CYBERLOGIC OPC SERVER MBX DRIVER AGENT

For All Modicon Programmable Controller Families
Version 6.00 for Windows® XP/2000/NT/Server 2003

Document last revision date January 3, 2006

TABLE OF CONTENTS

INTRODUCTION

The MBX OPC Server is part of the Cyberlogic OPC Server architecture. In that architecture, a single OPC Server can handle various types of communications, provided by the device driver agent specific modules. This document covers only the MBX driver agent specific information. For information on the Cyberlogic OPC Server, refer to the Cyberlogic OPC Server Help.

The MBX driver agent allows the Cyberlogic OPC Server to communicate to 184, 384, 484, 584, 884, 984, Micro 84, Modcell, Quantum and Momentum controllers or any Modbus-compatible device over Modbus (RTU and ASCII), Modbus Plus and Ethernet networks.

The MBX driver agent is compatible with all MBX family products. The industry-standard MBX driver suite, including the MBX Driver, Ethernet MBX driver, Serial MBX Driver and MBX Gateway Driver, provide the low-level communication services. It is also compatible with the MBX companion products, such as the Virtual MBX Driver for 16-bit legacy DOS/Windows applications and the MBX Bridge, which routes messages between Ethernet, Modbus, Modbus Plus and MBX Gateway nodes.

Main Features

- Supports Modicon Modbus (RTU and ASCII), Modbus Plus and Ethernet networks
- Supports 184, 384, 484, 584, 884, 984, Micro 84, Modcell, Quantum and Momentum controllers and all Modbus-compatible devices
- Supports solicited and unsolicited communications
- Supports I/O to Coils (0x), Inputs (1x), Input Registers (3x), Holding Registers (4x), General Reference Registers (6x) and Global Data
- Supports raw data bit-swapping at the bit, nibble, byte, word or double-word level
- Supports health-monitoring of Network Connections and Network Nodes
- Auto-configure mode detects all standard data sources (PLCs) to automatically set its message optimization parameters
- High performance due to advanced multithreaded design
- DirectAccess to all registers in any device
- Supports a variety of data types at the data source
- Compatible with Cyberlogic OPC Server architecture
- Compatible with all MBX family products
- Includes the standard MBX Driver Suite software

What Should I Do Next?

This document describes the features of the MBX OPC Driver Agent. For information on features common to all Cyberlogic OPC Server products, refer to the Cyberlogic OPC Server Help.

For architectural and implementation details of the MBX OPC Driver Agent, read the Theory of Operation section. This section describes the implementation of various features, as well as troubleshooting aids.

You must configure the Cyberlogic OPC Server after installing it. You will find information on this topic in the Configuration section. This section contains a step-by-step tutorial for first-time users along with a detailed explanation of the configuration tools.

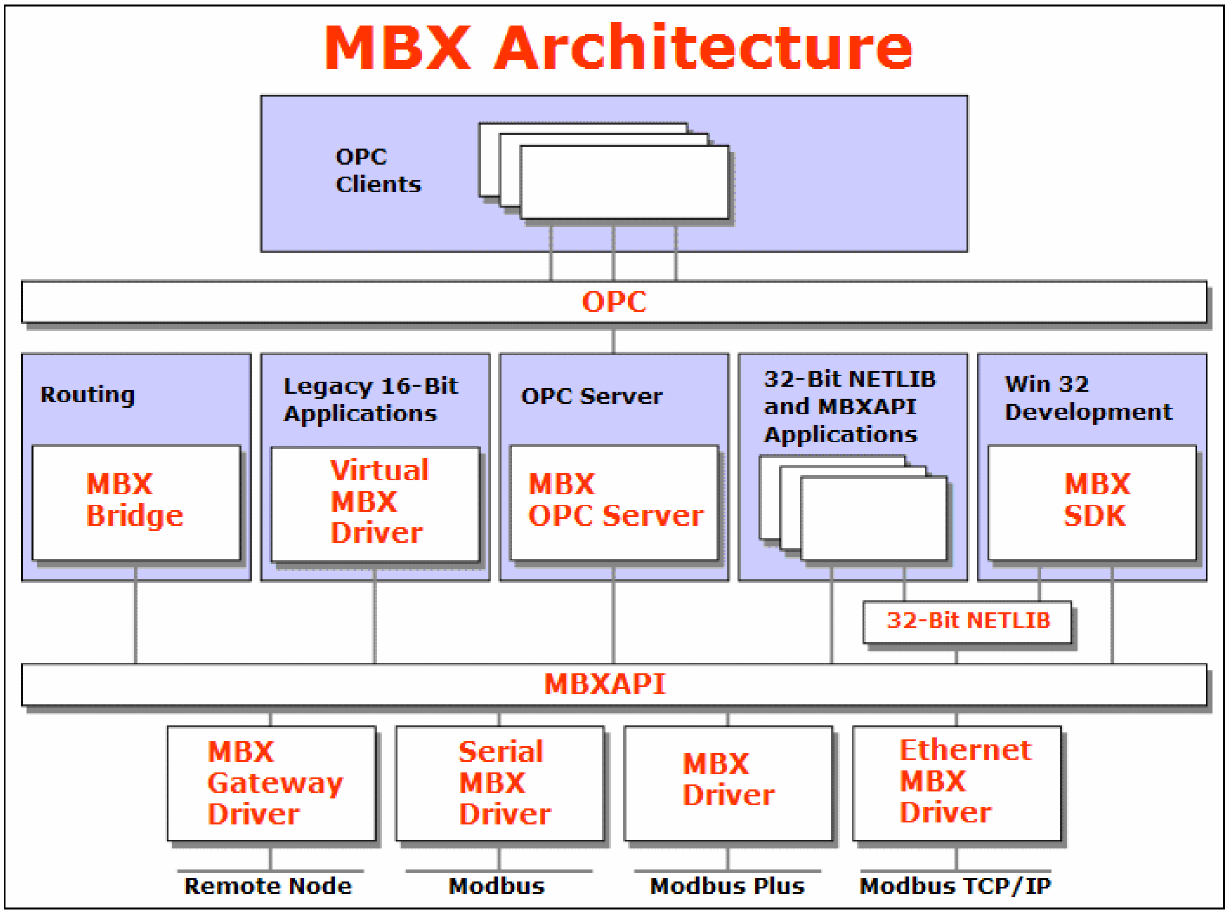
If you have already configured the Server, refer to the Validation & Troubleshooting section to verify that it operates as expected. In case of problems, this section also includes problem solving hints.

This document is also provided in the PDF file format. You can use the Adobe® Reader program to view and print the PDF files.

MBX ARCHITECTURE AND COMPANION PRODUCTS

This section illustrates the layout of the MBX architecture. It includes a description of each MBX product along with suggested methods for employing these products to support Modicon networks.

The Cyberlogic MBX family for Windows XP/2000/NT consists of several well-integrated products that provide connectivity for Modicon's Modbus, Modbus Plus and Modbus TCP/IP (Ethernet) networks in distributed environments.



The MBX architecture presents a consistent framework to address different connectivity needs.

Software products available in the MBX family are:

MBX Driver: This is Cyberlogic's device driver for Modbus Plus host interface adapters. The MBX Gateway Server is included for remote connectivity.

Ethernet MBX Driver: This driver provides Modbus Plus emulation over TCP/IP. The MBX Gateway Server is included for remote connectivity.

Serial MBX Driver: This driver provides Modbus Plus emulation over serial Modbus. The MBX Gateway Server is included for remote connectivity.

MBX Gateway Driver: This product provides access to Modicon's Modbus, Modbus Plus and Modbus TCP/IP networks from remote locations.

Virtual MBX Driver: This driver works with the other MBX drivers to permit 16-bit legacy software to run in 32-bit Windows operating systems.

MBX Bridge: This product allows you to bridge any combination of Modicon networks by routing messages between MBX devices.

MBX OPC Server: Cyberlogic's premium OPC Server connects OPC compliant client software applications to data sources over all Modicon networks.

MBX SDK: This is a software development kit for MBXAPI and NETLIB compliant development.

MBX Driver

The 32-bit MBX Driver provides connectivity between Modicon ModConnect host interface adapters and 32-bit applications running under Windows XP/2000/NT.

The kernel mode device driver of the MBX Driver is the highest performance Modbus Plus driver in the industry. The driver operates in either interrupt or polled mode and supports all current Modicon ModConnect host interface adapters for ISA, EISA, MCA, PCI and PC Card (PCMCIA) buses. Multiple interface cards can be installed at the same time, limited only by the number of available slots. Full implementation of all Modbus Plus features provides support for Data Master/Slave, Program Master/Slave, Global Data and Peer Cop. The high-performance native API (MBXAPI) of the MBX Driver takes advantage of the event-driven, multitasking, multithreaded features of 32-bit operating systems.

The driver includes the MBX Gateway Server for remote access by the MBX Gateway Driver and is fully compatible with all other MBX family products.

Ethernet MBX Driver

The 32-bit Ethernet MBX Driver provides connectivity between Modbus TCP/IP compatible processors and Windows XP/2000/NT based 32-bit applications using either Modicon NETLIB or Cyberlogic's high-performance MBXAPI interface specification. It provides Data Master/Slave and Program Master/Slave features of Modbus Plus on Ethernet networks.

The driver includes the MBX Gateway Server for remote access by the MBX Gateway Driver and is fully compatible with all other MBX family products. The Ethernet MBX Driver does not require a special Ethernet adapter. It is compatible with all Ethernet cards supported by Windows.

Serial MBX Driver

The Serial MBX Driver provides connectivity to Modbus-compatible devices through the standard serial COM ports. It supports both master and slave node communications.

The driver includes the MBX Gateway Server for remote access by the MBX Gateway Driver and is fully compatible with all other MBX family products.

MBX Gateway Driver

The MBX Gateway Driver lets you access Modbus, Modbus Plus and Modbus TCP/IP networks from a remote location. Through a standard LAN, your local applications can use MBX devices on Gateway server nodes as though they were on your local system.

The remote client running the MBX Gateway Driver must be a Windows XP/2000/NT node. By accessing the Modbus, Modbus Plus and Ethernet networks connected to server nodes on a network, the MBX Gateway Driver provides complete MBX Driver functionality to the client node, including support for Data Master/Slave, Program Master/Slave, Global Data and Peer Cop. A host interface adapter, such as a Modicon SA85 card, is not required on the client node. MBX Gateway Driver nodes can communicate with multiple Gateway servers and all Windows XP/2000/NT-compatible computer networks are supported.

The MBX Gateway Driver is compatible with all other MBX family products.

Virtual MBX Driver

The Virtual MBX Driver enables 16-bit NETLIB/NetBIOS-compatible applications, such as Modsoft and Concept, to run concurrently with 32-bit applications on the same computer. It allows multiple 16-bit applications and multiple instances of a single 16-bit application to run under the 32-bit Windows operating systems.

The Virtual MBX Driver is fully compatible with all MBX components and requires at least one of these drivers to operate:

- MBX Driver
- Ethernet MBX Driver
- Serial MBX Driver
- MBX Gateway Driver

MBX Bridge

The MBX Bridge seamlessly routes messages between MBX-compatible devices. For example, the MBX Bridge can route messages between Ethernet and Modbus Plus networks, between Modbus and Modbus Plus networks or any other combination of the supported networks. Depending on the user’s needs, it requires one or more of the following products to operate:

- MBX Driver
- Ethernet MBX Driver
- Serial MBX Driver
- MBX Gateway Driver

MBX OPC Server

The Cyberlogic MBX OPC Server connects OPC-compliant clients to Modicon Modbus, Modbus Plus and Ethernet networks. It supports the latest OPC Data Access and OPC Alarms and Events specifications and uses the MBX drivers for connectivity to Modicon networks.

The MBX OPC Server supports multiple, priority-based access paths for reliable, redundant communications. It also supports both solicited and unsolicited communications and uses an advanced transaction optimizer to guarantee minimum load on your networks. With only a couple of mouse clicks, the MBX OPC Server will automatically detect and configure the attached networks and node devices in seconds. Other noteworthy features include DirectAccess, Data Write Protection and Health Watchdog.

MBX SDK

Software developers can use the MBX SDK to provide connectivity to Modbus, Modbus Plus and Ethernet networks from their 32-bit C/C++ applications.

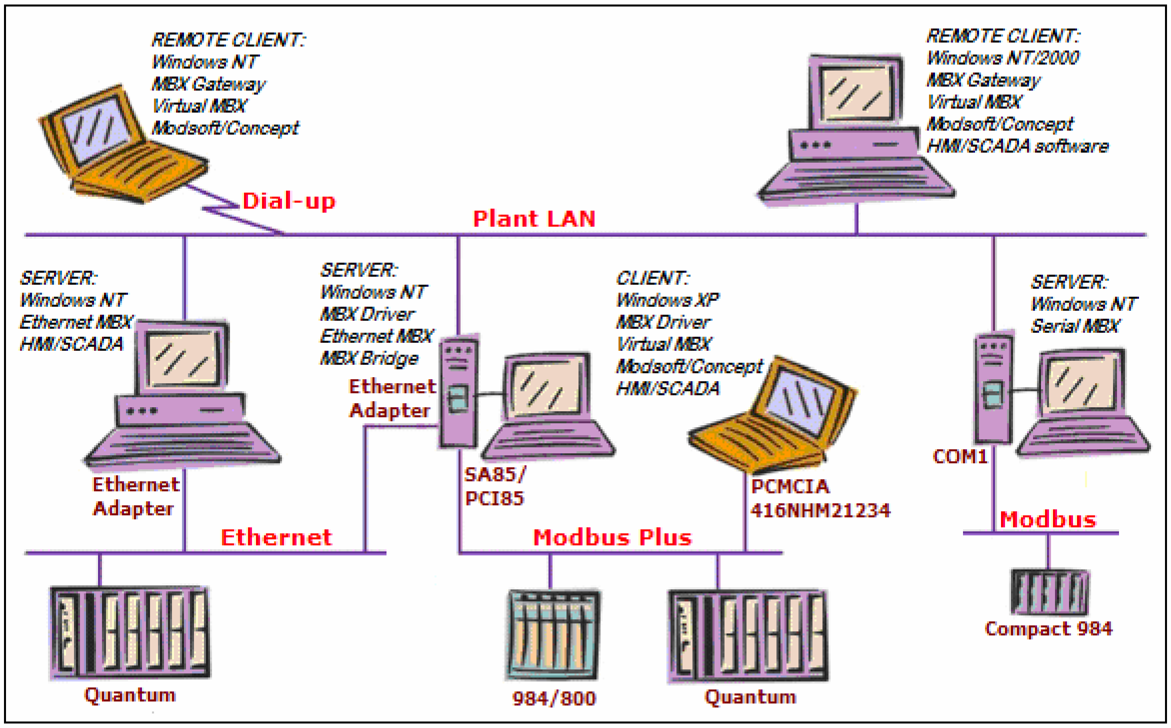
The SDK supports two styles of interfaces, the industry-standard NETLIB and Cyberlogic's high-performance MBXAPI. The NETLIB interface is an excellent bridge for developers who would like to port their 16-bit applications to the 32-bit Windows environments. Developers of new applications can use either the NETLIB or the MBXAPI interface.

Since all MBX driver products are built on the same MBX architecture, applications developed with the MBX SDK can be used with all MBX drivers and can execute under all 32-bit Windows operating systems.

Blending MBX Supported Networks

The MBX driver products provide support for all Modicon networks through a common architecture, with identical programming interfaces: the MBXAPI and the industry-standard NETLIB. This ensures that virtually all of the existing Modbus Plus compatible software programs can operate over all Modicon supported networks with no code modifications. A product operating with one of the MBX driver products, such as the MBX Driver, will operate with the rest of the MBX driver products as well.

Migration of existing installations to new hardware products does not require the user to discard working, proven software solutions. As depicted in the following diagram, a user can now mix Modbus, Modbus Plus and Ethernet based hardware products in existing installations without losing software, network or integration investment.



MBX enabled system deployment:
New hardware solutions will blend into existing installations without software or network modifications

THEORY OF OPERATION

This section will familiarize you with the main features of the Cyberlogic OPC Server as they relate to the MBX driver agent. Refer to the Cyberlogic OPC Server Help for a full discussion.

OPC is based on Microsoft's COM/DCOM architecture and allows client applications access to plant floor data in a consistent manner. The OPC specification defines a set of industry specific COM interfaces through which applications can read and write data to a variety of industrial devices.

The Cyberlogic OPC Server provides full compliance with the OPC Data Access 3.00, 2.05a and 1.0a specifications and the OPC Alarms and Events 1.1 specification.

Main Server Features

The Cyberlogic OPC Server supports multiple, priority-based access paths for redundant communications. Data items can be configured for both solicited and unsolicited data updates. The DirectAccess feature allows an OPC client to connect to a minimally configured Server and still access any registers in the device by using the register address. A sophisticated transaction optimizer guarantees minimum loading of the communication networks. The Server's advanced multithreaded design ensures minimum system loading and, at the same time, delivers unmatched data transfer speeds.

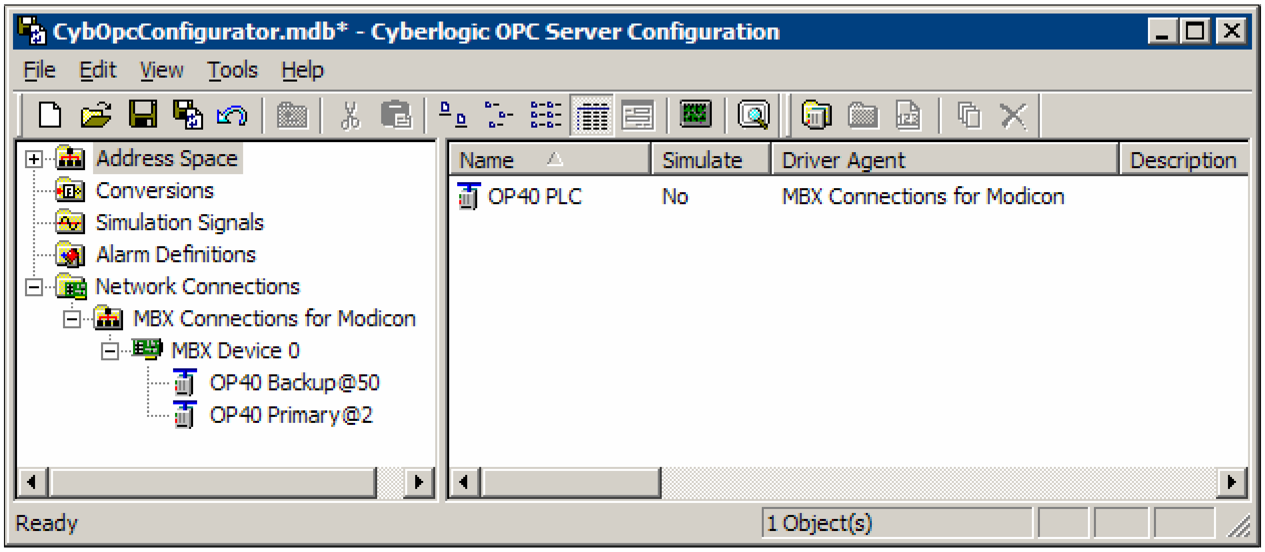
The Cyberlogic OPC Server Configuration Editor allows easy configuration of the Server's hierarchical tag database. It supports advanced features such as the automatic detection of network connections and nodes, configuration import/export and a built-in Data Monitor.

The following sections describe these operational features of the Server.

Network Connections and Network Nodes

The MBX driver agent module uses the Cyberlogic MBX drivers for its low-level communication services. Network Connections in the Cyberlogic OPC Server architecture correspond to MBX devices and contain all of the parameters for these devices. Each Network Connection may contain a number of Network Nodes. A Network Node represents a physical device on the network that is accessible through the corresponding Network Connection. The Network Node contains communication parameters for the physical node device.

In the figure below, the Network Connection MBX Device 0 has two Network Nodes, OP40 Backup@50 and OP40 Primary@2.



Name	Simulate	Driver Agent	Description
OP40 PLC	No	MBX Connections for Modicon	

The Cyberlogic OPC Server Configuration Editor can automatically create Network Connections for existing MBX devices. It also detects the nodes attached to those Network Connections and creates Network Nodes for them.

Caution:	The Auto Configuration feature detects Network Connections and Network Nodes associated with the MBX devices you have configured. Before you can use Auto Config, you must use the MBX configuration editor to create the devices that the OPC Server will use.

Health Watchdog

The Server can monitor the health of the connection to each physical device. If a device becomes inaccessible, the Server attempts to re-establish communications until the node can be marked as healthy again.

Address Space Tree

The Address Space Tree allows you to organize Data Items in a way that makes sense for your project. You can group and name related Data Items regardless of where they exist on the physical devices.

The branches of the tree are Device Folders, Devices and Folders. These establish how the Data Items are organized. The Data Items themselves are the “leaves” of the tree. You will begin construction of the tree at the Address Space root folder, which may contain Device Folders and Devices.

Device Folders

Device Folders logically group Devices and other Device Folders. You can place a Device Folder directly in the Address Space root folder or in another Device Folder, up to four levels deep.

Devices

Devices in the Address Space tree are associated with Network Nodes to which the Server communicates.

Note:	An Address Space Device should not be confused with a MBX device. An Address Space Device is tied to one or more Network Nodes. MBX devices such as network cards, on the other hand, are directly related to Network Connections.

You may place a Device directly in the Address Space root folder or in a Device Folder.

Access Paths

Access paths are required for solicited communications and represent a logical connection to a Network Node. These connections link the Data Items in an Address Space Device with their values in a physical device. They tell the Server where and how to obtain these values.

From the Access Paths Tab, you may set up more than one Access Path for a Device, allowing you to improve communication reliability by using redundant networks.

Unsolicited Message Filters

Unsolicited message filters are required for unsolicited communications. In an unsolicited update, the device decides when to send data to the Server, rather than waiting for the Server to request data. This helps to minimize the amount of traffic on the network.

Unsolicited messages must pass through user-defined filters that help ensure that unsolicited messages are accepted only from trusted sources. The unsolicited message filters are organized in groups. Each group has an equal priority and a message must pass through at least one of these groups in order to be accepted by the Server. You will configure the filters on the Unsolicited Message Filters Tab.

Folders

Folders logically group Data Items and other Folders. A Folder can be placed directly under a Device or under another Folder, up to four levels deep.

Data Items

Data Items represent registers in the physical device, ranges of registers, bits inside a register or ranges of bits. The MBX driver agent supports all Modicon register types — Coils (0x), Inputs (1x), Input Registers (3x), Holding Registers (4x) and General Reference Registers (6x) — including Global Data. For more information, refer to Appendix A: PLC Addresses. The user can individually configure each Data Item for solicited updates, unsolicited updates or both.

The MBX driver agent supports a number of integer, floating point and string data types. It also supports single-dimensional arrays of these types. The following table shows all supported simple data types.

Data Type	Data Size (in bits)	Default Canonical Data Type	Description
Default			Default type based on the Data Item address
BIT	1	VT_BOOL	1-bit boolean
SINT8	8	VT_I1	Signed 8-bit integer
UINT8	8	VT_UI1	Unsigned 8-bit integer
SINT16	16	VT_I2	Signed 16-bit integer
UINT16	16	VT_UI2	Unsigned 16-bit integer
SINT32	32	VT_I4	Signed 32-bit integer
UINT32	32	VT_UI4	Unsigned 32-bit integer
SINT64	64	VT_I8	Signed 64-bit integer
UINT64	64	VT_UI8	Unsigned 64-bit integer
FLOAT32	32	VT_R4	IEEE format 32-bit floating point number
FLOAT64	64	VT_R8	IEEE format 64-bit floating point number
BCD16	16	VT_UI2	BCD value in the range of 0 - 9999
BCD32	32	VT_UI4	BCD value in the range of 0 - 99999999
STRING	String size * 8	VT_BSTR	Zero terminated ASCII string of 8-bit characters
WSTRING	String size * 16	VT_BSTR	Zero terminated UNICODE string of 16-bit characters
FIELD	Field size	The best fitting VT_UIx or array of VT_UI1 if size > 64	Multiple bit field

For each simple data type, a user can select a specific VT_XXX type or choose the default type. When the default type is selected, the Server selects the appropriate canonical data type that can best store the selected data inside the Server.

Data Write Protection

In general, the MBX driver agent supports read and write operations to its Data Items. However, writing to some Data Items may create a safety hazard. Some registers, such as Inputs, are read-only and require no additional protection. For read/write registers, you can disable the write capability at any level. That is, you can disable writes for a:

- Data Item
- Folder
- Device
- Device Folder
- Network Node
- Network Connection
- Driver Agent

You can disable DirectAccess writes at each Network Node, Network Connection or driver agent.

Bit Order

The MBX OPC Server provides the ability to manipulate the order of the bits within Data Items to accommodate the different formats used by data sources and client software. The software can reverse the order of bits within each four-bit nibble, swap nibbles within each byte, swap bytes within each word, swap words within each double word and swap double words within each long word. Users can specify any combination of these as needed.

For maximum flexibility and ease of use, the user can specify a bit swap pattern for each Device, and can override the Device setting for each Data Item, as desired.

DirectAccess

At run time, in addition to the user-configured branches, the Cyberlogic OPC Server dynamically creates a branch called DirectAccess at the root of the Address Space. OPC clients can use this branch to access any register in any configured Network Node by directly specifying the register address.

The structure of this branch depends upon the driver agent and the configuration of the Network Connections. The DirectAccess branch acts like a Device Folder that contains all configured Network Connections. Each Network Connection branch contains its configured Network Nodes. However, only Network Nodes that enable DirectAccess are present.

Conversions

The raw data associated with a Data Item may represent a signal value from some instrument. In most cases, this value is not expressed in the engineering units of the measured signal. To simplify operating on the signal’s data, the Cyberlogic OPC Server can associate a Conversion with each Data Item.

A user can define a number of different Conversions. The Server can then apply each Conversion to a number of Data Items. As a result, a user need not define the same Conversion for each Data Item.

Simulation Signals

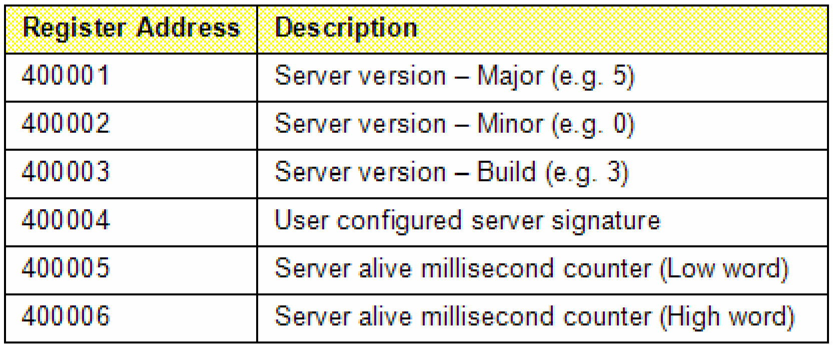
To facilitate client-side testing without the need for a physical device, a predefined formula can simulate the data for each Data Item. A user can define several different types of Simulation Signals. Each signal can then simulate a number of Data Items. As a result, a user need not define the same Simulation Signal for each Data Item.

Alarms and Events

The Cyberlogic OPC Server supports the OPC Alarms and Events specification. It allows a user to create a number of Alarm Definitions, each of which can then be used by a number of Data Items. As a result, a user need not define the same alarm condition for each Data Item. There are two categories of alarms: digital and limit (analog).

Server Status Block

The Cyberlogic OPC Server has a set of 16-bit registers called the Server Status Block. Physical devices on the network, such as PLCs, can read these registers to determine the current health and status of the Server. The Server Status Block is located in the 4xxxxx register range, starting at 400001:



Register Address	Description
400001	Server version – Major (e.g. 5)
400002	Server version – Minor (e.g. 0)
400003	Server version – Build (e.g. 3)
400004	User configured server signature
400005	Server alive millisecond counter (Low word)
400006	Server alive millisecond counter (High word)

On-Line Configuration Changes

You can make and apply configuration changes on-line, while OPC clients are connected to the server. A user has full control over when these changes are applied: once following several changes or after each configuration change. When the changes are applied, all connected clients are updated with the new configuration.

Undoing Configuration Changes

The Cyberlogic OPC Server Configuration Editor keeps track of recent configuration changes. Before saving the changes, a user can revert to the previously saved configuration.

Configuration Import/Export

To speed up the configuration of multiple similar servers, the Cyberlogic OPC Server Configuration Editor has an Import/Export capability. An entire database can be exported and then re-imported or just selected portions of it can be imported.

The utility can export to and import from many different file formats, such as imports of the configuration files of other OPC server vendors , including csv (comma separated values) files from Ingear OPC Server, Kepware Kepserver and Matrikon OPC Server. It can also import Concept text delimited variable files. This feature is explained in detail in the Cyberlogic OPC Server Help.

Data Monitor

The Cyberlogic OPC Server Configuration Editor includes a simple tag monitoring utility. This allows a user to quickly test changes to the Server's configuration. It can also aid in troubleshooting. This feature is explained in detail in the Cyberlogic OPC Server Help.

CONFIGURATION

You must properly configure the Cyberlogic OPC Server before you can use it. To do this, you must run the Cyberlogic OPC Server Configuration Editor after the installation.

This section will describe the features of the Configuration Editor that are specific to the MBX driver agent. Refer to the Cyberlogic OPC Server Help for a discussion of the editor's features that are common to all driver agents, such as alarms, simulations and conversions.

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

Typical Configuration Session

For a step-by-step guide through a typical configuration session, refer to the Cyberlogic OPC Server Help.

Creating Network Connections and Nodes

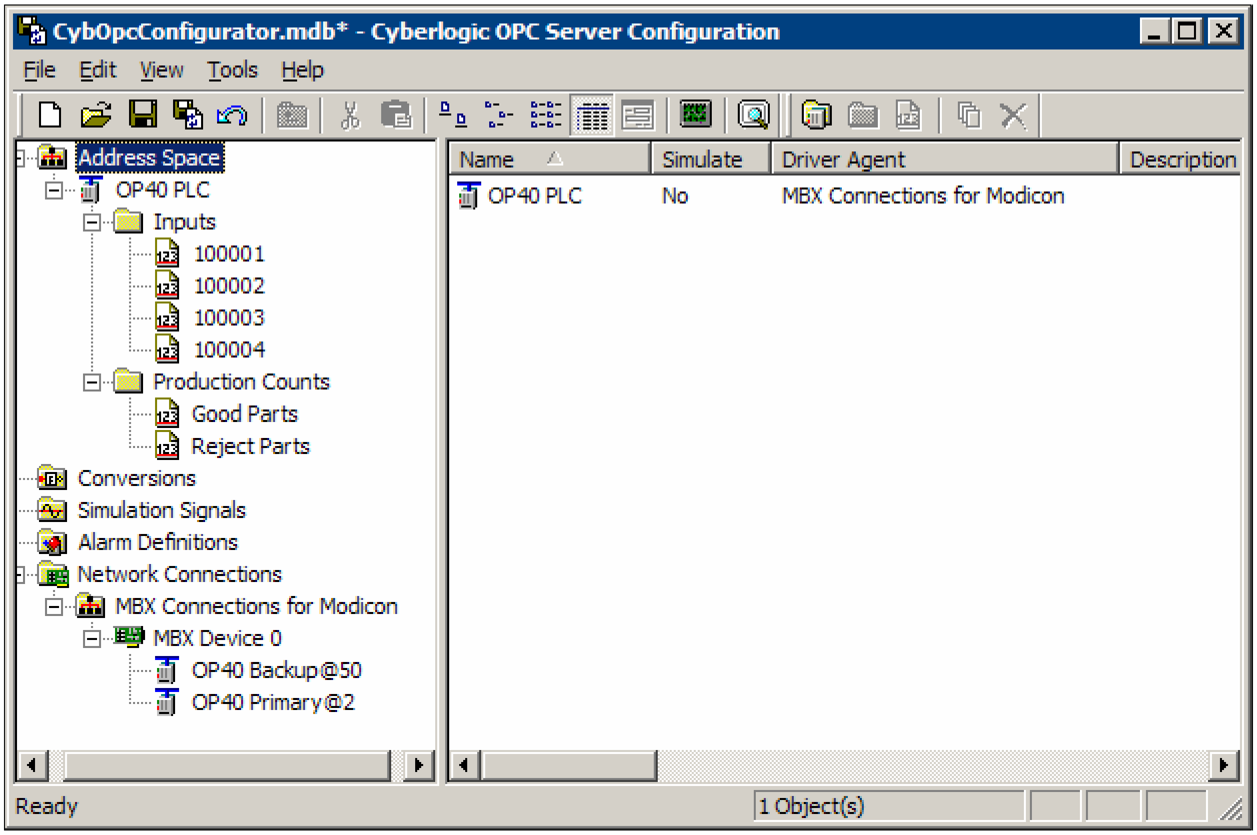
The procedure for creating network connections and nodes is similar for all Cyberlogic OPC Server driver agents, and can be found in the Cyberlogic OPC Server Help. Once you have created them, the next section will cover the details of editing the MBX Network Connections and Network Nodes.

OPC Server Configuration Editor

To start the editor, open the Windows Start menu, locate the MBX OPC Server submenu and select the *OPC Server Configuration* menu item.

The Cyberlogic OPC Server Configuration Editor allows the user to create and modify the configuration file used by the runtime module. It is needed only to generate configuration files and is not otherwise required for the operation of the runtime module.

An Explorer-like user interface allows easy manipulation of the configuration file and provides numerous common Windows functions, such as copy and paste, drag and drop, and context-based pop-up menus.



Name	Simulate	Driver Agent	Description
OP40 PLC	No	MBX Connections for Modicon	

The left pane of the main workspace window shows the five main configuration topics.

- Address Space Tree
- Conversions
- Simulation Signals
- Alarm Definitions
- Network Connections Tree

The following sections provide complete descriptions of these topics.

Network Connections Tree

The Cyberlogic OPC Server uses the industry-standard MBX drivers for its low-level communication services. All of these products implement the same MBX architecture and expose an identical programming interface, the MBXAPI. Although all MBX devices implement the same architecture, the underlying networks may have significantly different characteristics. Some networks can be fast, while others may be slow. The maximum number of nodes on the network or other resources may vary from network to network.

In the MBX architecture, MBX devices represent network connections. In some cases a MBX device corresponds to a physical network card, such as an SA-85. In other cases it is just an abstract object, such as an Ethernet MBX device, that behaves like a network card.

To optimize the network communication parameters, the Cyberlogic OPC Server introduces the concepts of Driver Agents, Network Connections and Network Nodes. The next sections describe configuration of these objects.

Driver Agents

There are various kinds of driver agents available for use with the Cyberlogic OPC Server. This help file deals only with the MBX driver agent.

Auto Configuration

The simplest method of configuration is Auto Configuration.

Caution:	The Auto Configuration feature detects Network Connections and Network Nodes associated with the MBX devices you have configured. Before you can use Auto Config, you must use the MBX configuration editor to create the devices that the OPC Server will use. For Ethernet networks, Auto Configuration will detect only the nodes that are mapped to Modbus Plus node addresses. Refer to the driver help file for information on how to configure MBX devices.

To begin this process, select the Network Connections root folder.

To find all Network Connections available on your system, open the Edit menu and select *Find Network Connections* (or right-click the Network Connections root folder and select *Find Network Connections* from the context menu).

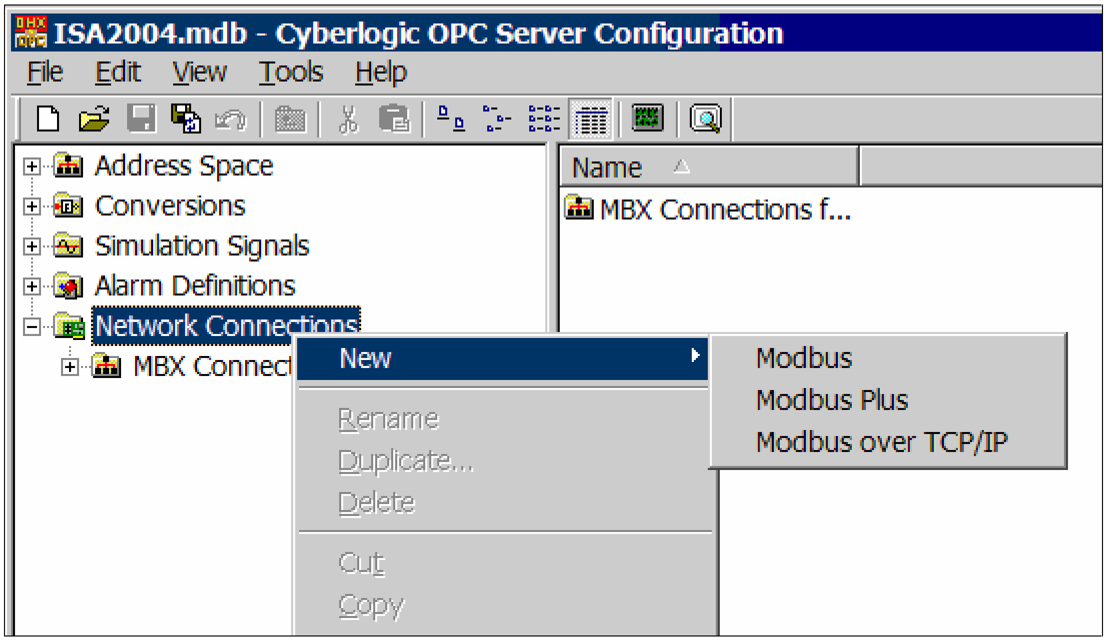
To find all Network Connections and automatically detect and configure Network Nodes, select *Auto Config* from the Edit menu (or right-click on the *Network Connections* root folder and select *Auto Config* from the context menu). If you manually configure a Network Connection, Auto Config will find the Network Nodes on that Network Connection.

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

Manual Configuration

You may prefer to configure your communications manually. This will be necessary if you are doing the configuration on a computer that is not connected to the target networks or if you wish to change the default values selected during an Auto Configuration.

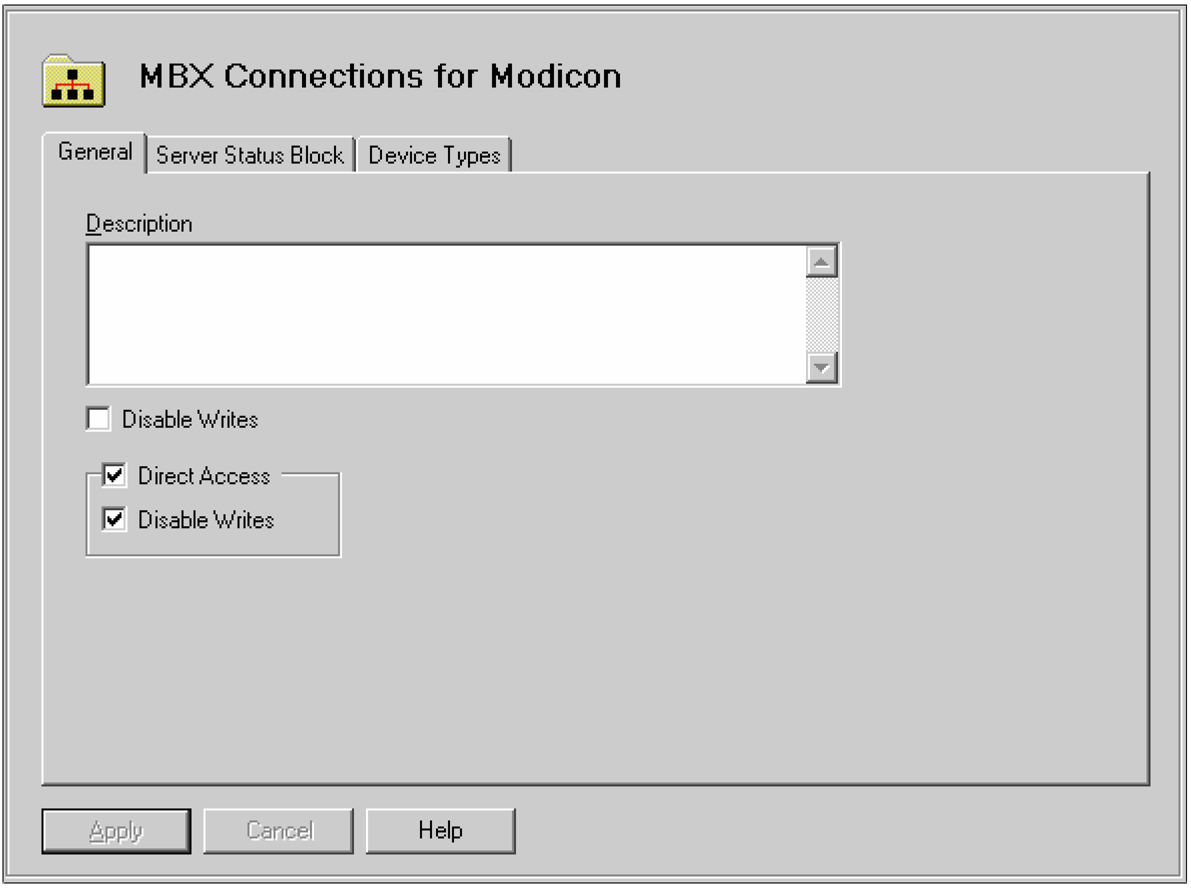
Select the *Network Connections* root folder, open the *Edit* menu item and select *New*, and then select the desired network type (or right-click on the Network Connections root folder and select *New*, then select the network type from the context menu). The editor will create the proper driver agent branch.



Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

The MBX Driver Agent has a configuration screen with three tabs, General, Server Status Block and Device Type.

General Tab


MBX Connections for Modicon

General | Server Status Block | Device Types

Description

☐ Disable Writes

☑ Direct Access

☑ Disable Writes

Description

This optional field can be used to describe the device. It can be up to 255 characters long.

Disable Writes

If this box is checked, the Server will not write data to any of the nodes that connect through this driver agent. The default state is unchecked, enabling writes.

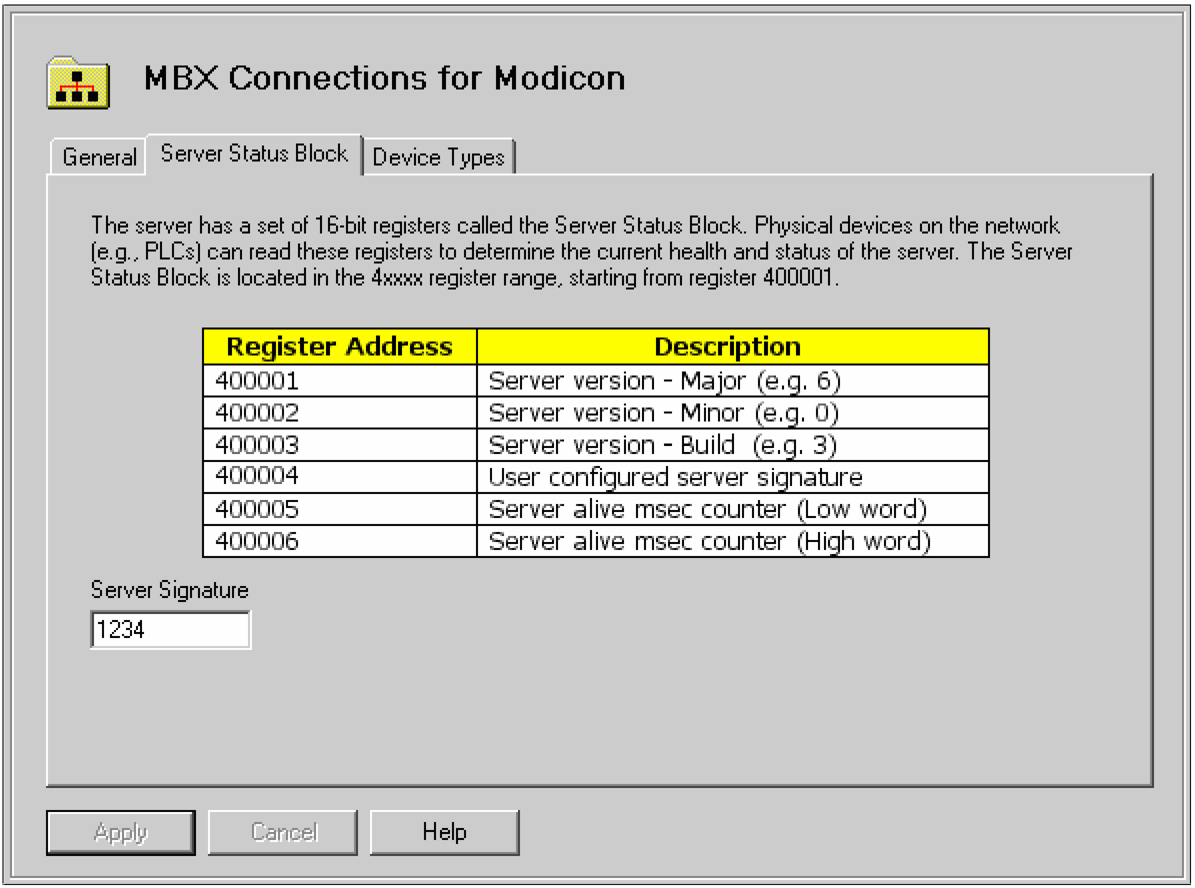
DirectAccess

If this box is checked, the user is permitted to configure DirectAccess to the nodes that connect through this driver agent. The default state is checked, allowing DirectAccess.

DirectAccess Disable Writes

If this box is checked, the Server will not write data via DirectAccess to any of the nodes that connect through this driver agent. This does not affect writes through configured Data Items. The default state is checked, disabling DirectAccess writes.

Server Status Block Tab



The server has a set of 16-bit registers called the Server Status Block. Physical devices on the network (e.g., PLCs) can read these registers to determine the current health and status of the server. The Server Status Block is located in the 4xxxx register range, starting from register 400001.

Register Address	Description
400001	Server version - Major (e.g. 6)
400002	Server version - Minor (e.g. 0)
400003	Server version - Build (e.g. 3)
400004	User configured server signature
400005	Server alive msec counter (Low word)
400006	Server alive msec counter (High word)

Server Signature: 1234

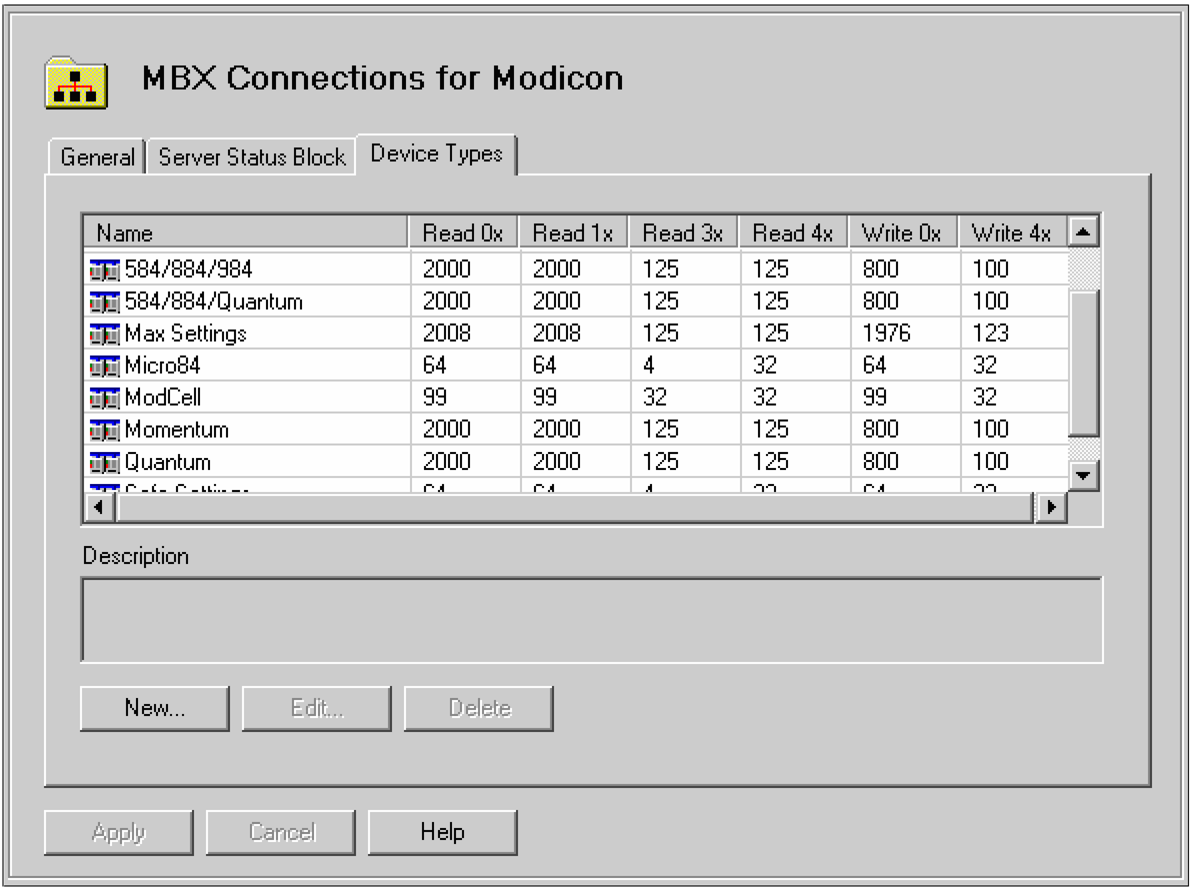
This tab shows the addresses for the Server Status Block associated with this driver agent. The Network Nodes associated with this device can read this block of registers to obtain information about the health and status of the Server.

Server Signature

Enter a number that will uniquely identify this Server. Devices that read this signature value will then be able to identify which Server they are communicating with. The default value is *1234*.

Device Types Tab

At run time, the MBX OPC Server tries to optimize its solicited communications based on the types of physical devices to which it communicates. It does this by adjusting the maximum message size for different types of registers. By default, the Server defines a number of standard network types. Users can create additional custom device types for physical devices that do not match one of the standard types.

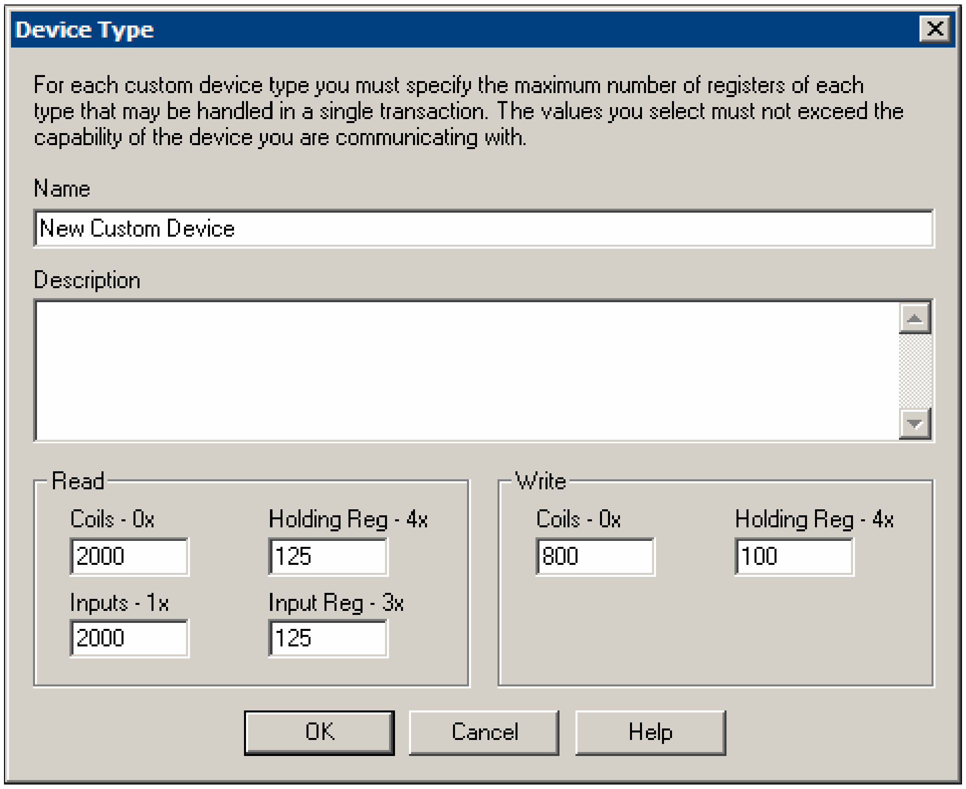


Name	Read 0x	Read 1x	Read 3x	Read 4x	Write 0x	Write 4x
584/884/984	2000	2000	125	125	800	100
584/884/Quantum	2000	2000	125	125	800	100
Max Settings	2008	2008	125	125	1976	123
Micro84	64	64	4	32	64	32
ModCell	99	99	32	32	99	32
Momentum	2000	2000	125	125	800	100
Quantum	2000	2000	125	125	800	100

Note:	The Safe Settings selection shows a set of conservative settings that will work with any device type, but will not provide particularly good performance.

Creating a New Device Type

Click the *New…* button. The following dialog box will open.



For each custom device type you must specify the maximum number of registers of each type that may be handled in a single transaction. The values you select must not exceed the capability of the device you are communicating with.

Name: New Custom Device

Description:

Read

Coils - 0x: 2000

Holding Reg - 4x: 125

Inputs - 1x: 2000

Input Reg - 3x: 125

Write

Coils - 0x: 800

Holding Reg - 4x: 100

Enter the desired maximum message sizes in the six Read and Write fields and then enter a name for the Device Type. Optionally, you may enter a description. Click *OK* when you are done.

Network Connections

A Network Connection corresponds to an MBX device and contains all of the parameters that are common to that device. Each Network Connection may contain a number of Network Nodes.

Auto Configuring Network Connections

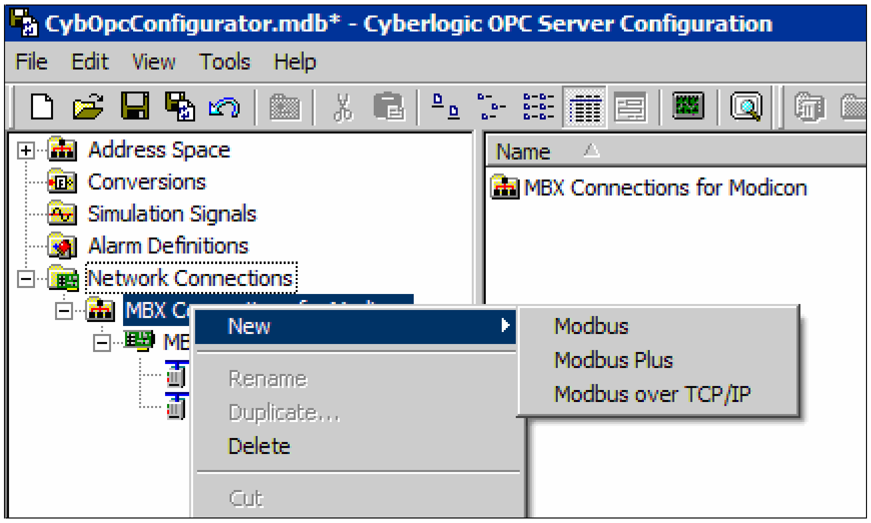
The simplest way to configure the Network Connections for your system is to use the Auto Configuration capability discussed in the Driver Agents section.

Deleting a Network Connection

Select an existing network connection and press the *Delete* key (or right-click on the existing network connection and select *Delete* from the context menu).

Creating a New Network Connection

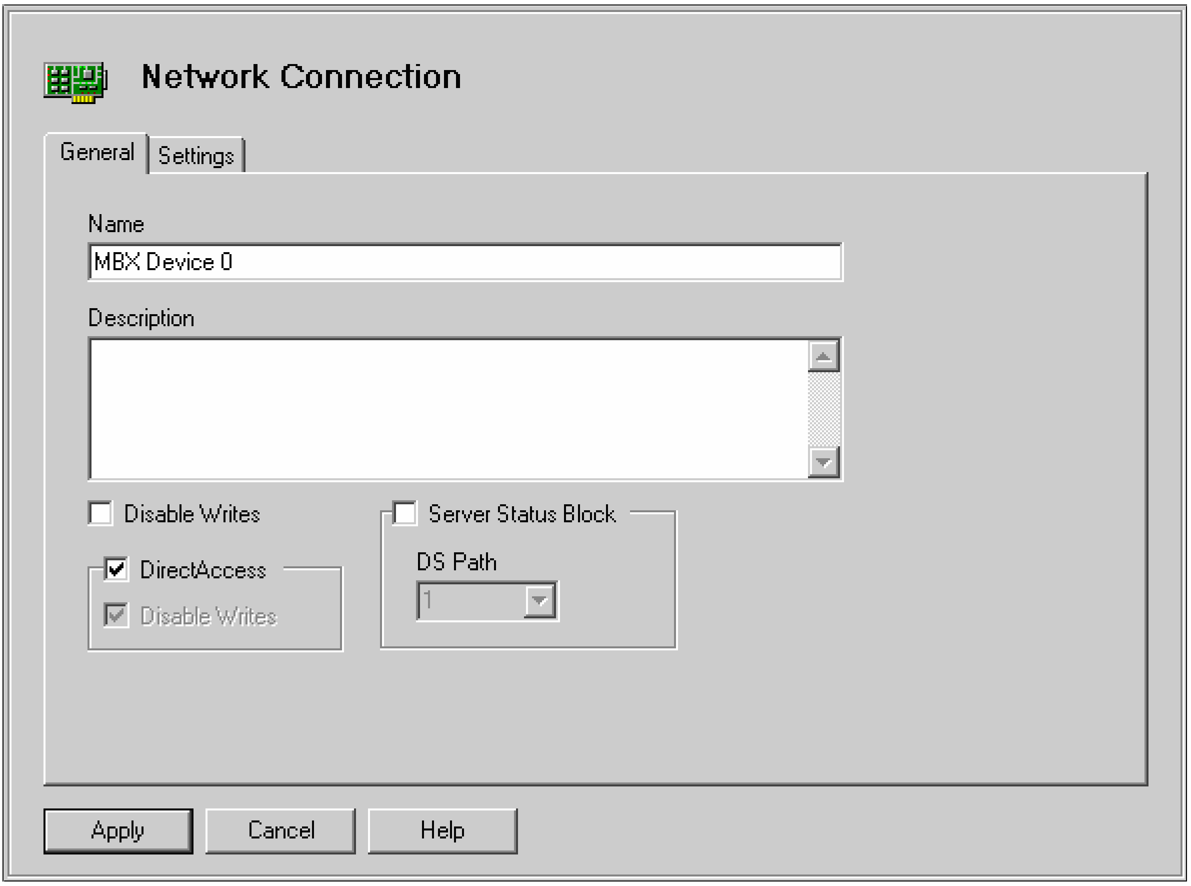
Select the *MBX Connections for Modicon* folder and, from the Edit menu, select *New*, then select the type of network connection you wish to create. (You can instead right-click on the *MBX Connections for Modicon* folder and select *New* then the type of network connection from the context menu). Enter a unique name and set all of the relevant parameters. Click *Apply* when you are done.



Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

For each Network Connection, the editor will provide two tabs containing fields to be edited, the General tab and the Settings tab.

General Tab



Name: MBX Device 0
Description:
☐ Disable Writes
DirectAccess: ✓
Disable Writes: ✓
☐ Server Status Block
DS Path: 1

Name

The name identifies this Network Connection. It can be up to 50 characters long, may contain spaces, but must not begin with a space. It also must not contain any periods.

Description

This optional field further describes the network connection. It can be up to 255 characters long.

Disable Writes

If this box is checked, the Server will not write data to any of the nodes on this Network Connection. The default state is unchecked, enabling writes.

Note:	If the Disable Writes checkbox is grayed-out, it indicates that writes have already been disabled at a higher level.

DirectAccess

If this box is checked, the user is permitted to configure DirectAccess to the nodes on this Network Connection. The default state is checked, allowing DirectAccess.

Note:	If the DirectAccess checkbox is grayed-out, it indicates that DirectAccess has already been disabled at a higher level.

DirectAccess Disable Writes

If this box is checked, the Server will not write data via DirectAccess to any of the nodes on this Network Connection. This does not affect writes through configured Data Items. The default state is checked, disabling DirectAccess writes.

Note:	If the DirectAccess Disable Writes checkbox is grayed-out, it indicates that DirectAccess writes have already been disabled at a higher level.

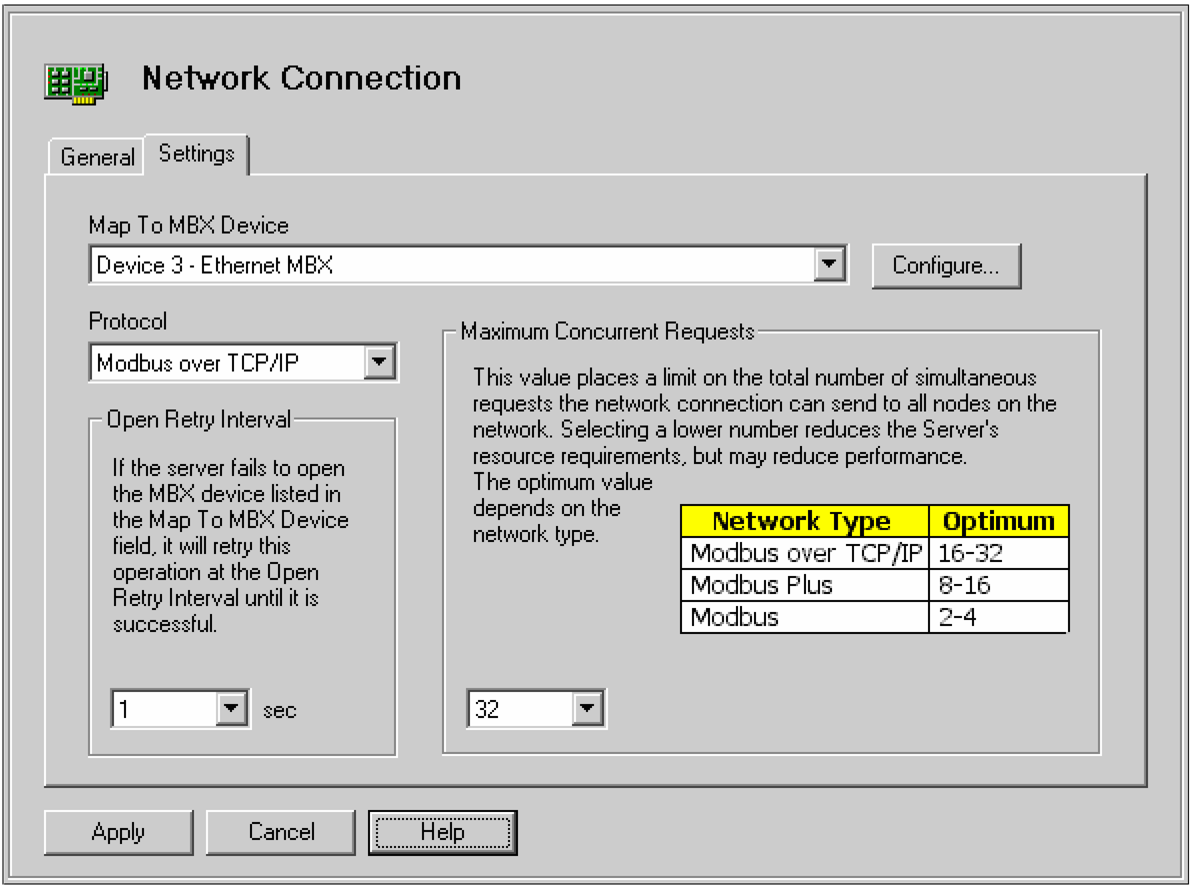
Enable Server Status Block

If this box is checked, the network nodes under this Network Connection will be able to read the Server Status Block values. The default state is unchecked, disabling access to the Server Status Block.

DS Path

This selection specifies the Data Slave path on which the Server will listen for Server Status Block read requests. The default path is *1*.

Settings Tab



Map To MBX Device: Device 3 - Ethernet MBX

Protocol: Modbus over TCP/IP

Open Retry Interval

If the server fails to open the MBX device listed in the Map To MBX Device field, it will retry this operation at the Open Retry Interval until it is successful.

1 sec

Maximum Concurrent Requests

This value places a limit on the total number of simultaneous requests the network connection can send to all nodes on the network. Selecting a lower number reduces the Server's resource requirements, but may reduce performance. The optimum value depends on the network type.

Network Type	Optimum
Modbus over TCP/IP	16-32
Modbus Plus	8-16
Modbus	2-4

32

Map To MBX Device

Click the drop-down button to select the MBX device this Network Connection will use to communicate. Clicking the *Configure…* button runs the MBX Driver Configuration Editor, allowing you to modify the configuration of the MBX devices. For more information on configuring MBX devices, refer to the help files for each of the MBX driver products.

Protocol

Click the drop-down button to select the specific protocol used by this network.

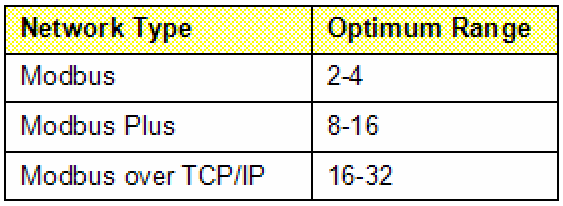
Open Retry Interval

The Server will try to open the MBX device shown in the Map To MBX Device field. If it fails, it will retry repeatedly until it succeeds. This field specifies the interval at which the Server will attempt these retries. This interval is also used when opening DS paths used for unsolicited communications. The valid range for the Open Retry Interval is 1-60 seconds.

Max Concurrent Requests

This value defines the maximum number of simultaneous transactions that the Server will allow to handle the nodes on this network connection. Lower numbers reduce the Server's resource requirements, but may reduce performance.

The optimum value to use depends upon the network type. Use the following table as a guideline when selecting this value.



Network Type	Optimum Range
Modbus	2-4
Modbus Plus	8-16
Modbus over TCP/IP	16-32

Caution:	Selecting values above the recommended range consumes more system resources but typically does not improve the performance of the Server.

Network Nodes

A Network Node represents a physical device on the network that is accessible through one of the Network Connections. An example would be a 984 that is accessible through a Modbus Plus connection. The Network Node contains communication parameters for the physical node device. A user can define multiple Network Connections, each having multiple Network Nodes.

It is important to understand the difference between Network Nodes and Access Paths. A Network Node is a physical device that provides or accepts data over a network. An Access Path is a logical association between an Address Space Data Item (such as an input, output or register) and an input, output or register on a Network Node. You will learn more about Access Paths and Data Items in the Address Space Tree section.

For now, you need only know that multiple Access Paths can reference the same Network Node. This greatly simplifies the configuration process. You will not need to specify the network parameters when configuring each Access Path. Instead, you will specify these parameters only once, when you configure the Network Node. You may then reference that node in any number of Access Paths. Access Paths that reference a Network Node are automatically updated when changes are made to that node.

Auto Configuring Network Nodes

The simplest way to configure the Network Nodes for your system is to use the auto configuration capability discussed in the Driver Agents section.

Deleting a Network Node

Select an existing Network Node and press the *Delete* key (or right-click on the Network Node and select *Delete* from the context menu).

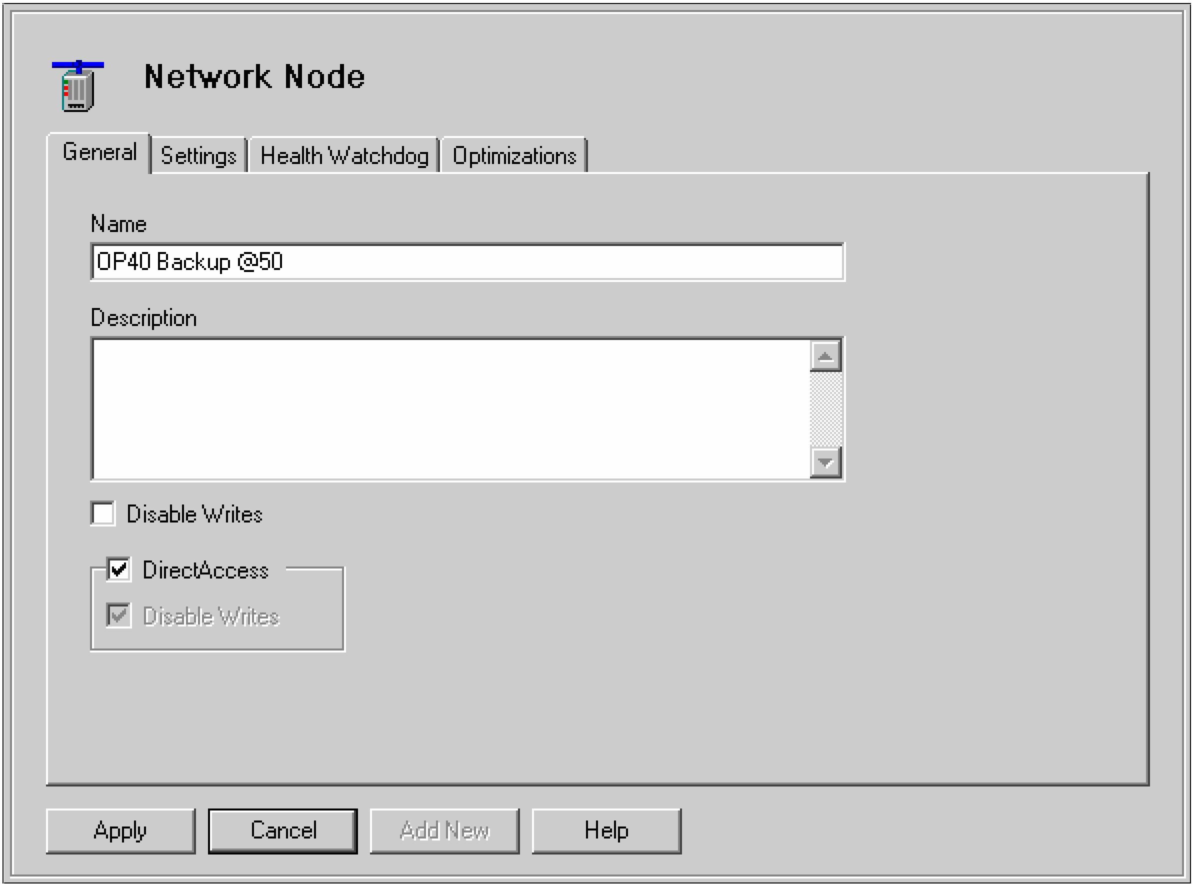
Creating a New Network Node

Select an existing Network Connection and select *New/Network Node* from the Edit menu (or right-click on the Network Connection and select *New/Network Node* from the context menu). Enter a unique name and set all relevant parameters. Click *Apply* when you are done.

For each Network Connection, the editor will provide four tabs containing fields to be edited: the General, Settings, Health Watchdog and Optimizations tabs.

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

General Tab



Name
OP40 Backup @50

Description

☐ Disable Writes

☑ DirectAccess
☑ Disable Writes

Name

The name identifies the Network Node. It can be up to 50 characters long, may contain spaces, but must not begin with a space. It also may not contain any periods.

Description

This optional field further describes the network node. It can be up to 255 characters long.

Disable Writes

If this box is checked, the Server will not write data to this node. The default state is unchecked, enabling writes.

Note:	If the Disable Writes checkbox is grayed-out, it indicates that writes have already been disabled at a higher level.

DirectAccess

If this box is checked, the user is permitted to configure DirectAccess to this node. The default state is checked, allowing DirectAccess.

Note:	If the DirectAccess checkbox is grayed-out, it indicates that DirectAccess has already been disabled at a higher level.

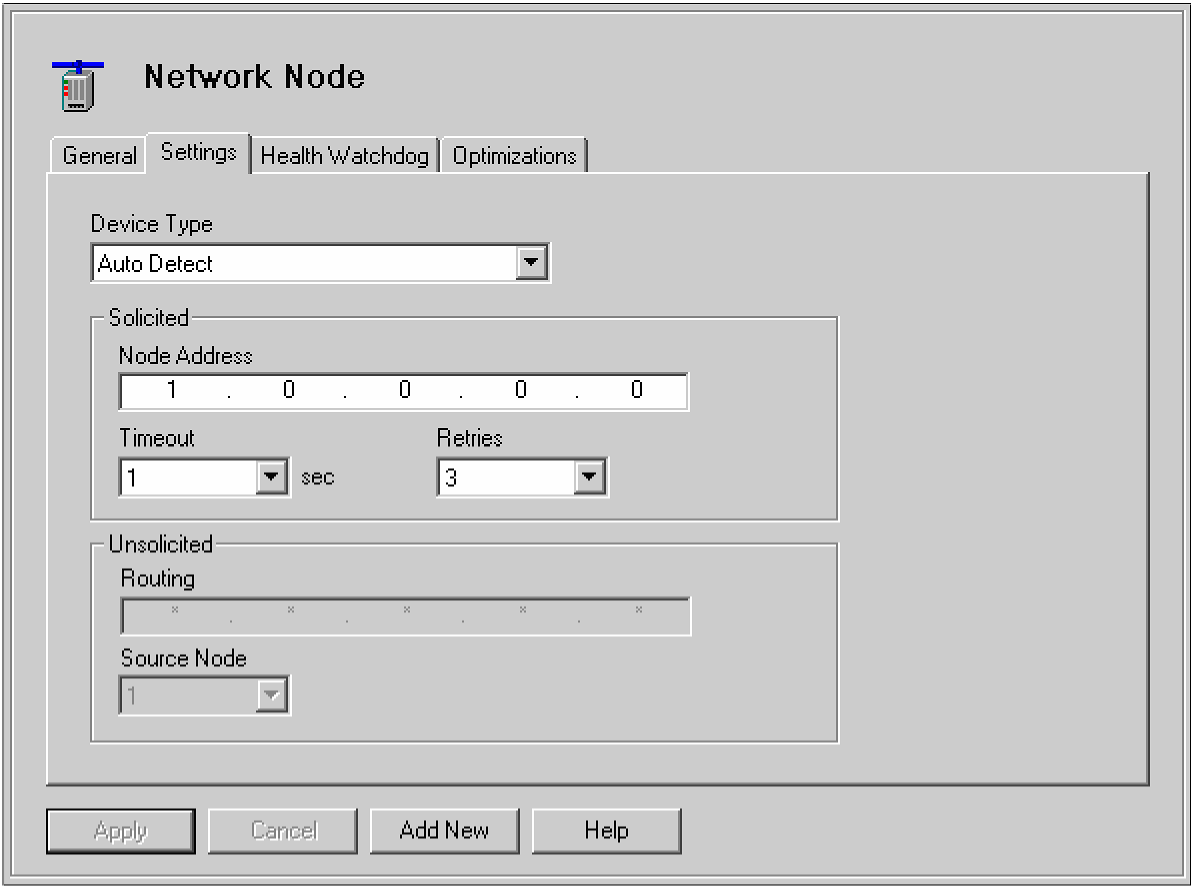
DirectAccess Disable Writes

If this box is checked, the Server will not write data via DirectAccess to this node. This does not affect writes through configured Data Items. The default state is checked, disabling DirectAccess writes.

Note:	If the DirectAccess Disable Writes checkbox is grayed-out, it indicates that DirectAccess writes have already been disabled at a higher level.

Settings Tab (Modbus Plus Networks)

The Settings tab layout is specific to the network type. This is the layout and description when a Modbus Plus network is configured. For other network types, refer to the Settings Tab (Ethernet Networks) and Settings Tab (Modbus Networks) sections.



Device Type: Auto Detect

Solicited
Node Address: 1 . 0 . 0 . 0 . 0
Timeout: 1 sec
Retries: 3

Unsolicited
Routing: × . × . × . × . ×
Source Node: 1

Device Type

This field identifies the device type of this Network Node. You may select from the device types you configured on the Network Connections Device Types Tab. The Server uses this information to optimize network communications.

Caution:	If you select *Auto Detect,* the Server uses FNC 17 (11 hex) *Report Slave ID* to determine the device type. All Modicon controllers support this function, but some third-party devices may not. For those devices, the Auto Detect will fail and the Server will default to the *Safe Settings* optimization. Because this setting may not provide the best performance, you should choose a different *Device Type* selection for these devices.

If you use global data and you want to read or write the OPC Server's global data, you must create a node of type *This Node – Global Data*. This will then allow you to create global data type Data Items in the Address Space Tree and connect them to the local node. For more information on global data, refer to Appendix A: PLC Addresses.

Solicited Node Address

This is the node's network address in the form of a standard five-field Modbus Plus routing array. For more information on node addressing, refer to the MBX Driver help file.

Solicited Timeout

This is the amount of time that the Server will wait to receive a reply to a command message. If the Server does not receive a reply within that interval, it cancels the transaction and marks it as timed out. This interval is specified in seconds.

Solicited Retries

This is the number of times the Server will reattempt each transaction that fails. The Health Watchdog uses this value to determine when to mark the node as unhealthy.

Unsolicited Section

For security and safety reasons, you may want to configure your system to permit unsolicited updates only from trusted sources. To do this, you will set up address filters when you configure the Address Space Devices. (Refer to the Unsolicited Message Filters Tab section for more details.) The Unsolicited section allows you to provide the Server with information that will identify this node for filtering purposes.

To identify the source of an unsolicited message, the Server uses information contained in a five-byte Modbus Plus routing array and a single byte source address. These are appended to each message the Server receives. If the sending node is located on another Modbus Plus network accessed through one or more bridges, it may not be possible to exactly identify that node. In that case, you can configure the Unsolicited Routing field and the Unsolicited Source Node field to at least narrow down the group of nodes that are permitted to provide unsolicited updates.

Note:	If the sending node is located on the local Modbus Plus network, the Unsolicited section will be configured for you automatically and will not be available for editing. The Routing field will be set to accept all routing array values (*.*.*.*.*) and the Unsolicited Source Node filed will be set to the value of the first byte in the Solicited section's Node Address field.

Unsolicited Routing

Here you will specify, as nearly as possible, the five-byte routing path that will be provided to the Server for unsolicited messages coming from this node. Each field may contain a number or an asterisk. If it contains a number, the incoming message must have that value in that position. If it contains an asterisk, any value is acceptable.

Each message received by the Server from a Modbus Plus node contains a five-byte routing array as follows.

Byte 1 Server Node Address	Byte 2 Server Slave Path	Byte 3 Normally unused	Byte 4 Normally unused	Byte 5 Normally unused
1 ... 64	1 ... 8	0 ... 255	0 ... 255	0 ... 255

- **Server Node Address.** The first byte specifies the node address of the Server on the Modbus Plus network.
- **Server Slave Path.** The second byte specifies the slave path that the Server will use to process the message.
- **Normally Unused Bytes.** The third through fifth bytes are normally unused, with values of 0. However, if the sending node does not need all five routing bytes to specify the routing of the message, the extra bytes can be used to provide additional information to identify the source of the message.

For example, suppose the OPC Server has Modbus Plus node address 17 and we want to specify slave path 1. The sending node is on another Modbus Plus network and must route the message through a bridge at node address 6. The routing it would use would be 6.17.1.0.0. After passing through the bridge, the routing would shift left one place, to become 17.1.0.0.0.

The problem is that this provides no information to help identify the source node. Every unsolicited message the Server receives would have 17 in the first byte and a slave path number in the second byte, followed by three zeros. You could, however, have the sending node put a value – let's say 20 – in the first unused byte, to identify itself. The resulting routing would be 6.17.1.20.0 and would route correctly. The routing information received by the Server would then be 17.1.20.0.0, giving it some additional information to work with. You could then set the Unsolicited Routing field to 17.1.20.0.0, and it would pass only those messages that originated from this node. Actually, since the value in the third byte is all that is needed to identify the sending node, it might be clearer to set the Unsolicited Routing to *.*.20.*.*. By using a different identifier byte value for each sending node, you could positively identify the sender.

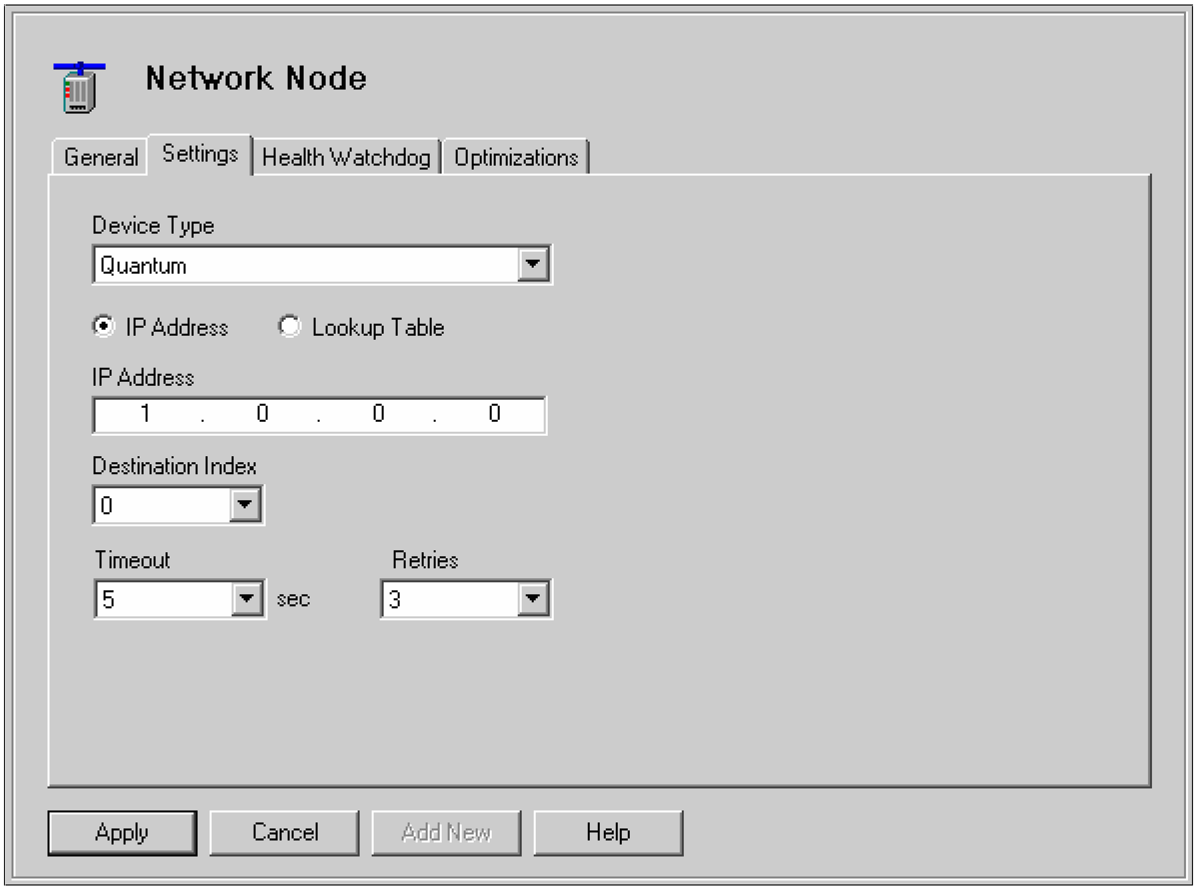
Unsolicited Source Node

Enter an address that will be reported to the Server as the source address of the unsolicited messages.

The reported address will be the node address of the source device or bridge on the Server's local network that passed the message to the Server. If the originating device is on the Server's local network, the reported node will be the node address of that device. If the originating device is on another network, the reported node will be the node address of the bridge on the Server's local network that routed the message to the Server.

Settings Tab (Ethernet Networks)

The Settings tab layout is specific to the network type. This is the layout and description when an Ethernet (Modbus TCP/IP) network is configured. For other network types, refer to the Settings Tab (Modbus Plus Networks) and Settings Tab (Modbus Networks) sections.



Device Type: Quantum

[✓ IP Address] Lookup Table

IP Address: 1 . 0 . 0 . 0

Destination Index: 0

Timeout: 5 sec

Retries: 3

Device Type

This field identifies the device type of this Network Node. You may select from the device types you configured on the Network Connections Device Types Tab. The Server uses this information to optimize network communications.

Caution:	If you select *Auto Detect,* the Server uses FNC 17 (11 hex) *Report Slave ID* to determine the device type. All Modicon controllers support this function, but some third-party devices may not. For those devices, the Auto Detect will fail and the Server will default to the *Safe Settings* optimization. Because this setting may not provide the best performance, you should choose a different *Device Type* selection for these devices.

If you use global data and you want to read or write the OPC Server's global data, you must create a node of type *This Node – Global Data*. This will then allow you to create global data type Data Items in the Address Space Tree and connect them to the local node. For more information on global data, refer to Appendix A: PLC Addresses.

IP Address / Lookup Table

You may specify the desired node by entering its IP address or its entry in the Lookup Table. Select which you prefer here.

Lookup Table Index

This field is available only if you chose the *Lookup Table* selection, and therefore is not shown in the above illustration. Select the MB+ Node value from the Ethernet MBX Driver's lookup table.

To view the lookup table, edit the Network Connection that this node uses and go to its Settings Tab. Click the *Configure...* button to open the MBX Driver Configuration Editor. Select the device that the Network Connection uses and click the *Edit* button. The lookup table is on the Master Path tab.

IP Address

This field is available only if you chose the *IP Address* selection. Enter the node's network address in the form of a standard IP address.

Destination Index

This field is available only if you chose the *IP Address* selection. Select the value you wish to use for the destination index.

Timeout

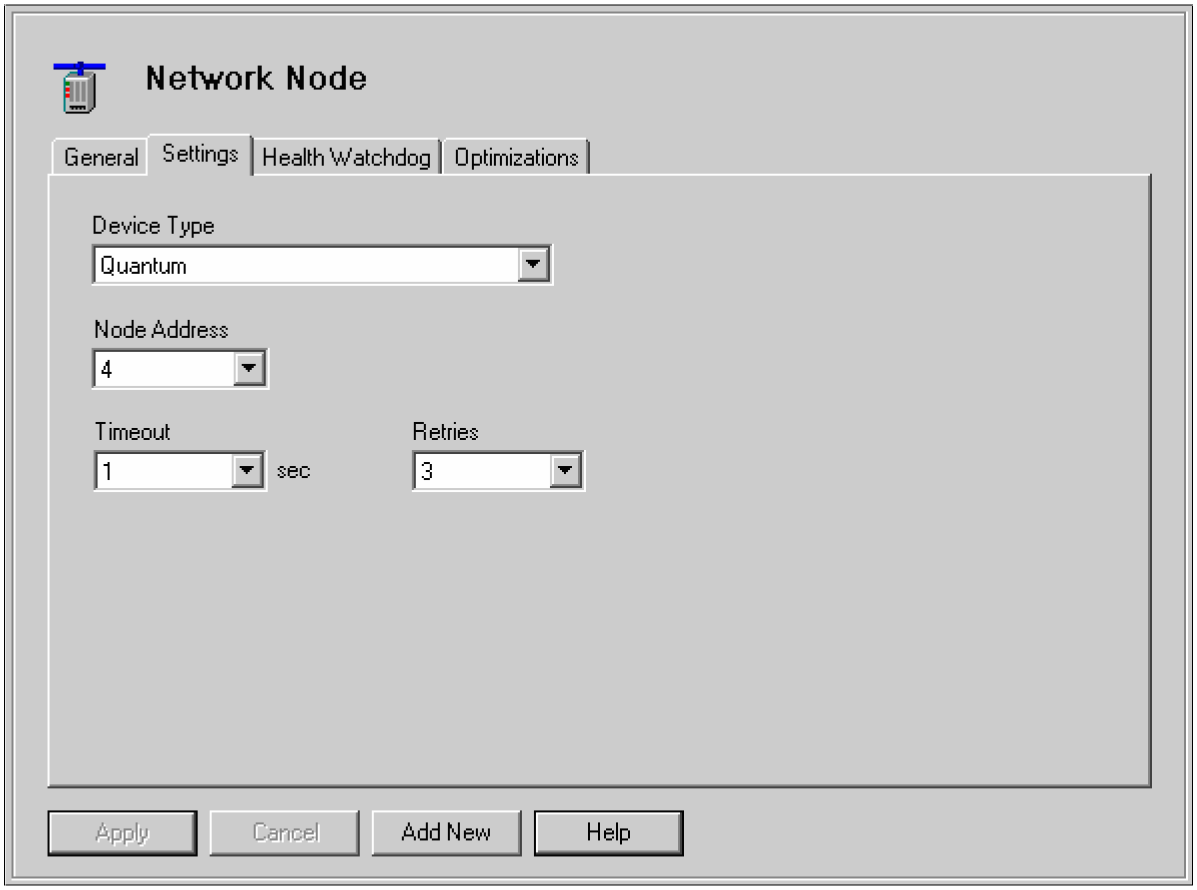
This is the amount of time that the Server will wait to receive a reply to a command message. If the Server does not receive a reply within that interval, it cancels the transaction and marks it as timed out. This interval is specified in seconds.

Retries

This is the number of times the Server will reattempt each transaction that fails. The Health Watchdog uses this value to determine when to mark the node as unhealthy.

Settings Tab (Modbus Networks)

The Settings tab layout is specific to the network type. This is the layout and description when a Modbus network is configured. For other network types, refer to the Settings Tab (Modbus Plus Networks) and Settings Tab (Ethernet Networks) sections.



Device Type: Quantum

Node Address: 4

Timeout: 1 sec

Retries: 3

Device Type

This field identifies the device type of this Network Node. You may select from the device types you configured on the Network Connections Device Types Tab. The Server uses this information to optimize network communications.

Caution:	If you select *Auto Detect,* the Server uses FNC 17 (11 hex) *Report Slave ID* to determine the device type. All Modicon controllers support this function, but some third-party devices may not. For those devices, the Auto Detect will fail and the Server will default to the *Safe Settings* optimization. Because this setting may not provide the best performance, you should choose a different *Device Type* selection for these devices.

If you use global data and you want to read or write the OPC Server’s global data, you must create a node of type *This Node – Global Data*. This will then allow you to create global data type Data Items in the Address Space Tree and connect them to the local node. For more information on global data, refer to Appendix A: PLC Addresses.

Node Address

This is the node's Modbus node number. For more information on node addressing, refer to the Serial MBX Driver help file.

Timeout

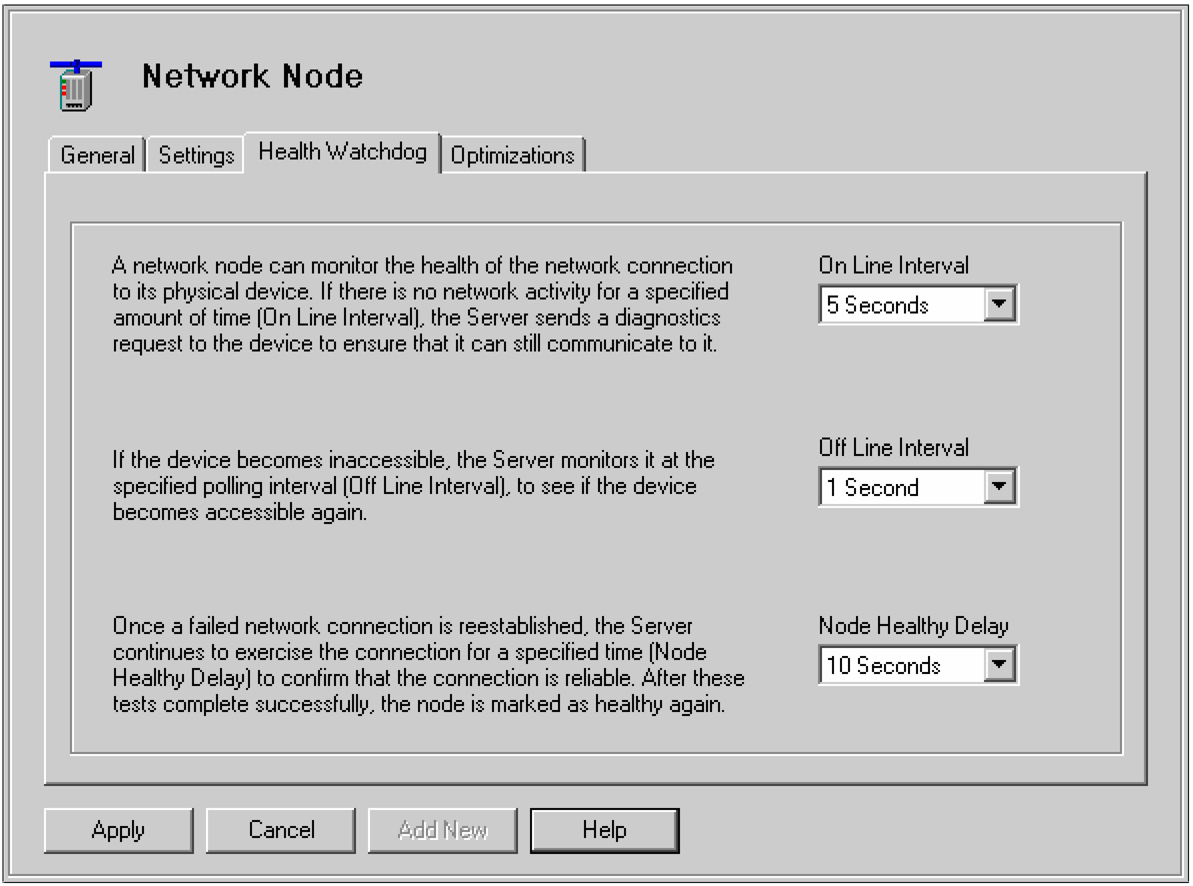
This is the amount of time that the Server will wait to receive a reply to a command message. If the Server does not receive a reply within that interval, it cancels the transaction and marks it as timed out. This interval is specified in seconds.

Retries

This is the number of times the Server will reattempt each transaction that fails. The Health Watchdog uses this value to determine when to mark the node as unhealthy.

Health Watchdog Tab

Each Network Node monitors the health of the connection to its physical device. If there is no communication for a long time, the Server sends a diagnostic request to the device to see if it can still communicate. If it cannot, the Server will re-check the connection until communication is reestablished. Once a failed network connection is reestablished, the Server continues to exercise the connection until it is satisfied that the connection is reliable. After this, the node is marked as healthy again.



A network node can monitor the health of the network connection to its physical device. If there is no network activity for a specified amount of time (On Line Interval), the Server sends a diagnostics request to the device to ensure that it can still communicate to it.

On Line Interval: 5 Seconds

If the device becomes inaccessible, the Server monitors it at the specified polling interval (Off Line Interval), to see if the device becomes accessible again.

Off Line Interval: 1 Second

Once a failed network connection is reestablished, the Server continues to exercise the connection for a specified time (Node Healthy Delay) to confirm that the connection is reliable. After these tests complete successfully, the node is marked as healthy again.

Node Healthy Delay: 10 Seconds

On Line Interval

If there is no traffic to a healthy node for this length of time, the Server will send a status request to the node to verify that it is still online. The On Line Interval is a value in the range of 1-60 seconds or None. Selecting *None* disables the on-line health monitoring.

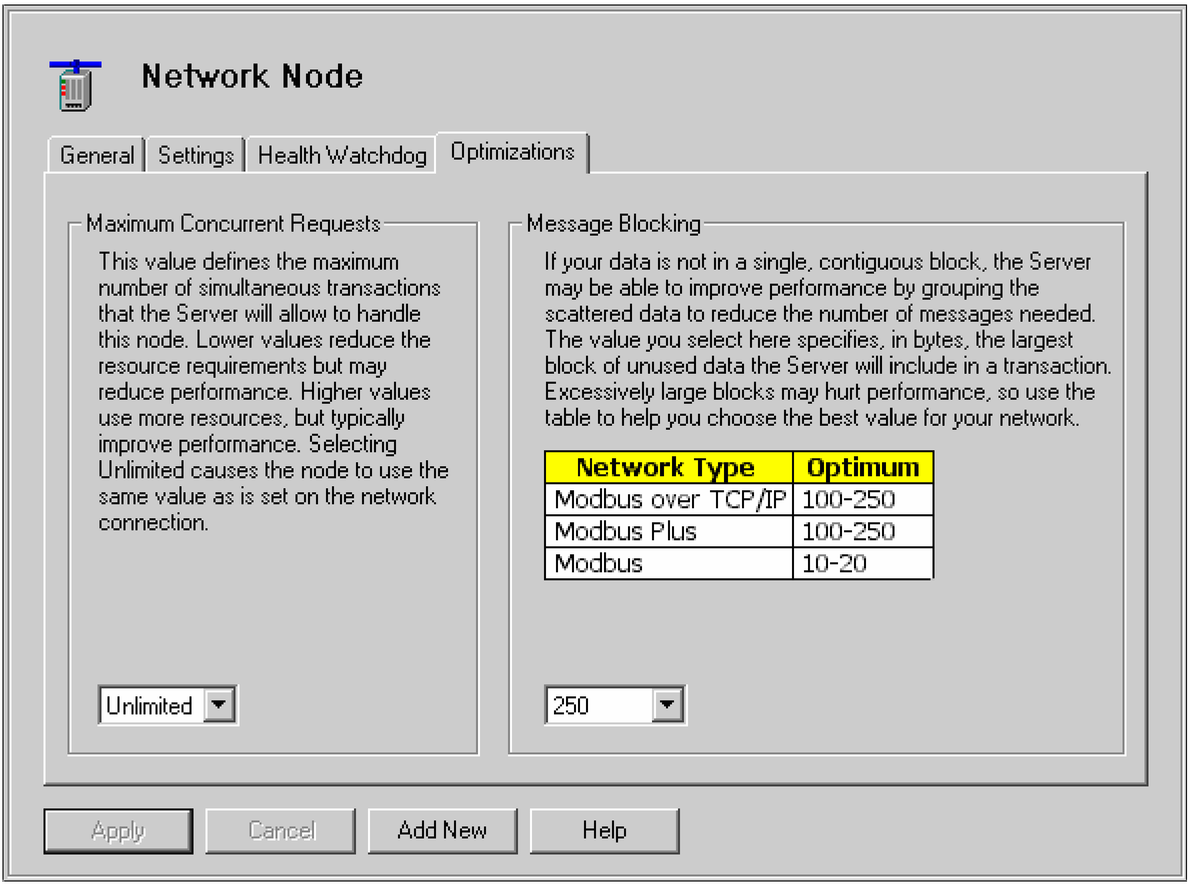
Off Line Interval

Once communication to a node has failed, the Server will send status requests to the node at this interval to determine whether it can communicate. The valid range for the Off Line Interval is 1-60 seconds.

Node Healthy Delay

Once the Server reestablishes communication to an unhealthy node, it waits for the communication to stay active for this length of time before considering the node healthy. The Server then returns the node to service. The valid range for the Node Healthy Delay is 1-60 seconds.

Optimizations Tab



Maximum Concurrent Requests

This value defines the maximum number of simultaneous transactions that the Server will allow to handle this node. Lower values reduce the resource requirements but may reduce performance. Higher values use more resources, but typically improve performance. Selecting Unlimited causes the node to use the same value as is set on the network connection.

Unlimited

Message Blocking

If your data is not in a single, contiguous block, the Server may be able to improve performance by grouping the scattered data to reduce the number of messages needed. The value you select here specifies, in bytes, the largest block of unused data the Server will include in a transaction. Excessively large blocks may hurt performance, so use the table to help you choose the best value for your network.

Network Type	Optimum
Modbus over TCP/IP	100-250
Modbus Plus	100-250
Modbus	10-20

250

Maximum Concurrent Requests

This value defines the maximum number of simultaneous transactions that the Server will allow to handle this node. Lower values reduce the resource requirements but may reduce performance. Higher values use more resources, but typically improve performance. Selecting *Unlimited* causes the node to use the same value as is set on the network connection.

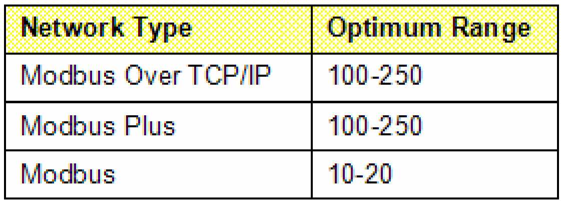
Note:	Setting this value to a lower number may prevent overloading the PLC with messages and allow for better operation of other applications, such as PLC programming software, that access the same PLC.

Caution:	Remember that the maximum number of simultaneous transactions for the entire network is still limited by the Maximum Concurrent Requests setting for the network connection.

Message Blocking

Typically, the data you will want to access will not be in a single, contiguous block. For the Server to access only the information you really require, it may need to make many small transactions. In that case, the Server may obtain better performance by grouping the data into fewer, larger blocks. Although this will mean that the system will transfer extra information, the overall throughput will be improved because of the reduced overhead.

This field allows you to tell the Server how to group the scattered data. The value you enter specifies, in bytes, the largest block of extra data that the Server will include in a transaction. The optimum value to use will depend upon the type of network. Faster networks can handle larger gaps; slower networks will get better performance with smaller gaps.



Network Type	Optimum Range
Modbus Over TCP/IP	100-250
Modbus Plus	100-250
Modbus	10-20

Address Space Tree

The Address Space Tree describes the hierarchical address structure of the Cyberlogic OPC Server. The “branches” of the tree are Device Folders, Devices and Folders. Its “leaves” are Data Items. The intent of this structure is to permit the user to organize the Data Items into logical groups.

Device Folders

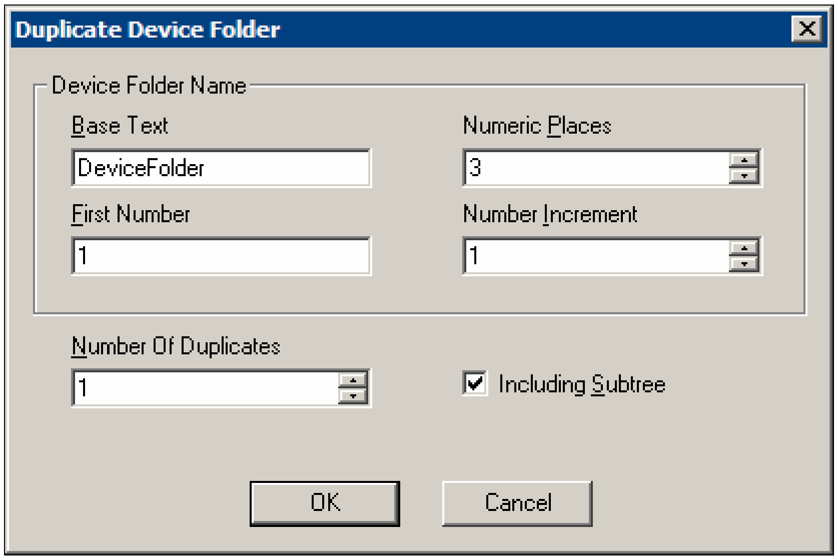
A Device Folder groups Devices and other Device Folders. You can place a Device Folder directly under the Address Space root folder or under another Device Folder, up to four levels deep.

Creating a New Device Folder

Select either the Address Space root folder or an existing Device Folder. From the Edit menu, select *New/Device Folder* (or right-click on the device folder and select *New/Device Folder* from the context menu). Enter the information required on the General tab, as described below. Click *Apply* when you are done.

Duplicating a Device Folder

To speed up the creation of similarly configured Device Folders, you can create multiple Device Folders in a single operation by duplicating an existing Device Folder. To do this, select an existing Device Folder and then select *Duplicate…* from the Edit menu (or right-click on the Device Folder and select *Duplicate…* from the context menu.). You will see the following dialog:



Base Text: DeviceFolder
Numeric Places: 3
First Number: 1
Number Increment: 1
Number Of Duplicates: 1
[✓] Including Subtree

To generate names for the duplicated Device Folders, the editor appends a number to the text from the Base Text field. The first duplicate uses the number from the First Number field. This number is then incremented by the Number Increment value for the consecutive folders. The Numeric Places field specifies the number of digits from the generated number that will be appended to the Base Text. As an example, if Numeric Places is *3* and First Number is *2*, the number 002 will be appended to the Base Text.

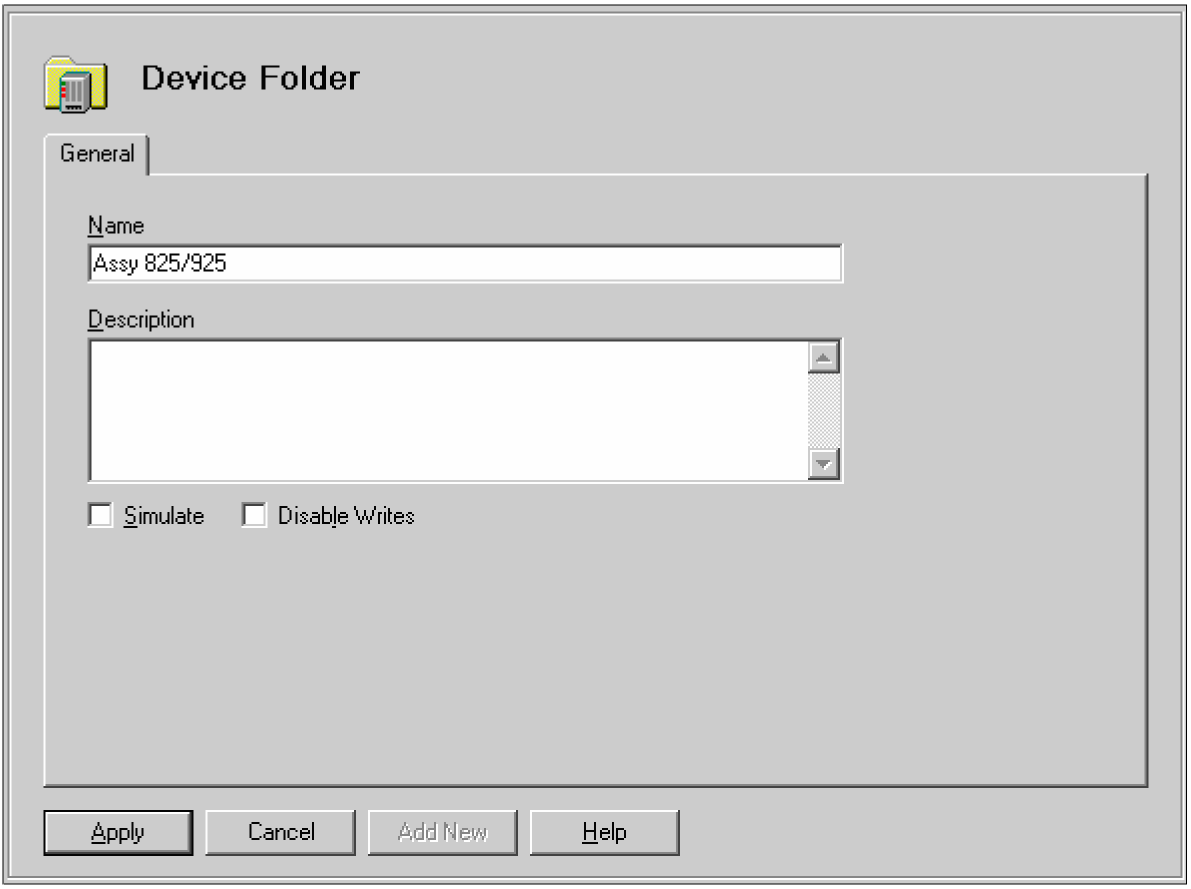
The Number Of Duplicates value specifies the number of Device Folders that will be created. The Including Subtree checkbox indicates whether all subtrees of the original Device Folder will be duplicated.

Deleting a Device Folder

To delete an existing Device Folder, select it and press the *Delete* key (or right-click on the device folder and select *Delete* from the context menu).

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

General Tab



Name
Assy 825/925

Description

Name

The Name identifies this Device Folder. It can be up to 50 characters long, may contain spaces, but must not begin with a space. It also may not contain any periods.

Description

This optional field further describes the Device Folder. It can be up to 255 characters long.

Simulate

Checking this box enables data simulation for all Data Items found at this level or below. This provides a quick way to switch between real and simulated data for a large number of Data Items. Refer to Simulation Signals for more information on simulating data.

Note:	If the Simulate checkbox is grayed-out, it indicates that simulation has already been selected at a higher level.

Disable Writes

Checking this box disables write requests for all Data Items found at this level or below. By default, this box is not checked and writes are enabled.

Note:	If the Disable Writes checkbox is grayed-out, it indicates that writes have already been disabled at a higher level.

Devices

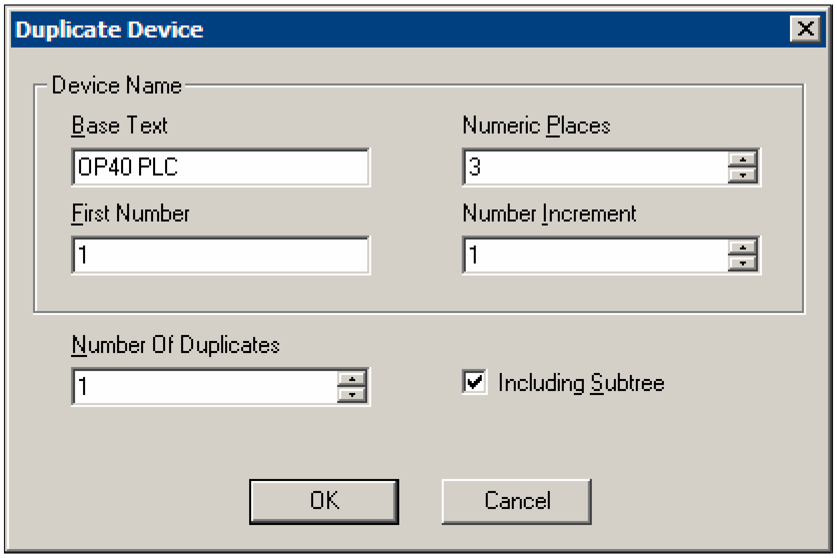
A Device in the Address Space tree represents a source of data to which the server communicates. This data source may be a PLC, a group of PLCs or other physical data sources. You can place Devices directly in the Address Space root folder or in Device Folders. In addition to its device-specific functionality, a Device operates as a folder. It can contain Folders and Data Items.

Creating a New Device

Select either the Address Space root folder or an existing Device Folder. From the Edit menu, select *New/Device* (or right-click on the folder and select *New/Device* from the context menu). Enter the required information as explained below. Click *Apply* when you are done.

Duplicating a Device

To speed up the creation of similarly configured Devices, you can create multiple Devices in a single operation by duplicating an existing Device. To do this, select an existing Device and then select *Duplicate…* from the Edit menu (or right-click on the Device and select *Duplicate…* from the context menu.). You will see the following dialog:


Duplicate Device

Device Name

Base Text: OP40 PLC

Numeric Places: 3

First Number: 1

Number Increment: 1

Number Of Duplicates: 1

[✓] Including Subtree

To generate names for the duplicated Devices, the editor appends a number to the text from the Base Text field. The first duplicate uses the number from the First Number field. This number is then incremented by the Number Increment value for the consecutive Devices. The Numeric Places field specifies the number of digits from the generated number that will be appended to the Base Text. As an example, if Numeric Places is *3* and First Number is *2*, the number *002* will be appended to the Base Text.

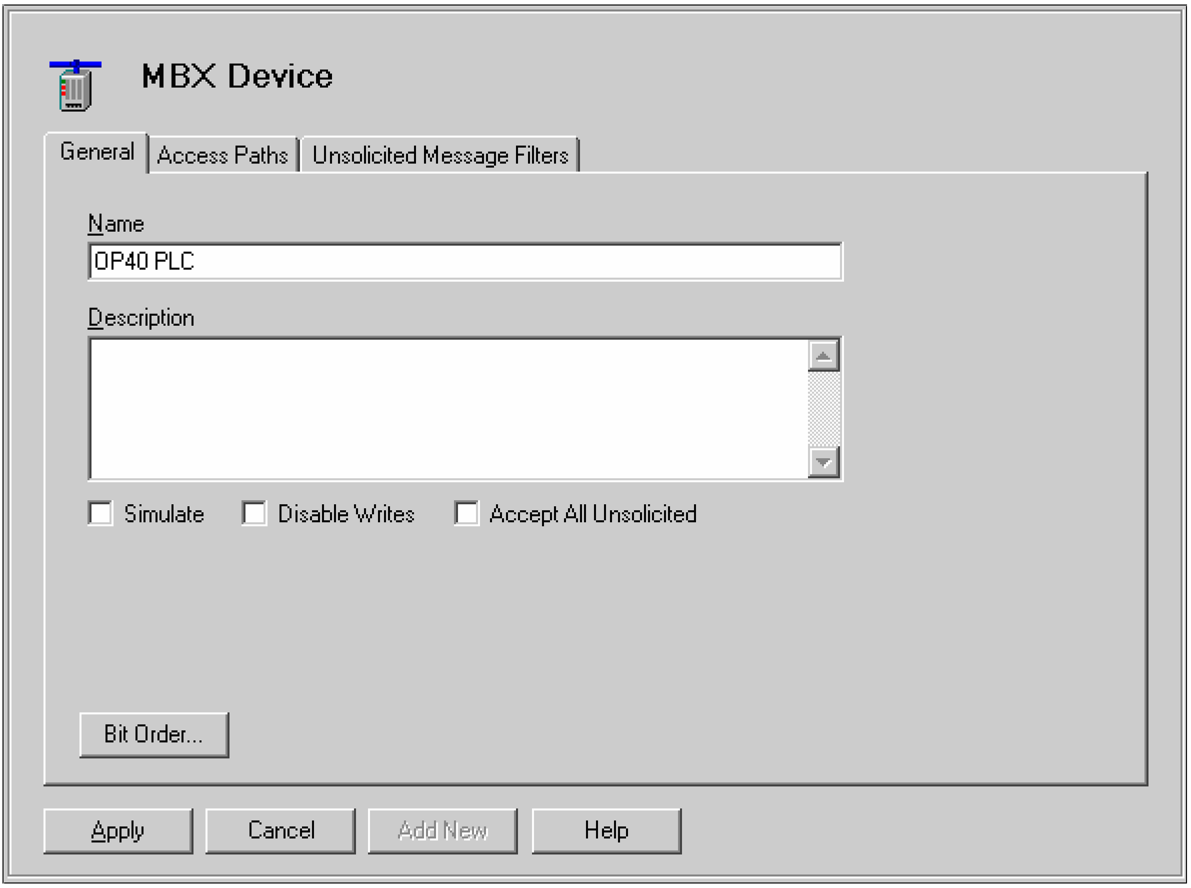
The Number Of Duplicates value specifies the number of Devices that will be created. The Including Subtree checkbox indicates whether all subtrees of the original Device will be duplicated.

Deleting a Device

To delete an existing Device, select it and press the *Delete* key (or right-click on the Device and select *Delete* from the context menu).

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

General Tab



Name
OP40 PLC

Name

The name identifies the Device. It can be up to 50 characters long, may contain spaces, but must not begin with a space. It also may not contain any periods.

Description

This optional field further describes the Device. It can be up to 255 characters long.

Simulate

Checking this box enables data simulation for all Data Items found at this level or below. This provides a quick way to switch between real and simulated data for a large number of Data Items. Refer to Simulation Signals for more information on simulating data.

Note:	If the Simulate checkbox is grayed-out, it indicates that simulation has already been selected at a higher level.

Disable Writes

Checking this box disables write requests for all Data Items found at this level or below. By default, this box is not checked and writes are enabled.

Note:	If the Disable Writes checkbox is grayed-out, it indicates that writes have already been disabled at a higher level.

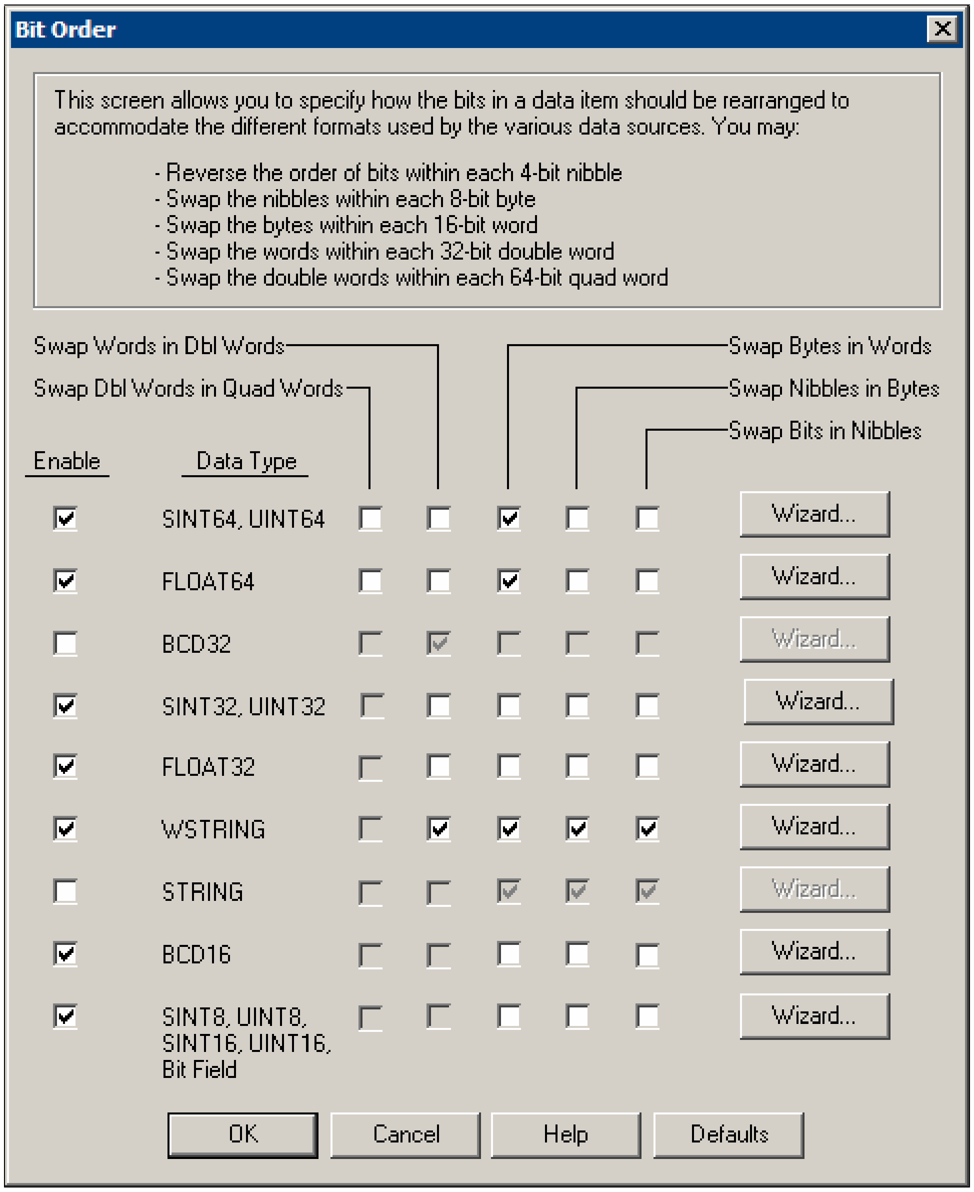
Accept All Unsolicited

When this box is checked, the Server will ignore the Unsolicited Message Filters and accept all unsolicited messages. By default, this box is not checked and unsolicited messages will be required to pass the filter criteria.

Bit Order…

Controllers and other devices will vary in the way they arrange the bits within their data items. For example, some store string data in low byte / high byte order, while others use high byte / low byte. Some have the bits in descending order, others in ascending order. The Bit Order configuration allows you to tell the Server how to rearrange the bits to accommodate these differences.

When you click the *Bit Order…* button, the Bit Order editing screen opens.



This screen allows you to specify how the bits in a data item should be rearranged to accommodate the different formats used by the various data sources. You may:

- Reverse the order of bits within each 4-bit nibble
- Swap the nibbles within each 8-bit byte
- Swap the bytes within each 16-bit word
- Swap the words within each 32-bit double word
- Swap the double words within each 64-bit quad word

Enable	Data Type	Swap Dbl Words in Quad Words	Swap Words in Dbl Words	Swap Bytes in Words	Swap Nibbles in Bytes	Swap Bits in Nibbles	
✓	SINT64, UINT64			✓			Wizard...
✓	FLOAT64			✓			Wizard...
	BCD32		✓				Wizard...
✓	SINT32, UINT32						Wizard...
✓	FLOAT32						Wizard...
✓	WSTRING		✓	✓	✓	✓	Wizard...
	STRING			✓	✓	✓	Wizard...
✓	BCD16						Wizard...
✓	SINT8, UINT8, SINT16, UINT16, Bit Field						Wizard...

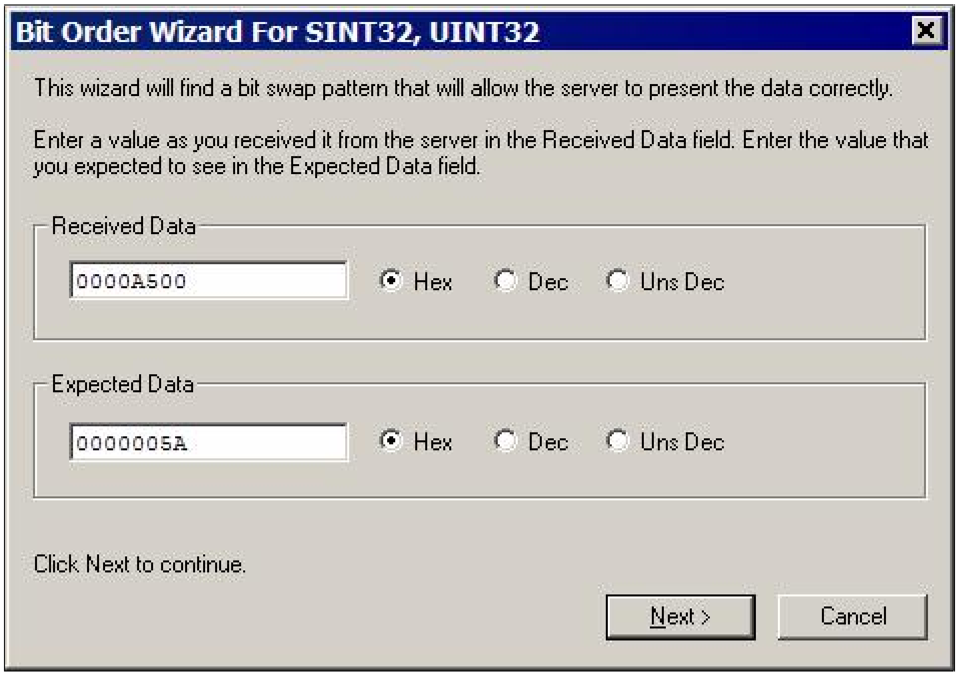
For each data type, you may rearrange groups of bits as needed by checking or unchecking the boxes. When the Enable box is checked, the specified swapping will be applied to that data type. When you uncheck this box, swapping is disabled, but the swap pattern you configured is retained so that it can be turned back on if you wish. Disabled swap patterns will be grayed out.

Each of the swapping check boxes affects the order only at one level. If you check the rightmost box, the Server will reverse the order of the four bits within each nibble, but the order of the nibbles within the bytes, words, etc. will not be affected. Checking the second box from the right will swap the order of the nibbles within each 8-bit byte, but will not affect the order of the bits within the nibbles, nor will it affect the order of the bytes within each word and so on.

You may check and uncheck the boxes as needed to properly arrange the data. Click *Defaults* to return the configuration to the default setting that works for most devices.

Note:	A common requirement is to completely reverse the order of all of the bits in a data item. To do this, you must check all of the available boxes for that type.

If you need help in determining the correct swap pattern, click *Wizard…* . The following wizard opens.



Bit Order Wizard For SINT32, UINT32

This wizard will find a bit swap pattern that will allow the server to present the data correctly.

Enter a value as you received it from the server in the Received Data field. Enter the value that you expected to see in the Expected Data field.

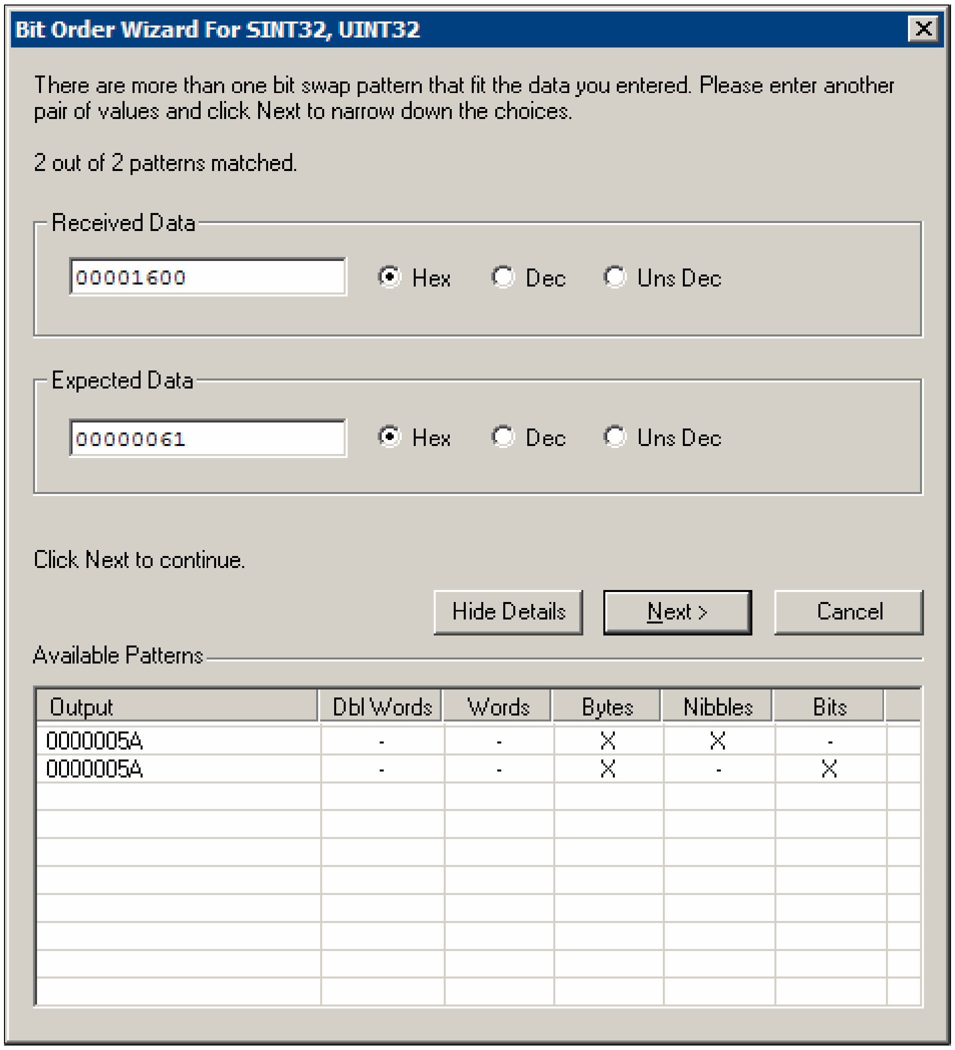
Received Data: 0000A500 [✓ Hex] Dec Uns Dec

Expected Data: 0000005A [✓ Hex] Dec Uns Dec

Click Next to continue.

The wizard allows you to enter the data as you see it and as you expected to see it – what you have and what you want. You are given a choice of formats for the data entry. The choices available will depend on the data type you are configuring and include hex, decimal, unsigned decimal, ASCII, Unicode, BCD and floating point.

In the example shown, you see A500, but expected to see 005A. After entering these values, click *Next*. The wizard will check all swap patterns to try to identify which ones will give you 005A.


Bit Order Wizard For SINT32, UINT32

There are more than one bit swap pattern that fit the data you entered. Please enter another pair of values and click Next to narrow down the choices.

2 out of 2 patterns matched.

Received Data: 00001600 [✓ Hex] Dec Uns Dec

Expected Data: 00000061 [✓ Hex] Dec Uns Dec

Click Next to continue.

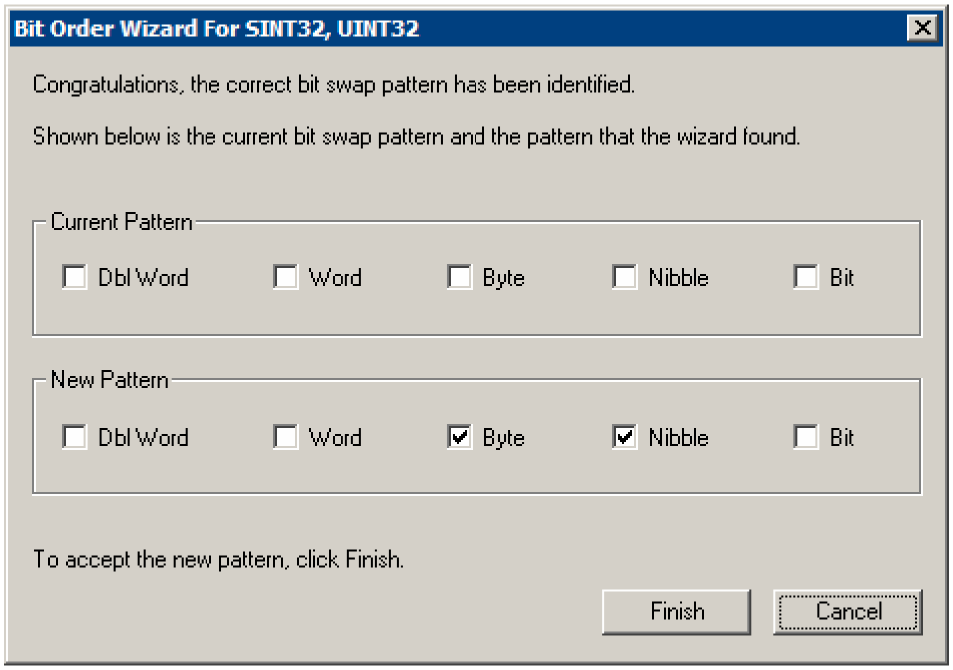
Available Patterns

Output	Dbl Words	Words	Bytes	Nibbles	Bits
0000005A	-	-	X	X	-
0000005A	-	-	X	-	X

In this case, it found two swap patterns will work for the given values. Click the *Show/Hide Details* button to open or close the Available Patterns display, which shows you the bit swap patterns that the wizard has identified.

You must identify one specific swap pattern to use, so you must enter another pair of Received Data and Expected Data values. This time you enter *1600* as the data you see and *0061* as what you expected. Now click *Next*. The wizard will only test the swap patterns that passed the first step to see if they match the current data. By doing this, it identifies those patterns that match both of the data pairs you entered.

The result below shows that the wizard has now narrowed down the possibilities to just one. Click *Finish* to exit from the wizard and use this swap pattern.



Congratulations, the correct bit swap pattern has been identified.

Shown below is the current bit swap pattern and the pattern that the wizard found.

Current Pattern: ☐ Dbl Word ☐ Word ☐ Byte ☐ Nibble ☐ Bit

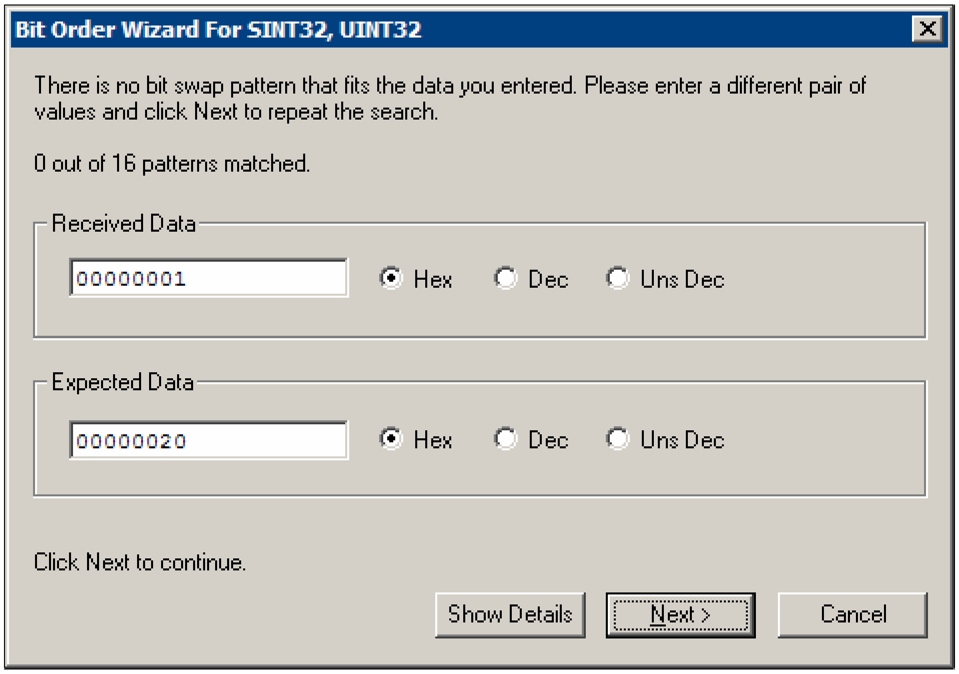
New Pattern: ☐ Dbl Word ☐ Word ☑ Byte ☑ Nibble ☐ Bit

To accept the new pattern, click Finish.

Note:	If there had been more than one pattern that matched both sets of data, you would have been returned to the previous screen and asked to provide a third pair of values. This process would continue until the wizard identified just one bit pattern that matched all of the data pairs you entered, after which you would reach this screen.

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

If none of the possible bit swap patterns match your data, the wizard will display the following screen. When you enter a new pair of values, the search process starts over and the earlier values are ignored.


Bit Order Wizard For SINT32, UINT32

There is no bit swap pattern that fits the data you entered. Please enter a different pair of values and click Next to repeat the search.

0 out of 16 patterns matched.

Received Data: 00000001 [✓ Hex] Dec Uns Dec

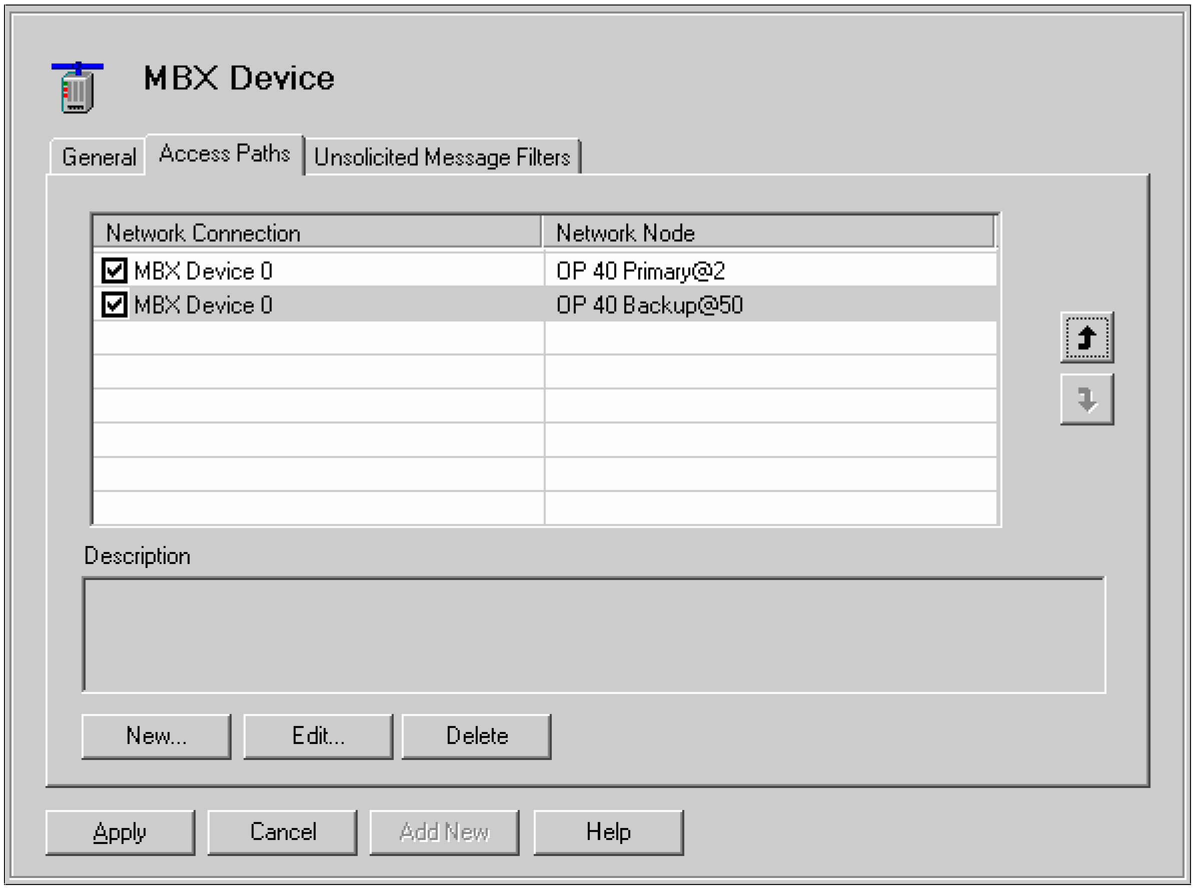
Expected Data: 00000020 [✓ Hex] Dec Uns Dec

Click Next to continue.

Access Paths Tab

Each Device has an associated list of Access Paths. Depending upon your network configuration, the Access Paths may include multiple paths to the same PLC, paths to different PLCs or some combination of the two. The Access Path at the top of the list is the primary Access Path; the rest are backups. If the current Access Path fails, the server switches to the highest Access Path that is available and enabled.

Note:	Access Paths are required for solicited communications only. If you are planning to use only unsolicited data updates, no Access Paths need to be configured.



Network Connection	Network Node
✓ MBX Device 0	OP 40 Primary@2
✓ MBX Device 0	OP 40 Backup@50

Enable Checkbox

To the left of each Access Path is a checkbox that, when checked, enables the Access Path. The Server uses only enabled Access Paths.

Network Connection

The Network Connection column displays the Network Connections associated with each of the Access Paths.

Network Node

The Network Node column displays the Network Nodes associated with each of the Access Paths.

Description

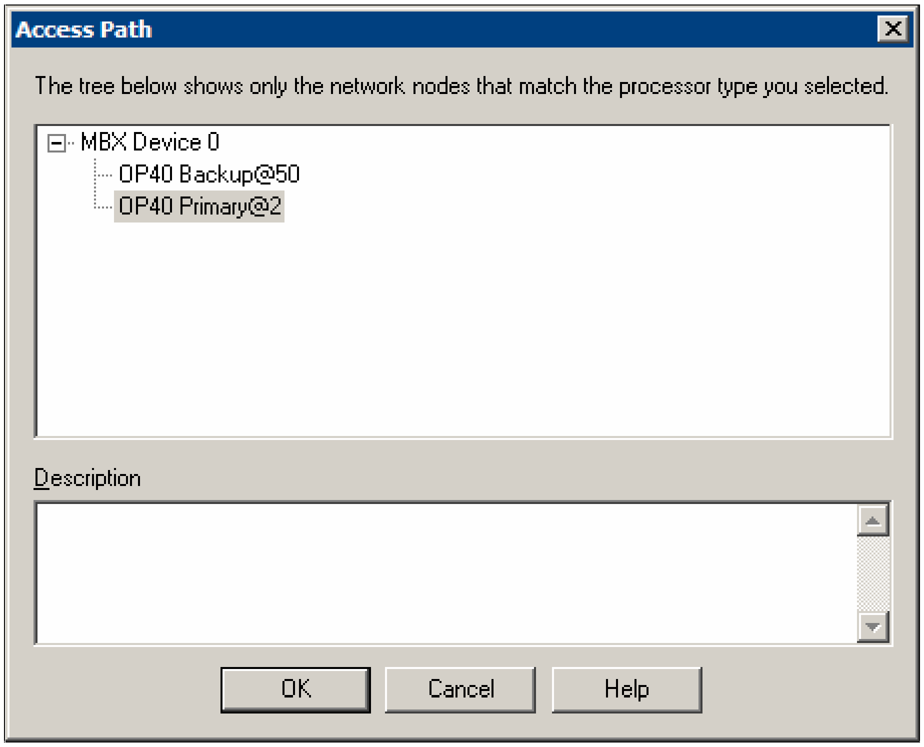
This optional field further describes the Access Path. It can be up to 255 characters long.

Changing Access Path Priority

The Server assigns priority to the Access Paths from top to bottom, with the Access Path at the top having the highest priority. Use the up and down arrows on the right side of the list box to adjust the priorities as needed.

Creating a New Access Path

Click the *New…* button (or right-click inside the list window and select *New…* from the context menu). You will see the following dialog:


Access Path

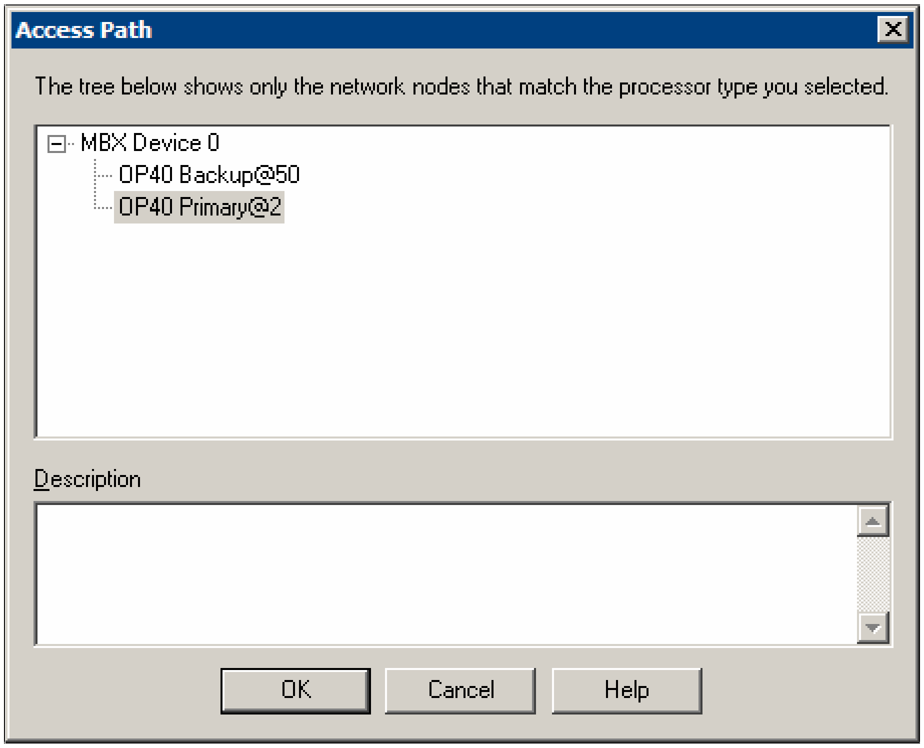
The tree below shows only the network nodes that match the processor type you selected.

MBX Device 0
OP40 Backup@50
OP40 Primary@2

Select the Network Node for this Access Path. You can also enter an optional Description text. Click *OK* when you are done.

Editing an Access Path

Select an existing Access Path and click the *Edit…* button (or right-click on the access path and select *Edit...* from the context menu). You will see the following dialog:


Access Path
The tree below shows only the network nodes that match the processor type you selected.
MBX Device 0
OP40 Backup@50
OP40 Primary@2
Description
OK
Cancel
Help

Modify the current selections and click *OK* when you are done.

Deleting an Access Path

Select an existing Access Path and press the *Delete* key (or right-click and select *Delete* from the context menu).

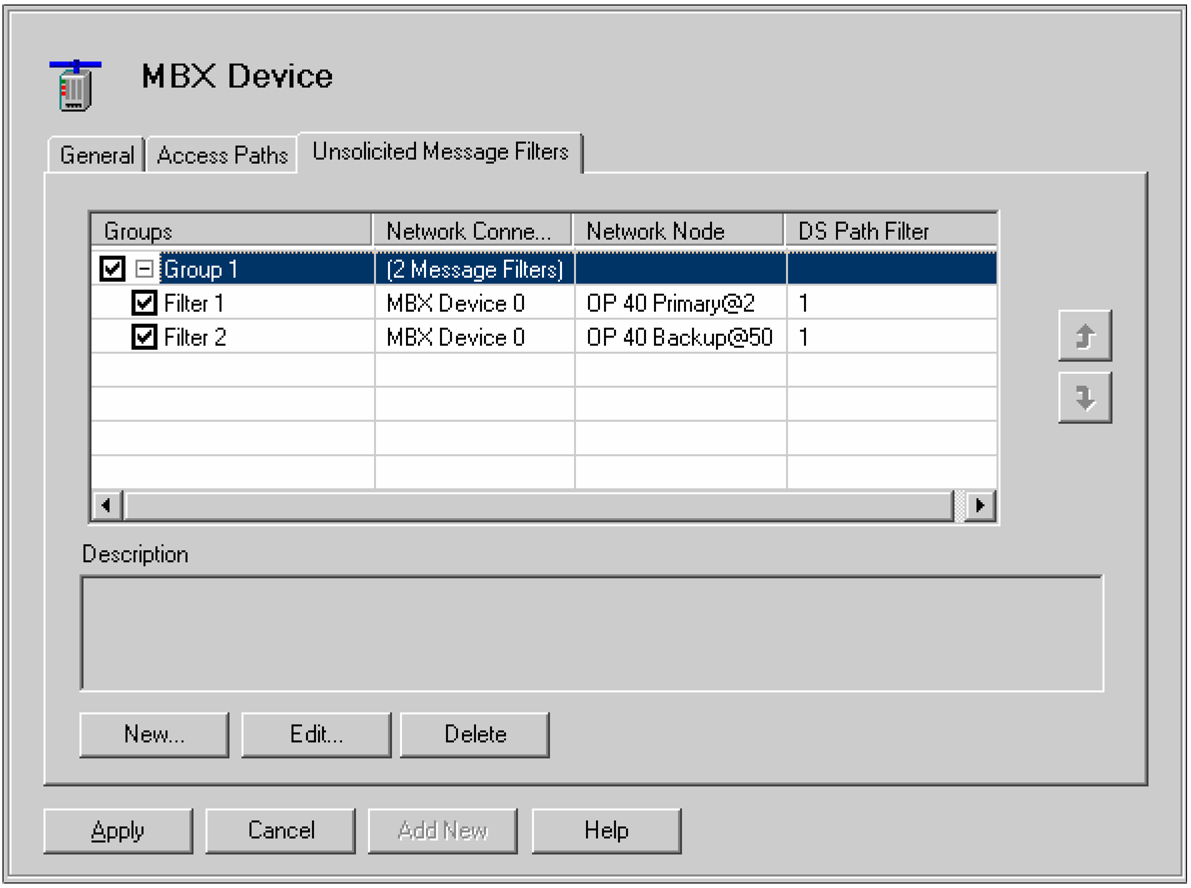
Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

Unsolicited Message Filters Tab

If the Accept All Unsolicited box on the General tab is checked, the Server will accept all unsolicited messages. Otherwise, unsolicited messages must pass the filter criteria to be accepted. These filters help ensure that unsolicited messages are accepted only from trusted sources.

The Unsolicited Message Filters are organized in groups. Each group has an equal priority and a message must pass through at least one of these groups in order to be accepted by the server.

An unsolicited message filter group has a list of trusted Network Nodes and/or trusted Network Connections and supports two modes of operation. In the default mode, the server will accept messages that pass any of the configured filters. In the alternative mode, the server treats the filter list as a ranked list of preferred and backup data sources. It monitors the connections to each unsolicited message source and accepts messages only from the highest ranked node that has a healthy connection.



Groups	Network Conne...	Network Node	DS Path Filter
☑ ⊟ Group 1	(2 Message Filters)		
☑ Filter 1	MBX Device 0	OP 40 Primary@2	1
☑ Filter 2	MBX Device 0	OP 40 Backup@50	1

Group

You can arrange the filters in one or more groups. Each group can be configured independently as either a prioritized or a non-prioritized list. There is no implied priority from one group to another.

Enable Checkbox

To the left of each unsolicited message group or filter is a checkbox that, when checked, enables that group/filter. The Server uses only the enabled groups and filters.

Network Connection

The Network Connection column displays the Network Connection associated with the filter.

Network Node

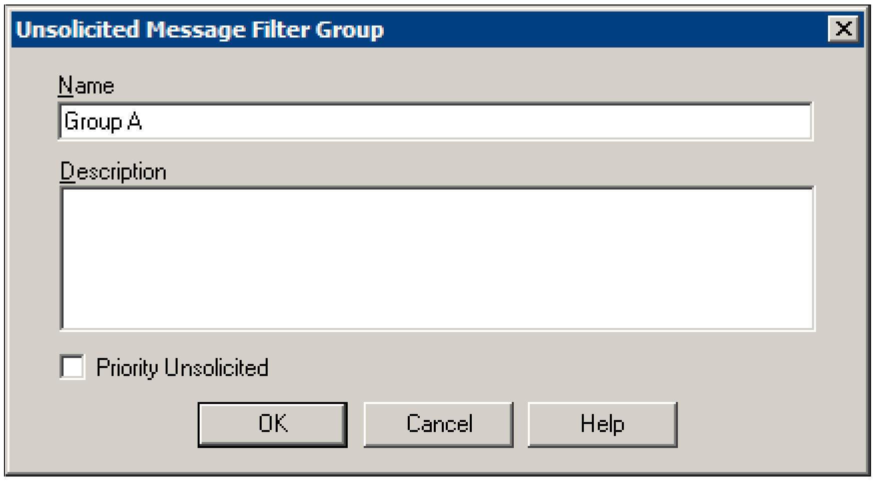
The Network Node column displays the Network Node associated with the filter.

Changing Filter Priority

The Server assigns priority to the filters in a group from top to bottom, with the filter at the top having the highest priority. Use the up and down arrows on the right side of the list box to adjust the priorities as needed.

Creating a New Unsolicited Filter Group

Click the *New…* button and select *Group…* (or right-click inside the list window and select *New* then Group… from the context menu). You will see the following dialog:



Enter the name of the group and an optional description, if desired. If you want this to be a prioritized list of filters, check the *Priority Unsolicited* box. Click *OK* when you are done.

Editing an Unsolicited Filter Group

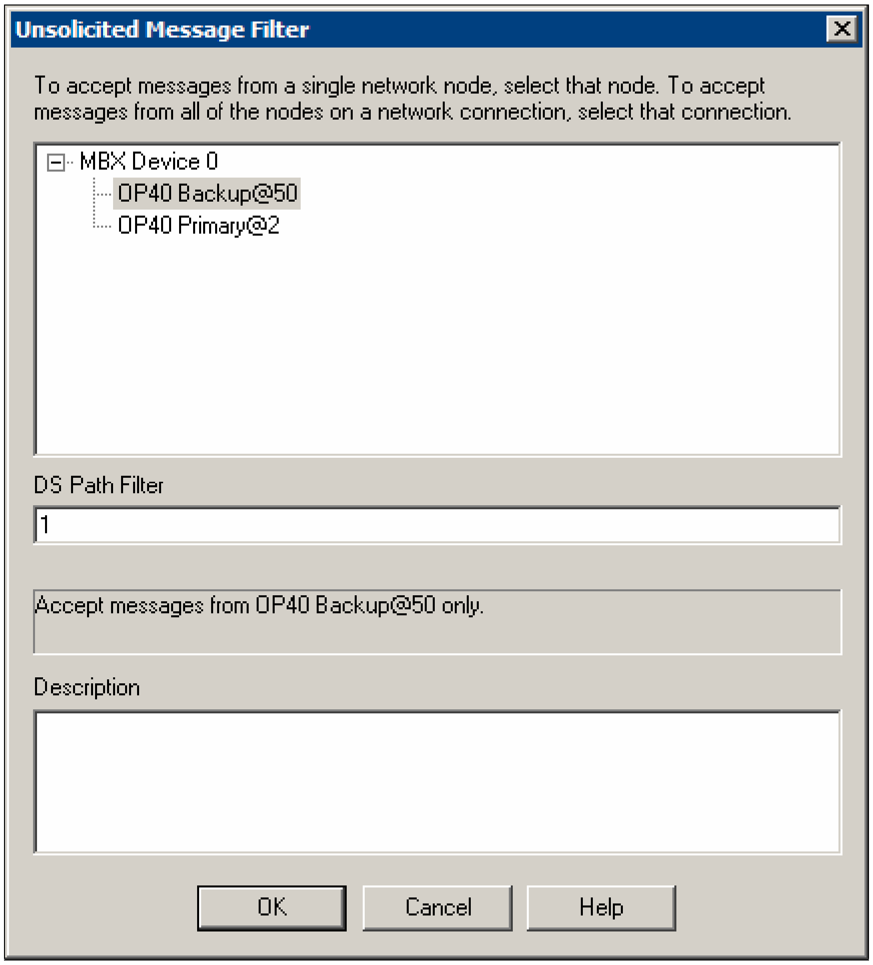
Select an existing group and click the *Edit…* button (or right-click on an existing filter and select *Edit...* from the context menu). You will see the same dialog box. Modify the current selections and click *OK* when you are done.

Deleting an Unsolicited Filter Group

Select an existing group and press the *Delete* key (or right-click on the filter and select *Delete* from the context menu).

Creating a New Unsolicited Filter

Select the Group in which you wish to create the filter. Click the *New* button and select *Filter…* (or right-click on the desired Group and select *New* then *Filter…* from the context menu). You will see the following dialog:


Unsolicited Message Filter

To accept messages from a single network node, select that node. To accept messages from all of the nodes on a network connection, select that connection.

- MBX Device 0
  - OP40 Backup@50
  - OP40 Primary@2

DS Path Filter: 1

Accept messages from OP40 Backup@50 only.

Description:

Select the Network Connection or Network Node you wish to use and enter an optional Description, if desired.

Click *OK* when you are done.

Editing an Unsolicited Filter

Select an existing filter and click the *Edit…* button (or right-click on an existing filter and select *Edit...* from the context menu). You will see the same dialog box. Modify the current selections and click *OK* when you are done.

Deleting an Unsolicited Filter

Select an existing filter and press the *Delete* key (or right-click on the filter and select *Delete* from the context menu).

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

Folders

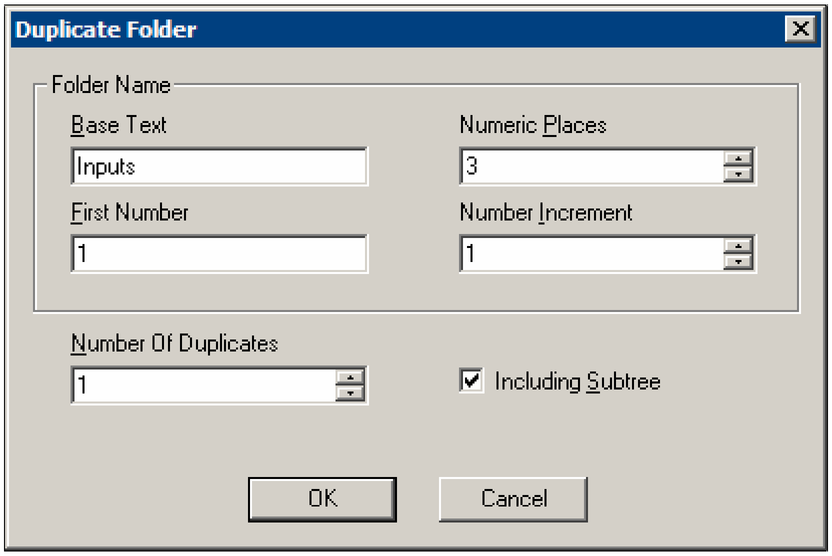
A Folder logically groups Data Items and other Folders. You can place Folders directly under Devices or under other Folders, up to four levels deep.

Creating a New Folder

Select an existing Device or Folder. From the Edit menu, select *New/Folder* (or right-click on the Device or Folder and select *New/Folder* from the context menu). Enter the information required on the General tab, as described below. Click *Apply* when you are done.

Duplicating a Folder

To speed up the creation of similarly configured Folders, you can create multiple Folders in a single operation by duplicating an existing Folder. To do this, select an existing Folder and then select *Duplicate…* from the Edit menu (or right-click on the Folder and select *Duplicate…* from the context menu). You will see the following dialog:



Folder Name
Base Text: Inputs
Numeric Places: 3
First Number: 1
Number Increment: 1
Number Of Duplicates: 1
☑ Including Subtree

To generate names for the duplicated folders, the editor appends a number to the text from the Base Text field. The first duplicate uses the number from the First Number field. This number is then incremented by the Number Increment value for the consecutive folders. The Numeric Places field specifies the number of digits from the generated number that will be appended to the Base Text. As an example, if Numeric Places is *3* and First Number is *2*, the number *002* will be appended to the Base Text.

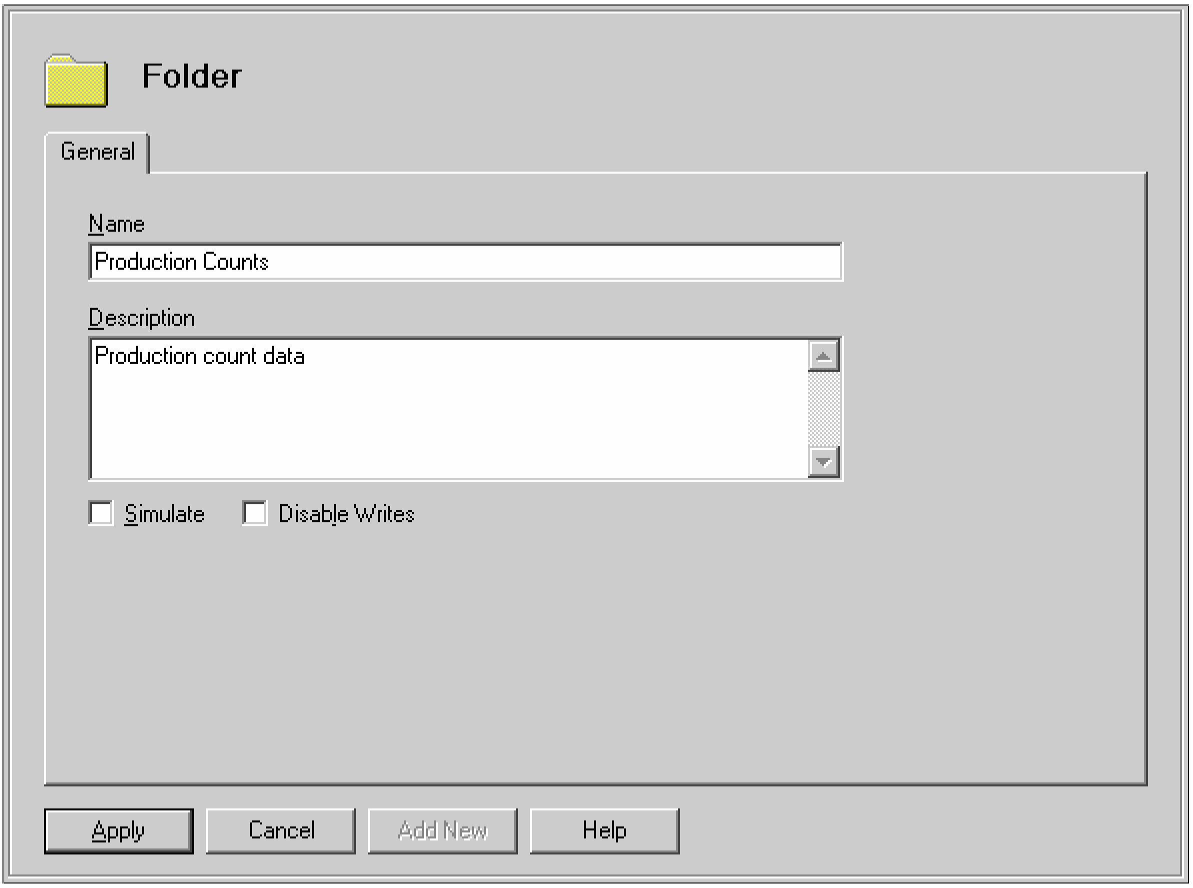
The Number Of Duplicates value specifies the number of folders that will be created. The Including Subtree checkbox indicates whether all subtrees of the original folder will be duplicated.

Deleting a Folder

Select an existing folder and press the *Delete* key (or right-click on the folder and select *Delete* from the context menu).

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

General Tab



Folder

Name: Production Counts

Description: Production count data

☐ Simulate ☐ Disable Writes

Name

The Name identifies this Folder. It can be up to 50 characters long, may contain spaces, but must not begin with a space. It also may not contain any periods.

Description

This optional field further describes the folder. It can be up to 255 characters long.

Simulate

Checking this box enables data simulation for all data items found at this level and below. This provides a quick way to switch between real and simulated data for a large number of Data Items. Refer to Simulation Signals for more information on simulating data.

Note:	If the Simulate checkbox is grayed-out, it indicates that simulation has already been selected at a higher level.

Disable Writes

Checking this box disables write requests for all Data Items found at this level or below. By default, this box is not checked and writes are enabled.

Note:	If the Disable Writes checkbox is grayed-out, it indicates that writes have already been disabled at a higher level.

Data Items

A Data Item represents a register in the physical device, a range of registers, a bit inside a register or a range of bits. Five types of registers are supported: coils (0xxxx), inputs (1xxxx), input registers (3xxxx), holding registers (4xxxx) and general registers (6xxxx). Coils, holding registers and general registers are read/write. Inputs, input registers and bits within registers are read-only.

Creating a New Data Item

Select one of the existing Devices or Folders and select *New/Data Item* from the Edit menu (or right-click on the device or folder and select *New/Data Item* from the context menu). Enter the information required in the Data Item dialog box, as described below. Click *Apply* when you are done.

Duplicating a Data Item

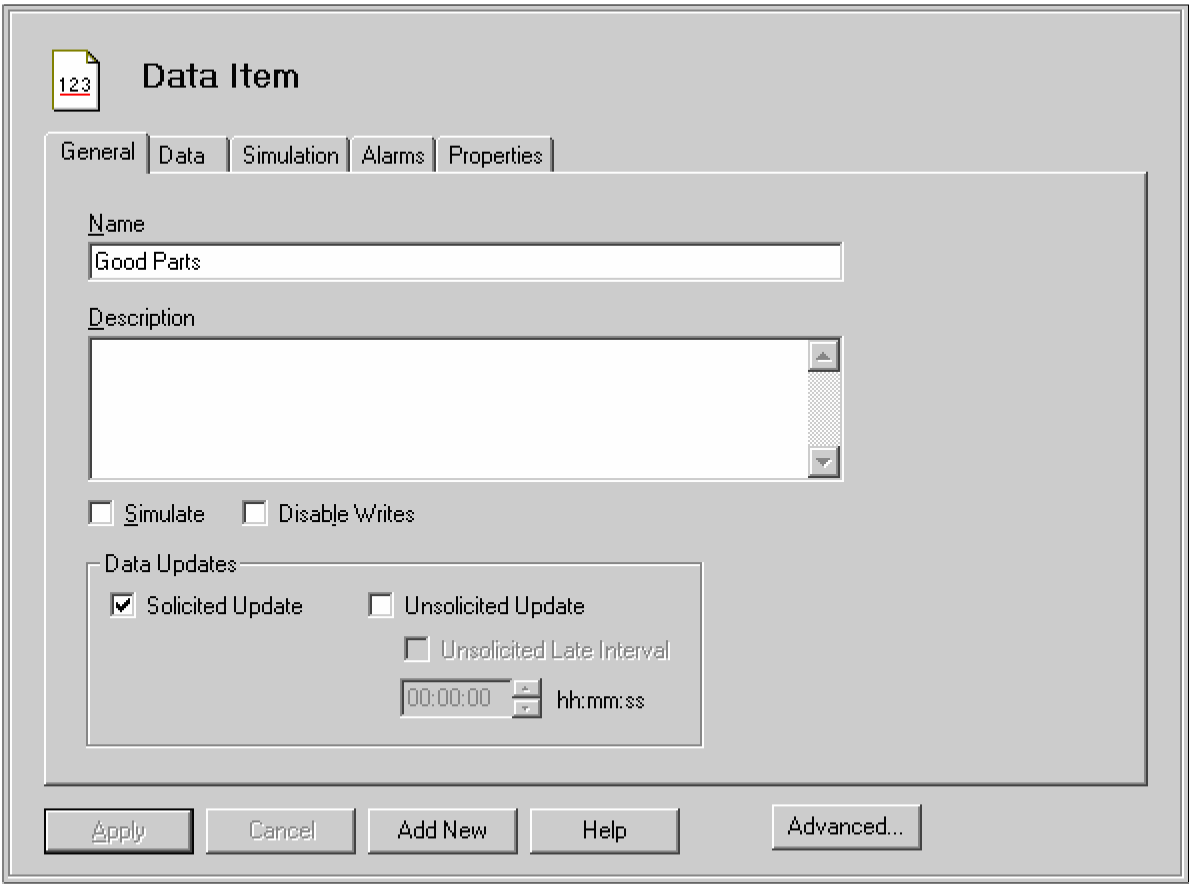
To speed up the creation of similarly configured Data Items, you can easily create multiple Data Items by duplicating an existing Data Item. To do this, select an existing Data Item and then select *Duplicate…* from the Edit menu (or right-click on the Data Item and select *Duplicate…* from the context menu). The Duplicate Data Item Wizard will open and will guide you through the duplication process. For details of this operation, refer to Appendix B: Data Item Duplication Wizard.

Deleting a Data Item

Select an existing data item and press the *Delete* key (or right-click on the data item and select *Delete* from the context menu).

Caution:	After you edit the configuration, you must open the *File* menu and select *Save & Update Server* for the changes you have made to take effect. Otherwise, the Server will still be running with the old configuration.

General Tab



Name
Good Parts

Description

[ ] Simulate [ ] Disable Writes

Data Updates
[✓] Solicited Update [ ] Unsolicited Update
[ ] Unsolicited Late Interval
00:00:00 hh:mm:ss

Name

The Name identifies the Data Item. It can be up to 50 characters long, may contain spaces, but must not begin with a space. It also may not contain any periods.

Description

This optional field further describes the Data Item. It can be up to 255 characters long.

Simulate

Checking this box enables data simulation for this data item. Refer to Simulation Signals for more information on simulating data signals.

Note:	If the Simulate checkbox is grayed-out, it indicates that simulation has already been selected at a higher level.

Disable Writes

Checking this box disables all write requests for this Data Item. By default, this box is not checked and writes are enabled.

Note:	If the Disable Writes checkbox is grayed-out, it indicates that writes have already been disabled at a higher level.

Solicited Update

When checked, this box tells the Server to update this Data Item by polling. The Server will use the Access Paths specified in this item's parent to retrieve that data.

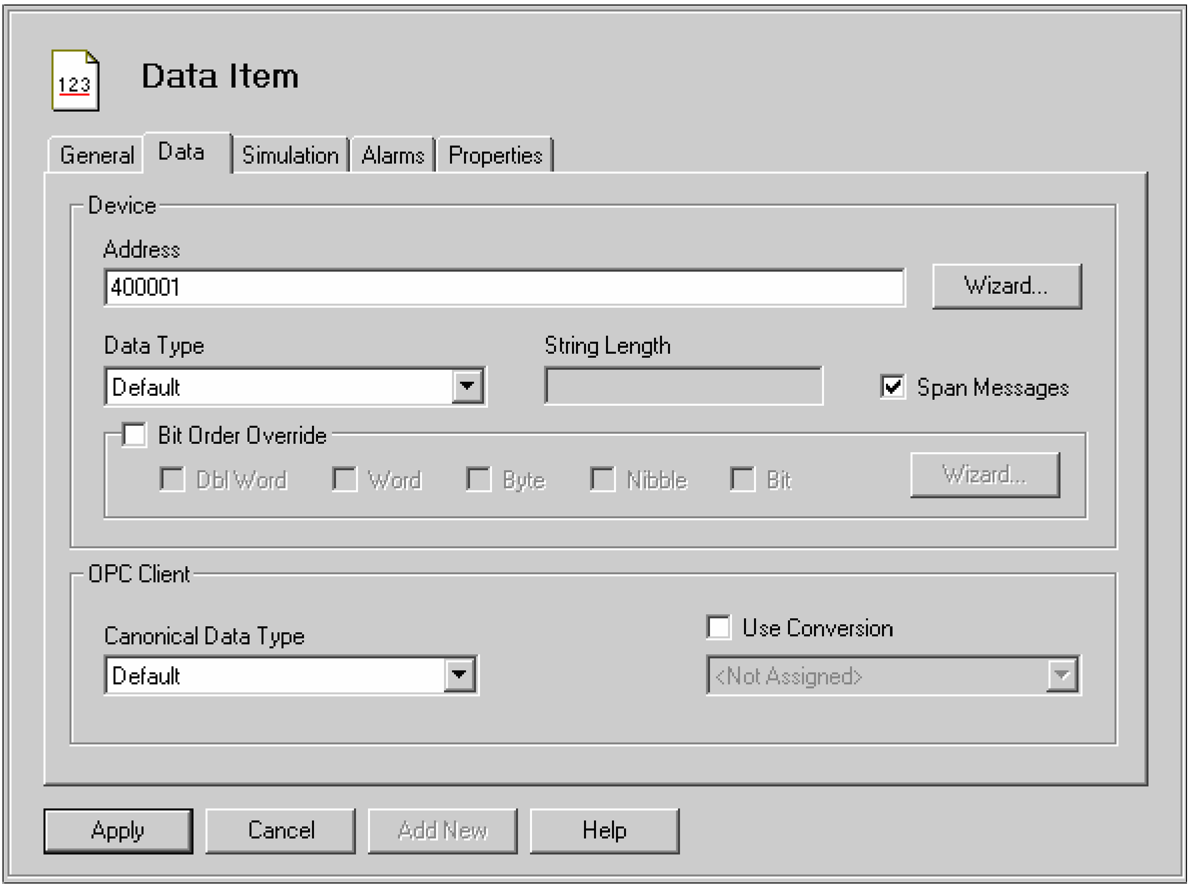
Unsolicited Update

When checked, this box tells the Server to allow this Data Item to be updated through unsolicited messages. The Server will use the Unsolicited Message Filters specified in this item's parent to decide which unsolicited messages will be allowed to modify this value. This box is unchecked by default, disabling unsolicited updates.

Unsolicited Late Interval

This is the maximum interval at which the Server will expect to receive unsolicited updates for this Data Item. If the Server does not receive an unsolicited update for this Data Item within this interval, the item's quality is downgraded to *Uncertain*.

Data Tab



Address: 400001

Data Type: Default

String Length:

[✓] Span Messages

[ ] Bit Order Override

Canonical Data Type: Default

[ ] Use Conversion

<Not Assigned>

Address

This is the address of the data in the physical device. For a complete list of all valid Modicon addresses, refer to Appendix A: PLC Addresses. In general, all addresses, as documented in Modicon manuals, are acceptable. However, the MBX driver agent extends this syntax when it comes to specifying arrays, array elements, register bits and bit fields.

If you have difficulties with specifying valid addresses, the address Wizard can help you define the address correctly. For details on running the wizard, refer to Appendix C: Address Wizard.

Arrays: Single-dimensional arrays of most *Data Types* are supported with the exception of strings and bit fields. To specify an array of *Data Type* elements, use the following syntax:

{Valid Register Address}**[**{Number of Elements}**]**

or

{Valid Register Address}**[**{Number of Elements}**,**{Lower Bound}**]**

The Lower Bound specifies the array index value for the first element in an array. If not specified, the Lower Bound defaults to zero. Visual Basic applications may expect the first index to be 1, in which case you would set the Lower Bound value to 1. Here are a few examples of arrays:

400001[5]	Array of five registers starting from address 400001 and a lower bound of 0
0:1[4,1]	Array of four bits starting from address 000001 and a lower bound of 1
4:00010/0[16]	Array of sixteen bits starting from bit 400010/0 and a lower bound of 0

Array Elements: The 16-bit registers may be viewed as arrays of two bytes. To specify an element of such an array, use the following syntax:

{Valid Register Address}**(**{ Element Index}**)**

Here are a few examples of array elements:

400001(0)	First byte of register 400001
300001(1)	Second byte of register 300001

Notice that it is possible to combine this specification with the array syntax to specify an array of 8-bit bytes beginning at a specific byte. For example:

400001(1)[10]	Array of 10 bytes starting from the second byte of 400001

Register Bits: To specify a specific bit within a 16-bit register, use the following syntax:

{Valid Register Address}**/**{Bit Number}

The Bit Number is in the range of 0-15. Here are a few examples of valid bit addresses:

400001/1	Bit #1 in register 400001
300001/15	Bit #15 in register 300001

Bit Fields: To specify a sequence of bits as a bit field, use the following syntax:

{Valid Bit Address}**,**{Bit Count}

Here are a few examples of bit fields:

400001/1,5	Bit field of five bits starting from bit 400001/1
100001,16	Bit field of sixteen bits starting from bit 100001

Data Type

Selects the native type of the data as it is found in the physical device. This tells the Server how many bytes of data to read and write for this Data Item. Not all data types are allowed for all register types. Normally, you will set the type to Default, which selects the usual data format for that type of register. However, in unusual applications where the data is stored in the physical device in a non-standard way, you can specify a different format to simplify the access to the data.

String Length

If the Data Type is a string format, this field specifies the number of bytes in the string. Remember that Unicode strings require two bytes per character.

Span Messages

This check box is available only for strings, arrays and bit fields. If the box is checked, the Server may take more than a single message to obtain all data for this item. This may lead to data tearing if the data changes between the messages. If the box is not checked, the Server will deliver all of the data for this Data Item in a single message. However, large Data Items may not fit into a single message. In that case the Span Messages box must be checked.

Bit Order Override

The Bit Order feature of the MBX OPC Server allows you to control how the bits are arranged within a Data Item. Each of the configured Devices can have its own default bit order configuration, but you can override the default for this Data Item by checking the *Bit Order Override* box. When you do, some or all of the check boxes and the Wizard... button will be activated. Check boxes that are not activated do not apply to data of the type used for this Data Item.

If you know the bit-ordering configuration you want to use, you may set it by checking the appropriate boxes.

- *Dbl Word* swaps the 32-bit double words within each 64-bit long word.
- *Word* swaps the 16-bit words within each 32-bit double word.
- *Byte* swaps the 8-bit bytes within each 16-bit word.
- *Nibble* swaps the 4-bit nibbles within each 8-bit byte.
- *Bit* reverses the order of the bits within each 4-bit nibble.

If you are unsure of the pattern to use, click *Wizard...*, which will help you find the right pattern by comparing the data you see with the data you expected to see.

For more information about setting the bit-ordering pattern and using the Wizard, refer to the Bit Order… discussion in the Devices section.

Canonical Data Type

This selection specifies the default variant format (VT_xxx) in which the data will be sent to OPC clients. The most common selection is *Default*, in which case the server matches the variant type to the items Data Type. The following table shows this default mapping:

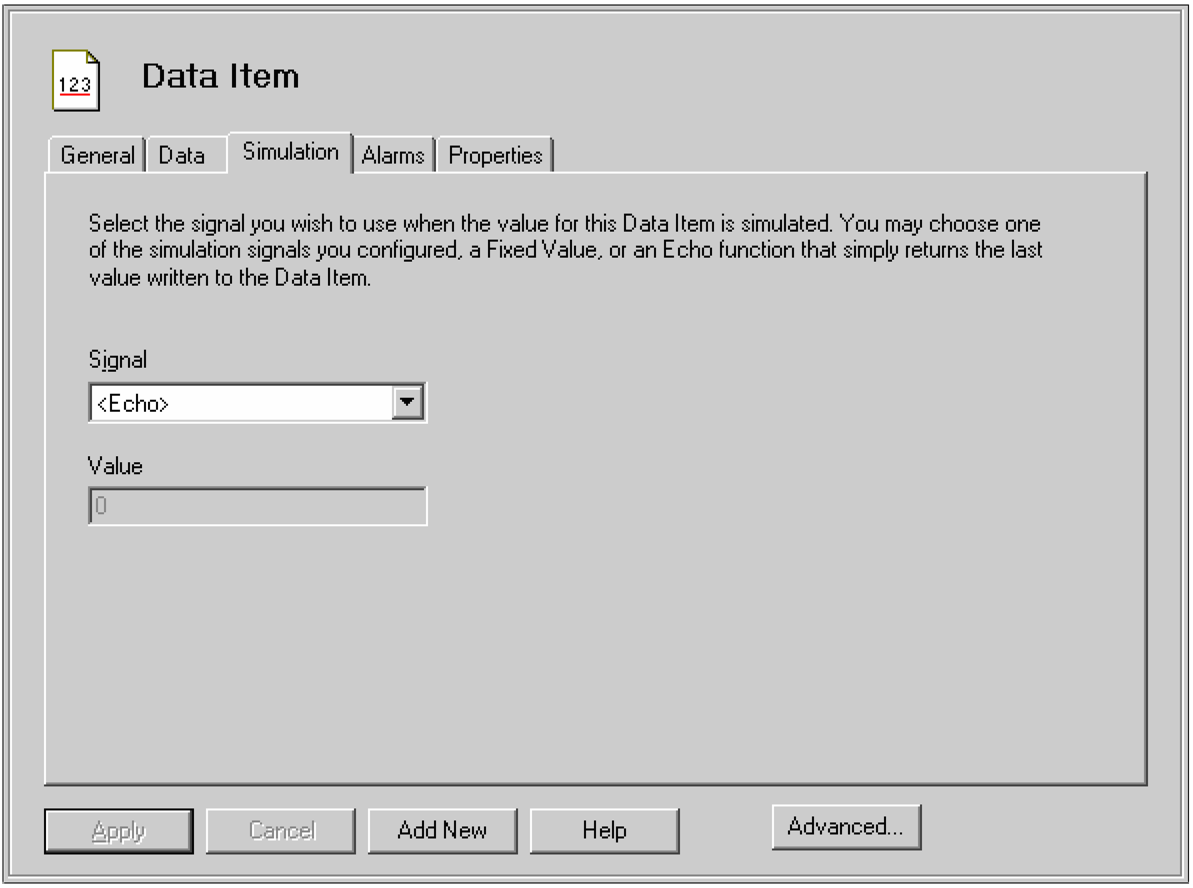
Data Type	Default Canonical Data Type	Description
BIT	VT_BOOL	1-bit Boolean
SINT8	VT_I1	Signed 8-bit integer
UINT8	VT_UI1	Unsigned 8-bit integer
SINT16	VT_I2	Signed 16-bit integer
UINT16	VT_UI2	Unsigned 16-bit integer
SINT32	VT_I4	Signed 32-bit integer
UINT32	VT_UI4	Unsigned 32-bit integer
SINT64	VT_I8	Signed 64-bit integer
UINT64	VT_UI8	Unsigned 64-bit integer
FLOAT32	VT_R4	IEEE format 32-bit floating point number
FLOAT64	VT_R8	IEEE format 64-bit floating point number
BCD16	VT_UI2	BCD value in the range of 0 - 9999
BCD32	VT_UI4	BCD value in the range of 0 - 99999999
STRING	VT_BSTR	Zero terminated ASCII string of 8-bit characters
WSTRING	VT_BSTR	Zero terminated UNICODE string of 16-bit characters
FIELD	The best fitting VT_UIx or array of VT_UI1 if size > 64	Bit field

Each OPC client can request data in any variant format and consequently override the Canonical Data Type setting.

Use Conversion

By checking this box, the user indicates that the selected Conversion should be applied to the data before the value is stored in the Data Item cache. You must select one of the previously-configured Conversions from the drop-down box.

Simulation Tab



Data Item

Select the signal you wish to use when the value for this Data Item is simulated. You may choose one of the simulation signals you configured, a Fixed Value, or an Echo function that simply returns the last value written to the Data Item.

Signal: <Echo>

Value: 0

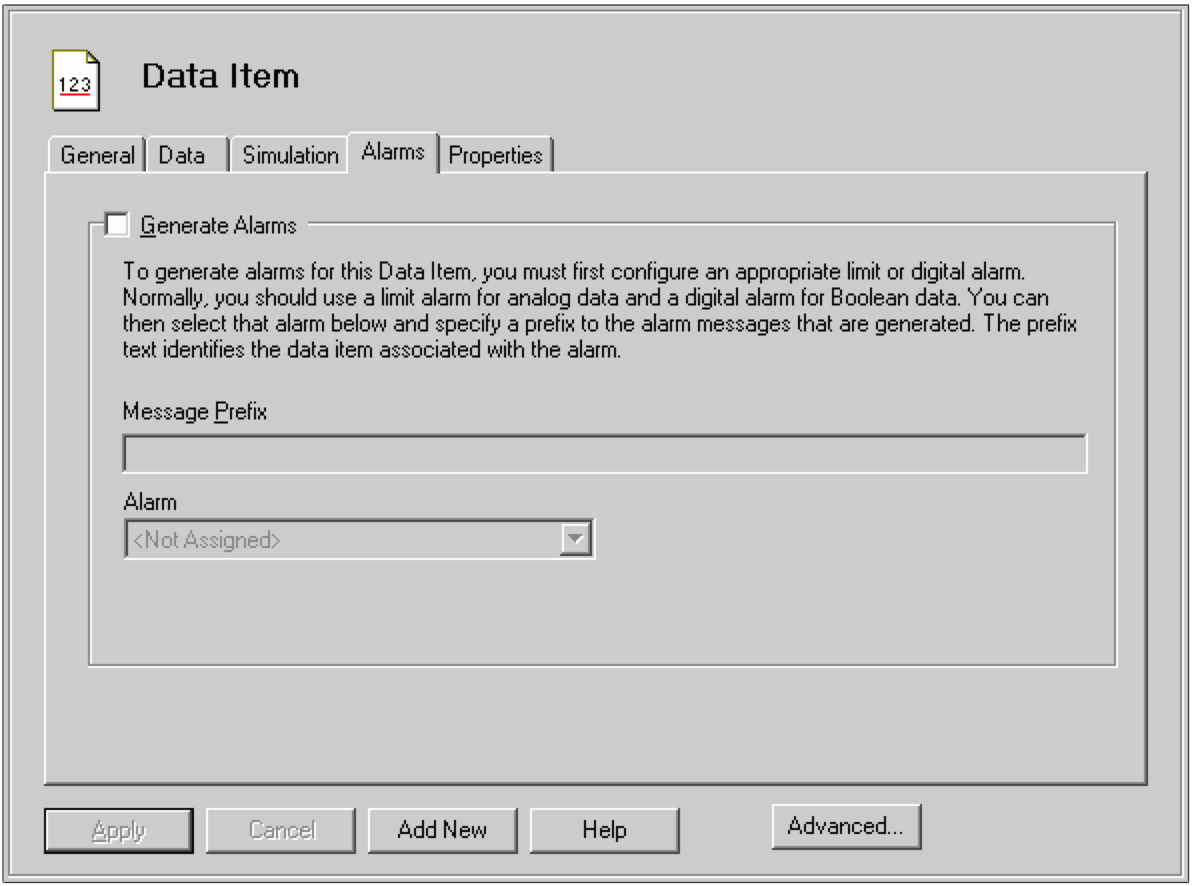
Signal

If you enabled simulation on the General tab, you may choose to simulate the Data Item value with one of the previously defined Simulation Signals, a fixed value or an echo of the last value written to the item.

Value

When Signal is set to *Fixed Value*, the Data Item will be set to this value.

Alarms Tab



To generate alarms for this Data Item, you must first configure an appropriate limit or digital alarm. Normally, you should use a limit alarm for analog data and a digital alarm for Boolean data. You can then select that alarm below and specify a prefix to the alarm messages that are generated. The prefix text identifies the data item associated with the alarm.

Message Prefix

Alarm

<Not Assigned>

Generate Alarms

If this box is checked, the Server will test the alarm conditions for this Data Item. Refer to the Alarm Definitions section for more information on creating and using alarms.

Message Prefix

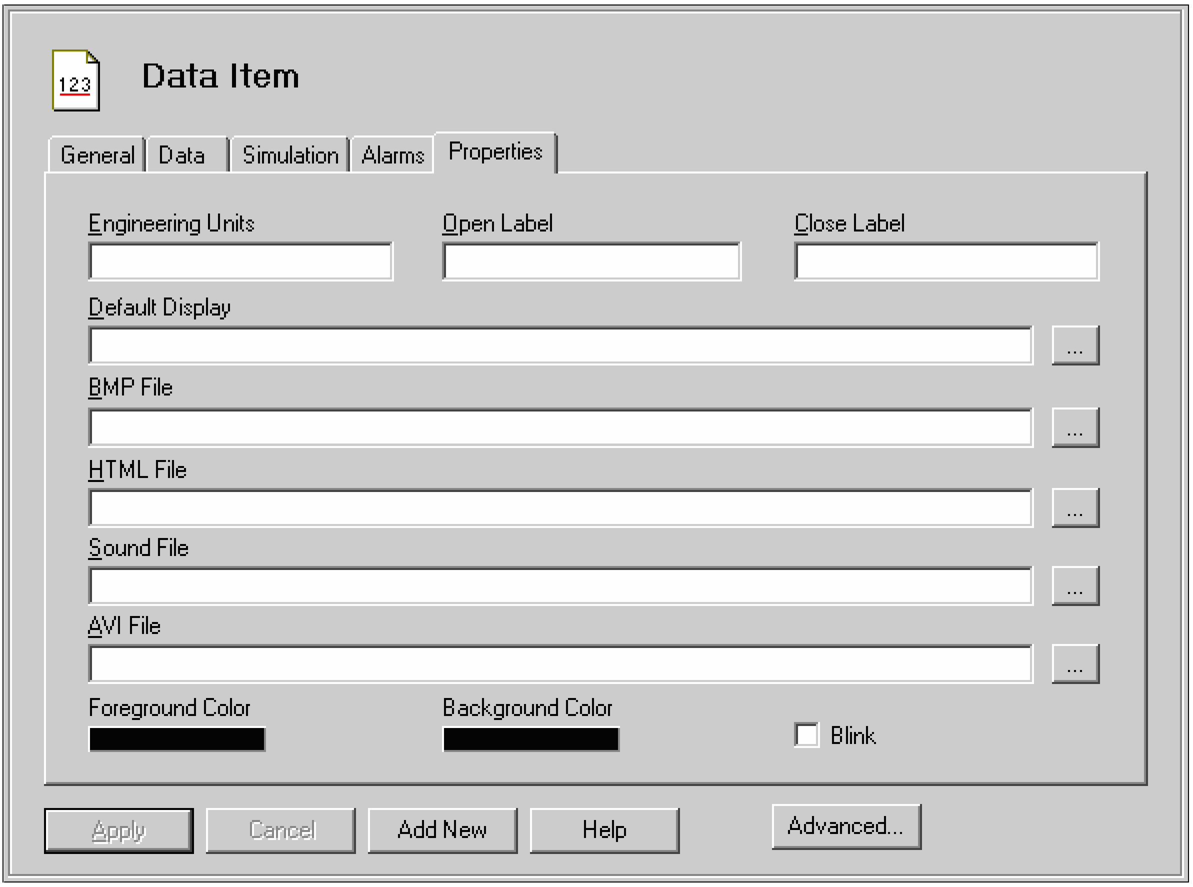
Enter the text for the first part of the alarm message. The second part will be the body text of the specific alarm that is generated.

Alarm

Select the Alarm template for this Data Item.

Properties Tab

In addition to the main Data Item properties – value, quality and timestamp – the OPC specification includes several optional properties that your client application may use. This tab allows you to set these Data Item properties. These properties are static and do not change while the Server is running.



Engineering Units

This is OPC property ID 100. It specifies the engineering units text, such as *DEGC* or *GALLONS*. It can be up to 50 characters long.

Open Label

This is OPC property ID 107 and is presented only for discrete data. This text describes the contact when it is in the open (zero) state, such as *STOP, OPEN, DISABLE* or *UNSAFE*. It can be up to 50 characters long.

Close Label

This is OPC property ID 106 and is presented only for discrete data. This text describes the contact when it is in the closed (non-zero) state, such as *RUN, CLOSE, ENABLE* or *SAFE*. It can be up to 50 characters long.

Default Display

This is OPC property ID 200. It is the name of an operator display associated with this Data Item. It can be up to 255 characters long.

BMP File

This is OPC property ID 204. It is the name of a bitmap file associated with this Data Item, for example *C:\MEDIA\FIC101.BMP*. It can be up to 255 characters long.

HTML File

This is OPC property ID 206. It is the name of the HTML file associated with this Data Item, for example *http://mypage.com/FIC101.HML*. It can be up to 255 characters long.

Sound File

This is OPC property ID 205. It is the name of the sound file associated with this Data Item, for example *C:\MEDIA\FIC101.WAV*. It can be up to 255 characters long.

AVI File

This is OPC property ID 207. It is the name of the AVI file associated with this Data Item, for example *C:\MEDIA\FIC101.AVI*. It can be up to 255 characters long.

Foreground Color

This is OPC property ID 201. Click on the box and select the foreground color used to display the item.

Background Color

This is OPC property ID 202. Click on the box and select the background color used to display the item.

Blink

This is OPC property ID 203. Check this box to indicate that displays of the item should blink.

Conversions

The raw data associated with Data Items may be process values from instruments. In most cases, these measurements are not expressed in engineering units. To simplify operations on the data, the Cyberlogic OPC Server allows you to associate a Data Conversion with each Data Item.

A user can define many different Conversions. A number of Data Items can then use each Conversion. As a result, the user need not define the same Conversion many times over.

Refer to the Cyberlogic OPC Server Help for a full discussion.

Simulation Signals

The Server can simulate the data for each of the Data Items according to a predefined formula. This makes it easy to perform client-side testing without the need for a physical device.

A user can define many different types of Simulation Signals. A number of Data Items can then use each such signal. As a result, the user need not define the same Simulation Signal many times over.

The Server can generate the following types of Simulation Signals:

- Read count
- Write count
- Random
- Ramp
- Sine
- Square
- Triangle
- Step

Each signal has parameters that define properties such as amplitude, phase and number of steps.

Refer to the Cyberlogic OPC Server Help for a full discussion.

Alarm Definitions

The Cyberlogic OPC Server supports the OPC Alarms and Events specification. The user may define many different alarm conditions. A number of Data Items can then use each such condition. As a result, the user need not define the same alarm condition many times over.

There are two categories of alarms: digital and limit. Digital alarms are normally used with Boolean Data Items and limit alarms are normally used with numeric Data Items, but both types of alarms may be used with either data type. Alarms cannot be used with string or array Data Items or with bit fields larger than 64 bits.

Refer to the Cyberlogic OPC Server Help for a full discussion.

Note:	Configuring alarms is meaningful only if your client software supports the OPC Alarms & Events specification. Consult your client software documentation to see what specifications it supports.

Saving Configuration Changes

When you are finished making changes, you must save them and update the Server before they will take effect. The procedure to do this is covered in the Cyberlogic OPC Server Help.

Undoing Configuration Changes

To undo configuration changes and revert to the previously saved configuration, select *Undo Changes* from the File menu (or click the *Undo Changes* button on the standard buttons toolbar).

Configuration Import/Export

The Import/Export feature allows you to export the configuration data to text file format and import configuration data from these exported files and comma separated values files from other vendors' OPC servers and programming software, including:

- Ingear OPC Server.csv
- Kepware KepserverEx.csv
- Matrikon OPC Server.csv
- Concept (text delimited variables)

For details on this important feature and instructions in its operation, refer to the Cyberlogic OPC Server Help.

Options

The editor has several options that may be set to adjust the operation of the editor to suit your preferences and to set security levels as needed for communication with client software. For a full discussion, refer to the Cyberlogic OPC Server Help.

VALIDATION & TROUBLESHOOTING

The following sections describe features that will help you to verify and troubleshoot your Server's operation. The Data Monitor and Cyberlogic OPC Client allow you to view the data as it is received by the Server. Microsoft's Performance Monitor allows you to view relevant performance information. The DirectAccess feature lets you look at data values even if they have not been configured as Data Items. The Event Viewer may provide important status or error messages. Finally, there is a list of MBX Driver Agent Messages and Frequently Asked Questions to assist in your troubleshooting.

Data Monitor

The Data Monitor lets you monitor the values and status of the Data Items. Its use is described in detail in the Cyberlogic OPC Server Help.

Cyberlogic OPC Client

The Cyberlogic OPC Client is a simple Data Access client that lets you see how the Server interacts with a client and lets you test its response to various loads. Its use is described in detail in the Cyberlogic OPC Server Help.

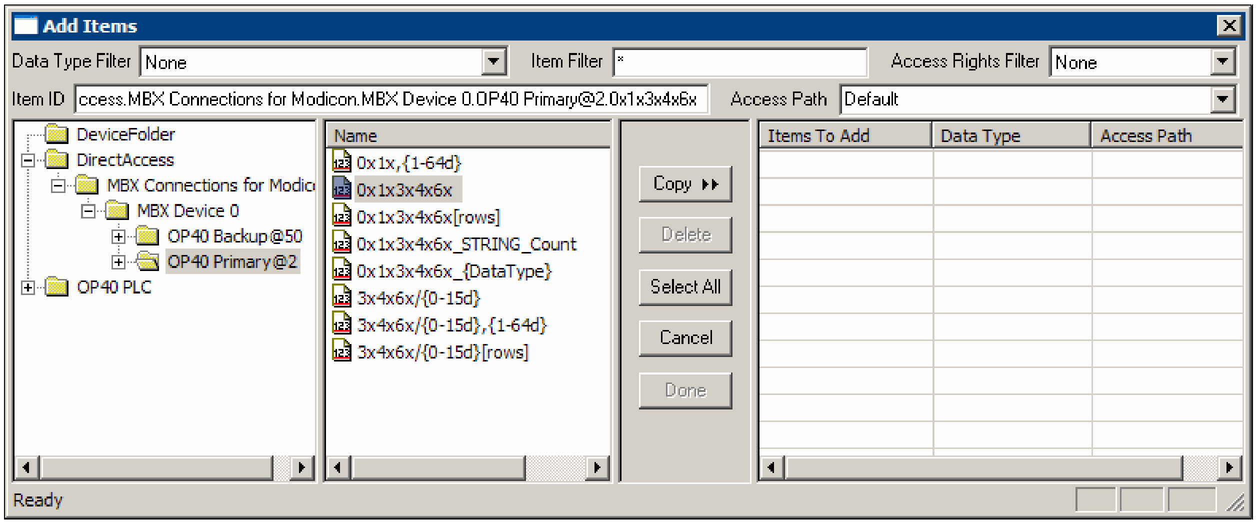
Performance Monitor

The Performance Monitor is a Microsoft diagnostic tool that the Cyberlogic drivers support. Its use is described in detail in the Cyberlogic OPC Server Help.

DirectAccess

At run time, in addition to the user-configured branches, the Cyberlogic OPC Server dynamically creates a branch called *DirectAccess* at the root of the Address Space Tree. The DirectAccess branch acts like one of the Device Folders and contains all of the configured Network Connections and Network Nodes. However, only Network Nodes that enable DirectAccess are present. OPC clients can then use this branch to access any register in any configured Network Node by directly specifying the register address.

For Network Nodes in the DirectAccess.MBX Connections for Modicon branch, the Cyberlogic OPC Server reports data items of each type found on the selected node. These are not valid item addresses. Rather, they are just hints for the user to help you specify a proper address.



In the example above, 0x1x3x4x6x is an address hint. This hint lets you know the form of address used to access data from coil bits (0x), input bits (1x), input registers (3x), holding registers (4x) and general registers (6x). You must replace the hint with the desired address.

Therefore, to access holding register 400007, you would edit the Item ID field at the top of the dialog box to read:

DirectAccess.MBX Connections for Modicon.MBX Device 0.OP40 Primary@2.400007

Here is a brief explanation of the other address hints:

0x1x,{1-64d}	This specifies a bit field and applies only to input and output bits. The {1-64d} hint means that you must specify the number of bits, in decimal form, in the range 1-64. For example, 000007,5 would specify a group of five bits, beginning with coil 000007.
0x1x3x4x6x[rows]	This form specifies an array of values, with [rows] specifying how many elements in the array. Thus, 600015[10] specifies an array of ten general registers, beginning with 600015.
3x4x6x/{0-15d}	This form specifies a bit within a register, with d indicating that the bit number is in decimal form. The specification 400002/1 would be the second-least significant bit of holding register 400002.
3x4x6x/{0-15d},{1-64d}	This specifies a field of bits within one or more registers. You must identify the first bit in the field by specifying the register and bit position, then you must specify the length of the field. Both the starting bit and the width are decimal numbers. For example, 300016/15,8 specifies a bit field that is eight bits wide, starting with the most significant bit of input register 300016. This means that it would include the first seven bits of 300017.
3x4x6x/{0-15d}[rows]	This form specifies an array of bits within registers. The specification 400007/0[5] would identify an array of five bits, each the least significant bit of holding registers 400007 – 400011.
3x4x6x_STRING_Count	You would use this form to specify a string of characters in one or more registers. Use 300010_STRING_6 to specify a string of six characters in input registers 300010 – 300015.

3x4x6x_{DataType}	This form tells the Server to provide the data for a register in a specific format. As an example, 400003_BCD16 would provide the value in holding register 400003, taken in binary-coded-decimal form.

The listed hints cover only the most common address formats. In fact, any item address in one of the following formats is valid:

- {Register Address}
- {Register Address}_{Data Type Override}

Register Address

The Register Address may be any address that is acceptable in the Address field of a Data Item. The PLC addressing requirements are discussed in Appendix A: PLC Addresses.

Data Type Override

This capability permits you to display the value in a format other than the native format of the register. For example, a register might normally contain binary data, but is being used to hold BCD data for a particular application. You could then specify that it should be treated as BCD. Any data type that is acceptable in the Data Type field of a Data Item may be specified as the override type. Notice that not all data types are valid for all register addresses. The following table shows all supported data types:

Data Type Override	Default Canonical Data Type	Description
BIT	VT_BOOL	1-bit boolean
SINT8	VT_I1	Signed 8-bit integer
UINT8	VT_UI1	Unsigned 8-bit integer
SINT16	VT_I2	Signed 16-bit integer
UINT16	VT_UI2	Unsigned 16-bit integer
SINT32	VT_I4	Signed 32-bit integer
UINT32	VT_UI4	Unsigned 32-bit integer
SINT64	VT_I8	Signed 64-bit integer
UINT64	VT_UI8	Unsigned 64-bit integer
FLOAT32	VT_R4	IEEE format 32-bit floating point number
FLOAT64	VT_R8	IEEE format 64-bit floating point number
BCD16	VT_UI2	BCD value in the range of 0 - 9999
BCD32	VT_UI4	BCD value in the range of 0 - 99999999
STRING{_Count}	VT_BSTR	Zero terminated ASCII string of 8-bit characters
WSTRING{_Count}	VT_BSTR	Zero terminated UNICODE string of 16-bit characters

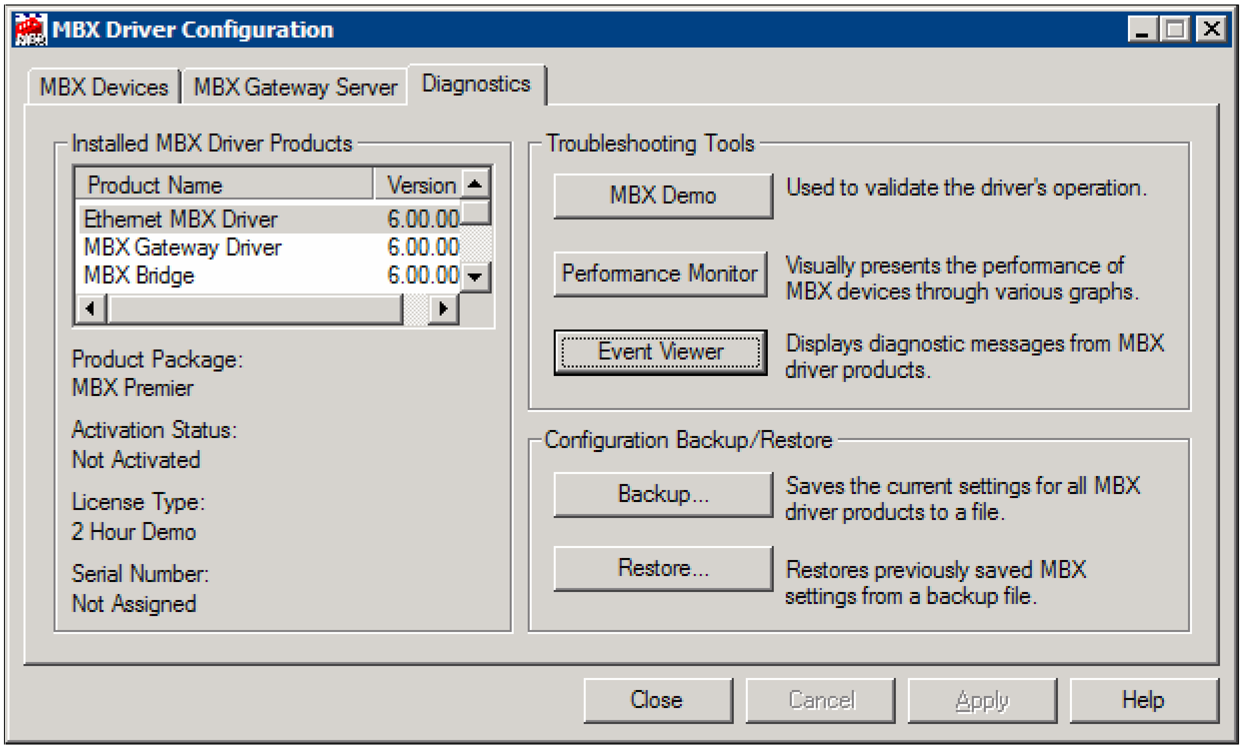
The STRING and WSTRING data types may require the Count value, which specifies the maximum number of characters in a string. By default, the Count value is 1.

DirectAccess allows read and write operations. However, for extra security, write operations are disabled by default. If writes are permissible, they can be enabled on a node-by-node basis as part of the configuration of Network Nodes.

Event Viewer

The Cyberlogic OPC Server may detect a run-time error. When it does, the Server sends an appropriate message to the Windows XP/2000/NT Event Logger. You can then view these messages using the following procedure:

1. From the Control Panel/Administrative Tools program group, run *Event Viewer*. Alternatively, an Event Viewer button is located in the Diagnostics tab of the MBX Driver Configuration Editor:



Product Name	Version
Ethernet MBX Driver	6.00.00
MBX Gateway Driver	6.00.00
MBX Bridge	6.00.00

Product Package:
MBX Premier

Activation Status:
Not Activated

License Type:
2 Hour Demo

Serial Number:
Not Assigned

Used to validate the driver's operation.

Visually presents the performance of MBX devices through various graphs.

Displays diagnostic messages from MBX driver products.

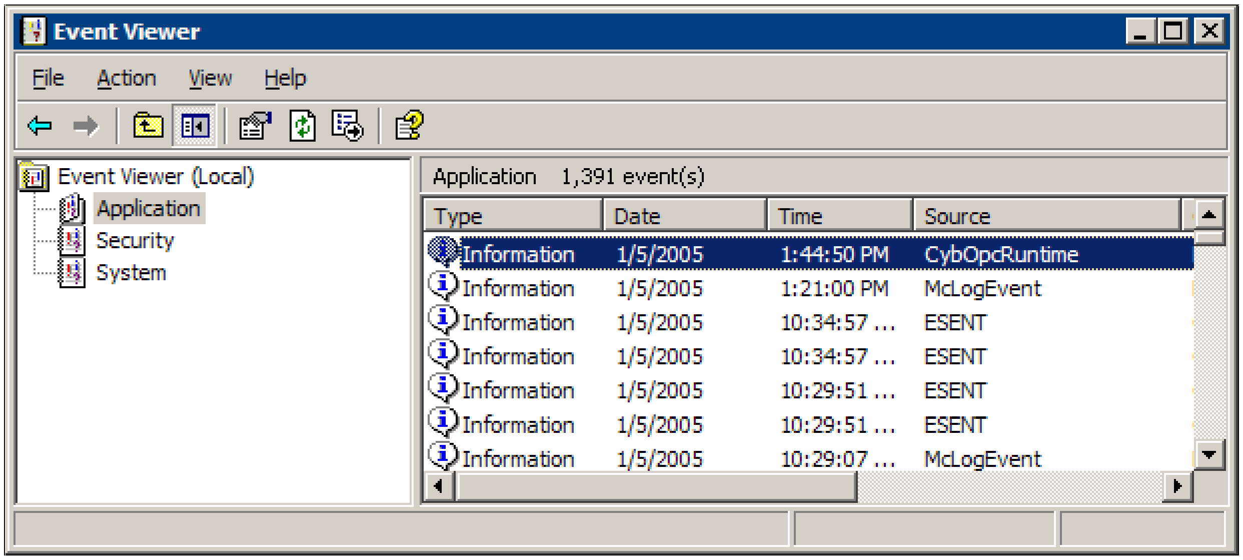
Saves the current settings for all MBX driver products to a file.

Restores previously saved MBX settings from a backup file.

2. Select *Application* from the Log menu or from the tree, depending upon your operating system.

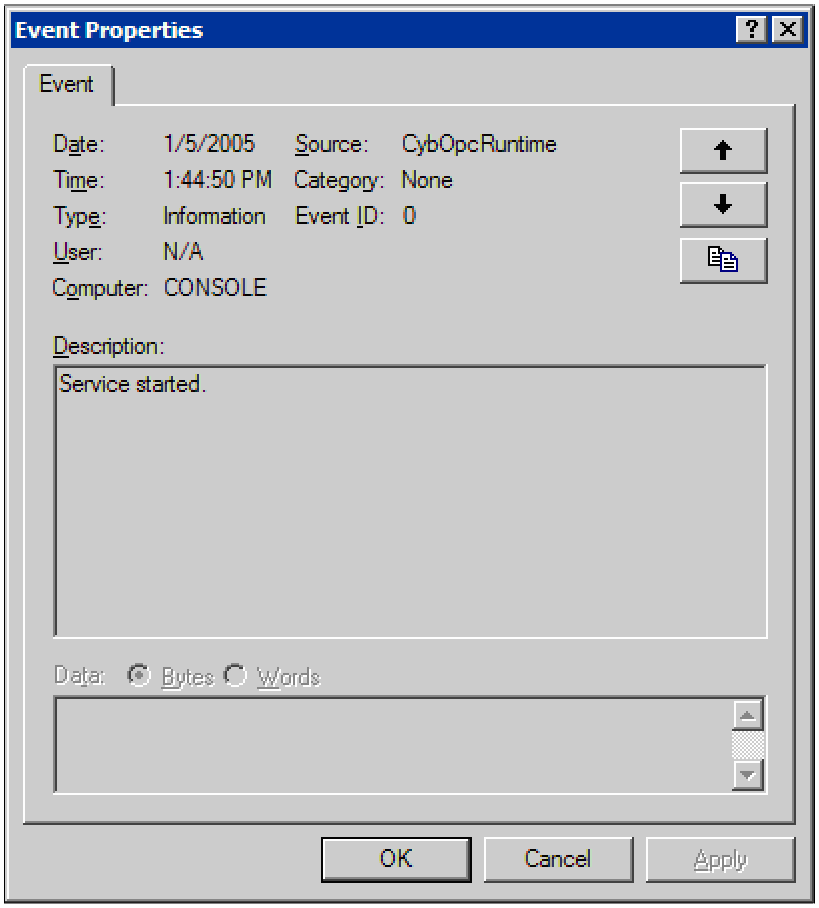
3. Look for entries with *CybOpcRuntime* or *MBXDriverClassSrv* or *LMBXApi* in the Source column.

Caution:	The Event Viewer does not clear itself after the system reboots. Check the time-stamps of the messages to make sure you are not looking at an old error message.



Type	Date	Time	Source
Information	1/5/2005	1:44:50 PM	CybOpcRuntime
Information	1/5/2005	1:21:00 PM	McLogEvent
Information	1/5/2005	10:34:57 ...	ESENT
Information	1/5/2005	10:34:57 ...	ESENT
Information	1/5/2005	10:29:51 ...	ESENT
Information	1/5/2005	10:29:51 ...	ESENT
Information	1/5/2005	10:29:07 ...	McLogEvent

4. Double-click on the selected entry to display the complete event message as seen below.



Date: 1/5/2005 Source: CybOpcRuntime
Time: 1:44:50 PM Category: None
Type: Information Event ID: 0
User: N/A
Computer: CONSOLE

Description:
Service started.

5. For further descriptions of the error log messages, refer to the MBX Driver Agent Messages section.

MBX Driver Agent Messages

This section shows Error Log messages that can be generated by the MBX driver agent module (*MBXDriverClassSrv* in the Source column). The main Cyberlogic OPC Server module can also log error messages (*CybOpcRuntime* in the Source column). For a list of these messages, refer to the Cyberlogic OPC Server Help.

Errors

Registration DLL failed to load. The I/O operations of the server have been disabled. Reinstall the product.

The installation program should have copied this DLL into the Windows system32 directory. Reinstall the product.

Registration verification failed. The I/O operations of the server have been disabled. Reinstall the product.

The Server could not access the registration information, which is gathered and stored during the installation process. Reinstall the product.

Memory allocation error in <function name>. Close some applications. Add more memory to your system. Contact the manufacturer's technical support.

The specified function failed to allocate the needed memory. This is a fatal error. If you are running low on memory, close some applications or add more memory to your system. If the problem continues, contact technical support for more information on a possible solution.

Unexpected error in <function name>. Please contact the manufacturer's technical support.

Indicates a possible programming bug in the Server. Contact technical support for more information on a possible solution.

Unexpected error in <function name> (Error code = <error code>). Please contact the manufacturer's technical support.

Indicates a possible programming bug in the Server. Contact technical support for more information on a possible solution.

Warnings

Memory allocation error in <function name>. The server may not operate correctly. Close some applications. Add more memory to your system. Contact the manufacturer's technical support.

The specified function failed to allocate the needed memory. The server will continue to operate, but some functions may not work. If you are running low on memory, close some applications or add more memory to your system. If the problem continues, contact technical support for more information on a possible solution.

This is a promotional copy of the MBX OPC Server. The server will operate for <time limit> hours.

This is a time-limited version of the Cyberlogic OPC Server. For information on upgrading to the full version, contact your Cyberlogic Software, Inc. distributor or visit Cyberlogic's website at www.cyberlogic.com.

Informational

This is a promotional copy of the MBX OPC Server. The allowed operation time has expired. The I/O operations of the server have been disabled.

This is a time-limited version of the MBX OPC Server and the allowed operation time has expired. For information on upgrading to the full version, contact your Cyberlogic Software, Inc. distributor or visit Cyberlogic's website at www.cyberlogic.com.

Frequently Asked Questions

For FAQs common to all driver agents refer to the Cyberlogic OPC Server Help.

I have a PCI-85 card installed in my system, but when I run the MBX demo it doesn't show any other nodes on the network. I know that the network cable is good, because I tested it with another computer.

Plug-and-Play adapter cards, such as the PCI-85, default to node address 1. If there is another one of these cards anywhere on your network, it is likely that it is using this default address, resulting in a conflict. Use the MBX Configuration Editor to change the adapter's node address to a different value.

I have an SA-85 card connected to the Modbus Plus network. I successfully installed the MBX OPC Server, but when I try the auto configuration, the editor fails to detect any network connections. What is the problem?

For the auto configuration to work you must first create at least one MBX device. To do that, at least one of the following MBX driver products must be installed: MBX Driver, Ethernet MBX Driver, Serial MBX Driver or MBX Gateway Driver. (In your case, you should install at least the MBX Driver.) Refer to the driver-specific help file for information on creating MBX devices. Be sure the driver is running. The drivers default to automatic operation when they are installed, but may have been switched to manual. This selection is done from the driver configuration editor.

APPENDIX A: PLC ADDRESSES

The allowed syntax for the Address field on the Data Tab of a Data Item includes all register address notations supported by the various Modicon products. In addition, the MBX driver class extends this syntax when it comes to specifying arrays, array elements, bit addresses and bit fields.

Standard (400001) Notation

In the standard notation the first digit identifies the register type and the following 5 (or 4) digits represent the register number:

0xxxxx – Coil

1xxxxx – Input

3xxxxx – Input register

4xxxxx – Holding register

6xxxxx – General register

Here are a few examples of register addresses in this notation:

400001	First holding register
300002	Second input register

Separator (4:00001) Notation

In the separator notation the first digit identifies the register type followed by a colon character and a 5-digit number representing the register number:

0:xxxxx – Coil

1:xxxxx – Input

3:xxxxx – Input register

4:xxxxx – Holding register

6:xxxxx – General register

Here are a few examples of register addresses in this notation:

4:00001	First holding register
3:00002	Second input register

Compact (4:1) Notation

In the compact notation the first digit identifies the register type followed by a colon character and the register number without leading zeros:

0:xxxxx – Coil

1:xxxxx – Input

3:xxxxx – Input register

4:xxxxx – Holding register

6:xxxxx – General register

Here are a few examples of register addresses in this notation:

4:1	First holding register
3:2	Second input register

IEC (QW00001) Notation

In the IEC notation the first two letters identify the register type and the following 5 digits represent the register number:

QXxxxxx – Coil

IXxxxxx – Input

IDxxxxx – Input register

QWxxxxx – Holding register

Here are a few examples of register addresses in this notation:

QW00001	First holding register
ID00002	Second input register

Global Data (G01) Notation

Global data notation consists of a G followed by two decimal digits in the range of 01 – 32. All global data is handled as 16-bit registers.

Gxx – Global data register.

Here are a few examples of register addresses in this notation:

G01	First global data register
G32	Last global data register

If you use global data and you want to read or write the OPC Server's global data, you must create a node of type *This Node – Global Data*. This will then allow you to create global data type Data Items in the Address Space Tree and connect them to the local node. For more information on configuring network nodes, refer to the Network Nodes section.

Arrays

Single-dimensional arrays of most Data Types are supported with the exception of strings and bit fields. In order to specify an array of Data Type elements, use the following syntax:

{Valid Register Address}**[**{Number of Elements}**]**

or

{Valid Register Address}**[**{Number of Elements}**,**{Lower Bound}**]**

The Lower Bound specifies the array index value for the first element in an array. If not specified, the Lower Bound defaults to zero. Visual Basic applications may expect the first index to be equal to a one, in which case you would set the Lower Bound value to a 1. Here are a few examples of arrays:

400001[5]	Array of five registers starting from address 400001 and a lower bound of 0
0:1[4,1]	Array of four bits starting from address 000001 and a lower bound of 1.
4:00010/0[16]	Array of sixteen bits starting from bit 400010/0 and a lower bound of 0

Array Elements

The 16-bit registers may be viewed as arrays of two bytes. In order to specify an element of such array, use the following syntax:

{Valid Register Address}**(**{ Element Index}**)**

Here are a few examples of array elements:

400001(0)	First byte of register 400001
300001(1)	Second byte of register 300001

Notice that it is possible to combine this specification with the array syntax to specify an array of 8-bit bytes beginning at a specific byte. For example:

400001(1)[10]	Array of 10 bytes starting from the second byte of 400001

Register Bits

To specify a specific bit within a 16-bit register, use the following syntax:

{Valid Register Address}**/**{Bit Number}

The Bit Number is in the range of 0-15. Here are a few examples of valid bit addresses:

400001/1	Bit #1 in register 400001
300001/15	Bit #15 in register 300001

Bit Fields

To specify a sequence of bits as a bit field, use the following syntax:

{Valid Bit Address}**,**{Bit Count}

Here are a few examples of bit fields:

400001/1,5	Bit field of five bits starting from bit 400001/1
100001,16	Bit field of sixteen bits starting from bit 100001

APPENDIX B: DATA ITEM DUPLICATION WIZARD

To speed up the creation of similarly configured Data Items, you can easily create multiple Data Items by duplicating an existing Data Item. To do this, select an existing Data Item and then select *Duplicate…* from the Edit menu (or right-click on the Data Item and select *Duplicate…* from the context menu) and the Duplicate Data Item Wizard will open.

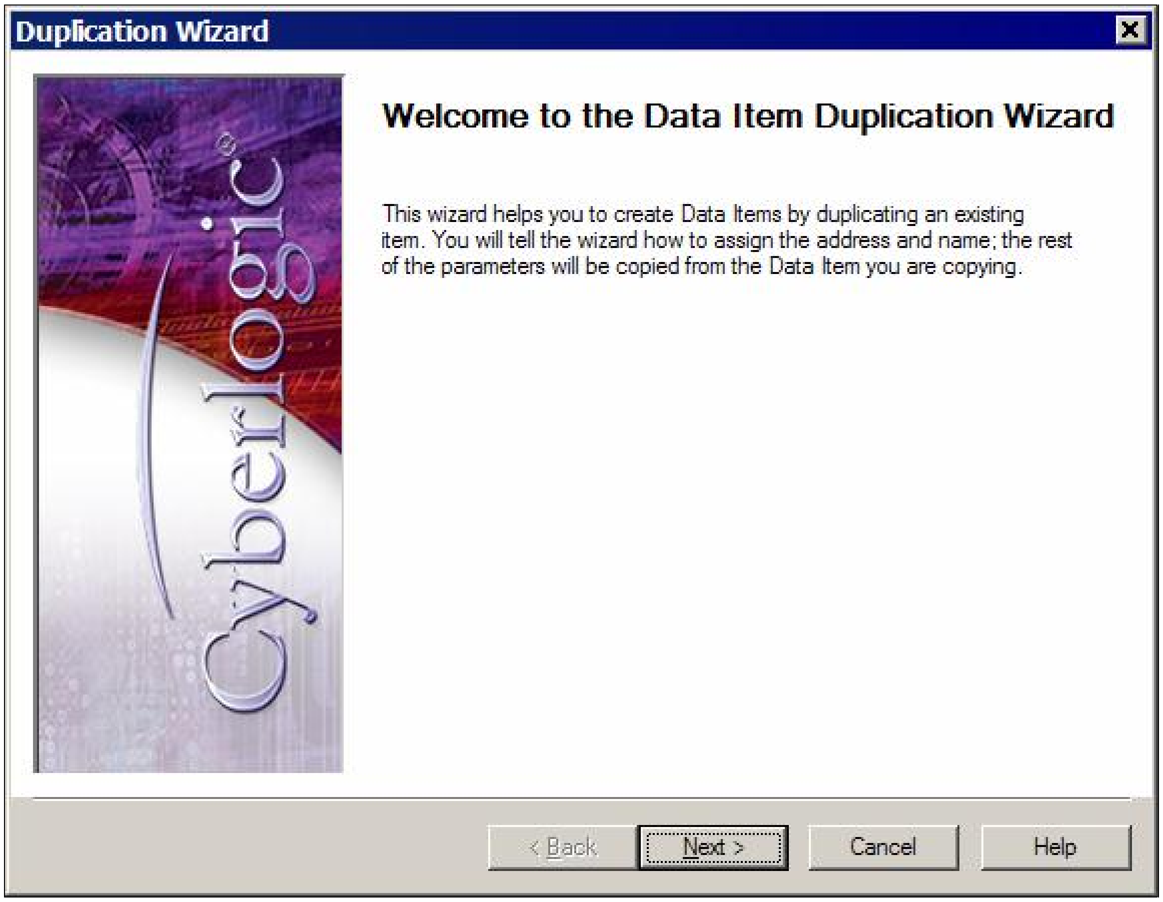
We will present two examples here.

Example 1: Simple Addressing show how to duplicate input bits. This same procedure would be used for other data using simple addressing.

Example 2: Addressing Bits Within Registers is more complex and shows how to handle data addressed as bits within a register or array.

Example 1: Simple Addressing

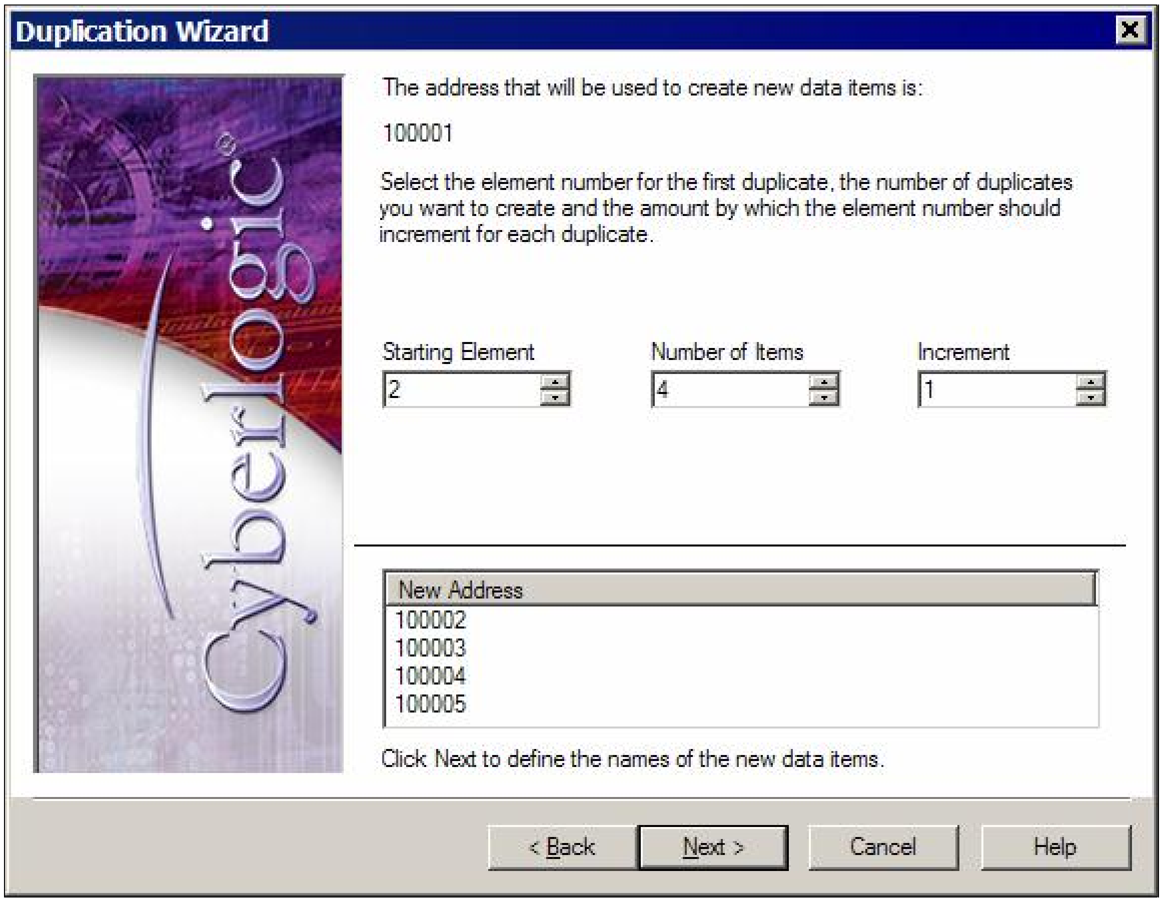
For this example, we selected a Data Item called 100001. On the Welcome screen, click the *Next* button to continue.



Welcome to the Data Item Duplication Wizard

This wizard helps you to create Data Items by duplicating an existing item. You will tell the wizard how to assign the address and name; the rest of the parameters will be copied from the Data Item you are copying.

On this screen, you will specify the duplicate Data Items you want to create. Notice that the screen tells you which Data Item you are duplicating. Enter *2* for the Starting Element, *4* for the Number of Items and *1* for the Increment. The New Address box shows you the Data Items that the wizard will create. Click *Next* to continue.



The address that will be used to create new data items is:

100001

Select the element number for the first duplicate, the number of duplicates you want to create and the amount by which the element number should increment for each duplicate.

Starting Element: 2

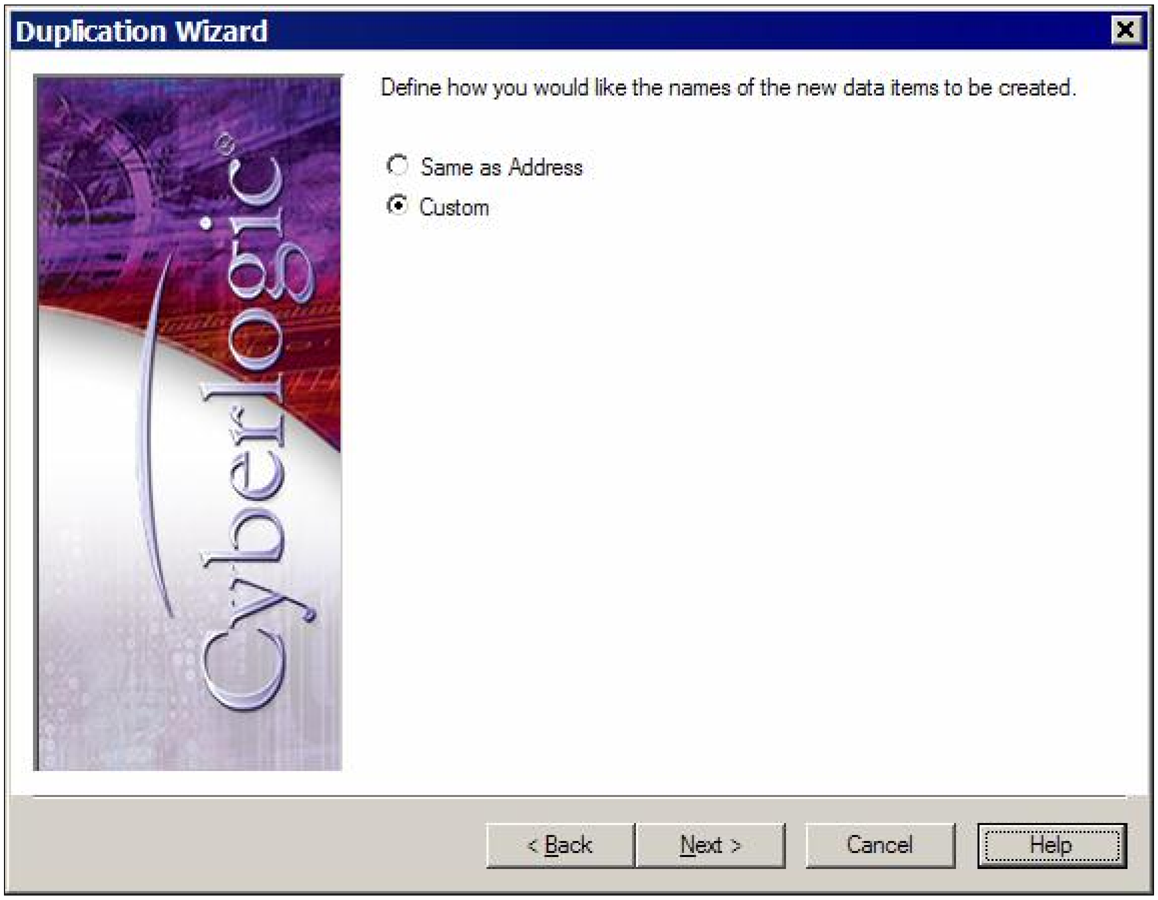
Number of Items: 4

Increment: 1

New Address
100002
100003
100004
100005

Click Next to define the names of the new data items.

You must now decide how you wish to name the Data Items you will create. You may simply use the address as the name or you may create a custom naming scheme. Select *Custom* and click *Next* to continue.

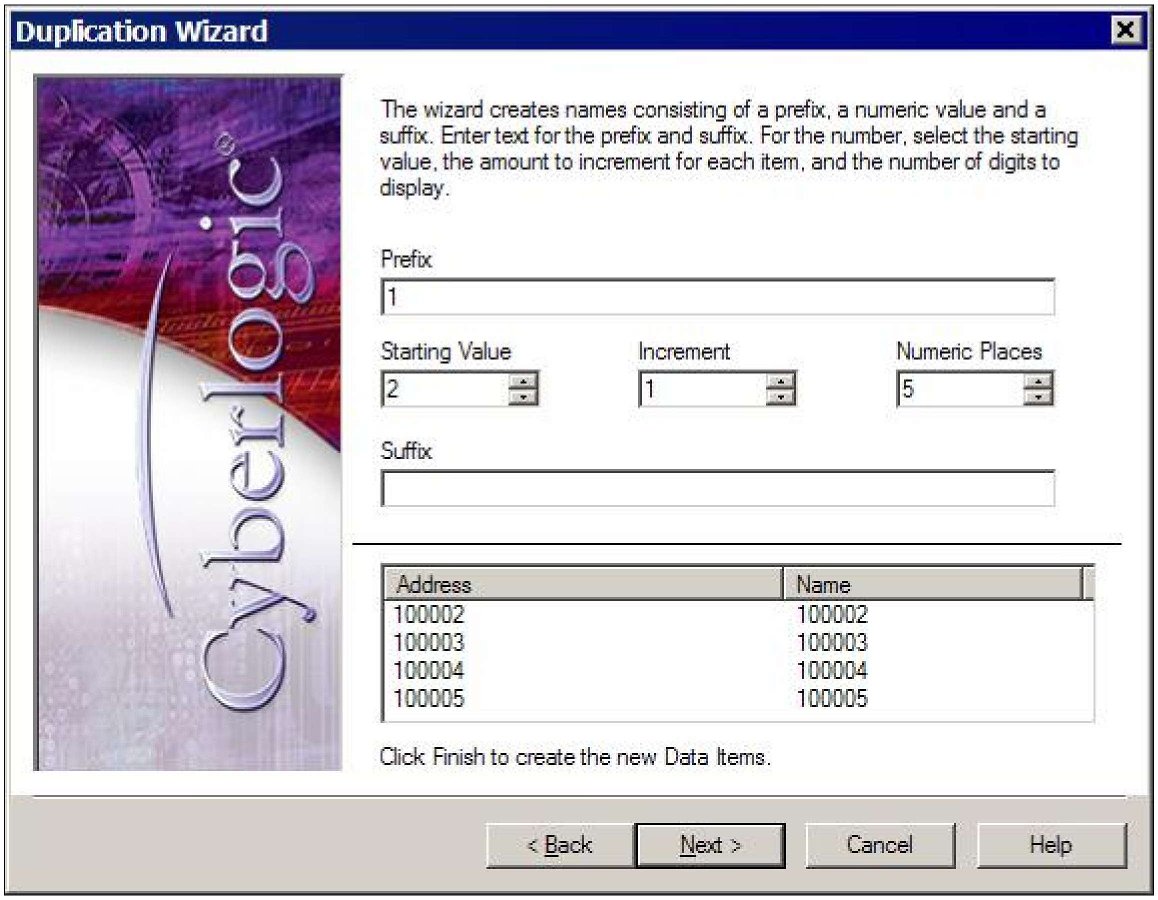


Define how you would like the names of the new data items to be created.

Same as Address [✓ Custom]

The wizard will create names for the Data Items for you. These names will consist of a prefix, a numeric value and a suffix. The first Data Item we created was named 100001 and we would like the duplicates to have names of the same style. Enter *1* in the Prefix field. The next three fields define the numeric values to be used. Enter *2* as the Starting Value, *1* for the Increment and *5* for Numeric Places. This causes the duplicates to be consecutively numbered, beginning at 2. It also forces the names to use six digits, inserting leading zeros as needed. No suffix is necessary for this naming scheme, so leave the Suffix field blank. The lower window will show you the names that will be used for each Data Item.

Click *Next* to continue.



The wizard creates names consisting of a prefix, a numeric value and a suffix. Enter text for the prefix and suffix. For the number, select the starting value, the amount to increment for each item, and the number of digits to display.

Prefix: 1

Starting Value: 2

Increment: 1

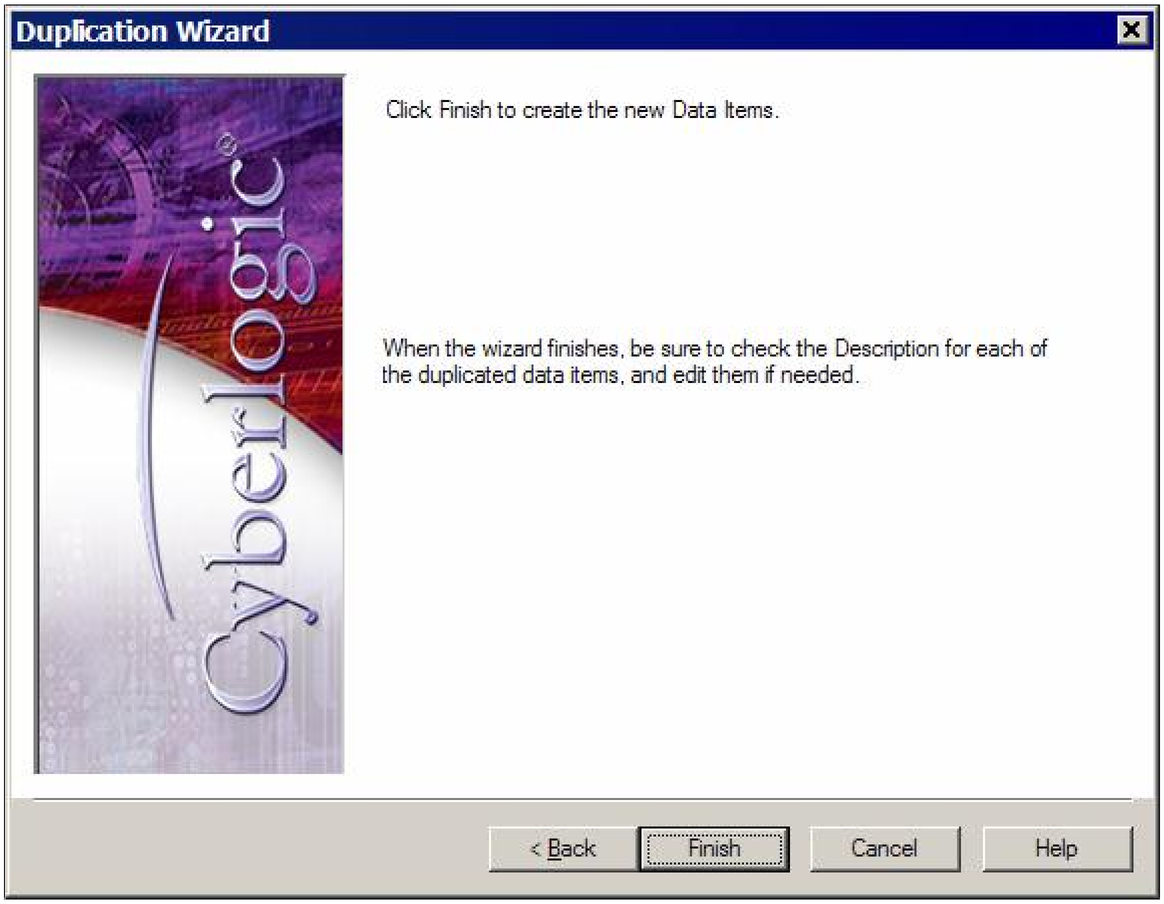
Numeric Places: 5

Suffix:

Address	Name
100002	100002
100003	100003
100004	100004
100005	100005

Click Finish to create the new Data Items.

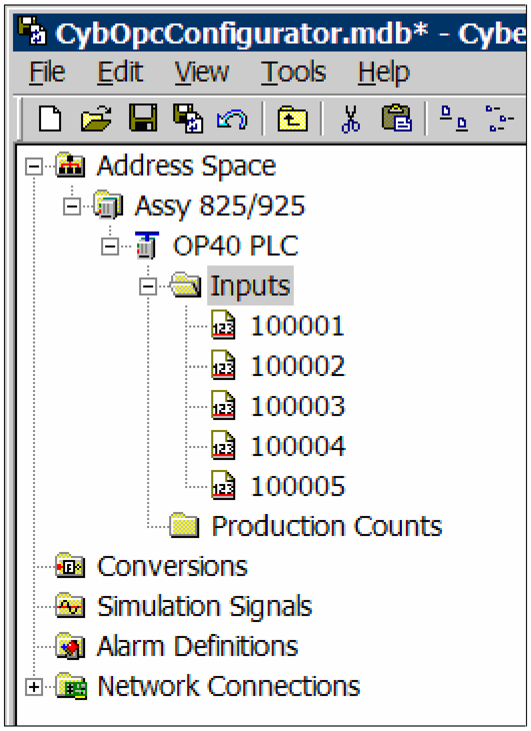
Click *Finish* to create the duplicate Data Items and exit the wizard.



Click Finish to create the new Data Items.

When the wizard finishes, be sure to check the Description for each of the duplicated data items, and edit them if needed.

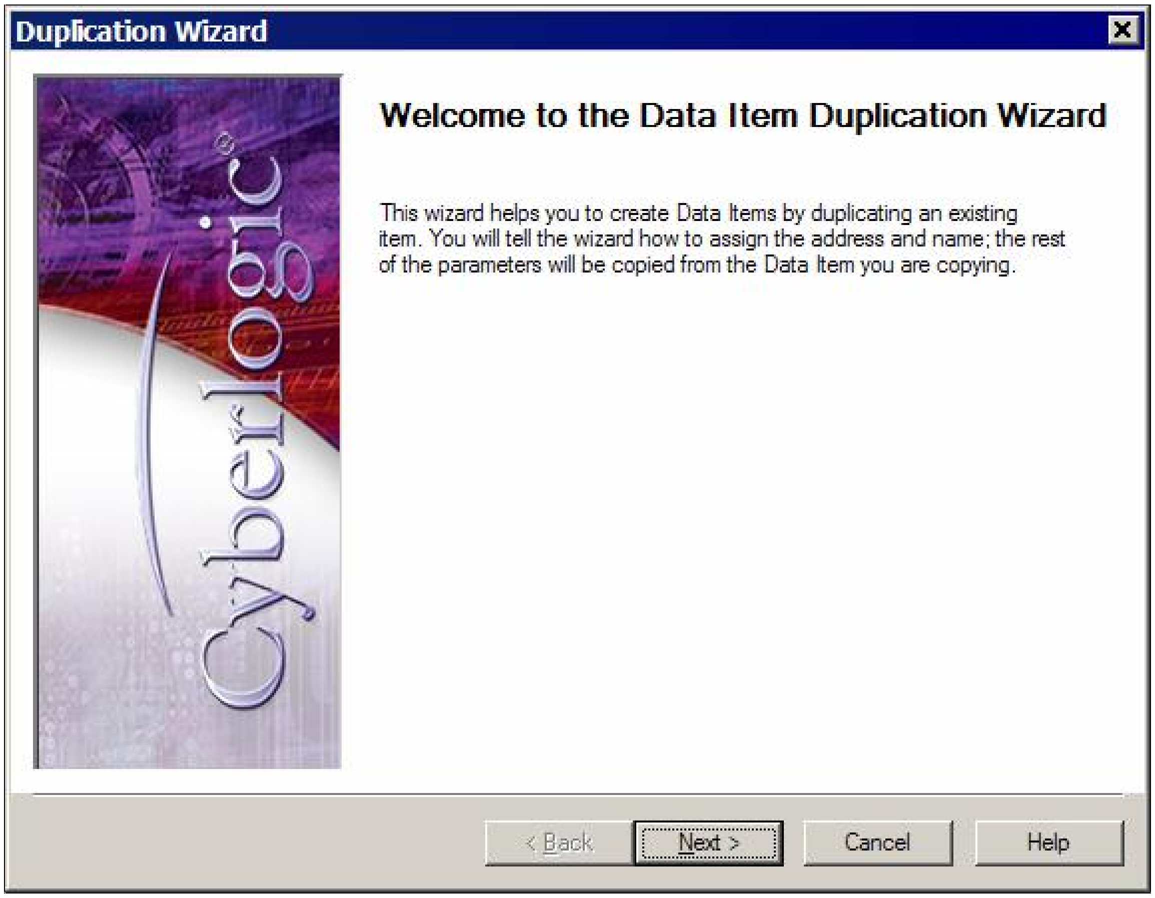
Your duplicates will be created. You will find, however, that the Description field for each duplicate is the same as the original. You may wish to edit these descriptions. You should also check the other parameters to verify that they are set properly.



By using the Duplication Wizard, you are able to create multiple Data Items using just a short, simple procedure. This method is very useful when creating similarly configured Data Items.

Example 2: Addressing Bits Within Registers

For this example, we selected a Data Item called StatusItem3. It is a bit within a word and is addressed as 400015/3,1. We want to address additional bits within this same word. On the Welcome screen, click the *Next* button to continue.

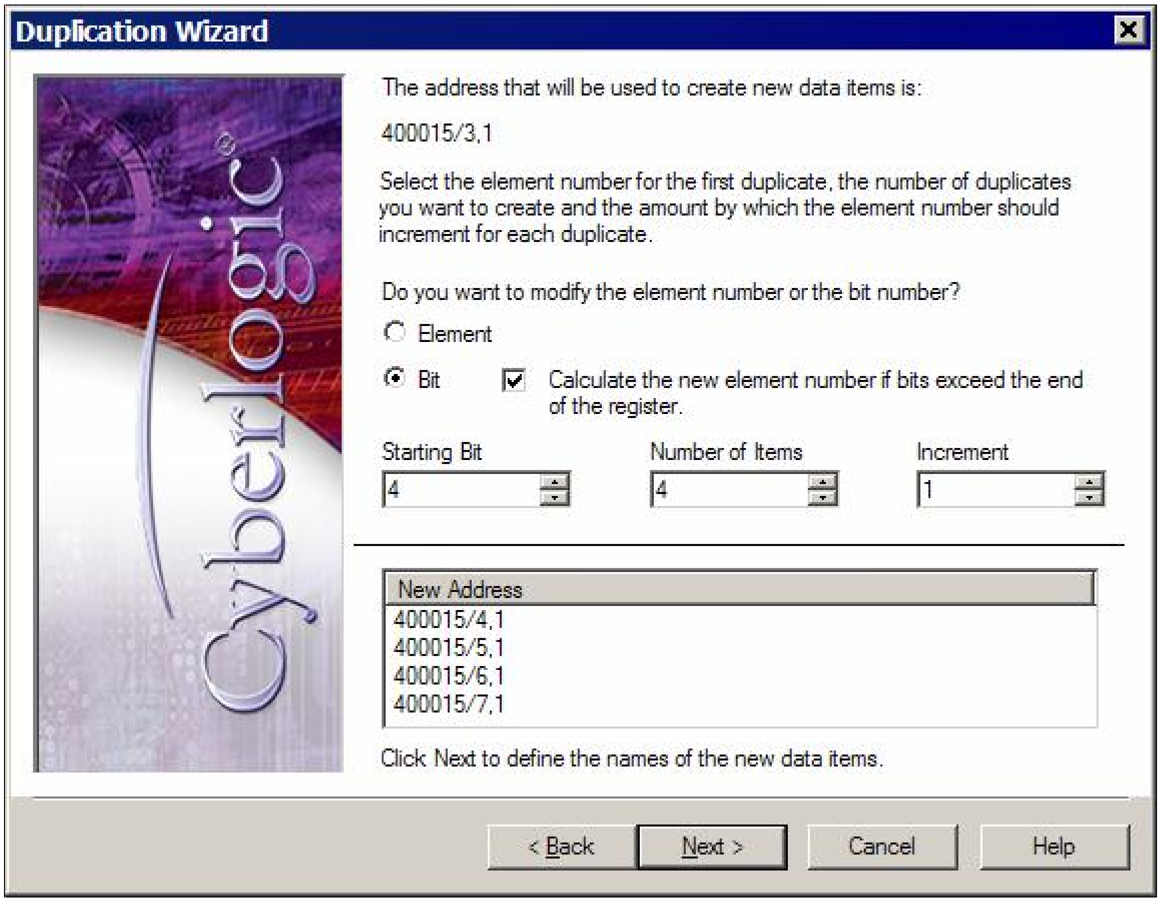


Welcome to the Data Item Duplication Wizard

This wizard helps you to create Data Items by duplicating an existing item. You will tell the wizard how to assign the address and name; the rest of the parameters will be copied from the Data Item you are copying.

Select *Bit* to indicate that you want to address additional bits within the same register, rather than additional registers. Enter *4* for the Starting Bit value to specify that the first duplicate should be 400015/4,1. Enter *4* for the Number of Items, to create four duplicates. Enter *1* for the Increment so that you will access consecutive bits.

The wizard will show the new addresses that will be accessed. Click *Next* to continue.



The address that will be used to create new data items is:

400015/3,1

Select the element number for the first duplicate, the number of duplicates you want to create and the amount by which the element number should increment for each duplicate.

Do you want to modify the element number or the bit number?

Element [✓ Bit]

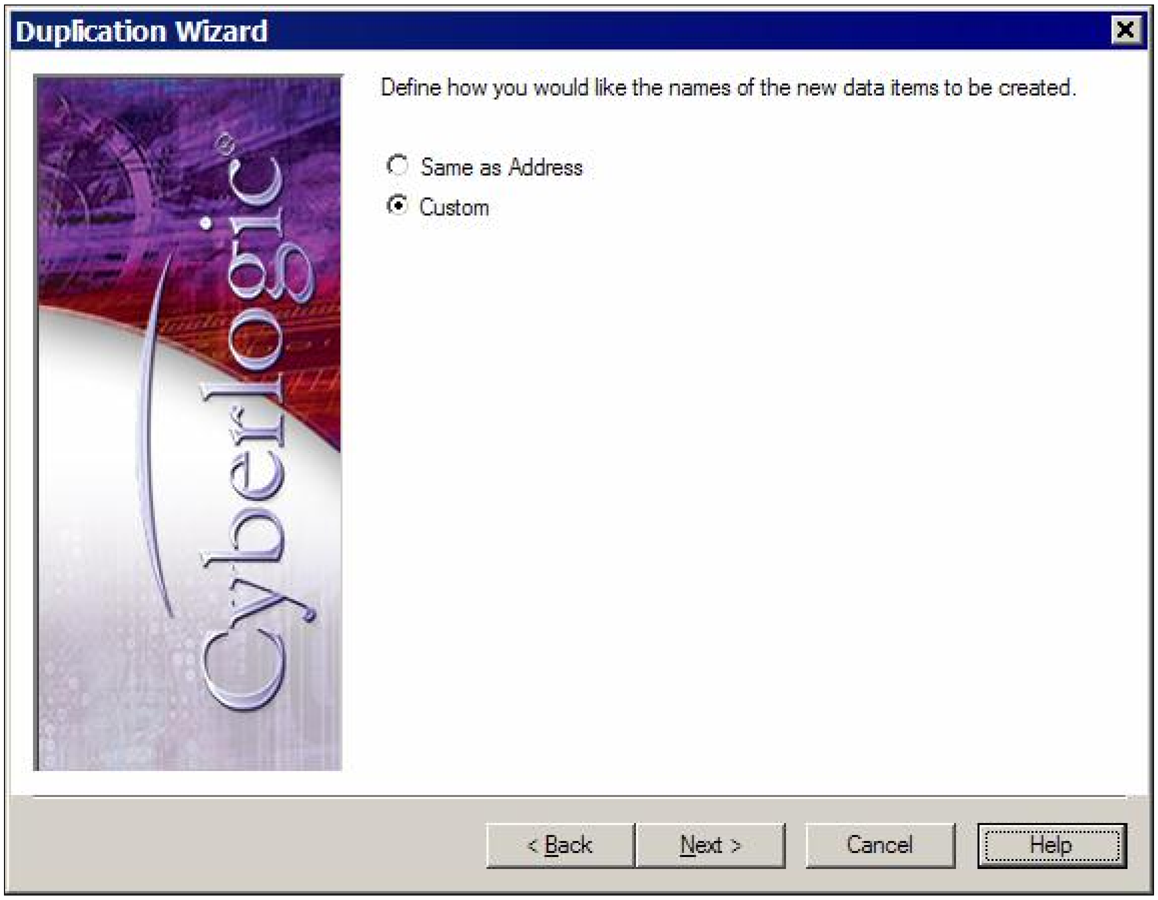
[✓] Calculate the new element number if bits exceed the end of the register.

Starting Bit: 4
Number of Items: 4
Increment: 1

New Address
400015/4,1
400015/5,1
400015/6,1
400015/7,1

Click Next to define the names of the new data items.

Select *Custom* to indicate that you want to specify names for the new Data Items, rather than simply using each item's address as its name. Click *Next* to continue.

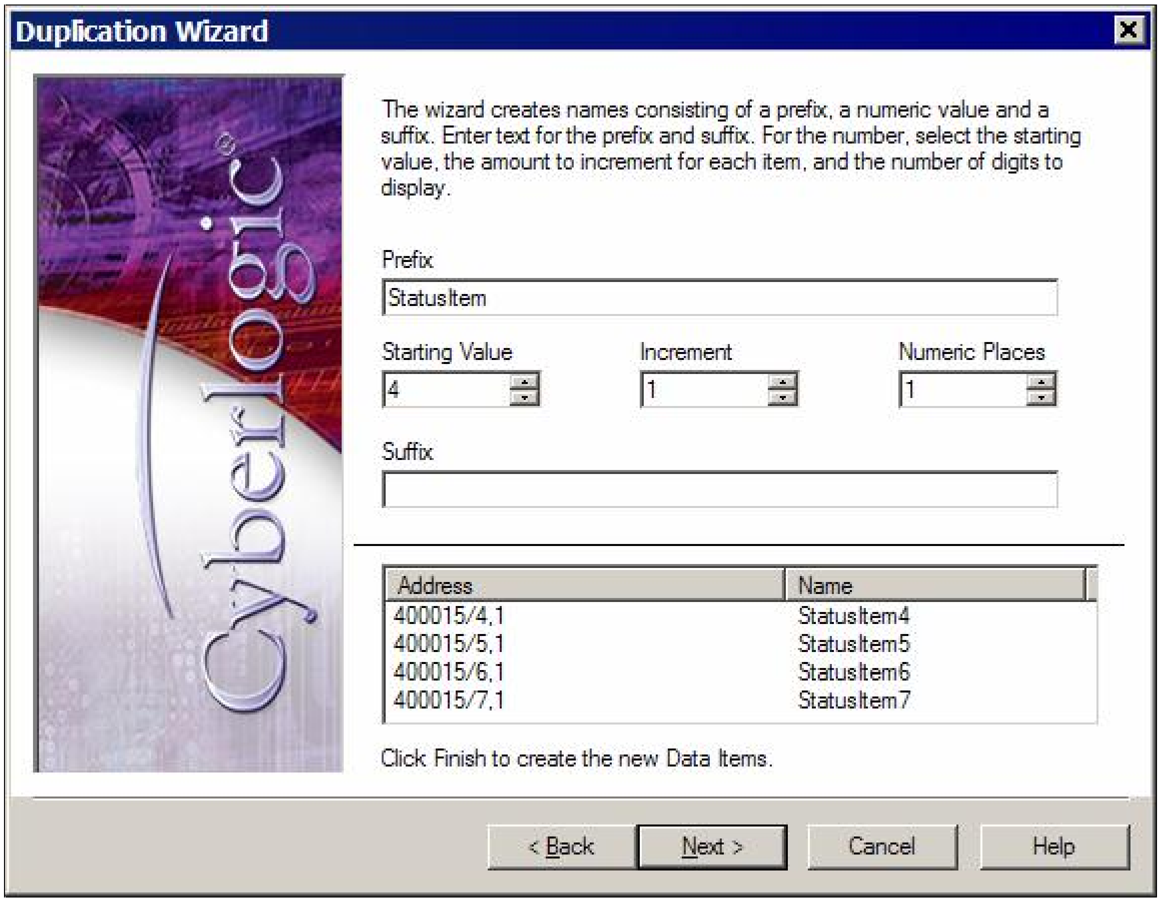


Define how you would like the names of the new data items to be created.

Same as Address [✓ Custom]

The name for each item will be built by concatenating a prefix, a number and a suffix. Enter *StatusItem* as the Prefix. To specify the number, select a Starting Value of *4* and an Increment of *1*, to number the items consecutively from 4 to 7. Select *1* for the Numeric Places because there is no need to force the use of leading zeros in the name. The Suffix field is optional and is not used in this example, so leave it blank.

The wizard will show you the names that will be assigned to each address. Click *Next* to continue.



The wizard creates names consisting of a prefix, a numeric value and a suffix. Enter text for the prefix and suffix. For the number, select the starting value, the amount to increment for each item, and the number of digits to display.

Prefix: StatusItem

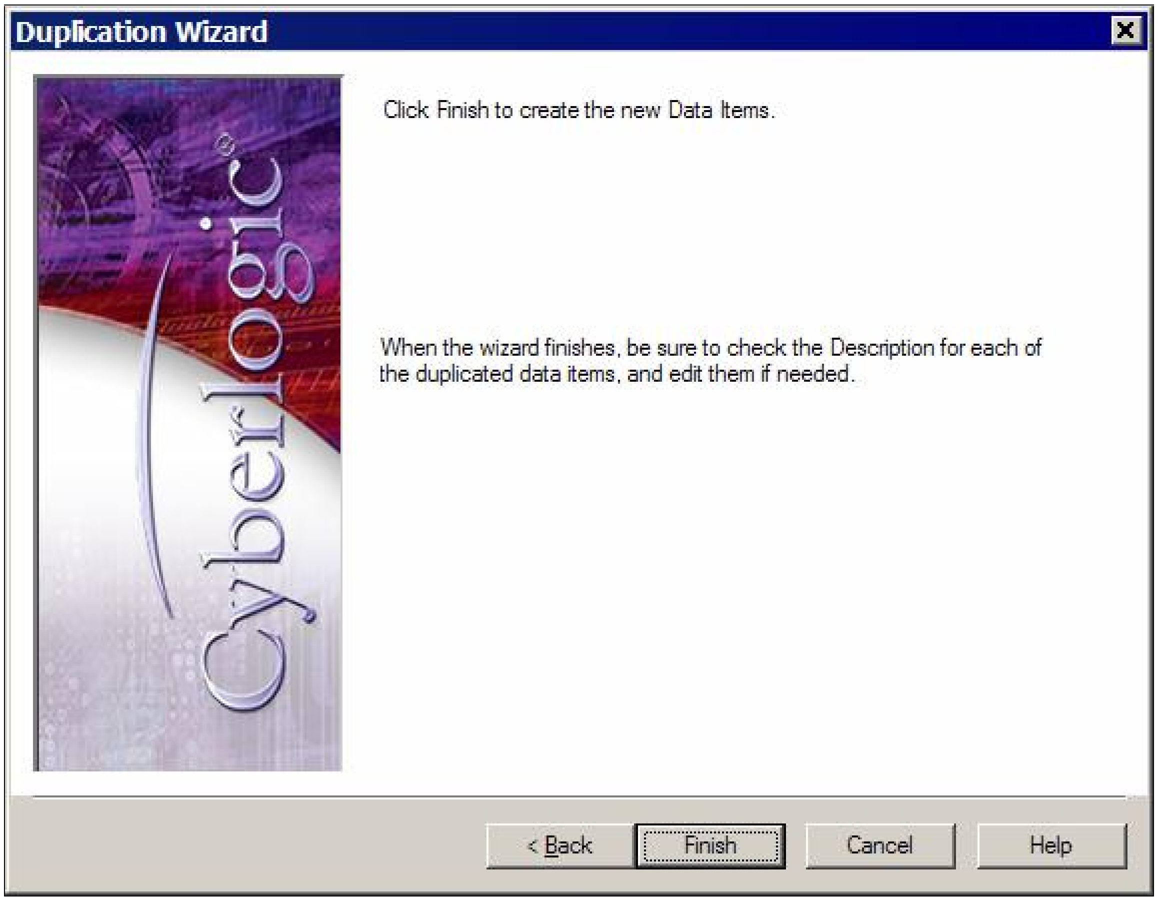
Starting Value: 4 Increment: 1 Numeric Places: 1

Suffix:

Address	Name
400015/4,1	StatusItem4
400015/5,1	StatusItem5
400015/6,1	StatusItem6
400015/7,1	StatusItem7

Click Finish to create the new Data Items.

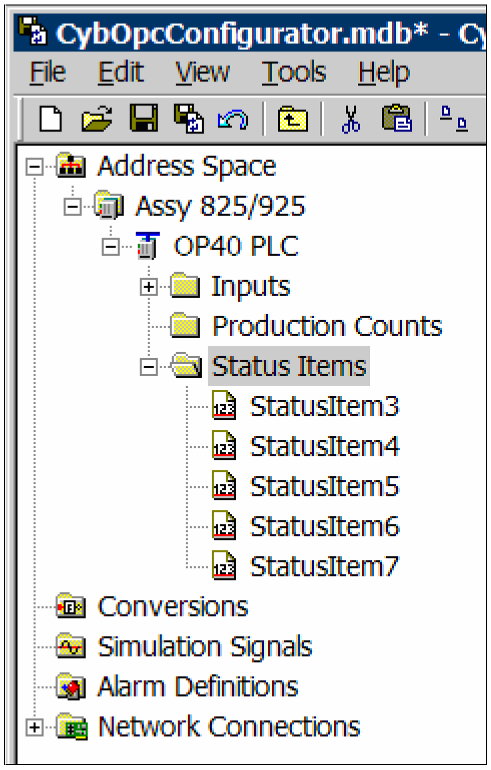
Click *Finish* to create the duplicate Data Items and exit the wizard.



Click Finish to create the new Data Items.

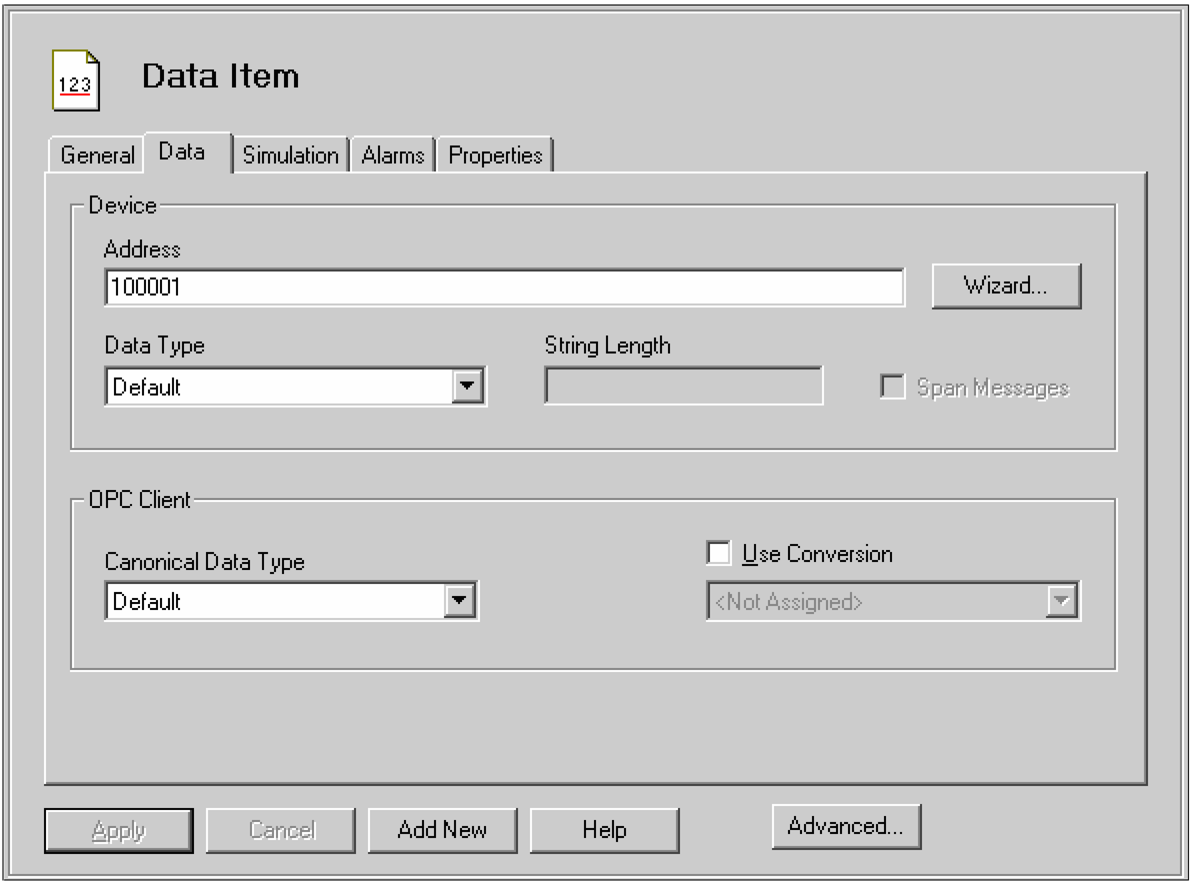
When the wizard finishes, be sure to check the Description for each of the duplicated data items, and edit them if needed.

Your duplicates will be created. Remember that the Description field for each duplicate is the same as the original. You may wish to edit these descriptions. You should also check the other parameters to verify that they are set properly.



APPENDIX C: ADDRESS WIZARD

The Address Wizard will help you to define the correct address for the data you want to access. You activate the Address Wizard by clicking the Wizard button on the Data tab of the Data Item dialog.



Device

Address: 100001

Data Type: Default

String Length:

Span Messages

OPC Client

Canonical Data Type: Default

Use Conversion: <Not Assigned>

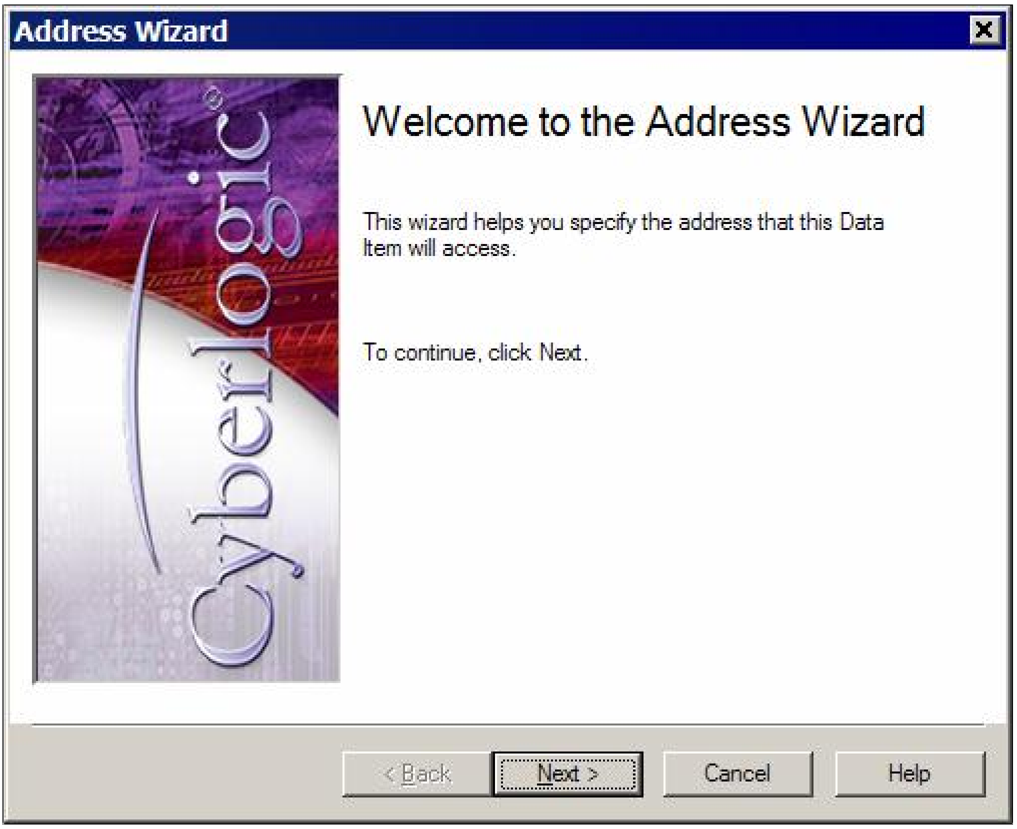
This appendix provides two examples of the use of the Addressing Wizard.

Example 1: Simple Addressing covers setting up a simple address for a single register.

Example 2: Arrays of Registers shows how to address registers as an array.

Example 1: Simple Addressing

This example shows how to set up the addressing for a single register. From the Welcome screen, click *Next* to continue.

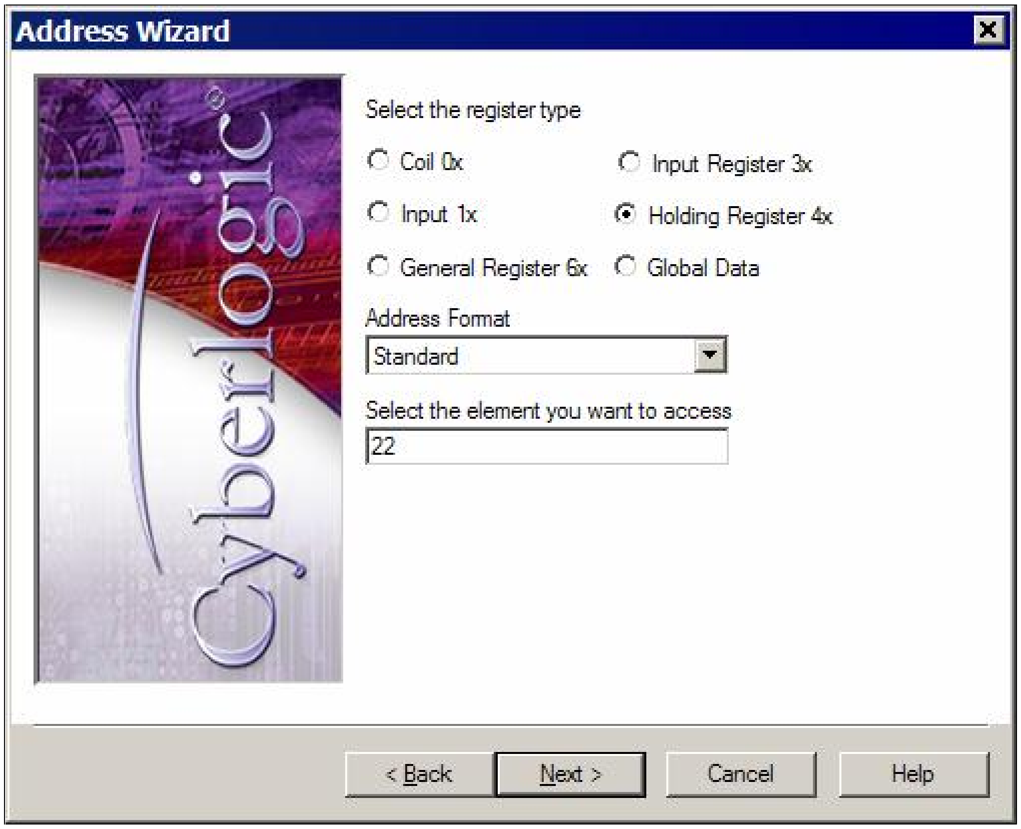


Welcome to the Address Wizard

This wizard helps you specify the address that this Data Item will access.

To continue, click Next.

On this screen, you must select the specific register you want to access. For this example, we chose holding register number 22, which we want to address using the standard format. Make your selections and click *Next* to continue.



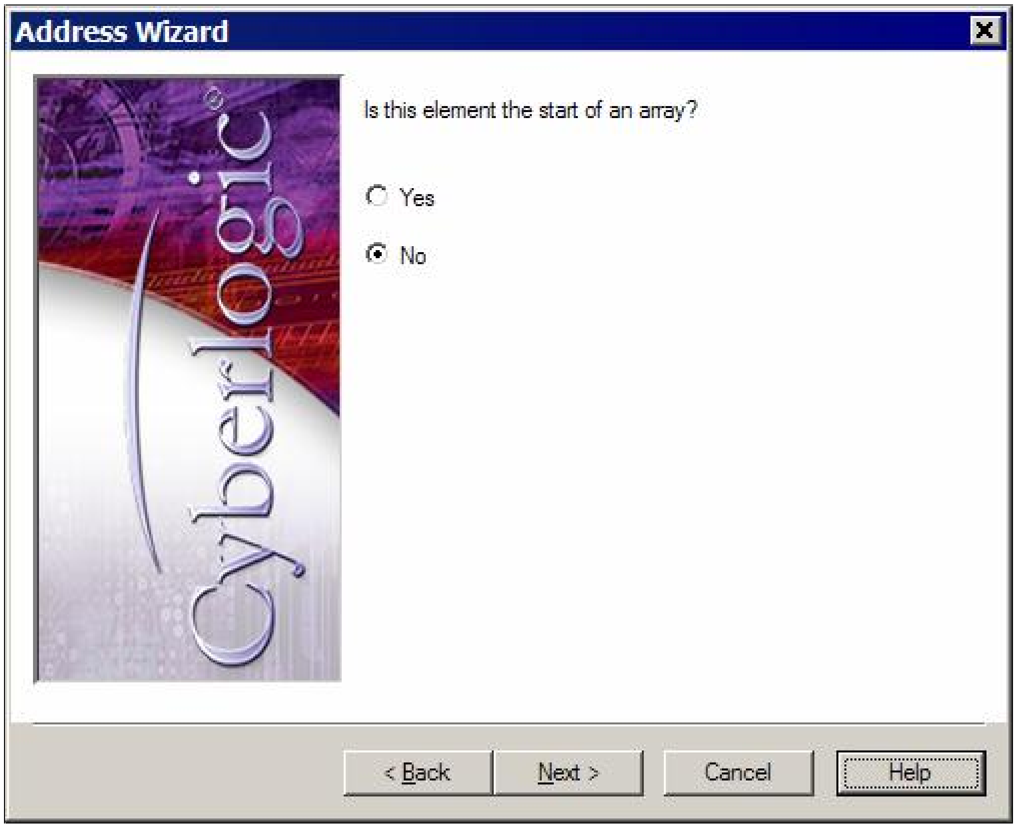
Select the register type

( ) Coil 0x
( ) Input Register 3x
( ) Input 1x
[✓ Holding Register 4x]
( ) General Register 6x
( ) Global Data

Address Format: Standard

Select the element you want to access: 22

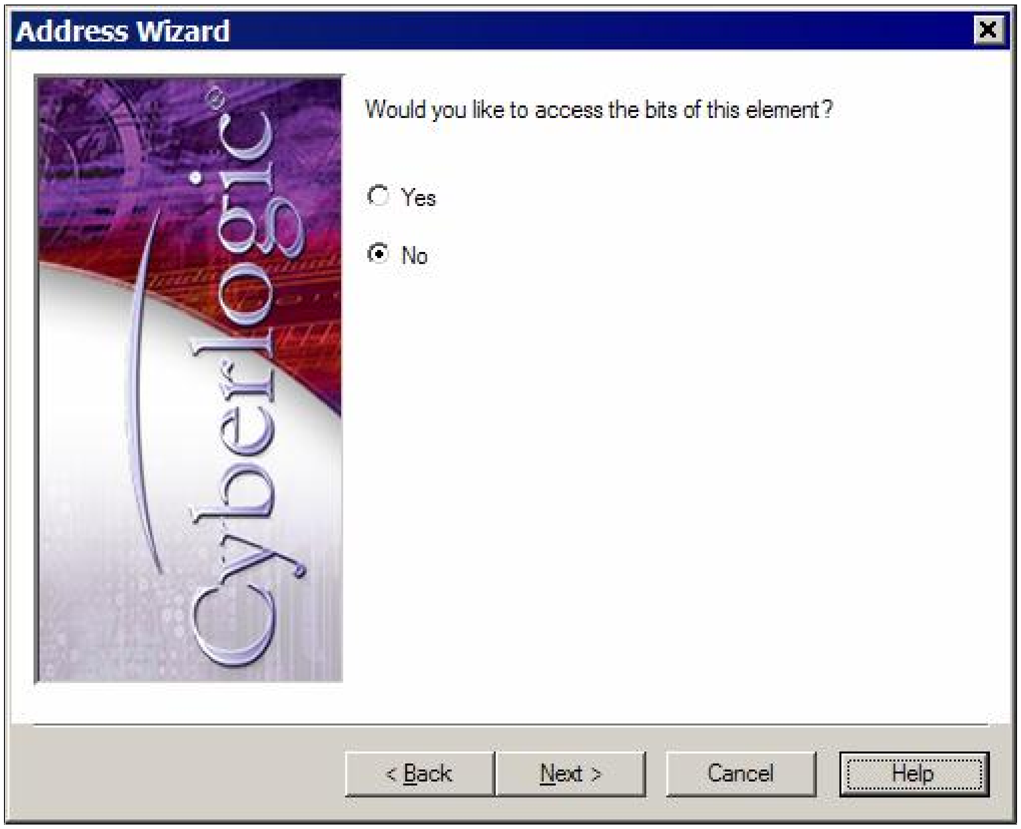
For this example, we do not want to access the data as an array, so select *No* and click *Next*.



Is this element the start of an array?

Yes [✓ No]

It is possible to access individual bits or groups of bits within the register. To access the register as a word, select *No* and then click *Next*.

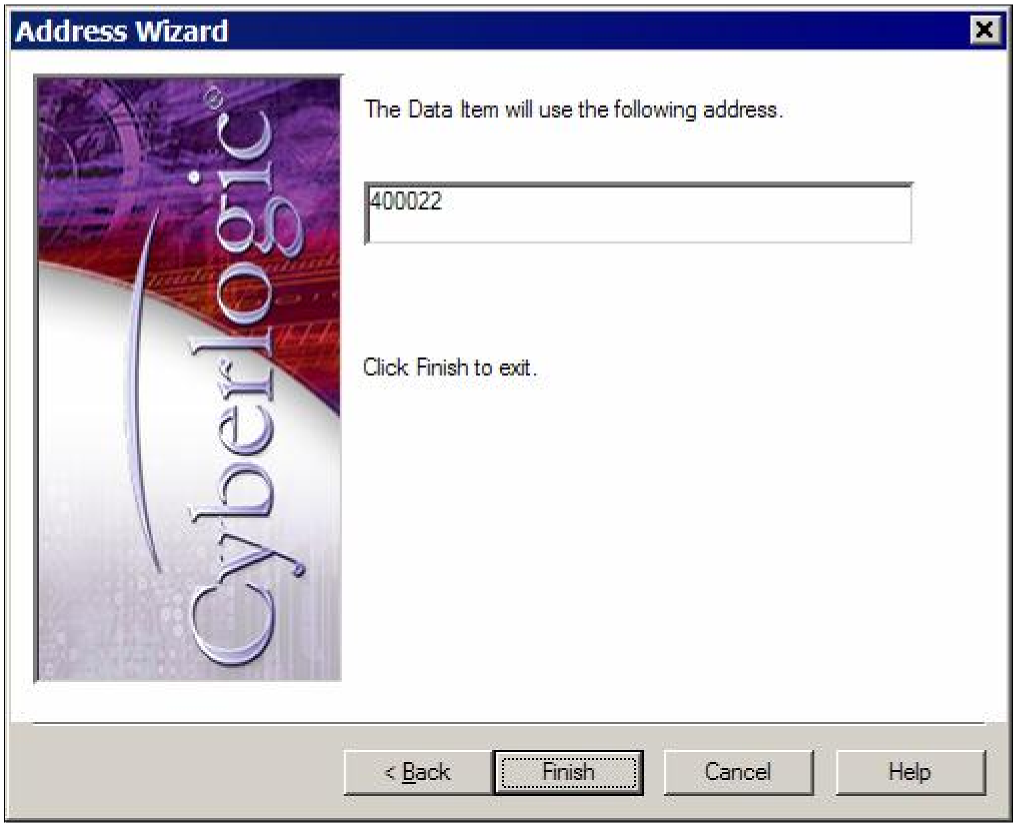


Address Wizard

Would you like to access the bits of this element?

Yes [✓ No]

This screen shows the result of your selections. The register address to be created is 400022. Click *Finish* to allow the wizard to complete and exit.



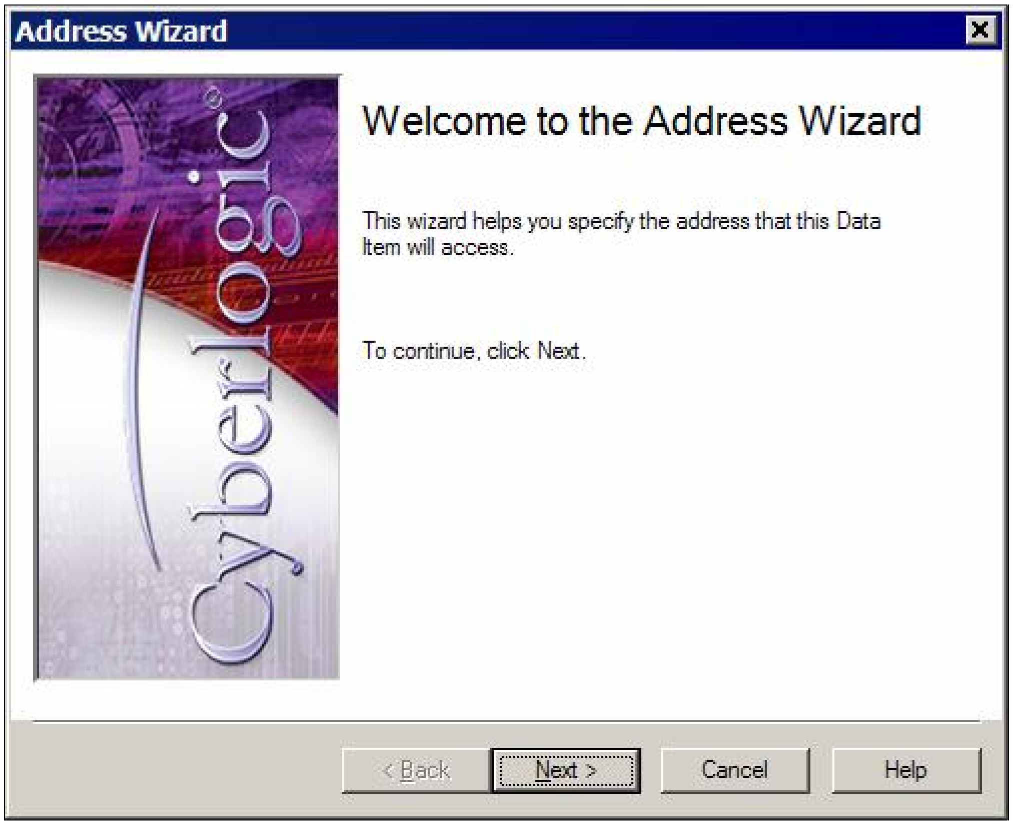
The Data Item will use the following address.

400022

Click Finish to exit.

Example 2: Arrays of Registers

This example shows how to set up the addressing to allow you to access several registers as an array. From the Welcome screen, click *Next* to continue.

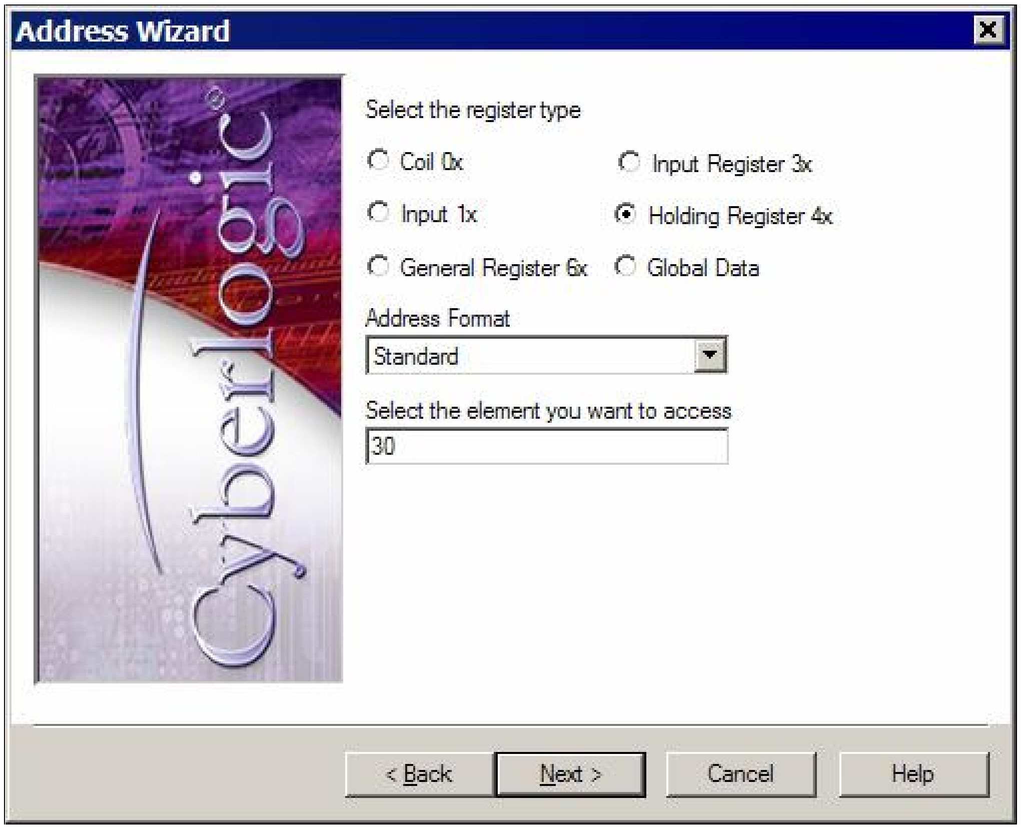


Welcome to the Address Wizard

This wizard helps you specify the address that this Data Item will access.

To continue, click Next.

On this screen, you must select the first element of the array, that is, the first register you want to access. For this example, we chose holding register number 30, which we want to address using the standard format. Make your selections and click *Next* to continue.



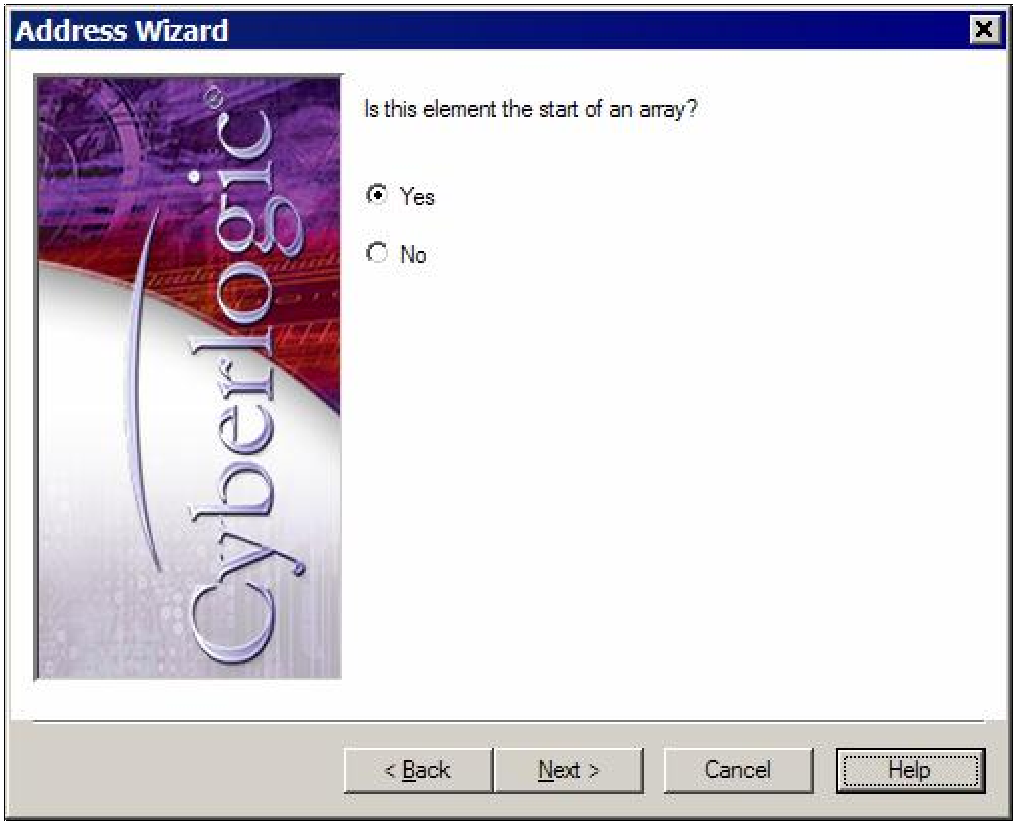
Select the register type

( ) Coil 0x
( ) Input Register 3x
( ) Input 1x
[✓ Holding Register 4x]
( ) General Register 6x
( ) Global Data

Address Format: Standard

Select the element you want to access: 30

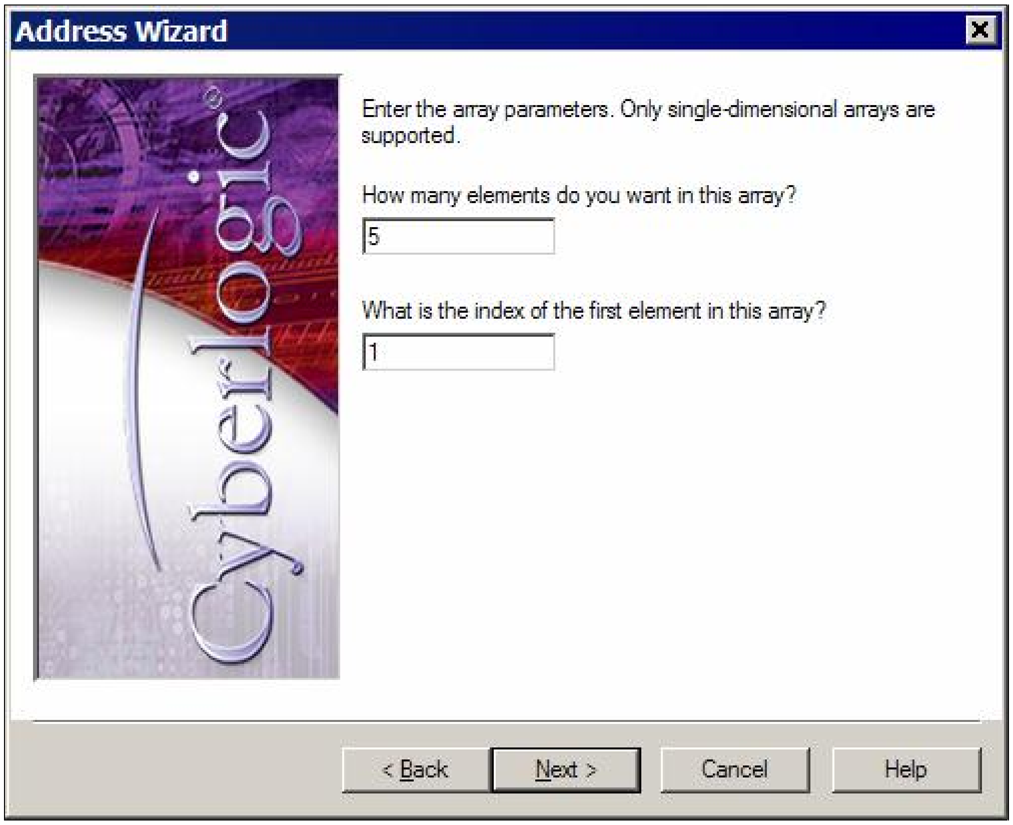
For this example, we want to access the data as an array, so select *Yes* and click *Next*.



Is this element the start of an array?

[✓ Yes] No

Here you must specify the size of the array and how the elements will be referenced. In this example, the array will have five elements, registers 400030 through 400034. The value entered as the index of the first element of the array specifies how the client will address the array. In this case, the array reference will start at 1, corresponding to 400030. The register 400031 would then be addressed using reference 2 and so on. After you enter these values, click *Next* to continue.



Enter the array parameters. Only single-dimensional arrays are supported.

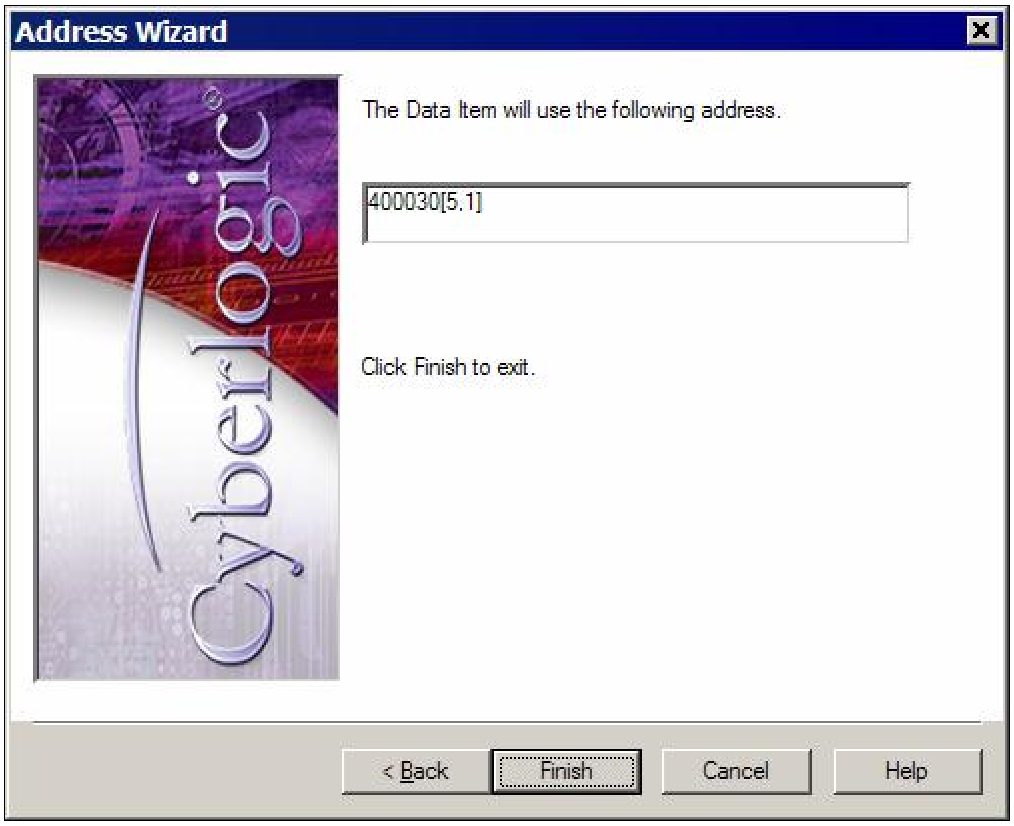
How many elements do you want in this array?

5

What is the index of the first element in this array?

1

This screen shows the result of your selections. The bracketed values show that the array contains 5 elements and that the index for addressing starts at 1. Click *Finish* to allow the wizard to complete and exit.



The Data Item will use the following address.

400030[5,1]

Click Finish to exit.

APPENDIX D: UNSOLICITED MESSAGE PROGRAMMING

The MBX driver agent accepts both read and write unsolicited messages. The read requests are used to read the content of the Server Status Block. The write requests allow trusted sources to update data in selected Data Items. The user identifies the trusted sources by configuring the Unsolicited Message Filters.

Programmable controllers can send unsolicited messages by activating MSTR block instructions in their logic program. For more information on how to send Modbus messages from various programmable controllers, refer to the appropriate Schneider documentation.

Custom applications written in C/C++ can use the MBX Software Development Kit to send communication requests over Modbus, Modbus Plus and Ethernet networks. Visit www.cyberlogic.com for more information.

Read Requests

The read requests are used to read the content of the Server Status Block. The MBX driver agent accepts the Read Holding Registers (Fnc 3) and Read/Write 4X Registers (Fnc 23) query messages with the read register addresses in the range of 400001-400006.

Write Requests

Write requests update data in Data Items configured for unsolicited update. A single message can update data in several Data Items. However, the message must contain data for the entire Data Item. Partial data updates are not allowed.

There are a number of query messages that can be used depending on the register address and the data type assigned to the Data Item. The following sections describe all supported MODBUS functions:

Force Single Coil (Fnc 5)

This function can be used to update data in Data Items that represent a single coil (address 0x).

Force Multiple Coils (Fnc 15)

This function can be used to update data in Data Items that represent either a single coil or multiple coils (address 0x). The maximum number of coils that can be sent by this query message is 1968.

Preset Single Register (Fnc 6)

This function can be used to update data in Data Items that require no more than a single holding register (address 4x). That includes bit-fields in the range of 1-16 bits.

Preset Multiple Registers (Fnc 16)

This function can be used to update data in Data Items that range in size from 1 bit to 120 16-bit holding registers (address 4x).

Write General Reference (Fnc 21)

This function can be used to update data in Data Items that range in size from 1 bit to 122 sixteen-bit general reference registers (address 6x).

Mask Write 4X registers (Fnc 22)

This function can be used to update data in Data Items that require no more than a single holding register (address 4x). Since individual bits within a register can be modified, this function is typically used for bit-fields that fit into a single register.

Read/Write 4X Registers (Fnc 23)

This function can be used to update data in Data Items that range in size from 1 bit to 118 sixteen-bit holding registers (address 4x).