ProSoft® TECHNOLOGY
A BELDEN BRAND

Where Automation Connects.

ProLinx® edge

ELX-EIP-MB

EtherNet/IP™ and Modbus® TCP/RTU/ASCII Protocol Gateway Container

October 13, 2026

USER MANUAL

Your Feedback Please

We always want you to feel that you made the right decision to use our products. If you have suggestions, comments, compliments or complaints about our products, documentation, or support, please write or call us.

ProSoft Technology, Inc.
+1 (661) 716-5100
+1 (661) 716-5101 (Fax)
www.prosoft-technology.com
ps.support@belden.com

ELX-EIP-MB Container User Manual
For Public Use.

October 13, 2026

ProSoft Technology®, is a registered copyright of ProSoft Technology, Inc. All other brand or product names are or may be trademarks of, and are used to identify products and services of, their respective owners.

Content Disclaimer

This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation, and testing of the products with respect to the relevant specific application or use thereof. Neither ProSoft Technology nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. Information in this document including illustrations, specifications and dimensions may contain technical inaccuracies or typographical errors. ProSoft Technology makes no warranty or representation as to its accuracy and assumes no liability for and reserves the right to correct such inaccuracies or errors at any time without notice. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of ProSoft Technology. All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components. When devices are used for applications with technical safety requirements, the relevant instructions must be followed. Failure to use ProSoft Technology software or approved software with our hardware products may result in injury, harm, or improper operating results. Failure to observe this information can result in injury or equipment damage.

© 2026 ProSoft Technology. All Rights Reserved.



For professional users in the European Union
If you wish to discard electrical and electronic equipment (EEE), please contact your dealer or supplier for further information.



Prop 65 Warning – Cancer and Reproductive Harm – www.P65Warnings.ca.gov

Agency Approvals and Certifications

Please visit our website: www.prosoft-technology.com

Open-Source Information

Open-Source Software used in the product

The product contains, among other things, Open-Source Software files, as defined below, developed by third parties and licensed under an Open-Source Software license. These Open-Source Software files are protected by copyright. Your right to use the Open-Source Software is governed by the relevant applicable Open-Source Software license conditions. Your compliance with those license conditions will entitle you to use the Open-Source Software as foreseen in the relevant license. In the event of conflicts between other ProSoft Technology, Inc. license conditions applicable to the product and the Open-Source Software license conditions, the Open-Source Software conditions shall prevail. The Open-Source Software is provided royalty-free (i.e. no fees are charged for exercising the licensed rights). Open-Source Software contained in this product and the respective Open-Source Software licenses are stated in the module webpage, in the link Open Source.
If Open-Source Software contained in this product is licensed under GNU General Public License (GPL), GNU Lesser General Public License (LGPL), Mozilla Public License (MPL) or any other Open-Source Software license, which requires that source code is to be made available and such source code is not already delivered together with the product, you can order the corresponding source code of the Open-Source Software from ProSoft Technology, Inc. - against payment of the shipping and handling charges - for a period of at least 3 years since purchase of the product. Please send your specific request, within 3 years of the purchase date of this product, together with the name and serial number of the product found on the product label to:

ProSoft Technology, Inc.
Head of R&D
9201 Camino Media, Suite 200
Bakersfield, CA 93311
USA

Warranty regarding further use of the Open-Source Software

ProSoft Technology, Inc. provides no warranty for the Open-Source Software contained in this product, if such Open-Source Software is used in any manner other than intended by ProSoft Technology, Inc. The licenses listed define the warranty, if any, from the authors or licensors of the Open-Source Software. ProSoft Technology, Inc. specifically disclaims any warranty for defects caused by altering any Open-Source Software or the product's configuration. Any warranty claims against ProSoft Technology, Inc. if the Open-Source Software contained in this product infringes the intellectual property rights of a third party are excluded. The following disclaimer applies to the GPL and LGPL components in relation to the rights holders:
"This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License and the GNU Lesser General Public License for more details."
For the remaining Open-Source components, the liability exclusions of the rights holders in the respective license texts apply. Technical support, if any, will only be provided for unmodified software.

Table of Contents

1 Start Here 7

1.1 About the ELX-EIP-MB Container 7
1.2 Configuring the ELX-EIP-MB Container 7

2 Quick Setup Guide 8

2.1 ProSoft Configuration Builder 9
2.1.1 Creating a Project 9
2.2 Establishing Communications as a Server 11
2.3 Establishing Communications as a Client 12
2.4 Data Exchange 13
2.4.1 EtherNet/IP Class 1 Adapter 13
2.4.2 EtherNet/IP Class 3 Server 15
2.4.3 EtherNet/IP Class 3 Client 16
2.4.4 Modbus TCP Server 20
2.4.5 Modbus TCP Client 22
2.4.6 Modbus RTU Slave 24
2.4.7 Modbus RTU Master 27

3 Container Deployment through the Local User Interface 30

3.1 Container Network Configuration 30
3.1.1 Adding a Network 30
3.2 Container Storage Configuration 35
3.2.1 Adding a Storage Volume (optional) 35
3.2.2 Deleting a Volume 37
3.3 Deploying Container 38
3.3.1 Importing the Image 38
3.3.2 Adding the Application 39
3.3.3 Container Status 59
3.3.4 Verifying Successful Container Deployment 60

4 Belden Horizon 61

4.1 Container Network Configuration 61
4.1.1 Adding a Network to Gateway 61
4.2 Accessing Application Details 65
4.2.1 Overview 66
4.2.2 Details 67
4.2.3 Configuration 68
4.3 Deploying the Container 70
4.3.1 Verifying Successful Container Deployment 76
4.4 Gateways 77
4.5 Removing the Container Application from the Gateway 77

5 ProSoft Configuration Builder 79

5.1 Overview 79
5.1.1 Compatible Operating Systems 79
5.1.2 Hash Signature 79

5.2 Using ProSoft Configuration Builder 80
5.2.1 Selecting the ELX-EIP-MB Container 81
5.2.2 Connecting the PC to the ELX3 81
5.3 Downloading the Project to the Container 82
5.3.1 Downloading an ELX-EIP-MB Project 82
5.3.2 Downloading an ELX-EIP-MBTCP Project 84
5.4 Uploading the Configuration from the Container 87
5.4.1 Uploading a Configuration to an ELX-EIP-MB Project 87
5.4.2 Uploading a Configuration to an ELX-EIP-MBTCP Project 90
5.5 Diagnostics and Troubleshooting 92
5.5.1 Diagnostics Menu 94
5.5.2 Capturing a Diagnostic Session to a Log File 95
5.5.3 Warm Boot / Cold Boot 95

6 Internal Database 96

7 EIP Protocol 98

7.1 EIP Functional Overview 98
7.1.1 EtherNet/IP Specifications 98
7.2 EIP Internal Database 99
7.2.1 EIP Client Access to ELX-EIP-MB Container Database 99
7.2.2 EIP Server Access to ELX-EIP-MB Container Database 99
7.3 EIP Driver Configuration 100
7.3.1 EIP Class 1 Connection 100
7.3.2 EIP Class 3 Client [x] Connection 102
7.3.3 EIP Class 3 UClient Connection 107
7.3.4 EIP Class 3 Server 112
7.4 Network Diagnostics 113
7.4.1 EIP PCB Diagnostics 113
7.4.2 EIP Status Data in Upper Memory 114
7.5 EIP Error Codes 117
7.5.1 EIP Error Codes 117
7.5.2 EtherNet/IP CIP Generic Error Codes 117
7.5.3 TCP/IP Interface Error Codes 117
7.5.4 Common Response Error Codes 118
7.5.5 Register Session Response Error Codes 118
7.5.6 Forward Open Response Error Codes 118
7.5.7 EIP Class 1 Connection Error Codes 118
7.6 EIP Reference 119
7.6.1 Adding the ELX-EIP-MB Container to RSLogix 119
7.6.2 ControlLogix and CompactLogix Processor Specifics 128

8 MBTCP Protocol 132

8.1 MBTCP Functional Overview 132
8.1.1 MBTCP Specifications 133
8.2 MBTCP Internal Database 134
8.2.1 MBTCP Client Access to the ELX-EIP-MB Container Database 134
8.2.2 MBTCP Server Access to the ELX-EIP-MB Container Database 135
8.2.3 Enron-Daniels Float Data 135
8.3 MBTCP Driver Configuration 137
8.3.1 MBTCP Servers 137
8.3.2 MBTCP Client [x] 139
8.3.3 MBTCP Client [x] Commands 141

8.4 Network Diagnostics 145
8.4.1 MBTCP PCB Diagnostics 145
8.4.2 MBTCP Status Data in Upper Memory 146
8.5 MBTCP Error Codes 149
8.5.1 Standard Modbus Exception Code Errors 149
8.5.2 MBTCP Client Specific Errors 149
8.5.3 MBTCP Communication Error Codes 149
8.5.4 MBTCP Command List Error Codes 149

9 MBS Protocol 150

9.1 MBS Functional Overview 150
9.1.1 Modbus Master/Slave Port Specifications 151
9.2 MBS Internal Database 152
9.2.1 Modbus Port Access to Database 153
9.3 MBS Driver Configuration 154
9.3.1 Configuring MBS Port 1 154
9.3.2 MBS Port 1 Commands 158
9.4 Network Diagnostics 161
9.4.1 MBS PCB Diagnostics 161
9.4.2 MBS Status Data in Upper Memory 161
9.5 MBS Error Codes 166
9.5.1 Standard Modbus Exception Code Errors 166
9.5.2 Container Communication Error Codes 166
9.5.3 MBS Command List Error Codes 166
9.5.4 MBS Configuration Error Codes 167

10 Mapping Data in Container Memory 168

11 Security 171

12 Appendix A - CIP Objects 172

12.1 Identity Object (0x01) 172
12.2 Port Object (0xF4) 173
12.3 TCP/IP Interface Object (0xF5) 174
12.4 Ethernet Link Object (0xF6) 175
12.5 LLDP Management Object (0x109) 176
12.6 LLDP Data Table Object (0x10A) 178

13 Support, Service, and Warranty 180

13.1 Contacting Technical Support 180
13.2 Warranty Information 180

1 Start Here

1.1 About the ELX-EIP-MB Container

The ELX-EIP-MB is a Docker Container designed to provide protocol gateway conversion to the ProSoft Technology ELX3 hardware platform. The protocol gateway container facilitates the data exchange between the EtherNet/IP™ (EIP) and Modbus® TCP/RTU/ASCII (MB) protocols, supporting both Client and Server functionalities.

The protocol gateway container is configured with the ProSoft Configuration Builder (PCB) software. PCB is also used to download and upload the EtherNet/IP and Modbus TCP/RTU/ASCII network configurations to the container.

The EIP driver is tested and certified by the ODVA organization and includes a shared database (10,000 16-bit registers) between EtherNet/IP and Modbus TCP/RTU/ASCII protocols.

Note: To install the ELX-EIP-MB container, please see chapter *3 Container Deployment through the Local User Interface* or the *ELX3 User Manual* (at www.prosoft-technology.com).

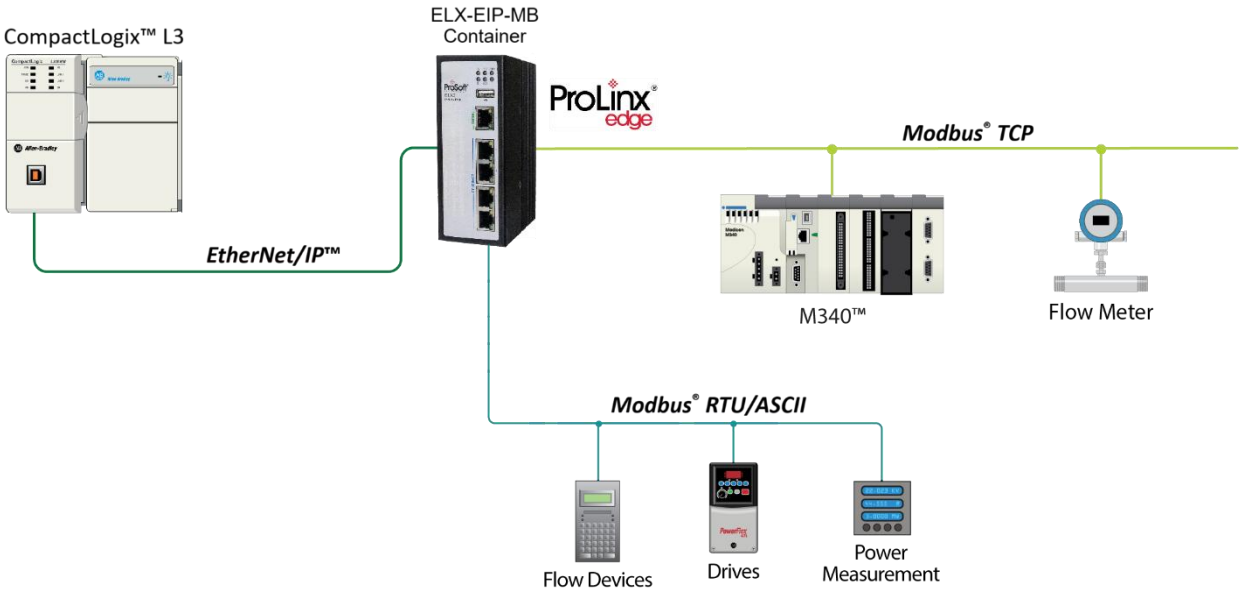
Note: For Security information, please see the *ELX3 User Manual.*

1.2 Configuring the ELX-EIP-MB Container

ProSoft Configuration Builder software is used to configure the ELX-EIP-MB container. It can be downloaded from: https://www.prosoft-technology.com/Products/ProSoft-Software/ProSoft-Configuration-Builder

2 Quick Setup Guide

This chapter provides a quick startup guide with examples on how to get the ELX-EIP-MB container application up and running. An application can consist of a Client/Master (source) sending data to a Server/Slave (destination) through the container as follows:

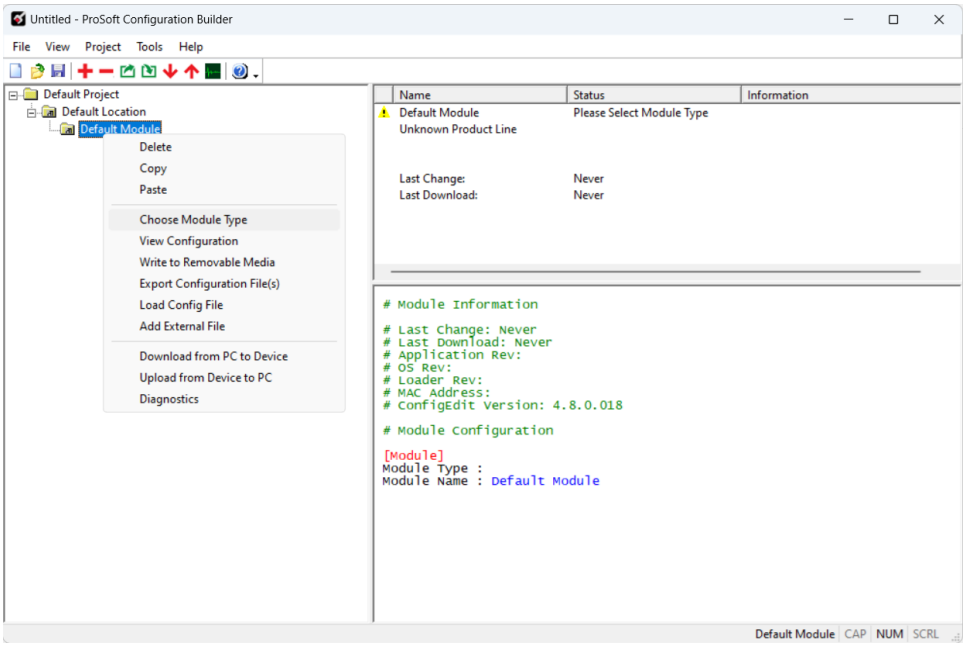


2.1 ProSoft Configuration Builder

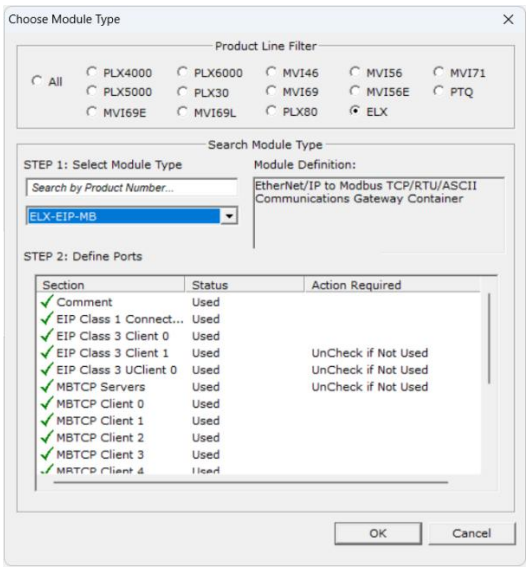
The ProSoft Configuration Builder (PCB) software provides a quick and easy way to manage container configuration files. For more information, please see chapter *5 ProSoft Configuration Builder*.

2.1.1 Creating a Project

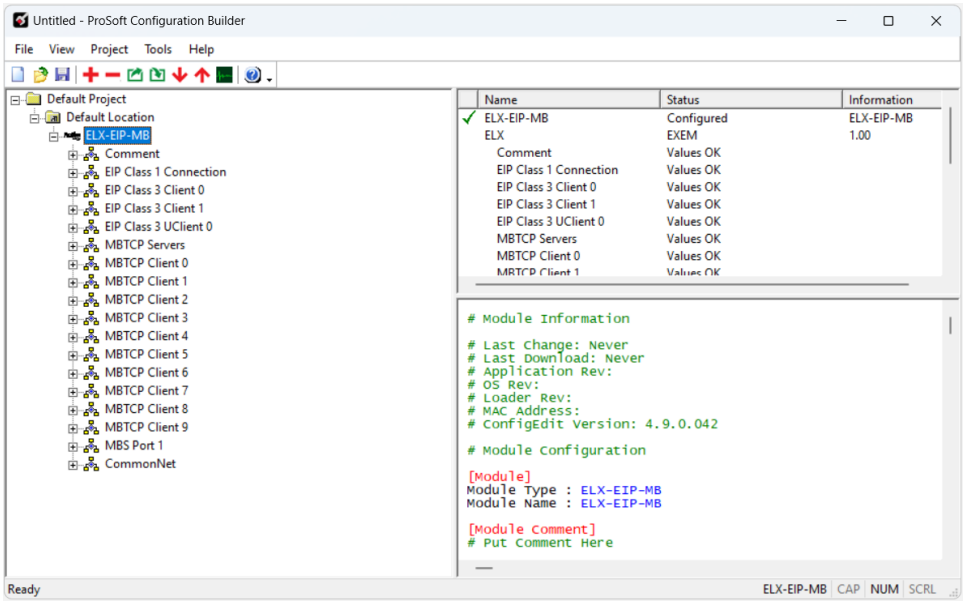
1 Right-click on **DEFAULT MODULE** and select **CHOOSE MODULE TYPE**:



2 At the *Product Line Filter* select **ELX**. Under *Select Module Type* select **ELX-EIP-MB**. Then click **OK**.

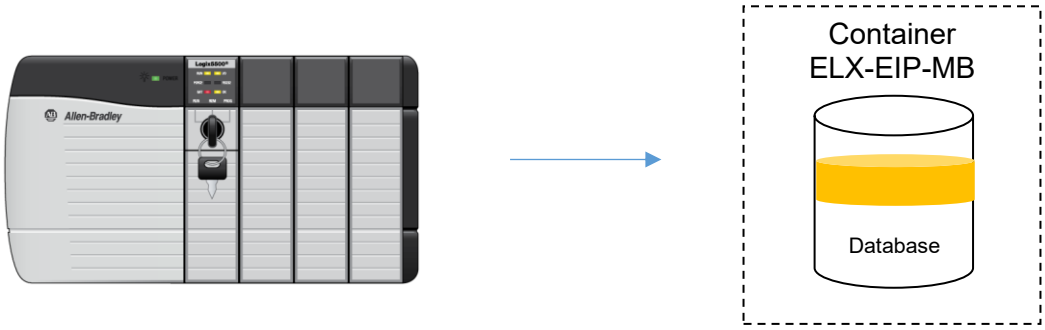


3 The ELX-EIP-MB container project is now ready for configuration.



2.2 Establishing Communications as a Server

The ELX-EIP-MB container can be configured as a Server (Slave). The container will establish communications with the Client (Master) and it will store any received data at a specific data address of the container database.

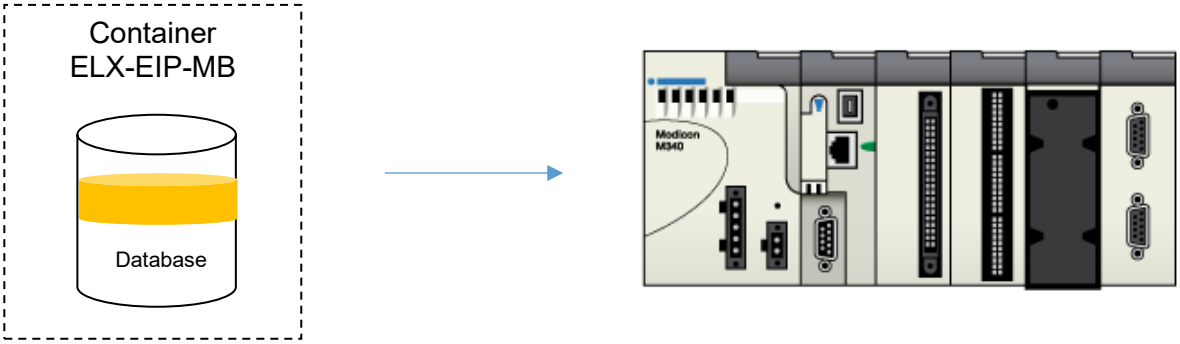


For more information on how to configure the container as a Server/Slave based on the communication protocol used, please see the following sections:

Client Protocol	Client EtherNet/IP Method	Client Role	Container Role	Section
EtherNet/IP	Class 1	Scanner	Adapter	*2.4.1 EtherNet/IP Class 1 Adapter*
EtherNet/IP	Class 3	Client	Server	*2.4.2 EtherNet/IP Class 3 Server*
Modbus TCP	N/A	Client	Server	*2.4.4 Modbus TCP*
Modbus RTU	N/A	Master	Slave	*2.4.6 Modbus RTU*

2.3 Establishing Communications as a Client

The ELX-EIP-MB container can be configured as a Client to communicate with a Server. The container will establish communications with the Server. Using Client commands to exchange data with the Server, the container database is used to store the received and sent data.



For more information on how to configure the container as a Client based on the communication protocol used, please see the following sections:

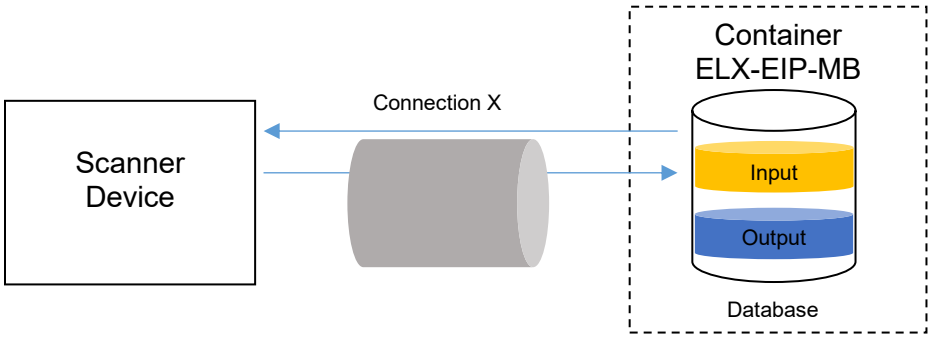
Server Protocol	Server EtherNet/IP Method	Server Role	Container Role	Section
EtherNet/IP	Class 3	Server	Client	*2.4.3 EtherNet/IP Class 3*
Modbus TCP	N/A	Server	Client	*2.4.5 Modbus TCP*
Modbus RTU	N/A	Slave	Master	*2.4.7 Modbus RTU*

2.4 Data Exchange

The following sections describe the functions of ELX-EIP-MB container data exchange using the EtherNet/IP and Modbus TCP/RTU protocols.

2.4.1 EtherNet/IP Class 1 Adapter

The Class 1 data is exchanged between the ELX-EIP-MB container adapter and an EtherNet/IP scanner through implicit connections. The container supports up to 8 connections where each can transfer a maximum of 248 words in each direction. The data exchanged through each connection is stored at the container database input and output sections as follows:



By default, the ELX-EIP-MB container is configured with specific internal I/O memory locations that map to the PLC's Input Image (data received from the PLC) and Output Image (data sent to the PLC) for each connection.

Connection #	Input Start Address	Input Size	Output Start Address	Output Size
1	0	248	2000	248
2	250	248	2250	248
3	500	248	2500	248
4	750	248	2750	248
5	1000	248	3000	248
6	1250	248	3250	248
7	1500	248	3500	248
8	1750	248	3750	248

The ELX-EIP-MB container database configuration for Class 1 data storage can be edited in the *EIP Class 1 Connection X* configuration in PCB. For more information, please see section *7.3.1 EIP Class 1 Connection*.

2.4.1.1 Example 1

Consider a ControlLogix controller transferring the first two data bytes to the ELX-EIP-MB container through Connection #1 using class 1 communication:

ELX_EIP_MB:O1	{...}
ELX_EIP_MB:O1.Data	{...}
ELX_EIP_MB:O1.Data[0]	1
ELX_EIP_MB:O1.Data[1]	2

The container will automatically store the received data at database word address 2000 (since *Output Start Address* for connection #1 is set as **2000** by default).

2.4.1.2 Example 2

Consider a ControlLogix controller receiving the first two data bytes of Connection #1 from the ELX-EIP-MB container using class 1 communication. The container will send the data stored at database address **0** (word size) to the controller tags below. The reason for using database address **0** is because the *Input Start Address* for connection #1 is set as **0** by default.

ELX_EIP_MB:I1.Data	{...}
ELX_EIP_MB:I1.Data[0]	5
ELX_EIP_MB:I1.Data[1]	6

2.4.2 EtherNet/IP Class 3 Server

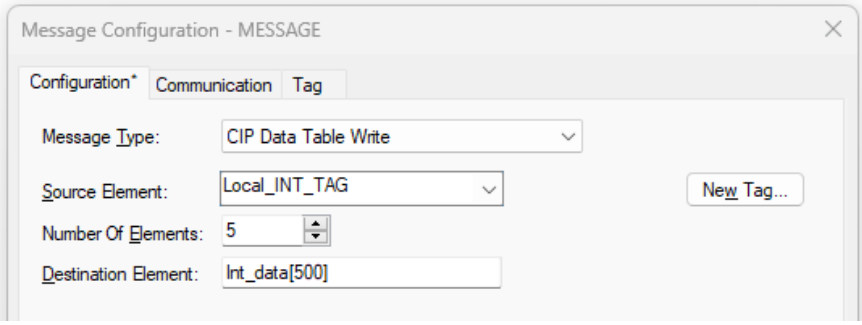
There is no user configuration required for this application since the ELX-EIP-MB container automatically operates as an EtherNet/IP Class 3 Server to exchange data with an EtherNet/IP Class 3 Client.

When the container operates as an EtherNet/IP Class 3 Server, the container database is automatically mapped to the following tag names to provide data access for the EtherNet/IP Class 3 device:

Database Address	CIP Integer Tag	CIP Boolean Tag	CIP Bit Array Tag	CIP SINT Tag	CIP DINT Tag	CIP Real Tag
0	INT_Data[0]	BOOLData[0]	BITAData[0]	SINTData[0]	DINTData[0]	REALData[0]
999	INT_Data[999]	BOOLData[15984]		SINTData[1998]		
1000	INT_Data[1000]	BOOLData[16000]	BITAData[500]	SINTData[2000]	DINTData[500]	REALData[500]
1999	INT_Data[1999]	BOOLData[31984]		SINTData[3998]		
2000	INT_Data[2000]	BOOLData[32000]	BITAData[1000]	SINTData[4000]	DINTData[1000]	REALData[1000]
2999	INT_Data[2999]	BOOLData[47984]		SINTData[5998]		
3000	INT_Data[3000]	BOOLData[48000]	BITAData[1500]	SINTData[6000]	DINTData[1500]	REALData[1500]
3999	INT_Data[3999]	BOOLData[63999]		SINTData[7998]		
9999	INT_Data[9999]	BOOLData[159984]		SINTData[19998]		

2.4.2.1 Example

Consider an EtherNet/IP Class 3 Client device sending 5 words of data to the ELX-EIP-MB container through the following CIP message:



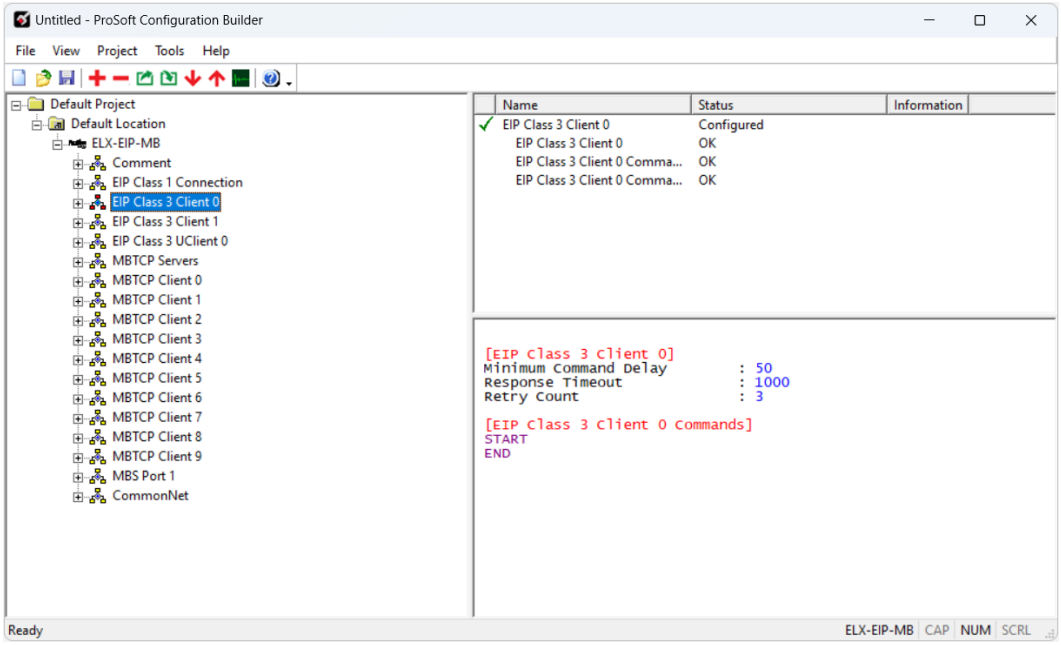
The ELX-EIP-MB container will store the received data at the following container database addresses:

Class 3 Client Device Source Tags	Container Destination Tags	Container Database Address
Local_INT_TAG[0]	INT_Data[500]	500
Local_INT_TAG[1]	INT_Data[501]	501
Local_INT_TAG[2]	INT_Data[502]	502
Local_INT_TAG[3]	INT_Data[503]	503
Local_INT_TAG[4]	INT_Data[504]	504

2.4.3 EtherNet/IP Class 3 Client

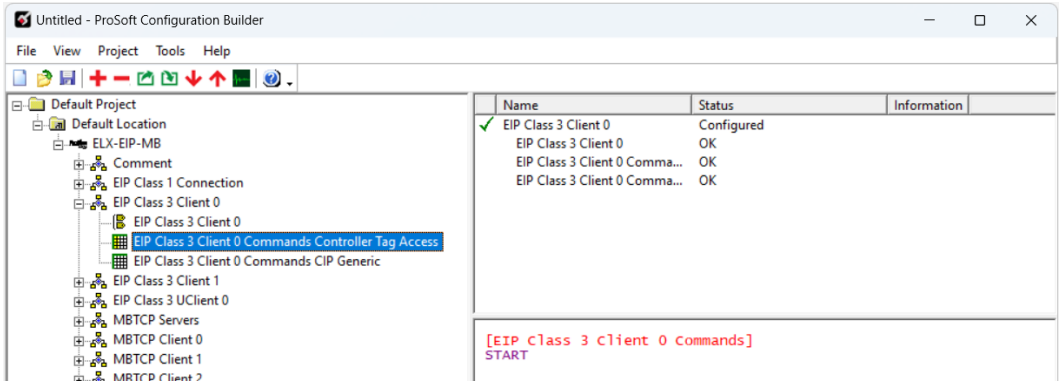
The following steps will configure the ELX-EIP-MB container EtherNet/IP Class 3 Client to exchange data with an EtherNet/IP Class 3 Server.

1 In PCB refer to the **EIP CLASS 3 CLIENT X** section (connected Client configuration) or **EIP CLASS 3 UCLIENT 0** (unconnected Client configuration).

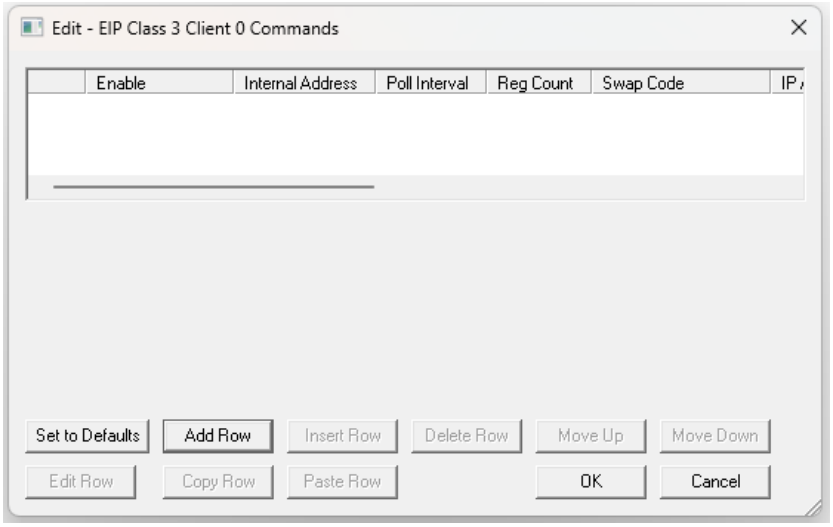


2 To configure a CIP Data Table Read/Write command, double-click on **EIP CLASS 3 CLIENT X COMMANDS CONTROLLER TAG ACCESS**.

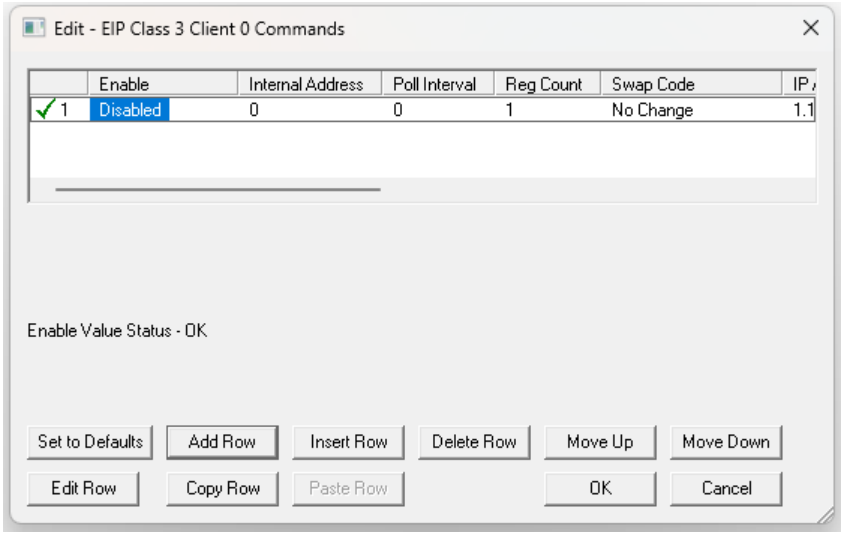
For unconnected Client commands, double-click on **EIP CLASS 3 UCLIENT 0 COMMANDS CONTROLLER TAG ACCESS.**



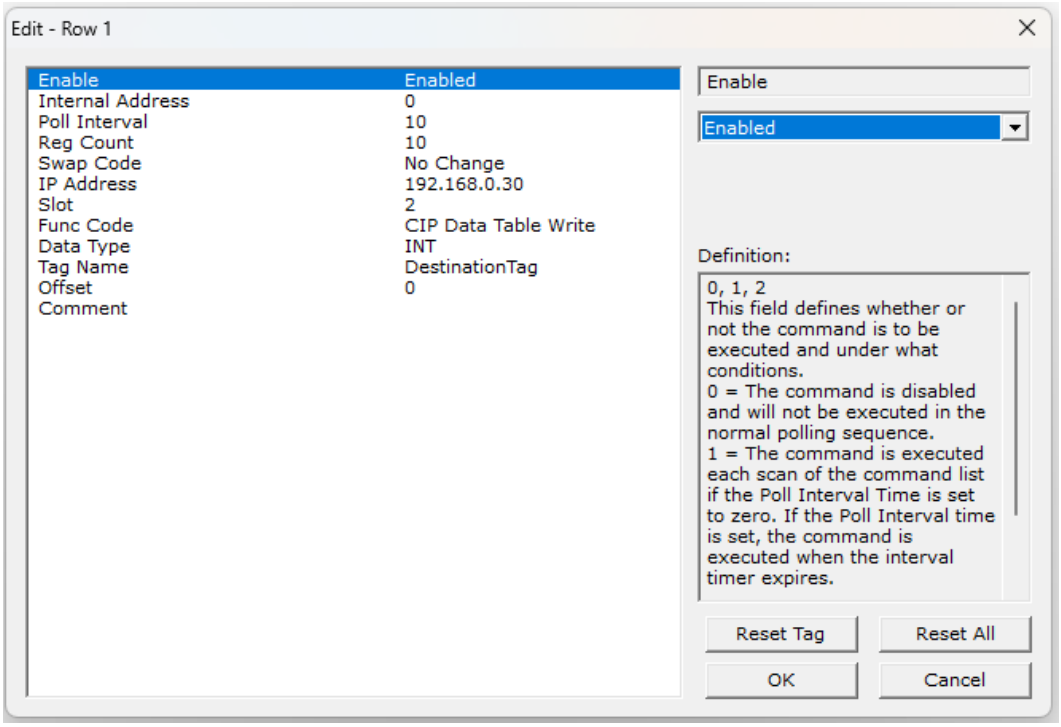
3 In the *Edit – EIP Class 3 Client X Commands* (or *Edit – EIP Class 3 UClient 0 Commands*) dialog, click **ADD ROW** to create a new command.



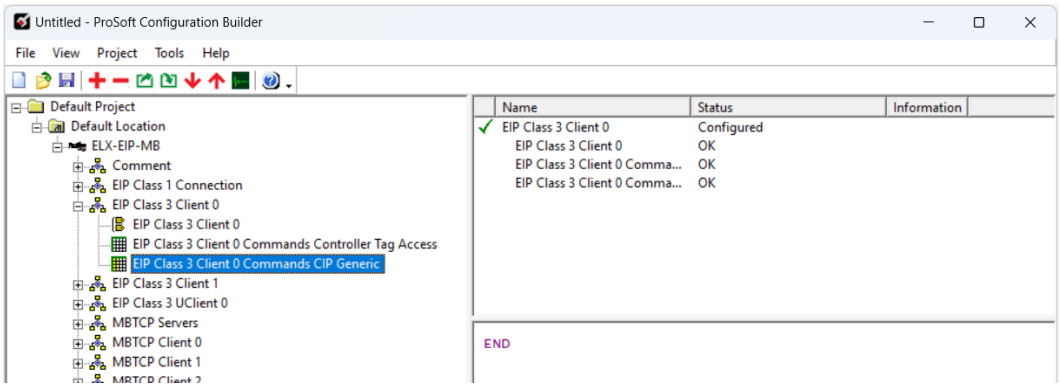
4 To edit the command parameters, double-click on the command or click on **EDIT ROW**.



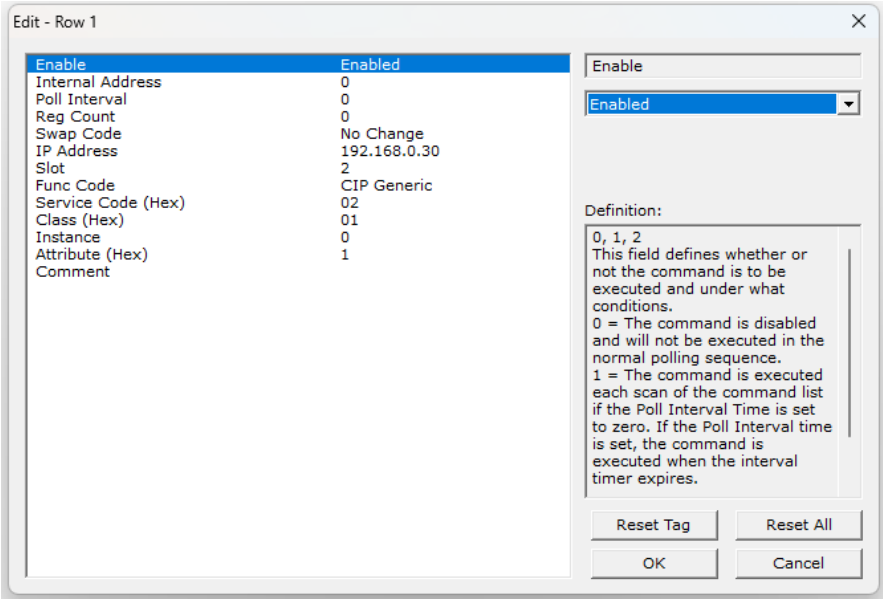
5 Edit the parameter values for the CIP data table command. The example below will configure the ELX-EIP-MB container to write **10** integer values (stored at *Internal Addresses* **0** to **9**) to the Server IP **192.168.0.30** (controller tags *DestinationTag[0]* through *DestinationTag[9]*). The command will continuously be sent every second.



6 To create a Generic CIP Class 3 Client 0 command, refer to the **EIP CLASS 3 CLIENT X COMMANDS CIP GENERIC** section. For an unconnected Client configuration, refer to the **EIP CLASS 3 CLIENT 0 COMMANDS CIP GENERIC** section.



7 The procedure is the same as previously covered for the Controller Tag Access commands, except the generic CIP parameters (*service*, *class*, *instance*, *attribute*) are required instead of the CIP Data Table Read/Write parameters.



8 Refer to section *7.3.2 EIP Class 3 Client [x] Connection* for more information on the command parameters.

2.4.4 Modbus TCP Server

The ELX-EIP-MB container automatically operates as a Modbus TCP Server to exchange data with a Modbus TCP Client.

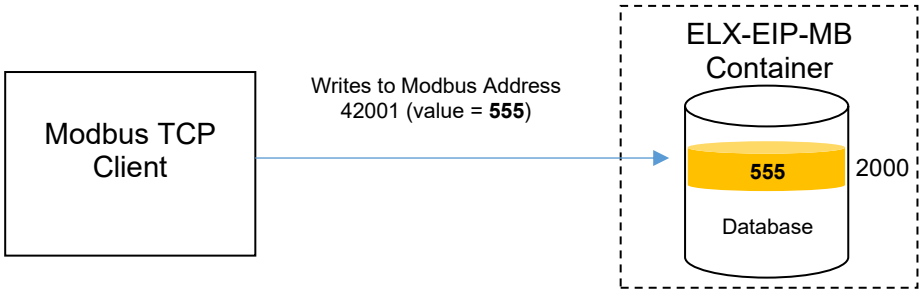
When the container operates as a Modbus TCP Server, the database is mapped by default to the following Modbus addresses to provide data access for the Modbus TCP Client:

Container Database Word Address	Read Modbus Input Register Address (Function Code 4)	Read Modbus Holding Register Address (Function Code 3)
0	30001	40001
1000	31001	41001
2000	32001	42001
3000	33001	43001
…	…	…
9998	39999	49999

Container Database Bit Address	Read Output Coils (Function Code 1)	Read Discrete Inputs (Function Code 2)
0	00001	10001
1000	01001	11001
2000	02001	12001
3000	03001	13001
…	…	…
9998	09999	19999

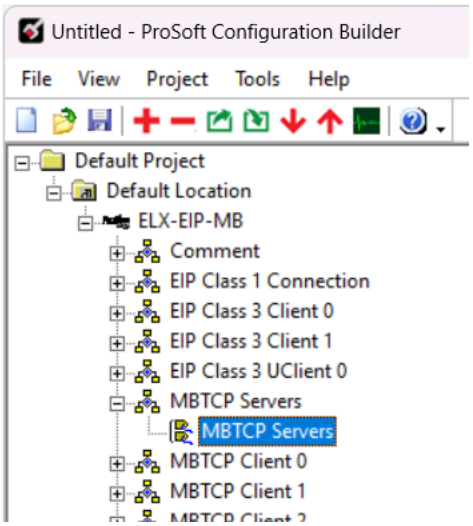
2.4.4.1 Example

Consider a Modbus TCP Client device that uses Function Code 6 or 16 to write a value of '**555**' to Modbus address 42001 in the ELX-EIP-MB container. The received value will be stored at database address 2000 in the ELX-EIP-MB container:



The user can optionally configure a database offset to the database mapping, along with timing parameters associated with the Modbus TCP Server in the *MBTCP Servers* section in PCB.

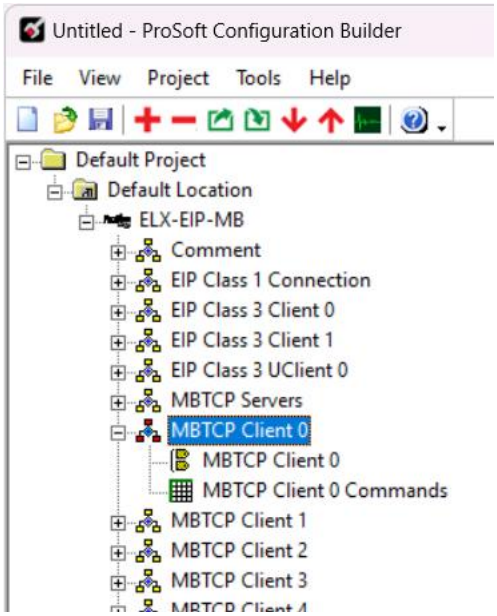
For more information, please see section *8.3.1 MBTCP Servers*.



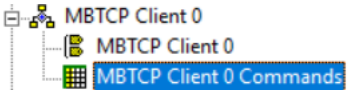
2.4.5 Modbus TCP Client

This section configures the ELX-EIP-MB container as a Modbus TCP Client to exchange data with a Modbus TCP Server.

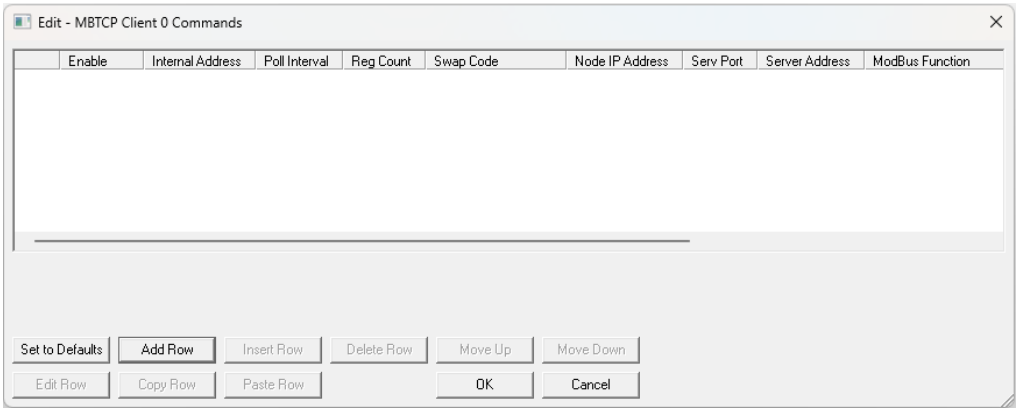
1 In PCB refer to the MBTCP *Client X* section for the Modbus TCP Client configuration.



2 Double-click on **MBTCP CLIENT X COMMANDS** to configure the commands for data exchange between the ELX-EIP-MB container and the Modbus TCP Server.

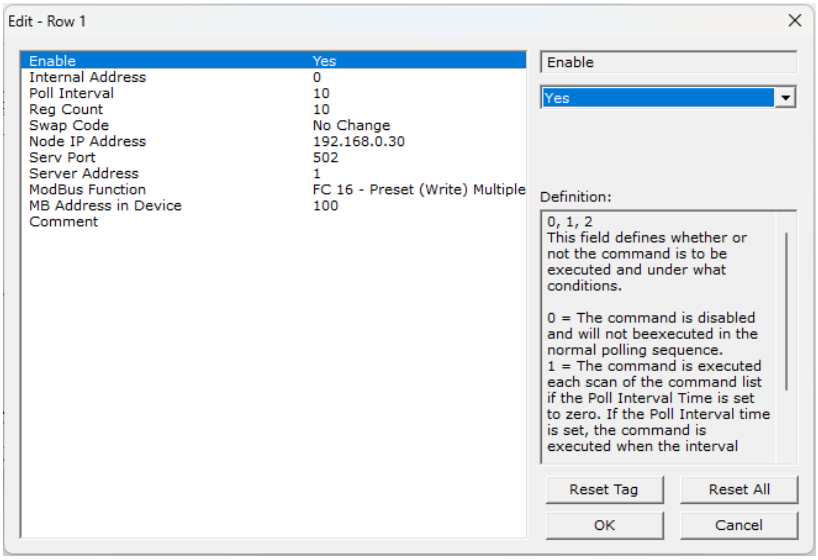


3 In the *Edit – MBTCP Client X Commands* dialog, click on **ADD ROW** to create a new command. Then double-click on the command or click on **EDIT ROW**.



4 The example below configures the ELX-EIP-MB container Modbus Client to write **10** holding register values stored at container database addresses **0** to **9** to the Modbus TCP Server IP **192.168.0.30** starting at address 40101. The command will be sent continuously every second.

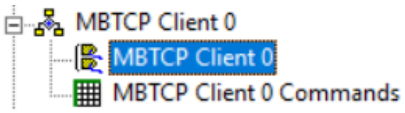
Refer to section *8.3.3 MBTCP Client [x] Commands* for more information on the command parameters.



5 Click on **OK** to confirm the command.

6 Repeat the same procedure to create additional commands for data exchange between the ELX-EIP-MB container and the Modbus TCP Server.

7 Modbus TCP Client timing and other advanced parameters are available through the *MBTCP Client X* section in PCB. For more information, please see section *8.3.2 MBTCP Client [x]*.



8 Save and download the configuration to the Container. For more information, see section *5.3 Downloading the Project to the Container*.

9 Diagnostics can be accessed in ProSoft Configuration Builder. For more information, see section *5.5.1 Diagnostics Menu*.

2.4.6 Modbus RTU Slave

The ELX-EIP-MB container can be configured as a Modbus RTU Slave to exchange data with a Modbus RTU Master.

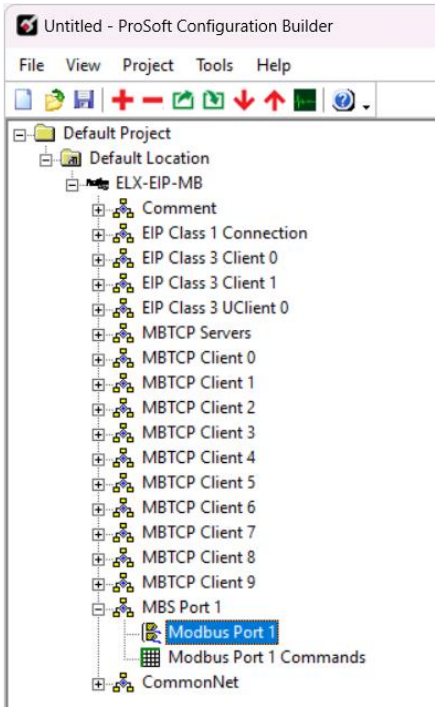
When the container operates as a Modbus RTU Slave, the database is mapped to the following Modbus addresses to provide data access for the Modbus RTU Master:

Container Database Word Address	Read Modbus Input Register Address (Function Code 4)	Read Modbus Holding Register Address (Function Code 3)
0	30001	40001
1000	31001	41001
2000	32001	42001
3000	33001	43001
…	…	…
9998	39999	49999

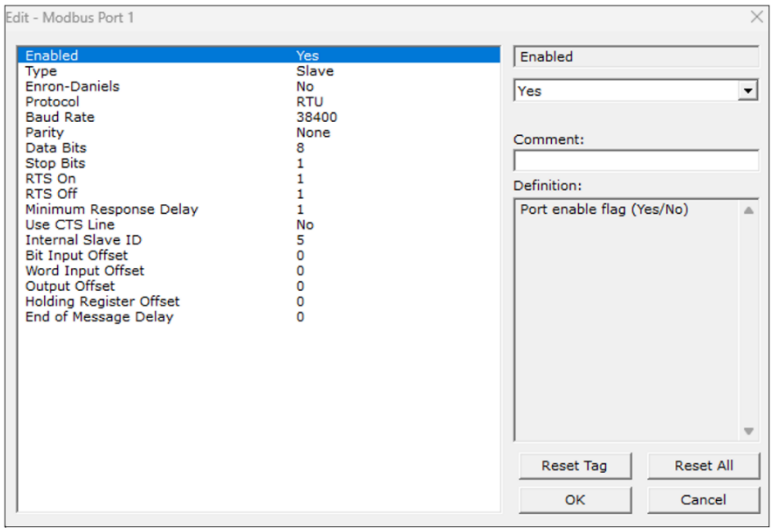
Container Database Bit Address	Read Output Coils (Function Code 1)	Read Discrete Inputs (Function Code 2)
0	00001	10001
1000	01001	11001
2000	02001	12001
3000	03001	13001
…	…	…
9998	09999	19999

To configure the ELX-EIP-MB container as a Modbus RTU Slave:

1 In PCB double-click on the *MBS Port 1 > Modbus Port 1* section.



2 In the *Edit - Modbus Port 1* dialog, enable port 1 by setting the *Enabled* parameter to **YES**.

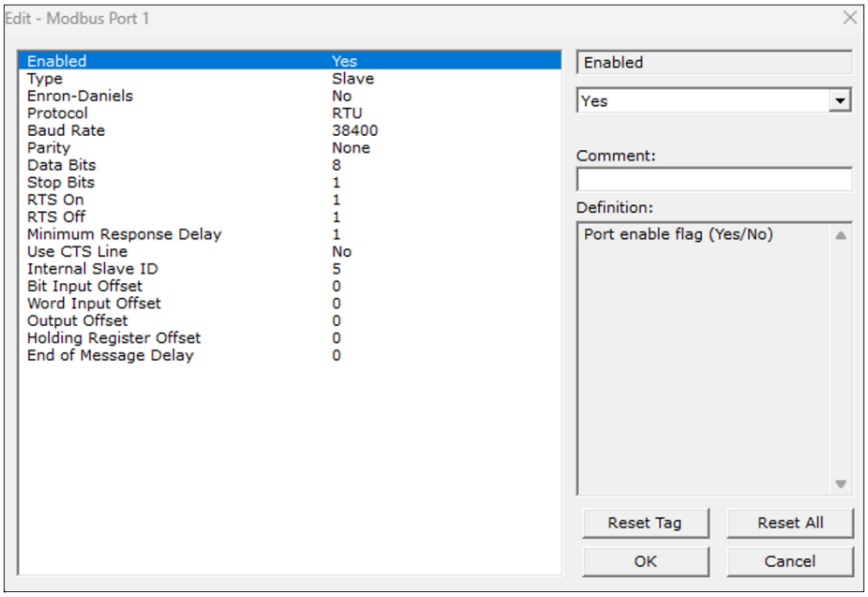


3 Set the *Type* parameter to **SLAVE** (default).

4 Set the *Protocol* parameter to **RTU** (default).

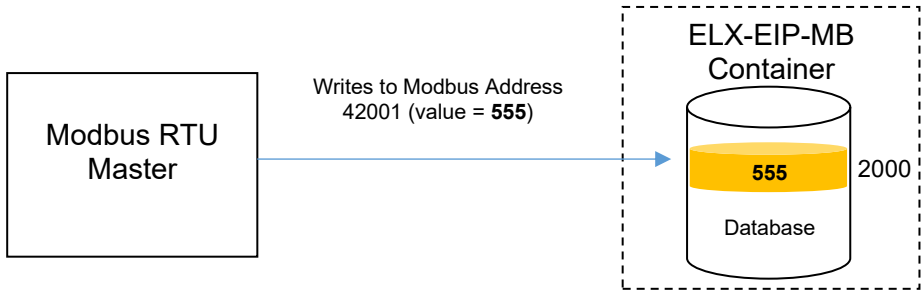
5 Edit the other serial settings as needed. The following configures the ELX-EIP-MB container with a Baud Rate of 38400bps, No Parity, 8 Data Bits, 1 Stop Bit, and an Internal Slave ID of 5.

2.4.6.1 Example



Note: When using an ELX3, the *Port Mode* must be selected/configured in the ELX3 UI.

Consider a Modbus RTU Master device that uses Function Code 6 or 16 to write a value of '**555**' to Modbus address 42001 in the ELX-EIP-MB container. The received value will be stored at database address 2000 in the ELX-EIP-MB container:



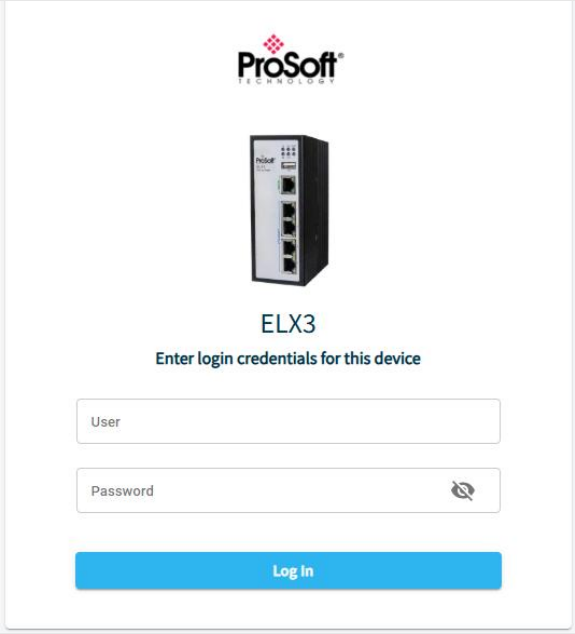
The user can optionally configure a database offset to the database mapping, along with timing parameters associated with the Modbus RTU Slave in the *Modbus Port 1* section in PCB.

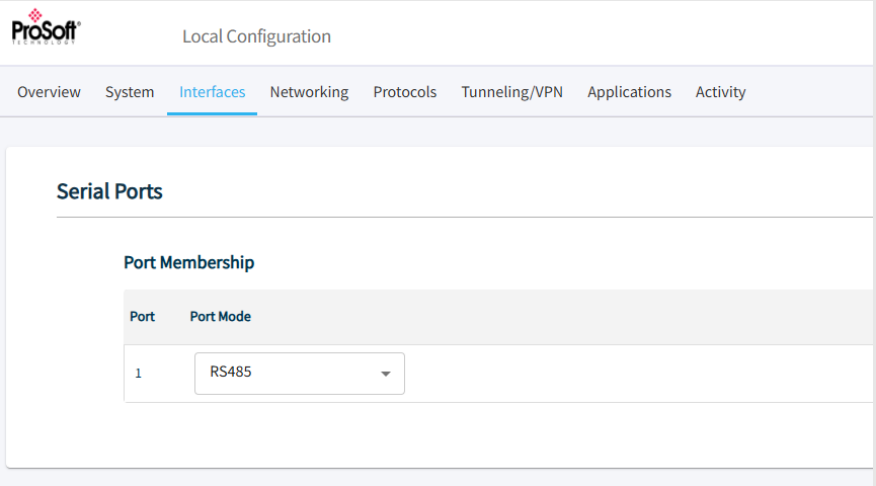
For more information, please see section *9.3.1.3 MBS Slave Configuration Parameters*.

2.4.7 Modbus RTU Master

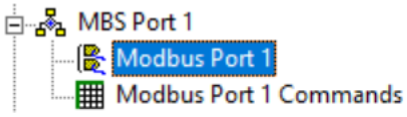
This section configures the ELX-EIP-MB container as a Modbus RTU Master to exchange data with a Modbus RTU Slave.

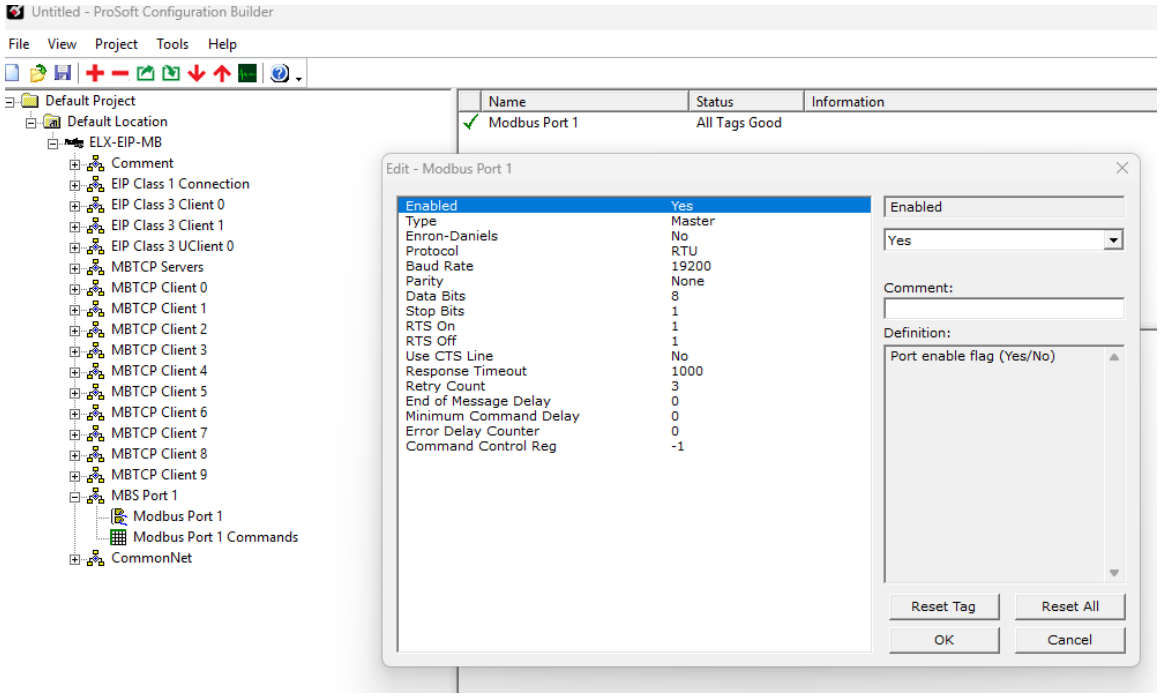
Note: In the ELX3 UI, ensure the *Port Mode* parameter is configured.



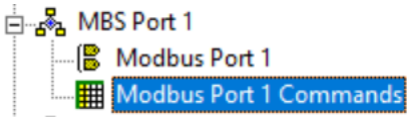


1 In PCB double-click on the *MBS Port 1 > Modbus Port 1* section to configure the serial port connection parameters.

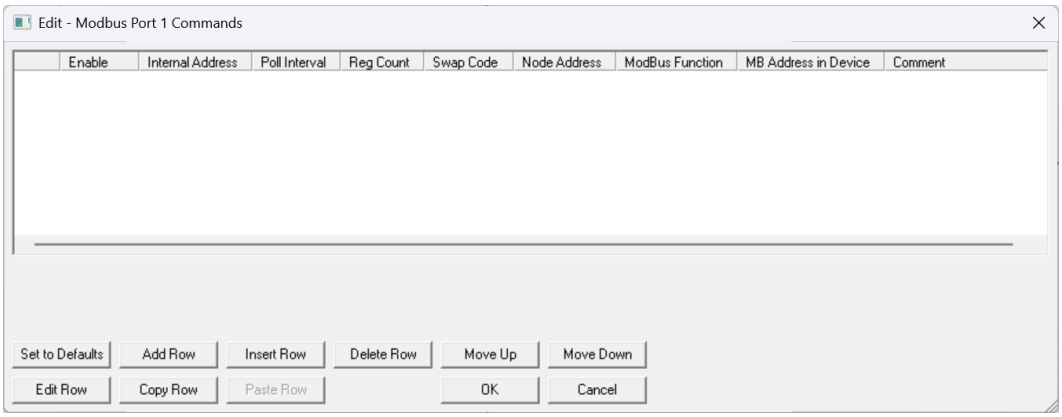




2 Double-click the *MBS Port 1 > Modbus Port 1 Commands* section to configure the commands for data exchange between the ELX-EIP-MB container and the Modbus RTU Slave.

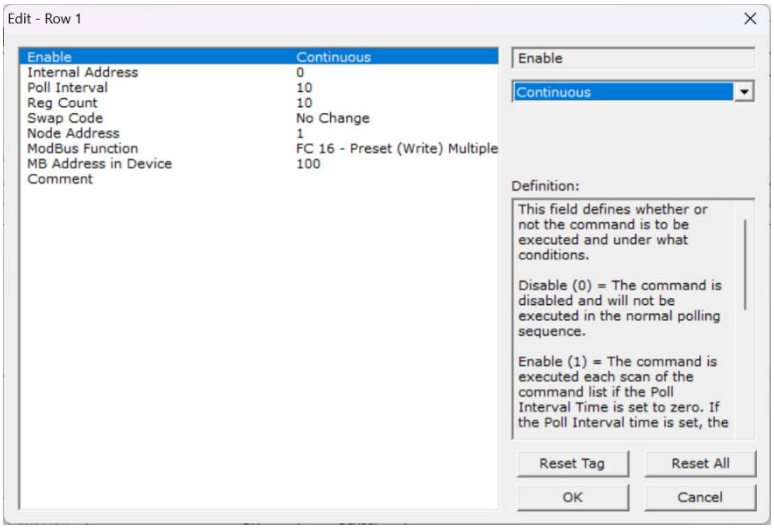


3 In the *Edit – Modbus Port 1 Commands* dialog, click on **ADD ROW** to create a new command. Then double-click on the command or click on **EDIT ROW**.



4 The example below configures the ELX-EIP-MB container MBS Master to write **10** holding register values stored at container database addresses **0** to **9** to the Modbus RTU Slave node 1, starting at address 40101. The command will be sent continuously every second.

Refer to section *9.3.2 MBS Port 1 Commands* for more information on the command parameters.



5 Click on **OK** to confirm the command.

6 Repeat the same procedure to create additional commands for data exchange between the ELX-EIP-MB container and the Modbus RTU Slave(s).

7 Save and download the configuration to the Container. For more information, see section *5.3 Downloading the Project to the Container*.

8 Diagnostics can be accessed in ProSoft Configuration Builder. For more information, see section *5.5.1 Diagnostics Menu*.

3 Container Deployment through the Local User Interface

3.1 Container Network Configuration

An ELX3 gateway container network must be configured before deploying the container.

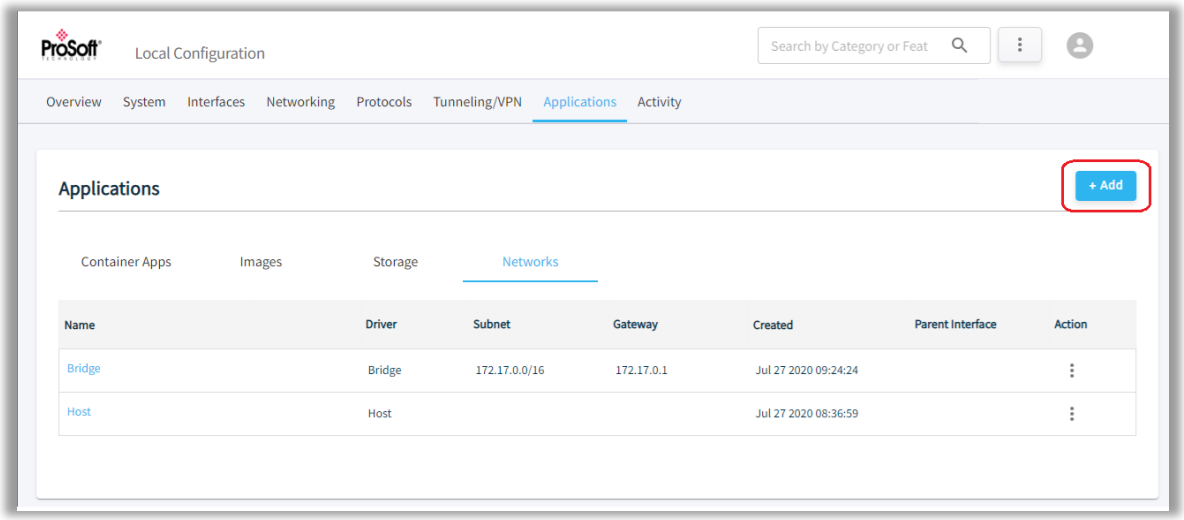
A network is the functionality that allows a container to communicate to external devices by using the host's configured LANs. This section concentrates specifically on the virtual network between containers also known as Docker networks.

A Docker network is a powerful feature that enables containers to communicate with each other and the outside world. It provides isolated and secure networking environments, allowing seamless connectivity and easy management of containerized applications.

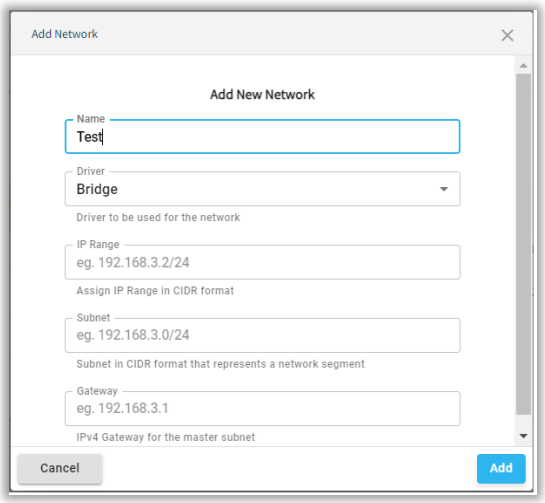
3.1.1 Adding a Network

Note: If the desired network subnets differ from the gateway's default network configuration, ensure the gateway's LAN interfaces are configured before continuing.

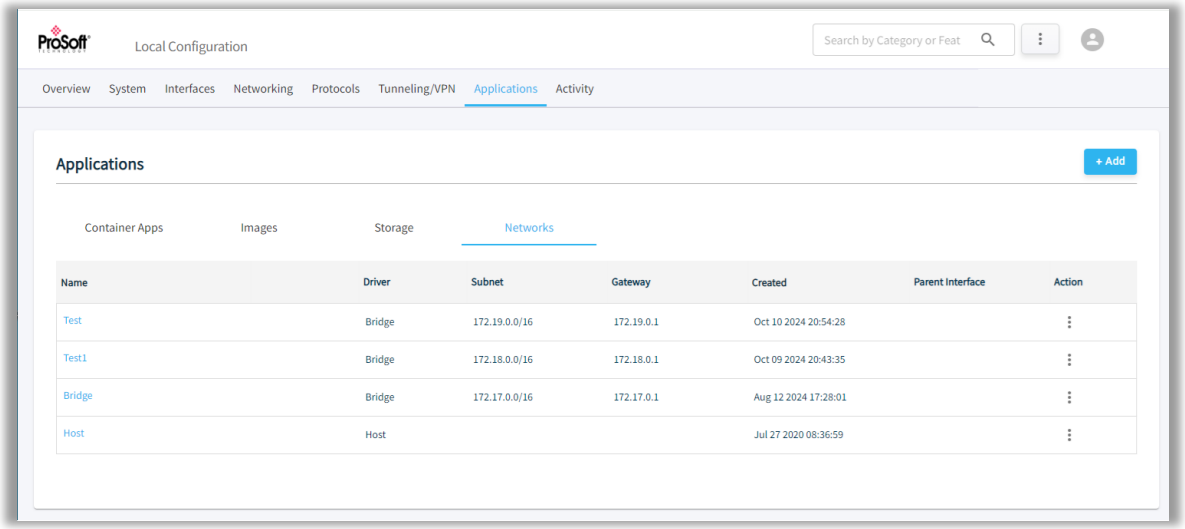
1 In the *Applications* tab > *Networks* tab and click the **ADD** button.



2 In the *Add Network* dialog, enter the network information then click the **ADD** button.



3 The new network will display in the refreshed *Networks* tab.

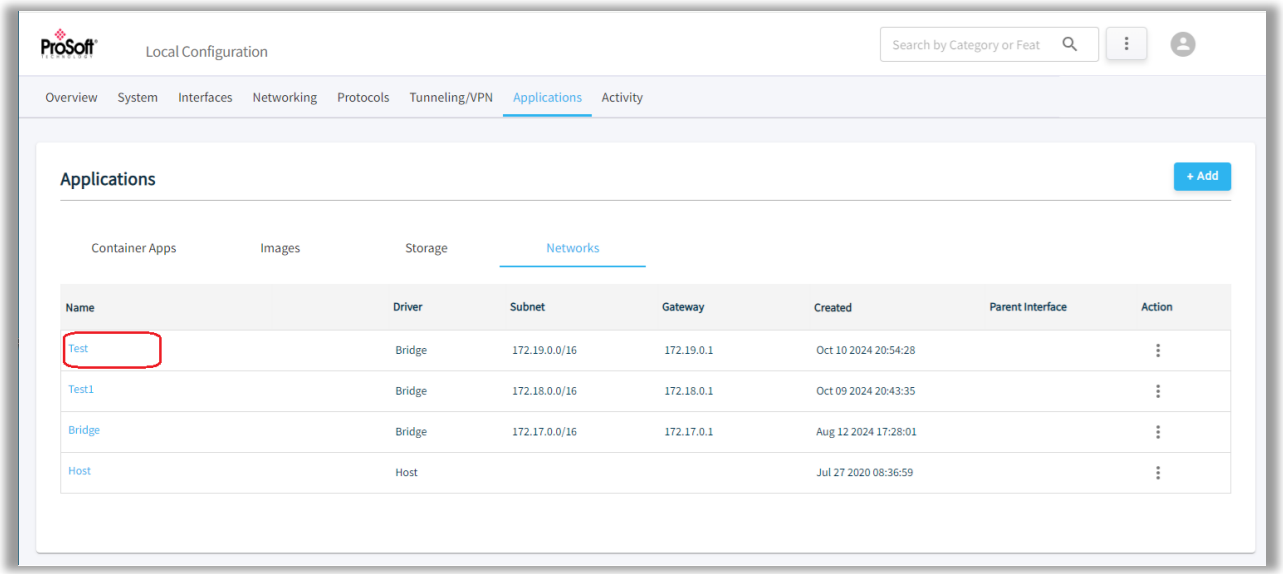


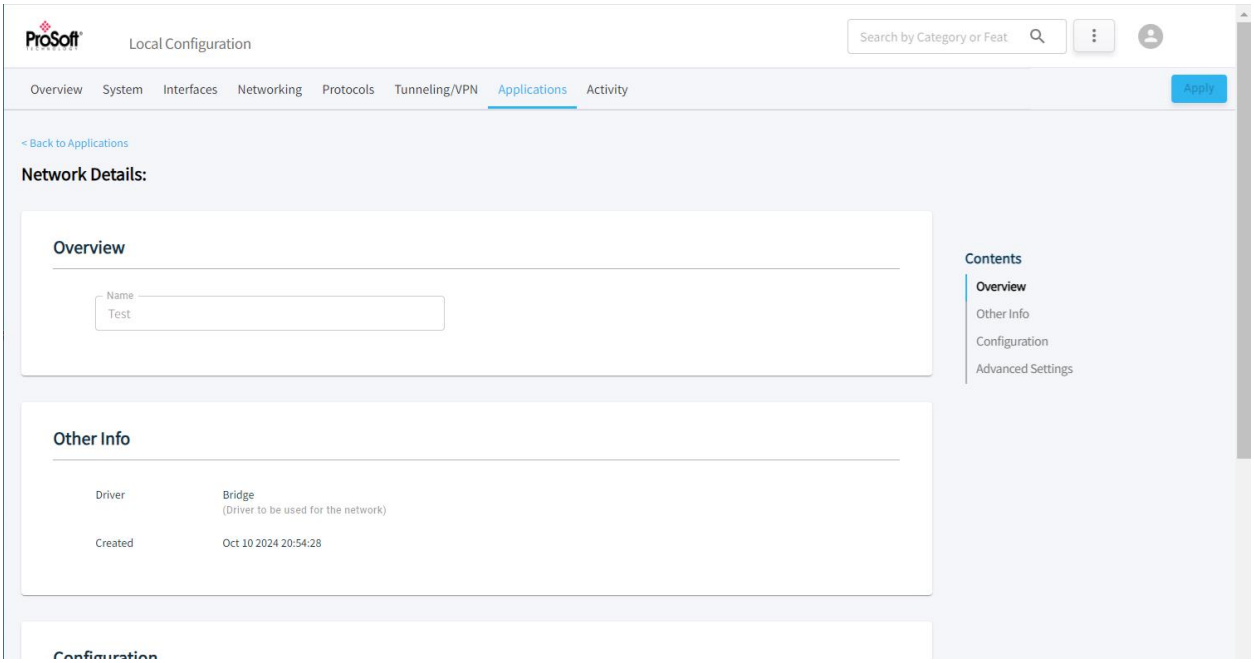
Note: The *Network* tab has two default networks - *Bridge* and *Host*. These two networks cannot be deleted.

Parameter	Description
Name	Name of the network. **Note:** Create a network name with an alphanumeric character with a minimum length of 2 and maximum length of 49. The following characters are allowed: a to z A to Z 0 to 9 Only Special character “_” is allowed in network name creation.
Driver	The Driver used for the network. Select **MACVLAN** or **Bridge**
Parent Interface	LAN port to be used for MACVLAN network. **Note:** MACVLAN network only
IP Range	IP address range for the master IP address in CIDR (Classless Inter Domain Routing) format.
Subnet	Subnet range for the master subnet in CIDR (Classless Inter Domain Routing) format.
Gateway	IP address of the gateway associated with master subnet in IPv4 format.
Created	Time stamp of network creation.
Action	Delete the network using this parameter.

3.1.1.1 Network Details

In the local UI, clicking on the network name opens the *Network Details* page.





Parameter		Description
Overview		
	Name	Name of the network.
Other Info		
	Driver	The driver used for the network.
	Created	Date of network creation.
Configuration		
	IP Range	IP Range in CIDR format
	Parent Interface	Interface on the device
	Subnet	Subnet in CIDR format
	Gateway	IPv4 Gateway for the master subnet
Advanced Settings		**Details of the Container Utilizing the Network**
	Name	Name of the container using the network.
	IP Address	IP Address assigned to the container.
	MAC Address	MAC address assigned to the container.

3.2 Container Storage Configuration

A container volume allows data to persist, even when a container is deleted. Volumes are also a convenient way to share data between two or more containers.

Note: Volume size is dynamic and subject to host storage.

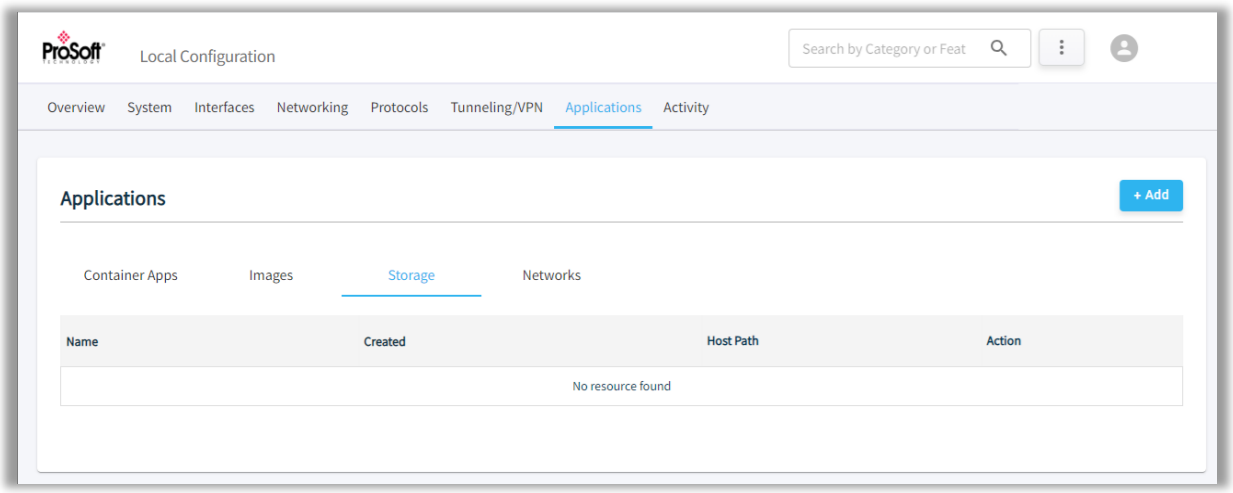
From the container, the volume acts like a folder to store and retrieve data. The volume can be mounted on the container directory. If a container is recreated due to a failure, a reboot, a new release, or any other reason, the volume data will not be lost.

For volume deletion, a scheduler will run every 5 minutes to check the consumed volume space when it exceeds 90% of the reserved space.

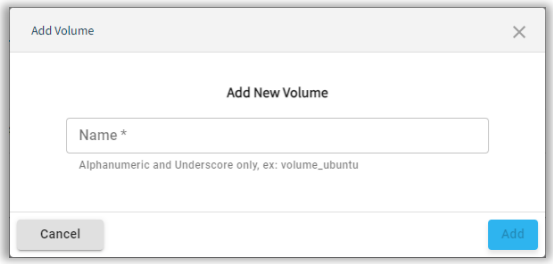
3.2.1 Adding a Storage Volume (optional)

The storage volume acts as persistent storage for a container and may not be required.
If more storage volumes are required, create additional volumes here.

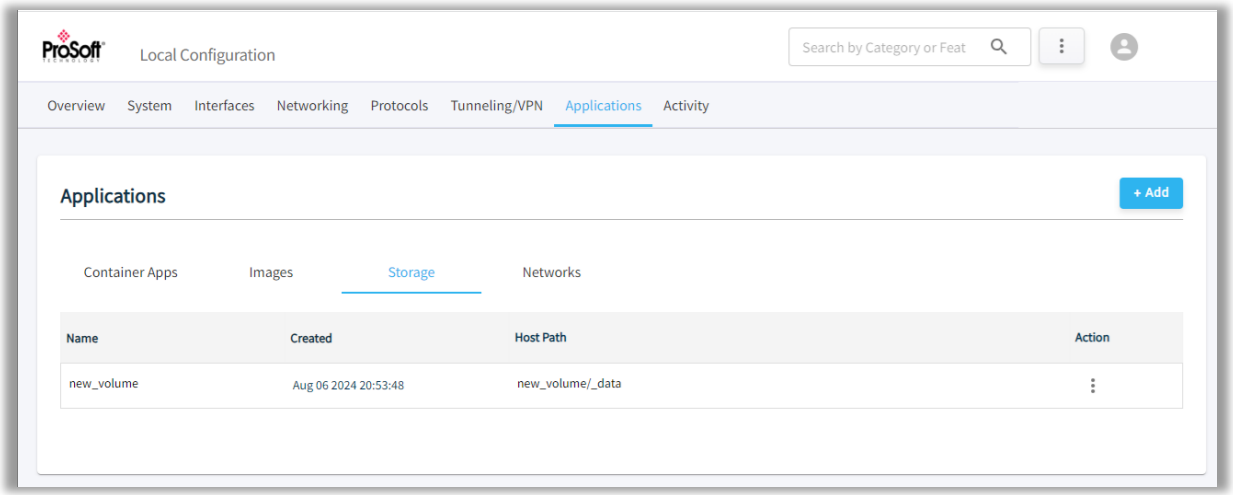
1 Navigate to the *Applications* tab > *Storage* tab and click on the **ADD** button.



2 Enter name of the volume in the *Name* field, then click **ADD**.



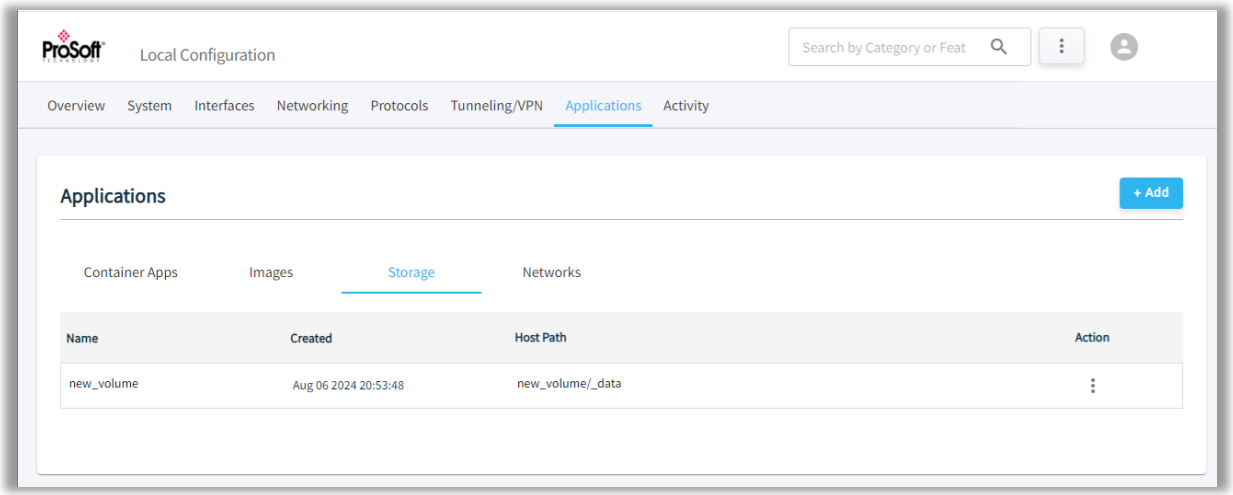
3 The list of *Volumes* is updated.



3.2.2 Deleting a Volume

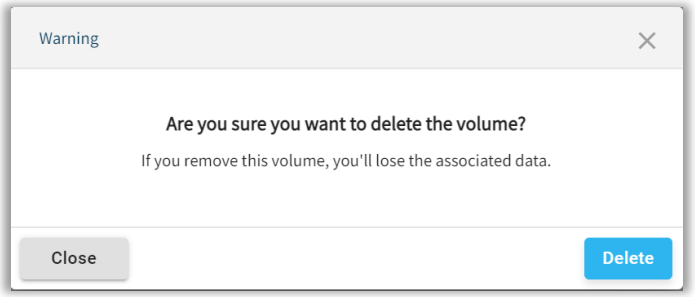
To delete a volume:

1 Click on the *Action* button ⋮.



2 Click the **DELETE** option.

3 The user will be asked for confirmation.



4 Click **DELETE** to confirm.

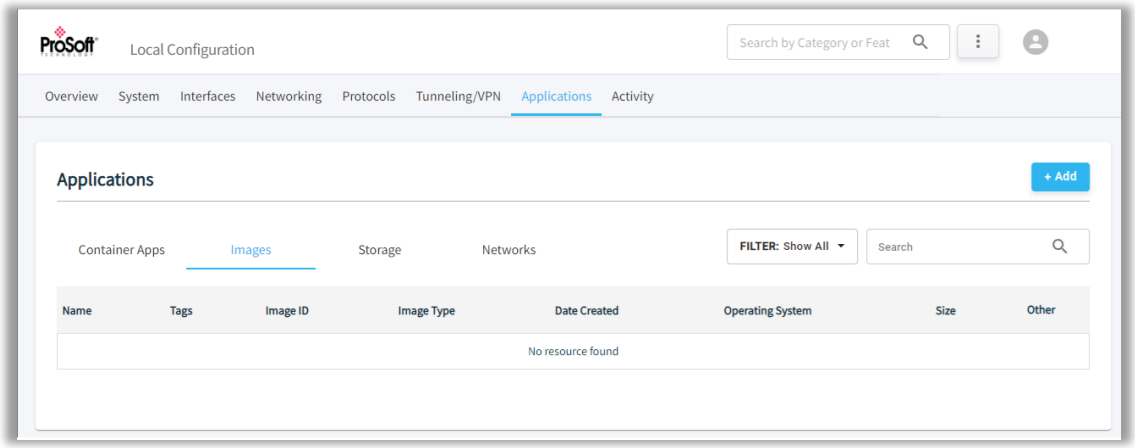
3.3 Deploying Container

This section covers the deployment of the Container.

3.3.1 Importing the Image

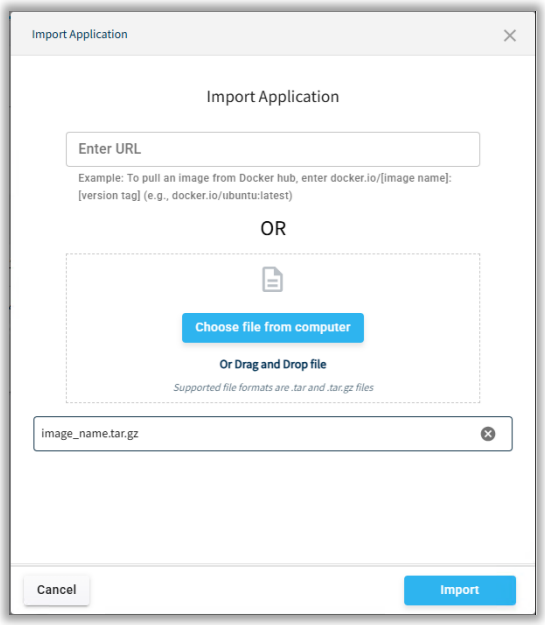
Perform the following steps to import the docker image (.tar or .tar.gz) to the ELX3 environment.

1 Navigate to the *Applications* tab > *Images* tab.



2 Click the **+ ADD** button to open the *Import Application* wizard.

3 In the *Import Application* window, click on **CHOOSE FILE FROM COMPUTER** and select the docker image from the local PC.



4 Click **IMPORT** to add image.

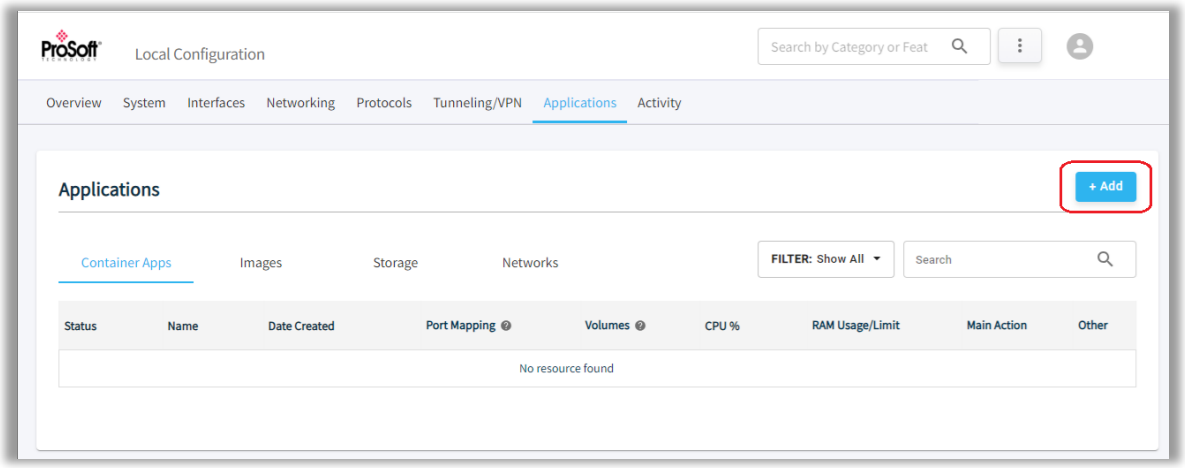
3.3.2 Adding the Application

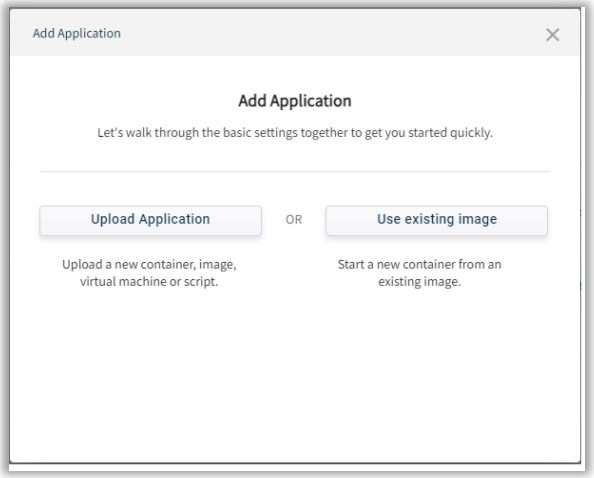
There are two ways to deploy the Application to a target device.

- Uploading the Application (Section *3.3.2.1 Upload Application*)
- Using an Existing Image (Section *3.3.2.2 Use Existing Image*)

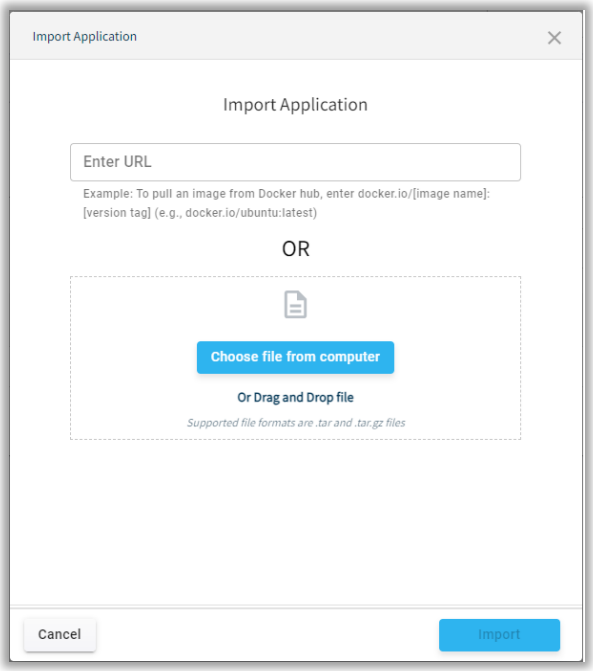
3.3.2.1 Upload Application

1 Navigate to the *Applications* tab > *Container Apps* tab and click the **ADD** button to open the *Add Application* wizard.

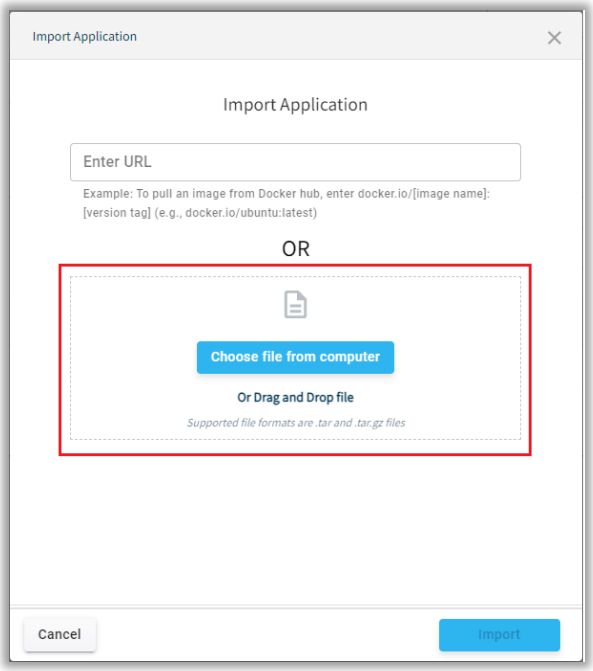




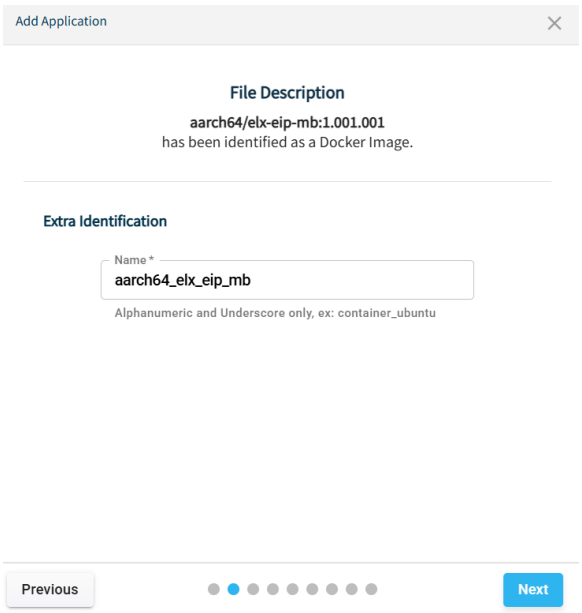
2 Click on the **UPLOAD APPLICATION** button to open the *Import Application* wizard. The application can be imported by selecting an existing file.



3 Select the **.tar** or **.tar.gz** file and click **Import**.



4 After the image file has been successfully imported, enter a name of the container in the *Name* field. Click **NEXT**.



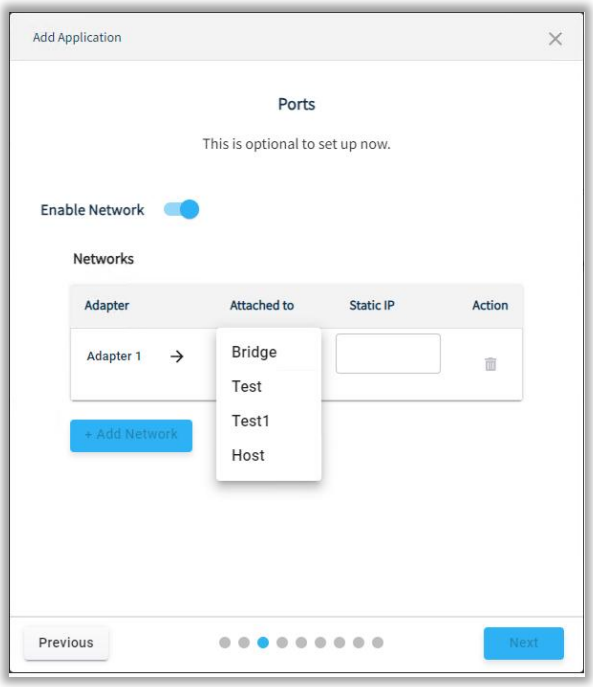
Note: The user can create a container name with an alphanumeric character with a minimum length of 1 and a maximum length of 49.

The following characters are allowed:
a to z
A to Z
0 to 9
Only Special character "_" is allowed for container name creation.

5 The **Ports** configuration defines the network configuration. Select an option for attaching the network adapter to the container in the *Attached to* parameter and click **Next**.

- Bridge
- Host
- User-created custom network (MACVLAN/Bridge)

The user can also enter the (optional) Static IP corresponding to the selected network in the *Static IP* field.



Note: The user must first create the custom network to be able to create a container using that network. Detailed information regarding the creation of a custom network can be found in section *4.1.1 Adding a Network*.

Note: A maximum of four network adapters can be added.

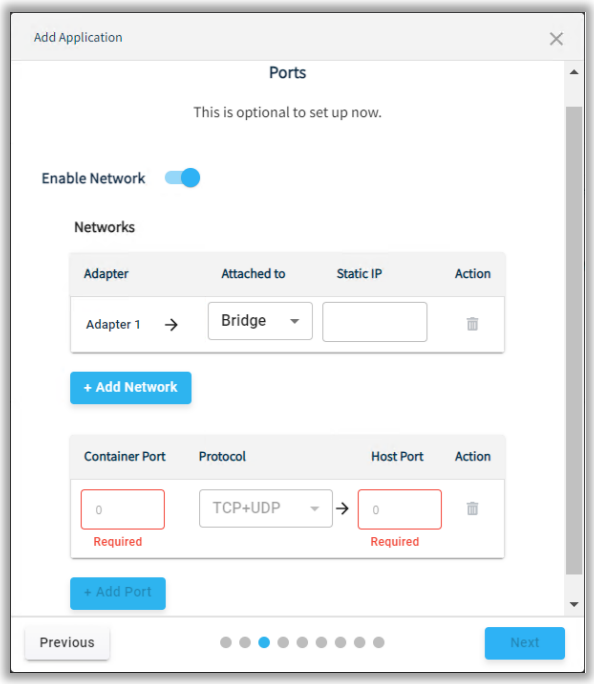
a) For *Bridge* networks, the container and host ports must be configured.

i. In the *Container Port* field, enter the container port number.

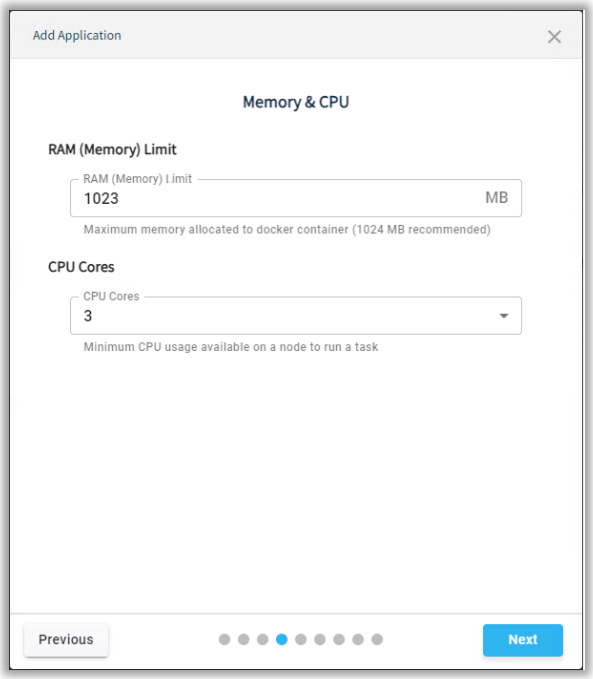
ii. In the *Host Port* field, enter the host port number.

Note: A maximum of four Container and Host ports can be added.

The user is not allowed to create a container without a Container port and Host port in **Bridge** mode. A minimum of one Docker and Host port is required to create a container with a Bridge network.



6 The **Memory & CPU** configuration defines the memory and CPU. Click **NEXT**.



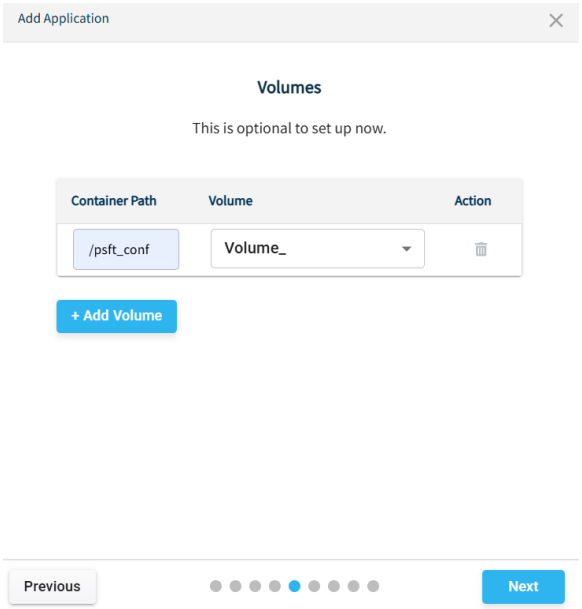
- In the *RAM (Memory) Limit* field, enter the size of memory (MB) for the container. The minimum memory value for creating containers is 4MB.
- In the *CPU Cores* field, enter the number of CPU cores to be used by the container. The number of processors is expressed in number of physical CPU cores.

Note: When configuring an ELX-EIP-MB container and 2 communication protocols are utilized in the application, then 1 CPU Core is needed. If all 3 communication protocols are utilized, then 2 CPU Cores are needed.

7 (Optional) In the **Volumes** configuration, enter `/psft_conf` in the *Container Path* and select the *Volume* from an existing list to attach to the container. Click **NEXT**.

Note: `/psft_conf` is the internal Linux path of the stored configuration files.

Note: Refer to section *5.2.1 Adding a Storage Volume (optional)* to add a new volume when there is no volume available to attach to the container.



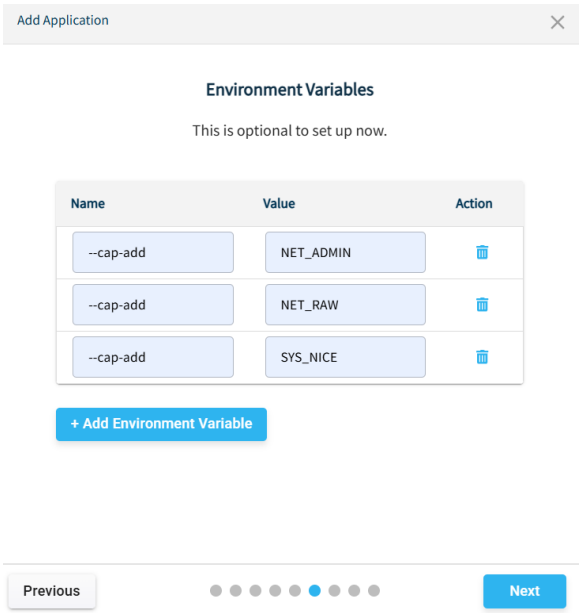
8 In the **Environment Variables** configuration, click **ADD ENVIRONMENT VARIABLE** and enter the following environment variables:

--cap-add | NET_ADMIN

--cap-add | NET_RAW

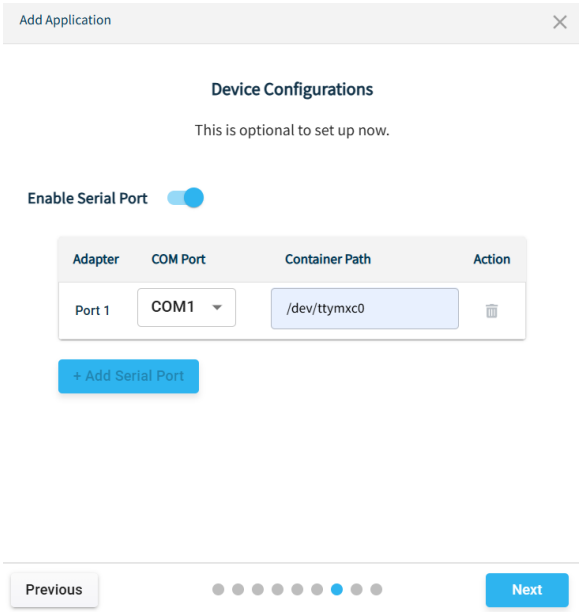
--cap-add | SYS_NICE

The ELX-EIP-MB container will not run if these variables are not configured. Add any additional environment variables of your choosing. Click **NEXT**.

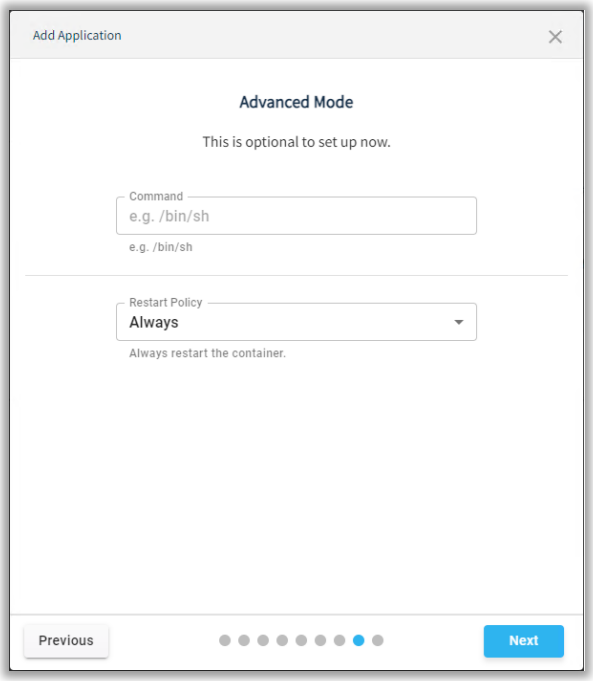


9 (Optional) In the **Device Configurations** configuration, toggle **ENABLE SERIAL PORT**, select *COM 1* from the *COM Port* dropdown, and enter `/dev/ttymxc0` in the *Container Path* text box. Click **NEXT**.

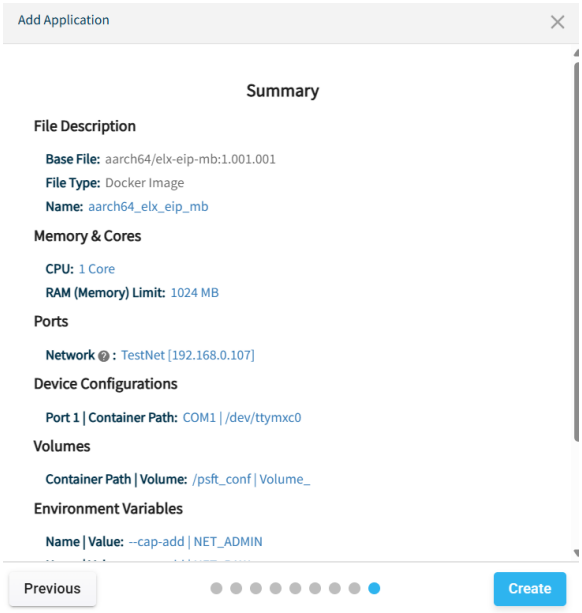
Note: If the container application uses serial communications, be sure to enable the serial port in this step. If the serial port is not controlled by the container, the CLI will be available for use. Please refer to the *ELX3 User Manual* for more information about changing serial port settings.



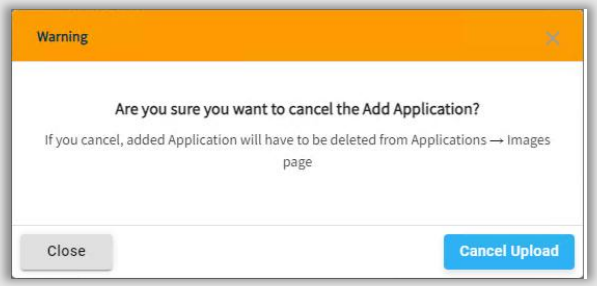
10 (Optional) In the **Advanced Mode** configuration, enter the advanced container *Command* and *Restart Policy*. Click **NEXT**.



11 In the **Summary** dialog, check all details and click **ADD APPLICATION** to create the container. If edits are needed before creating the container, click the **PREVIOUS** button.



Note: Clicking the "**X**" button on the top-right corner of the dialog at any step during container configuration will display the following:

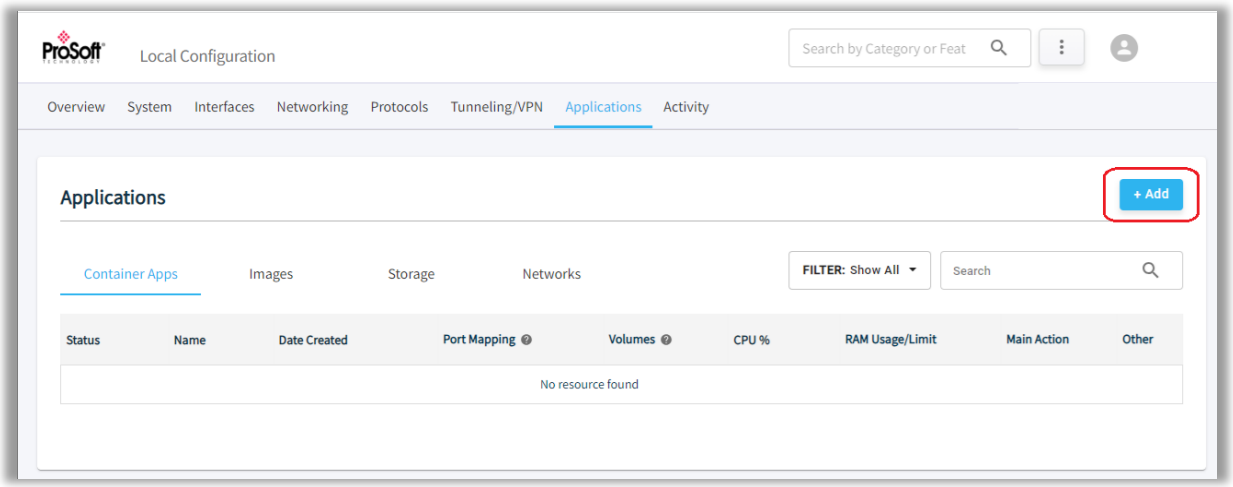


When clicking the **CANCEL UPLOAD** button, the container creation will be stopped and the image will be stored under the *Images* tab. It must be manually deleted.

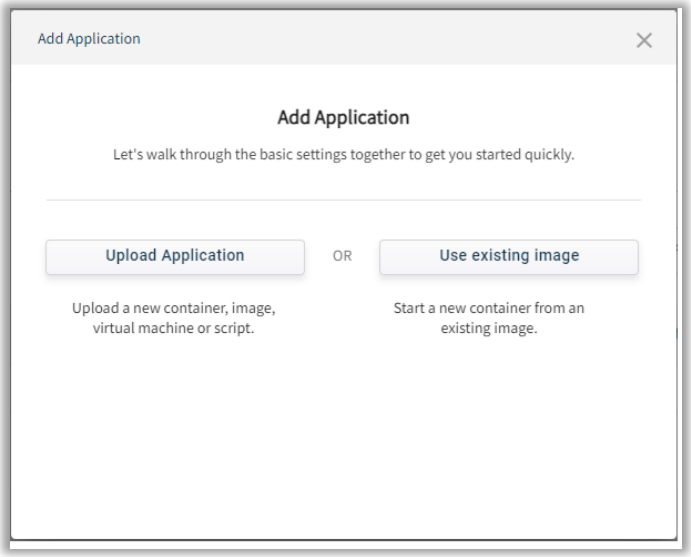
3.3.2.2 Use Existing Image

Use this option if the image file has been previously imported.

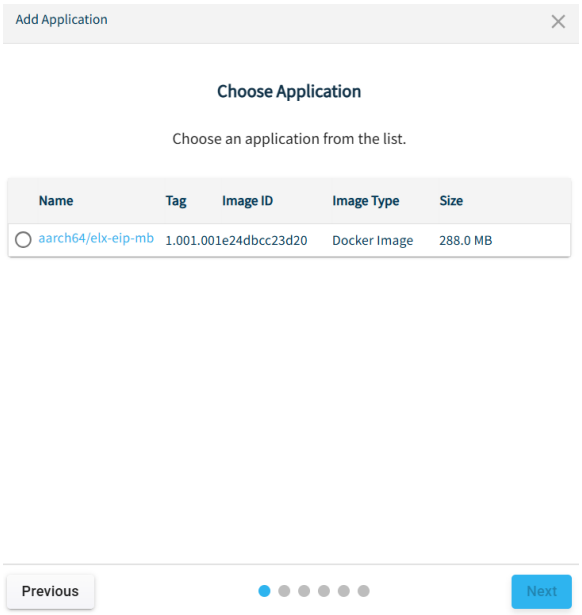
1 Navigate to the *Applications* tab > *Container Apps* tab and click the **ADD** button to open the *Add Application* wizard.



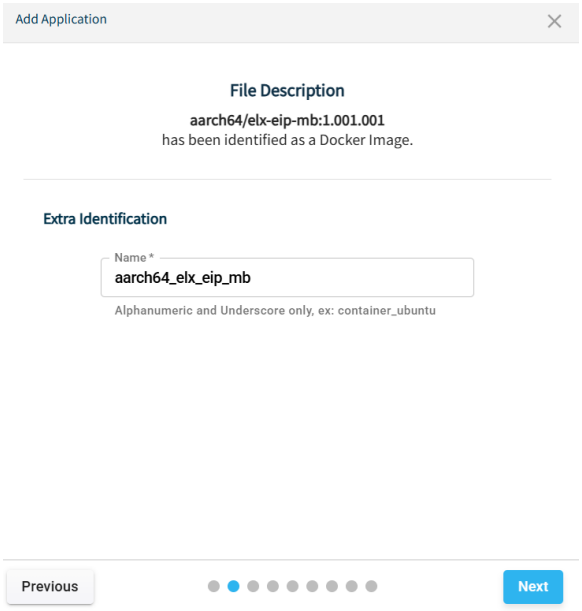
2 Select **USE EXISTING IMAGE**.



3 In the *Choose Application* configuration, select the application. Click **NEXT**.



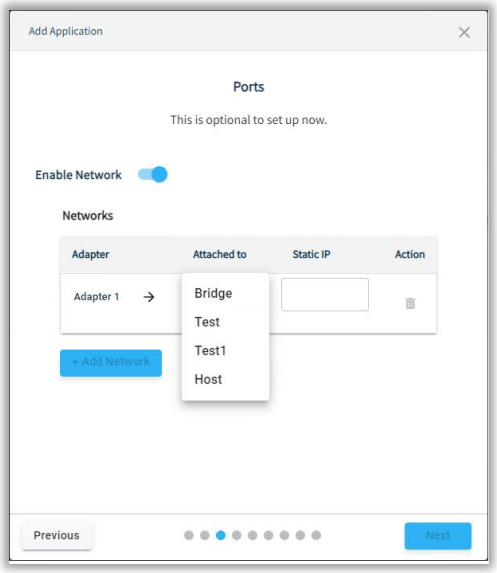
4 In the *File Description* configuration, enter a *Name* and *Labels*. Click **NEXT**.



5 The **Ports** configuration defines the network configuration. Select an option for attaching the network adapter to the container in the *Attached to* parameter and click **NEXT**.

- Bridge
- Host
- User-created custom network (MACVLAN/Bridge)

The user can also enter the (optional) Static IP corresponding to the selected network in the *Static IP* field.



Note: The user must first create the custom network to be able to create a container using that network. Detailed information regarding the creation of a custom network can be found in section *4.1.1 Adding a Network*.

Note: A maximum of four network adapters can be added.

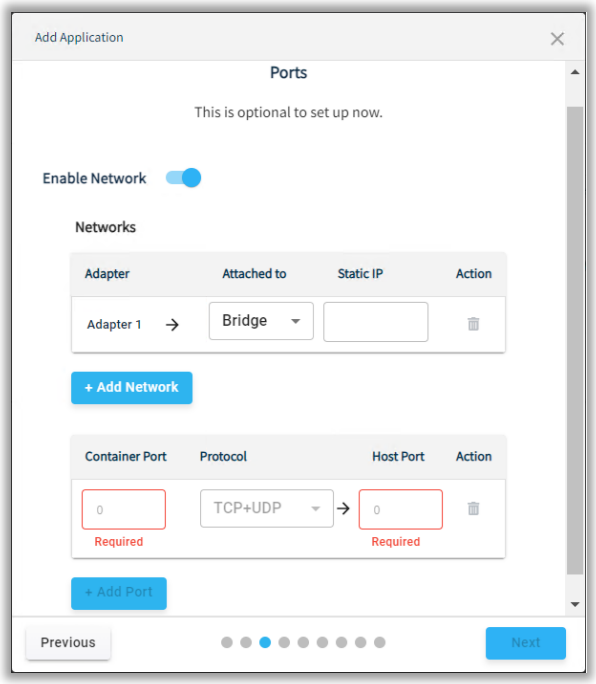
a) For *Bridge* networks, the container and host ports must be configured.

i. In the *Container Port* field, enter the container port number.

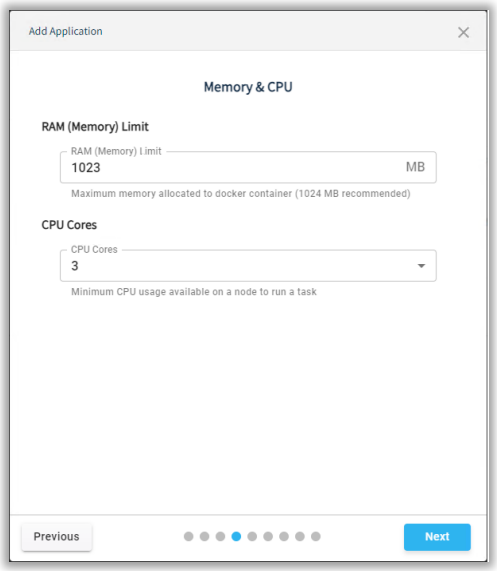
ii. In the *Host Port* field, enter the host port number.

Note: A maximum of four Container and Host ports can be added.

The user is not allowed to create a container without a Container port and Host port in **Bridge** mode. A minimum of one Docker and Host port is required to create a container with a Bridge network.



6 The **Memory & CPU** configuration defines the memory and CPU. Click **NEXT**.



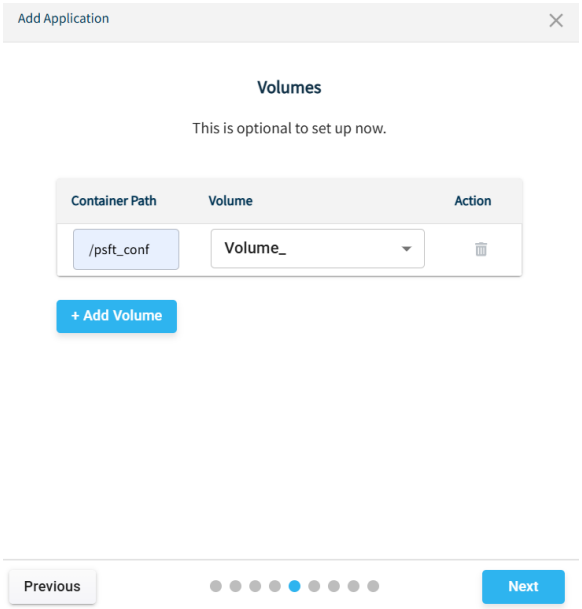
- In the *RAM (Memory) Limit* field, enter the size of memory (MB) for the container. The minimum memory value for creating containers is 4MB.
- In the *CPU Cores* field, enter the number of CPU cores to be used by the container. The number of processors is expressed in number of physical CPU cores.

Note: When configuring an ELX-EIP-MB container and 2 communication protocols are utilized in the application, then 1 CPU Core is needed. If all 3 communication protocols are utilized, then 2 CPU Cores are needed.

7 (Optional) In the **Volumes** configuration, enter `/psft_conf` in the *Container Path* and select the *Volume* from an existing list to attach to the container. Click **NEXT**.

Note: */psft_conf* is the internal Linux path of the stored configuration files.

Note: Refer to section *5.2.1 Adding a Storage Volume (optional)* to add a new volume when there is no volume available to attach to the container.



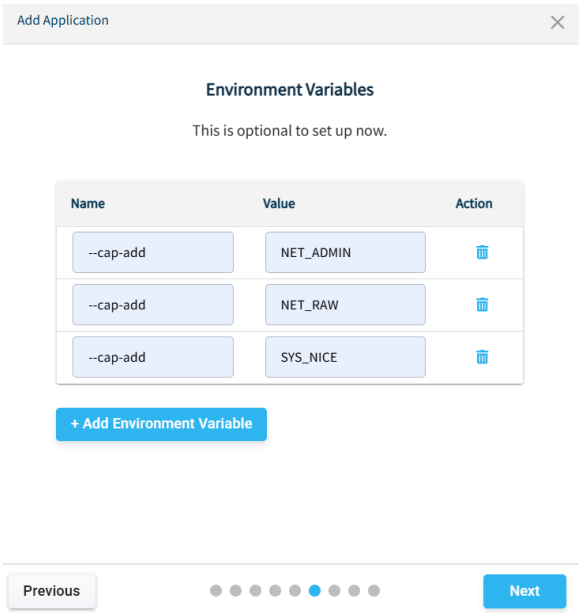
8 In the **Environment Variables** configuration, click *Add Environment Variable and* enter the following environment variables:

--cap-add | NET_ADMIN

--cap-add | NET_RAW

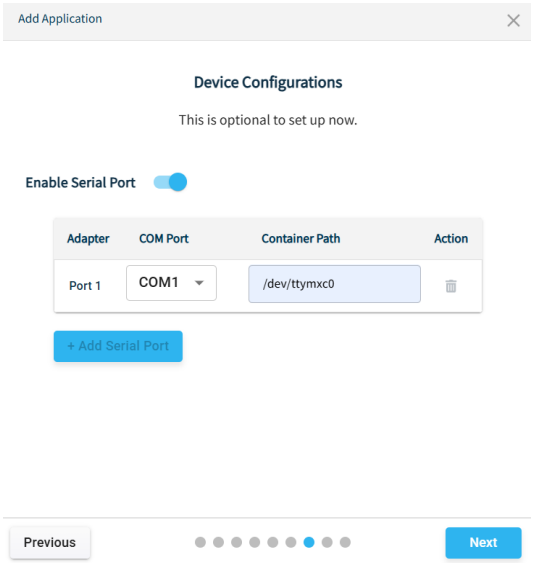
--cap-add | SYS_NICE

The ELX-EIP-MB container will not run if these variables are not configured. Add any additional environment variables of your choosing. Click **NEXT**.

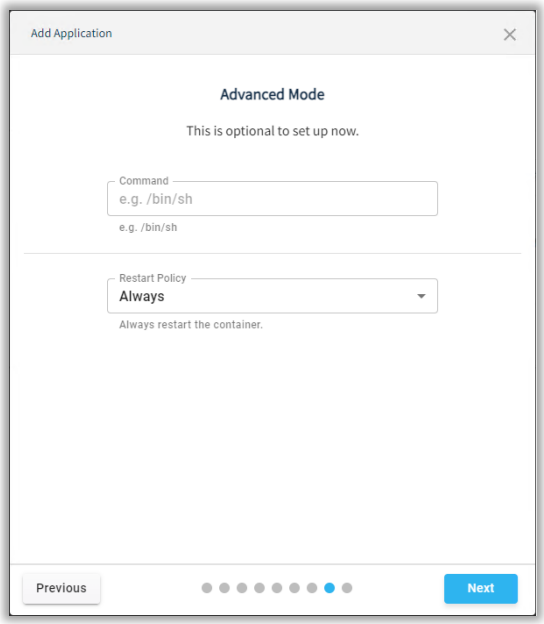


9 (Optional) In the **Device Configurations** configuration, toggle **ENABLE SERIAL PORT**, select *COM 1* from the *COM Port* dropdown, and enter `/dev/ttymxc0` in the *Container Path* text box. Click **NEXT**.

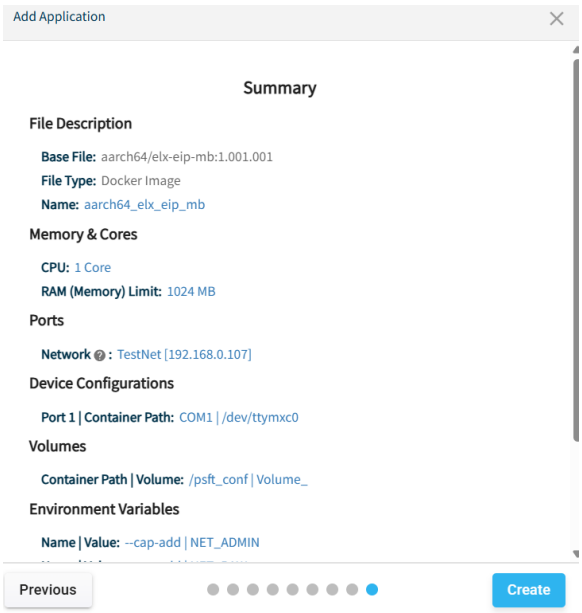
Note: If the container application uses serial communications, be sure to enable the serial port in this step. If the serial port is not controlled by the container, the CLI will be available for use. Please refer to the *ELX3 User Manual* for more information about changing serial port settings.



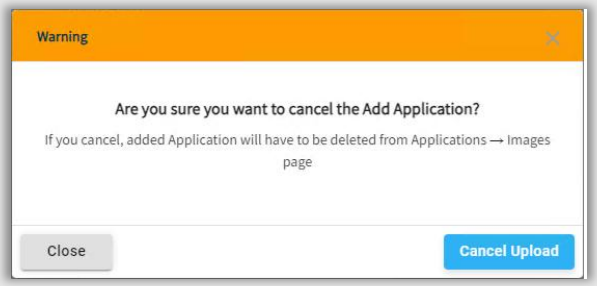
10 (Optional) In the **Advanced Mode** configuration, enter the advanced container *Command* and *Restart Policy*. Click **NEXT**.



11 In the **Summary** dialog, check all details and click **ADD APPLICATION** to create the container. If edits are needed before creating the container, click the **PREVIOUS** button.



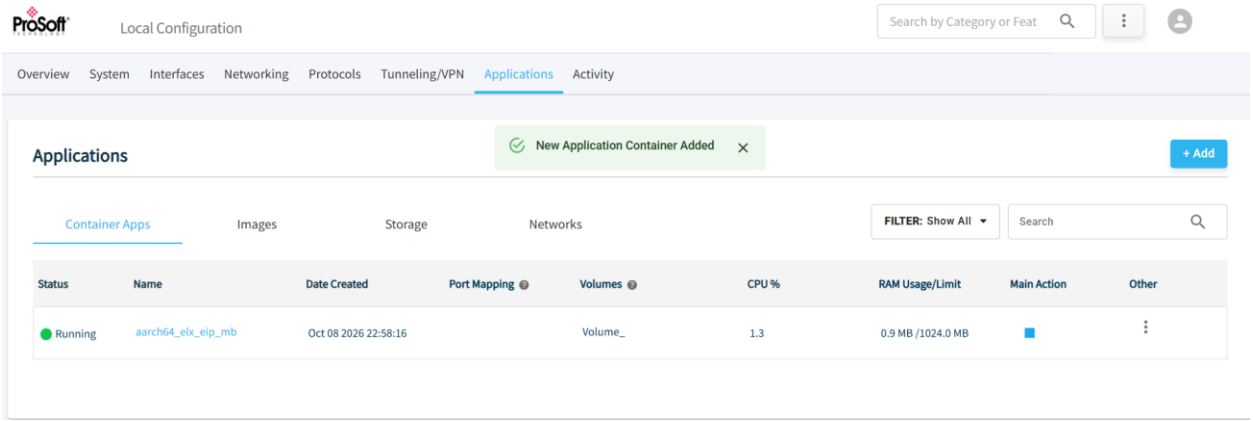
Note: Clicking the "**X**" button on the top-right corner of the dialog at any step during container configuration will display the following:



When clicking the **Cancel Upload** button, the container creation will be stopped and the image will be stored under the *Images* tab. It must be manually deleted.

3.3.3 Container Status

Upon successful creation of a container, the status information is displayed as follows:



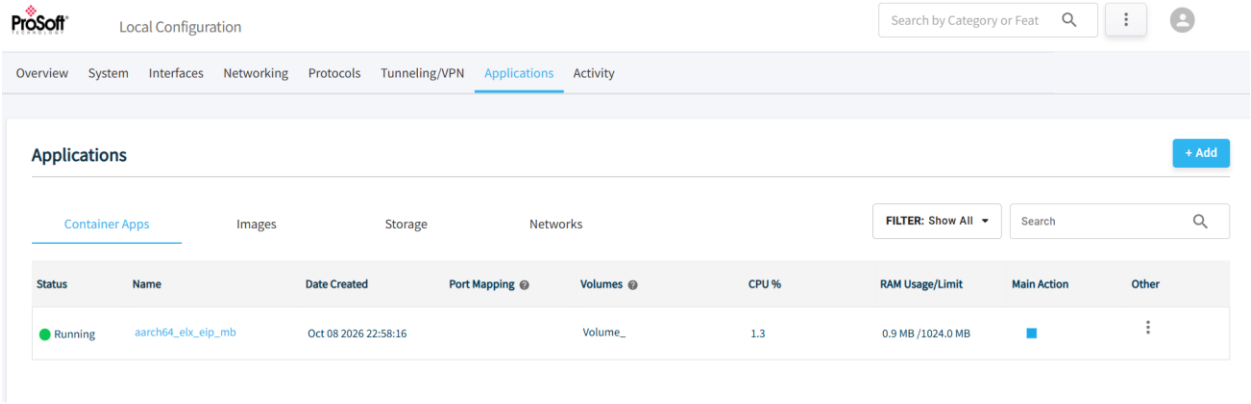
Parameter	Description
Status	The current operating status of a container: • Running • Stopped • Paused
Name	Name of the container.
Date Created	Date of container creation.
Port Mapping	This field describes the following ports: • Container Port: The Container port number. • Host Port: The Host port number.
Volumes	The container volumes that are attached to a particular container.
CPU %	The sum of work handled by a processor on the container. It is also used to estimate system performance.
RAM Usage/Limit	The memory utilization of a container and total allocated memory to a container.
Main Action	Quick action available according to the state of container.
Other	Click on the *Other* button ⋮ on a container to reveal the following:

Action Button	Description
Start	Power On the Stopped container.
Stop	Stop the container.
Pause	Pause the container.
Restart	Restart the container.
Shell	User can log into a Docker container from GUI with the help of Docker exec shell functionality.
Save	Save the container as an image.
Edit Container Details	Edit the *Name* of a container.
Delete	Delete the container.
Resume	Resume a Paused container.

Note: The *Restart*, *Pause* and *Shell* buttons are disabled when a container is in the **Stopped** or **Paused** state.

3.3.4 Verifying Successful Container Deployment

The *Status* will display as **RUNNING**.



4 Belden Horizon

This chapter covers the deployment of the ELX-EIP-MB container on the ELX3 Gateway via the Belden Horizon Console. This chapter provides the configuration steps.

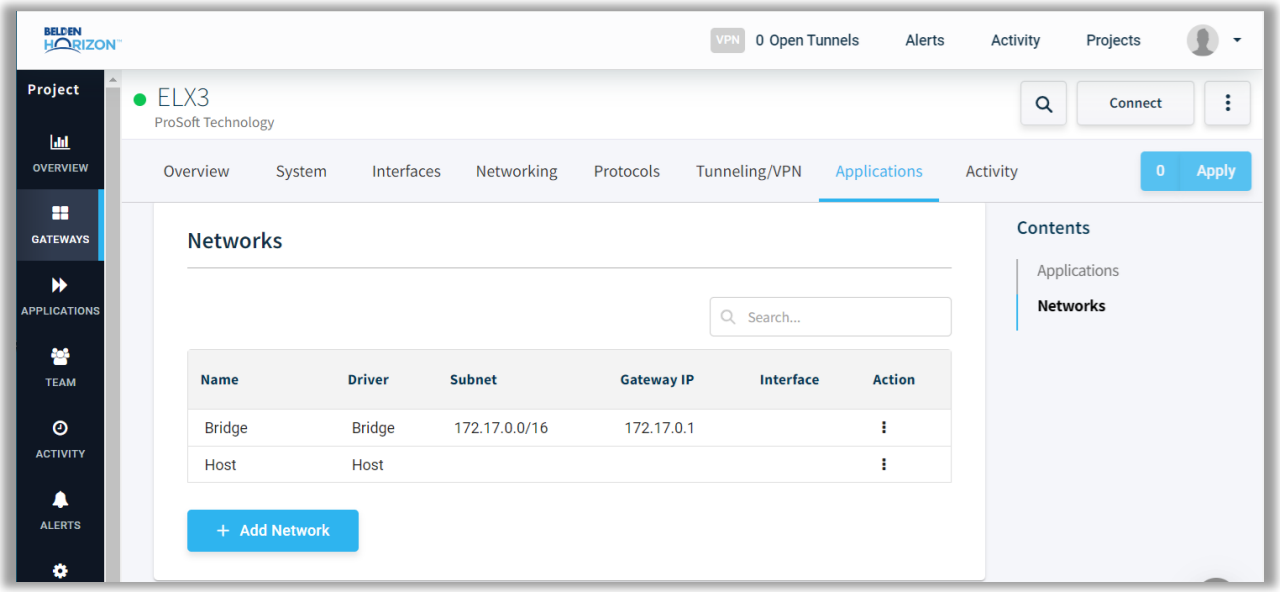
4.1 Container Network Configuration

An ELX3 gateway container network must be configured before deploying the container.

A network is the functionality that allows a container to communicate to external devices by using the host's configured LANs. This section concentrates specifically on the virtual network between containers also known as Docker networks.

A Docker network is a powerful feature that enables containers to communicate with each other and the outside world. It provides isolated and secure networking environments, allowing seamless connectivity and easy management of containerized applications.

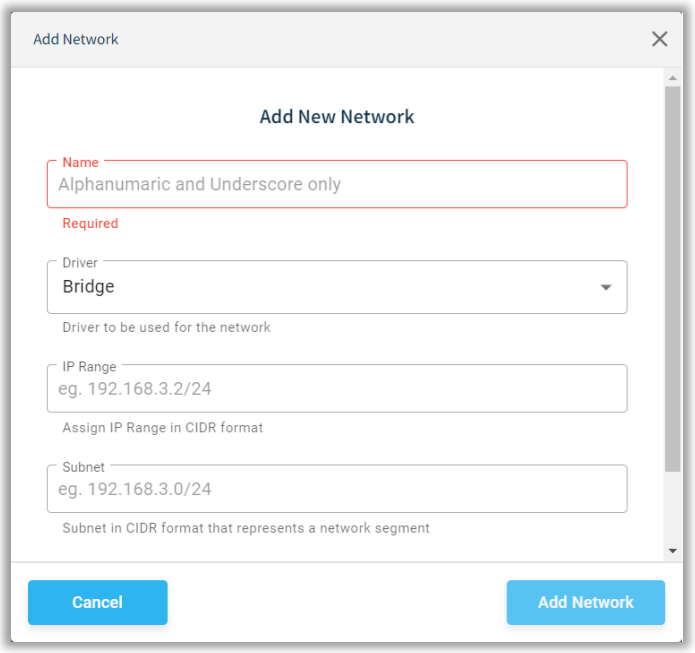
4.1.1 Adding a Network to Gateway



Note: The *Network* tab has two default networks - *Bridge* and *Host*. These two networks cannot be deleted.

To create a network:

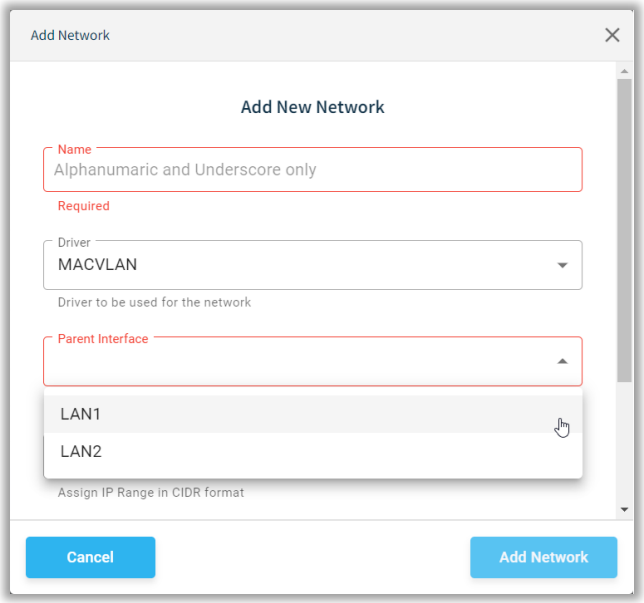
1. Ensure the ELX3 has been registered and activated in Belden Horizon. For more information, please see the *ELX3 User Manual*.
2. In the Belden Horizon Console, navigate to the *Gateway* > *Applications* > *Networks* section.
3. Click **ADD NETWORK** to open the *Add Network* wizard.
4. Enter the name of the network to be created.



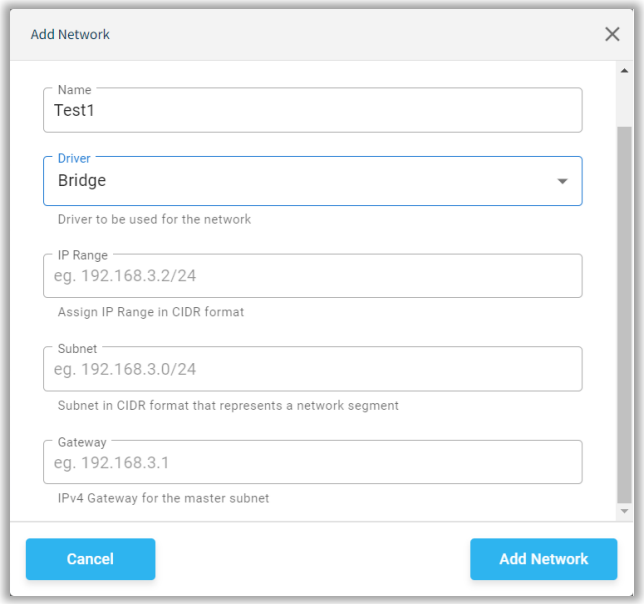
Note: The user can create a network name with an alphanumeric character with a minimum length of 2 and maximum length of 49.

The following characters are allowed:
a to z
A to Z
0 to 9
Only Special character "_" is allowed in network name creation.

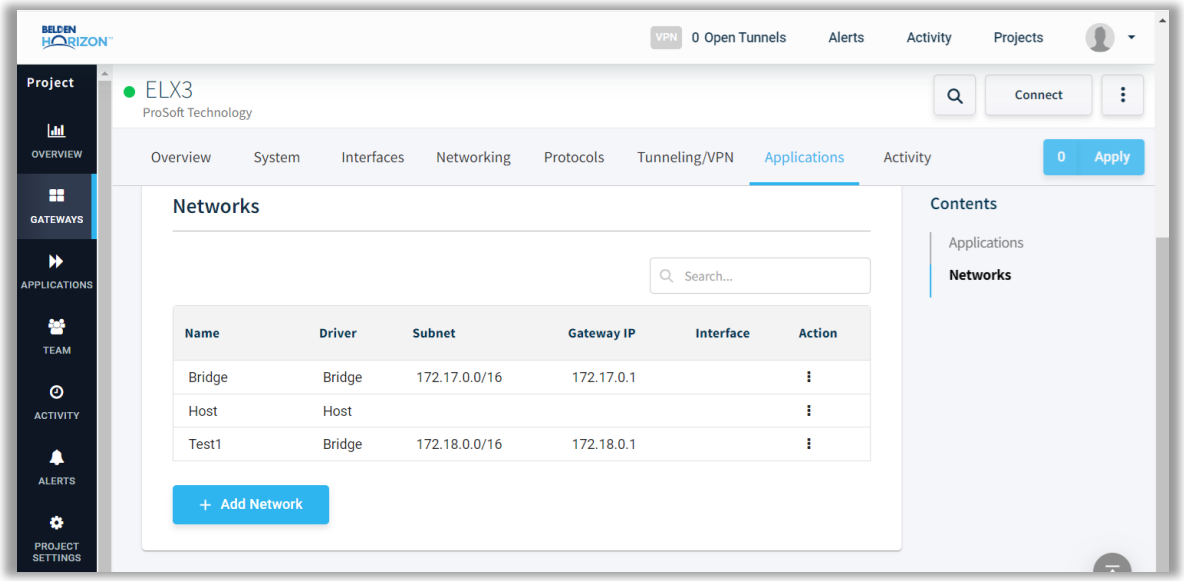
5 Select a *Driver* from the dropdown menu. **Bridge** or **MACVLAN**
If **MACVLAN** is selected, the *Parent Interface* field is required. Select the **LANx** interface from the dropdown.



6 Assign the *IP Range* to the network in CIDR (Classless Inter Domain Routing) format.
7 Assign the IP range to the *Subnet* in CIDR format.
8 Assign the IP address for the *Gateway* to master subnet in IPv4 format.
9 Click the **ADD NETWORK** button.



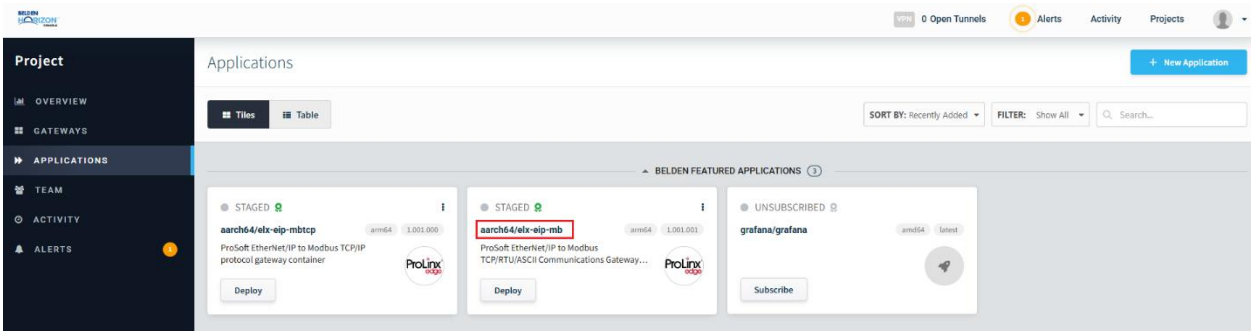
10 Upon successful network creation, the network will display in the *Gateway* > *Applications* > *Networks* section.

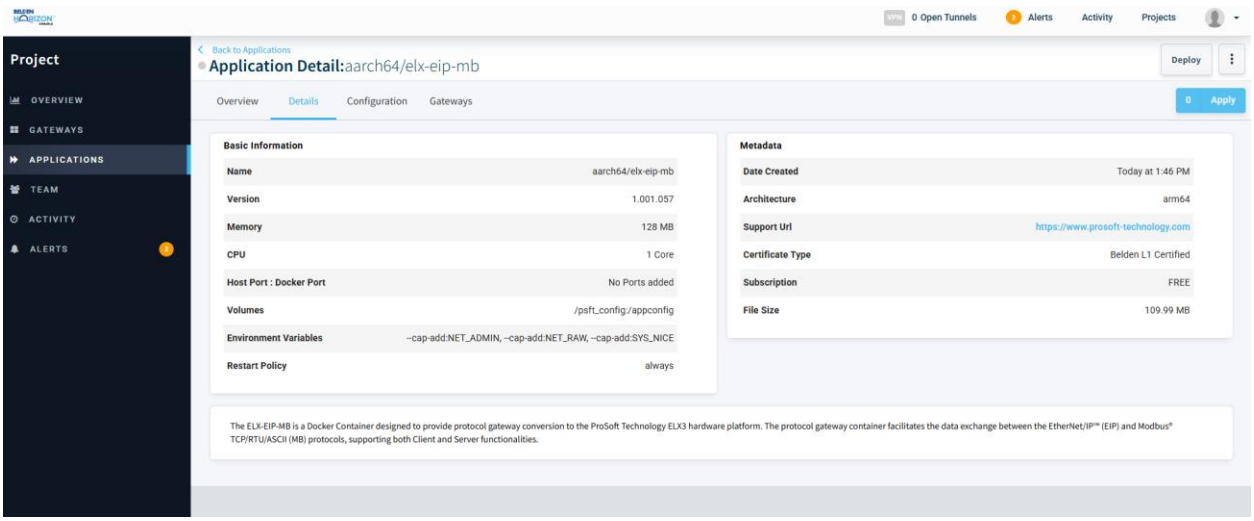


Parameter	Description
Name	Name of the network.
Driver	The Driver selected from MACVLAN and bridge during network creation.
Subnet	IP range for the master subnet.
Gateway IP	IP address of the gateway associated with master subnet.
Interface	Interface (on host) to be used for MACVLAN network.
Action	Delete the network using this parameter.

4.2 Accessing Application Details

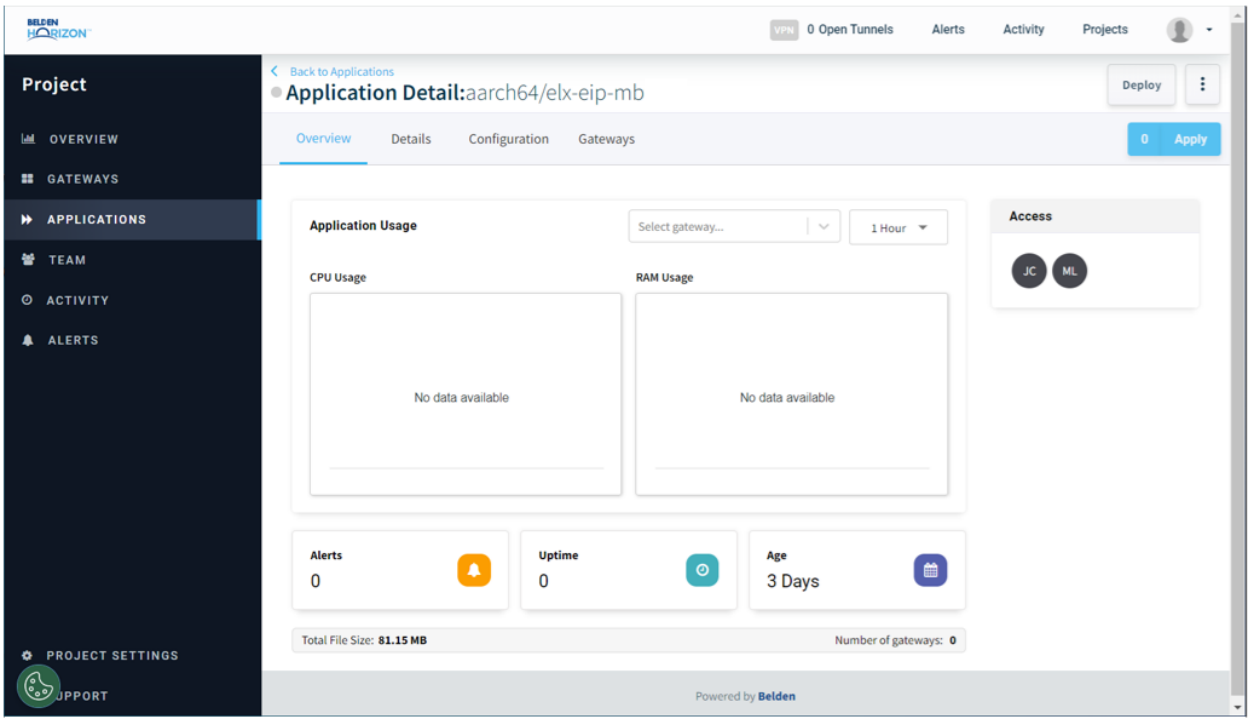
The configuration of an application can be edited by clicking on the Application name in the tile.





4.2.1 Overview

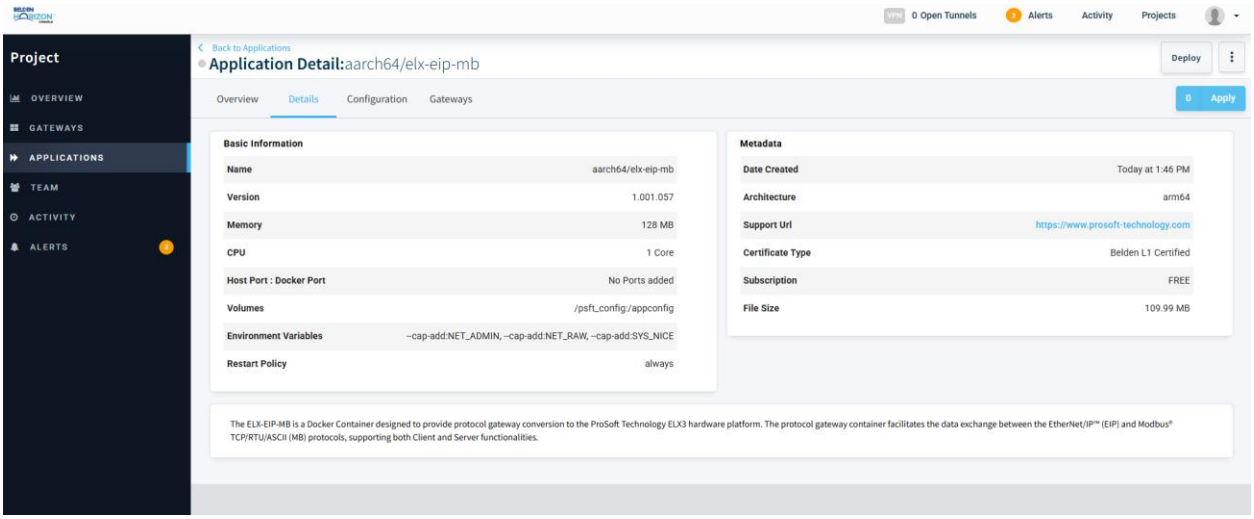
The *Overview* tab contains the *Application Usage* information of the container.



Parameter	Description
CPU Usage	The sum of work that is handled by a processor on the application. It is also used to estimate system performance.
RAM Usage	The memory utilization of the application.
Alerts	Alert notifications
Uptime	Application uptime
Age	Application age in days

4.2.2 Details

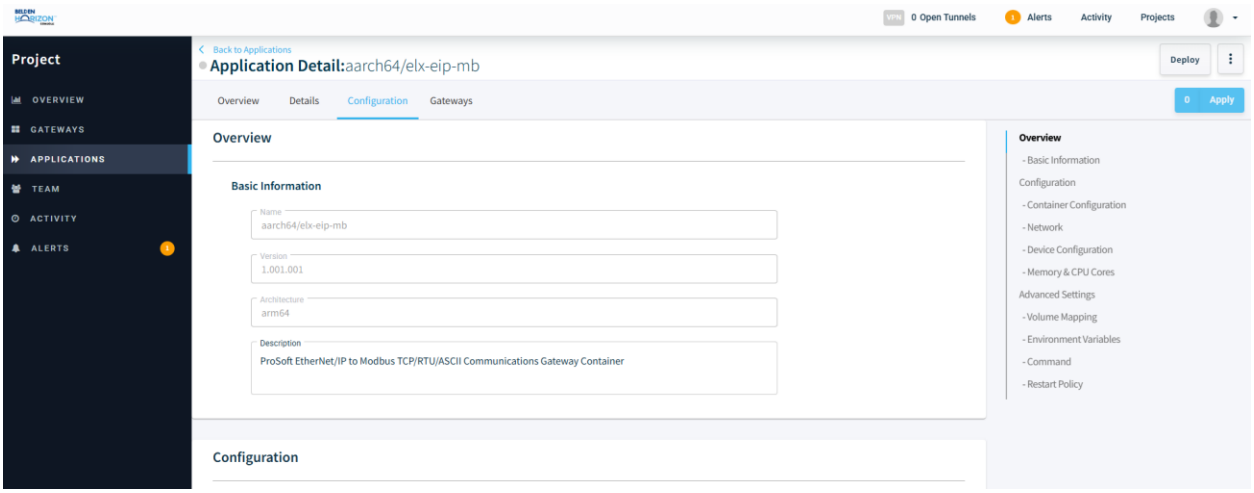
The *Details* tab contains the *Basic* and *Metadata* information of the container.



Parameter	Description
Basic Information	
Name	Name of container
Version	Version of container
Memory	The size of memory (MB) for the container.
CPU	The number of CPU cores used by the container. The number of processors is expressed in number of physical CPU cores.
Network Adapter	Name of network adapter used by the container.
Host Port : Docker Port	• Host Port: The Host port number. • Docker Port: The Container port number.
Volumes	Includes the container path and volume name.
Environment Variables	The Name and Value of the environment variables.
Restart Policy	Behavior of restarting the container. **Always**: Always restarts the container. **No**: Do not restart the container. **Unless-Stopped**: Always restarts unless the container is stopped. **On-Failure**: Restart if the container exits due to an error.
Metadata	
Date Created	Date of container creation
Architecture	Container architecture
Support URL	www.prosoft-technology.com
Certificate Type	Belden L1 Certified
Subscription	FREE
File Size	Size of container file in MB.

4.2.3 Configuration

The *Configuration* tab contains the *Overview*, *Configuration*, and *Advanced Settings* information of the container.

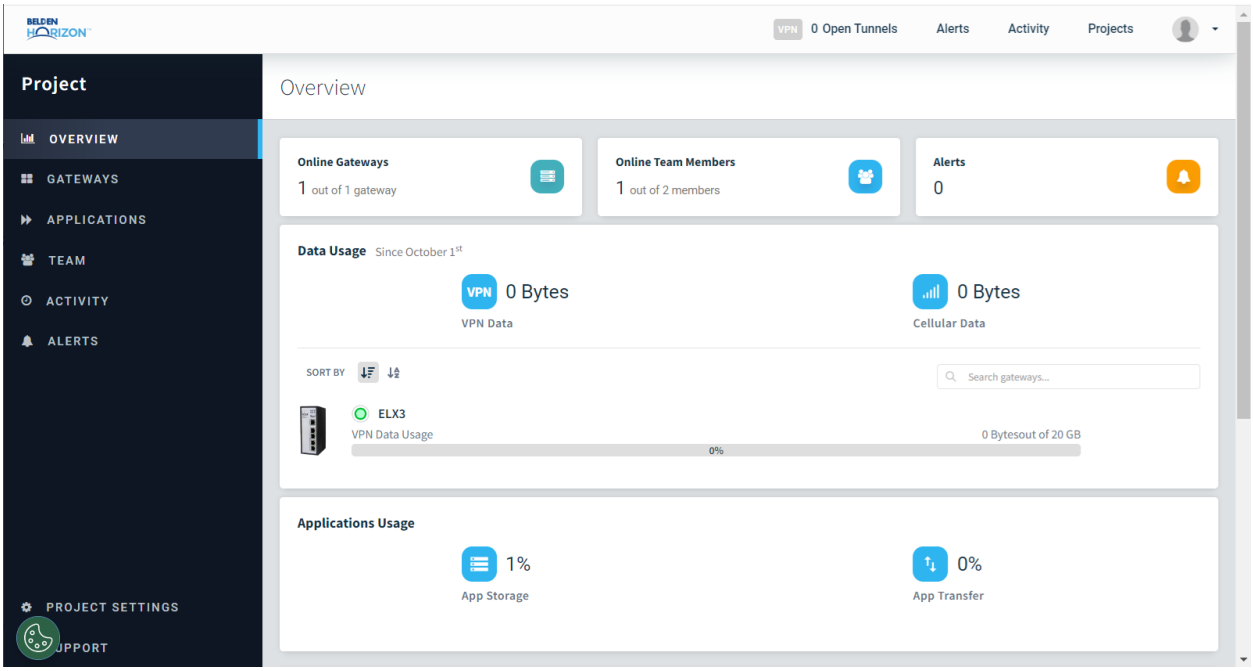


Parameter		Description
Overview		
	Basic Information	
	Name	Name of the Container Application
	Version	Version of the Container Application
	Architecture	Container architecture
	Description	Brief description of the Container Application
Configuration		
	Container Configuration	
	Container Name	Name of the Container
	Network	
	Enable Network Adapter	Enables the configuration of the network adapter for the container.
	Adapter	Adapter ID of network adapter used by the container.
	Attached to	Name of network adapter used by the container.
	Static IP	IP address of network adapter.
	Device Configuration	
	Enable Serial Port	Allows access of an onboard serial port to the container.
	Adapter	Entry name of the serial port adapter.
	COM Port	Onboard Serial Port name
	Container Path	Typically, **/dev/ttyUSB0** (for Linux applications) **/dev/ttymxc0** (for the ELX-EIP-MB container)
	Memory & CPU Cores	
	RAM (Memory) Limit	The size of memory (MB) for the container.
	CPU cores	The number of CPU cores used by the container. The number of processors is expressed in number of physical CPU cores.
Advanced Settings		
	Volume Mapping	
	Container Path	Path of directory to mount to a persistent data volume.
	Volume	Host storage volume created to provide persistent data storage to a container.
	Environment Variables	
	Name	Name of the environment variable.
	Value	Value of the environment variable.

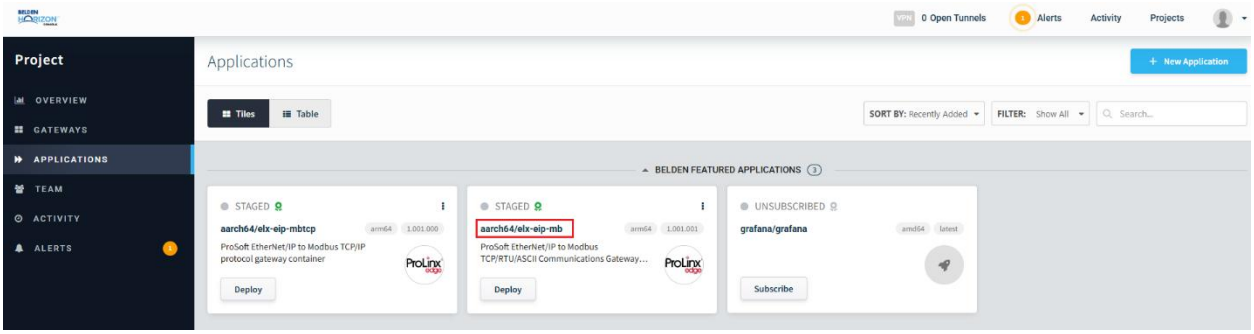
Command	
Command	Advanced Docker commands that are supported by the specific Docker image
Restart Policy	
Restart Policy	Behavior of restarting the container. **Always**: Always restarts the container. **No**: Do not restart the container. **Unless-Stopped**: Always restarts unless the container is stopped. **On-Failure**: Restart if the container exits due to an error.

4.3 Deploying the Container

The ELX3 gateway must be activated within its Belden Horizon Console project. An activated device is indicated by a green dot next to the device graphic, as shown below.



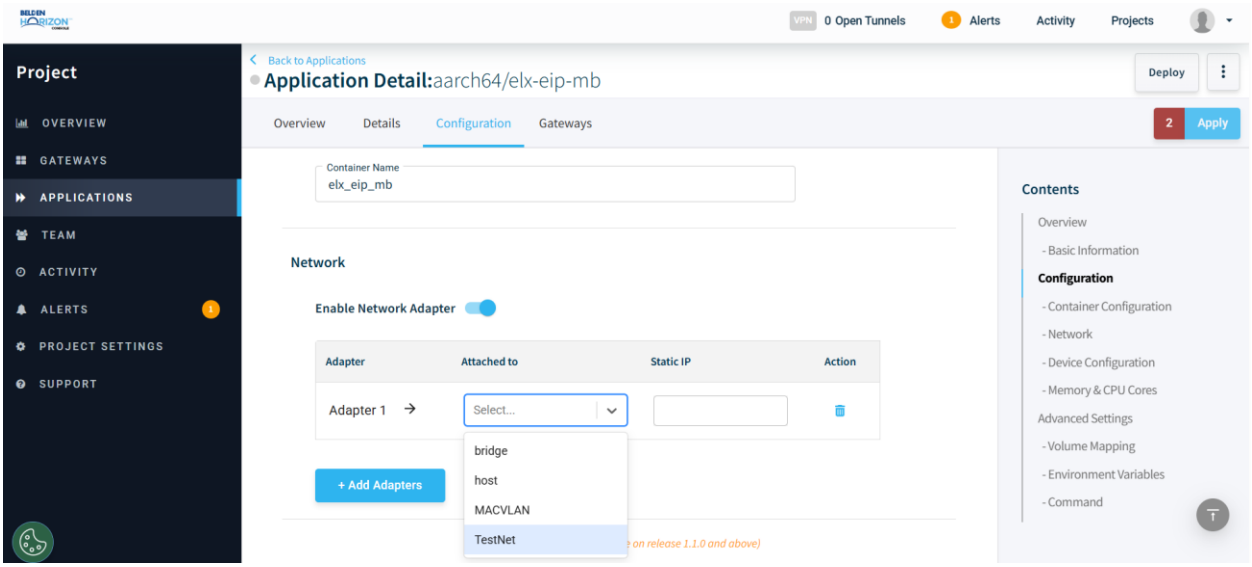
1 Click on the *Applications* tab, then navigate to the Container Application.



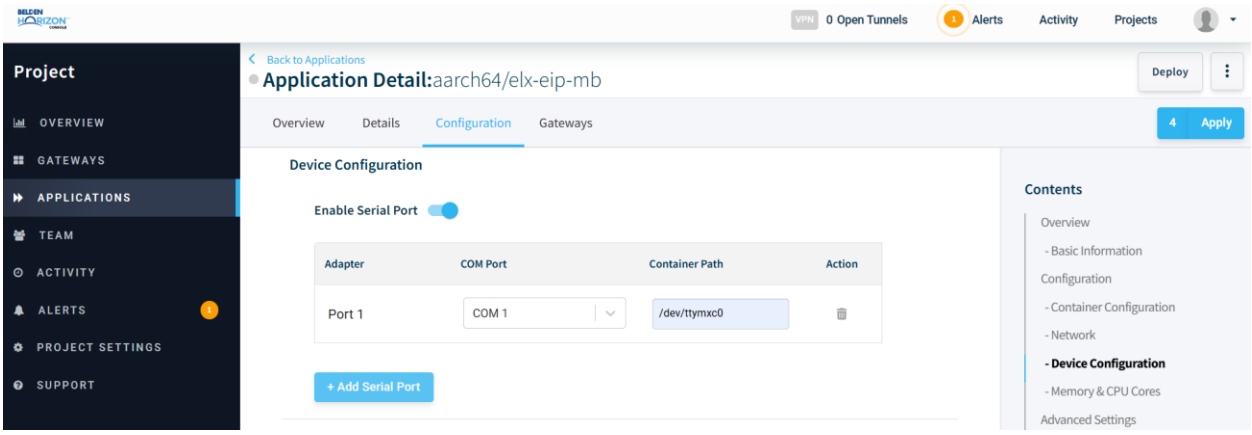
2 In the Container Application tile, click the **SUBSCRIBE** button and accept the *End User License Agreement*.

3 After subscribing to the app, click on the name of the app and navigate to the *Configuration* tile.

4 Under *Network*, click on the **ENABLE NETWORK ADAPTER** button and select the dropdown in the *Attach to* column to choose the network driver.



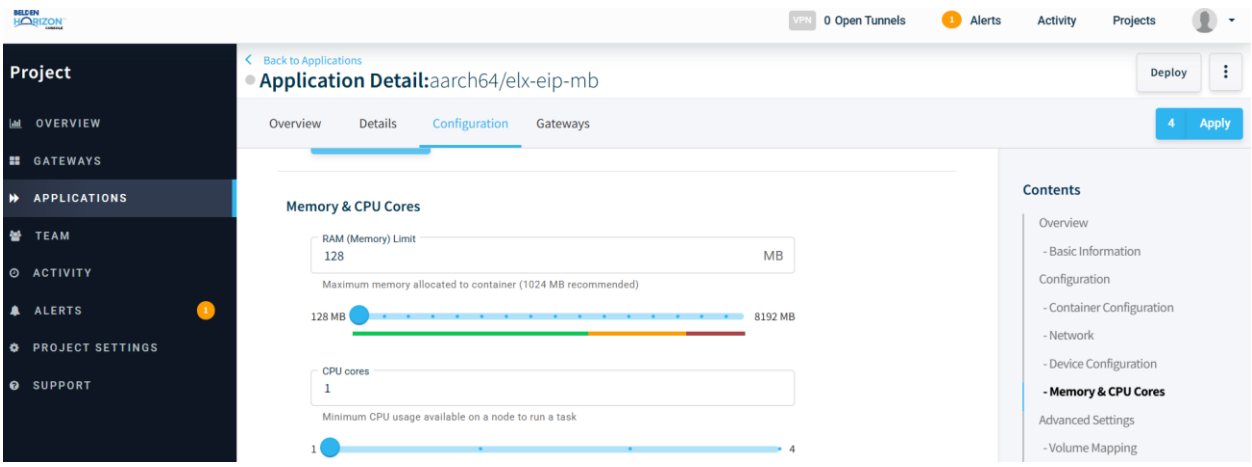
5 If your application needs access to the serial port, click on the **ENABLE SERIAL PORT** button, select the dropdown in the *COM Port* column to choose COM 1, and enter `/dev/ttymxc0` in the *Container Path* text box.



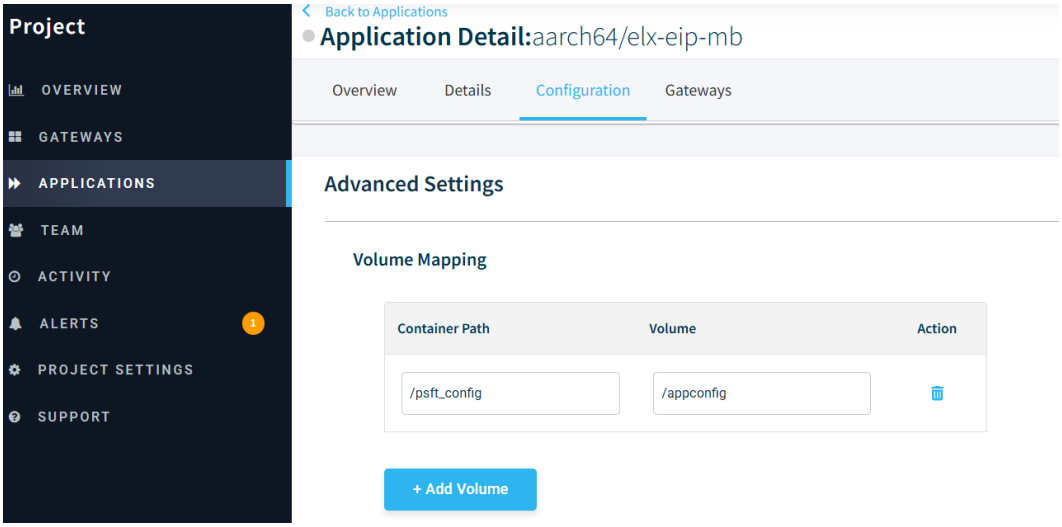
6 Under *Memory & CPU Cores,* in the *RAM (Memory) Limit* field, enter the size of memory (MB) for the container. The minimum memory value for creating containers is 4MB.

7 In the *CPU Cores* field, enter the number of CPU cores to be used by the container. The number of processors is expressed in number of physical CPU cores.

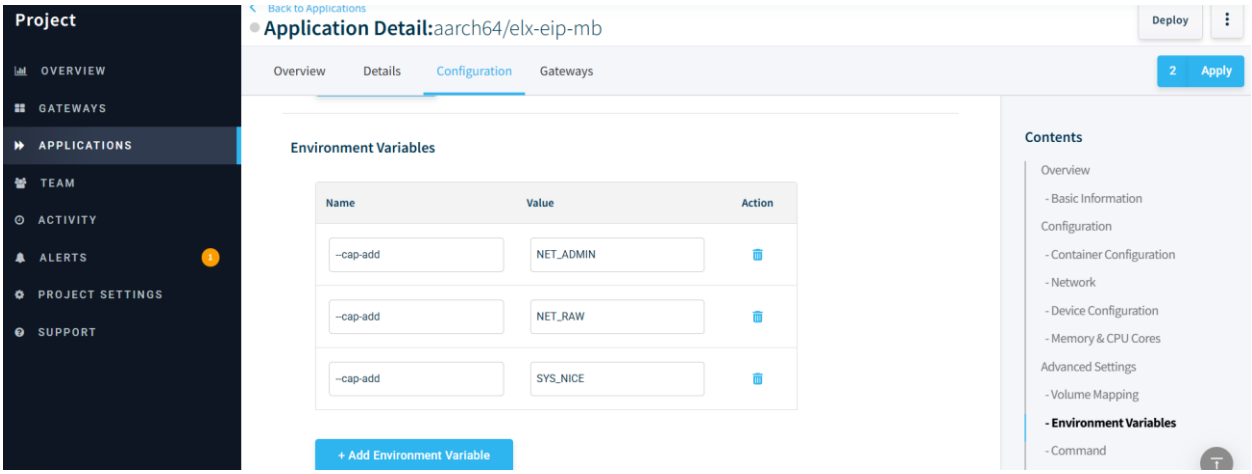
Note: When configuring an ELX-EIP-MB container and 2 communication protocols are utilized in the application, then 1 CPU Core is needed. If all 3 communication protocols are utilized, then 2 CPU Cores are needed.



8 (Optional) Under the *Advanced Settings* section, in the *Volume Mapping* configuration, enter the *Container Path* and enter the *Volume* name to attach to the container. Click the **ADD VOLUME** button to add additional volumes to the container.

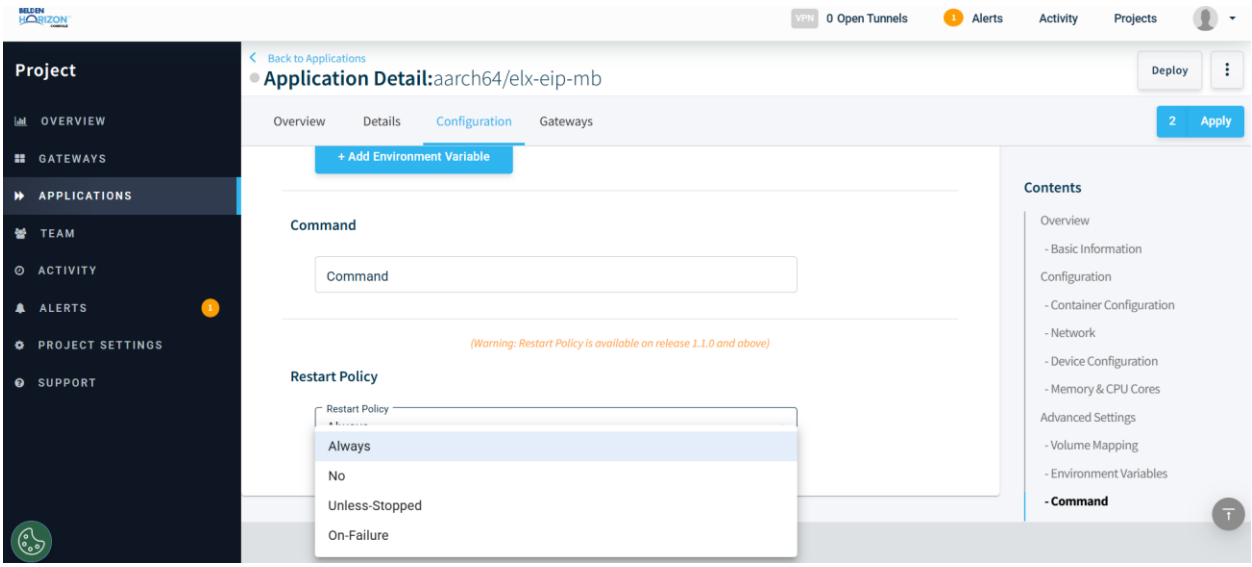


9 (Optional) Under the *Advanced Settings* section, in the *Environment Variables* configuration, select the **ADD ENVIRONMENT VARIABLE** button and enter the *Name* and *Value* of the environment variable. Do not delete the existing environment variables, as they are needed for the ELX-EIP-MB container application.



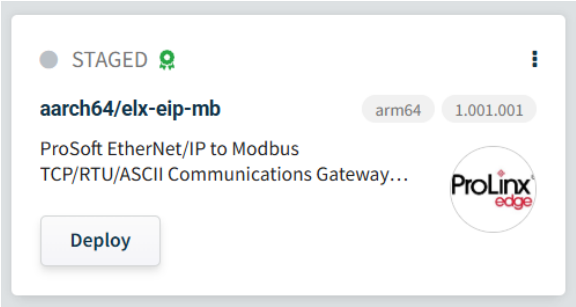
10 (Optional) Under the *Advanced Settings* section, in the *Command* text box, enter the advanced Docker commands that will run at the startup of the container.

11 (Optional) Under the *Advanced Settings* section, in the *Restart Policy* configuration, select the **Restart Policy** of the container.

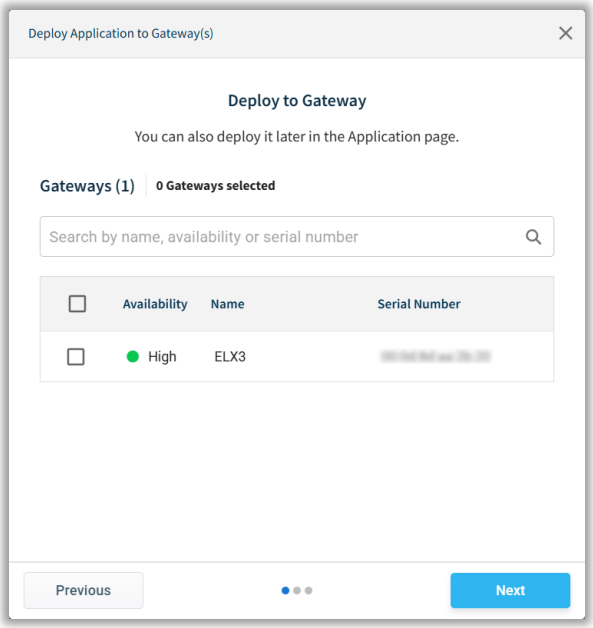


12 Click **APPLY** when finished configuring the ELX-EIP-MB container application to save your changes.

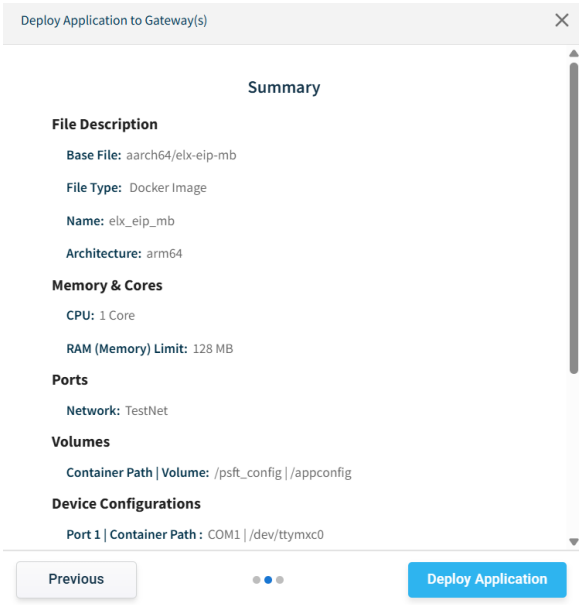
13 Go back to the *Applications* tab and click **DEPLOY** for the ELX-EIP-MB application.



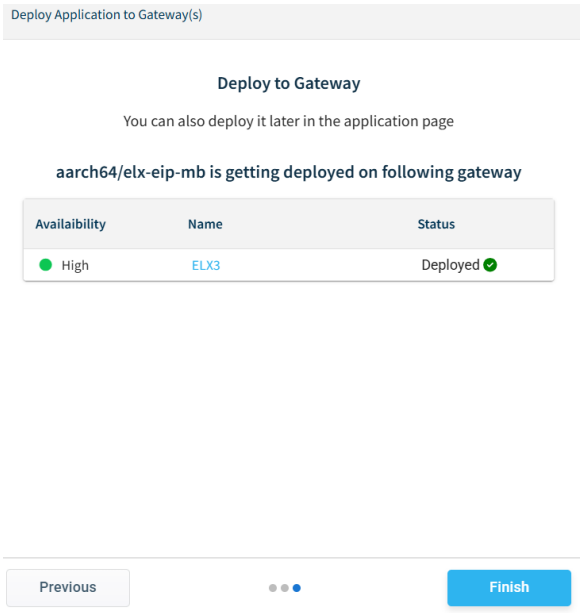
14 In the *Deploy Application to Gateway(s)* dialog, select the ELX3 gateway and click **NEXT**.



15 Review the configuration information as needed in the **Summary** dialog. Click **DEPLOY APPLICATION**.

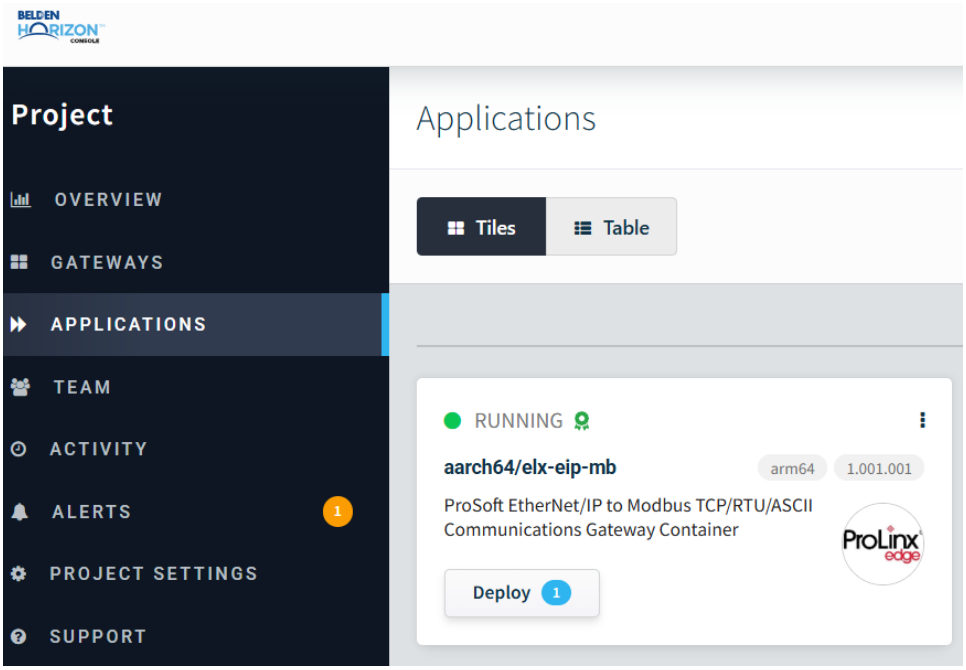


16 The *In Progress* status is displayed and the *Deployed* status completes the deployment process. Click **FINISH**.



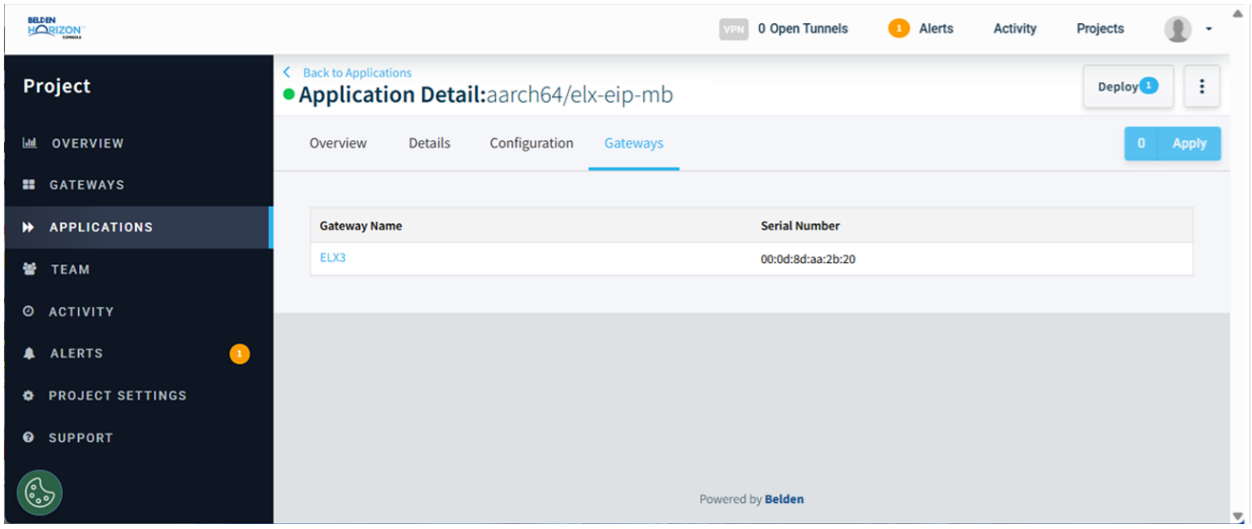
4.3.1 Verifying Successful Container Deployment

In the *Applications* tab, the application will indicate it is running.



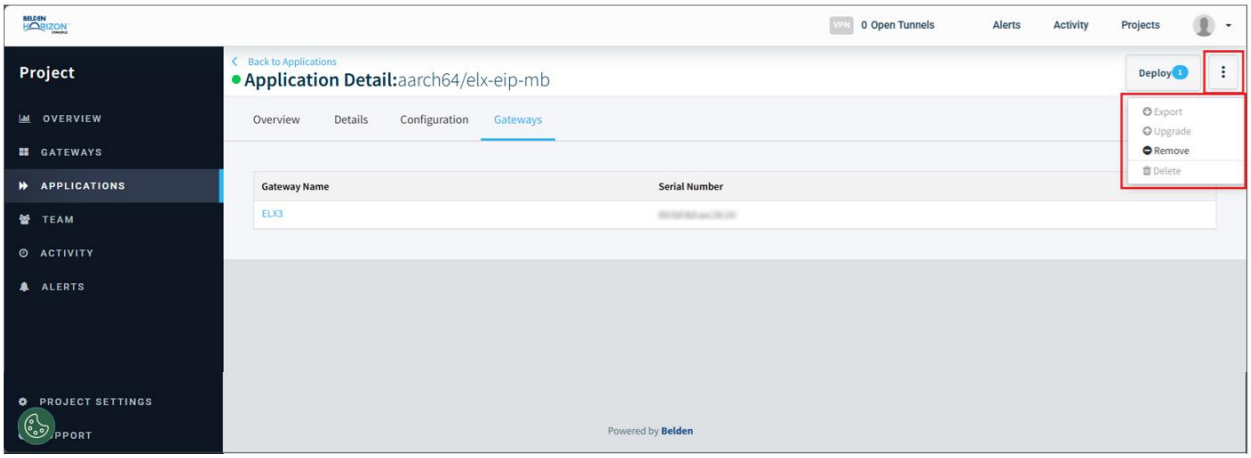
4.4 Gateways

The *Gateways* tab contains the ELX3 or OpEdge device(s) that are using the ELX-EIP-MB Container.

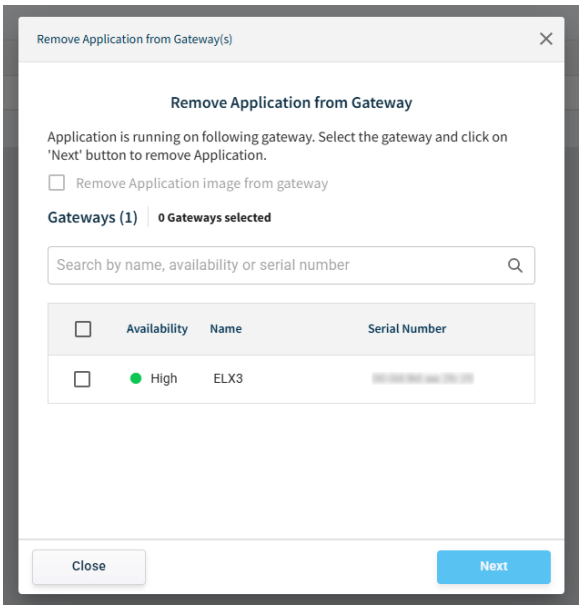


4.5 Removing the Container Application from the Gateway

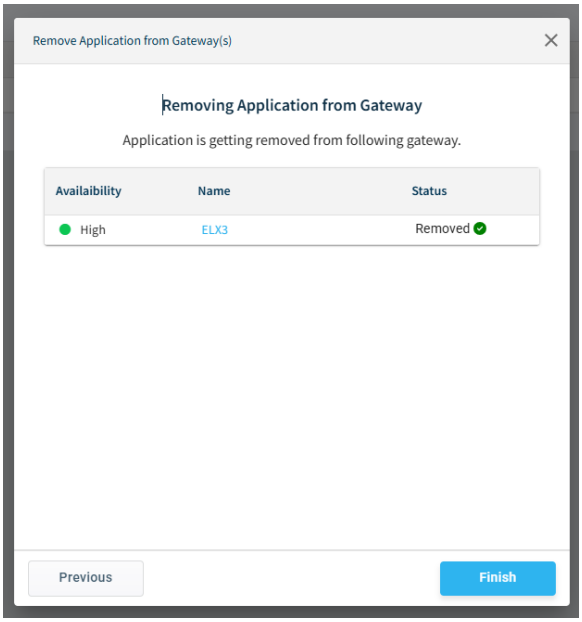
1 In the kebab menu ⋮ at the upper right part of the window, select the **REMOVE** option.



2 In the *Remove Application from Gateway(s)* dialog, select the application and click **NEXT** to initiate the removal process.



3 Click **FINISH** when complete.



5 ProSoft Configuration Builder

5.1 Overview

ProSoft Configuration Builder (PCB) software provides a quick and easy way to manage the ELX-EIP-MB container configuration files in a graphical, tree-structured layout. It includes extensive online help to define configuration parameters and database configuration.

It can be downloaded at: www.prosoft-technology.com.

5.1.1 Compatible Operating Systems

- Windows 11
- Windows 10 Pro x64
- Windows Server 2019

Note: To use PCB with older Windows operating systems, be sure to install it using the *Run as Administrator* option. To find this option, right-click the Setup.exe program icon, and then click **RUN AS ADMINISTRATOR** on the context menu. Using the *Run as Administrator* option allows the installation program to create folders and files on the PC with proper permissions and security.

If the **RUN AS ADMINISTRATOR** option is not used, the ProSoft Configuration Builder may appear to install correctly but multiple file access errors will appear when ProSoft Configuration Builder is running. If this happens, completely uninstall the ProSoft Configuration Builder, then re-install using the *Run as Administrator*.

5.1.2 Hash Signature

Before installing PCB, compare the sha256 hash on the PCB webpage with the sha256 hash calculated from the PCB *.exe* file. Use the Windows "**certutil**" utility to calculate the sha256 hash.

In the Windows Command Prompt navigate to the file location, then enter the following command:

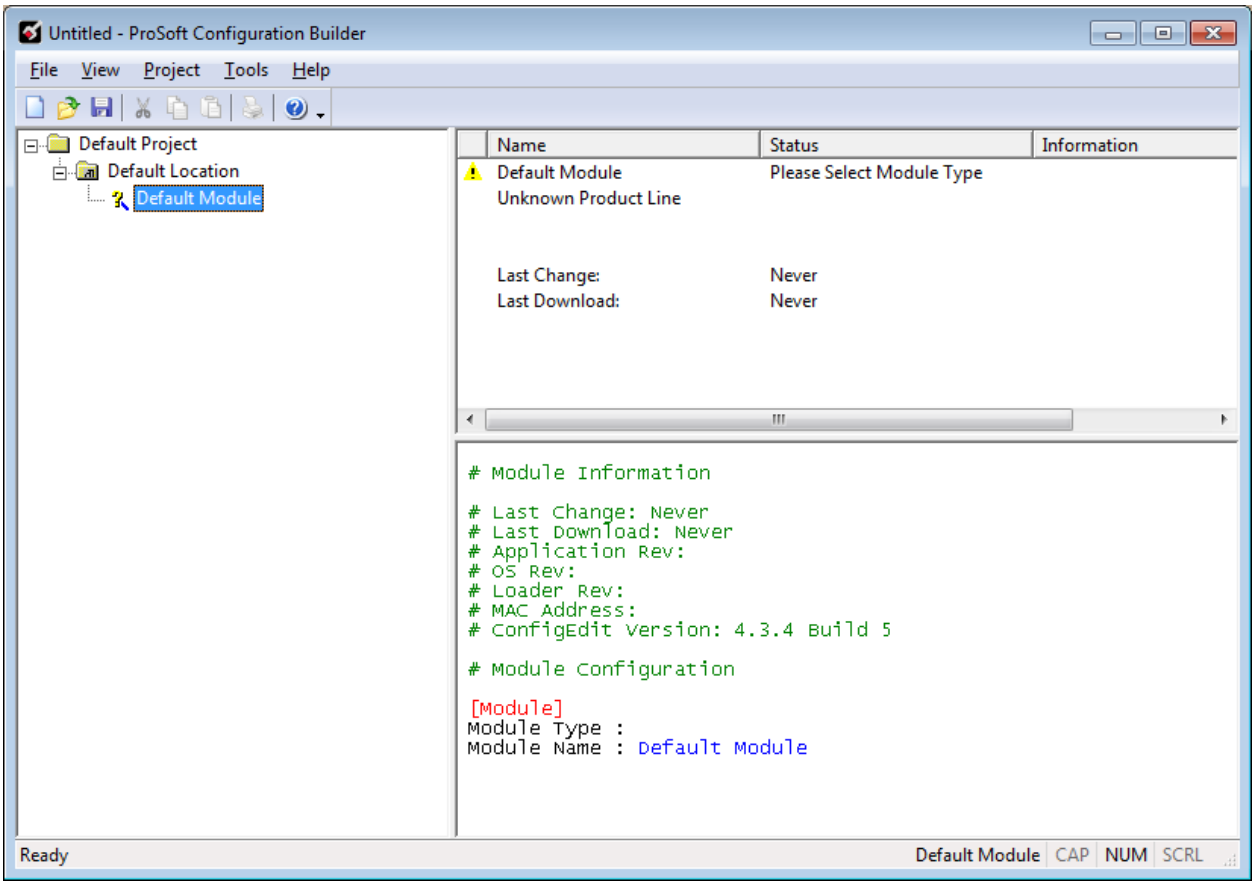
```
CertUtil -hashfile path\file_name.ext SHA256
```

Example:

```
CertUtil -hashfile C:\Downloads\PCB_4.9.0.042.exe SHA256
```

5.2 Using ProSoft Configuration Builder

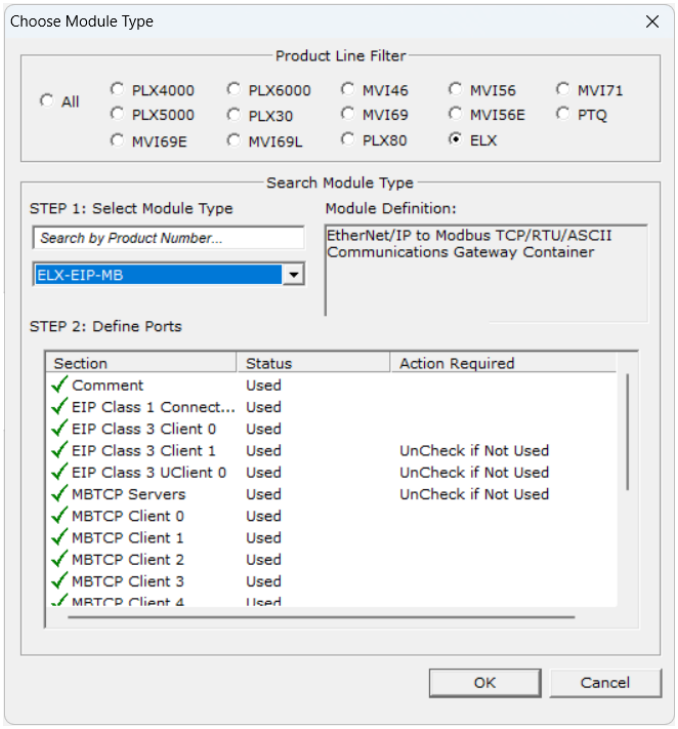
The ProSoft Configuration Builder window consists of a tree view on the left, an information pane, and a configuration pane on the right side of the window. The following illustration displays the ProSoft Configuration Builder window with a new project.



5.2.1 Selecting the ELX-EIP-MB Container

To add the ELX-EIP-MB container to the project:

1. Right-click **DEFAULT MODULE** in the tree view and then choose **CHOOSE MODULE TYPE**. This opens the *Choose Module Type* dialog.



2. In the *Product Line Filter* area of the dialog, select the **ELX** radio button.
3. Select the **ELX-EIP-MB** model.
4. Click **OK**.

5.2.2 Connecting the PC to the ELX3

Connect one end of the Ethernet cable to the ELX3 ProLinx Edge gateway and the other end to an Ethernet hub or switch accessible from the same network as the PC. Or, directly connect from the Ethernet Port on the PC to the ELX3 ProLinx Edge gateway.

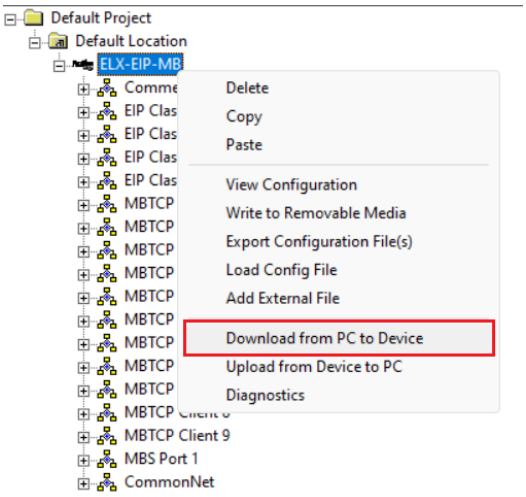
5.3 Downloading the Project to the Container

5.3.1 Downloading an ELX-EIP-MB Project

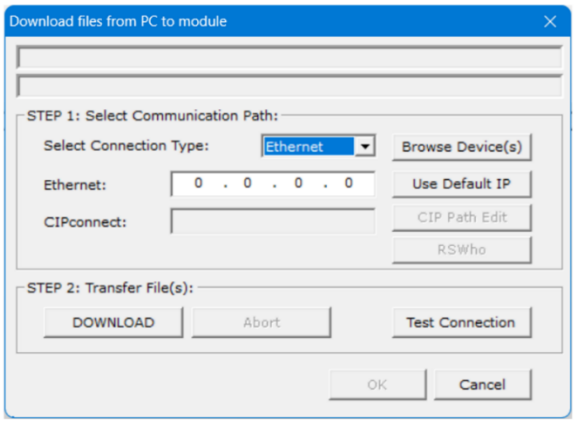
Note: When downloading an ELX-EIP-MB project to an ELX-EIP-MBTCP container all the configuration settings of the ELX-EIP-MB container will be downloaded to the ELX-EIP-MBTCP container, including the *MBS Port 1* configuration settings. However, the MBS port will not be enabled/active.

To download the project file from the PC to the container:

1 In the tree view in *ProSoft Configuration Builder*, right-click the **ELX-EIP-MB** icon and select **DOWNLOAD FROM PC TO DEVICE.**

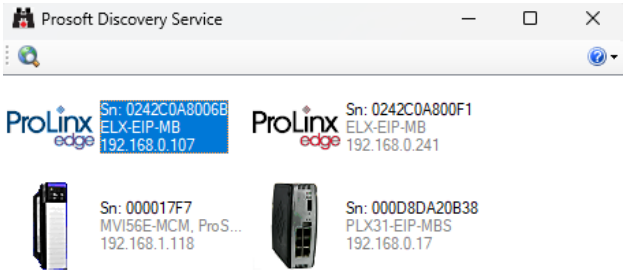


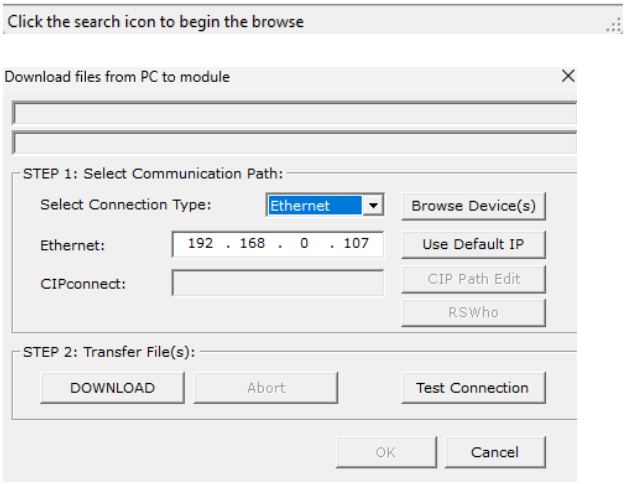
2 In the *Download files from PC to module* dialog, in the *Select Connection Type* dropdown box, select the default **ETHERNET** option.



3 Click **BROWSE DEVICE(S)** to select the container from the *ProSoft Discovery Service* dialog.

4 Double-click on the container icon. This will populate the target IP address for the download.





5 Click **TEST CONNECTION** to verify that the IP address allows access to the ELX-EIP-MBTCP container.

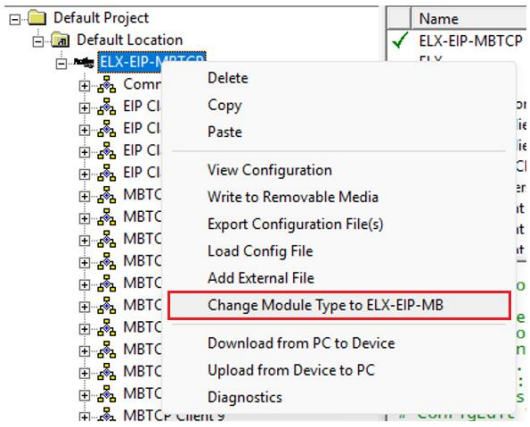
6 If the connection succeeds, click **DOWNLOAD** to transfer the project to the ELX-EIP-MBTCP container.

7 Click **OK** when complete.

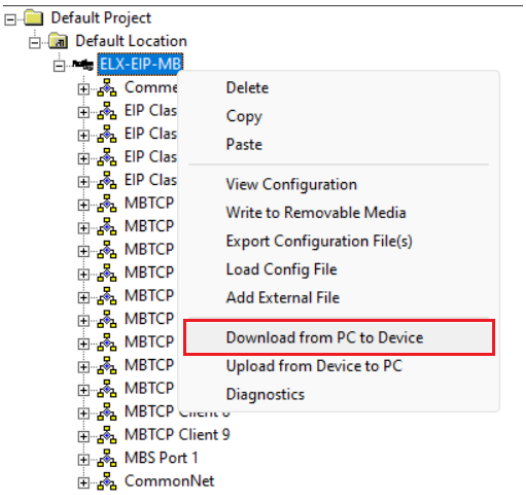
5.3.2 Downloading an ELX-EIP-MBTCP Project

To download the project file from the PC to the container:

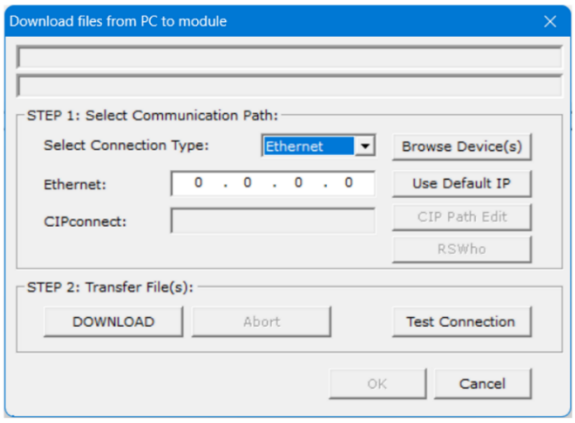
1 In the tree view in *ProSoft Configuration Builder*, right-click the **ELX-EIP-MBTCP** icon and select **CHANGE MODULE TYPE TO ELX-EIP-MB**. This will update the ELX-EIP-MBTCP to ELX-EIP-MB while maintaining the configuration settings of the original project.



2 Right-click the **ELX-EIP-MB** icon and select **DOWNLOAD FROM PC TO DEVICE.**

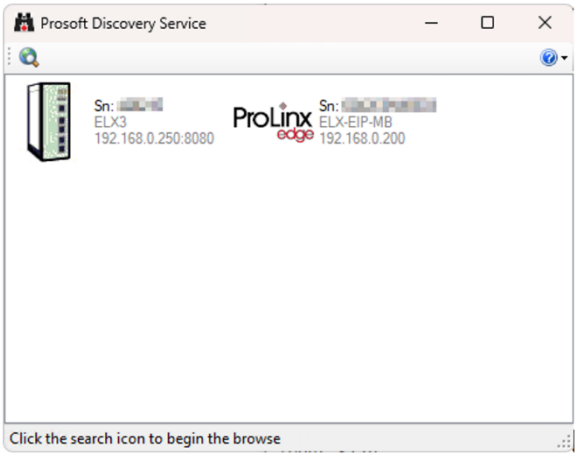


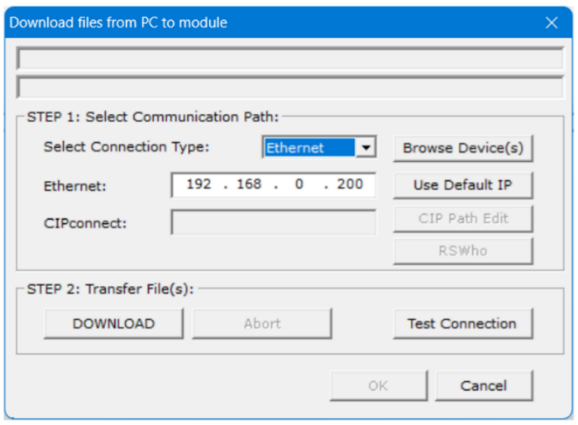
3 In the *Download files from PC to module* dialog, in the *Select Connection Type* dropdown box, select the default **ETHERNET** option.



4 Click **BROWSE DEVICE(S)** to select the container from the *ProSoft Discovery Service* dialog.

5 Double-click on the container icon. This will populate the target IP address for the download.





6 Click **TEST CONNECTION** to verify that the IP address allows access to the ELX-EIP-MB container.

7 If the connection succeeds, click **DOWNLOAD** to transfer the project to the ELX-EIP-MB container.

8 Click **OK** when complete.

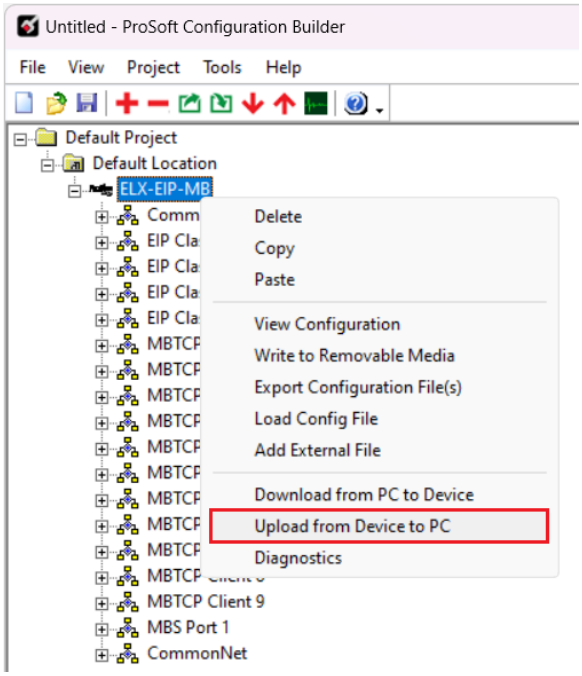
5.4 Uploading the Configuration from the Container

5.4.1 Uploading a Configuration to an ELX-EIP-MB Project

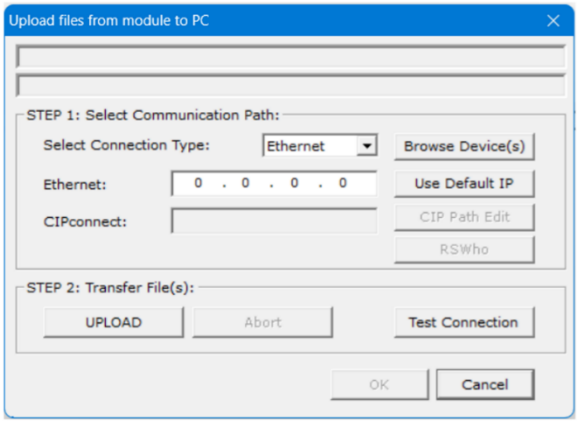
Note: If uploading a configuration from an ELX-EIP-MBTCP container, be aware that all the configuration settings of the ELX-EIP-MBTCP container (which does not include the MBS Port 1 settings) will be uploaded to the ELX-EIP-MB project. If needed, the module in the PCB project can be converted to an ELX-EIP-MB by right-clicking on the ELX-EIP-MBTCP icon and selecting the **CONVERT MODULE TYPE TO ELX-EIP-MB** option.

To upload the project file from the container to the PC:

1 In the tree view in ProSoft Configuration Builder, right-click the **ELX-EIP-MB** icon and then select **UPLOAD FROM DEVICE TO PC**. This opens the *Upload* dialog.

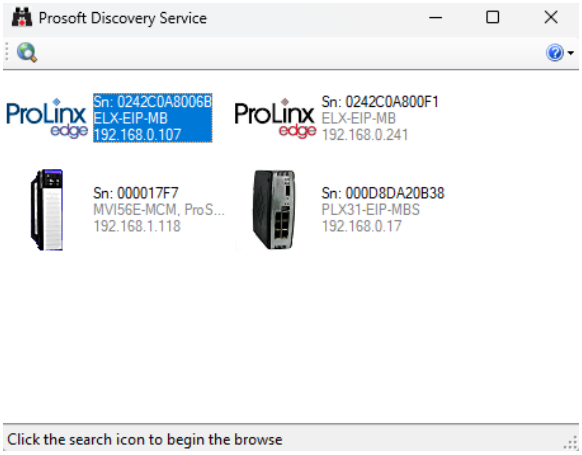


2 In the *Upload files from module to PC* dialog, in the *Select Connection Type* dropdown box, select the default **ETHERNET** option.



3 Click **BROWSE DEVICE(S)** to select the container from the ProSoft Discovery Service dialog.

4 Double-click on the container icon. This will populate the target IP address for the download.



5 Click **TEST CONNECTION** to verify that the IP address allows access to the container.

6 If the connection succeeds, click **UPLOAD** to upload the project to PCB.

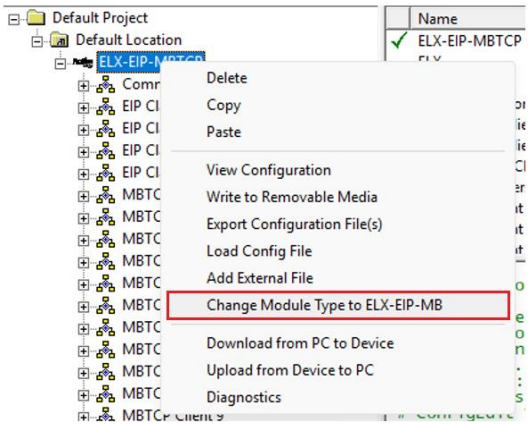
Note: When uploading an ELX-EIP-MBTCP configuration, proceed with steps **7** and **8** to convert to an ELX-EIP-MB.

7 The *Module Type* displays as **ELX-EIP-MBTCP** in the *Module Configuration* view.

```
# Module Configuration

[Module]
Module Type : ELX-EIP-MBTCP
```

8 Right-click the **ELX-EIP-MBTCP** icon and select **CHANGE MODULE TYPE TO ELX-EIP-MB**. to update the ELX-EIP-MBTCP to ELX-EIP-MB.

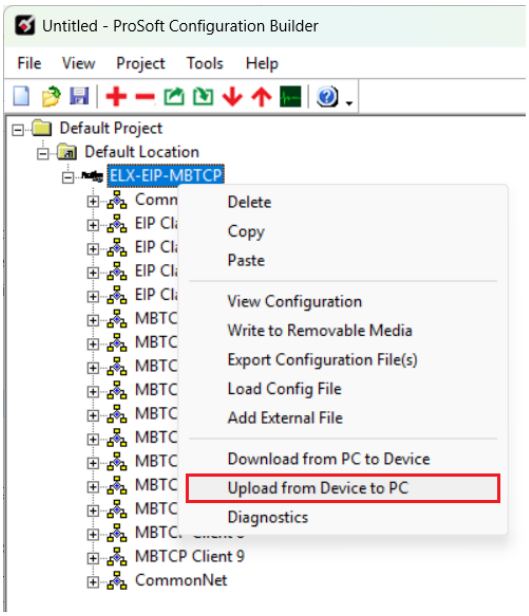


5.4.2 Uploading a Configuration to an ELX-EIP-MBTCP Project

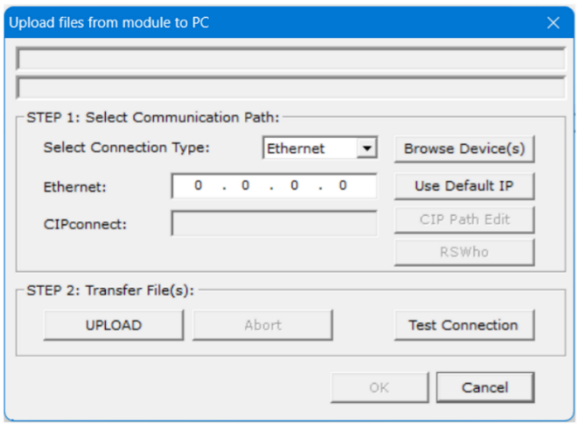
Note: If uploading an ELX-EIP-MB configuration to an ELX-EIP-MBTCP project, all the configuration settings of the ELX-EIP-MB container will be uploaded to the ELX-EIP-MBTCP project, including the *MBS Port 1* configuration settings. However, the MBS port will not be enabled/active.

To upload the project file from the container to the PC:

1 In the tree view in *ProSoft Configuration Builder*, right-click the **ELX-EIP-MBTCP** icon and then select **UPLOAD FROM DEVICE TO PC**. This opens the *Upload* dialog.

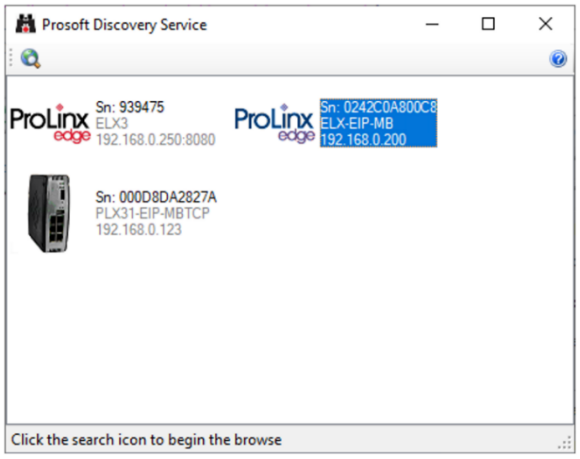


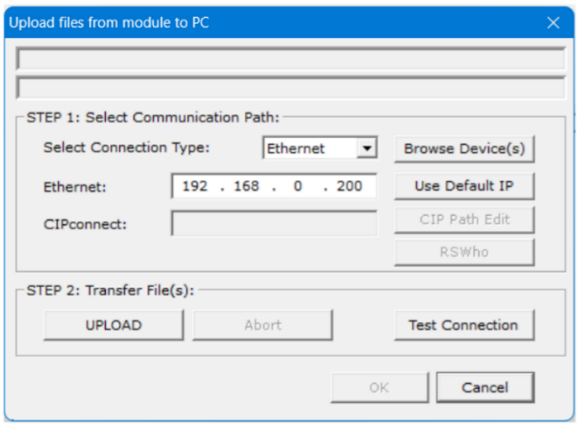
2 In the *Upload files from module to PC* dialog, in the *Select Connection Type* dropdown box, select the default **ETHERNET** option.



3 Click **BROWSE DEVICE(S)** to select the container from the ProSoft Discovery Service dialog.

4 Double-click on the container icon. This will populate the target IP address for the download.





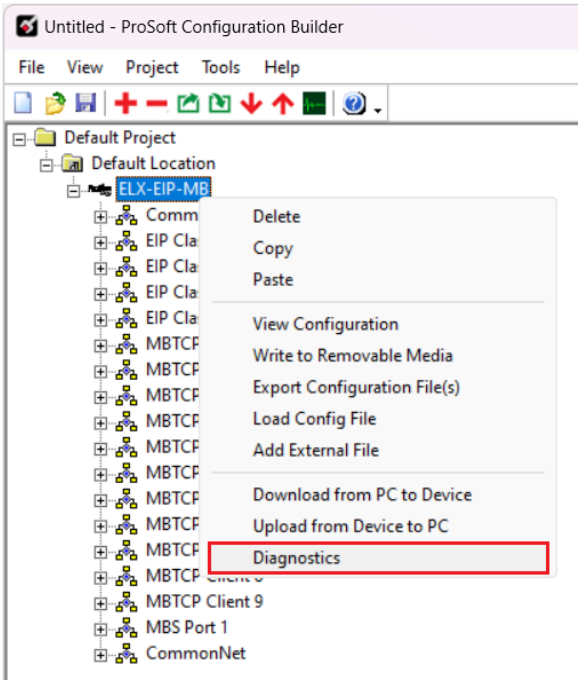
5 Click **TEST CONNECTION** to verify that the IP address allows access to the container.

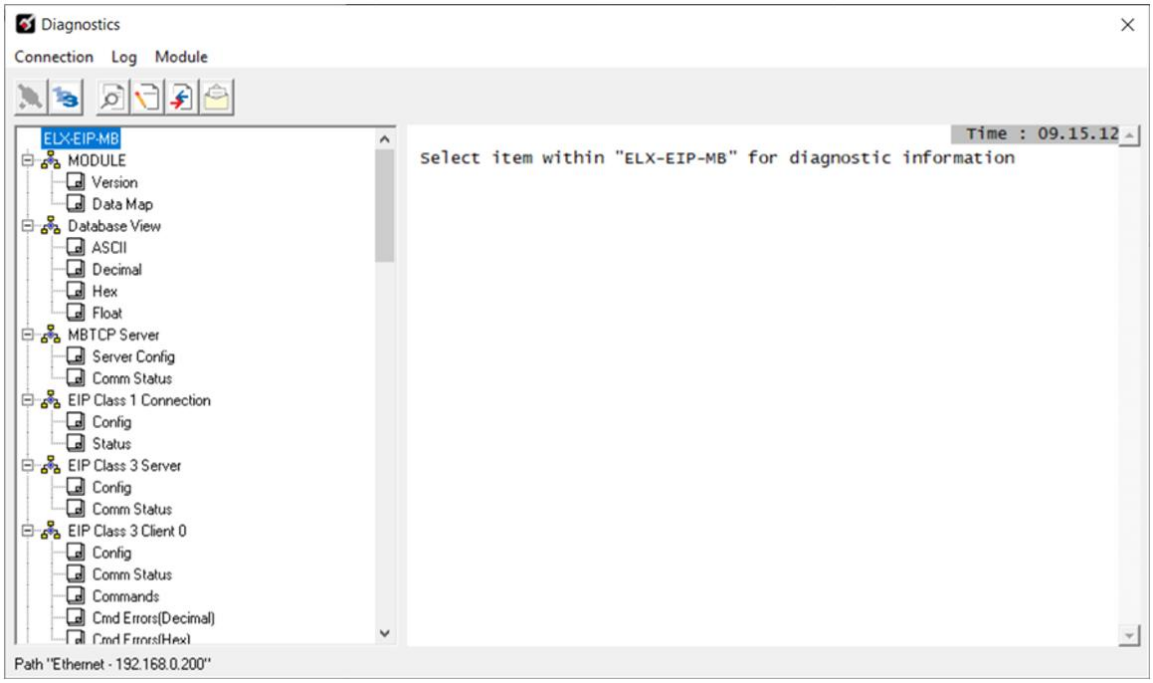
6 If the connection succeeds, click **UPLOAD** to upload the project to PCB.

5.5 Diagnostics and Troubleshooting

ProSoft Configuration Builder has many useful tools to help with diagnostics and troubleshooting. Use PCB to connect to the container and retrieve status and configuration data.

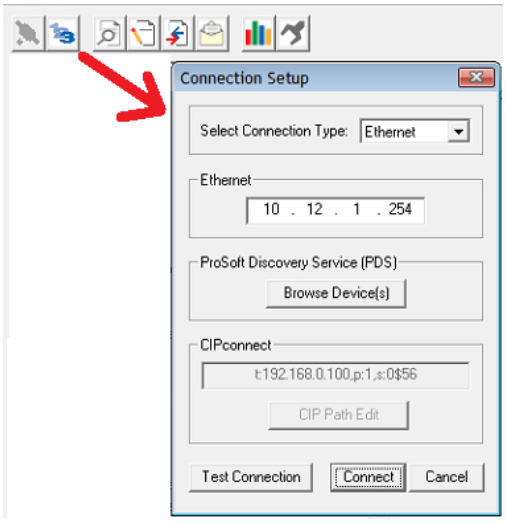
In PCB, right-click on the container name and select **DIAGNOSTICS** to open the *Diagnostics* dialog.





If there is no response from the container, follow these steps:

1 From the toolbar, click the **SETUP CONNECTION** button.



2 In the *Connection Setup* dialog box, select **ETHERNET** from the **SELECT CONNECTION TYPE** list.

3 Enter the container's IP address in the **ETHERNET** field, or click **BROWSE DEVICE(S)**.

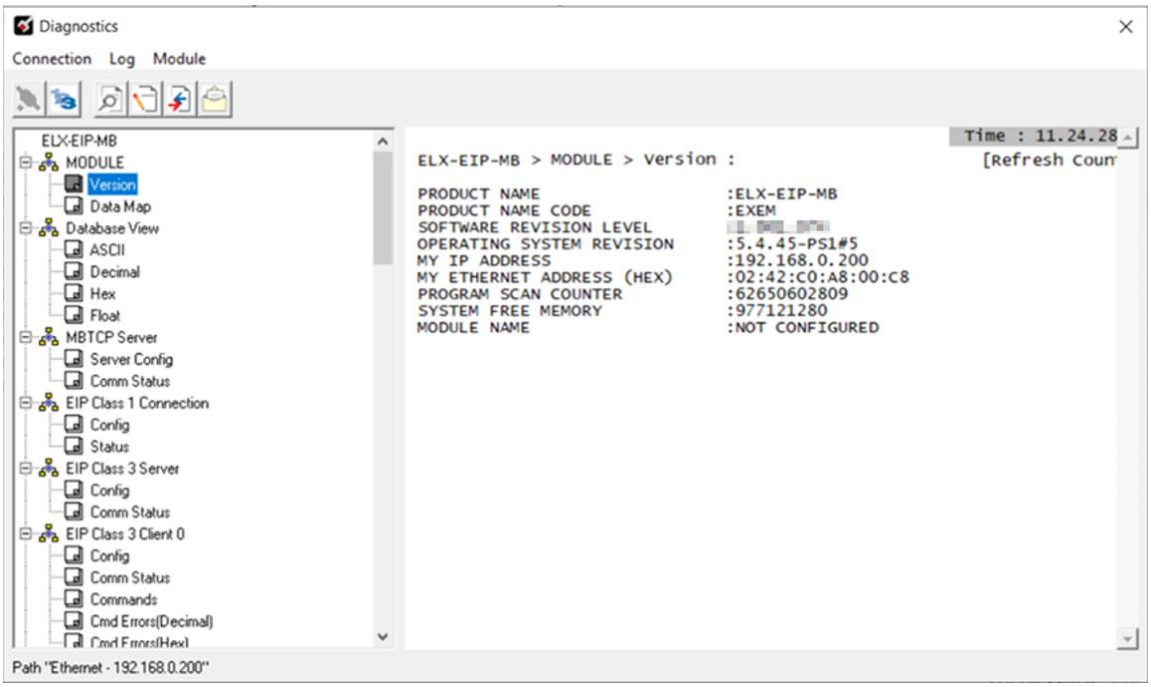
4 Click **CONNECT**.

5 Verify that the Ethernet cable is properly connected between the computer's communication port and the container's assigned ethernet port (Port 1 by default).

6 If a connection is still not established, contact ProSoft Technology Technical Support for assistance.

5.5.1 Diagnostics Menu

The Diagnostics menu is arranged as a tree structure in the left side of the *Diagnostics* window.



Menu Command	Submenu Command	Description
Module	Version	Displays the container's current software version and other important values.
	Data Map	Displays the container's Data Map configuration.
Database View	ASCII	Displays the contents of the container's database in ASCII character format.*
	Decimal	Displays the contents of the container's database in decimal number format.*
	Hex	Displays the contents of the container's database in hexadecimal number format.*
	Float	Displays the contents of the container's database in floating-point number format.*

* Use the scroll bar on the right edge of the window to navigate through the database. Each page displays 100 words of data.

5.5.2 Capturing a Diagnostic Session to a Log File

A Diagnostics session can be captured to a log file. This feature can be useful for troubleshooting and record-keeping purposes, and for communication with ProSoft Technology's Technical Support team.

To capture session data to a log file

1 Open a *Diagnostics* window.

2 To log a Diagnostics session to a text file, from the toolbar, click the **LOG FILE** button. Click the button again to stop the capture.



3 To view the log file, from the toolbar, click the **VIEW LOG FILE** button. The log file opens as a text file that can be renamed and saved to a different location.

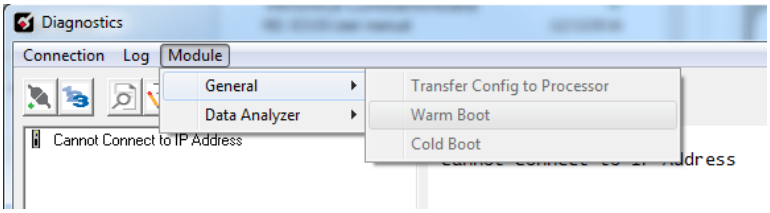


4 If multiple sequential sessions are captured, PCB appends the new data to the end of the previously captured data. To clear the previous data from the log file, click the **CLEAR DATA** button each time before capturing data.



5.5.3 Warm Boot / Cold Boot

Warm and Cold booting the container can be done by clicking **MODULE** > **GENERAL** > **WARM BOOT** or **COLD BOOT**.

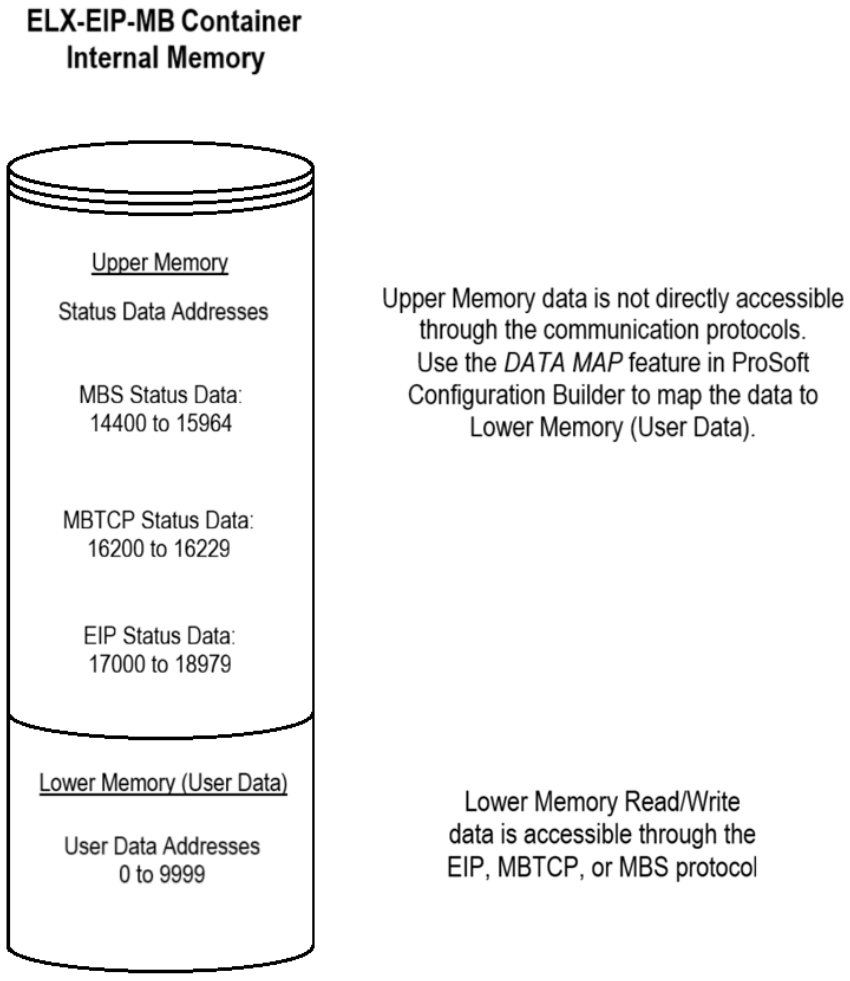


6 Internal Database

The internal database is central to the functionality of the ELX-EIP-MB container. The container shares this database between all the protocols of the ELX-EIP-MB container and uses it as a conduit to pass information from one protocol to another protocol. This permits data from devices using one protocol to be accessed and controlled by devices using another protocol.

In addition to data from the Client and Server, status and error information generated by the container can be mapped into the User Data area of the internal database. The internal database is divided into two areas:

- **Upper Memory**: Internal database range for ELX-EIP-MB container status data. This data range is reserved for the status data of the protocol drivers supported by the container.
- **Lower Memory (User Data)**: Internal database range for ELX-EIP-MB container User Data area. This is where incoming and outgoing EIP and MB protocol data from external devices is stored and accessed.



The protocols in the ELX-EIP-MB container can write data to and read data from the Lower Memory User Data area.

- If the protocol gateway container is acting as an EtherNet/IP or Modbus TCP/RTU/ASCII Client, commands are configured to read/write data from/to external Server devices. The incoming read-data is stored at a specified address of the ELX-EIP-MB container User Data area. The outgoing write-data is accessed from a specified address of the ELX-EIP-MB container User Data area.
- If the protocol gateway container is acting as an EtherNet/IP or Modbus TCP/RTU/ASCII Server, the external Client reads/writes data from/to a specific location in the ELX-EIP-MB container User Data area.

Note: For more information on specific EtherNet/IP and Modbus TCP/RTU/ASCII status data, please see:

EtherNet/IP: Section *7.4.2 EIP Status Data in Upper Memory*

Modbus TCP: Section *8.4.2 MBTCP Status Data in Upper Memory*

Modbus RTU/ASCII: Section *9.4.2 MBS Status Data in Upper Memory*

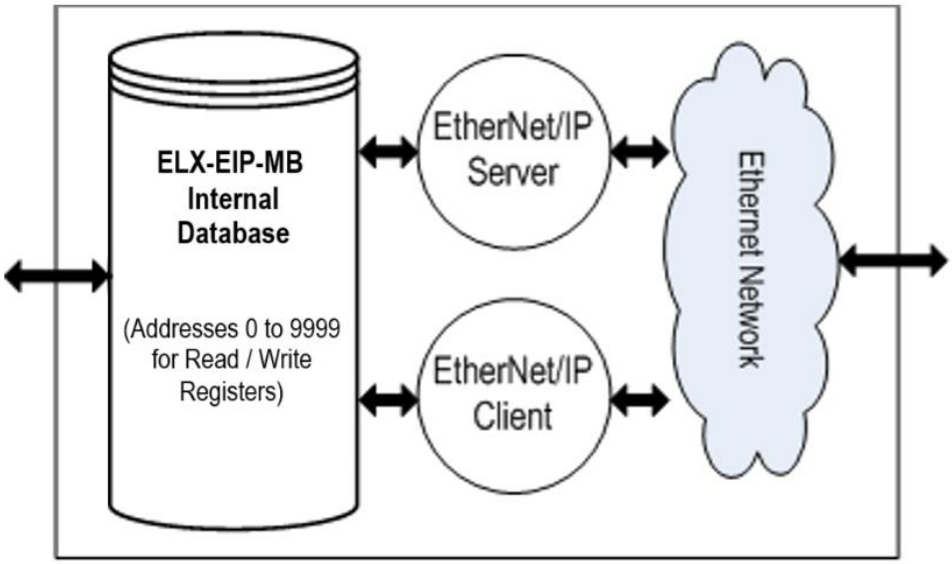
To access the protocol gateway container status data, use the *Data Mapping* feature to copy data from the container status data area (Upper Memory) to the User Data area (Lower Memory). For more information, please see chapter *10 Mapping Data in Container Memory*.

7 EIP Protocol

7.1 EIP Functional Overview

The EtherNet/IP™ (EIP) driver of the ELX-EIP-MB container is used to interface Rockwell Automation processors or other software-based EtherNet/IP™ solutions.

The following illustration shows the functionality of the EtherNet/IP protocol.



7.1.1 EtherNet/IP Specifications

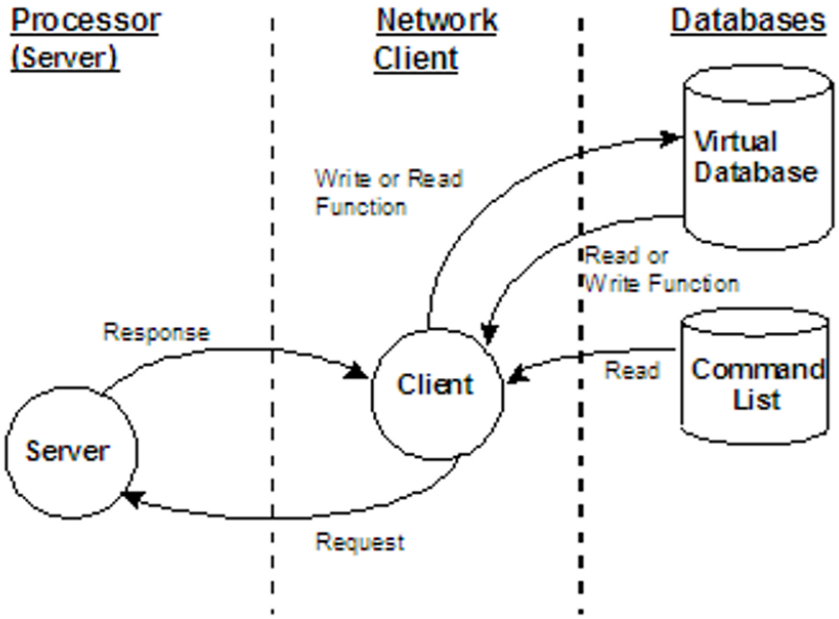
Specification	Description
Number of Class 3 Server Connections	5
Supported PLC Types	ControlLogix®, CompactLogix™
Supported Message Types	CIP
Class 3 Client Connections	Connected: 2 Unconnected: 1
Number of Class 1 I/O Connections	8
I/O connection sizes	Input: 1 to 248 Output: 1 to 248
RPI Time Ranges	8 Connections: 10ms minimum RPI for all connections
CIP Services Supported	0x4C: CIP Data Table Read 0x4D: CIP Data Table Write CIP Generic
Command List	Support for 100 commands per Client, each configurable for command type, IP address, register to/from addressing and word/bit count.

7.2 EIP Internal Database

7.2.1 EIP Client Access to ELX-EIP-MB Container Database

The Client functionality exchanges data between the ELX-EIP-MB container's internal database and EtherNet/IP Servers on the network. The command list defined in ProSoft Configuration Builder specifies the data to be transferred between the ELX-EIP-MB container and each of the EtherNet/IP Servers on the network.

The following illustration describes the flow of data between the EIP Client and the internal database.



7.2.2 EIP Server Access to ELX-EIP-MB Container Database

EtherNet/IP Server support in the ELX-EIP-MB container allows Client applications (such as HMI software and processors) to read from and write to the ELX-EIP-MB container’s database. The Server driver is able to support multiple concurrent connections from several Clients.

The EIP Server supports Class 3 Server functionality for CIP Generic and CIP Data Table Read/Write.

When configured as a Server, the user data 0 be seen on the network.

7.3 EIP Driver Configuration

7.3.1 EIP Class 1 Connection

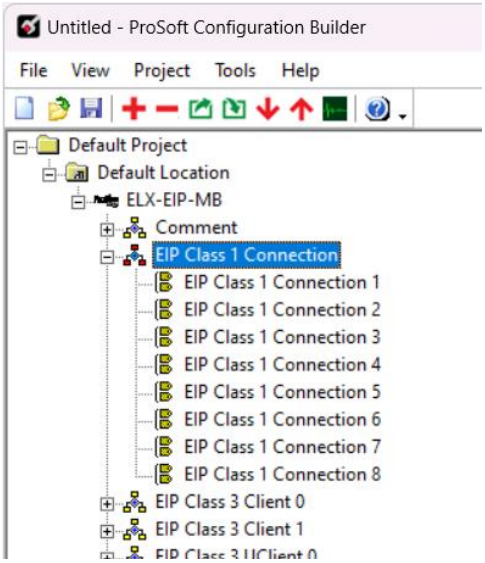
Use the *EIP Class 1 Connection* when the container acts as an EtherNet/IP Server transferring data to and from a PLC Client using a direct I/O connection. Direct I/O connections can quickly transfer large amounts of data.

The EIP driver can handle up to eight I/O connections, each with 248 words of input data and 248 words of output data.

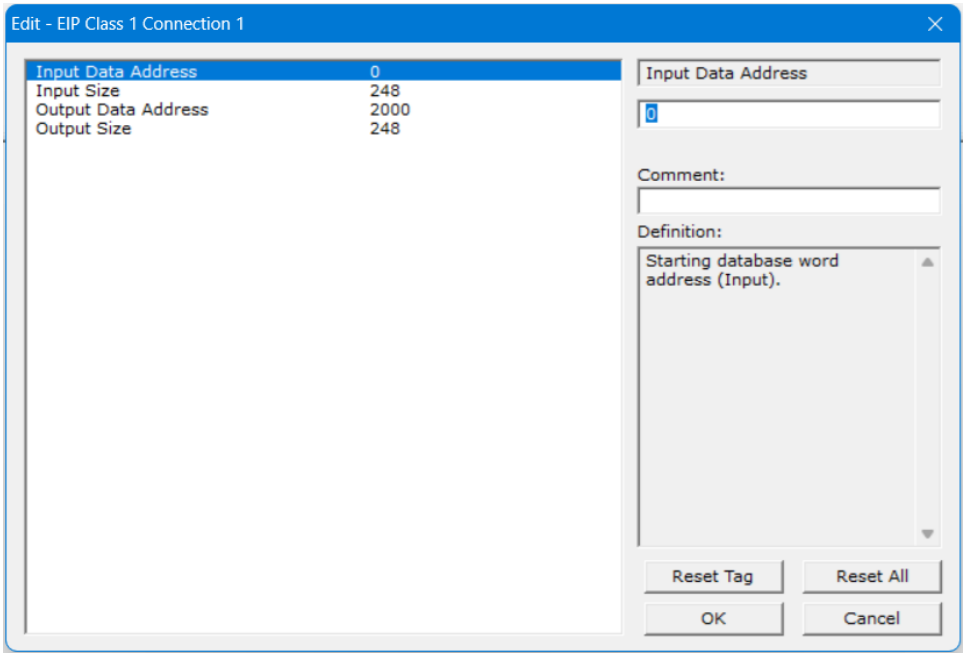
7.3.1.1 Configuration

After the ELX-EIP-MB container has been created in Studio/RSLogix 5000, the connections in the ELX-EIP-MB container can be configured in PCB.

1 In *ProSoft Configuration Builder*, click the **[+]** next to the *EIP Class 1 Connection*.



2 Double-click the *EIP Class 1 Connection [x]* to display the *Edit - EIP Class 1 Connection [x]* dialog.



3 In the dialog, click a parameter and enter a value for the parameter. There are four configurable parameters for each I/O connection:

Parameter	Value Range	Description
Input Data Address	0 to 9999	Specifies the starting address within the container's virtual database for data transferred from the container to the PLC.
Input Size	1 to 248	Specifies the number of Integers being transferred to the PLC's input image (248 integers max).
Output Data Address	0 to 9999	Specifies the starting address within the container's virtual database for data transferred from the PLC to the container.
Output Size	1 to 248	Specifies the number of integers being transferred to the PLC's output image (248 integers max).

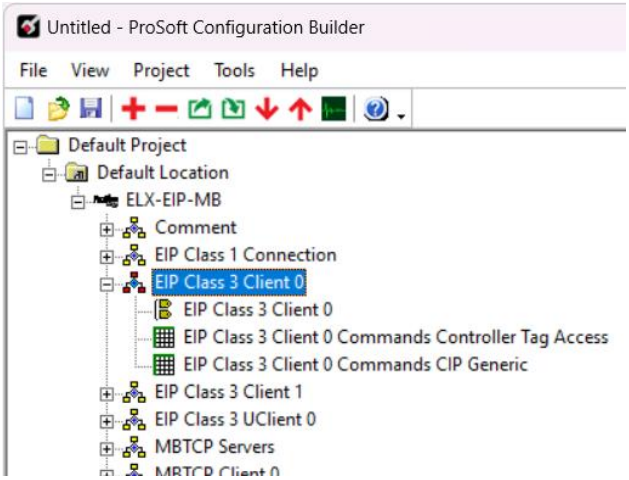
7.3.2 EIP Class 3 Client [x] Connection

Use the *EIP Class 3 Client [x]* connections when the ELX-EIP-MB container is acting as a Client initiating message instructions to the Server devices. The EIP driver supports 2 Connected Clients and 1 Unconnected Client, each configurable with up to 100 commands per Client. The EIP driver supports 1 unconnected Client connection. Typical applications include SCADA systems, and SLC communication.

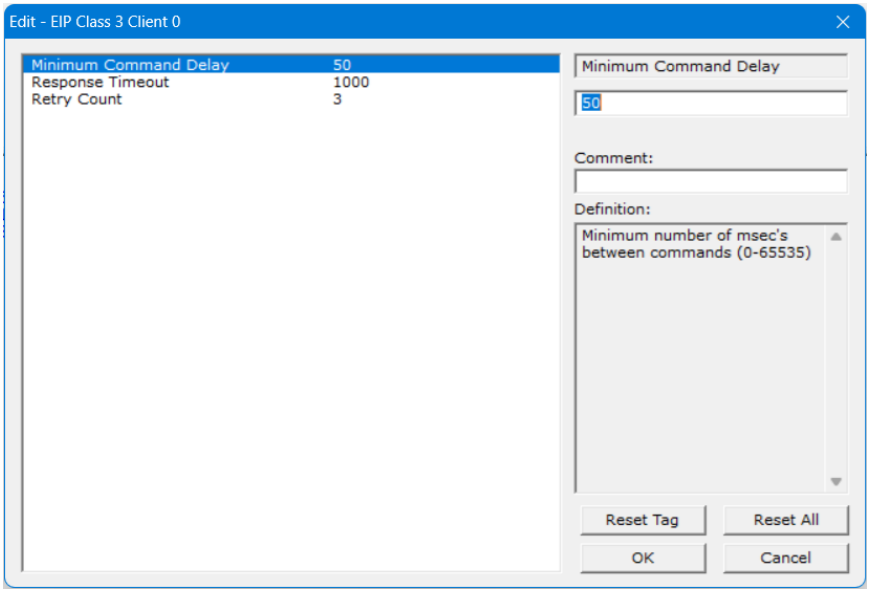
7.3.2.1 Configuration

To configure Class 3 Client [x] connections:

1 Click the **[+]** next to *EIP Class 3 Client [x]*.



2 Double-click the lower *EIP Class 3 Client [x]* icon to display the *Edit - EIP Class 3 Client [x]* dialog.



3 In the dialog, click any parameter to change its value. The following table specifies the configuration for the EIP Client device on the network port:

Parameter	Value	Description
Minimum Command Delay	0 to 65535 milliseconds	Specifies the number of milliseconds to wait between the initial issuances of a command. This parameter can be used to delay all commands sent to Servers to avoid "flooding" commands on the network. This parameter does not affect retries of a command as they will be issued when failure is recognized.
Response Timeout	0 to 65535 milliseconds	Specifies the amount of time in milliseconds that a Client will wait before re-transmitting a command if no response is received from the addressed Server. The value to use depends on the type of communication network used, and the expected response time of the slowest device connected to the network.
Retry Count	0 to 10	Specifies the number of times a command will be retried if it fails.

7.3.2.2 Commands

Each of the EIP Class 3 Client's configuration includes a separate command list for each supported command type (i.e. CIP Generic).

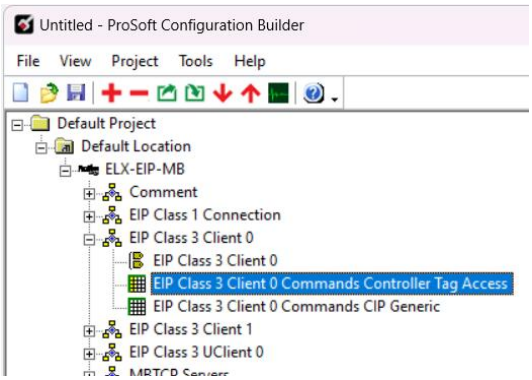
Each list is processed from top to bottom, one after the other, until all specified commands are completed, and then the polling process begins again.

This section defines the Class 3 Client [x] commands to be issued from the container to Server devices on the network. Use these commands for data collection and control of devices on the TCP/IP network. In order to interface the virtual database with Rockwell Automation Programmable Automation Controllers (PACs), Programmable Logic Controllers (PLCs), or other EtherNet/IP Server devices, a command list must be constructed using the parameters for each message type.

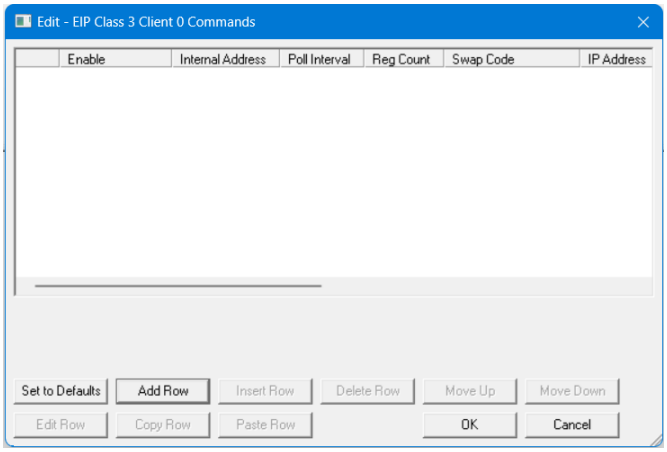
Note: It is recommended to group the commands based on the target IP Address of the End Device(s).

To add Class 3 Client [x] commands:

1. In ProSoft Configuration Builder, click the **[+]** next to *EIP Class 3 Client [x]*. Then double-click the desired command type to display the *Edit - EIP Class 3 Client [x] Commands* dialog.



2. Double-click the desired command type to display the *Edit - EIP Class 3 Client [x] Commands* dialog.



3 Click **ADD ROW** to add a new command.

4 Click **EDIT ROW** or double-click the row to display the *Edit* dialog to configure the command.

7.3.2.2.1 Controller Tag Access Command

Parameter	Value	Description
Enable	Enable Disable Conditional Write	Specifies if the command should be executed and under what conditions. **ENABLE**: The Command is executed each scan of the command list **DISABLE**: The command is disabled and will not be executed **CONDITIONAL WRITE**: The Command executes only if the data in the associated *Internal Address* has changed.
Internal Address	0 to 9999	Specifies the database address in the container's internal database to be associated with the command. If the command is a read function, the data received in the response message is placed at the specified location. If the command is a write function data used in the command is sourced from specified data area.
Poll Interval	0 to 65535	Specifies the minimum interval to execute this command. The parameter is entered in 1/10 of a second. For example, if a value of 100 is entered, this command executes no more frequently than every 10 seconds.
Reg Count	0 to x*	Specifies the number of data points to be read from or written to the target device.
Swap Code	None Word swap Word and Byte swap Byte swap	Specifies if the data from the Server is to be ordered differently than it was received. This parameter is typically used when dealing with floating-point or other multi-register values. **NONE**: No change is made (ABCD) **WORD SWAP**: The words are swapped (CDAB) **WORD AND BYTE SWAP**: The words and bytes are swapped (DCBA) **BYTE SWAP**: The bytes are swapped (BADC)
IP Address	xxx.xxx.xxx.xxx	Specifies the IP address of the Server to be addressed by this command.
Slot	-1	Specifies the slot number for the device. When addressing a processor in a ControlLogix or CompactLogix, the slot number corresponds to the slot in the rack containing the controller being addressed.
Func Code	332 333	Specifies the function code to be used in the command. **332**: CIP Data Table Read **333**: CIP Data Table Write
Data Type	BOOL SINT INT DINT REAL DWORD	Specifies the data type of the target controller tag name.
Tag Name		Specifies the controller tag in the target PLC.
Offset	0 to 65535	Specifies the offset database where the value corresponds to the Tag Name parameter.
Comment		Optional 32-character comment for the command.

7.3.2.2.2 CIP Generic Command

Parameter	Value	Description
Enable	Disabled Enabled Conditional Write	Specifies the condition to execute the command. **DISABLED**: The command is disabled and will not be executed. **ENABLED**: The command is executed on each scan of the command list if the *Poll Interval* is set to zero. If the *Poll Interval* is non-zero, the command is executed when the interval timer expires. **CONDITIONAL WRITE**: The command executes only if the internal data value(s) to be sent has changed.
Internal Address	0 to 9999	Specifies the database address in the container's internal database to be associated with the command. If the command is a read function, the data received in the response message is placed at the specified location. If the command is a write function, data used in the command is sourced from specified data area.
Poll Interval	0 to 65535	Specifies the minimum interval to execute continuous commands. The parameter is entered in 1/10 of a second. For example, if a value of **100** is entered for a command, the command executes no more frequently than every 10 seconds.
Reg Count	0 to x*	Specifies the number of data points to be read from or written to the target device. *The maximum value depends on the selected *Data Type* used in the command.
Swap Code	None Word swap Word and Byte swap Byte swap	Specifies if the data from the Server is to be ordered differently than it was received. This parameter is typically used when dealing with floating-point or other multi-register values. **NONE**: No change is made (ABCD) **WORD SWAP**: The words are swapped (CDAB) **WORD AND BYTE SWAP**: The words and bytes are swapped (DCBA) **BYTE SWAP**: The bytes are swapped (BADC)
IP Address	xxx.xxx.xxx.xxx	Specifies the IP address of the target device to be addressed by this command.
Slot	-1	Use **-1** to target a connected device. Use > **-1** to target a device in a specific slot number within the rack.
Func Code	CIP Generic	Used to read/write the attributes of any object by using an explicit address
Service Code	00 to FF (Hex)	An integer identification value which denotes a particular Object Instance and/or Object class function. For more information refer to ODVA CIP specification.
Class	00 to FFFF (Hex)	An integer identification value assigned to each Object Class accessible from the network. For more information, refer to ODVA CIP specification.
Instance	Application-dependent	An integer identification value assigned to an Object Instance that identifies it among all Instances of the same Class. For more information, refer to ODVA CIP specification.
Attribute	00 to FFFF (Hex)	An integer identification value assigned to a Class and/or Instance Attribute. For more information, refer to ODVA CIP specification.
Comment		This field can be used to give a 32-character comment to the command. The "**:**" and "**#**" characters are reserved characters. It is strongly recommended not be use in the comment section.

7.3.3 EIP Class 3 UClient Connection

The ELX-EIP-MB container supports one unconnected Client (most devices use connected Clients; be sure refer to the user manual for the target device for verification).

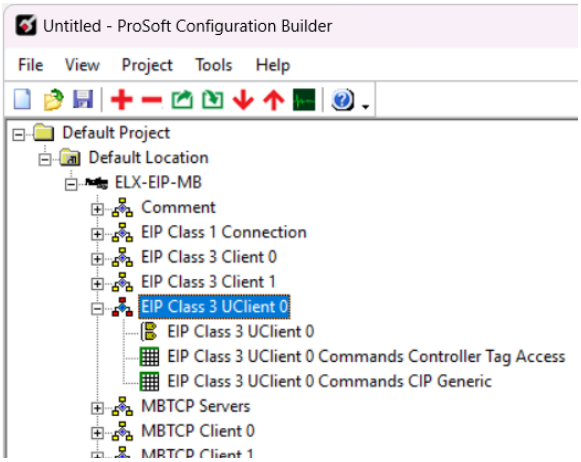
Use the *EIP Class 3 UClient 0* connection when the ELX-EIP-MB container is acting as a Client initiating message instructions to the Server devices. Unconnected messaging is a type of EtherNet/IP explicit messaging that uses TCP/IP implementation.

Certain devices, such as the AB Power Monitor 3000 series B, support unconnected messaging. Check the device documentation for further information about its EtherNet/IP implementation.

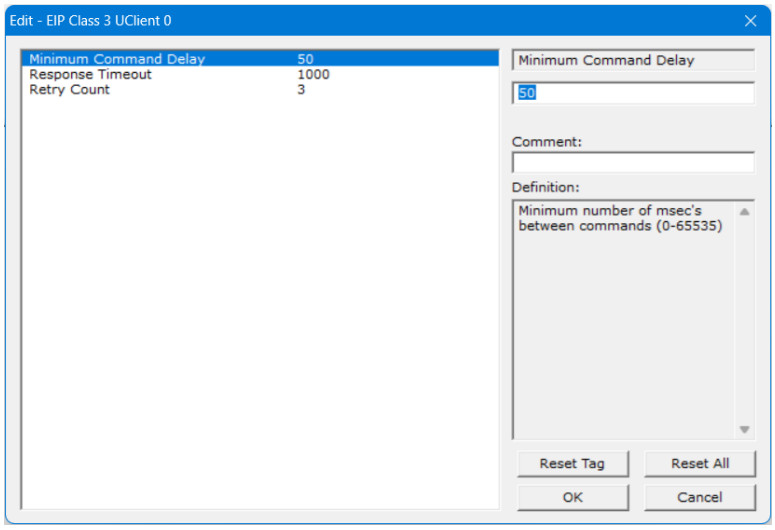
7.3.3.1 Configuration

To configure a Class 3 UClient connection:

1 Click the **[+]** next to *EIP Class 3 UClient 0*.



2 Double-click the lower *EIP Class 3 Client [x]* icon to display the *Edit - EIP Class 3 Client [x]* dialog.



3 In the dialog, click any parameter to change its value. The following table specifies the configuration for the EIP UClient device on the network port:

Parameter	Value	Description
Minimum Command Delay	0 to 65535 milliseconds	Specifies the number of milliseconds to wait between the initial issuances of a command. This parameter can be used to delay all commands sent to Servers to avoid "flooding" commands on the network. This parameter does not affect retries of a command as they will be issued when failure is recognized.
Response Timeout	0 to 65535 milliseconds	Specifies the amount of time in milliseconds that a Client will wait before re-transmitting a command if no response is received from the addressed Server. The value to use depends on the type of communication network used, and the expected response time of the slowest device connected to the network.
Retry Count	0 to 10	Specifies the number of times a command will be retried if it fails.

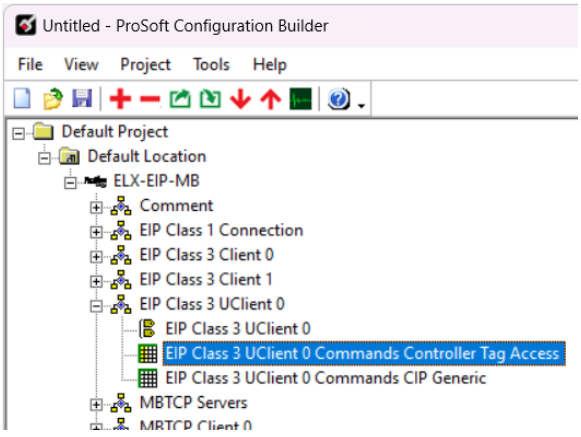
7.3.3.2 Commands

There is a separate command list for each of the different message types supported by the protocol. Each list is processed from top to bottom, one after the other, until all specified commands are completed, and then the polling process begins again.

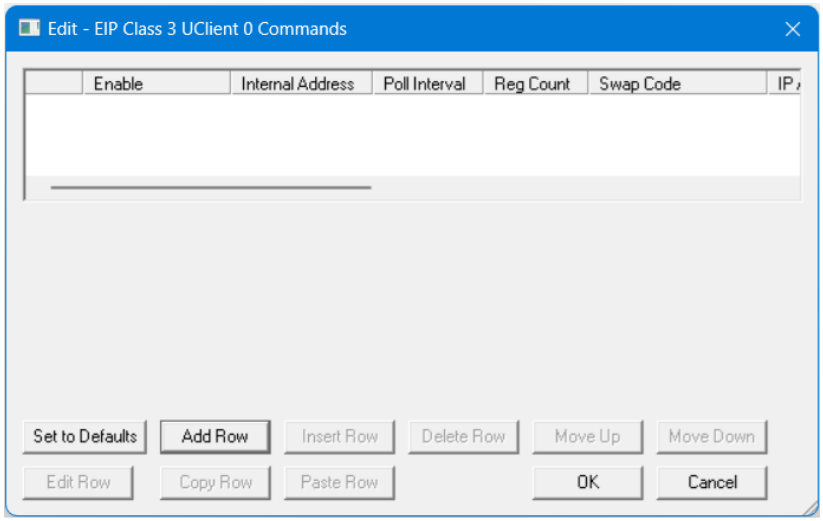
This section defines the EtherNet/IP commands to be issued from the container to Server devices on the network. Use these commands for data collection and control of devices on the TCP/IP network. In order to interface the virtual database with Rockwell Automation Programmable Automation Controllers (PACs), Programmable Logic Controllers (PLCs), or other EtherNet/IP Server devices, a command list must be constructed using the parameters for each message type.

To add Class 3 UClient 0 commands:

1 In ProSoft Configuration Builder, click the **[+]** next to *EIP Class 3 UClient 0*. Then double-click the desired command type to display the *Edit - EIP Class 3 UClient 0 Commands* dialog.



2 Double-click the desired command type to display the *Edit - EIP Class 3 UClient 0 Commands* dialog.



3 Click **ADD ROW** to add a new command.

4 Click **EDIT ROW** or double-click the row to display the *Edit* dialog to configure the command.

7.3.3.2.1 Controller Tag Access Command

Parameter	Value	Description
Enable	Enable Disable Conditional Write	Specifies if the command should be executed and under what conditions. **ENABLE**: The Command is executed each scan of the command list **DISABLE**: The command is disabled and will not be executed **CONDITIONAL WRITE**: The Command executes only if the internal data associated with the command changes
Internal Address	0 to 9999	Specifies the database address in the container's internal database to be associated with the command. If the command is a read function, the data received in the response message is placed at the specified location. If the command is a write function data used in the command is sourced from specified data area.
Poll Interval	0 to 65535	Specifies the minimum interval to execute continuous commands. The parameter is entered in 1/10 of a second. If a value of **100** is entered for a command, the command executes no more frequently than every 10 seconds.
Reg Count	0 to x*	Specifies the number of data points to be read from or written to the target device. *The maximum value depends on the selected *Data Type* used in the command.
Swap Code	None Word swap Word and Byte swap Byte swap	Specifies if the data from the Server is to be ordered differently than it was received. This parameter is typically used when dealing with floating-point or other multi-register values. **NONE**: No change is made (ABCD) **WORD SWAP**: The words are swapped (CDAB) **WORD AND BYTE SWAP**: The words and bytes are swapped (DCBA) **BYTE SWAP**: The bytes are swapped (BADC)
IP Address	xxx.xxx.xxx.xxx	Specifies the IP address of the target device to be addressed by this command.
Slot	-1	Specifies the slot number for the device. Use **-1** to target a connected device. Use > **-1** to target a device in a specific slot number within the rack.
Func Code	332 333	Specifies the function code to be used in the command. **332**: CIP Data Table Read **333**: CIP Data Table Write
Data Type	BOOL SINT INT DINT REAL DWORD	Specifies the data type of the target controller tag name.
Tag Name		Specifies the controller tag in the target PLC.
Offset	0 to 65535	Specifies the offset database where the value corresponds to the Tag Name parameter
Comment		Optional 32-character comment for the command.

7.3.3.2.2 CIP Generic Command

Parameter	Value	Description
Enable	Disabled Enabled Conditional Write	Specifies the condition to execute the command. **DISABLED**: The command is disabled and will not be executed. **ENABLED**: The command is executed on each scan of the command list if the *Poll Interval* is set to zero. If the *Poll Interval* is non-zero, the command is executed when the interval timer expires. **CONDITIONAL WRITE**: The command executes only if the internal data value(s) to be sent has changed.
Internal Address	0 to 9999	Specifies the database address in the container's internal database to be associated with the command. If the command is a read function, the data received in the response message is placed at the specified location. If the command is a write function, data used in the command is sourced from specified data area.
Poll Interval	0 to 65535	Specifies the minimum interval to execute continuous commands. The parameter is entered in 1/10 of a second. For example, if a value of **100** is entered for a command, the command executes no more frequently than every 10 seconds.
Reg Count	0 to x*	Specifies the number of data points to be read from or written to the target device. *The maximum value depends on the selected *Data Type* used in the command.
Swap Code	None Word swap Word and Byte swap Byte swap	Specifies if the data from the Server is to be ordered differently than it was received. This parameter is typically used when dealing with floating-point or other multi-register values. **NONE**: No change is made (ABCD) **WORD SWAP**: The words are swapped (CDAB) **WORD AND BYTE SWAP**: The words and bytes are swapped (DCBA) **BYTE SWAP**: The bytes are swapped (BADC)
IP Address	xxx.xxx.xxx.xxx	Specifies the IP address of the target device to be addressed by this command.
Slot	-1	Specifies the slot number for the device. Use **-1** to target a connected device. Use > **-1** to target a device in a specific slot number within the rack.
Func Code	CIP Generic	Used to read/write the attributes of any object by using an explicit address
Service Code	00 to FF (Hex)	An integer identification value which denotes a particular Object Instance and/or Object class function. For more information refer to ODVA CIP specification.
Class	00 to FFFF (Hex)	An integer identification value assigned to each Object Class accessible from the network. For more information, refer to ODVA CIP specification.
Instance	Application-dependent	An integer identification value assigned to an Object Instance that identifies it among all Instances of the same Class. For more information, refer to ODVA CIP specification.
Attribute	00 to FFFF (Hex)	An integer identification value assigned to a Class and/or Instance Attribute. For more information, refer to ODVA CIP specification.
Comment		This field can be used to give a 32-character comment to the command. The "**:**" and "**#**" characters are reserved characters. It is strongly recommended not be use in the comment section.

7.3.4 EIP Class 3 Server

The ELX-EIP-MB container can act as an EtherNet/IP Class 3 Server device responding to Connected/Unconnected messages initiated from a Client device such as an HMI, DCS, PLC, or PAC. It supports CIP Generic, CIP Data Table Read and Write messages.

7.3.4.1 Explicit Messaging Support

The ELX-EIP-MB Container supports the following explicit message types:

- **CIP Controller Tag Access**
  This command is used to read or write a controller tag value. A maximum of 125 elements can be configured for each command.
- **CIP Generic**
  This command is used to configure the CIP parameters (Class, Service, Instance, Attribute) for the container to issue a CIP command to an EtherNet/IP device. A maximum of 125 elements can be configured for each command.

The following table refers to the User Data area in the ELX-EIP-MB container's 10,000 register memory:

Data Type	Tag Name	Length of Each Element in CIP Message	Array Range
BOOL	BOOLData[ ]	1	0 to 159999
Bit Array	BITAData[ ]	4	0 to 4999
SINT	SINTData[ ]	1	0 to 19999
INT	INT_Data[ ]	2	0 to 9999
DINT	DINTData[ ]	4	0 to 4999
REAL	REALData[ ]	4	0 to 4999

Note: The maximum length for a CIP message is 500 bytes.

7.3.4.1.1 CIP Data Table Read and Write

The following table defines the relationship of the User Data area in the ELX-EIP-MB container's internal database to the addresses required in the MSG CIP instructions:

Database Address	CIP Integer (INT)	CIP Boolean (BOOL)	CIP Bit Array (BOOL[32])	CIP Short Integer (SINT)	CIP Double Integer (DINT)	CIP Floating Point (REAL)
0	INT_Data[0]	BOOLData[0]	BITAData[0]	SINTData[0]	DINTData[0]	REALData[0]
999	INT_Data[999]	BOOLData[15984]		SINTData[1998]		
1000	INT_Data[1000]	BOOLData[16000]	BITAData[500]	SINTData[2000]	DINTData[500]	REALData[500]
1999	INT_Data[1999]	BOOLData[31984]		SINTData[3998]		
2000	INT_Data[2000]	BOOLData[32000]	BITAData[1000]	SINTData[4000]	DINTData[1000]	REALData[1000]
2999	INT_Data[2999]	BOOLData[47984]		SINTData[5998]		
3000	INT_Data[3000]	BOOLData[48000]	BITAData[1500]	SINTData[6000]	DINTData[1500]	REALData[1500]
3999	INT_Data[3999]	BOOLData[63999]		SINTData[7998]		
9998	INT_Data[9998]	BOOLData[159968]	BITAData[4999]	SINTData[19996]	DINTData[4999]	REALIData[4999]
9999	INT_Data[9999]	BOOLData[159984]		SINTData[19998]		

7.4 Network Diagnostics

7.4.1 EIP PCB Diagnostics

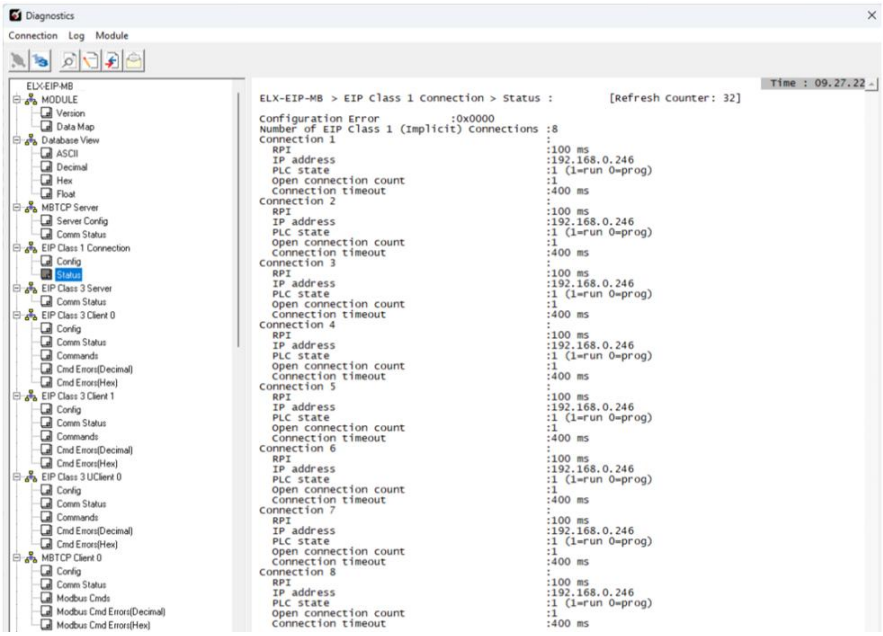
Use ProSoft Configuration Builder to troubleshoot the EIP driver communications. For instructions on how to access the diagnostics, see section *5.5 Diagnostics and Troubleshooting*.

The following table summarizes the status information available in ProSoft Configuration Builder for the EIP driver:

Connection Type	Submenu Item	Description
EIP Class 1	Config	Configuration settings for Class 1 Connections.
	Status	Status of the Class 1 Connections. Displays any configuration error, as well as the number of Class 1 Connections.
EIP Class 3 Server	Comm Status	Status information for each Class 3 Server Connection. Displays port numbers, IP addresses, socket status, and read/write counts.
EIP Class 3 Client and UClient 0	Config	Configuration settings for Class 3 Client/UClient Connections.
	Comm Status	Status information for Class 3 Client/UClient commands. Displays a summary of all the errors resulting from Class 3 Client/UClient commands.
	Commands	Configuration for the Class 3 Client/UClient command list.
	Cmd Errors (Decimal)	Current error codes for each command on the Class 3 Client/UClient command list in decimal number format. A zero means there is currently no error for the command.
	Cmd Errors (Hex)	Current error codes for each command on the Class 3 Client/UClient command list in hexadecimal number format. A zero means there is currently no error for the command.

Example:

EIP Class 1 Connection Status in PCB:



For specific information on error codes, see section *7.5 EIP Error Codes*.

7.4.2 EIP Status Data in Upper Memory

The EIP driver has an associated status data area located in the ELX-EIP-MB container's upper memory. The *Data Map* functionality of the ELX-EIP-MB container can be used to map this data into the normal user data range of the ELX-EIP-MB container database.

Note that all the status values are initialized to zero (0) at power-up, cold boot and during warm boot.

7.4.2.1 EIP Client Status Data

The following table lists the addresses in upper memory the ELX-EIP-MB container stores general error and status data for each EIP connected and unconnected Client:

EIP Client	Address Range
Connected Client 0	17900 through 17909
Connected Client 1	18100 through 18109
Unconnected Client 0	22800 through 22809

The content of each Client's status data area is structured in the same way. The following table describes the content of each register in the status data area:

Offset	Description
0	Number of Command Requests
1	Number of Command Responses
2	Number of Command Errors
3	Number of Requests
4	Number of Responses
5	Number of Errors Sent
6	Number of Errors Received
7	Reserved
8	Current Error Code
9	Last Error Code

7.4.2.2 EIP Client Command List Error Data

The ELX-EIP-MB container stores a status/error code in upper memory for each command in each EIP Client's command list. The following table lists the addresses in upper memory where the ELX-EIP-MB container stores the command list error data for each EIP Client:

EIP Client	Address Range
Connected Client 0	17910 through 18009
Connected Client 1	18110 through 18209
Unconnected Client 0	22810 through 22909

The first word in each Client's command list error data area contains the status/error code for the first command in the Client's command list. Each successive word in the command error list is associated with the next command in the list. Therefore, the size of the command list error data area depends on the number of commands defined.

The structure of the command list error data area (same for all Clients) is displayed in the following table:

Offset	Description
0	Command #1 Error Code
1	Command #2 Error Code
2	Command #3 Error Code
3	Command #4 Error Code
4	Command #5 Error Code
…	…
97	Command #98 Error Code
98	Command #99 Error Code
99	Command #100 Error Code

7.4.2.3 EIP Class 1 Server Status Data

The following table lists the addresses in upper memory where the ELX-EIP-MB container stores the Open Connection Count for each EIP Class 1 Server.

EIP Class 1 Server	Address Range	Description
	17000	Bit map of PLC State for each Connection 1 to 8. **0** = Run, **1** = Program
1	17001	Open Connection Count for Connection 1
2	17002	Open Connection Count for Connection 2
3	17003	Open Connection Count for Connection 3
4	17004	Open Connection Count for Connection 4
5	17005	Open Connection Count for Connection 5
6	17006	Open Connection Count for Connection 6
7	17007	Open Connection Count for Connection 7
8	17008	Open Connection Count for Connection 8

7.4.2.4 EIP Class 3 Server Status Data

The following table lists the addresses in upper memory where the ELX-EIP-MB container stores status data for each EIP Class 3 Server:

EIP Class 3 Server	Address Range
0	18900 through 18915
1	18916 through 18931
2	18932 through 18947
3	18948 through 18963
4	18964 through 18979

The content of each Server's status data area is structured the same. The following table describes the content of each register in the status data area:

Offset	Description
0 through 1	Connection State
2 through 3	Open Connection Count
4 through 5	Socket Read Count
6 through 7	Socket Write Count
8 through 15	Peer IP

7.5 EIP Error Codes

The ELX-EIP-MB container stores error codes returned from the command list process in the command list error memory region. A register is allocated for each command in the memory area. The error codes are formatted in the register as follows: The least-significant byte of the word contains the extended status code and the most-significant byte contains the status code.

Use the error codes returned for each command in the list to determine the success or failure of the command. If the command fails, use the error code to determine the cause of failure.

Warning: The application-specific error codes (not EtherNet/IP compliant) are returned from the gateway and never returned from an attached EtherNet/IP Server device. These are error codes that are part of the EtherNet/IP protocol or are extended codes unique to the ELX-EIP-MB container. The most common errors for the EtherNet/IP protocol are shown below.

7.5.1 EIP Error Codes

Code (Int)	Code (Hex)	Description
-8	0xFFF8	Data size too large
-11	0xFFF5	Timeout waiting for response after request
-12	0xFFF4	Reply data does not match requested byte count
-15	0x000F	Write protection enabled
5	0x0005	Class or instance not supported

7.5.2 EtherNet/IP CIP Generic Error Codes

Code (Int)	Code (Hex)	Description
8	0x0008	Service not supported
20	0x0014	Attribute not supported
21	0x0015	Too much data

7.5.3 TCP/IP Interface Error Codes

Error (Int)	Error (Hex)	Description
-33	0xFFDF	Failed to connect to target
-34	0xFFDE	Failed to register session with target (timeout)
-35	0xFFDD	Failed forward open response timeout
-36	0xFFDC	Tag command response timeout
-37	0xFFDB	No TCP/IP connection error

7.5.4 Common Response Error Codes

Error (Int)	Error (Hex)	Description
-40	0xFFD8	Invalid response length
-41	0xFFD7	CPF item count is not correct
-42	0xFFD6	CPF address field error
-43	0xFFD5	CPF packet tag invalid
-44	0xFFD4	CPF bad command code
-45	0xFFD3	CPF status error reported
-46	0xFFD2	CPF incorrect connection ID value returned
-47	0xFFD1	Context field not matched
-48	0xFFD0	Incorrect session handle returned
-49	0xFFCF	CPF not correct message number

7.5.5 Register Session Response Error Codes

Error (Int)	Error (Hex)	Description
-50	0xFFCE	Message length received not valid
-51	0xFFCD	Status error reported
-52	0xFFCC	Invalid version

7.5.6 Forward Open Response Error Codes

Error (Int)	Error (Hex)	Description
-55	0xFFC9	Message length received not valid
-56	0xFFC8	Status error reported

7.5.7 EIP Class 1 Connection Error Codes

Error (Int)	Error (Hex)	Description
0	0x0000	No error
1	0x0001	Input Data Address is either less than 0 or greater than DB Address range
2	0x0002	Input Size is less than 0 or greater than 250
4	0x0004	Output Data Address is less than 0 or greater than DB Address range
8	0x0008	Output Size is less than 0 or greater than 248
16	0x0010	Data address is overlapped with connection Input data address
32	0x0020	Data address is overlapped with connection Output data address

7.6 EIP Reference

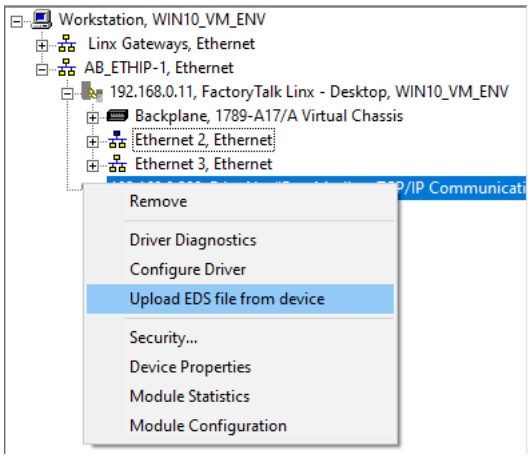
7.6.1 Adding the ELX-EIP-MB Container to RSLogix

The ELX-EIP-MB container can be added to RSLogix via EDS file (v20 and newer) or by adding it as a New Module (v19 and older).

7.6.1.1 Adding the ELX-EIP-MB Container EDS File to RSLogix/Studio 5000 v20+

1 Open Rockwell Automation *RSLinx* and browse to the ELX-EIP-MB container.

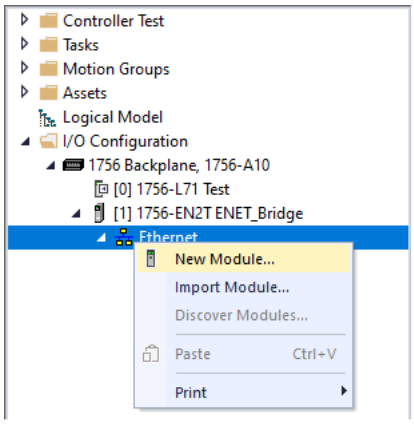
2 Right-click on the container and then choose **UPLOAD EDS FILE FROM DEVICE**.



Note: RSLogix5000 may need to be restarted to complete the EDS installation.

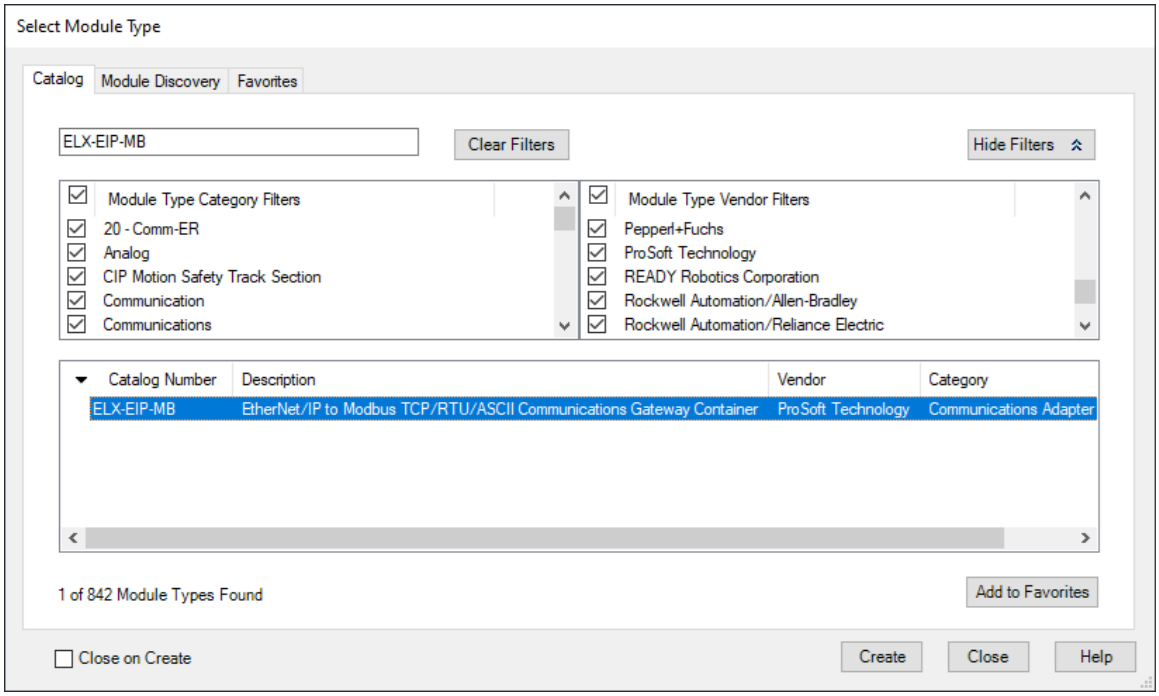
3 After RSLogix 5000 is restarted, open the desired RSLogix 5000 project.

4 In the *Controller Organizer*, right-click the EtherNet/IP bridge in the I/O tree and choose **NEW MODULE**.



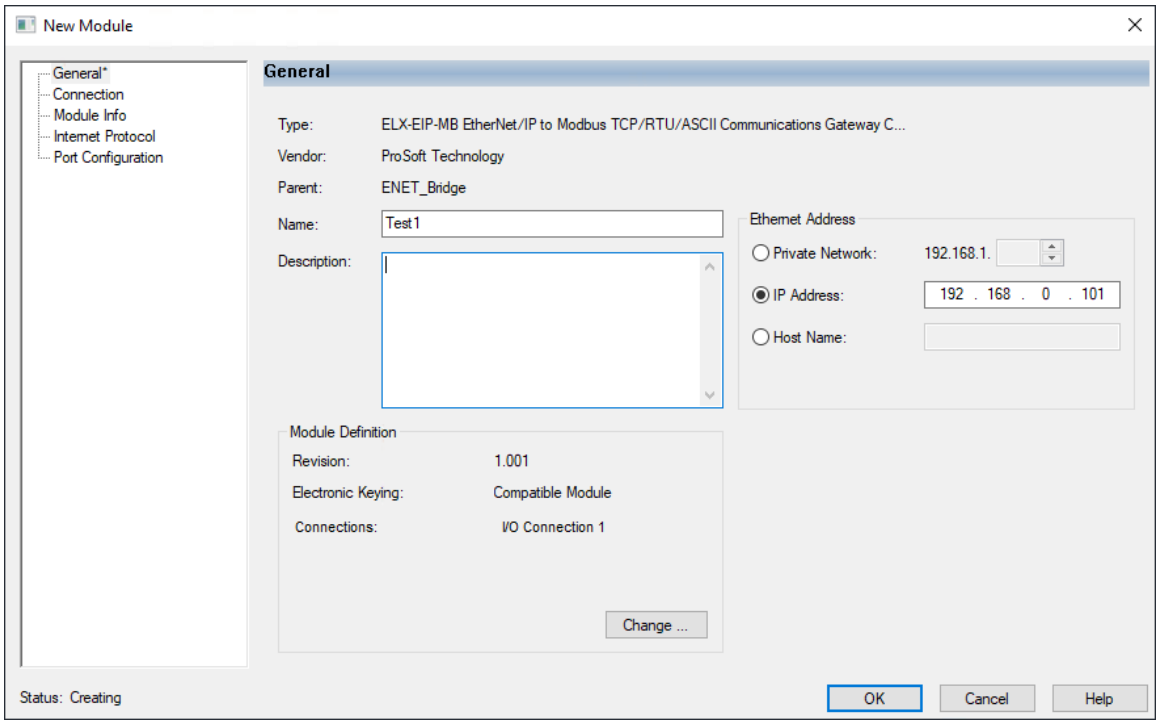
5 In the *Select Module Type* dialog, in search for **ELX-EIP-MB**.

6 Click on the **ELX-EIP-MB**, and then click **CREATE**. This opens the *New Module* dialog.



Note: If the ELX-EIP-MB does not appear in this list, upload EDS file from the container again, or download the EDS file from www.prosoft-technology.com and run the EDS Hardware Installation Tool.

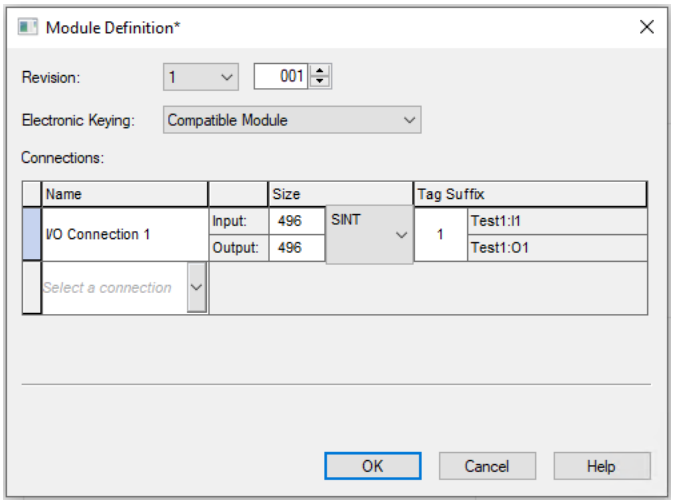
7 In the *New Module* dialog, enter a name for the container, then enter the IP address of the ELX-EIP-MB container. Please see the *ELX3 User Manual* for more information about configuring the ELX-EIP-MB container IP address.



8 To add I/O connections click **CHANGE.**

a) In the *Module Definition* dialog, enter the I/O connections.

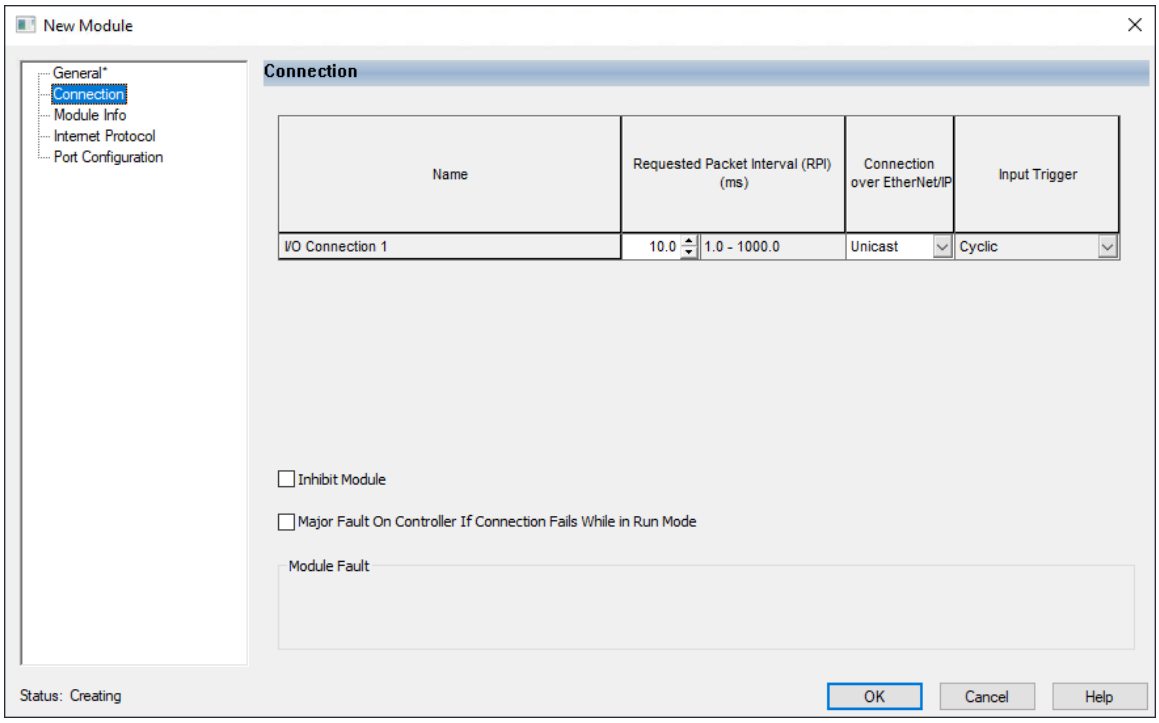
Size I/O: Maximum 496 bytes
Type: SINT/INT/DINT/REAL



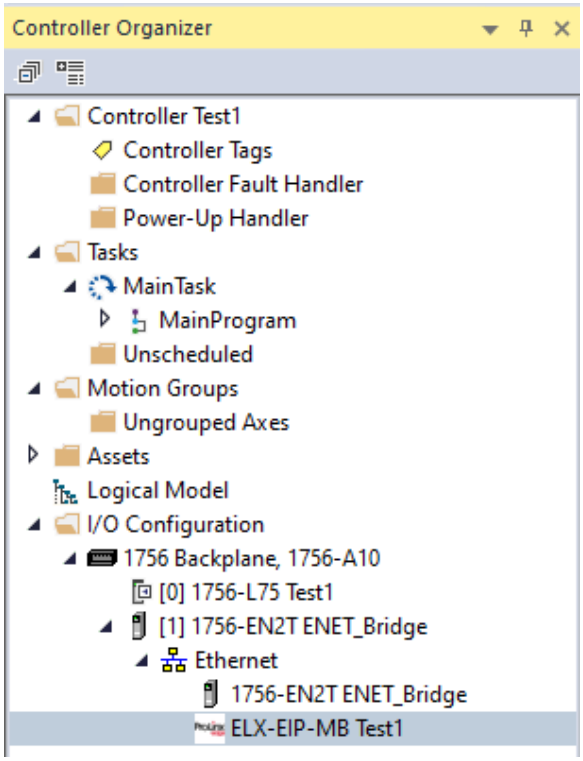
b) Up to eight I/O connections can be added. The I/O connections can have different *Input* and *Output* data sizes, as long as it matches the *Input* and *Output* sizes configured in the *EIP Class 1 Connection* configuration in PCB. For more information, see section *7.3.1 EIP Class 1 Connection*. When finished click **OK**.

9 In the *New Module* dialog, click on the **CONNECTION** tab to configure each I/O connection with its own RPI time. When finished, click **OK**.

Note: The RPI time range for each of the 8 connections is 10ms.



10 The newly-created ELX-EIP-MB container appears in the *Controller Organizer* under the EtherNet/IP bridge.



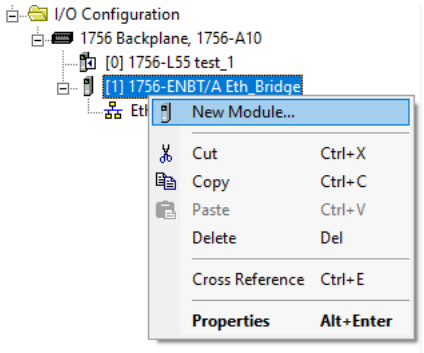
7.6.1.2 Adding the Container to RSLogix 5000 v16 through v19

Note: Class 1 connections are not supported in RSLogix v15 and older.

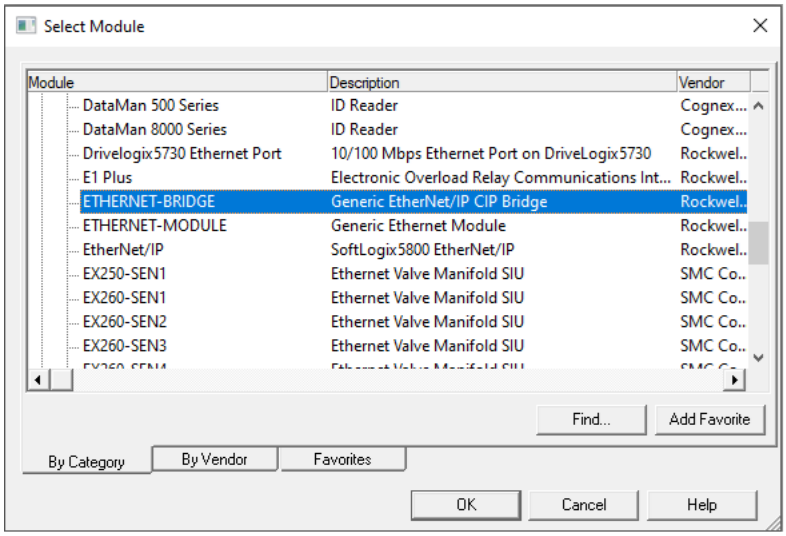
1 Start Rockwell Automation *RSLogix 5000*.

2 Add an EtherNet/IP bridge to the controller.

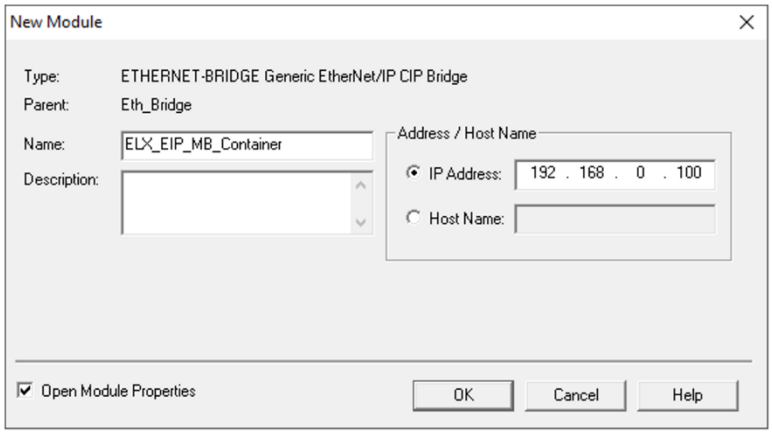
3 In the *Controller Organizer*, right-click the EtherNet/IP bridge in the I/O tree and click on **NEW MODULE**.



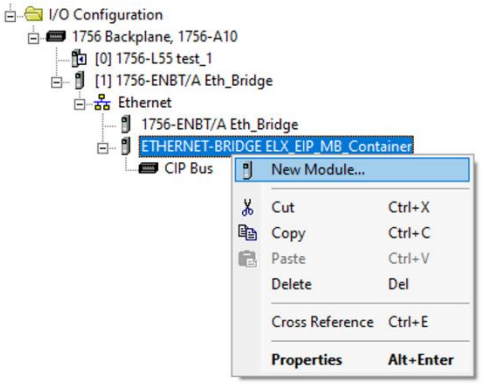
4 In the *Select Module* dialog, under the *Communications* category, select the **ETHERNET-BRIDGE** module and then click **OK**.



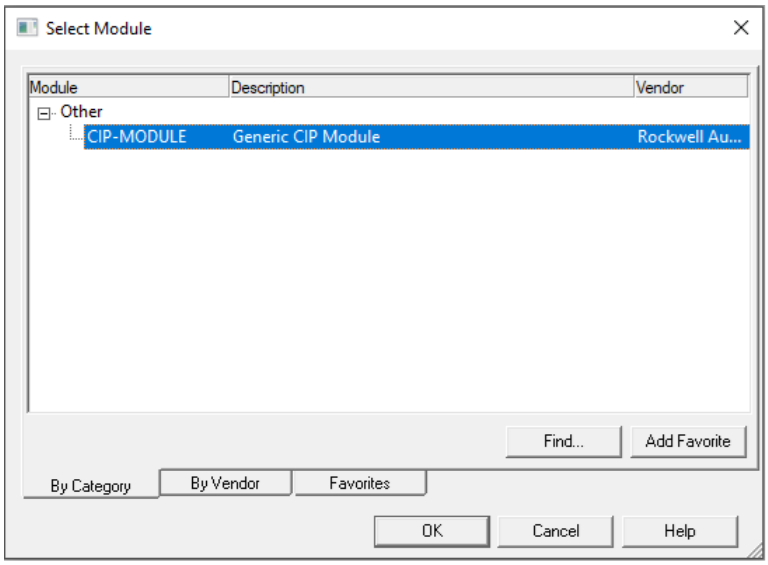
5 In the *New Module* dialog, enter a *Name* and *IP Address* of the ELX-EIP-MB container. This creates the communication path from the processor to the ELX-EIP-MB container. Click **OK** to add the bridge to the rack.



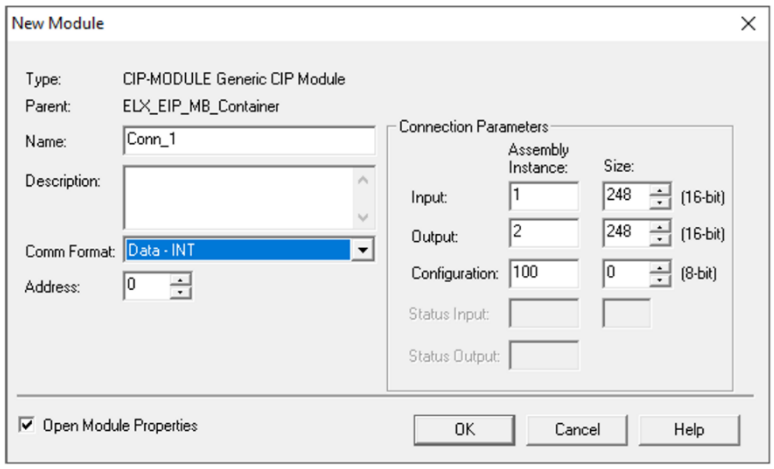
6 In the I/O tree, right-click on the newly-created **ETHERNET-BRIDGE** module and click on **NEW MODULE**.



7 In the *Select Module* dialog, select the **CIP-MODULE** and then click **OK**.



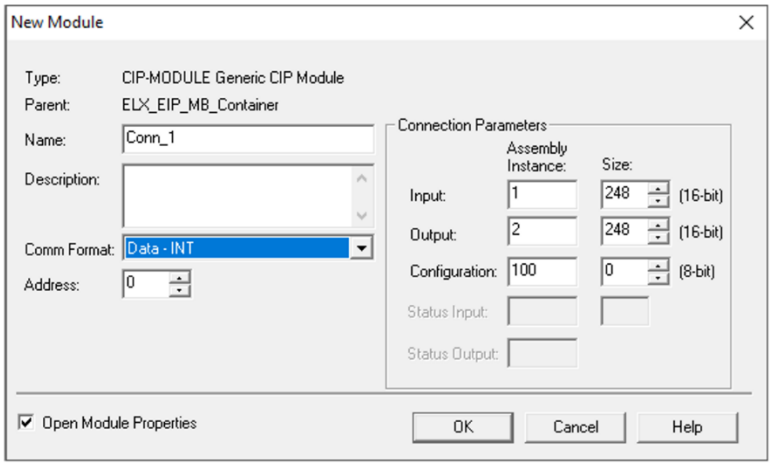
8 In the *New Module* dialog, specify the parameters for the I/O connection. The input and output sizes must match the *Input* and *Output* sizes configured in PCB *EIP Class 1 Connection* configuration. For more information, see section *7.3.1 EIP Class 1 Connection*.



The *Address* parameter represents the connection number in PCB.

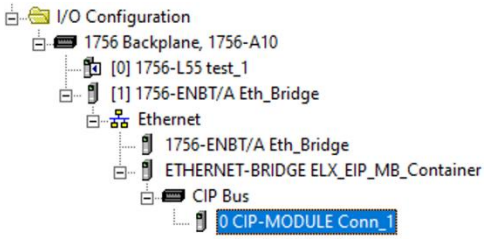
Address	Input	Output	Configuration
1	1	2	100
2	5	6	100
3	9	10	100
4	13	14	100
5	17	18	100
6	21	22	100
7	25	26	100
8	29	30	100

9 Set the *Comm Format* to **DATA TYPE INT**.



Parameter	Value
Input Assembly Instance	1
Input Size	248
Output Assembly Instance	2
Output Size	248
Configuration Assembly Instance	100
Configuration Size	0

10 Click **OK** to add to CIP-Module to the Ethernet Bridge.



11 Repeat the steps above to add and configure additional CIP Connections.

7.6.2 ControlLogix and CompactLogix Processor Specifics

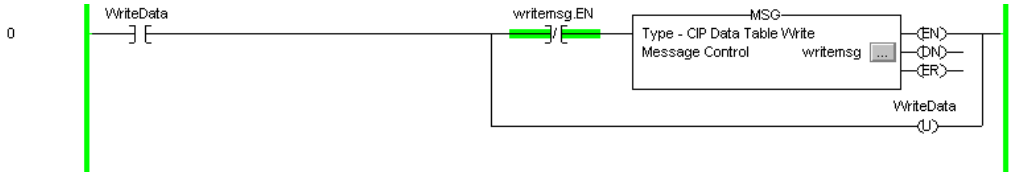
CIP Data Table MSG instructions are used to exchange data between a Control/CompactLogix processor and the ELX-EIP-MB container.

7.6.2.1 CIP Data Table Operations

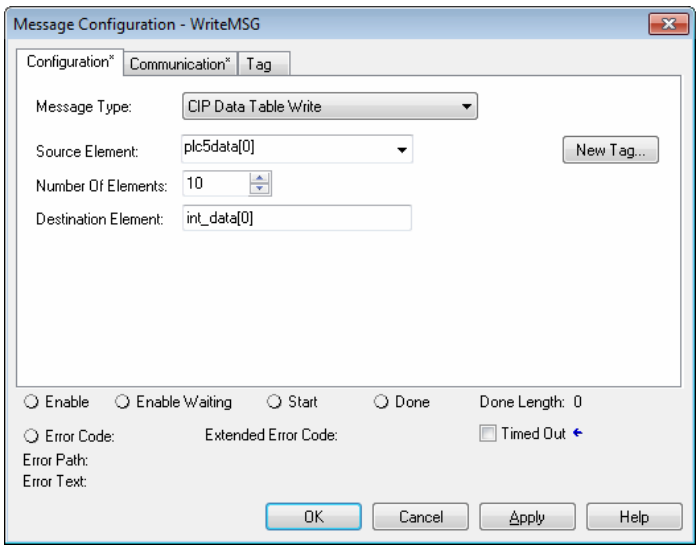
Use CIP messages to transfer data between the ControlLogix or CompactLogix processor and the ELX-EIP-MB container. Tag names define the elements to be transferred. The ELX-EIP-MB container supports both read and write operations.

7.6.2.1.1 CIP Data Table Write

CIP data table write messages transfer data from the processor to the ELX-EIP-MB container. The following diagram shows an example rung that executes a write command.



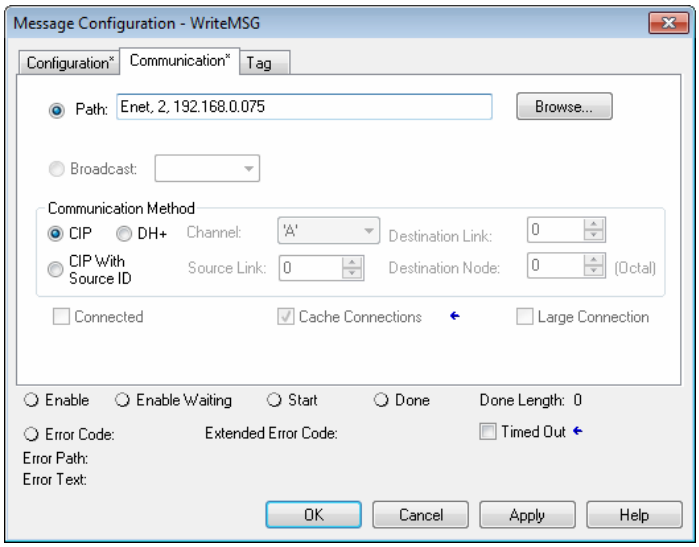
1 In the *Message Configuration* dialog, define the data set to be transferred from the processor to the ELX-EIP-MB container as shown in the following image.



2 Complete the dialog for the data area to be transferred. CIP Data Table messages require a tag database element for both the source and destination.

- The **SOURCE ELEMENT** is a tag defined in the Controller Tag database.
- The **NUMBER OF ELEMENTS** is the number of words written to the ELX-EIP-MB Container virtual/internal database.
- The **DESTINATION ELEMENT** is the tag element in the ELX-EIP-MB container.
- The ELX-EIP-MB container simulates a tag database as an array of elements defined by the maximum register size for the ELX-EIP-MB container with the tag name **INT_DATA** (with the maximum value of *int_data[9999]*).

3 In the previous example, the first element in the database is the starting location for the write operation of ten words. Click the **COMMUNICATION** tab and complete the communication information as shown in the following image.



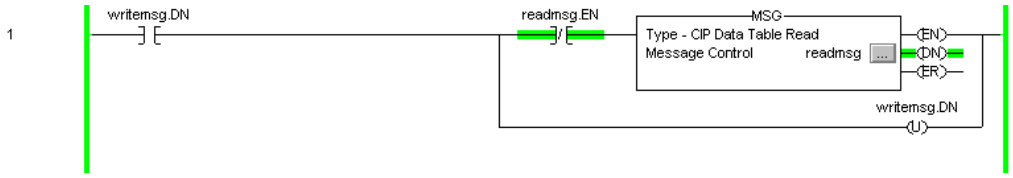
4 Select **CIP** as the **COMMUNICATION METHOD**. The **PATH** specifies the message route from the processor to the container. Path elements are separated by commas. In the example path shown:

- The first element is *Enet*, which is the user-defined name given to the 1756-ENET gateway in the chassis (substitute the slot number of the ENET gateway for the name)
- The second element, *2*, represents the Ethernet port on the 1756-ENET gateway.
- The last element of the path, *192.168.0.75*, is the ELX-EIP-MB container IP address, which is the target for the message.

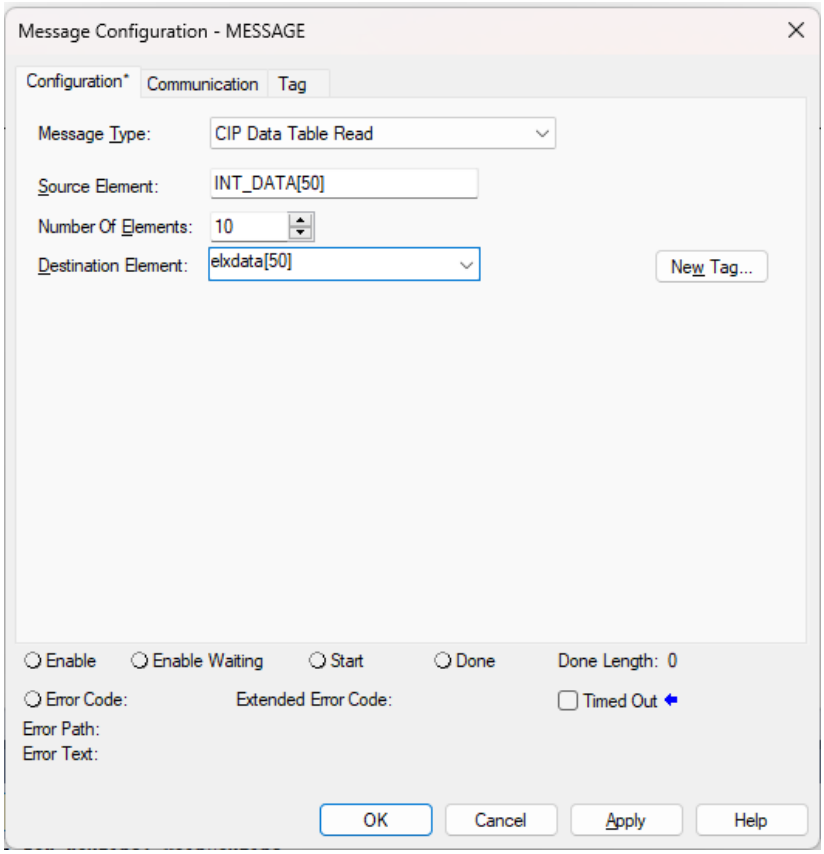
More complex paths are possible if routing to other networks using multiple 1756-ENET gateways and racks. Refer to the ProSoft Technology Technical Support Knowledgebase for more information on Ethernet routing and path definitions.

7.6.2.1.2 CIP Data Table Read

CIP data table read messages transfer data to the processor from the ELX-EIP-MB container. The following diagram shows an example rung that executes a read command.



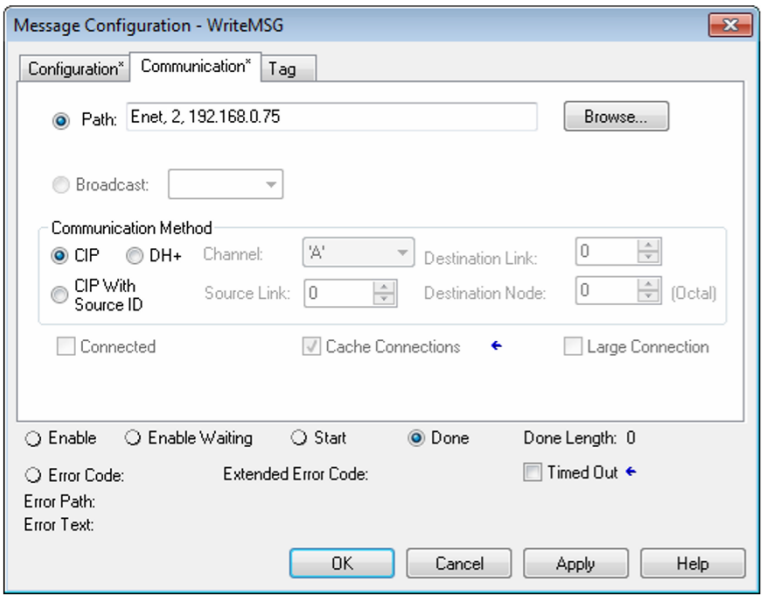
1 In the *Message Configuration* dialog, define the data set to be transferred from the processor to the ELX-EIP-MB container as shown in the following image.



2 Complete the dialog for the data area to be transferred. CIP Data Table messages require a tag database element for both the source and destination.

- The **DESTINATION ELEMENT** is a tag defined in the Controller Tag database.
- The **SOURCE ELEMENT** is the tag element in the ELX-EIP-MB container.
- The **NUMBER OF ELEMENTS** is the number of words written to the ELX-EIP-MB Container virtual/internal database.
- The ELX-EIP-MB container simulates a tag database as an array of elements defined by the maximum register size for the *Maximum Register* parameter in the [*Gateway*] section with the tag name **INT_DATA.**

3 In the previous example, the first element in the database is the starting location for the read operation of 10 elements. Click the **COMMUNICATION** tab and complete the communication information as shown in the following image.



4 Select **CIP** as the *Communication Method*. The *Path* specifies the message route from the processor to the EIP driver. Path elements are separated by commas. In the example path shown:

- The first element is *Enet*, which is the user-defined name given to the 1756-ENET gateway in the chassis (substitute the slot number of the ENET gateway for the name)
- The second element, *2*, represents the Ethernet port on the 1756-ENET gateway.
- The last element of the path, *192.168.0.75*, is the ELX-EIP-MB container IP address, which is the target for the message.

More complex paths are possible if routing to other networks using multiple 1756-ENET gateways and racks. Refer to the ProSoft Technology Technical Support Knowledgebase for more information on Ethernet routing and path definitions.

8 MBTCP Protocol

8.1 MBTCP Functional Overview

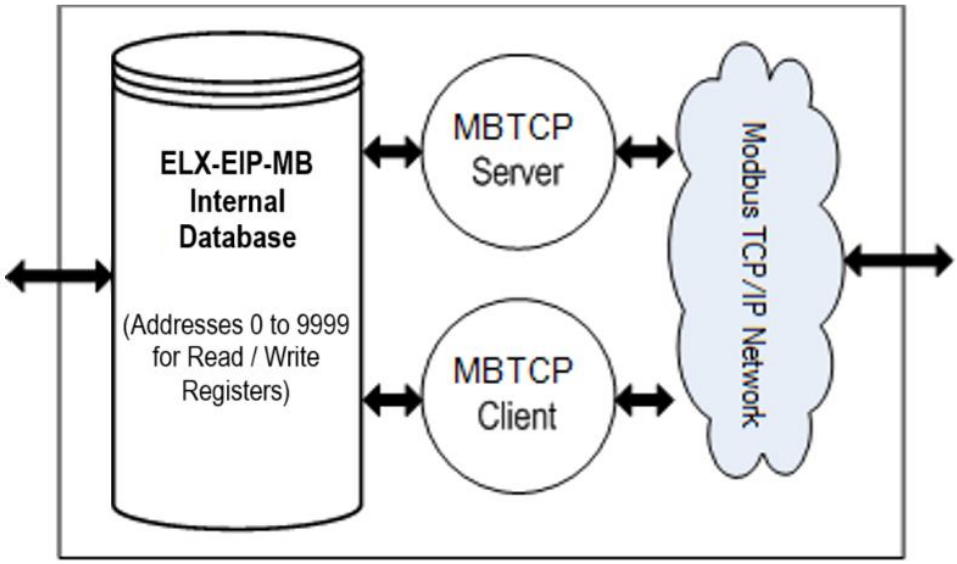
The Modbus TCP driver (MBTCP) of the ELX-EIP-MB container is used to interface with Modbus TCP Client and Server devices.

The ELX-EIP-MB container supports up to 10 Modbus TCP Client connections to interface with processors (and other Server based devices). Each Client can be configured with 16 individual commands for a total of 160 commands for the entire MBTCP driver.

Data stored in the ELX-EIP-MB container's internal database is accessible for read and write operations from the Modbus TCP network. The container supports the MBAP (Service Port 502) or MBTCP (Service Port 2000) TCP/IP protocols.

The MBAP protocol (Port 502) is a standard implementation defined by Schneider Electric and used on their Quantum processor. This open protocol is a modified version of the Modbus serial protocol. The MBTCP protocol is an embedded Modbus protocol message in a TCP/IP packet. The container supports up to five active Server connections on Service Port 502, five additional active Server connections on Service Port 2000, and 10 Client connections.

The following image illustrates the functionality of the MBTCP protocol.



8.1.1 MBTCP Specifications

Specification	Description	
Supported Modbus Function Codes	1: Read Coil Status 2: Read Input Status 3: Read Holding Registers 4: Read Input Registers 5: Force (Write) Single Coil 6: Preset (Write) Single Holding Register 8: Diagnostics, Sub-function 00: Echo (Server Only)	15: Force (Write) Multiple Coils 16: Preset (Write) Multiple Holding Registers 22: Mask Write Holding Register (Server Only) 23: Read/Write Holding Registers (Server Only)
Supported Clients	10	
Supported Servers	MBAP: 5 Encapsulated: 5	
Command List	Up to 160 fully configurable Client commands. (Maximum 16 commands per Client)	
Status Data	Error codes reported individually for each command	
Command List Polling	Each command can be individually enabled or disabled; write-only-on-data-change is supported	

8.1.1.1 MBTCP Client

- Actively reads data from and writes data to Modbus TCP devices using MBAP.
- Up to 10 Client connections (maximum 16 commands per Client) to communicate to multiple Servers.

8.1.1.2 MBTCP Server

- The Server driver accepts incoming connections on Service Port 502 for Clients using Modbus TCP/IP MBAP messages and connections on Service Port 2000 (or other Service Ports) for Clients using Encapsulated Modbus messages.
- Supports multiple independent Server connections of any combination on Service Port 502 (MBAP) and Service Port 2000 (Encapsulated).
- Up to 10 Servers total are supported (5 MBAP and 5 Encapsulated).

8.1.1.3 Supported Modbus TCP Function Codes

The following table lists the Function Codes supported by the ELX-EIP-MB container.

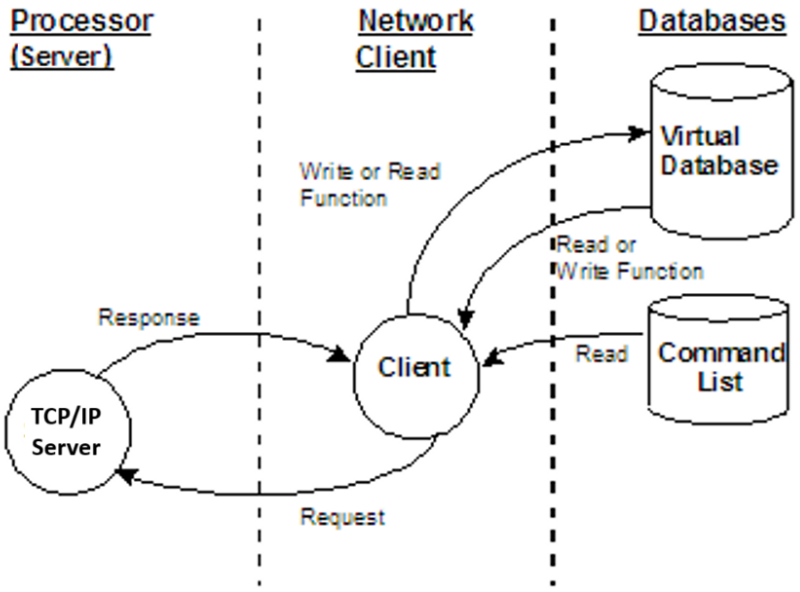
Function Code	Definition	Supported as Client	Supported as Server
1	Read Coil Status 0xxxx	✓	✓
2	Read Input Status 1xxxx	✓	✓
3	Read Holding Registers 4xxxx	✓	✓
4	Read Input Registers 3xxxx	✓	✓
5	Force (Write) Single Coil 0xxxx	✓	✓
6	Preset (Write) Single Holding Register 4xxxx	✓	✓
8	Diagnostics, Sub-function 00: Echo (Server Only)		✓
15	Force (Write) Multiple Coils 0xxxx	✓	✓
16	Preset (Write) Multiple Holding Registers 4xxxx	✓	✓
22	Mask Write Holding Registers		✓
23	Read/Write Holding Registers		✓

8.2 MBTCP Internal Database

8.2.1 MBTCP Client Access to the ELX-EIP-MB Container Database

The MBTCP Client functionality exchanges data between the ELX-EIP-MB container's internal database and remote Modbus TCP Server devices. The MBTCP Client command list defined in ProSoft Configuration Builder specifies what data is to be transferred between the container and each of the Servers on the network.

The following illustration describes the flow of data between the MBTCP Network Client and the internal database.



8.2.2 MBTCP Server Access to the ELX-EIP-MB Container Database

The MBTCP Server functionality supports Modbus TCP Client applications (for example: HMI software, Modbus TCP PLC's, etc.) to read from and write to the ELX-EIP-MB container's internal database.

The MBTCP Server supports Service Port 502 for Modbus TCP MBAP messages, as well as Service Port 2000 to support the TCP/IP Encapsulated Modbus version of the protocol.

The MBTCP Server supports multiple concurrent connections with Modbus TCP Clients. Up to five Clients can simultaneously connect on Service Port 502 and an additional five can simultaneously connect on Service Port 2000.

The ELX-EIP-MB container uses its internal database as the source for Modbus TCP Client read commands and the destination for write commands. The following table specifies the relationship of the ELX-EIP-MB container's internal database to Modbus addressing.

ELX-EIP-MB Container Database Address	Modbus Address
0	40001
1000	41001
2000	42001
3000	43001
…	…
9998	49999

8.2.3 Enron-Daniels Float Data

The following virtual addresses are not part of the ELX-EIP-MB container user database and are not valid addresses for standard data.

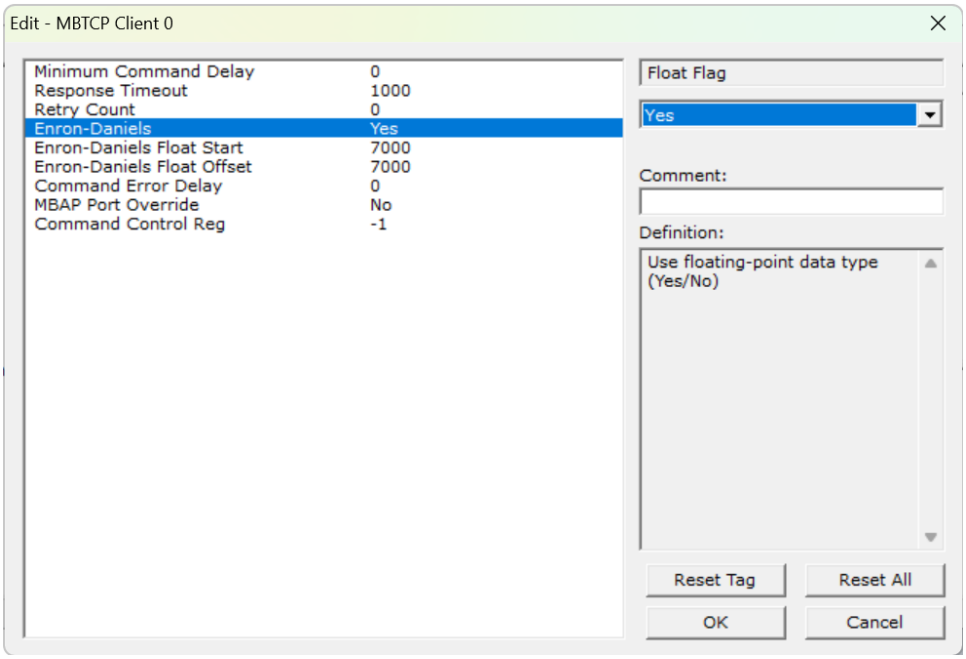
The addresses in this upper range requires configuration of the following *MBTCP Client* or *Server* parameters in *Prosoft Configuration Builder*:

- Set the *Enron-Daniels* parameter in the MBTCP Server or Client configuration to **YES.**
- Set the *Enron-Daniels Float Start* parameter to a database address in the range below.
- Set the *Enron-Daniels Float Offset* parameter to a database address in the ELX-EIP-MB container user database address range shown above.

As an MBTCP Client, Modbus function codes 3, 6, and 16 interpret floating-point values for registers as specified by the *Enron-Daniels Float Start* and *Enron-Daniels Float Offset* parameters. For more information, please see section *8.3.2 MBTCP Client [x]*.

As an MBTCP Server, these addresses may be used for incoming commands (Modbus function codes 3, 6, and 16) that are using floating-point data. For more information, please see section *8.3.1 MBTCP Servers*.

All data stored above the *Enron-Daniels Float Start* address 7000 (default) must be floating-point data. This assumes the *Enron-Daniels Float Offset* address is 7000 (default).



Modbus Address	ELX-EIP-MB Container Database Address	Description
40001	0	First 16-bit Modbus register (First database address)
47000	6999	Last 16-bit Modbus register
47001[a]	7000/7001[b]	First 32-bit floating point register
…	…	…
48001	9000/9001	
…	…	…
48500	9998/9999	Last Internal database register

[a] 7000 is the default *Enron-Daniels Float Start* address.

[b] Floating-point data uses two 16-bit database addresses.

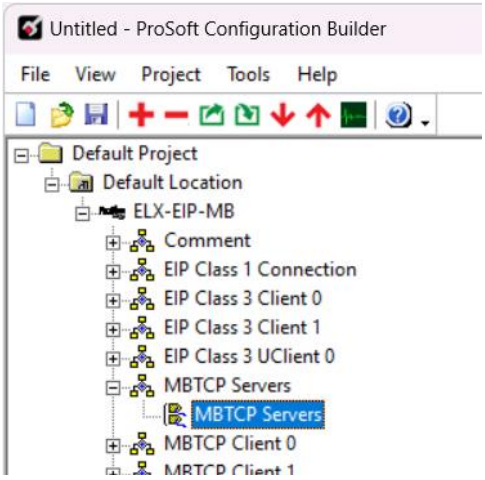
8.3 MBTCP Driver Configuration

8.3.1 MBTCP Servers

This section contains database offset information used by the MBTCP Server when accessed by external Modbus TCP Clients. Use these offsets to segment the database by data type.

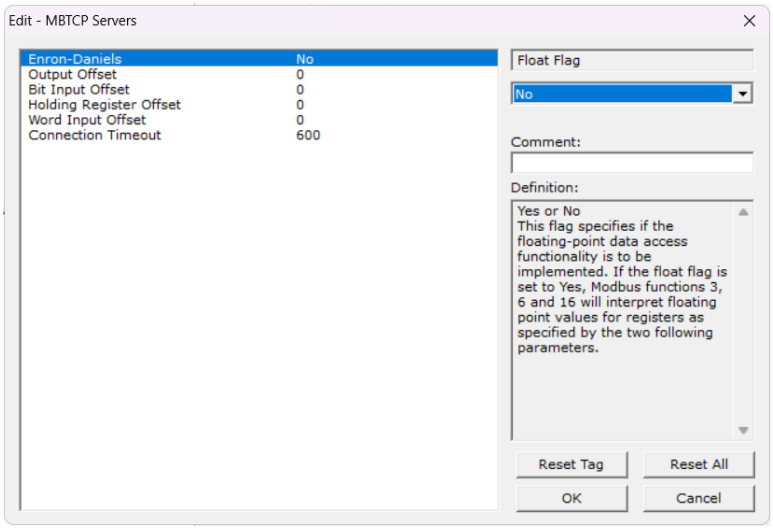
To configure the MBTCP Servers:

1 In *ProSoft Configuration Builder*, click the **[+]** next to the container, then click the **[+]** next to *MBTCP Servers*.



2 Double-click the second *MBTCP Servers* to display the *Edit - MBTCP Servers* dialog.

3 In the dialog, click a parameter and then enter a value for the parameter.



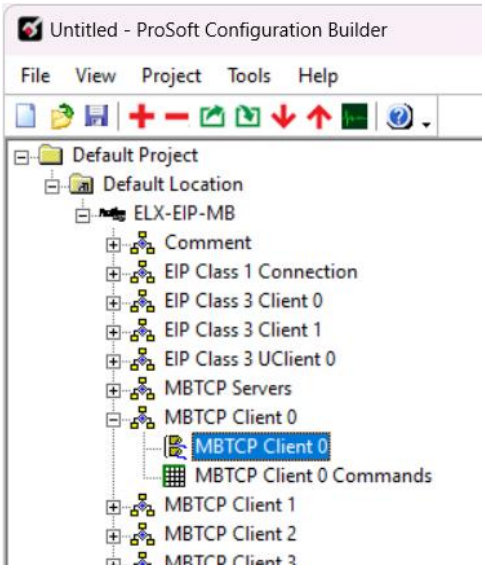
Parameter	Value	Description
Enron-Daniels	Yes or No	Specifies if the floating-point data access functionality is active. **YES**: Modbus functions 3, 6, and 16 interpret floating-point values for registers as specified by *Enron-Daniels Float Start* and *Enron-Daniels Float Offset*. **NO**: The container does not use floating point functionality.
Enron-Daniels Float Start	0 to 65535	This parameter defines the first register of floating-point data. All requests with register values greater-than or equal to this value will be considered floating-point data requests. This parameter is only used if the *Float Flag* is enabled. **Example:** If a value of **7000** is entered, all requests for registers 7000 and above will be considered as floating-point data.
Enron-Daniels Float Offset	0 to 9999	This parameter defines the start register for floating-point data in the internal database. This parameter is used only if the *Float Flag* is enabled. **Example:** If the *Float Offset* value is set to **3000** and the *Float Start* parameter is set to **7000**, data requests for register 7000 will use the internal Modbus register 3000.
Output Offset	0 to 9999	This parameter applies if the port is configured as a Server. Specifies the internal database address to use as the zero address or starting point for binary output Coil data. Coil data is read by Modbus Function Code 1 commands (Read Coils) and written by Function Codes 5 (Force Single Coil) or Function Code 15 (Force Multiple Coils). **Example:** If this parameter is set to **50** and the container receives a Function Code 1 command requesting Coil address 0 (virtual Modbus Coil address 00001 or 000001), the container returns the value at register 50, bit 0 in the container's database.
Bit Input Offset	0 to 9999	Specifies the offset address in the internal Modbus database for network requests for Modbus function 2 commands. **Example:** If this value is set to **150**, an address request of 0 returns the value at register 150 in the database.
Holding Register Offset	0 to 9999	Specifies the offset address in the internal Modbus database for network requests for Modbus functions 3, 6, or 16 commands. **Example:** If this value is set to **50**, an address request of 0 returns the value at register 50 in the database.
Word Input Offset	0 to 9999	Specifies the offset address in the internal Modbus database for network requests for Modbus function 4 commands. **Example:** If this value is set to **150**, an address request of 0 returns the value at register 150 in the database.
Connection Timeout	0 to 1200	Specifies the number of seconds the Server waits to receive new data. If the Server does not receive any new data during this time, it closes the connection.

8.3.2 MBTCP Client [x]

The *MBTCP Client [x]* section of the configuration specifies the parameters for the Client to be emulated on the container. The command list for the Client is entered in a separate section.

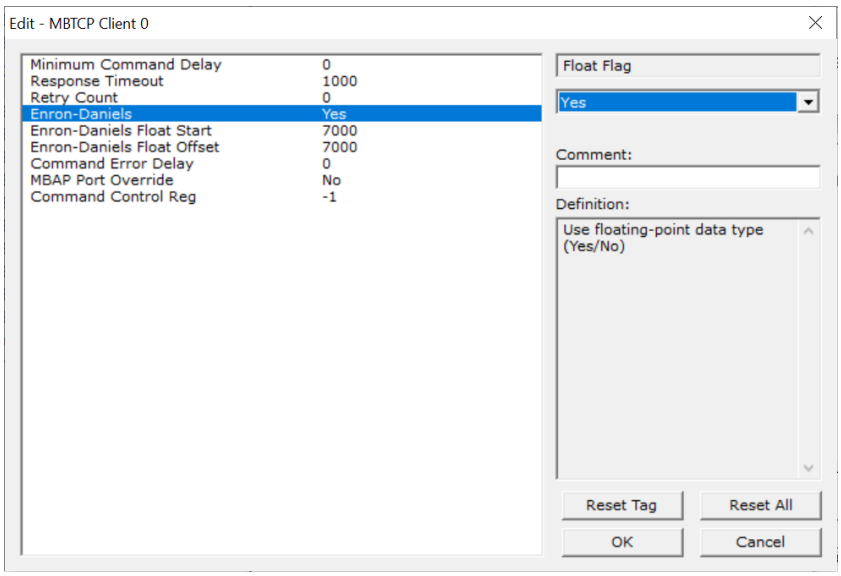
To configure the MBTCP Client [x]:

1 In *ProSoft Configuration Builder*, click the **[+]** next to the container, then click the **[+]** next to *MBTCP Client [x]*.



2 Double-click the second *MBTCP Client [x]* to display the *Edit - MBTCP Client [x]* dialog.

3 In the dialog, click a parameter and then enter a value for the parameter. Note that the *Enron-Daniels Float Start* and *Enron-Daniels Float Offset* parameters only appear if *Enron-Daniels* is set to **YES**.



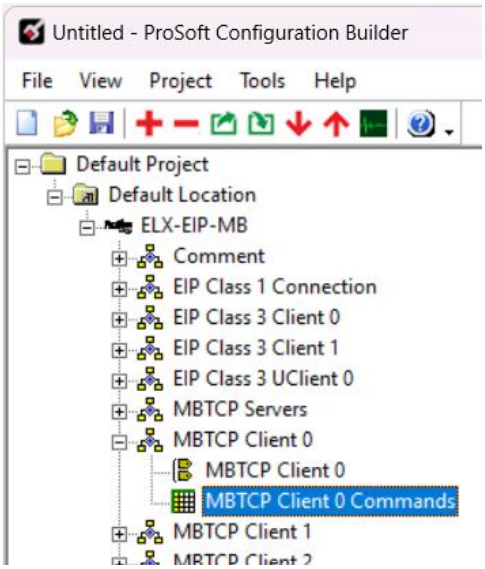
Parameter	Value	Description
Minimum Command Delay	0 to 65535	Specifies the number of milliseconds to wait between the initial issuance of a commands. Use this to delay all commands sent to Servers to avoid "flooding" commands on the network. This parameter does not affect retries of a command as they will be issued when failure is recognized.
Response Timeout	0 to 65535	Specifies the time in milliseconds that a Client waits before re-transmitting a command if no response is received from the addressed Server. The value depends on the type of communication network, and the expected response time of the slowest device on the network.
Retry Count	0 to 10	Specifies the number of times the container retries a command if it fails.
Enron-Daniels (Float Flag)	Yes or No	Specifies if the floating-point data access functionality is active. **YES**: Modbus functions 3, 6, and 16 interpret floating-point values for registers as specified by *Enron-Daniels Float Start* and *Enron-Daniels Float Offset*. **NO**: The container does not use floating point functionality.
Enron-Daniels Float Start	0 to 65535	This parameter only appears if *Enron-Daniels* is set to **Yes**. Specifies the first register of floating-point data. The container considers all requests with register values greater-than or equal to this value as floating-point data requests. For example, if **7000** is entered, the container considers all requests for registers 7000 and above as floating-point data.
Enron-Daniels Float Offset	0 to 9998	This parameter only appears if *Enron-Daniels* is set to **Yes**. Specifies the starting register for floating-point data in the container internal database. For example: If *Enron-Daniels Float Offset* is set to **3000** and set *Enron-Daniels Float Start* to **7000**, the container returns data as floating-point data for register 47001 (or 407001) comes from internal container registers 3000 and 3001. If the requested address is 47002 (407002), the container returns data from internal registers 3002 and 3003. If the requested address is 47101 (407101), the container returns data from internal registers 3200 and 3201; and so on.
Command Error Delay	0 to 300	Specifies the number of 100 millisecond intervals to turn off a command in the error list after an error is recognized for the command. If this is set to **0**, there is no delay.
MBAP Port Override	Yes or No	Specifies whether to override the default port settings. **YES**: The container uses MBAP format messages for all Service Port values. The container does not use RTU through TCP. **NO**: The container uses standard Service Port 502 with MBAP format messages. All other Service Port values use encapsulated Modbus message format (RTU through TCP).
Command Control Reg	0 to 9840, -1 = Disable	This parameter allows the control of command execution in the MBTCP Client Command List. This parameter reserves 16 registers, starting at the value entered. **Note:** This feature allows a command to be enabled, disabled, etc. regardless of how it is configured in the Client command list. A value of **0**, **1**, or **2** can be entered into each command control register: **0**: The command will be disabled. **1**: The command will continuously be executed. **2**: The command will be enabled for conditional writing, which will cause the command to execute only when the value to be written has changed.

8.3.3 MBTCP Client [x] Commands

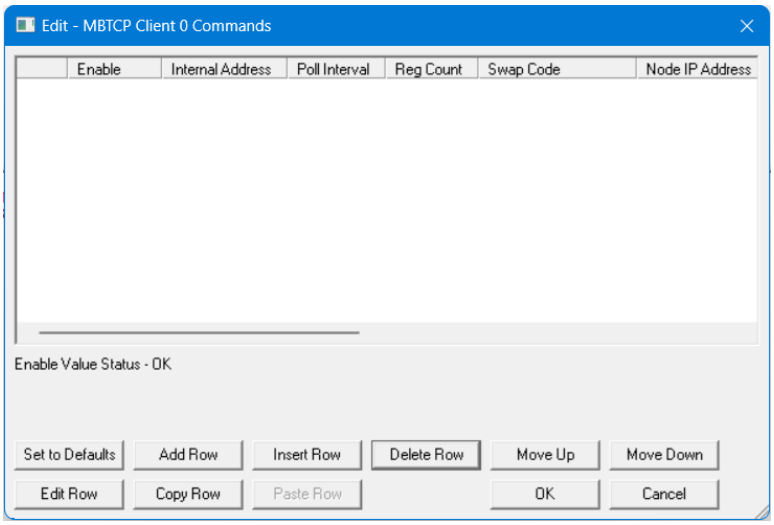
The *MBTCP Client [x] Commands* section defines the MBTCP commands to be issued to Modbus TCP Server devices. Use these commands for data collection and/or control.

To configure the MBTCP Client [x] commands:

1 In *ProSoft Configuration Builder*, click the **[+]** next to the container, then click the **[+]** next to *MBTCP Client [x]*.



2 Double-click *MBTCP Client [x] Commands* to display the *Edit - MBTCP Client [x] Commands* dialog.



3 In the dialog, click **ADD ROW** to add a command, then click **EDIT ROW** to enter values for the command.

The MBTCP Client commands specify the Modbus TCP Server device to be addressed, the function to be performed (read or write), the data area in the Server device to interface, and the registers in the ELX-EIP-MB container internal database to be associated with the device data. The MBTCP Client command list supports up to 16 commands per Client. The command list is processed from top (Command #0) to bottom.

The following table describes the command list configuration parameters:

Parameter	Value	Description
Enable	No Yes Conditional	Specifies if the command is to be executed and under what conditions. **NO (0)**: The command is disabled and is not executed in the normal polling sequence. **YES (1)**: The command is executed upon each scan of the Command List if the *Poll Interval* is set to zero (0). If the *Poll Interval* is set to a non-zero value, the command is executed when the interval timer for that command expires. **CONDITIONAL (2)**: The command is executed only if the internal bit data associated with the command changes. This parameter is valid for write commands (FC 5, 6, 15 and 16).
Internal Address	0 to 9999 (for register-level addressing) or 0 to 159,999 (for bit-level addressing)	Specifies the database address in the container's internal database to use as the destination for data from a read command, or as the source for data sent by a write command. The database address is interpreted as a bit address or a 16-bit register (word) address, depending on the Modbus Function Code used in the command. • For Modbus functions 1, 2, 5, and 15, this parameter is interpreted as a bit-level address. The allowable range is 0 to 159,999. **Note:** This bit address range is available with ProSoft Configuration Builder v4.6 or later. Previous versions have a range of 0 to 65535. • For Modbus functions 3, 4, 6, and 16, this parameter is interpreted as a register-level address.
Poll Interval	0 to 65535	Specifies the minimum interval between executions of continuous commands. The value is in tenths of a second. If a value of 100 is entered, the command is executed no more frequently than once every 10 seconds.
Reg Count	1 to 127 (for registers) or 1 to 2000 (for coils)	Specifies the number of 16-bit registers or binary bits to be transferred by the command. Modbus functions 5 and 6 ignore this field as they apply only to a single data point. • For Modbus functions 1, 2, and 15, this parameter sets the number of bits (inputs or coils) transferred by the command. **Note:** For Modbus functions 1 and 2; 2000 coils are supported. For Modbus function 15; 1968 coils are supported. • For Modbus functions 3, 4, and 16, this parameter sets the number of registers transferred by the command.
Swap Code	No Change Word Swap Word and Byte Swap Byte Swap	Specifies if and how the order of bytes in data received or sent is to be rearranged. Different manufacturers store and transmit multi-byte data in different combinations. Use this parameter when dealing with floating-point or other multi-byte values, as there is no standard method of storing these data types. Set this parameter to rearrange the byte order of data received or sent into an order that is more useful or convenient for other applications. **NO CHANGE (0)**: No change is made in the byte ordering (1234 = 1234). **WORD SWAP (1)**: The words are swapped (1234=3412). **WORD AND BYTE SWAP (2)**: The words are swapped, then the bytes in each word are swapped (1234=4321). **BYTE SWAP (3)**: The bytes in each word are swapped (1234=2143). These swap operations affect 4-byte (2-word) groups of data. Therefore, data swapping using *Swap Codes* should be done only when using an even number of words, such as 32-bit integer or floating-point data.

Parameter	Value	Description
Node IP Address	xxx.xxx.xxx.xxx	IP address of the device being addressed by the command.
Serv Port	502 or other supported port on Server	Service Port on which communication will occur. Use a value of 502 when addressing Modbus TCP/IP Servers that are compatible with the Schneider Electric MBAP specifications (this will be most devices). If the Server device supports another Service Port, enter the Service Port value for this parameter.
Server Address	1 to 255 (0 is a broadcast)	Specifies the node address of a remote Modbus Serial device through a Modbus Ethernet to Serial converter. **Note:** Most Modbus devices only accept addresses in the range of 1 to 247, so check with the Server device manufacturer to see if the Server device can use addresses 248 to 255. If the value is set to zero, the command will be a broadcast message on the network. The Modbus protocol permits broadcast commands for *write* operations. **Do not** use node address 0 for read operations.
Modbus Function	1, 2, 3, 4, 5, 6, 15, or 16	Specifies the Modbus Function Code to be executed by the command. These function codes are defined in the Modbus protocol. More information on the protocol is available from www.modbus.org. The following function codes are supported by the container. **1**: Read Coil Status **2**: Read Input Status **3**: Read Holding Registers **4**: Read Input Registers **5**: Force (Write) Single Coil **6**: Preset (Write) Single Register **15**: Force Multiple Coils **16**: Preset Multiple Registers
MB Address in Device	Varies	Specifies the starting Modbus register or bit address in the Server to be used by the command. Refer to the documentation of each Modbus Server device for the register and bit address assignments valid for that device. The Modbus Function Code determines whether the address is a register-level or bit-level OFFSET address into a given data type range. The offset is the target data address in the Server minus the base address for that data type. Base addresses for the different data types are: 00001 or 000001 (0x0001) for bit-level Coil data (Function Codes 1, 5, and 15). 10001 or 100001 (1x0001) for bit-level Input Status data (Function Code 2) 30001 or 300001 (3x0001) for Input Register data (Function Code 4) 40001 or 400001 (4x0001) for Holding Register data (Function Codes 3, 6, and 16). **Examples:** For bit-level Coil commands (FC 1, 5, or 15) to read or write a Coil 0X address 00001, specify a value of 0 (00001 - 00001 = 0). For Coil address 00115, specify 114 (00115 - 00001 = 114) For register read or write commands (FC 3, 6, or 16) 4X range, for 40001, specify a value of 0 (40001 - 40001 = 0). For 01101, 11101, 31101 or 41101, specify a value of 1100. (01101 - 00001 = 1100) (11101 -10001 = 1100) (31101 - 30001 = 1100) (41101 - 40001 = 1100)

Parameter	Value	Description
		Note: If the documentation for a particular Modbus Server device lists data addresses in hexadecimal (base16) notation, convert the hexadecimal value to a decimal value for this parameter. In such cases, it is not usually necessary to subtract 1 from the converted decimal number, as this addressing scheme typically uses the exact offset address expressed as a hexadecimal number.
Comment		Optional 32-character comment for the command.

8.4 Network Diagnostics

8.4.1 MBTCP PCB Diagnostics

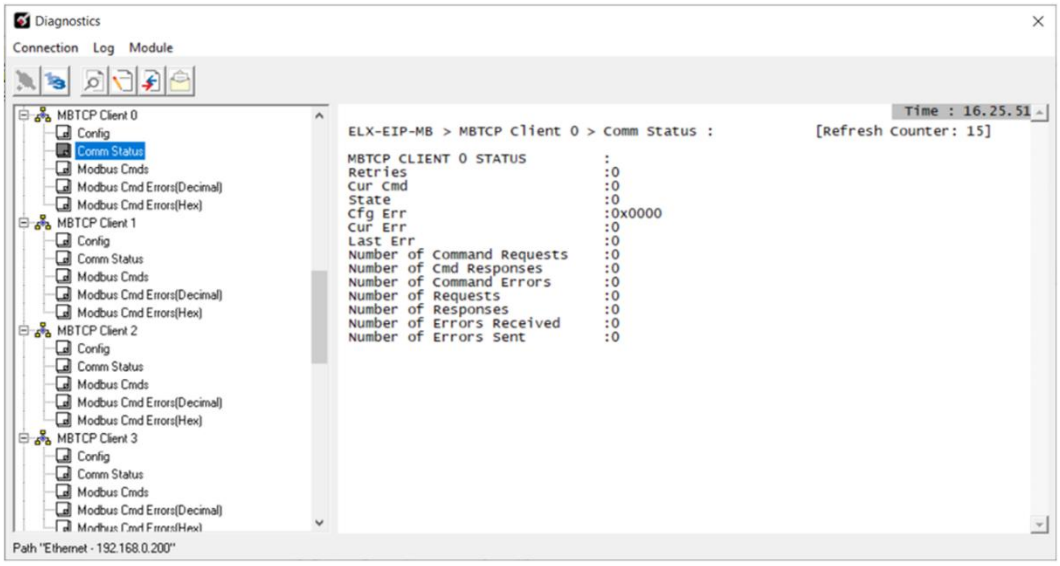
Use ProSoft Configuration Builder to troubleshoot the MBTCP driver communications. For instructions on how to access the diagnostics, see section *5.5 Diagnostics and Troubleshooting*.

The following table summarizes the status information available in ProSoft Configuration Builder for the MBTCP driver:

Connection Type	Submenu Item	Description
MBTCP Server	Config	Configuration settings for Server Connections.
	Comm Status	Status of the Server Connections. Displays a summary of the requests, responses, and errors.
MBTCP Client [x]	Config	Configuration settings for MBTCP Client [x] Connections.
	Comm Status	Status information for MBTCP Client [x] commands. Displays a summary of all the errors resulting from MBTCP Client [x] commands.
	Modbus Commands	Configuration for the MBTCP Client [x] command list.
	Modbus Cmd Errors (Decimal)	Current error codes for each command on the MBTCP Client [x] command list in decimal number format. A zero means there is currently no error for the command.
	Modbus Cmd Errors (Hex)	Current error codes for each command on the MBTCP Client [x] command list in hexadecimal number format. A zero (0) means there is currently no error for the command.

Example:

MBTCP Client Communication Status in PCB:



For specific information on error codes, see section *8.5 MBTCP Error Codes*.

8.4.2 MBTCP Status Data in Upper Memory

The MBTCP driver has an associated status data location in the ELX-EIP-MB container's upper memory. The *Data Map* functionality can be used to map this data into the lower memory range of the ELX-EIP-MB container.

All the status values are initialized to zero (0) at power-up, cold boot, and during warm boot.

8.4.2.1 MBTCP Server Status Data

The following table lists the addresses in upper memory where the ELX-EIP-MB container stores MBTCP Server status data:

Service Port	Address Range
2000	16200 to 16209
502	16210 to 16219

Each of the MBTCP Servers have their own set of status data. The following table describes the MBTCP Server status data contents:

Offset	Description
0	Number of Command Requests
1	Number of Command Responses
2	Number of Command Errors
3	Number of Requests
4	Number of Responses
5	Number of Errors Sent
6	Number of Errors Received
7	Configuration Error Word
8	Current Error Code
9	Last Error Code

8.4.2.2 MBTCP Client Status Data

The following table lists the addresses in upper memory where the ELX-EIP-MB container stores MBTCP Client status data:

Client	Address Range
0	25500 to 25509
1	25526 to 25535
2	25552 to 25561
...	...
8	25708 to 25717
9	25734 to 25743

Each of the MBTCP Clients has their own set of status data. The following table describes the MBTCP Client status data contents:

Offset	Description
0	Command Request Count (total Client commands sent)
1	Command Response Count (total command responses received)
2	Command Error Count
3	Number of Request Packets
4	Number of Response Packets
5	Errors Sent
6	Errors Received
7	Reserved
8	Current Error (most recent). A non-zero value means the current Client command had an error.
9	Last Error (most recent). Stores the most recent non-zero value error code that was reported by the Client the last time it experienced an error. This register holds the last error value until the memory is cleared by a restart, reset, cold-boot, or warm-boot operation. Therefore, any value here may be from an error that occurred at any time since the ELX-EIP-MB container was last restarted.

8.4.2.3 MBTCP Client Command List Error Data

The ELX-EIP-MB container stores a status/error code in upper memory for each command in a MBTCP Client command list. The following table lists the addresses in upper memory where the ELX-EIP-MB container stores the command list error data for each MBTCP Client:

Client	Address Range
0	25510 to 25525
1	25536 to 25551
2	25562 to 25577
...	...
8	25718 to 25733
9	25744 to 25759

The structure of the command list error data is displayed in the following table:

Offset	Description
0	Command #1 Error Code
1	Command #2 Error Code
2	Command #3 Error Code
…	…
13	Command #14 Error Code
14	Command #15 Error Code
15	Command #16 Error Code

A non-zero error code indicates an error. Please see section *8.5 MBTCP Error Codes* for more error code information.

8.5 MBTCP Error Codes

8.5.1 Standard Modbus Exception Code Errors

These error codes can be generated or returned by both Modbus TCP Clients and Servers. These codes are the standard Modbus errors.

Code	Description
1	Illegal Function
2	Illegal Data Address
3	Illegal Data Value
4	Failure in Associated Device
5	Acknowledge
6	Busy, Rejected Message

8.5.2 MBTCP Client Specific Errors

These error codes are specific to the MBTCP Client.

Code	Description
-33	Failed to connect to Server specified in command
-35	Wrong message length in the response
-36	MBTCP command response timeout (same as -11)
-37	TCP/IP connection ended before session finished

8.5.3 MBTCP Communication Error Codes

The ELX-EIP-MB container detects these error codes and are stored in the *Client Command List Error* memory area.

Code	Description
-2	Timeout while transmitting message
-11	Timeout waiting for response after request (same as -36)
253	Incorrect Server address in response
254	Incorrect function code in response
255	Invalid CRC/LRC value in response

8.5.4 MBTCP Command List Error Codes

The ELX-EIP-MB container detects these command-specific error codes during initial command list loading at ELX-EIP-MB container power-up or reset and are stored in the *Client Command List Error* memory region.

Code	Description
-40	Too few parameters
-41	Invalid enable code
-42	Internal address > maximum address
-43	Invalid node address (<0 or >255)
-44	Count parameter set to 0
-45	Invalid function code
-46	Invalid swap code

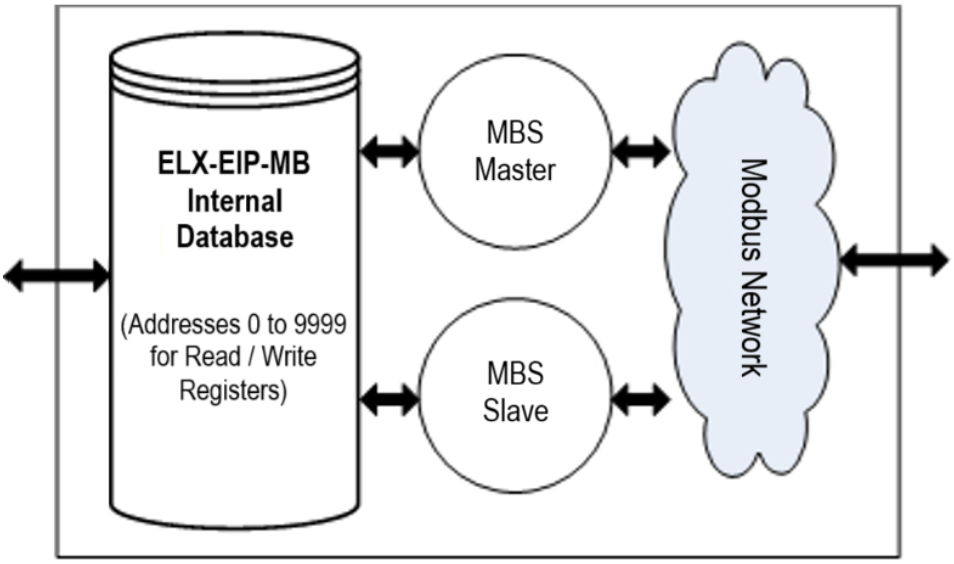
9 MBS Protocol

9.1 MBS Functional Overview

The Modbus RTU/ASCII driver (MBS) of the ELX-EIP-MB Container supports the Modbus RTU/ASCII protocol for both Master and Slave connections.

- As a Master, you can specify a command list of up to 100 entries. The Container stores the read and write command data in the Container's internal memory. See section *9.2 MBS Internal Database* for more information.
- As a Slave, data in the lower memory of the Container's internal database is accessible for read and write operations by a remote Modbus Master.

The following illustration shows the functionality of the Modbus Serial protocol:



9.1.1 Modbus Master/Slave Port Specifications

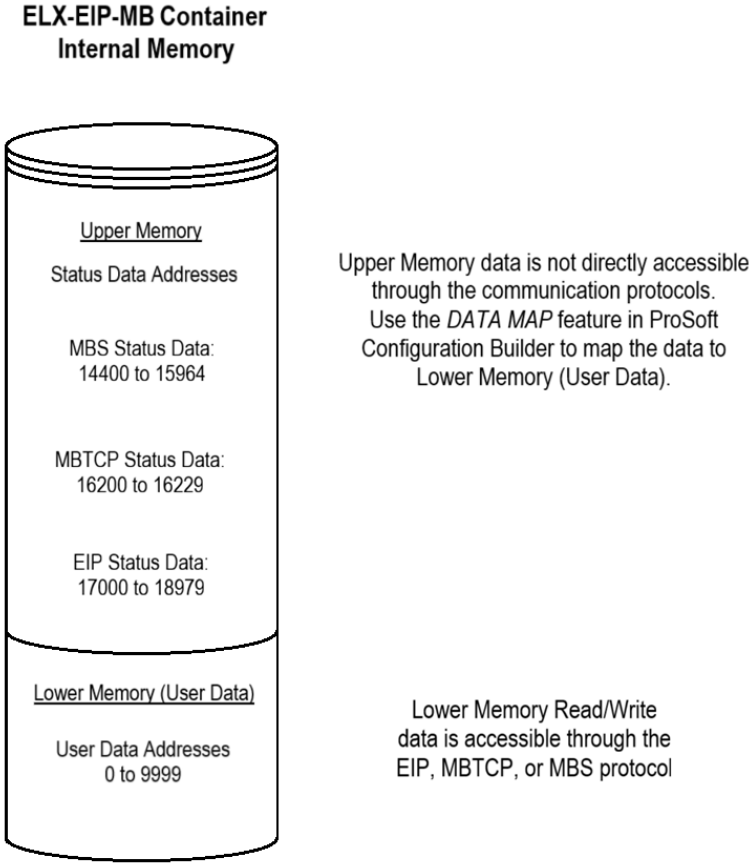
Type	Specifications
General Parameters	
Internal Database	Up to 10000 registers (16-bit) available.
Communication parameters	Port 1: Baud Rate: 110 to 115K baud Stop Bits: 1 or 2 Data Size: 7 to 8 bits Parity: None, Even, Odd RTS Timing delays: 0 to 65535 milliseconds
Modbus Modes	RTU mode (binary) with CRC-16 ASCII mode with LRC error checking
Floating Point Data	Floating point data movement supported, including configurable support for Enron-Daniels implementation
Modbus Function Codes	1: Read Coil Status 2: Read Input Status 3: Read Holding Registers 4: Read Input Registers 5: Force (Write) Single Coil 6: Preset (Write) Single Register 8: Diagnostics, Sub-function 00: Echo (Slave Only) 15: Force (Write) Multiple Coils 16: Force (Write) Multiple Register
RS Interface	RS-232 and RS-485
Modbus Master	
Command List	Up to 100 command per Master port, each fully configurable for function, Slave address, register to/from addressing and word/bit count
Status Data	Error codes available on an individual command basis. In addition, a Slave status list is maintained per active Modbus Master port.
Polling of command list	Configurable polling of command list, including continuous and on change of data
Modbus Slave	
Node address	1 to 247 (software selectable)
Status Data	Error codes, counters and port status available per configured Slave port starting at register 14400.

9.2 MBS Internal Database

The internal database is central to the functionality of the ELX-EIP-MB Container. The Container shares this database between all the communications ports on the Container and uses it as a conduit to pass information from one protocol to another device on one network, to one or more devices on another network. This permits data from devices on one communication port to be accessed and controlled by devices on another communication port.

Status and error information generated by the Container can be mapped into the User Data area of the internal database. The internal database is divided into two areas:

- Upper memory for the Container status data area. This is where the Container stores the MBS status data at addresses 14400 to 15964.
- Lower memory for the User Data area at addresses 0 to 9999. This is where incoming data from external devices is stored and accessed.



Either protocol in the ELX-EIP-MB Container can write data to and read data from the User Data area.

- If the Container is acting as a Client/Master, you create commands to read data from external Client/Server devices and store the data in a specific location in the User Data area.
- If the Container is acting as a Server/Slave, the external Client/Master devices write data to a specific location in the User Data area.

Note: The Modbus RTU status data is stored in addresses 14400 to 15964 in upper memory. To access the Modbus RTU status data in the upper memory, use the data mapping feature to copy the status data to the User Data area (0 to 9999). For more information, see chapter *10 Mapping Data in Container Memory*.

Otherwise, you can use the diagnostic functions in ProSoft Configuration Builder to view Container status data. For more information on the Container status data, see section *9.4.1 MBS PCB Diagnostics*.

9.2.1 Modbus Port Access to Database

When the Container is configured as a Slave, its internal database is the source for read requests and the destination for write requests from remote Masters. Access to the database is controlled by the command type in the incoming message from the remote Master. The following table defines the relationship of the Container's internal database to the addresses required in the incoming Modbus requests.

Database Address	Modbus Address
0	40001 (five-digit addressing) or 400001 (six-digit addressing)
1000	41001 or 401001
2000	42001 or 402001
3000	43001 or 403001
3999	44000 or 404000
9998	49999 or 409999

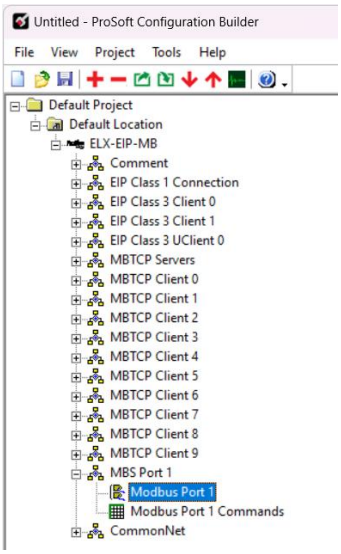
9.3 MBS Driver Configuration

9.3.1 Configuring MBS Port 1

In the PCB the *MBS Port 1* section configures the Modbus Master and Slave port communication settings.

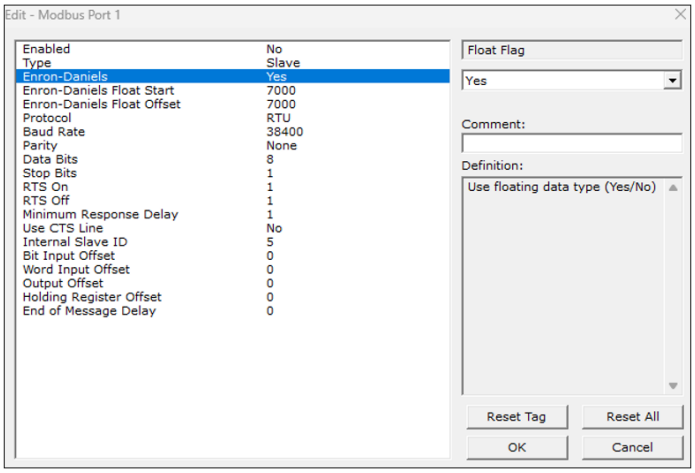
To configure the MBS Port 1 in PCB

1 In *ProSoft Configuration Builder*, click the **[+]** next to the Container, then click the **[+]** next to *MBS Port 1*.



2 Double-click *Modbus Port 1* to display the *Edit - Modbus Port 1* dialog box.

3 In the dialog box, click a parameter and then enter a value for the parameter. Note that the *Enron-Daniels Float Start* and *Enron-Daniels Float Offset* parameters only appear if you set *Enron-Daniels* to **YES**.



9.3.1.1 Configuration Parameters Common to Master and Slave

Parameter	Value	Description
Enabled	Yes or No	Specifies if the port will be used. Note: If the serial port is not attached to the container, then the MBS port will be automatically disabled.
Type	Master or Slave	Specifies if the port emulates a Master or Slave device. **MASTER** - The Container initiates Modbus commands to one or more Modbus devices **SLAVE** - The Container responds to Modbus commands initiated by a Modbus Master
Enron-Daniels (Float Flag)	Yes or No	Specifies if the Container implements floating-point data access. Modbus functions 3, 6, and 16 interpret floating-point values for registers as specified by the two following parameters (*Enron-Daniels Float Start* and *Enron-Daniels Float Offset*).
Enron-Daniels Float Start	0 to 65535	This parameter only appears if *Enron-Daniels* is set to Yes. Specifies the first register of floating-point data. All requests with register values greater than or equal to this value are considered floating-point data requests. For example, if you enter a value of 7000, the Container considers all requests for registers 7000 and above as floating-point data.
Enron-Daniels Float Offset	0 to 9998	This parameter only appears if *Enron-Daniels* is set to Yes. Specifies the start register for floating-point data in the internal database. For example, if you enter a value of 3000 and set the *Enron-Daniels Float Start* parameter to 7000, data requests for register 7000 use the internal Modbus register 3000.
Protocol	RTU or ASCII	Specifies the Modbus protocol version for the port.
Baud Rate	Various	Specifies the baud rate for the port.
Parity	None Odd Even	Specifies the type of parity checking to use. Parity is a simple error checking algorithm used in serial communication. All devices communicating through this port must use the same parity setting.
Data Bits	7 or 8	Specifies the number of data bits for each word for the protocol. All devices on this port must use the same number of data bits.
Stop Bits	1 or 2	Specifies the number of stop bits that signal the end of a character in the data stream. For most applications, use one stop bit. For slower devices that require more time to re-synchronize, use two stop bits. All devices on this port must use the same number of stop bits.
RTS On	0 to 65535	Specifies the number of milliseconds to delay data transmission after *Ready To Send* (RTS) is asserted.
RTS Off	0 to 65535	Specifies the number of milliseconds to delay after the last byte of data is sent before the RTS modem signal is set low.
Use CTS Line	Yes or No	Specifies if the *Clear To Send* (CTS) modem control line is used or not. **NO** - The Container does not monitor the CTS line. **YES** - The Container monitors the CTS line, and it must be high before the Container sends data. Normally, this parameter is required when half-duplex modems are used for communication (2-wire). This procedure is commonly referred to as *hardware handshaking*.

9.3.1.2 MBS Master Configuration Parameters

Parameter	Value	Description
Response Timeout	0 to 65535	Specifies the command response timeout period in 1 millisecond increments. This is the time that a port configured as a Master waits for a response from the addressed Slave before re-transmitting the command or skipping to the next command in the Command List (see *Retry Count*). The value to specify depends on the communication network and the expected response time (plus or minus) of the slowest device on the network.
Retry Count	0 to 10	Specifies the number of times the Container retries a command if it fails.
End of Message Delay	0 to 65535	Specifies a time delay in milliseconds added to the 3.5-character time delay used by the Container to recognize the end of a message. Certain applications may require validation of Modbus messages with more than 3.5-character time between consecutive bytes (for example, modem applications). A value of 0 causes the Container to use the default end of message delay. **Note:** This parameter only applies to the Modbus RTU protocol.
Minimum Command Delay	0 to 32767	Specifies the number of milliseconds the Container waits between receiving the end of a Slave's response to the most recently transmitted command and the issuance of the next command. Use this parameter to place a delay after each command to avoid sending commands on the network faster than the Slaves can receive them. This parameter does not affect retries of a command, as the Container issues retries when a command failure is recognized.
Error Delay Counter	0 to 60000	Specifies the number of poll attempts to be skipped before trying to re-establish communications with a Slave that has failed to respond to a command within the time limit set by *Response Timeout*. After the Slave fails to respond, the Master skips sending commands that would have been sent to the Slave until the number of skipped commands matches the value entered in this parameter. This creates a sort of *slow poll mode* for Slaves that are experiencing communication problems.
Command Control Reg	-1, 0 to 9899	This parameter allows the execution of commands in the Command List to be controlled by a 100-register area of the Container database, beginning at the address entered in this parameter. This Command Control feature can be disabled by setting this parameter to a value of 0 to -1. Each register in the 100-register area corresponds to the commands configured in the MBS Master command list. Make sure to configure the MBS commands as disabled (enable code = 0). The values of executing via Command Control are as follows: **0** - No commands are executed. **1** - The Container executes the Command continuously. **2** - The command is enabled for conditional writing, which causes the Container to execute the command whenever the values in the database registers associated with the command has changed. **3** - Used only for bit-level write commands, FC 5 and 15. The command is executed only if the internal bit data associated with the command has changed. It also clears the bit or bits in the internal database after the write command is built. For information on the Command List, see section *9.3.2 MBS Port 1 Commands*.

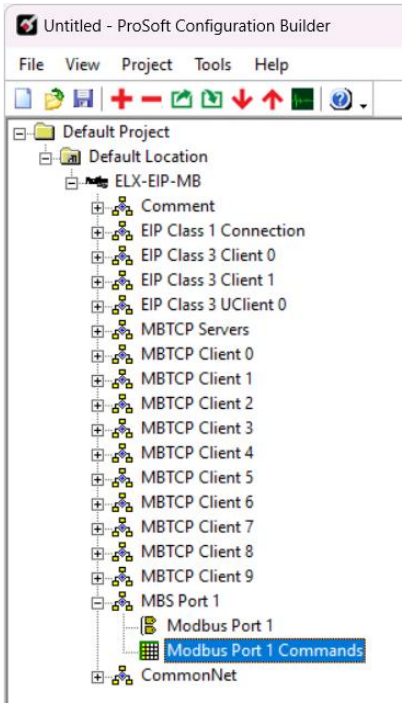
9.3.1.3 MBS Slave Configuration Parameters

Parameter	Value	Description
Minimum Response Delay	0 to 65535	Specifies the number of milliseconds the Container waits before responding to a command received on the port from a remote Master. This delay may be required to accommodate slower Master devices.
Internal Slave ID	0 to 247	Specifies the virtual Modbus Slave address for the port. The Container processes the commands received on the Slave port, addressed to this address. Each Slave device on the network must have a unique address.
Bit Input Offset	0 to 9998	Specifies the internal database address for the zero address or starting point for binary Input Status data. Input Status data is read-only data, requested by Modbus Function Code 2 commands (Read Input Status). For example, if you set this parameter to 150 and a Function Code 2 command is requesting Input Status address 0 (virtual Modbus Address 10001 or 100001), the Container returns the bit value at register 150, bit 0 in the Container's database.
Word Input Offset	0 to 9998	Specifies the internal database address for the zero address or starting point for Input Register (16-bit integer) data. Input Register data is read-only data, requested by Modbus Function Code 4 commands (Read Input Registers). For example, if you set this parameter to 500 and a Function Code 4 command is received requesting Input Register address 0 (virtual Modbus address 30001 or 300001), the Container returns the value at register 500 in the Container's database.
Output Offset	0 to 9998	Specifies the internal database address for the zero address or starting point for binary output Coil data. Coil data is read by Modbus Function Code 1 commands (Read Coils) and written by Function Codes 5 (Force Single Coil) or Function Code 15 (Force Multiple Coils). For example, if you set this parameter to 50 and a Function Code 1 command is received requesting Coil address 0 (virtual Modbus Coil address 00001 or 000001), the Container returns the value at register 50, bit 0 in the Container's database.
Holding Register Offset	0 to 9998	Specifies the internal database address to for the zero address or starting point for Holding Register (16-bit integer) data. Holding Register data is read by Modbus Function Code 3 commands (Read Holding Registers) and written by Function Code 6 (Preset Single Register) or Function Code 16 (Preset Multiple Registers). For example, if you set this parameter to 1000 and a Function Code 3 command is received requesting Holding Register address 0 (virtual Modbus address 40001 or 400001), the Container returns the value at register 1000 in the Container's database.
End of Message Delay	0 to 65535	Specifies a time delay in milliseconds to be added to the 3.5-character time delay that the Container uses to recognize the end of a message. Certain applications may require validation of Modbus messages with more than 3.5-character time between consecutive bytes (for example, modem applications). A value of 0 causes the Container to use the default end of message delay. **Note:** This parameter only applies to the Modbus RTU protocol.

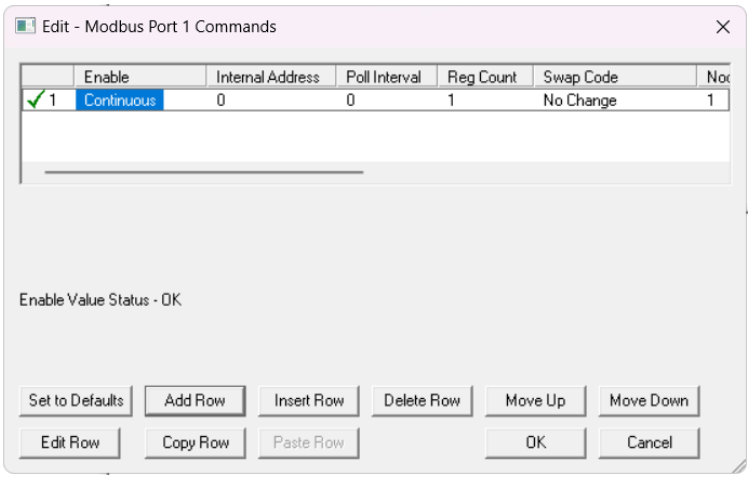
9.3.2 MBS Port 1 Commands

Use the Modbus Port 1 Commands to define a Master serial port *Command List*. This list contains the parameters needed to poll Slave devices attached to a Master port.

1. In *ProSoft Configuration Builder*, click the **[+]** next to the Container, then click the **[+]** next to *MBS Port 1*.
2. Double-click the *Modbus Port 1 Commands* to display the *Edit - Modbus Port 1 Commands* dialog box.



3. In the dialog box, click **ADD ROW** to add a command, then click **EDIT ROW** to enter values for the command.



Parameter	Value	Description
Enable	Disabled Continuous Event Command Conditional	Specifies if the command is to be executed and under what conditions. **DISABLED (0)** - The command is disabled and is not executed in the normal polling sequence. However, the command can still be activated using Command Control. **CONTINUOUS (1)** - The command is executed upon each scan of the Command List if you set *Poll Interval* to zero (0). If you set *Poll Interval* to a non-zero value, the Container executes the command when the interval timer for that command expires. **EVENT COMMAND (2)** - The Container executes the command only if the internal data associated with the command has changed. This parameter is valid only for write commands (FC 5, 6, 15, and 16). **CONDITIONAL (3)** - The Container executes the command only if the internal bit data associated with the command has changed. It also clears the bit or bits in the internal database after the write command is built. This parameter is valid only for bit-level write commands (FC 5 and 15).
Internal Address	0 to 9999 (for register-level addressing) or 0 to 159,999 (for bit-level addressing)	Specifies the database address in the Container's internal database to store the data from a read command, or as the source for data sent by a write command. The database address is interpreted as a bit address or a 16-bit register (word) address, depending on the Modbus Function Code used in the command. For Modbus functions 1, 2, 5, and 15, this parameter is interpreted as a bit-level address. The allowable range is 0 to 159,999. **Note:** This bit address range is available with ProSoft Configuration Builder v4.6 or later. Previous versions have a range of 0 to 65535. For Modbus functions 3, 4, 6, and 16, this parameter is interpreted as a register-level address.
Poll Interval	0 to 65535	Specifies the minimum interval between executions of continuous commands, in seconds. If you enter a value of 10, the command executes no more frequently than once every 10 seconds.
Reg Count	1 to 125 (for registers) or 1 to 800 (for coils)	Specifies the number of 16-bit registers or binary bits to be transferred by the command. Modbus functions 5 and 6 ignore this field as they apply only to a single data point. For Modbus functions 1, 2, and 15, this parameter sets the number of bits (inputs or coils) to be transferred by the command. For Modbus functions 3, 4, and 16, this parameter sets the number of registers to be transferred by the command.
Swap Code	No Change Word Swap Word and Byte Swap Byte Swap	Specifies if and how the order of bytes is arranged when reading or writing data. This option allows for the fact that different manufacturers store and transmit multi-byte data in different combinations. This parameter is helpful when dealing with floating-point or other multi-byte values, as there is no one standard method of storing these data types. You can set *Swap Code* to rearrange the byte order of data received or sent into an order more useful or convenient for other applications. **NO CHANGE (0)** - No change is made in the byte ordering (1234 = 1234) **WORD SWAP (1)** -The words are swapped (1234=3412) **WORD AND BYTE SWAP (2)** - The words are swapped, then the bytes in each word are swapped (1234=4321) **BYTE SWAP (3)** - The bytes in each word are swapped (1234=2143) These swap operations affect 4-byte (or 2-word) groups of data. Therefore, you should use data swapping only when using an even number of words, such as when 32-bit integer or floating-point data.
Node Address	1 to 255 (0 is a broadcast)	Specifies the node address of the Modbus Slave device.

Parameter	Value	Description
		Note: Most Modbus devices only accept addresses in the range of 1 to 247, so check with the Slave device manufacturer to see if a particular Slave can use addresses 248 to 255. If you set the value to zero (0), the Container broadcasts the command on the network. The Modbus protocol permits broadcast commands for *write* operations. **Do not** use node address 0 for read commands.
Modbus Function	1, 2, 3, 4, 5, 6, 15, or 16	Specifies the Modbus Function Code executed by the command. These function codes are defined in the Modbus protocol. The following function codes are supported by the Container. **1** - Read Coil Status **2** - Read Input Status **3** - Read Holding Registers **4** - Read Input Registers **5** - Force (Write) Single Coil **6** - Preset (Write) Single Register **15** - Force Multiple Coils **16** - Preset Multiple Registers
MB Address in Device	Varies	Specifies the starting Modbus register or bit address in the Server for the command. Refer to the documentation of each Modbus Server device for the register and bit address assignments valid for that device. The *Modbus Function* code determines whether the address is a register-level or bit-level offset address into a given data type range. The offset is the target data address in the Server minus the base address for that data type. Base addresses for the different data types are: 00001 or 000001 (0x0001) for bit-level Coil data (Function Codes 1, 5, and 15). 10001 or 100001 (1x0001) for bit-level Input Status data (Function Code 2) 30001 or 300001 (3x0001) for Input Register data (Function Code 4) 40001 or 400001 (4x0001) for Holding Register data (Function Codes 3, 6, and 16). Address calculation examples: For bit-level Coil commands (FC 1, 5, or 15) to read or write a Coil 0X address 00001, specify a value of 0 (00001 - 00001 = 0). For Coil address 00115, specify 114 (00115 - 00001 = 114) For register read or write commands (FC 3, 6, or 16) 4X range, for 40001, specify a value of 0 (40001 - 40001 = 0). For 01101, 11101, 31101 or 41101, specify a value of 1100. (01101 - 00001 = 1100) (11101 -10001 = 1100) (31101 - 30001 = 1100) (41101 - 40001 = 1100) **Note:** If the documentation for a particular Modbus Server device lists data addresses in hexadecimal (base16) notation, you must convert the hexadecimal value to a decimal value for this parameter. In such cases, you usually do not have to subtract 1 from the converted decimal number, as this addressing scheme typically uses the exact offset address expressed as a hexadecimal number.

9.4 Network Diagnostics

9.4.1 MBS PCB Diagnostics

ProSoft Configuration Builder is used to troubleshoot the MBS Driver. For instructions on how to access the diagnostics, see section section *5.5 Diagnostics and Troubleshooting*.

The following table summarizes the status information available in ProSoft Configuration Builder for the MBS driver:

Connection Type	Submenu Item	Description
MBS Port 1	Config	Configuration settings for MBS Port 1.
	Comm Status	Status information for MBS Port 1 commands. Displays a summary of all the errors resulting from MBS Port 1 commands.
	Slave Status	Current poll status of each Slave device on MBS Port 1.
	Modbus Cmds	Configuration for the MBS Port 1 Modbus command list.
	Modbus Cmd Errors (Decimal)	Current error codes for each command on the MBS Port 1 command list in decimal number format. A zero means there is currently no error for the command.
	Modbus Cmd Errors (Hex)	Current error codes for each command on the MBS Port 1 command list in hexadecimal number format. A zero means there is currently no error for the command.

9.4.2 MBS Status Data in Upper Memory

The MBS Port 1 has an associated status data area located in the Container's upper memory. The Data Map functionality of the Container can be used to map this data into the user data range (0 to 9999) of the Container's database. See chapter *10 Mapping Data in Container Memory*.

9.4.2.1 General Modbus Error and Status Data

The following table lists the starting addresses in upper memory where the Container writes general error and status data for each Modbus Master or Slave port. Note that all the status values are initialized to zero (0) at power-up, cold boot and during warm boot.

Modbus Port	Starting Address
1	14400

Warning: To access this data through a Modbus request, you must move the data into the 0 to 9999 user data range. See chapter *10 Mapping Data in Container Memory*.

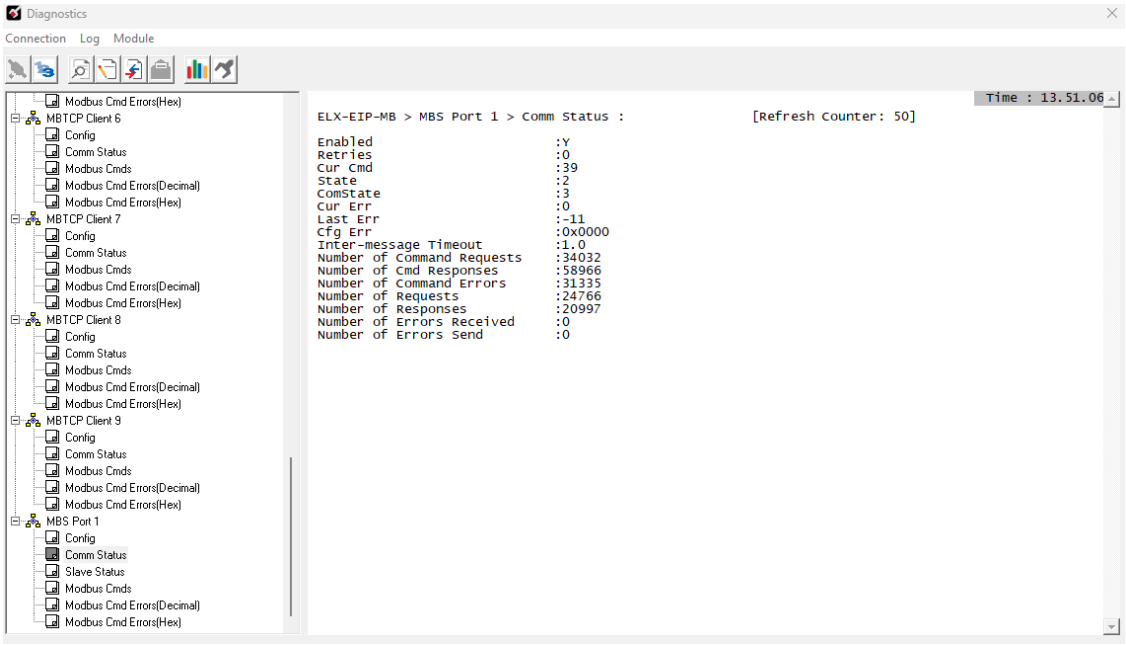
9.4.2.1.1 Port 1 General Modbus Error and Status Layout

The following status data is available starting at address 14400.

Example Internal Database Address	Offset	Description
14400	0	Number of Command Requests
14401	1	Number of Command Responses
14402	2	Number of Command Errors
14403	3	Number of Requests
14404	4	Number of Responses
14405	5	Number of Errors Sent
14406	6	Number of Errors Received
14407	7	Configuration Error Code
14408	8	Current Error/Index
14409	9	Last Error/Index

9.4.2.1.2 MBS Port 1 Status

Note: There are two additional port status parameters that are not available from the General Modbus Error and Status Data area. The status values, Port *State* and Port *ComState* (circled in red in the graphic below), can be found only in the ProSoft Configuration Builder diagnostic menus under *Comm Status* for a port.



These status values are state registers. They are used as "scratchpad" areas by the Container firmware to keep track of the current logical state of activities on the MBS port that is configured as a Slave. These state registers are constantly changing as the Container progresses through the various stages needed to process communication on the port. This processing happens faster than can be followed, unless a port error causes the value to remain constant for a noticeable length of time.

The diagnostic screen displays a snapshot of the data; that is, the current values at the time they are displayed, but the Container does not update the data. They are not live-data screens. Therefore, to see a change in *State* or *ComState*, you must repeatedly update the screen by pressing the appropriate menu key to have the screen refreshed. Depending on the refresh timing you may or may not see a change in the displayed values every time you call for an update.

The *State* register may display the following values in **Slave** mode:

State Value	Description
-1	Abort state / De-assert RTS (Request To Send)
0	Listening - Waiting for Master poll
1	Checking for response (receiving data)
2	Building response message from internal data
2000	Sending Slave response to the Master

The *State* register may display the following values in **Master** mode:

State Value	Description
-1	Abort state / De-assert RTS (Request To Send)
0	Building command state
1	Sending initial message to Slave
2	Checking for broadcast message
3	Getting response
4	Validating CRC, function code, and Slave ID
5	Parsing response message
7	Modbus Master is fetching next command
100	Waiting for command delay
10000	Set by another task to turn off the port

The *ComState* register may display any of the following values:

ComState Value	Description
0	Port not sending data (Wait for send)
1	Setting up minimum Response Delay Timer
101	Waiting for Minimum Response Delay to timeout
2	Turning on RTS line and starting RTS-On Delay Timer
3	Waiting for RTS-On Delay timer to timeout
4	Sending data
5	Waiting for all data to be sent and starting RTS-Off Delay Timer
6	Waiting for RTS-Off Delay timeout and then turning OFF RTS line after timeout

9.4.2.2 Master Port: Command List Errors

When the MBS port is configured as a Master, the individual command errors are returned to the address locations specified in the following table. The port can have up to 100 commands configured. Each configured command uses one word to store an execution status value from the most recent command execution attempt. Note that all the status values are initialized to zero (0) at power-up, cold boot and during warm boot.

Modbus Port	Address Range
1	14410 to 14509

Warning: To access this data through a Modbus request, you must move the data into the 0 to 9999 user data range. See chapter *10 Mapping Data in Container Memory*.

9.4.2.2.1 Port 1 Command Error List Layout

The first word in the specified register location contains the status/error code for the first command in the MBS port's *Command List*. Successive words in the *Command Error List* are associated with corresponding commands in the list.

Internal Database Address (Example)	Offset	Description
14410	0	Command #1 Error Code
14411	1	Command #2 Error Code
14412	2	Command #3 Error Code
14413	3	Command #4 Error Code
14414	4	Command #5 Error Code
...	...	...
14507	97	Command #98 Error Code
14508	98	Command #99 Error Code
14509	99	Command #100 Error Code

Note that the Container initializes the Command Error List tables to zero (0) at power-up, cold boot, and warm boot. If a command executes successfully, the value in the associated register remains at zero (0), indicating no command error was detected. Any non-zero value in this table indicates the command experienced an error.

The data in this table is dynamic and is updated each time a command is executed. Therefore, if the command fails once and succeeds on the next attempt, the error code from the previously failed attempt is overwritten. Error codes are not archived in the Container's database. To see if the port has experienced an error since the most recent restart and what the most recently occurring error was, if any, you can check the *Last Error/Index* parameter.

9.4.2.3 Master Port: Modbus Slave List Status

The Container stores *Slave List Status* values in the address locations specified in the following table. Note that all the status values are initialized to zero (0) at power-up, cold boot and during warm boot.

Modbus Port	Address Range
1	14510 to 14764

Warning: To access this data through a Modbus request, you must move the data into the 0 to 9999 user data range. See chapter *10 Mapping Data in Container Memory*.

9.4.2.3.1 Port 1 Slave List Status Layout

Internal Database Address (Example)	Offset	Description
14510	0	Slave #0 Status
14511	1	Slave #1 Status
14512	2	Slave #2 Status
14513	3	Slave #3 Status
14514	4	Slave #4 Status

The Slave status list contains the current poll status of each Slave device on the Master port. Slaves attached to a Master port can have one of three states:

Status	Description
0	The Slave is not defined in the command list for the Master port and is not polled from the Command List.
1	The Slave is configured to be polled by the Master port, and the most recent communication attempt was successful.
2	The Master port failed to communicate with the Slave device. Communication with the Slave is suspended for a user-defined period based on the scanning of the command list.

Slaves are defined to the system when the Container loads the *Master Command List* during start-up and initialization. Each Slave defined is set to a state value of 1 in this initial step. If the Master port fails to communicate with a Slave device (timeout expired on a command, retries failed), the Master sets the state of the Slave to a value of 2 in this status table. This suspends communication with the Slave device for a user-specified *Error Delay Count*.

When the Master first suspends polling of a particular Slave, it creates an *Error Delay Counter* for this Slave address and sets the value in that counter equal to the *Error Delay Counter* parameter. Then, each time the Container scans a command in the list that has the address of a suspended Slave, the Container decrements the delay counter value. When the value reaches zero, the Container sets the Slave state to 1 and re-enables the polling of the Slave.

The first word in the defined register locations contains the status code for Slave node address 0. Each successive word in the list is associated with the next node in sequence, up to Slave node address 247.

9.5 MBS Error Codes

These are error codes that are part of the Modbus protocol or are extended codes unique to the Container.

9.5.1 Standard Modbus Exception Code Errors

These error codes are generated or returned on both the Master and Slave ports. These codes are the standard Modbus errors.

Code	Description
1	Illegal Function
2	Illegal Data Address
3	Illegal Data Value
4	Failure in Associated Device
5	Acknowledge
6	Busy, Rejected Message

9.5.2 Container Communication Error Codes

These Container-specific error codes are also returned from the command polling process and stored in the *Command Error List* memory area.

Code	Description
-1	CTS modem control line not set before transmitting
-2	Timeout while transmitting message
-11	Timeout waiting for response after request
253	Incorrect Slave address in response
254	Incorrect function code in response
255	Invalid CRC/LRC value in response

9.5.3 MBS Command List Error Codes

These command-specific error codes are detected during command list loading at Container reboot and are stored in the *Command Error List* memory area.

Code	Description
-40	Missing fields in command line
-41	Invalid *Enable* code parameter; greater than 3.
-42	*Internal Address* parameter exceeds the database size
-43	Invalid node address; Node > 255 or Node = 0 for Function Code Read commands
-44	*Count* parameter set to 0
-45	Invalid *Function Code* parameter
-46	Invalid *Swap Code* parameter; greater than 3.

9.5.4 MBS Configuration Error Codes

The MBS configuration error codes are detected during configuration loading at gateway reboot and are stored as bits at internal address 14407. For more information, see section *9.4.2.1.1 Port 1 General Modbus Error and Status Layout*.

Bit	Description
0	Port not enabled
1	Invalid RS Interface
2	Invalid Type (neither Master or Slave)
3	Invalid Protocol (neither RTU or ASCII)
4	Invalid Baud Rate
5	Invalid Parity
6	Invalid Data Bits
7	Invalid Stop Bits
8	Invalid CTS value
9	Retry Count out of range (0 to 10)
10	Invalid Float Flag/Start/Offset
11	Invalid Slave ID
12	Invalid Offset (Bit/Word/Output/Hold)
13	Invalid Command Control Register

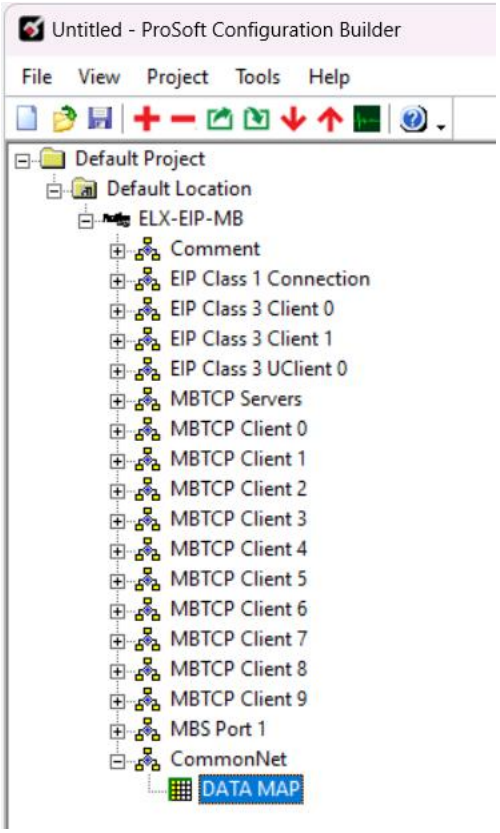
10 Mapping Data in Container Memory

In ProSoft Configuration Builder, the *DATA MAP* feature is used to copy data between areas in the ELX-EIP-MB container's internal database. This allows the copying of data to different addresses within the ELX-EIP-MB container database in order to create simpler data requests and control. Use this feature for the following tasks:

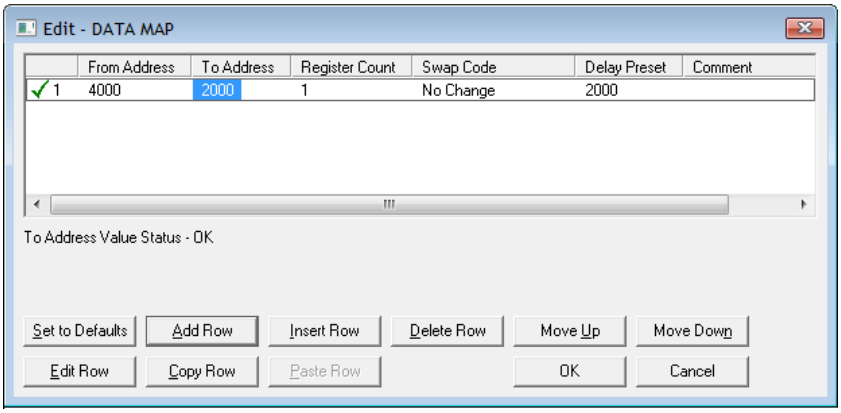
- Copy a maximum of 100 registers per Data Map command and configure a maximum of 200 separate copy commands.
- Copy data from the error or status tables in the ELX-EIP-MB container's upper memory to database registers in the User Data area to make status information available to the protocol drivers.
- Rearrange the byte and/or word order during the copy process. For example, by rearranging byte or word order, converting floating-point values to the correct format for a different protocol.
- Condense widely dispersed data into one contiguous data block, making it easier to access.

To create a data map:

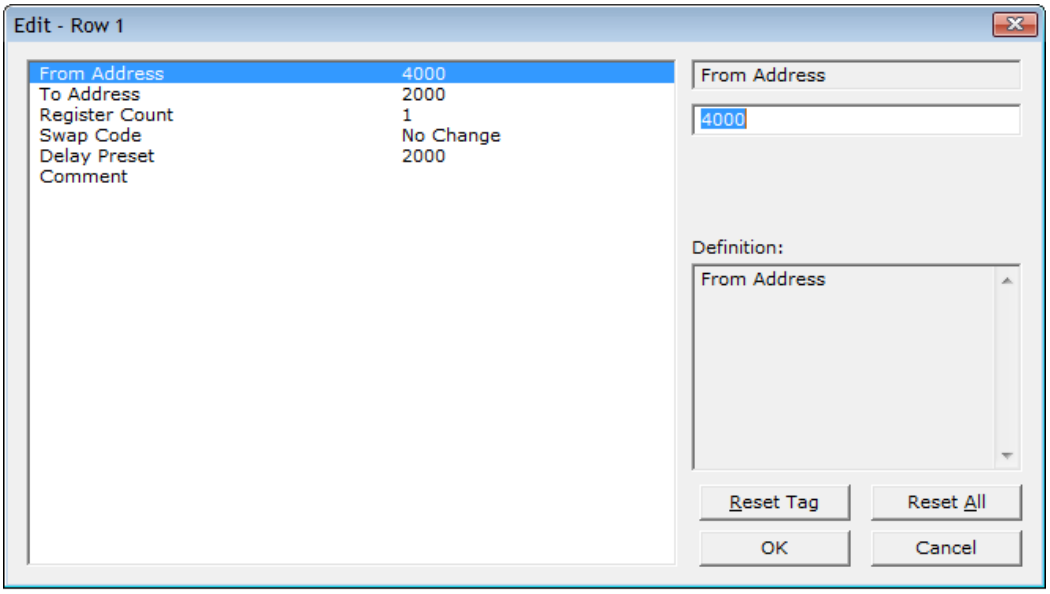
1 Expand the ELX-EIP-MB tree and click the **[+]** next to **COMMONNET** and then double-click **DATA MAP**.



2 In the *Edit - Data Map* dialog, click **ADD ROW**.



3 Click **EDIT ROW** to edit the parameters for the mapping.



Parameter	Description
From Address	0 to highest *Status Data* address This parameter specifies the beginning internal database register address for the copy operation. This address can be any valid address in the User Data area or the status data area of the ELX-EIP-MB container.
To Address	0 to 9999 This parameter specifies the beginning destination register address for the copy operation. This address must always be within the User Data area. Make sure to specify a destination address that does not overwrite data that is stored in memory by one of the communication protocols running on the ELX-EIP-MB container.
Register Count	1 to 100 This parameter specifies the number of registers to copy.

Parameter	Description
Swap Code	This parameter may be needed when dealing with floating-point or other multi-register values. Swapping the order of the bytes in the registers to change the alignment of bytes between different protocols may be needed. **NO SWAP**: No change is made in the byte ordering (1234 = 1234) **WORD SWAP**: The words are swapped (1234 = 3412) **WORD AND BYTE SWAP**: The words are swapped, then the bytes in each word are swapped (1234 = 4321) **BYTES**: The bytes in each word are swapped (1234 = 2143)
Delay Preset	This parameter sets an interval for each *Data Map* copy operation. The value for the *Delay Preset* is not a fixed amount of time. It is the number of firmware scans that must transpire between copy operations. The firmware scan cycle can take a variable amount of time, depending on the level of activity of the protocol drivers running on the ELX-EIP-MB container. Each firmware scan can take from one to several milliseconds to complete. Therefore, *Data Map* copy operations cannot be expected to happen at regular intervals. If multiple copy operations (several rows in the *Data map* section) happen too frequently or all happen in the same update interval, they could delay the process scan of the ELX-EIP-MB container protocols, which could result in slow data updates or missed data on communication ports. To avoid these potential problems, set the *Delay Preset* to different values for each row in the *Data Map* section and set them to higher, rather than lower, numbers. For example, *Delay Preset* values below 1000 could cause a noticeable delay in data updates through the communication ports. Do not set all *Delay Presets* to the same value. Instead, use different values for each row in the Data Map such as 1000, 1001, and 1002 or any other different *Delay Preset* values. This prevents the copies from happening concurrently and prevents possible process scan delays.

4 Repeat the steps above to add more memory mappings.
5 Click **OK** when complete.
6 Save the file as needed.

11 Security

For information about security recommendations for the ELX-EIP-MB Container, please see the following documents:

- Belden Horizon Console Security Architecture and Assurance (www.prosoft-technology.com)
- ELX3 User Manual (www.prosoft-technology.com)

12 Appendix A - CIP Objects

12.1 Identity Object (0x01)

Supported Services

Service Name	Service Code (HEX)
Get_Attribute_All	0x01
Get_Attribute_Single	0x0E

Class Attributes (Class 0)

Attribute	Access	Name	Data Type	Description
1	GET	Revision	UINT	Revision of this object
2	GET	Max Instance	UINT	Maximum instance number of this object

Instance Attributes (Class 1)

Attribute	Access	Name	Data Type	Description
1	GET	Vendor ID	UINT	Identification of the vendor
2	GET	Device Type	UINT	Identification of the product type
3	GET	Product Code	UINT	Identification of the product
4	GET	Revision	STRUCT of:	Revision of the product
		Major Revision	USINT	
		Minor Revision	USINT	
5	GET	Status	WORD	Status of device
6	GET	Serial Number	UDINT	Serial number of the product
7	GET	Product Name	SHORT_STRING	Name of the product

12.2 Port Object (0xF4)

Supported Services

Service Name	Service Code (HEX)
Get_Attribute_Single	0x0E

Class Attributes (Class 0)

Attribute	Access	Name	Data Type	Description
1	GET	Revision	UINT	Revision of this object
2	GET	Max Instance	UINT	Maximum instance number of this object
3	GET	Number of Instances	UINT	Number of ports currently instantiated
8	GET	Entry Port	UINT	Returns the instance of the Port Object that describes the port through which this request entered the device
9	GET	Port Instance Info	ARRAY of STRUCT of	Array of structures containing instance attributes 1 and 2 from the instance
		Port Type	UINT	Enumerates the Port Type
		Port Number	UINT	CIP port number of the port

Instance Attributes (Class 1)

Attribute	Access	Name	Data Type	Description
1	GET	Port Type	UINT	Enumerates the Port Type
2	GET	Port Number	UINT	CIP port number of the port
3	GET	Logical Link Object	STRUCT of:	
		Path Length	USINT	Number of 16-bit words in the following path
		Link Path	USINT	Logical path segments that identify the object for this port
4	GET	Port Name	SHORT_STRING	String that names the communications interface. The maximum number of characters in the string is 64. For example: “Port A”
7	GET	Port Number and Node Address	Padded EPATH	Single Port Segment containing the Port Number and Link Address of this device on this port
10	GET	Port Routing Capabilities	DWORD	Bit string that defines the routing capabilities of this port

12.3 TCP/IP Interface Object (0xF5)

Supported Services

Service Name	Service Code (HEX)
Get_Attribute_All	0x01
Get_Attribute_Single	0x0E
Set_Attribute_Single	0x10

Class Attributes (Class 0)

Attribute	Access	Name	Data Type	Description
1	GET	Revision	UINT	Revision of this object

Instance Attributes (Class 1)

Attribute	Access	Name	Data Type	Description
1	GET	Status	DWORD	Interface status
2	GET	Configuration Capability	DWORD	Interface capability flags
3	GET	Configuration Control	DWORD	Interface control flags
4	GET	Physical Link Object	STRUCT of:	Path to physical link object
		Path Size	UINT	Size of Path
		Path	Padded EPATH	Logical segments identifying the physical link object
5	GET	Interface Configuration	STRUCT of:	TCP/IP network interface configuration
		IP Address	UDINT	The device's IP address
		Network Mask	UDINT	The device's network mask
		Gateway Address	UDINT	Default gateway address
		Name Server	UDINT	Primary name Server
		Name Server 2	UDINT	Secondary name Server
		Domain Name	UDINT	Default domain name
6	GET	Host Name	STRING	Host name
13	SET	Encapsulation Inactivity Timeout	UINT	Number of seconds of inactivity before TCP connection is closed. Default value is 120. **0**: Disable timeout **1** to **3600**: Timeout in seconds

12.4 Ethernet Link Object (0xF6)

Supported Services

Service Name	Service Code (HEX)
Get_Attribute_All	0x01
Get_Attribute_Single	0x0E
Set_Attribute_Single	0x10

Class Attributes (Class 0)

Attribute	Access	Name	Data Type	Description
1	GET	Revision	UINT	Revision of this object

Instance Attributes (Class 1)

Attribute	Access	Name	Data Type	Description
1	GET	Interface Speed	UDINT	Interface speed currently in use
2	GET	Interface Flags	DWORD	Interface status flags
3	GET	Physical Address	ARRAY of 6 USINTs	MAC layer address
10	GET	Interface Label	SHORT_STRING	Human readable identification
11	GET	Interface Capability	STRUCT of:	Indication of capabilities of the interface
		Capability Bits	DWORD	Interface capabilities, other than speed/duplex
		Speed/Duplex Options	STRUCT of:	Indicates speed/duplex pairs supported in the Interface Control attribute
			USINT	Speed/Duplex Array Count
			ARRAY of: STRUCT of:	Speed/Duplex Array
			UINT	Interface Speed in Mbps
			USINT	Interface Duplex Mode **0**: Half duplex **1**: Full duplex **2** to **255**: Reserved

12.5 LLDP Management Object (0x109)

Supported Services

Service Name	Service Code (HEX)
Get_Attribute_All	0x01
Get_Attribute_Single	0x0E
Set_Attribute_Single	0x10

Class Attributes (Class 0)

Attribute	Access	Name	Data Type	Description
1	GET	Revision	UINT	Revision of this object
2	GET	Max Instance	UINT	Maximum instance number of this object
3	GET	Number of Instances	UINT	Number of object instances currently created at this class level of the device

Instance Attributes (Class 1)

Attribute	Access	Name	Data Type	Description
1	SET	LLDP Enable	UDINT	
		LLDP Enable Array Length	UINT	Number of bits defined in the LLDP Enable Array member of this structure. LLDP Enable Array Length (UINT): 1 plus the value of Class attribute 2 from the Ethernet Link Object (0xF6)
		LLDP Enable Array	ARRAY of: BYTE	Enable generation of LLDP Frames both Globally and per Port and the processing of received LLDP frames globally. LLDP Enable Array (BYTE): Bit **0**: Global Enable, LLDP Tx & Rx Enabled (1) Bit **1**: LLDP Tx Enabled (Instance 1 of Ethernet Link Object) (1)
2	SET	msgTxInterval	UINT	From 802.1AB-2016. The interval in seconds at which LLDP frames are transmitted from this device **0**: Reserved **1** to **3600**: Message Transmission Interval for LLDP frames **3601** to **65535**: Reserved Recommended default value is **30**.
3	SET	msgTxHold	USINT	From 802.1AB-2016. A multiplier of *msgTxInterval* to determine the value of the TTL TLV sent to neighboring devices **0**: Reserved **1** to **100**: Message Transmission Multiplier for LLDP Frames

				101 to **255**: Reserved Recommended default value is **4**.
4	GET	LLDP Datastore	WORD	An indication of the retrieval methods for the LLDP Datastore supported by this device
5	GET	Last Change	UDINT	The value of *sysUpTime* taken the last time any entry (ignoring TTL) in the LLDP Datastore (as indicated in instance attribute 4) changed

12.6 LLDP Data Table Object (0x10A)

Supported Services

Service Name	Service Code (HEX)
Get_Attribute_All	0x01
Get_Attribute_Single	0x0E

Class Attributes (Class 0)

Attribute	Name	Access	Data Type	Description
1	Revision	GET	UINT	Revision of this object
2	Max Instance	GET	UINT	Maximum instance number of an object currently created in this class level of the device
3	Number of Instances	GET	UINT	Number of object instances currently created at this class level of the device

Instance Attributes (Class 1)

Attribute	Access	Name	Data Type	Description
1	GET	Ethernet Link Instance Number	UDINT	The local instance number of the Ethernet Link Object that matches the physical Ethernet port the LLDP frame populating this instance was received on, if known. **0**: Unknown **1** to **65535**: Ethernet Link Object (0xF6) Instance Number
2	GET	MAC Address	ETH_MAC_ADDR	The neighboring MAC Address received from the CIP MAC Address, Chassis ID, or Port ID TLV
3	GET	Interface Label	SHORT_STRING	The neighboring Interface Label received from the CIP Interface Label, Chassis ID or Port ID TLV
4	GET	Time To Live	UINT	The number of seconds the neighboring information is to be considered valid **0**: Reserved **1** to **65535**: Time to Live (in seconds) **Note:** A received TTL TLV value **0** means the table entry should be removed per IEEE 802.1AB-2016
5	GET	System Capabilities TLV	STRUCT of:	The system capabilities field contains bitmaps of the capabilities that define the primary function(s) of the neighboring system
		System Capabilities	WORD	The capabilities supported by the neighboring device that are based on currently loaded firmware.
		Enabled Capabilities	WORD	The capabilities currently enabled on the neighboring device
6	GET	IPv4 Management Addresses	STRUCT of:	The IPv4 management addresses of the neighboring device
		Management Address Count	USINT	Number of implemented management addresses

		Management Address	ARRAY of UDINT	IPv4 Management Addresses of the neighboring device
7	GET	CIP Identification	STUCT of:	The CIP Identification TLV of the neighboring device
		Vendor ID	UINT	
		Device Type	UINT	
		Product Code	UINT	
		Major Revision	BYTE	
		Minor Revision	USINT	
		CIP Serial Number	UDINT	
8	GET	Additional Ethernet Capabilities	STRUCT of:	A TLV for Ethernet Preemption Support from the neighboring device **0**: Not Supported **1**: Supported
		Preemption Support	BOOL	Allows a link partner to know if preemption is supported on the link **0**: Not Enabled **1**: Enabled
		Preemption Status	BOOL	Allows a link partner to know if preemption is enabled on the link **0**: Not Active **1**: Active
		Preemption Active	BOOL	Allows a link partner to know if link preemption has passed embedded verification **0**: 64 octets **1**: 128 octets **2**: 192 octets **3**: 256 octets **4** to **255**: Reserved
		Additional Fragment Size	USINT	The number of octets that must be transmitted of a frame before preemption occurs
9	GET	Last Change	UDINT	The value of *sysUpTime* taken the last time any attribute in this instance changed

(The maximum number of Supported Instances is 48)

13 Support, Service, and Warranty

13.1 Contacting Technical Support

ProSoft Technology, Inc. is committed to providing the most efficient and effective support possible. Before calling, please gather the following information to assist in expediting this process:

1. Product Version Number
2. System architecture
3. Network details

If the issue is hardware related, we will also need information regarding:

1. Module configuration and associated ladder files, if any
2. Module operation and any unusual behavior
3. Configuration/Debug status information
4. LED patterns
5. Details about the interfaced serial, Ethernet or Fieldbus devices

North America (Corporate Location)	Europe / Middle East / Africa Regional Office
Phone: +1 661-716-5100 ps.support@belden.com Languages spoken: English, Spanish REGIONAL TECH SUPPORT ps.support@belden.com	Phone: +33.(0)5.34.36.87.20 ps.europe@belden.com Languages spoken: English, French, Hindi, Italian REGIONAL TECH SUPPORT ps.support.emea@belden.com
Latin America Regional Office	**Asia Pacific Regional Office**
Phone: +52.222.264.1814 ps.latinam@belden.com Languages spoken: English, Spanish, Portuguese REGIONAL TECH SUPPORT ps.support.la@belden.com	Phone: +60.3.2247.1898 ps.asiapc@belden.com Languages spoken: Bahasa, Chinese, English, Hindi, Japanese, Korean, Malay REGIONAL TECH SUPPORT ps.support.ap@belden.com

For additional ProSoft Technology contacts in your area, please see:
www.prosoft-technology.com/About-Us/Contact-Us

13.2 Warranty Information

For details regarding ProSoft Technology's legal terms and conditions, please see:
www.prosoft-technology.com/ProSoft-Technology-Legal-Terms-and-Conditions

For Return Material Authorization information, please see:
www.prosoft-technology.com/Services-Support/Return-Material-Instructions