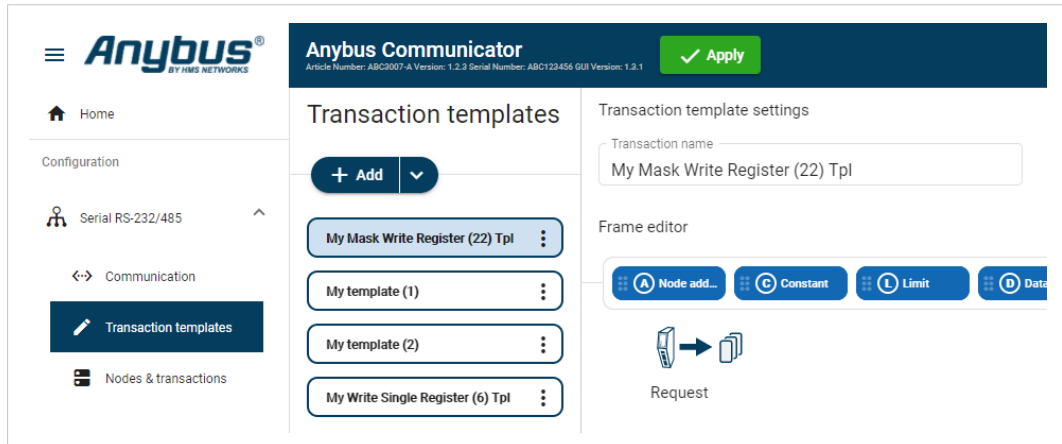


3. When a delimiter is set to End pattern: In the **End pattern** field, enter the value that will define the end of the payload.

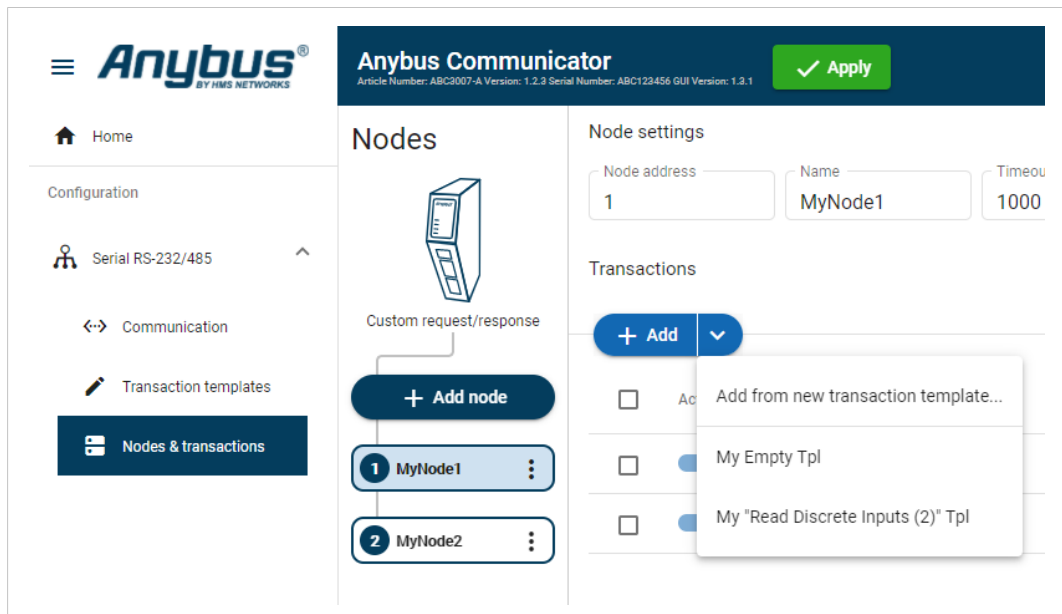
End pattern is used to define whether the delimiter is an end character or an end pattern, which depends on whether the message is forwarded from the subnetwork or sent as process data.

### 7.7.5 Store Transaction Templates

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



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



For information on how to add the transaction templates to the nodes, refer to [Transaction Settings, p. 64](#).

## 7.8 Nodes and Transactions

A node represents a single device on the serial subnetwork.

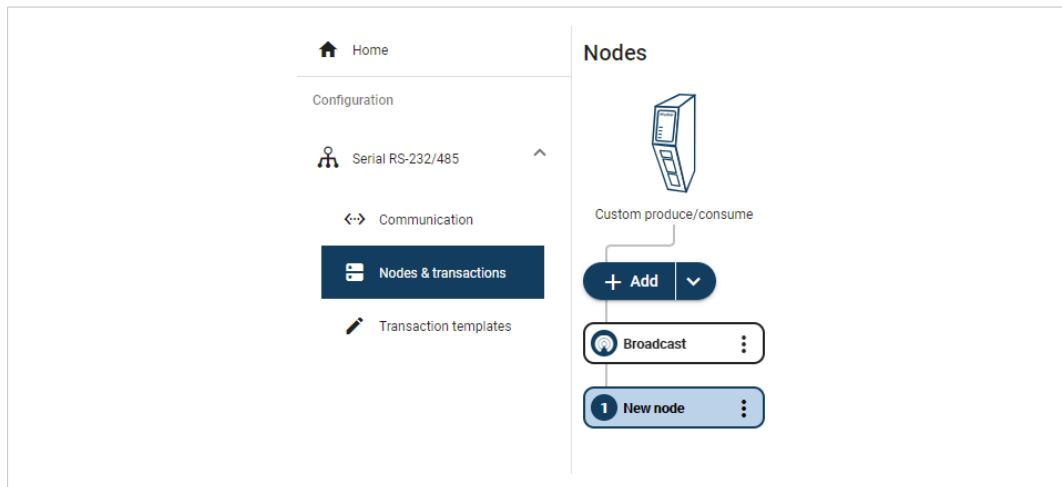
Add nodes and set up the communication between the nodes and the client.

### Before You Begin

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

### 7.8.1 Node and Broadcast Node

You can add two types of nodes, Node and 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, p. 40](#).

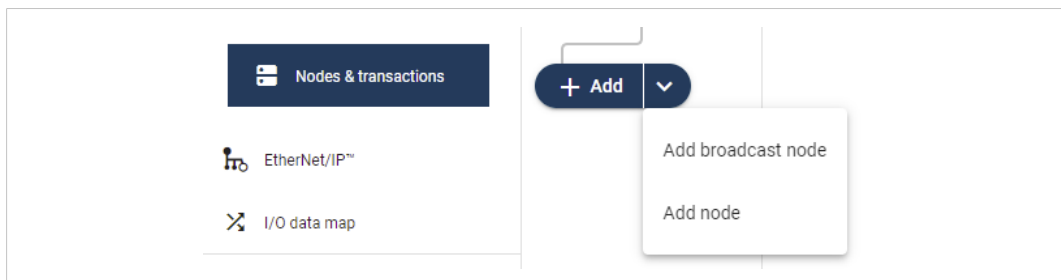
## 7.8.2 Add Node



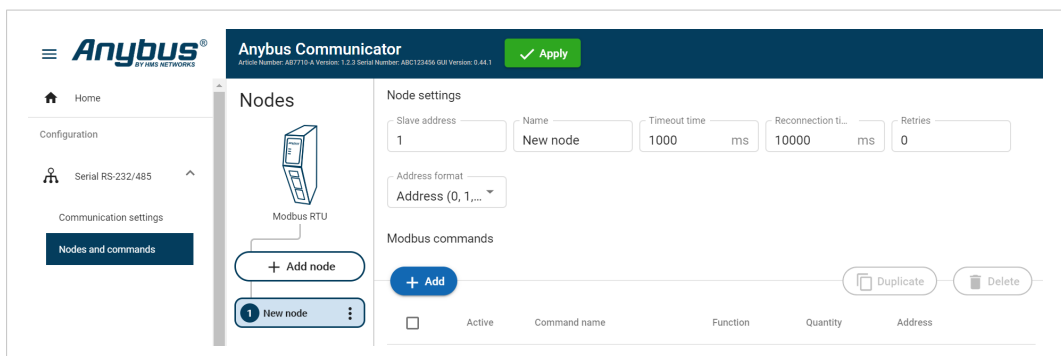
You can add one single Broadcast node.  
The maximum number of Nodes that can be added is 31.

### Procedure

1. In the web-interface left sidebar menu, click **Nodes & transactions**.
2. Click **Add node**.
3. Select **Add broadcast node** or **Add node**.



→ A new node/broadcast node is added to the nodes list.



### To Do Next

Configure the Node Settings, [Node Settings, p. 60](#).

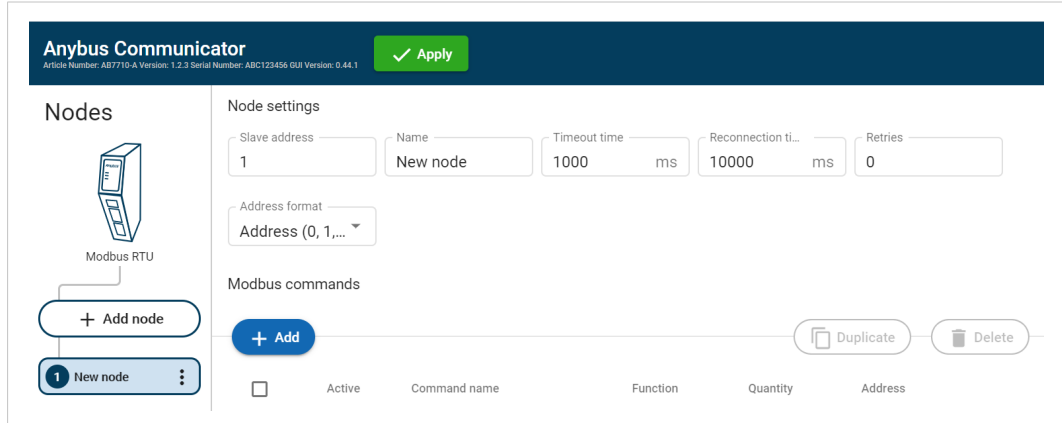
### 7.8.3 Node Settings

#### 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



1. In the nodes list, select a node to configure.

2. Configure the *Node settings*.

| Setting   | Value  | Description  |
|---|--|--|
| Node address  | 1 to 247   | <p>Node ID, also called node address, is the node's identity on the subnetwork.</p> <p>The node id is a number between 1 and 247.</p> <p>By default, the node is assigned the next available number.</p> <p>The same node id cannot be used on multiple nodes.</p>   |
| 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<br>Default value: 1000 ms.                      | <p>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.</p> <p>The maximum addition to the cycle length is only one instance of the timeout setting.</p> <p>Specify how long the Communicator should wait before sending the message again, when no response is received from the node.</p> <p>If the timeout time is exceeded, the Communicator continues to send the message until the maximum number of retries has been reached.</p> |
| Reconnection time   | Min 10 ms<br>Max 60 0000 ms<br>Default 10 000 ms                   | <p>Specify for how long the Communicator should wait before attempting to reconnect, if the node is disconnected.</p> <p>The default value is 10 000 ms.</p> <p>Reconnect time (10 ms) is not applicable for the <i>broadcast node</i>, that hold transactions destined to all nodes.</p>  |
| Retries   | 0 to 10<br>Default value: 3  | Specify the number of attempts the Communicator should make, when no response is received from the node.   |
| Address format<br><br>Available for the Modbus TCP serial protocol. | Default format: Address<br>Register<br>Modicon<br>Modicon extended | <p>Specify the address format for the node.</p> <p>Address: 0, 1, 2, ...<br/>           Register: 1, 2, 3, ...<br/>           Modicon: 00001/10001/30001/40001<br/>           Modicon extended: 000001/100001/...</p>  |

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

Add Transactions, [Add Transactions, p. 62](#).

## 7.8.4 Add Transactions

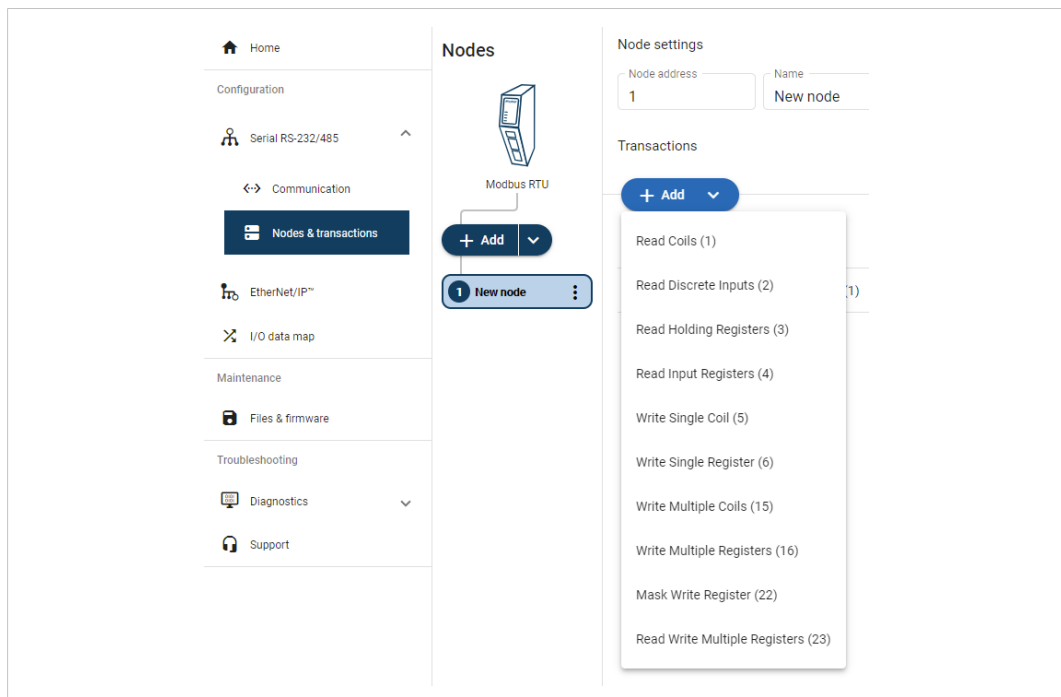


The maximum number of transactions that can be added to a node is 150.

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

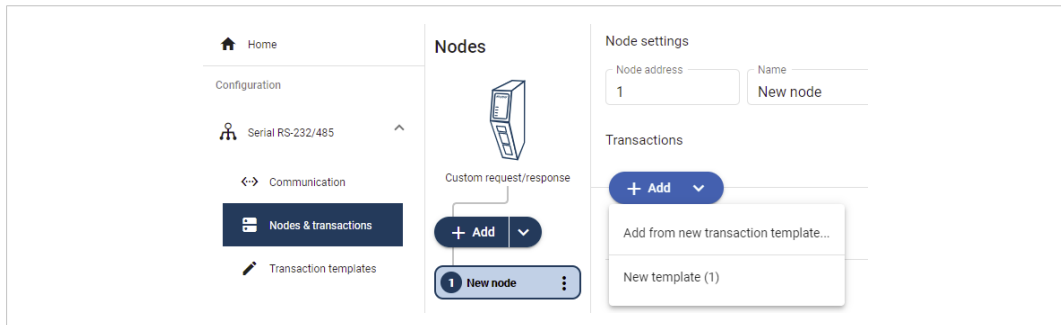
### When using the Modbus RTU Serial Protocol

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





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

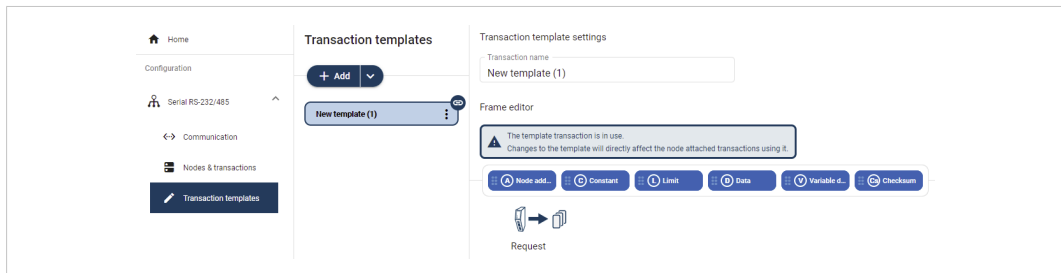


- ▶ Click **Add** and select **Add from new transaction template**.

→ You are redirected to the **Transaction template** page.

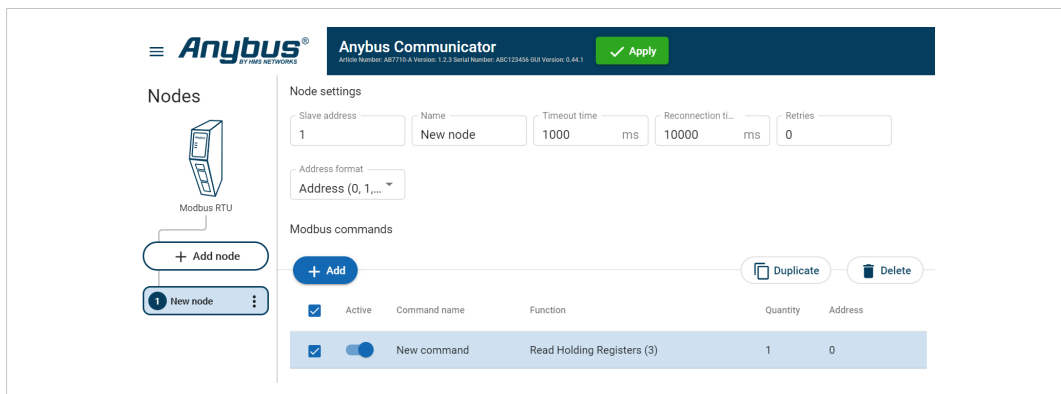
A new empty template is added to the Transaction templates list.

You need to build the transactions before you can use the template, refer to [Build Transaction Templates, p. 46](#).



- ▶ 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.




**To Do Next**

Configure the Transactions settings, [Transaction Settings, p. 64](#).

## 7.8.5 Transaction Settings

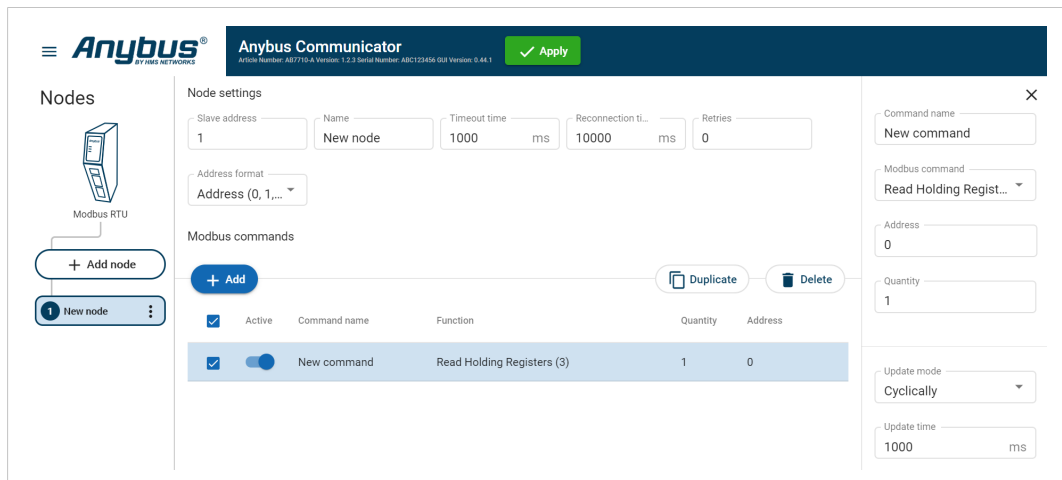
### Before You Begin

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

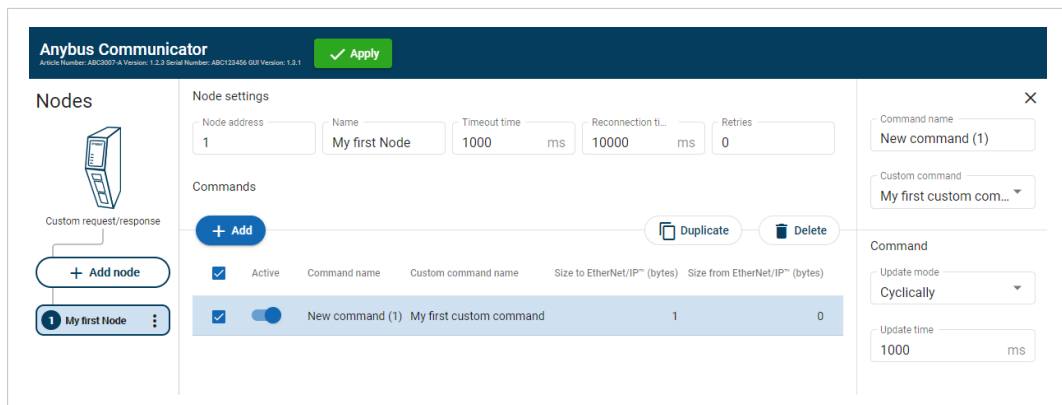
For Modbus transaction reference guide, refer to [Modbus Transactions, p. 121](#).

### Procedure

Modbus RTU Protocol:



Custom Request/Response Protocol:



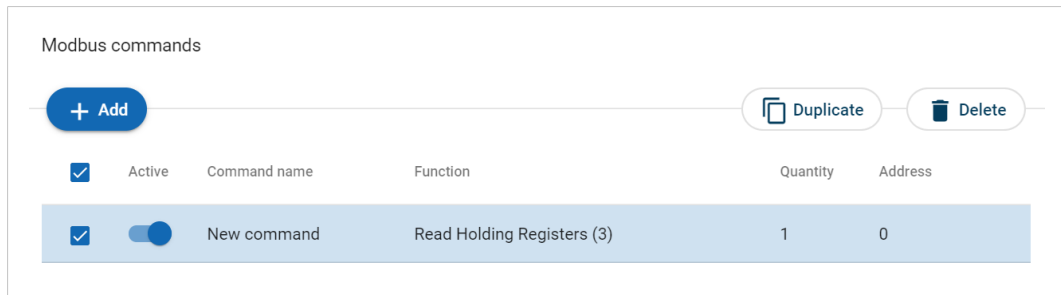
1. In the nodes list, select a node to configure.
2. In the transactions list, select a transaction to configure.
  - The transaction sidebar opens, on the right side of the screen.
3. Enter a transaction name.
  - By default, the node is assigned the name New transaction.
4. Select a transaction type 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 Transaction 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 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.<br><br>The address acts as an address to the data position, where the data is read from or written to.<br><br>Modbus holding register addresses starts at 0.<br>Modbus address 0 = Register 1   |
| Write quantity   | Read Write Multiple Registers (23)<br>1 to 123   | Specifies the quantity of registers/coils to write in the write data field. Appear when Modbus transaction Read Write Multiple Registers (23) is selected.  |
| Quantity         | Read Holding Registers (3)<br>Read Input Registers (4)<br>1 to 125<br><br>Write Multiple Coils (15)<br>1 to 1968<br><br>Write Multiple Registers (16)<br>1 to 123<br><br>Read Coils (1)<br>Read Discrete Inputs (2)<br>1 to 2000 | The Quantity parameter appear when you select a Modbus transaction that can address more than one data object.<br><br>Example when Quantity is set: For the Modbus Transaction <i>Read Input Registers (4)</i> you need to set the Quantity in order to define the array of data.<br><br>Example when no Quantity is set: For the Modbus Transaction <i>Write Single Coil (5)</i> 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.<br><br>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    | 0 to 255   | The maximum payload length of the variable data field.  |
| Update mode      | Cyclically<br>On data change<br>Single shot<br>Change of state on trigger  | Specify when a transaction shall be sent to the server.<br>The transaction is issued cyclically, at the interval specified in the Update time parameter.<br><br>Cyclically: The transaction is sent cyclically, at the interval specified in the Update time parameter.<br><br>On data change: The transaction is sent when the data is changed.<br><br>Single shot: The transaction is issued once at start up.<br><br>Change of state on trigger: The transaction is triggered when the content of a specified byte changes. In the I/O data map, the node will be marked with a flash icon. In the I/O data map you can also configure the area map and the trigger byte address. Refer to <a href="#">Trigger Byte, p. 74</a> . |
| 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.<br><br>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 data map 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 data map is incremented each time this transaction fails.   |

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

## 7.8.6 Activate/Deactivate Transaction

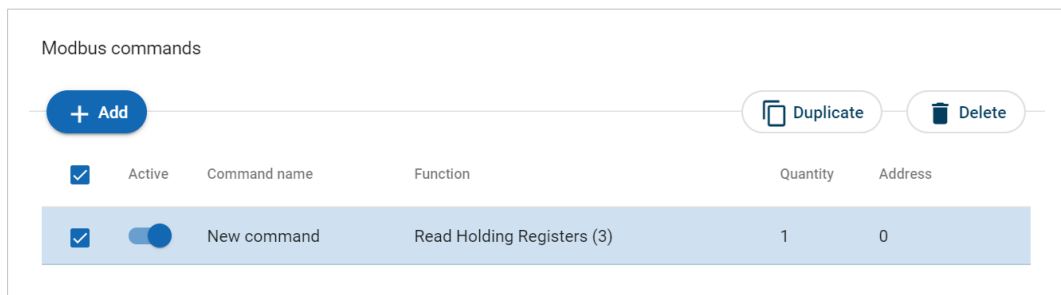


The transaction default status is **Active**.

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

## 7.8.7 Duplicate Transaction

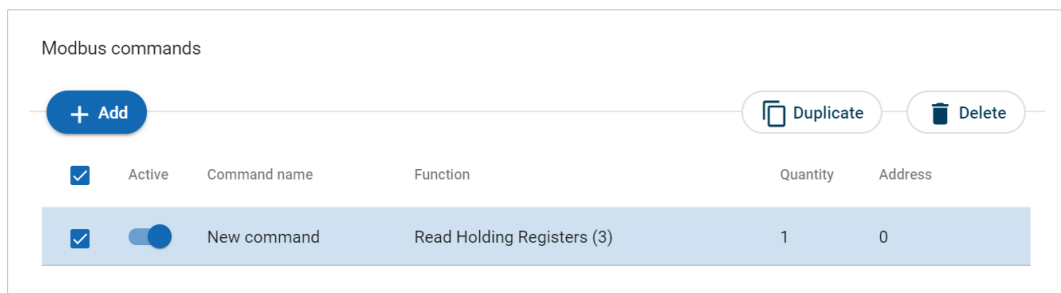
When you duplicate a transaction, all settings are preserved.



To duplicate:

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

## 7.8.8 Delete Transaction

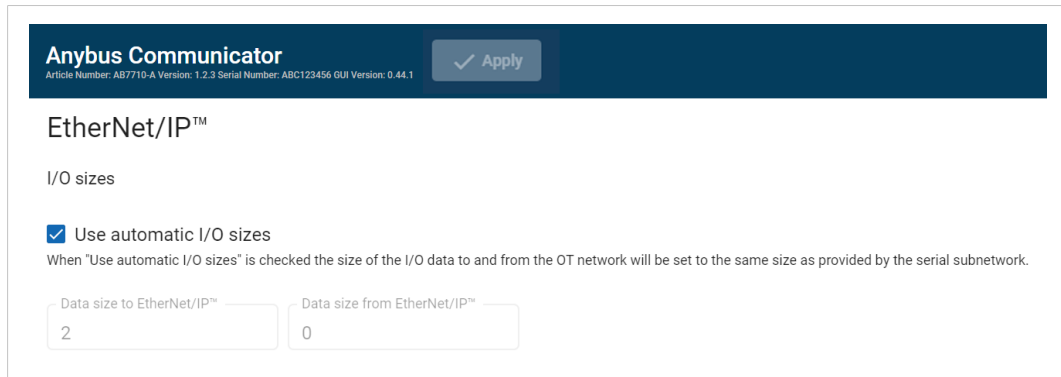


1. To delete:
  - One transaction, select the transaction and click **Delete**.
  - Multiple transactions, select the checkbox in front of each transaction and click **Delete**.
2. To confirm, click **Yes**.

## 7.9 High Level Network Settings

Configure the EtherNet/IP network settings.

### 7.9.1 To Use Automatic I/O Sizes



**Anybus Communicator**  
Article Number: AB7710-A Version: 1.2.3 Serial Number: ABC123456 GUI Version: 0.44.1

EtherNet/IP™

I/O sizes

Use automatic I/O sizes  
When "Use automatic I/O sizes" is checked the size of the I/O data to and from the OT network will be set to the same size as provided by the serial subnetwork.

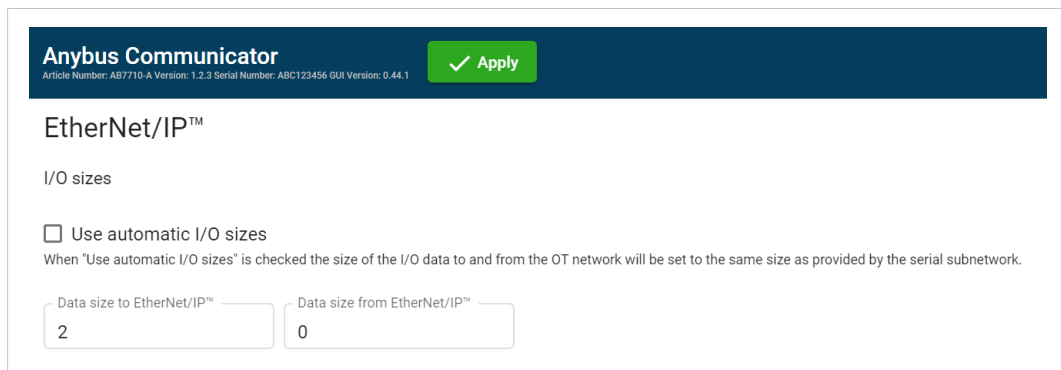
Data size to EtherNet/IP™: 2      Data size from EtherNet/IP™: 0

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

The size of the input data, *Data Size to EtherNet/IP™*, and the output data, *Data Size from EtherNet/IP™*, is determined by the subnetwork configuration.

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

### 7.9.2 To Configure I/O Sizes Manually



**Anybus Communicator**  
Article Number: AB7710-A Version: 1.2.3 Serial Number: ABC123456 GUI Version: 0.44.1

EtherNet/IP™

I/O sizes

Use automatic I/O sizes  
When "Use automatic I/O sizes" is checked the size of the I/O data to and from the OT network will be set to the same size as provided by the serial subnetwork.

Data size to EtherNet/IP™: 2      Data size from EtherNet/IP™: 0

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

### 7.9.3 To Use DHCP Server

**Anybus Communicator**  
Article Number: AB7710-A Version: 1.2.3 Serial Number: ABC123456 GUI Version: 0.44.1

✓ Apply

IP Settings

DHCP enabled

IP address: 192.168.0.111    Subnet mask: 255.255.255.0    Gateway address: 192.168.0.1

Primary DNS: 0.0.0.0    Secondary DNS: 0.0.0.0

By default, the IP settings are provided by the high level network DHCP server.

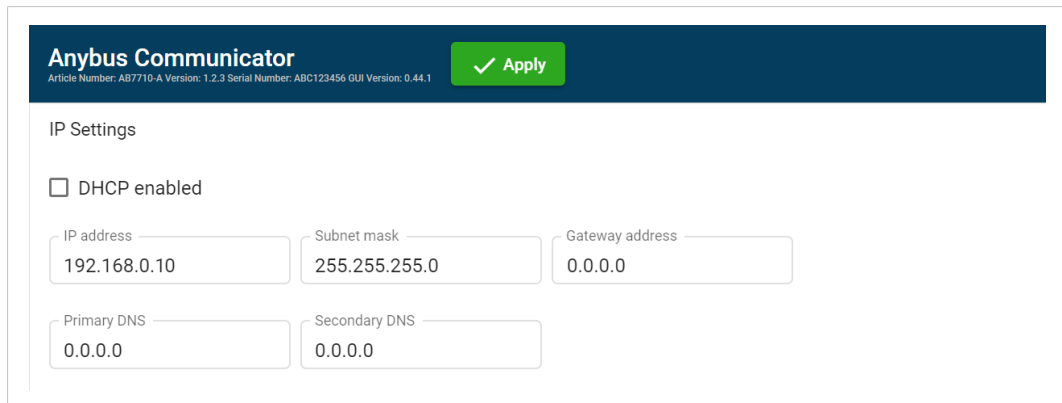
The **DHCP enabled** checkbox is selected.

#### Default Communicator IP Settings

The Communicator comes with the following factory default IP settings:

| Setting                      | Default value                             |
|------------------------------|---|
| Web configuration IP address | 192.168.0.10                              |
| Subnet mask                  | 255.255.255.0                             |
| Gateway address              | There is no default Gateway address.      |
| Primary DNS server           | There is no default Primary DNS server.   |
| Secondary DNS server         | There is no default Secondary DNS server. |
| DHCP                         | Enabled                                   |
| Host name                    | There is no default Host name.            |

### 7.9.4 To Configure IP Settings Manually



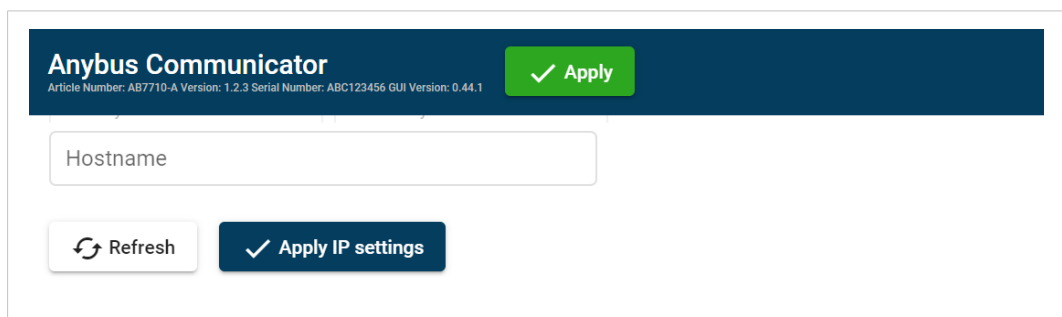
1. Deselect the **DHCP enabled** checkbox.
2. Configure the IP settings.

| Setting         | Description   |
|-----------------|---|
| IP address      | The EtherNet/IP network IP address in IPv4 dot-decimal notation   |
| Subnet mask     | The EtherNet/IP network Subnet mask in IPv4 dot-decimal notation.   |
| Gateway address | The EtherNet/IP network Gateway address in IPv4 dot-decimal notation. If there is no gateway available, set the Gateway address to: 0.0.0.0 |
| Primary DNS     | The EtherNet/IP network Primary DNS in IPv4 dot-decimal notation.   |
| Secondary DNS   | The EtherNet/IP network Secondary DNS in IPv4 dot-decimal notation.   |

If you change a value and click **Refresh**, the value is reset to the last applied value.

3. To apply the settings, click **Apply IP Settings**.

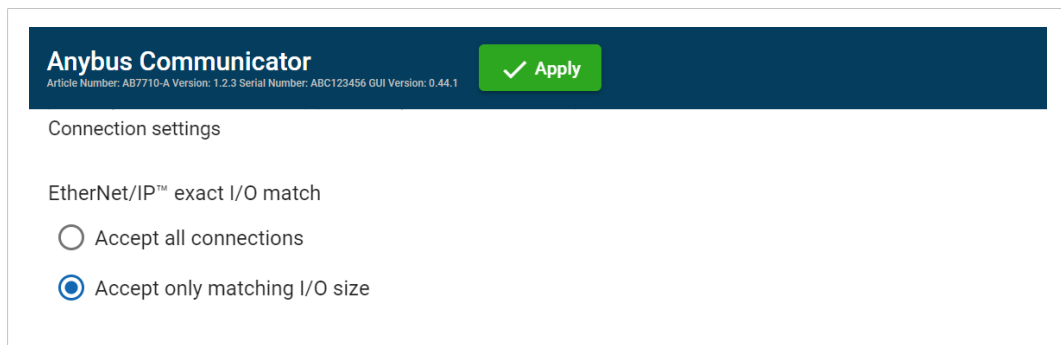
### 7.9.5 Naming the Host



You can label the Communicator.

- The maximum allowed length of the *Hostname* is 64 characters.
- No symbol characters, punctuation characters, or whitespace are permitted.
- Write the *Hostname* as one single word.

## 7.9.6 Connection Settings



When the EtherNet/IP Client (PLC) opens a connection to the Communicator, it specifies an I/O data size.

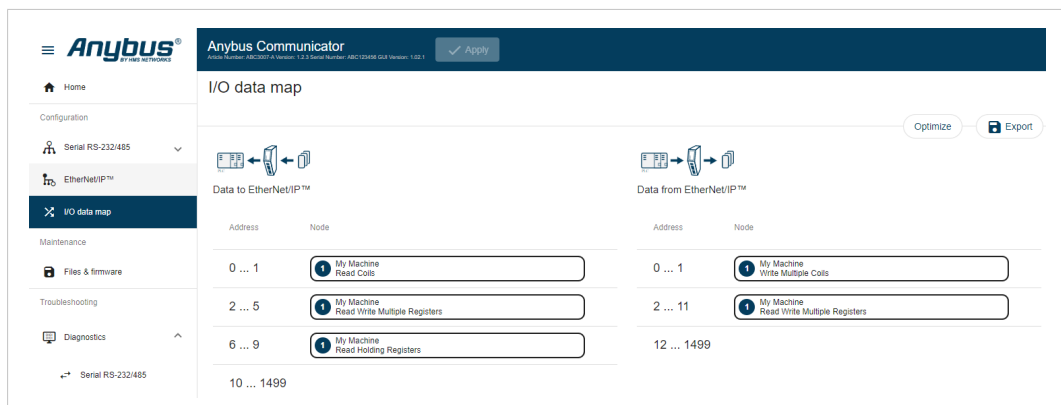
By default the Communicator is set to **Accept Only Matching I/O Sizes**.

The connections must match the I/O size configured on the EtherNet/IP page, refer to [To Use Automatic I/O Sizes, p. 67](#) and [To Configure I/O Sizes Manually, p. 67](#).

You can change to **Accept All Connections**.

The Communicator will accept all connections with an I/O size that is equal to or smaller than the configured I/O size in the Communicator.

## 7.10 I/O Data Map



On the **I/O data map** 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. Refer to [To Configure I/O Sizes Manually, p. 67](#).

There are three areas: Data from EtherNet/IP, Data to EtherNet/IP and General Areas. Refer to [Map Area, p. 73](#).



### 7.10.1 Optimize the I/O Data Map

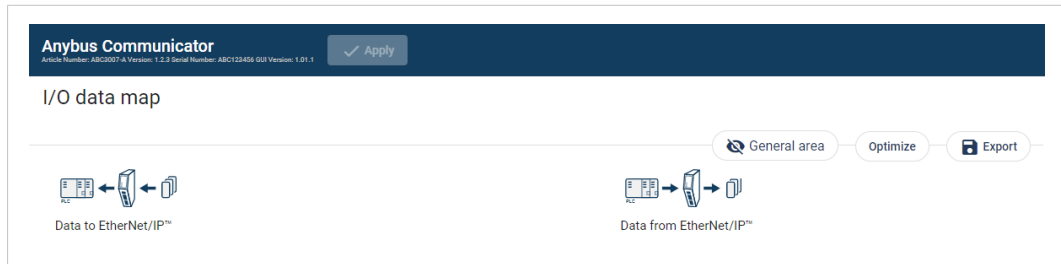
The optimize function is used to automatically remove gaps between the mapping.



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.



*If you optimize the I/O data map, the current I/O data map will be overwritten.*




To optimize the map:

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


### 7.10.2 Map Area Transactions Order

**I/O data map**

  
Data from EtherNet/IP™

Data exchange control


| Address    | Node                            |
|------------|---------------------------------|
| 0          | My Node<br>Transaction 1 Data 1 |
| 1 ... 1499 |                                 |

  
Data to EtherNet/IP™

Live list


| Address    | Node                            |
|------------|---------------------------------|
| 0          | My Node<br>Transaction 1 Data 3 |
| 0          | My Node<br>Transaction 1 Data 2 |
| 1          |                                 |
| 2 ... 1499 |                                 |

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



  
Data from EtherNet/IP™

Data exchange control

| Address    | Node                                |
|------------|-------------------------------------|
| 0          | My Node<br>New transaction (1) Data |
| 1 ... 1499 |                                     |

  
Data to EtherNet/IP™

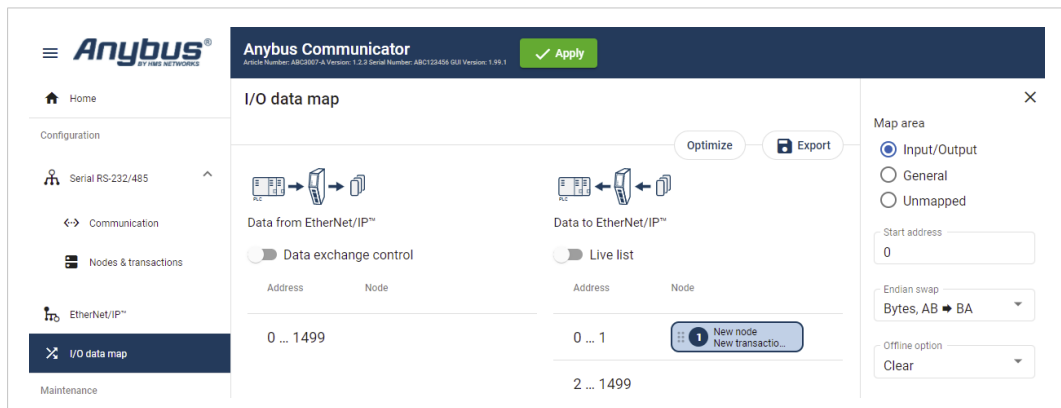
Live list

| Address    | Node   |
|------------|--|
| 0          | My Node<br>New transaction (1) Data1  |
| 0          | My Node<br>New transaction (1) Data2  |
| 1 ... 1499 |  |

Transactions can not share the same I/O are.

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

### 7.10.3 Map Area



#### Map area options

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

Select one of the following Map area options:

- **Input/Output:** The transaction data is sent/received 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.

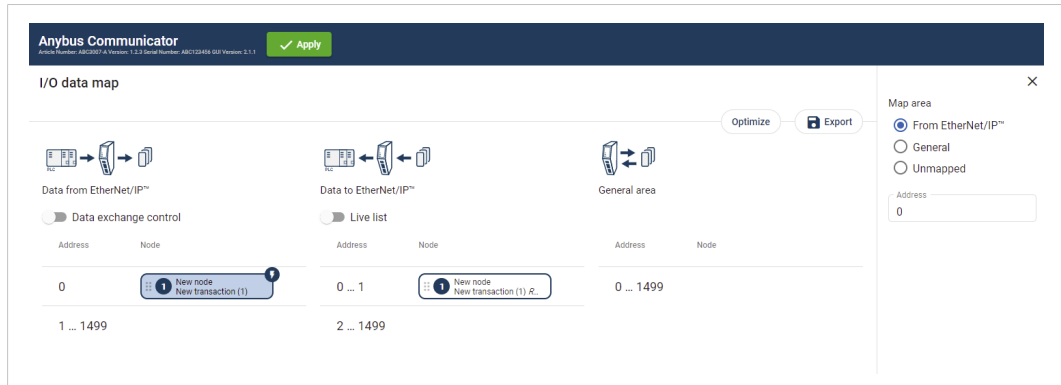
### 7.10.4 Trigger Byte

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 EtherNet/IP** area or the **General** area.

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



#### 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 data map** 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 EtherNet/IP:** The trigger byte is stored in the I/O data map Data from EtherNet/IP area.
- **General:** The trigger byte is stored in the I/O data map General area.
- **Unmapped:** The transaction data is not used.

##### Address

- Enter an Address, the location in the specified Map area (From EtherNet/IP or General) where the trigger byte will be saved.

Value: 0 (default) to 1499

### 7.10.5 Endian Swap

By default EtherNet/IP uses the little-endian format.

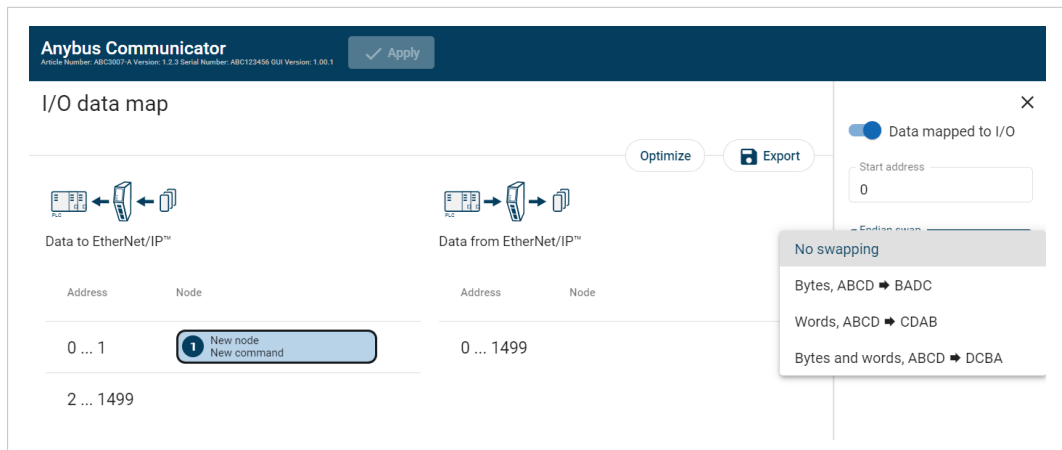
#### Big-endian

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

#### Little-endian

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

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



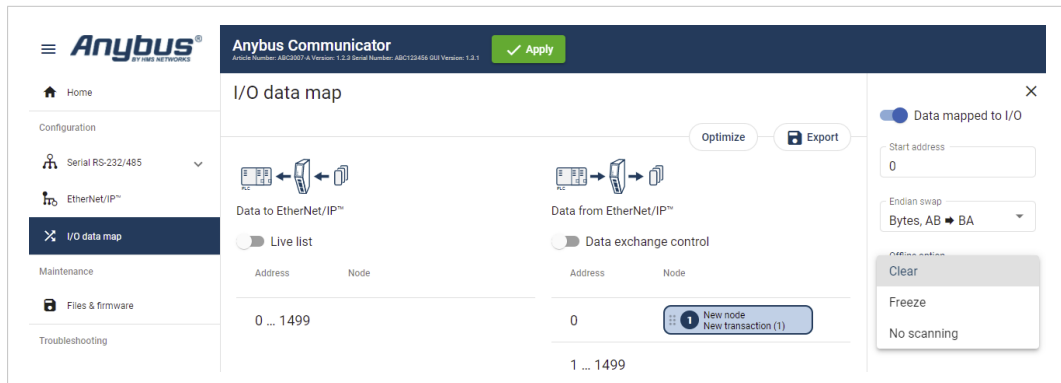
To reverse the byte order:

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

| Setting            | Description  |
|--------------------|--|
| No swapping        | Default setting<br>No swapping is performed on the data. |
| Byte swap          | Swap 2 bytes<br>A B C D becomes B A D C                  |
| Word swap          | Swap 4 bytes<br>A B C D becomes C D A B                  |
| Byte and Word swap | 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.

## 7.10.6 Offline Option



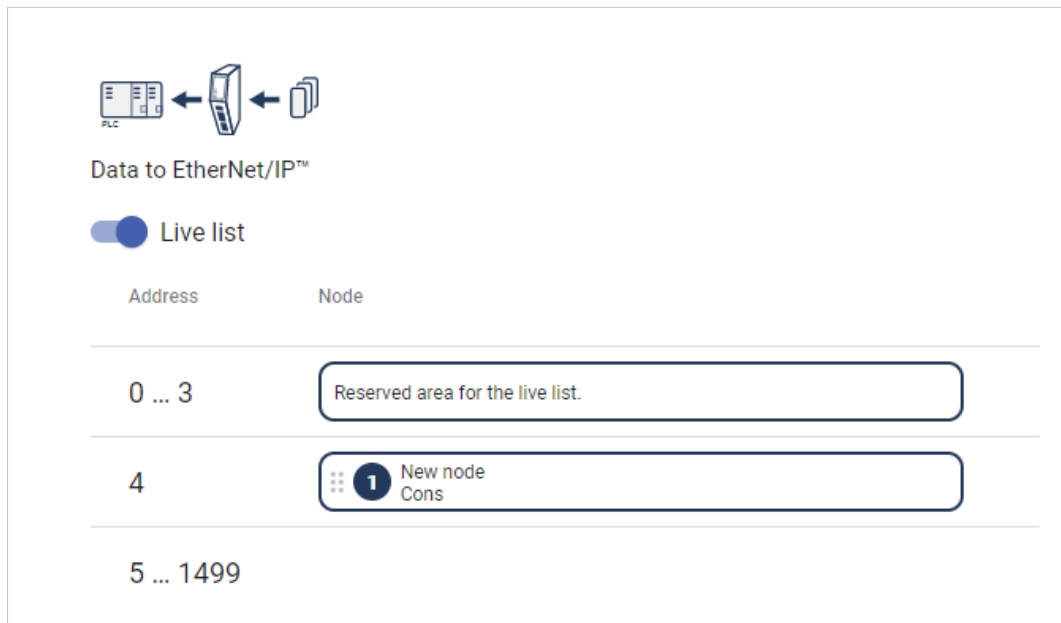
Offline mode is used to define what data to send if the network connection or connection with a specific node is lost.

You must specify the offline mode to use for each transaction on the I/O data map.

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.

### 7.10.7 Live List



By default Live list is disabled.

When Live list is enabled, the first four bytes of process data on the EtherNet/IP network contain the live list.

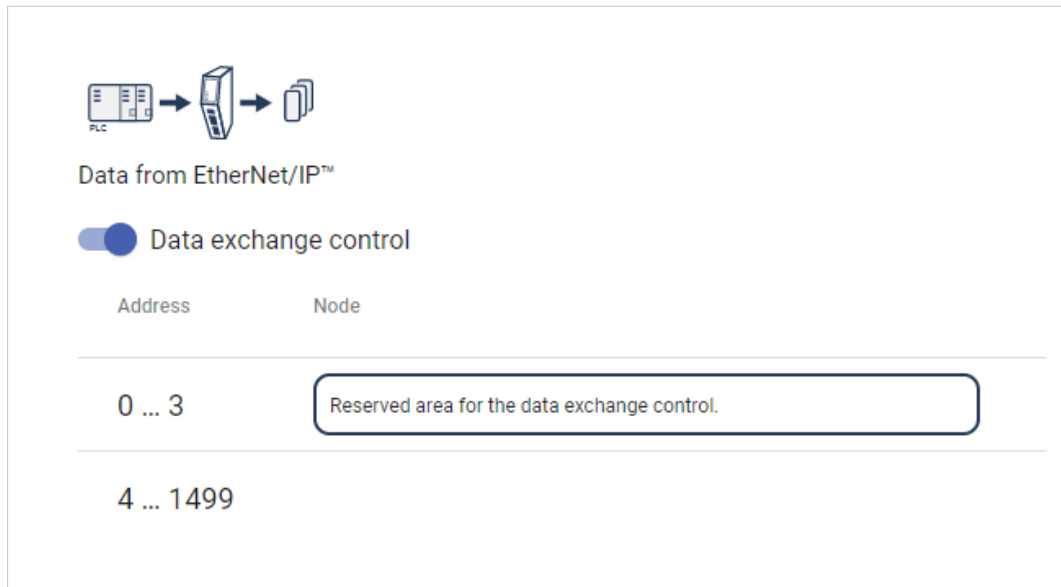
The Live list holds 32 bits.

Each bit in the Live list can hold the status for a total of 32 nodes connected to the Communicator.

The bit is 0 when the bit does not correspond to the configured node. For example, this occurs when the number of configured nodes is less than 32.

Each bit is 1 when the corresponding node is online.

### 7.10.8 Data Exchange Control



By default Data exchange control is disabled.

When Data exchange control is enabled, the first four bytes of process data on the EtherNet/IP 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 nodes 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.

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



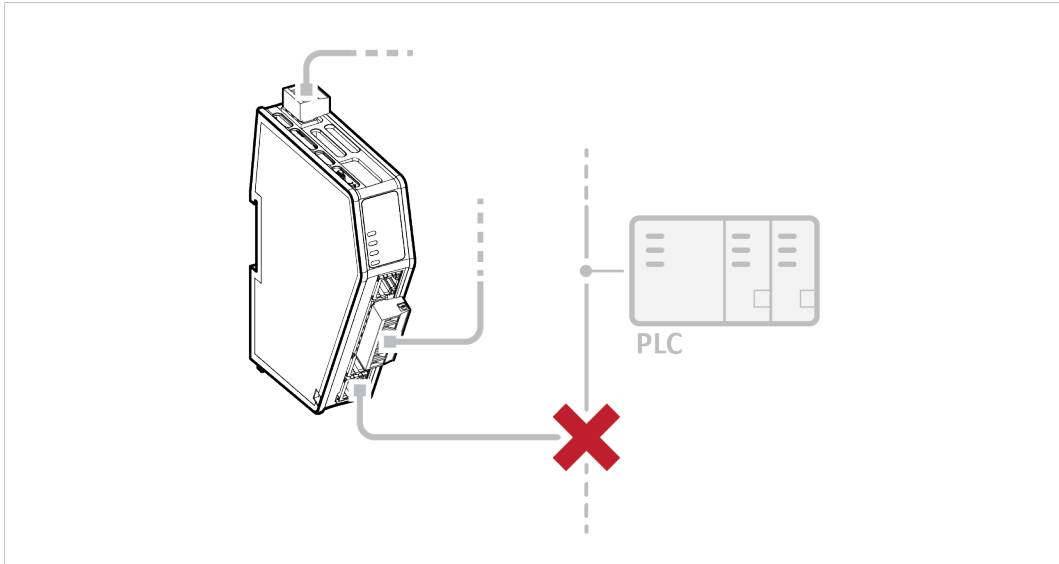
## 7.11 Apply Configuration

### Before You Begin



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

### Disconnect the Communicator from the EtherNet/IP™ network



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

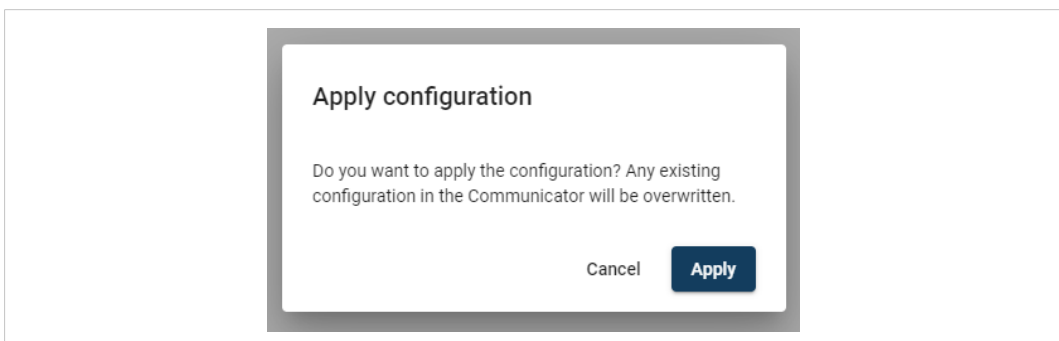
### Procedure

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

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



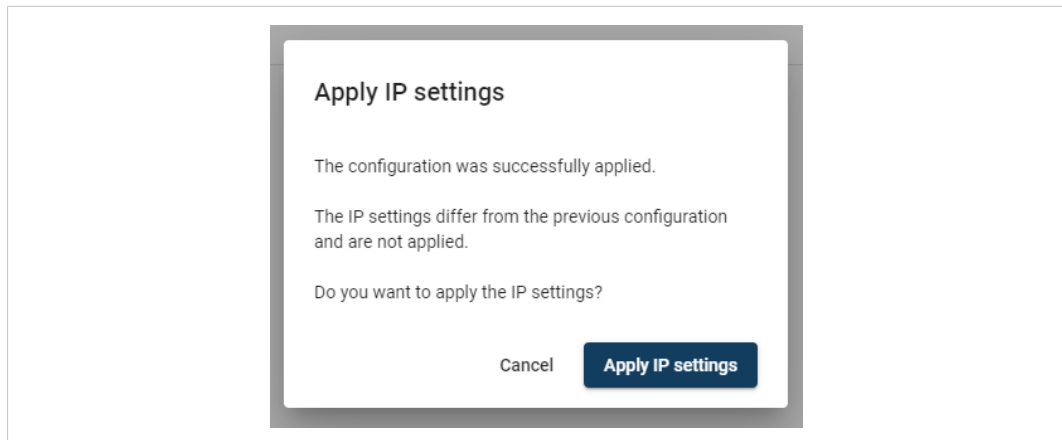
2. To confirm download, click **Apply**.



→ The configured settings are downloaded and applied to the system.

If you have made changes to the IP settings you are prompted to apply these settings.

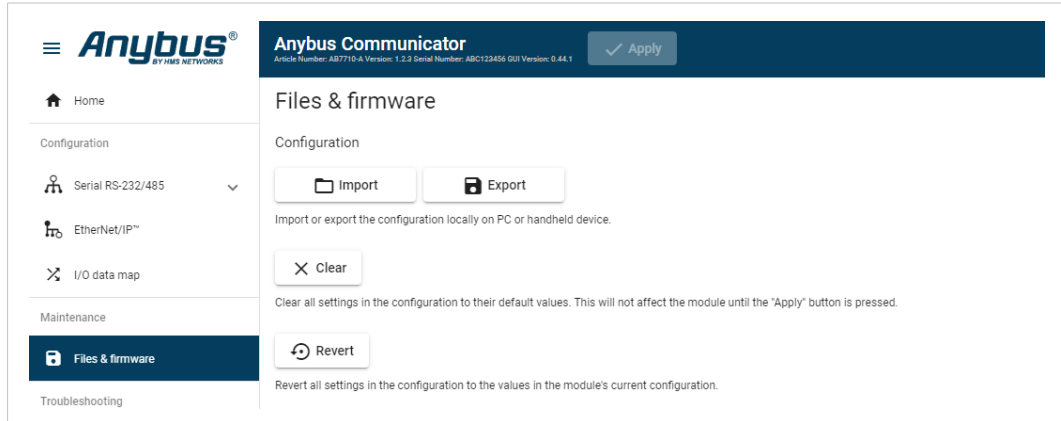
3. To apply the IP settings, click **Apply IP settings**.



## 7.12 Use an Existing Configuration

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

### Procedure



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 conf 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.

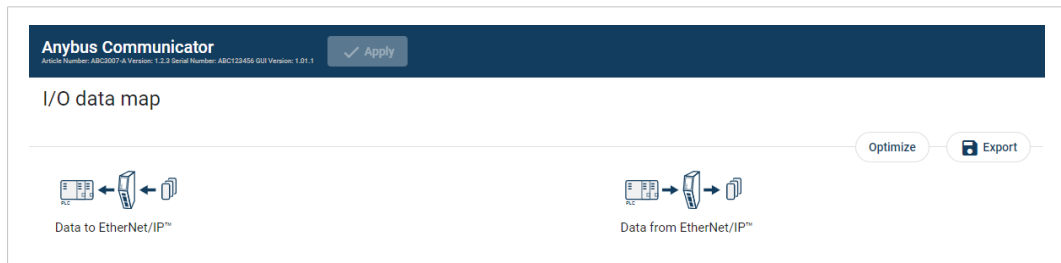
To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

## 8 PLC Configuration

### 8.1 Export I/O Data Map

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.

In the Communicator built-in web-interface:



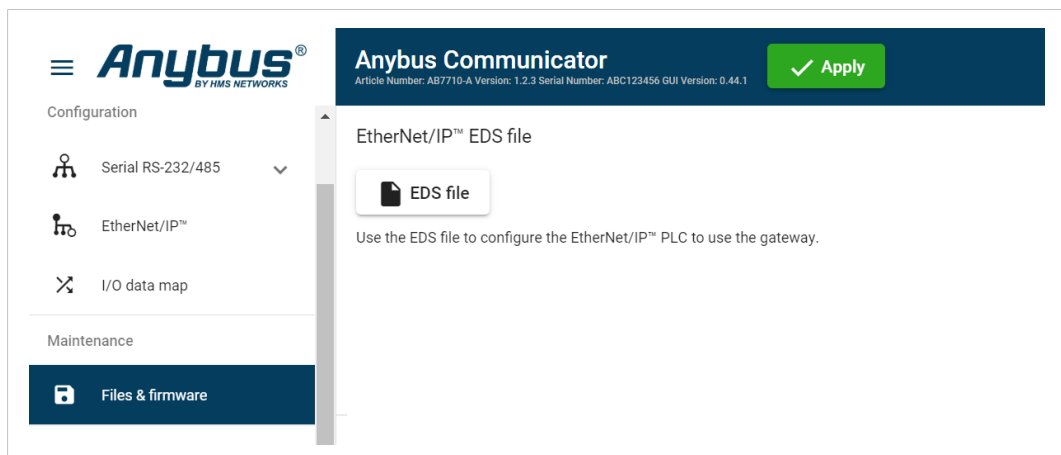
On the I/O data map page you can export the I/O data map in an Excel XLS file, where all the nodes and transactions are listed.

To export the I/O data map:

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

### 8.2 Export Product EDS File

Option if the PLC program requires a product file, EDS (Electronic Data Sheet) file, describing how the Communicator can be used on the high level network.



You find the EtherNet/IP™ EDS file on the Communicator built-in web interface EtherNet/IP™ page, Files & firmware page and on the Support page.

To export the EDS file:

1. Click **EDS file**.  
→ The EDS file is downloaded to your PC.

## 8.3 CIP Objects

Supported Common Industrial Protocol (CIP) objects.

| Object name               | Class | Description                       |
|---------------------------|-------|-----------------------------------|
| Identity object           | 0x01  | The identification object         |
| Message router object     | 0x02  | Message router                    |
| Assembly object           | 0x04  | Assembly object                   |
| Connection manager object | 0x06  | Connection manager object         |
| DLR object                | 0x47  | Device level ring object          |
| QoS object                | 0x48  | Quality service object            |
| TCP/IP Interface object   | 0xF5  | Handles TCP/IP configuration      |
| EtherNet/IP Link object   | 0xF6  | Handles EtherNet/IP configuration |

## 9 Verify Operation

### Before You Begin

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

Refer to [Installation, p. 16](#).

### 9.1 Communicator Status Monitor

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

The screenshot displays the Anybus Communicator web interface. The top navigation bar includes the Anybus logo and the title 'Anybus Communicator' with an 'Apply' button. A left sidebar contains navigation options: Home, Configuration (Serial RS-232/485, EtherNet/IP, I/O data map), Maintenance (Files & firmware), and Troubleshooting (Diagnostics, Support). The main content area shows three status cards:

- EtherNet/IP™**: Setup in progress. IP: 192.168.0.111. Data flow: 2 byte(s) up, 0 byte(s) down. Includes a diagram of a PLC connected to a network.
- Anybus Communicator**: Initial state. A status box shows 'Gateway' (green), 'EtherNet/IP™' (green), and 'Serial' (grey with a lock icon). Includes a diagram of the Anybus Communicator hardware.
- Serial RS-232/485**: Ready for initialization. Modbus RTU, RS-232, 9600 baud. Data flow: 0 byte(s) down, 2 byte(s) up. Includes a diagram of a 'New node'.

#### Gateway status

Overview the Communicator LED indications remotely.

Refer to [Communicator LED Indicators, p. 86](#).





#### Node Status

Overview the status for each node added to the subnetwork.

#### Network Status and Settings

Overview communication status and the current networks settings.

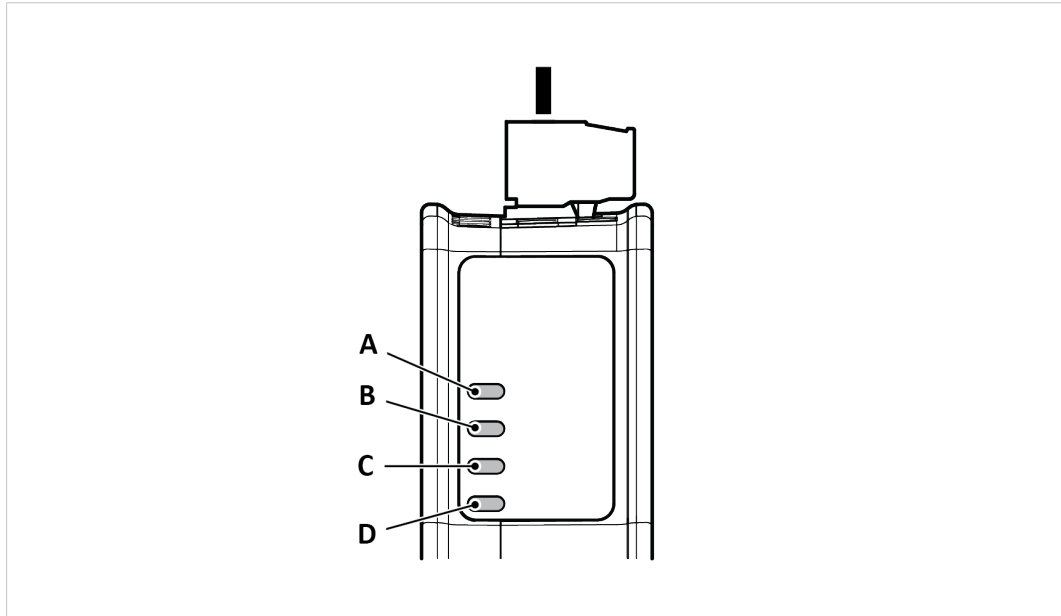
**Status Symbols**

| Symbol  | Description   |
|---|---|
|  | Internal error has occurred and operation cannot be guaranteed.   |
|  | Out of Specification.   |
|  | Check Function: <ul style="list-style-type: none"><li>• Initial state where non network components are started and configured.</li><li>• Network startup in progress.</li><li>• Invalid configuration detected.</li></ul> |
|  | Normal operation.   |

## 9.2 Communicator LED Indicators



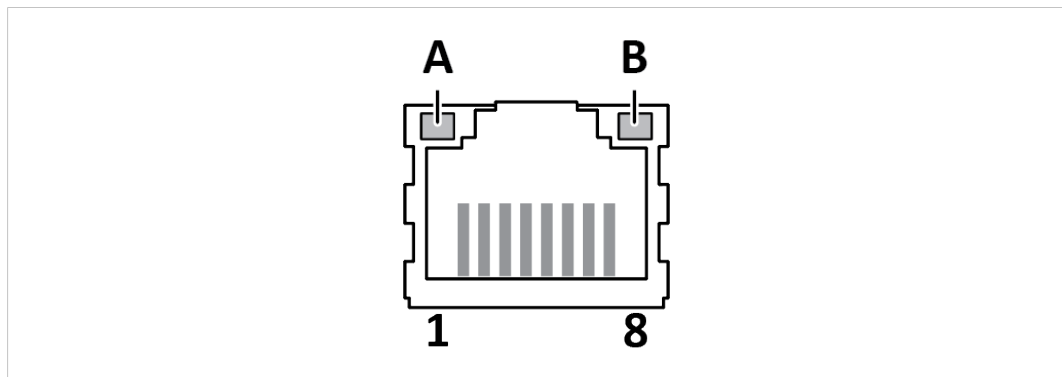
Before you can verify operation you must configure the Communicator.



|                            | LED A  | LED B   | LED C                                     | LED D  |
|----------------------------|--|---|---|--|
| <b>Operation Status</b>    | <b>Gateway status</b>                                | <b>EtherNet/IP - Adapter</b>                            | <b>Subnetwork</b>                         | <b>Security switch</b>                                     |
| <b>Off</b>                 | No power   | No power/Exception/No IP address                        | No power/Exception/Subnetwork not running | No power/Security switch is unlocked/Exception/Fatal error |
| <b>Green, flashing</b>     | Startup phase  | EtherNet/IP online, no connections established          | Running, one or more nodes are offline    | N/A  |
| <b>Green</b>               | Operational  | EtherNet/IP online, one or more connections established | Running                                   | Security switch is locked                                  |
| <b>Red</b>                 | Exception/Fatal error                                | Duplicated EtherNet IP address/Fatal error              | Fatal error                               | N/A  |
| <b>Red, flashing</b>       | Invalid configuration                                | One or more connections timed out                       | All nodes are offline                     | N/A  |
| <b>Green/Red, flashing</b> | Power up self-test/Firmware update/Firmware recovery | N/A   | N/A                                       | N/A  |



### 9.3 Ethernet LED Indicators



| LED A            | Function                      |
|------------------|-------------------------------|
| Off              | No link (or no power)         |
| Green            | Link (100 Mbit/s) established |
| Green, flashing  | Activity (100 Mbit/s)         |
| Yellow           | Link (10 Mbit/s) established  |
| Yellow, flashing | Activity (10 Mbit/s)          |

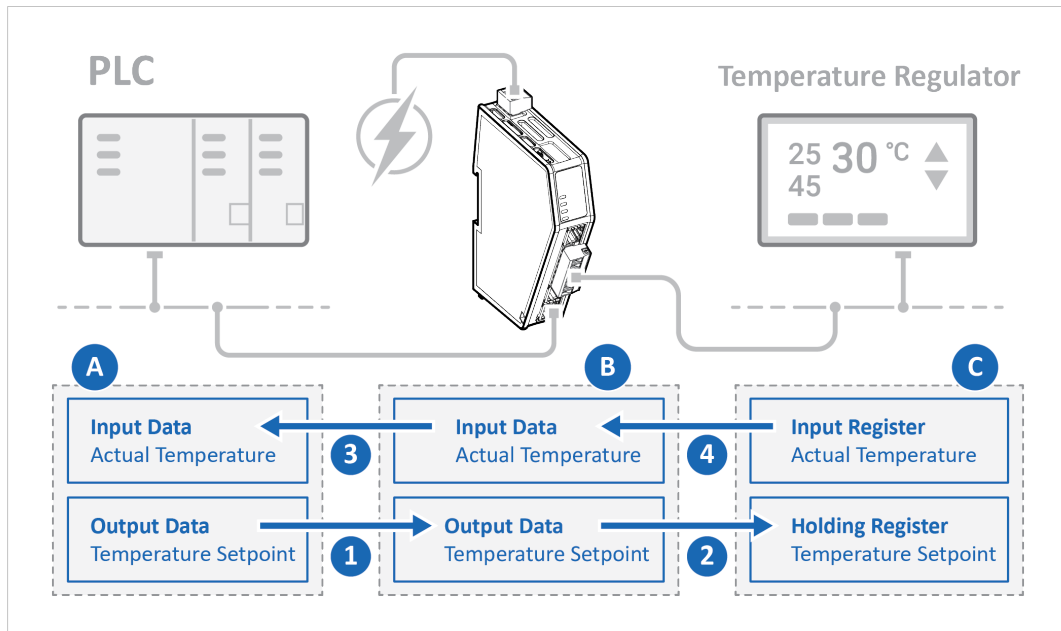
| LED B | Function |
|-------|----------|
| Off   | Not used |

## 10 Use Cases

### 10.1 Temperature Regulator - Modbus RTU Use Case

#### 10.1.1 About the Use Case

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



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 EtherNet/IP network (high level network).

#### 10.1.2 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, p. 31](#).

### 10.1.3 Choose Serial Protocol Type

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

The screenshot shows the 'Anybus Communicator' web interface. The top navigation bar includes the Anybus logo, the title 'Anybus Communicator', and an 'Apply' button. The main content area is titled 'Communication' and shows 'Serial protocol' options. The 'Modbus RTU' option is selected and highlighted with a blue border. Below it are three other options: 'Custom Request/Response', 'Custom Produce/Consume', and 'Custom Request/Response' (repeated). The left sidebar shows a navigation menu with 'Home', 'Configuration', 'Serial RS-232/485', 'Communication' (selected), 'Nodes & transactions', 'EtherNet/IP™', and 'I/O data map'.

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

### 10.1.4 Set Up Serial Communication

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

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

Basic settings

Physical standard: RS232

Baud rate: 19200 baud

Data bits: 8 data bits

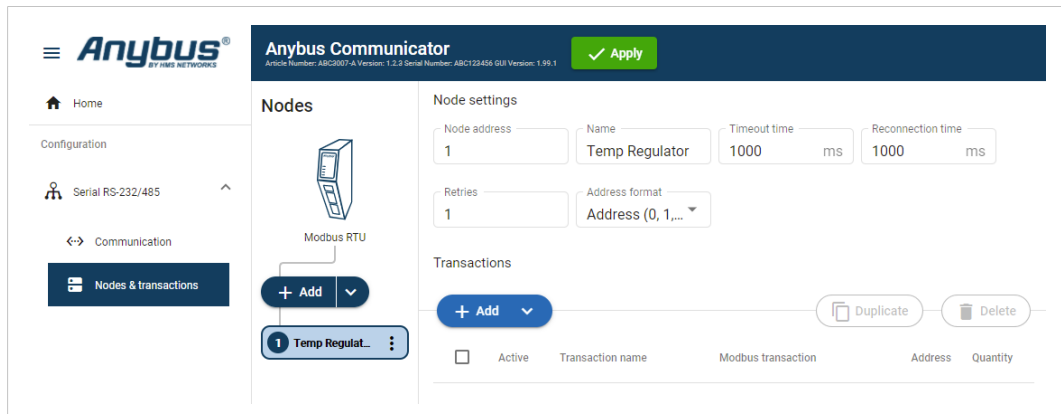
Parity: None

Stop bits: 2 stop bit

#### Used the following settings:

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

### 10.1.5 Set Up the Node



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

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

## 10.1.6 Set Up 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.

### Procedure

1. Select the **Temp Regulator** node.

Configure the **temperature setpoint** transaction:

The screenshot shows the 'Anybus Communicator' interface. On the left, a navigation menu includes 'Home', 'Configuration', 'Serial RS-232/485', 'Communication', 'Nodes & transactions', 'EtherNet/IP™', 'I/O data map', 'Maintenance', 'Files & firmware', 'Troubleshooting', 'Diagnostics', and 'Support'. The main area is titled 'Nodes' and shows a 'Modbus RTU' node named 'Temp Regulator'. Below the node name is an '+ Add' button and a 'Temp Regulat...' button. The 'Node settings' section includes fields for 'Node address' (1), 'Name' (Temp Regulator), 'Timeout time' (1000 ms), and 'Reconnection time' (1000 ms). The 'Retries' field is set to 1, and the 'Address format' is 'Address (0, 1,...)'. The 'Transactions' section has an '+ Add' button and a table with columns for 'Active', 'Transaction name', 'Modbus transaction', 'Address', and 'Quantity'. One transaction is listed: 'Temp Setpoint' (Active), 'Write Multiple Registers (16)', '0', and '1'. On the right, a sidebar for the selected transaction shows 'Transaction name' (Temp Setpoint), 'Modbus transaction' (Write Multiple Regi...), 'Address' (0), 'Quantity' (1), 'Update mode' (Cyclically), 'Update time' (1000 ms), and 'Positive ack' and 'Negative ack' checkboxes.

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.

#### 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<br>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:

5. To add a second transaction, click **Add**.
6. Select the transaction to configure.
7. In the transaction sidebar, on the right side of the screen.

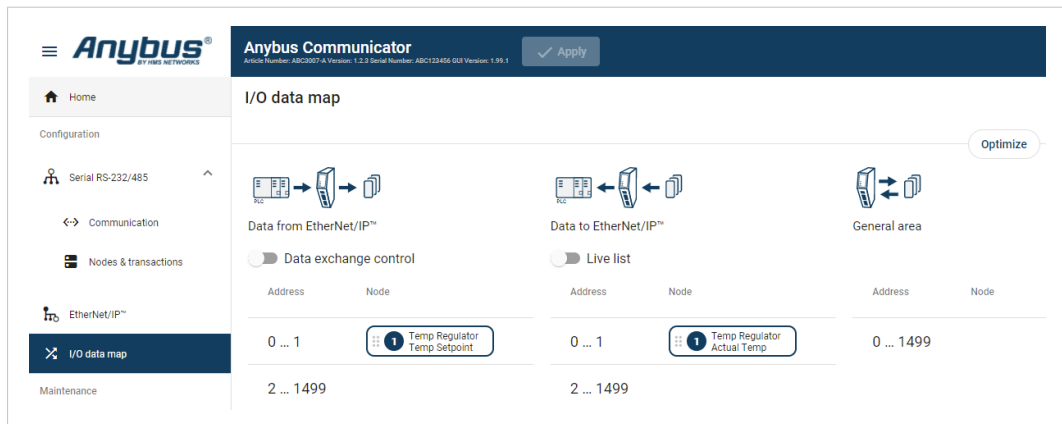
Enter values for the transaction settings.

#### Actual temperature transaction settings:

| Setting            | Value                      | Description  |
|--------------------|----------------------------|--|
| Transaction 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<br>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.   |

### 10.1.7 Check the I/O Data Map

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



| Address | Data to EtherNet/IP   |
|---------|---|
| 0-1     | Setpoint temperature from EtherNet/IP to the Temperature Regulator. |

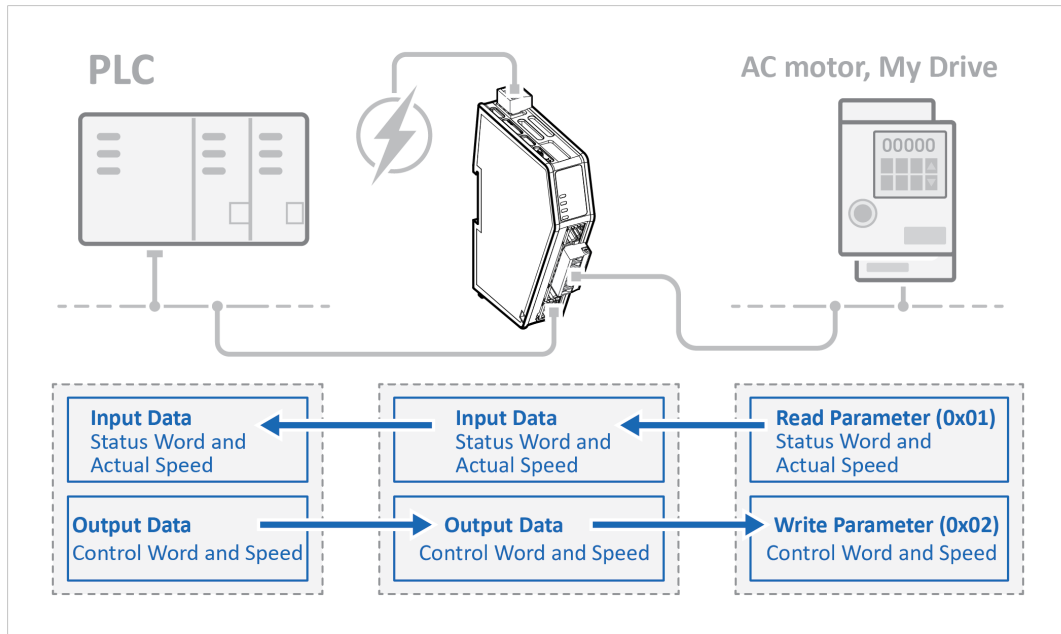
  

| Address | Data from EtherNet/IP   |
|---------|---|
| 0-1     | Actual temperature speed from the Temperature Regulator to EtherNet/IP. |

## 10.2 AC Motor Drive - Custom Request/Response Use Case

### 10.2.1 About the Use Case

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



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 EtherNet/IP network (high level network).

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

- Status word and actual speed from My Drive to the EtherNet/IP network.
- Control word and speed from the EtherNet/IP network to My Drive.

### 10.2.2 Before You Begin

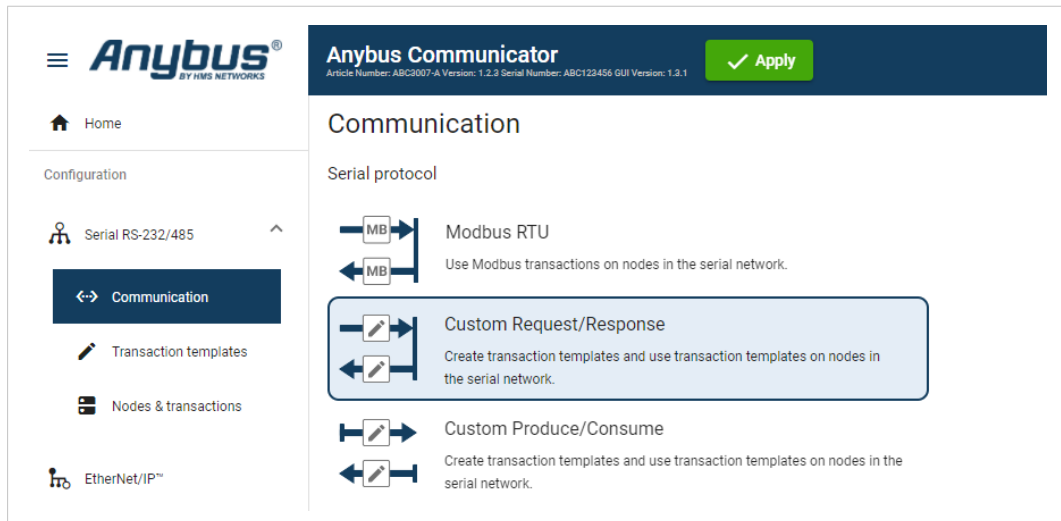
- Connect the Communicator configuration port to your computer.
- Power on the Communicator.
- Ensure that your computer can find the Communicator IP address.
- Enter the Communicator built-in web interface of the.

For more information refer to [Communicator Configuration, p. 31](#).



### 10.2.3 Choose Serial Protocol Type

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



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

### 10.2.4 Set Up Serial Communication

Set up the communication between the Communicator and My Drive.

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

Basic settings

Physical standard: RS-485

Baud rate: 57600 baud

Data bits: 8 data bits

Parity: None

Stop bits: 1 stop bit

#### Used the following settings:

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

## 10.2.5 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 EtherNet/IP to My Drive and status word and actual speed from the drive to EtherNet/IP.

### Create Read parameter (0x01)

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

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 fields** with the following settings:

| Request frame fields |               |              |                    |            |             |                   |
|----------------------|---------------|--------------|--------------------|------------|-------------|-------------------|
| Frame fields         | Name          | Bytes/Length | Type/Checksum type | Endianness | 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<br>Max 1000 |
| Constant             | Sub index     | 1            | Byte               | N/A        | No          | Min 0<br>Max 255  |
| Checksum             | Checksum      | 2            | CRC                | N/A        | N/A         | N/A               |

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

| Response frame fields |               |                  |                        |            |             |                   |
|-----------------------|---------------|------------------|------------------------|------------|-------------|-------------------|
| Frame field           | Name          | Bytes/<br>Length | Type/<br>Checksum type | Endianness | 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<br>bytes)    | Big-endian | No          | Min 0<br>Max 1000 |
| Constant              | Sub index     | 1                | Byte                   | N/A        | No          | Min 0<br>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.

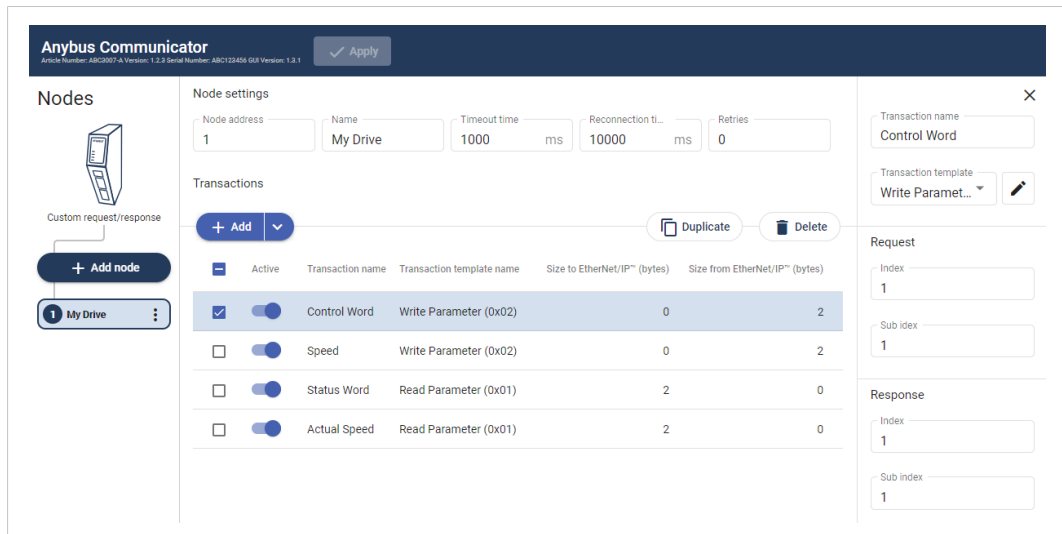
1. Add an **Empty template** and select it.
2. Name the template **Write parameter (0x02)**.
3. In the Frame editor **Request** area, add six **frame fields** with the following settings:

| Request frame fields |               |              |                        |            |             |                   |
|----------------------|---------------|--------------|------------------------|------------|-------------|-------------------|
| Frame field          | Name          | Bytes/Length | Type/<br>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<br>Max 1000 |
| Constant             | Sub index     | 1            | Byte                   | N/A        | No          | Min 0<br>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 fields** with the following settings:

| Response frame fields |               |       |                        |           |             |                   |
|-----------------------|---------------|-------|------------------------|-----------|-------------|-------------------|
| Frame fields          | Name          | Bytes | Type/<br>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<br>Max 1000 |
| Constant              | Sub index     | 1     | Byte                   | N/A       | No          | Min 0<br>Max 255  |
| Checksum              | Checksum      | 2     | CRC                    | N/A       | N/A         | N/A               |

### 10.2.6 Set Up Node and 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<br>My Drive is set up as a node with Node address 1. |
| Name              | My Drive   |
| Timeout time      | 1000 ms (default)                                      |
| Reconnecting time | 10000 ms (default)                                     |
| Retries           | 0 (default)  |

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

| 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         |

### 10.2.7 Check the I/O Data Map

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

#### Control word and speed from EtherNet/IP to My Drive

| Address | Data to EtherNet/IP |
|---------|---------------------|
| 0-1     | Control Word        |
| 2-3     | Speed               |

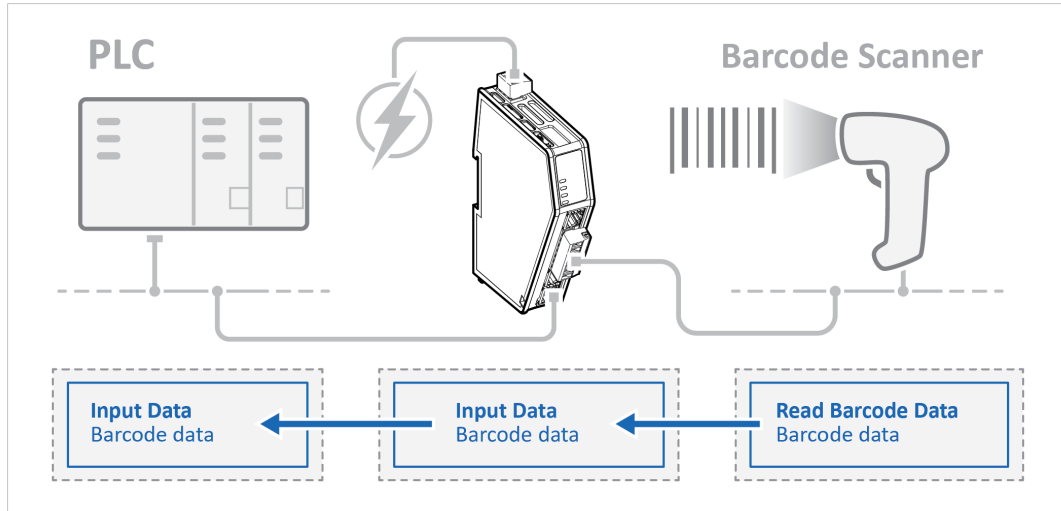
#### Status word and actual speed from My Drive to EtherNet/IP

| Address | Data from EtherNet/IP |
|---------|-----------------------|
| 0-1     | Control Word          |
| 2-3     | Speed                 |

## 10.3 Barcode Scanner - Custom Produce/Consume Use Case

### 10.3.1 About the Use Case

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



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 EtherNet/IP network (high level network).

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

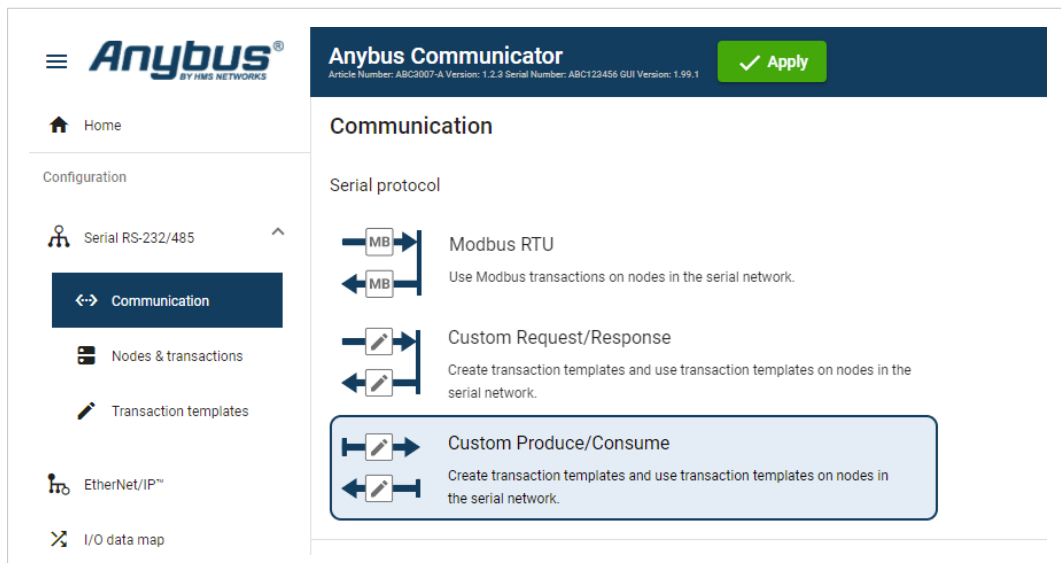
### 10.3.2 Before You Begin

- 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, p. 31](#).

### 10.3.3 Choose Serial Protocol Type

The Barcode Scanner is using a produce/consume protocol to access parameters addressed with index and sub index.



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

### 10.3.4 Set Up Serial Communication

Set up the communication between the Communicator and the Barcode Scanner.

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

Basic settings

Physical standard: RS-232

Baud rate: 9600 baud

Data bits: 8 data bits

Parity: None

Stop bits: 1 stop bit

#### Used the following settings:

| Frame fields      | Value      |
|-------------------|------------|
| Physical standard | RS-232     |
| Baud rate         | 9600 baud  |
| Data bits         | 8 bits     |
| Parity            | 1          |
| Stop bits         | 1 stop bit |



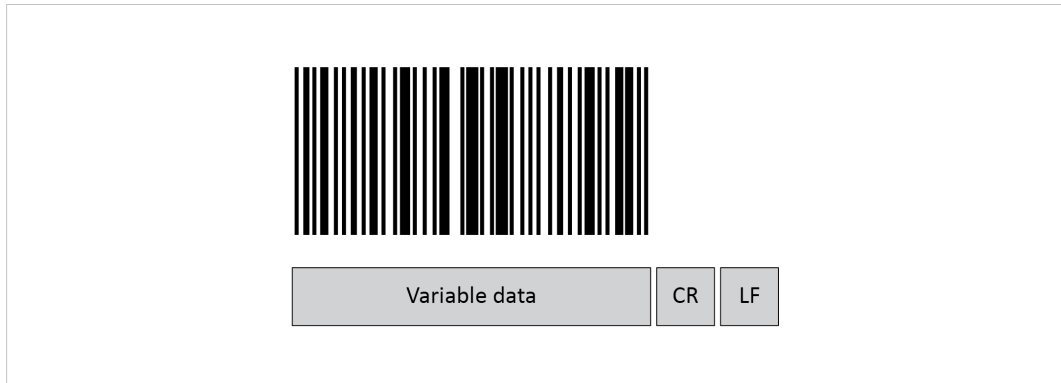
### 10.3.5 Create Transaction Templates

#### Create Read Barcode Data Parameter

##### Before You Begin

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

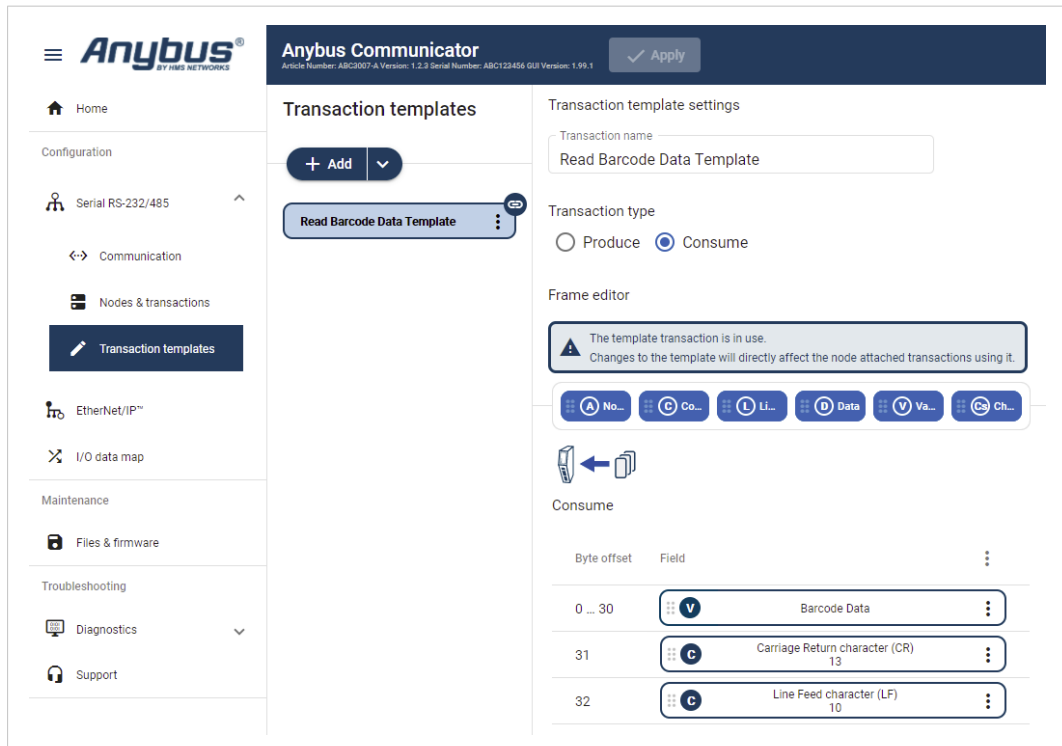
The Barcode Scanner sends data whenever it is available, without any request or handshake from the Communicator.



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).

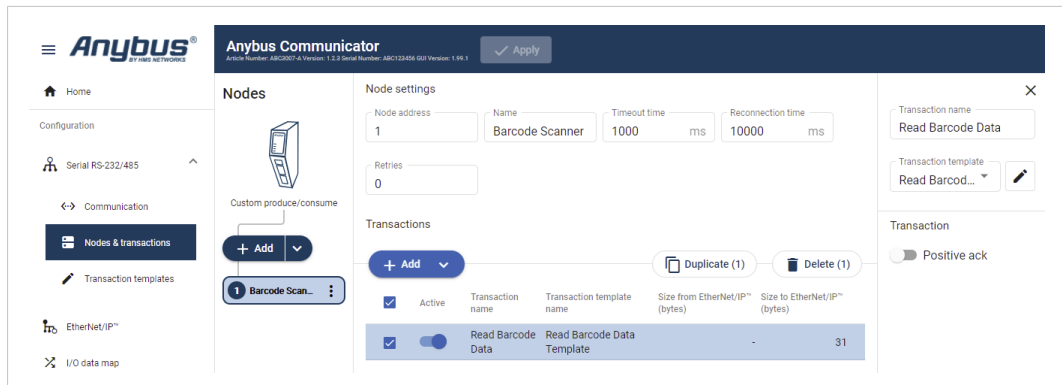
Procedure



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:

| 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         |

### Set Up Node and Transactions



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

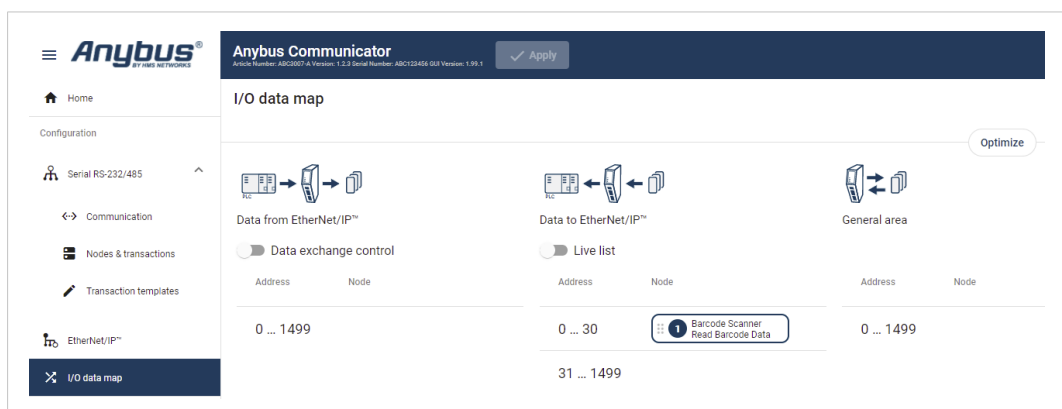
| Node settings     | Value  |
|-------------------|--|
| Node address      | The Barcode Scanner is set up as a node with Node address 1. |
| Name              | Barcode Scanner  |
| Timeout time      | 1000 ms (default)  |
| Reconnecting time | 1000 ms (default)  |
| Retries           | 0 (default)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

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

| The Barcode Scanner contains the following parameters |                            |
|---|----------------------------|
| Transaction name                                      | Transaction template       |
| Read Barcode Data                                     | Read Barcode Data Template |

### Check the I/O Data Map

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



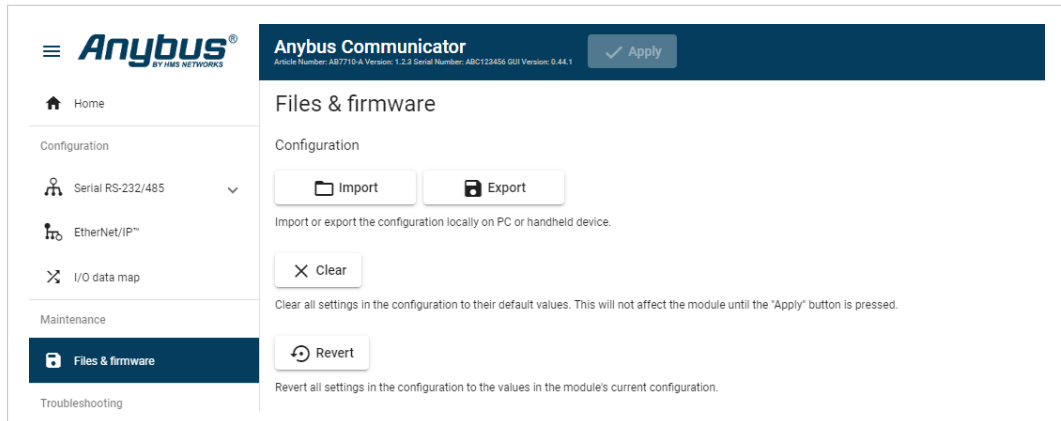
| Status word and actual speed from My Drive to EtherNet/IP |   |
|---|---|
| Address   | Barcode Scanner Parameter   |
| 0-31  | The variable data, 31 bytes, are forwarded from the Barcode Scanner to the PLC. |

# 11 Maintenance

## 11.1 Configuration File Handling

### 11.1.1 Export Configuration

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



To export a configuration file:

1. In Files & firmware, click **Export**.
  - The configuration settings are stored in a `.conf` file and downloaded to your PC.

### 11.1.2 Import Configuration

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

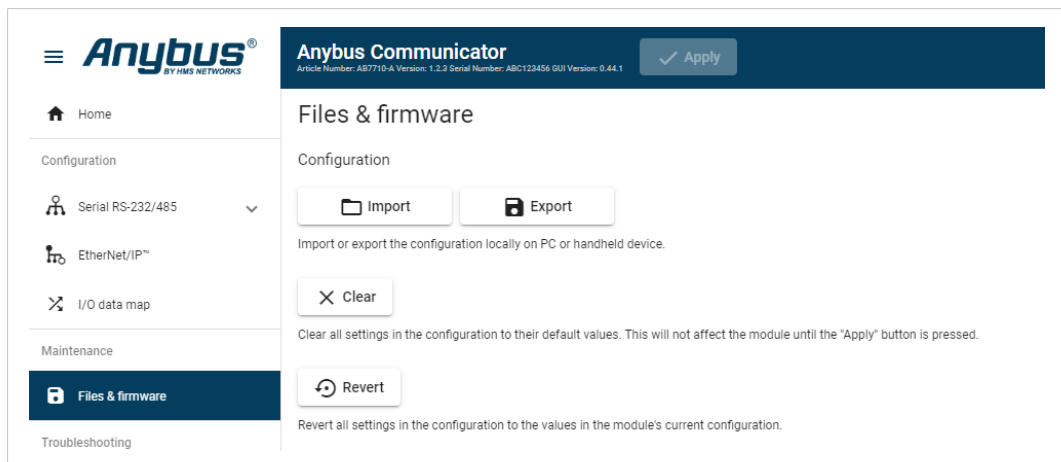
#### Before You Begin



Importing a configuration replaces the current applied configuration.

Supported file format is `.conf`.

#### Procedure



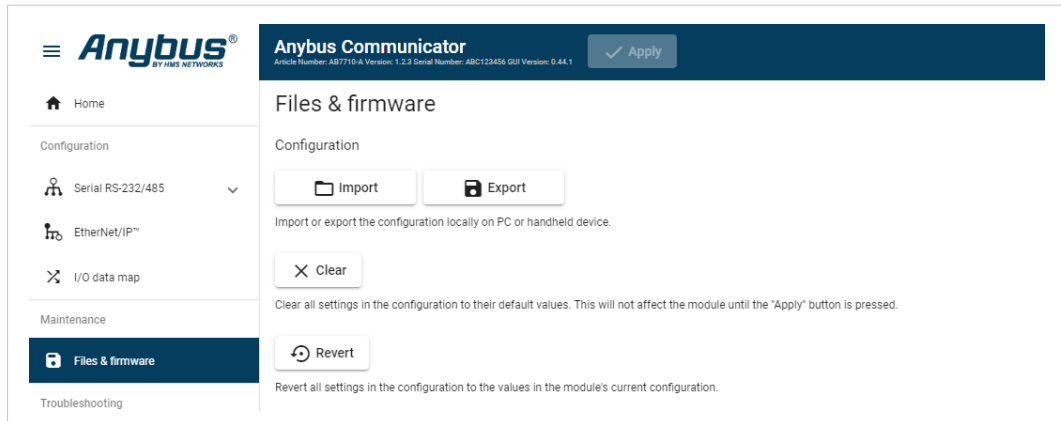
Import configuration file:

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 occurs, a message about the mismatch appears.
7. To apply the settings, click **Apply** in the web-interface header, and follow the instructions.

## 11.2 Clear and Revert Configuration

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

### Procedure



To clear the configuration:

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.

You can revert all changes done to the settings in the current configuration.

To Revert the configuration:

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.

## 11.3 Firmware Management

### 11.3.1 View the Firmware Version

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



### 11.3.2 Firmware and Configuration Compatibility

#### Compatibility after firmware upgrade

Current configuration is still compatible after upgrading the firmware.

#### Compatibility after firmware downgrade



Compatibility after a firmware downgrade can not be guaranteed.

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

### 11.3.3 Firmware File Validation

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

- the firmware is compatible with the Communicator hardware
- the firmware is suited for the product
- that 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.

### 11.3.4 Update Firmware

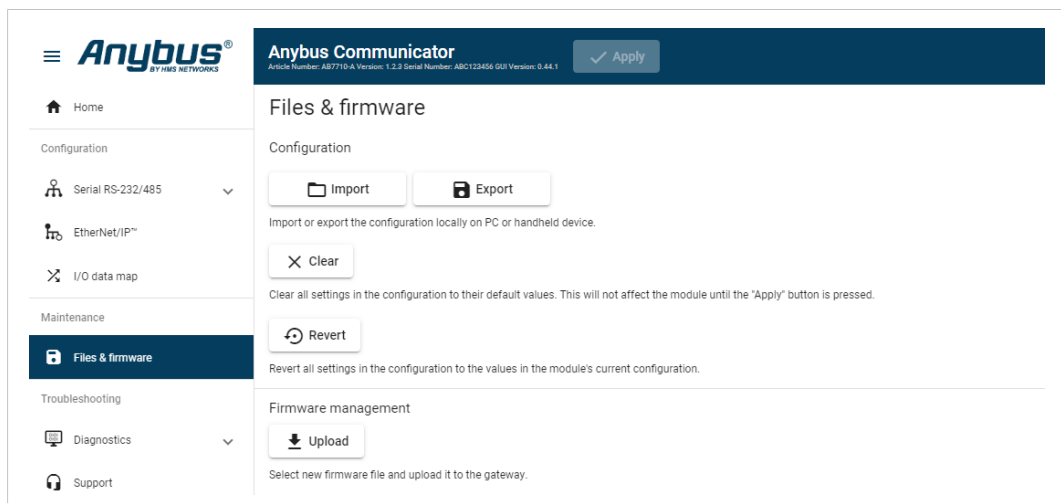
#### Before You Begin



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

Ensure that the Communicator is disconnected from the OT networks.

#### Procedure



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 pass 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 appear.

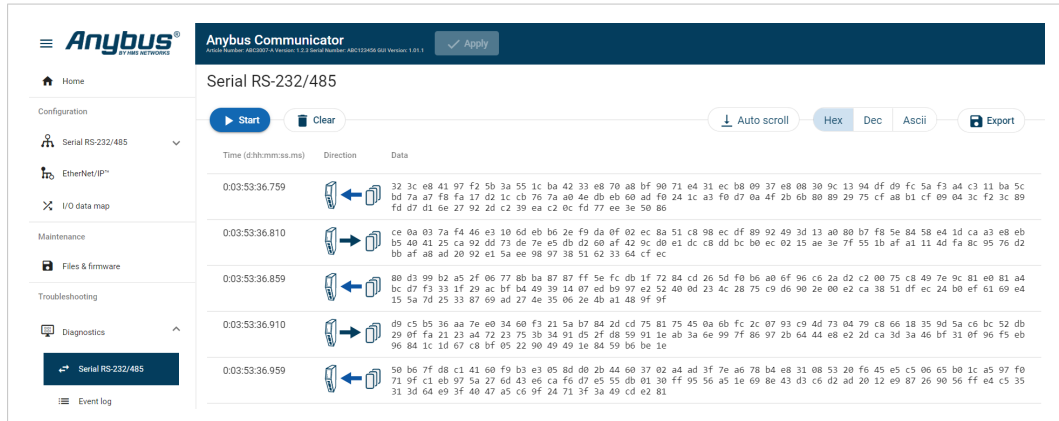


# 12 Troubleshooting

## 12.1 Diagnostics

### 12.1.1 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.



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

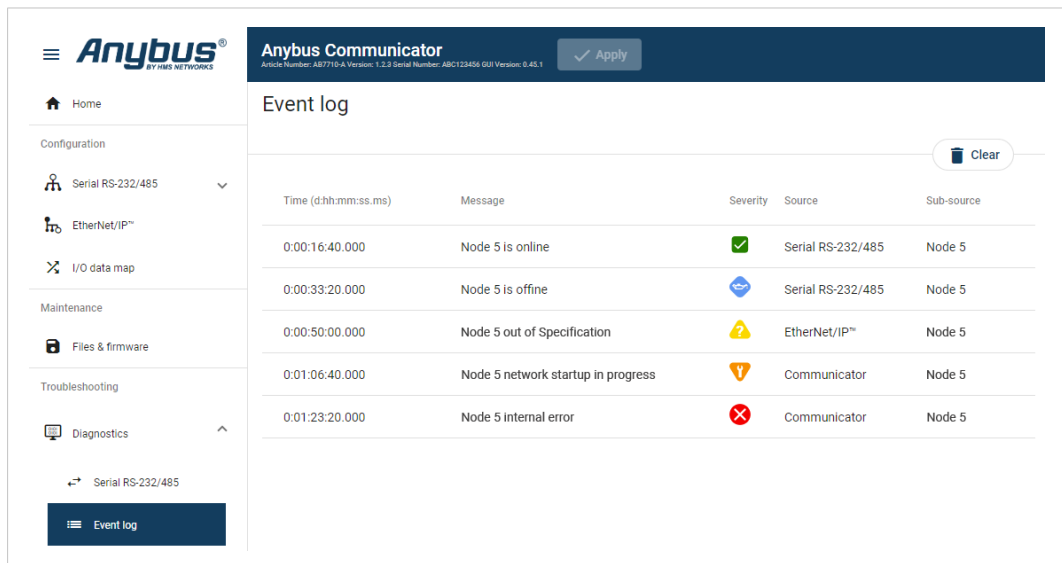
- To start the data flow, click **Start**.
- To end the data flow, click **Stop**.

#### Export data flow

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

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

## 12.1.2 Event Log



### How To Analyze the Information

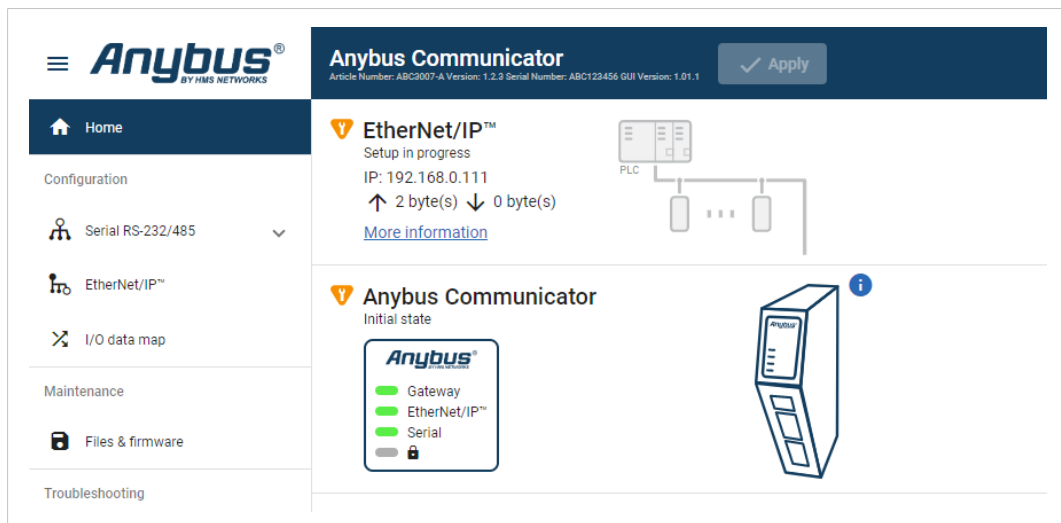
The log follows the FIFO principle, first in and first out. The oldest (first) value is processed first.

|                      |   |                                 |
|----------------------|---|---------------------------------|
| 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.<br>For description of the symbols, refer to <a href="#">Communicator Status Monitor, p. 84</a> .  |                                 |
| Source               | 0   | Communicator                    |
|                      | 1   | High level network, EtherNet/IP |
|                      | 2   | Subnetwork, Serial RS-232/485   |
| Sub-source           | The nodes connected to the subnetwork and the PLC connected to the high level network.<br>If there is a problem with a node the node name is displayed in the Sub-source column.<br>Example: If the node name is 5, number 5 is displayed in the Sub-source column. |                                 |

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

### 12.1.3 LED Status

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



For information about the LED indication, refer to [Communicator LED Indicators, p. 86](#).

## 12.2 Reset to Factory Settings

### Before You Begin

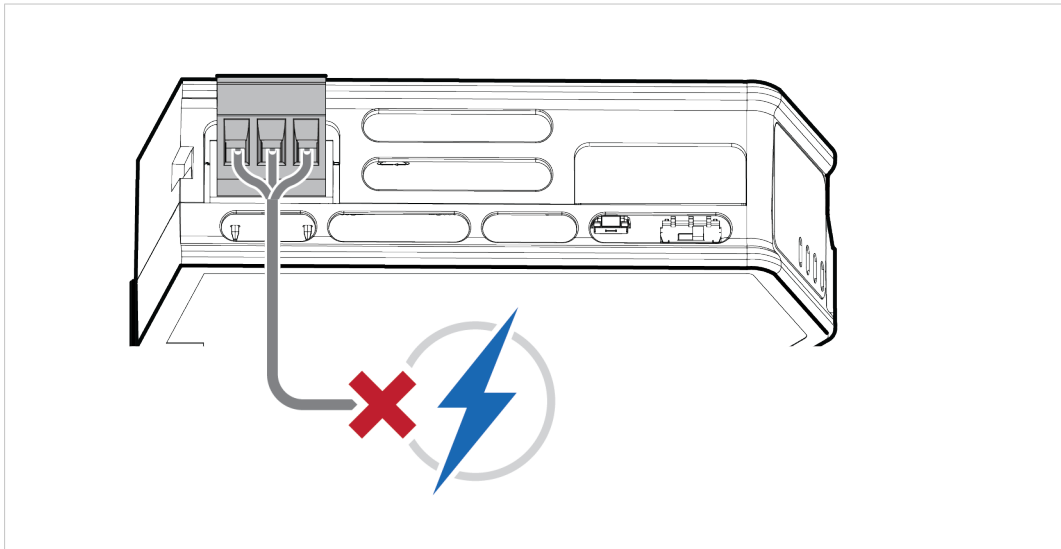
Factory reset will reset any on site made configuration changes and set the Communicator to the same state as leaving HMS production.

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

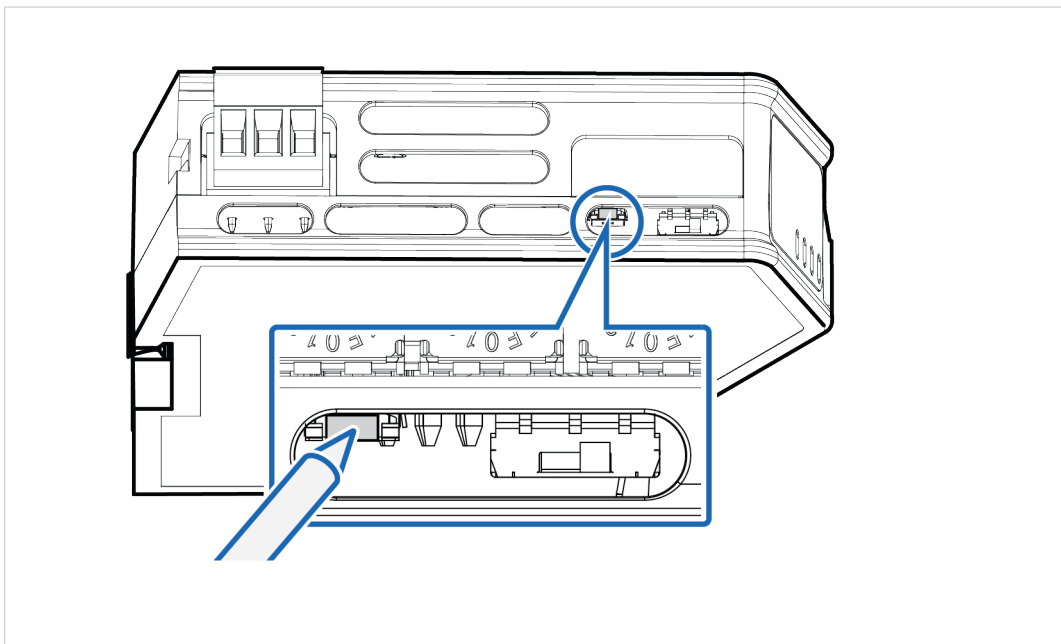
### Procedure

To reset the Communicator:

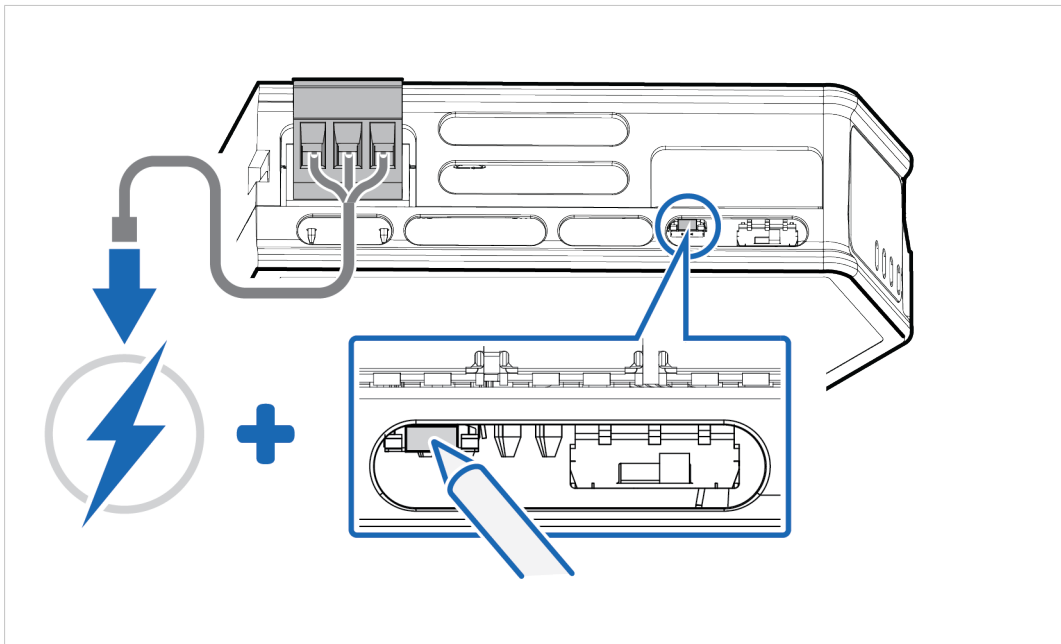
1. Disconnect the Communicator from power.



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



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

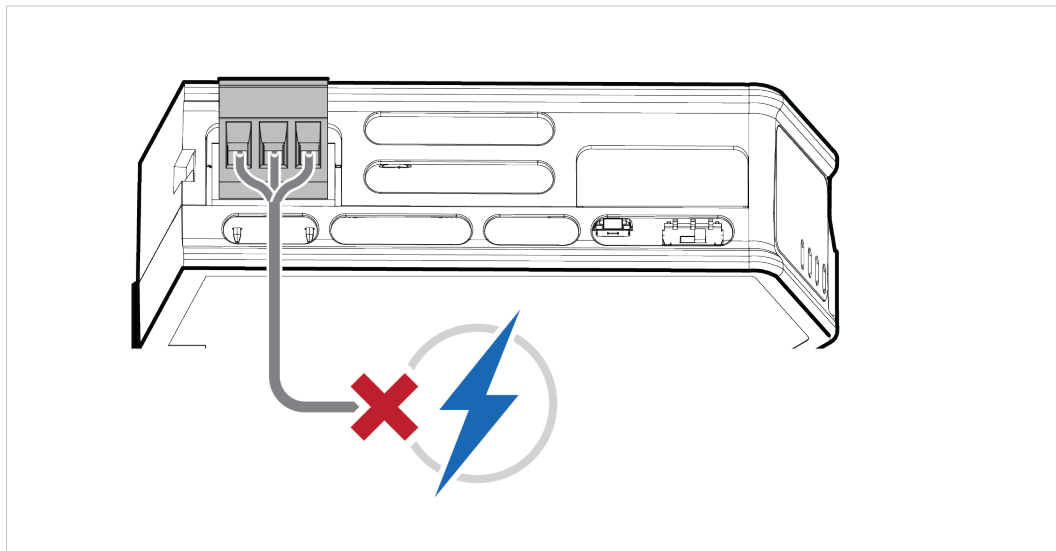
## 12.3 Firmware Upgrade Error Management

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

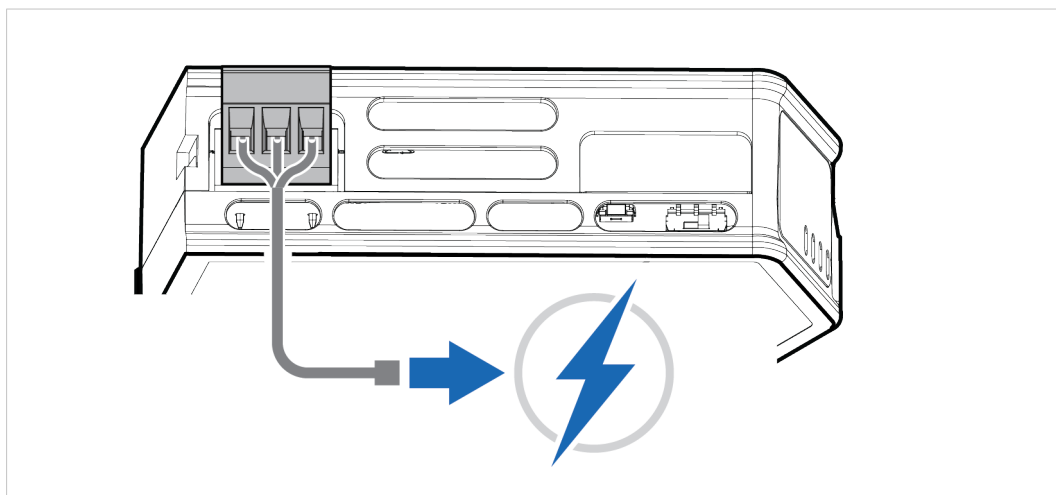
The last working firmware is still available on the flash, but it is not active.

To complete the interrupted firmware update:

1. Disconnect the Communicator from power.

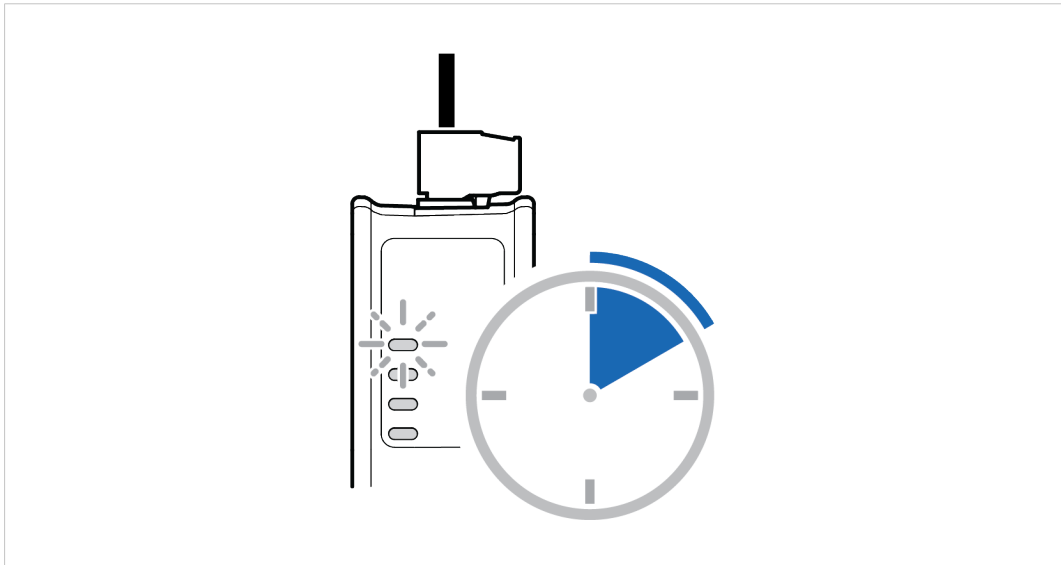


2. Reconnect the Communicator to power.

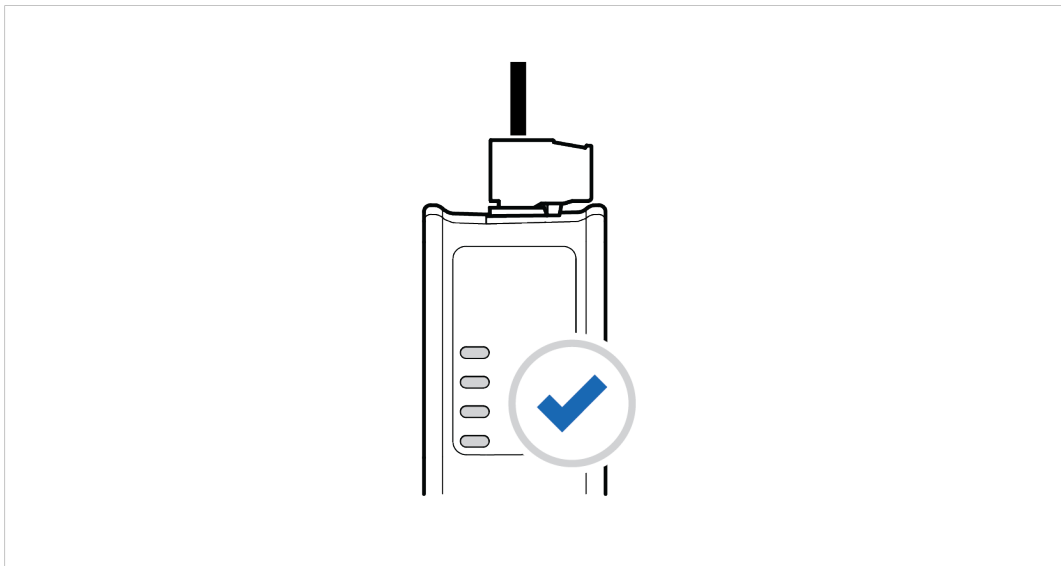


3. Leave the Communicator for 10 minutes.

The *Gateway status* led indicator flashes red and green until the firmware upgrade is completed.

**Result**

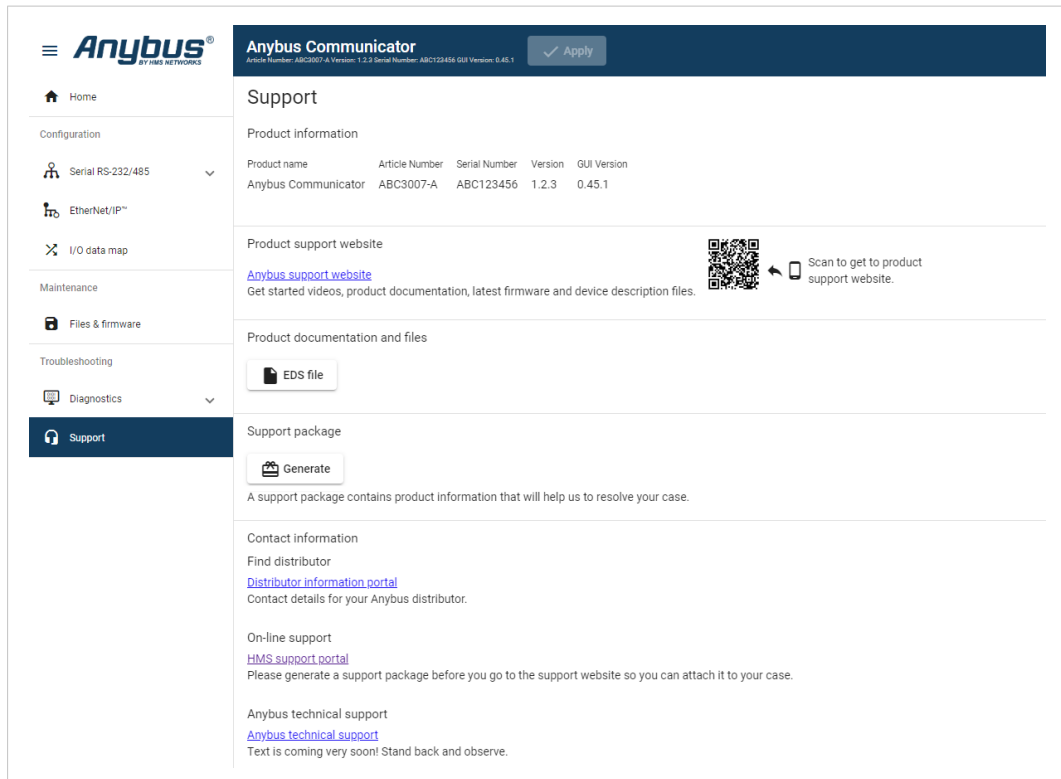
→ The Communicator recover and return to normal operation.



To check LED status, refer to [Communicator LED Indicators, p. 86](#).

## 12.4 Support

### 12.4.1 Support Package



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

The support package contain 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 are available to open and read, the files are not locked or encrypted.

#### Generate Support Package

1. 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 Support page, click **Anybus support website**.
  - You are redirected to the Anybus support website.
2. On the Anybus support website, create a support ticket and upload the support package.



## 13 Technical Data

### 13.1 Technical Specifications

|                         |   |
|-------------------------|---|
| Article identification  | ABC3007-A   |
| Communication connector | RJ45 x 2  |
| Configuration connector | RJ45  |
| Serial connector        | 7-pin screw connector   |
| Power connector         | 3-pin screw connector   |
| Power supply            | 12-30 VDC<br>Reverse voltage protection and short circuit protection  |
| Power consumption       | Typical: 160 mA @ 24 V<br>Max: 400 mA @ 12 V  |
| Storage temperature     | -40 to +85 °C   |
| Operating temperature   | -25 to +70 °C   |
| Humidity                | EN 60068-2-78: Damp heat, +40°C, 93% humidity for 4 days<br>EN 60068-2-30: Damp heat, +25°C – +55°C, 95% RH, 2 cycles |
| Vibration               | See datasheet   |
| Housing material        | Plastic<br>See datasheet for details  |
| Protection class        | IP20  |
| Product weight          | 150 g   |
| Dimensions              | 27 x 144 x 98 mm (H x W x D) with connectors included   |
| Mounting                | DIN-rail  |

Additional technical data and information related to the installation and use of this product can be found at [www.anybus.com/support](http://www.anybus.com/support).

**This page intentionally left blank**

## A Reference Guides

### A.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.

### A.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 organisation website.

### A.3 Modbus Transactions

| 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.<br>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.<br>The write operation is performed before the read.  |

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

For more information refer to the Modbus organisation website.

## A.4 Modbus 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.<br>If multiple registers were requested, at least one was not permitted. |

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

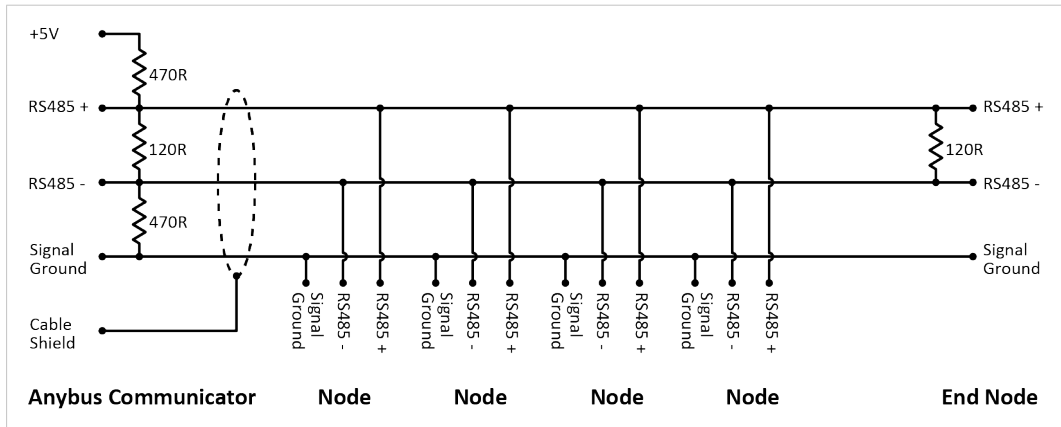
For more information refer to the Modbus organisation website.

## A.5 ASCII Table

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

## A.6 RS485/RS232 Electrical Connection

### A.6.1 RS485 Typical Connection



### A.6.2 RS232 Typical Connection

