Cyberlogic®

ControlLogix OPC Driver Agent Help

OPC Server Driver Agent for ControlLogix Controllers

Version 9

CONTROLLOGIX OPC DRIVER AGENT HELP

For ControlLogix Programmable Controllers

Version 9

Document last revision date May 1, 2019

TABLE OF CONTENTS

INTRODUCTION

The Cyberlogic OPC Server provides OPC Data Access, Alarms & Events and XML Data Access functions, and has a modular structure that supports a variety of industrial devices and communication networks. The various communication subsystems, which we call driver agents, are plug-ins that you can easily add as required. As a result, the server maintains a set of common features, but has the flexibility to allow additional features as required by the specific driver agent.

The ControlLogix Driver Agent is one of these plug-in modules. It allows the Cyberlogic OPC Server to communicate to the Logix family of Programmable Automation Controllers, such as ControlLogix, CompactLogix and FlexLogix, over Ethernet, Data Highway Plus and serial DF1 connections.

Note This document includes only the information that is specific to the ControlLogix Driver Agent. If you need help connecting to PLC-2, PLC-3, PLC-5 and SLC-500 controllers, refer to the DHX OPC Driver Agent Help. For information on the common features of the Cyberlogic OPC Server, refer to the Cyberlogic OPC Server Help.

Compatibility and Compliance

The ControlLogix Driver Agent is compatible with all DHX family products. The industry-standard DHX driver family provides the low-level communication services.

Cyberlogic OPC products provide full compliance with the OPC Foundation specifications for:

- Data Access 3.0, 2.05a and 1.0a
- Alarms & Events 1.1
- XML Data Access 1.0
- Data Access Automation 2.02

These products are tested for compliance to the OPC specifications using the latest test software from the OPC Foundation. All Cyberlogic OPC products are certified for compliance by the OPC Foundation's Independent Testing Laboratory. In addition, they are tested annually for interoperability with other OPC products at the OPC Foundation's Interoperability Workshops.

WHAT SHOULD I DO NEXT?

The links below will take you directly to the section of this manual that contains the information you need to configure, use and troubleshoot the ControlLogix Driver Agent.

This document describes only the features specific to the ControlLogix Driver Agent. For information on the common features of the Cyberlogic OPC Server, refer to the Cyberlogic OPC Server Help.

Learn How the OPC Server Works

If you are not familiar with the way that the ControlLogix Driver Agent obtains data, you should begin by reading the Theory of Operation.

Read a Quick-Start Guide

First-time users of the ControlLogix Driver Agent will want to refer to the Quick-Start Guide in the Cyberlogic OPC Server Help, which walks through a typical configuration session, step-by-step.

Get Detailed Information on the Configuration Editors

Experienced users who want specific information on features of the configuration editors will find it in the Configuration Editor Reference section.

Verify That It's Working or Troubleshoot a Problem

If you have already configured the server, you should verify that it operates as expected. Refer to the Validation & Troubleshooting section for assistance. In case of communication problems, this section also provides problem-solving hints.

Print a Copy of This Document

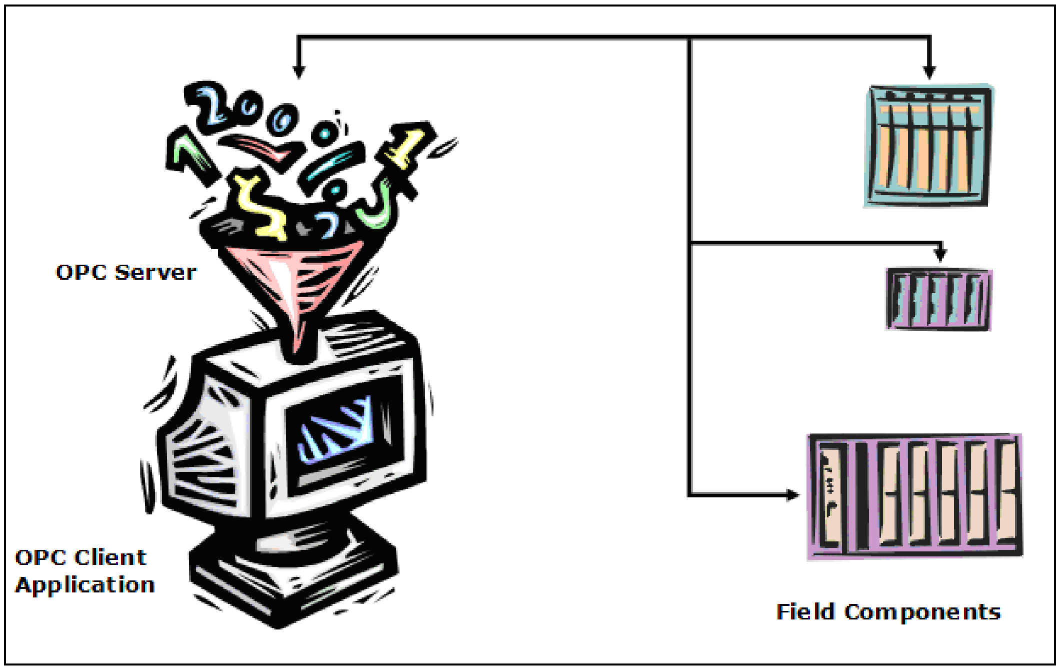
The content of this document is also provided in PDF format. PDF files can be viewed using the Adobe® Reader program, and can also be used to print the entire document.

Contact Technical Support

To obtain support information, open the Windows ***Start*** menu and go to ***Cyberlogic Suites***, and then select ***Product Information***.

THEORY OF OPERATION

This section will familiarize you with the main features of the Cyberlogic OPC Server as they relate to the ControlLogix Driver Agent. Refer to the Cyberlogic OPC Server Help for a full discussion of the common features of the Cyberlogic OPC Server. If you are new to OPC or the Cyberlogic OPC Server, we strongly recommend that you read the OPC Tutorial first. You will find it in the Help section of your product installation.

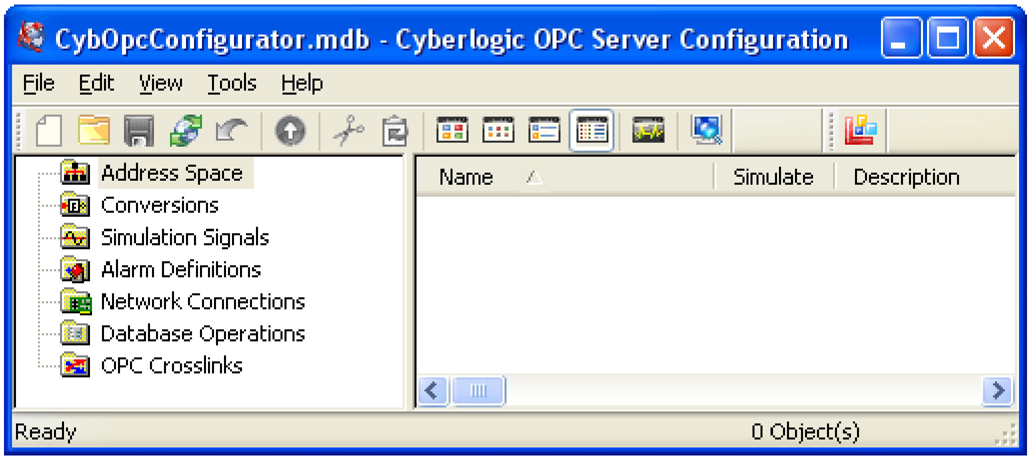


OPC servers obtain data from field components and present it, in a standard way, to OPC client applications. The basic functionality of the OPC server is the same for all types of field components and networks, but some of the communication and data-handling will vary from one family to another. The driver agent is a plug-in software module that accommodates these specific differences.

The ControlLogix Driver Agent handles communications over Data Highway Plus, Ethernet and serial DF1. It also provides the means to obtain and pass data from all Logix family Programmable Automation Controllers.

The remainder of this theory section will discuss the Main Server Features you will find in the Cyberlogic OPC Server, as they relate to Allen-Bradley communications.

Main Server Features



When you open the Cyberlogic OPC Server Configuration editor, you will find several main trees. These trees represent the main areas that you will configure. Note that some are for premium features that may not be part of the product you have installed, so they will not appear in your configuration. The trees are:

- The Address Space Tree is required for most configurations. Here you will create and organize the data items that will be available to the client application, and you will define how they are updated with new information.
- The Conversions Tree is optional. In it, you can define formulas that can be used to convert raw data values obtained from the field equipment into a form that is more useful to the client. For example, you can change a transducer's voltage value into a pressure value in psi. Refer to the Cyberlogic OPC Server Help for a full discussion of this tree.
- The Simulation Signals Tree is optional. If you want to be able to use simulated data item values instead of real values, you can create various types of simulated data functions in this tree. Simulations are often useful for troubleshooting client applications. Refer to the Cyberlogic OPC Server Help for a full discussion of this tree.
- The Alarm Definitions Tree is another optional tree. It is used when you will interface to Alarms & Events clients. This tree allows you to define the desired alarm conditions and specify what information should be passed as they occur and clear. Refer to the Cyberlogic OPC Server Help for a full discussion of this tree.
- The Network Connections Tree is required for all configurations. This is where you select the networks and interface devices you will use, and configure each of the field components as nodes on those networks.
- The Database Operations Tree is part of the logging feature, which is a premium feature. If this tree is in your product, you can use it to configure databases and data logging operations. Refer to the Data Logger Help for a full discussion of this tree.
- The OPC Crosslinks Tree is part of OPC Crosslink, which is a premium feature. If this tree is in your product, you can use it to configure data transfers between PLCs, between OPC servers and between PLCs and OPC servers. Refer to the OPC Crosslink Help for a full discussion of this tree.

The following sections describe these operational features of the server. Because the Network Connections Tree is normally configured first, we will start there.

Network Connections Tree

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

The ControlLogix Driver Agent uses various means for connecting to its devices or networks. For a DF1 connection, a serial COM port serves that purpose. In other cases, a Data Highway Plus or Ethernet adapter card is used. The Cyberlogic OPC Server refers to all of these using the generic term "network connection". The ControlLogix Driver Agent uses the Cyberlogic DHX family of drivers for low-level communication services to all of its network connections. Each network connection in the Cyberlogic OPC Server corresponds to a CLX device in the driver configuration, and contains all of the parameters for these devices.

The server refers to each physical device on the network as a "network node". A typical network node might be a Logix controller on an Ethernet network. The server accesses the network nodes through their corresponding network connection. The network node configuration contains the communication parameters for the physical node device.

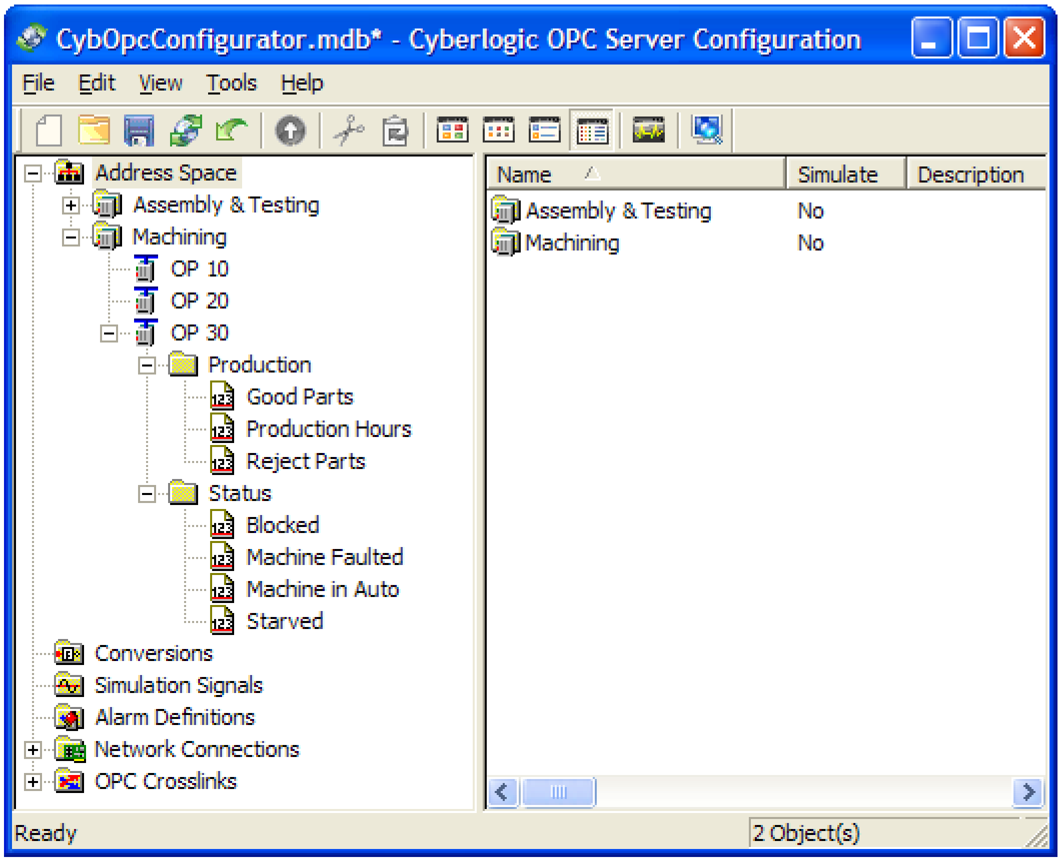
Network Automatic Configuration

The ControlLogix Driver Agent does not support the network automatic configuration feature. You must manually configure the ControlLogix network connections and network nodes.

Address Space Tree

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

The address space tree allows you to organize the data items in a way that makes sense for your project. You can group and name related data items regardless of where they exist in the physical devices.



Name	Simulate	Description
Assembly & Testing	No	
Machining	No	

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.

Access Paths

An access path is 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 during solicited data reads and writes. The ControlLogix Driver Agent requires that at least one access path is configured.

Each device in the server's address space can have a list of associated access paths. If there are multiple access paths, the one at the top of the list is the primary access path, and the rest are backups.

You can create and edit access paths on the device's Access Paths Tab.

Unsolicited Message Filters

In addition to the more common solicited updates, the Cyberlogic OPC Server supports unsolicited data updates for some driver agents. Unsolicited message filters are then used to guarantee that unsolicited messages are accepted only from trusted sources.

However, the ControlLogix Driver Agent does not support unsolicited messaging, so these filters are not available.

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. The ControlLogix Driver Agent supports all ControlLogix data types. For more information on the supported data types, refer to Appendix A: PLC Tag Names and Appendix B: Predefined ControlLogix Structures.

Conversions Tree

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

Simulation Signals Tree

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

Alarm Definitions Tree

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

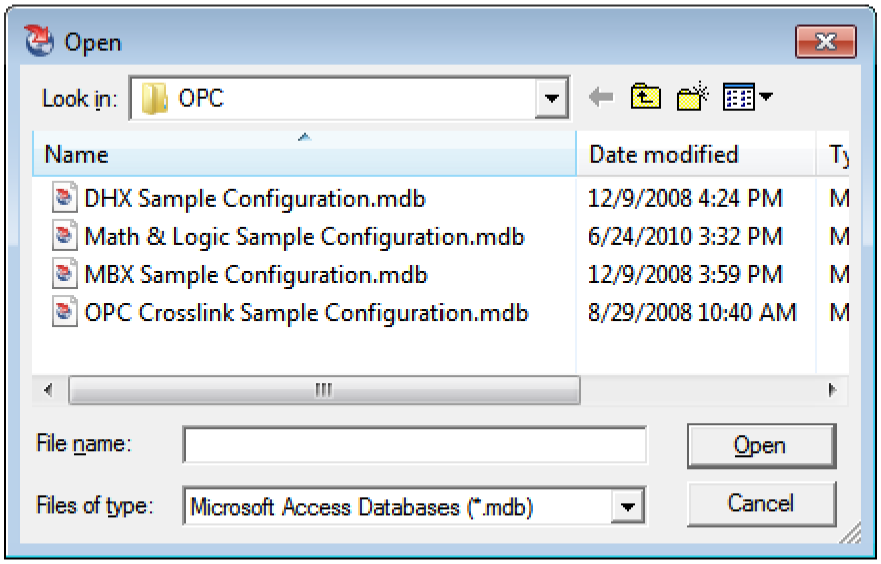
QUICK-START GUIDE

Before you can use the OPC server, you must configure it by using the OPC Server Configuration Editor. Every server requires configuration of the Network Connections tree, and most users will want to configure the Address Space tree. The remaining trees (Conversions, Simulation Signals and Alarm Definitions) are optional features used by some systems.

Sample Configuration Files

The default installation of all Cyberlogic OPC Server Suites includes a set of sample configuration files. These samples will help you to understand how to configure the OPC server for your project. In addition, the OPC Math & Logic sample provides you with numerous sample programs that you can modify and use in your system.

To open a sample configuration file from the OPC Server Configuration Editor, open the ***File*** menu and then select ***Open Sample...*** .



Name	Date modified	Ty
DHX Sample Configuration.mdb	12/9/2008 4:24 PM	M
Math & Logic Sample Configuration.mdb	6/24/2010 3:32 PM	M
MBX Sample Configuration.mdb	12/9/2008 3:59 PM	M
OPC Crosslink Sample Configuration.mdb	8/29/2008 10:40 AM	M

A browse window will open to allow you to select the configuration file you want. The available choices will depend on which OPC products you have installed.

The default location of the files is:

C:\Program Files\Common Files\Cyberlogic Shared\OPC.

Step-By-Step Example

For a step-by-step guide through a typical configuration session, refer to the Cyberlogic OPC Server Help. After you have created the basic configuration using that procedure, the Configuration Editor Reference will explain the details of how to edit your configuration.

CONFIGURATION EDITOR REFERENCE

Before you can use the OPC server, you must configure it by using the OPC Server Configuration Editor. Every server requires configuration of the Network Connections tree, and most users will want to configure the Address Space tree. The remaining trees (Conversions, Simulation Signals and Alarm Definitions) are optional features used by some systems.

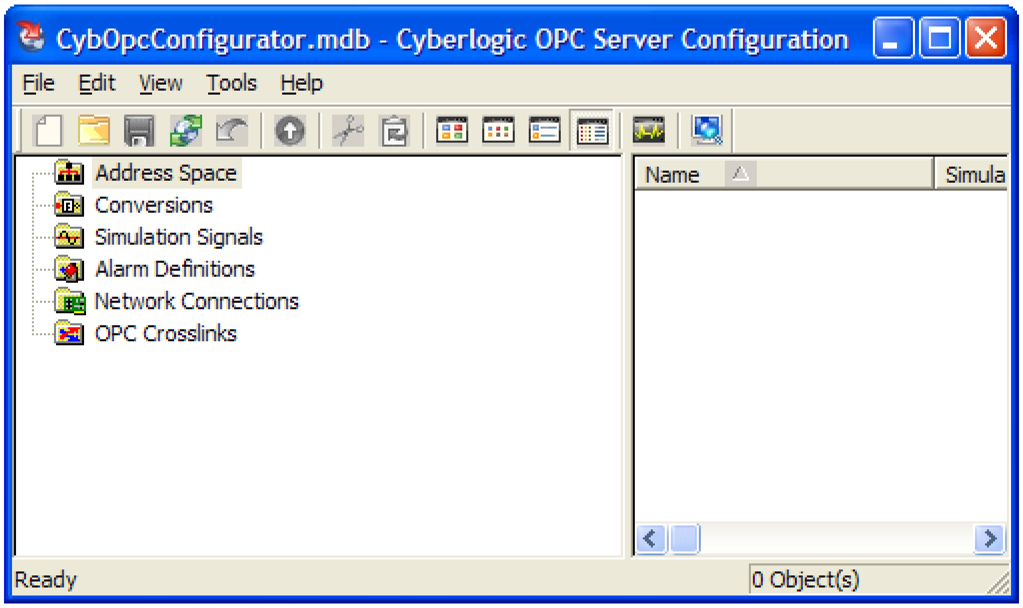
This section provides a detailed description of each of the configuration editor features. If you are a new user and want a procedure to guide you through a typical configuration session, refer to the Quick-Start Guide in the Cyberlogic OPC Server Help.

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.

Caution!

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

To launch the editor from the Windows ***Start*** menu, go to ***Cyberlogic Suites***, then open the ***Configuration*** sub-menu, and then select ***OPC Server***.



The left pane of the main workspace window includes the five main configuration trees. Two of these, the Network Connections and Address Space trees, are described in detail here, followed by a section on Saving and Undoing Configuration Changes.

For a discussion of the Conversions, Simulation Signals and Alarm Definitions trees and other important configuration topics including Configuration Import/Export, Editor

Options and Connecting to OPC Client Software, please refer to the Cyberlogic OPC Server Help.

Network Connections

The ControlLogix Driver Agent is a plug-in software module that permits the Cyberlogic OPC Server to communicate with the Logix family of Programmable Automation Controllers. You will set up your OPC Server to do this by configuring appropriate network connections and network nodes in the Network Connections Tree.

The ControlLogix Driver Agent uses the DHX drivers for its low-level communication services. A network connection in the OPC server corresponds to a CLX device in the driver architecture. A CLX device may relate to a physical network card, such as a 1784-PKTX, or an abstract object, such as an Ethernet CLX device, which behaves like a network card. A network node in the OPC server corresponds to a Logix controller that is connected to the OPC server over one of the networks.

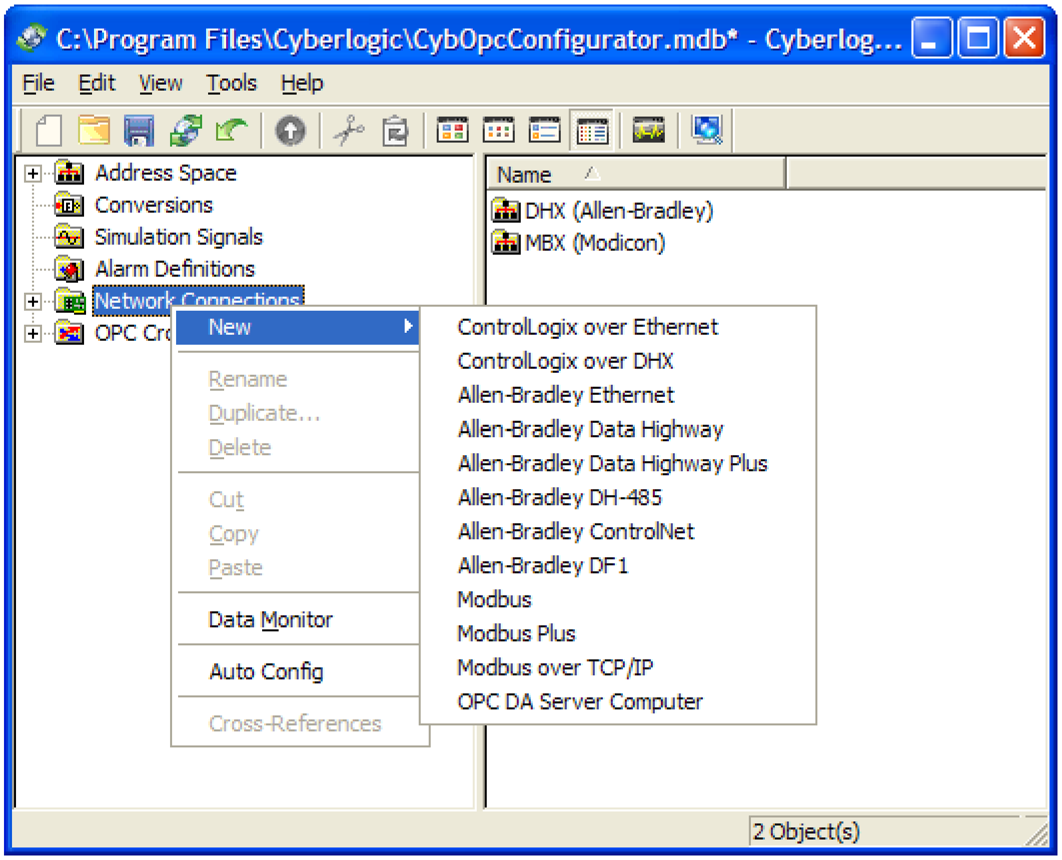
Creating and Deleting

In the Cyberlogic OPC Server, there are two ways to create network connections and network nodes: automatic and manual. However, the ControlLogix Driver Agent requires manual configuration.

Caution!

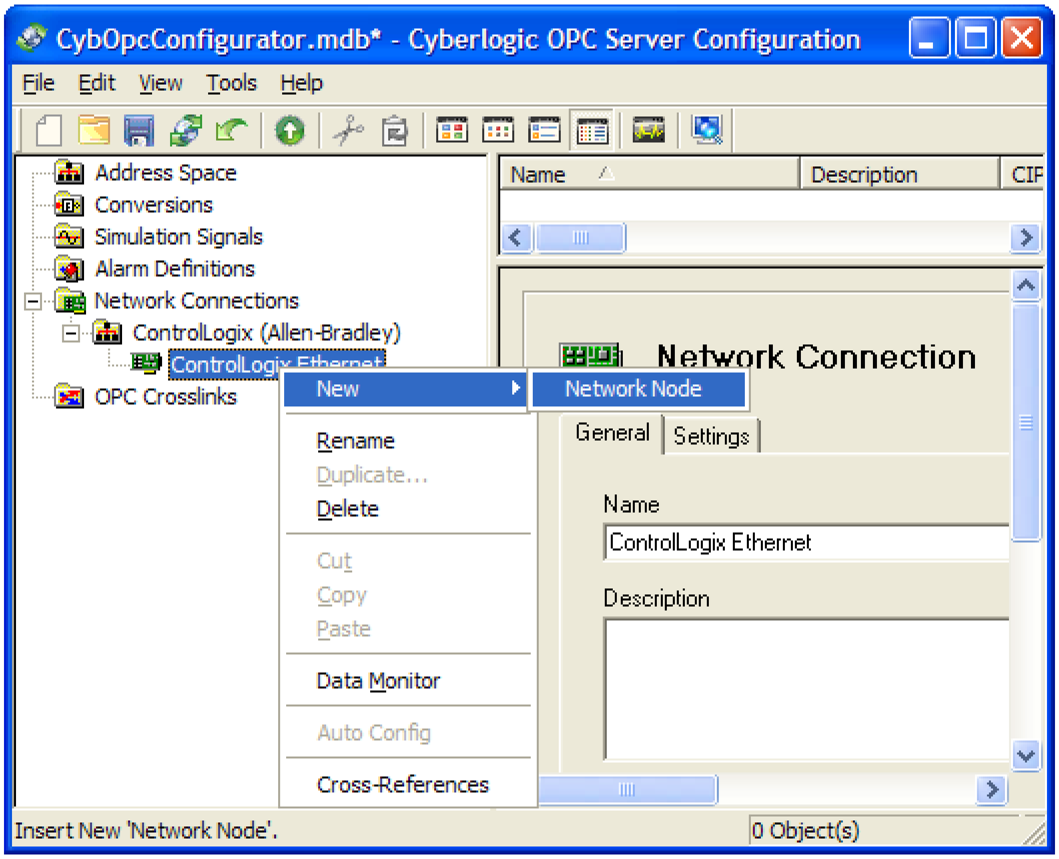
After you edit the configuration, you must open the ***File*** menu and select ***Save & Update Server***, or click the ***Save & Update Server*** toolbar button, for the changes you have made to take effect. Otherwise, the server will still be running with the old configuration.

Creating Network Connections



To create a network connection, right-click on the ***Network Connections*** root folder and select ***New***, then select the desired network type from the context menu.

The editor will create the proper driver agent and network connection folders.



You can now right-click on the network connection and select ***New***, and then ***Network Node*** to manually create a network node.

Deleting

To delete the ControlLogix Driver Agent folder, a network connection or a network node, select it and press the ***Delete key***, or right-click on it and select ***Delete*** from the context menu.

Deleting the ControlLogix Driver Agent folder will also delete all ControlLogix network connections and network nodes.

Deleting a ControlLogix network connection will also delete all of its network nodes.

Note You cannot delete a network node that is used as part of an access path for a device. If you wish to delete a network node that is in use, right-click on it and select ***Cross-References*** to obtain a list of devices that use the network node. You must then edit those devices to remove the access path that contains the network node you want to delete.

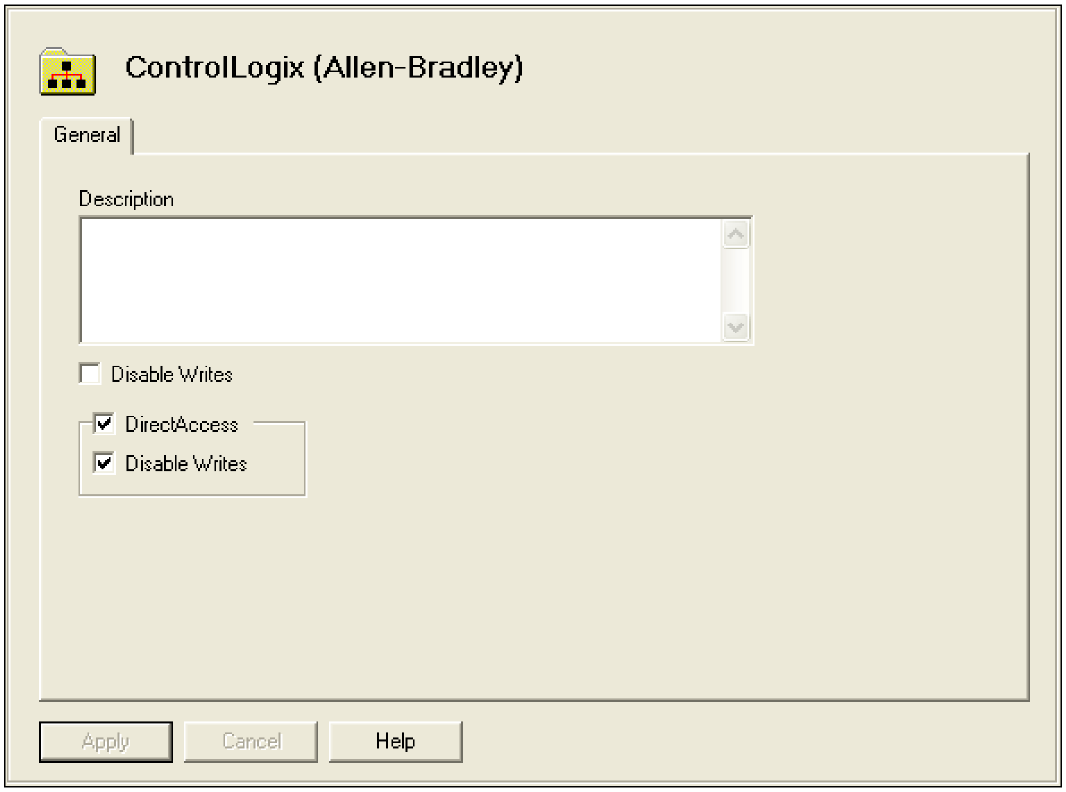
Editing the ControlLogix Driver Agent

Once you have created a ControlLogix network connection, simply click on the ***ControlLogix (Allen-Bradley)*** folder, and the configuration screen will appear on the right side of the editor.

The ControlLogix Driver Agent configuration has one tab, called General.

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

General Tab


ControlLogix (Allen-Bradley)

General

Description

☐ Disable Writes

☑ DirectAccess
☑ 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 network 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 network 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 network nodes that connect through this driver agent. This does not affect writes through configured data items.

The default state is checked, disabling DirectAccess writes.

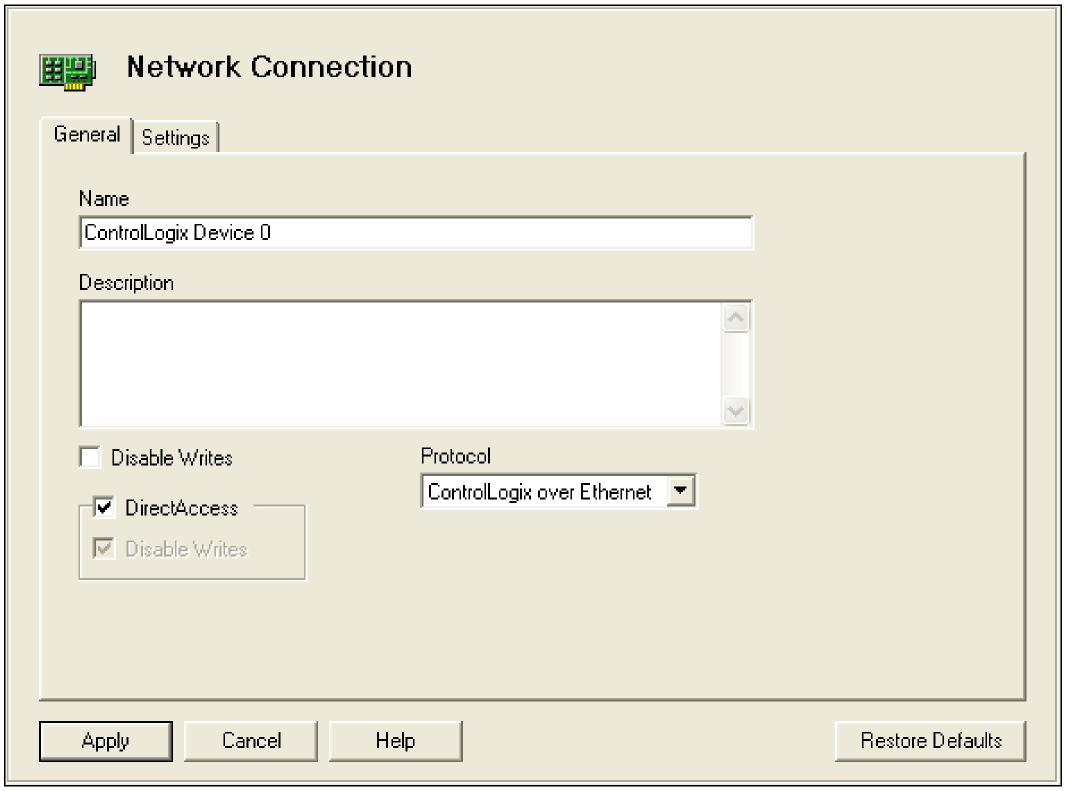
Editing Network Connections

Once you have created a ControlLogix network connection, simply select it and the configuration screen will appear on the right side of the editor.

The ControlLogix network connection configuration has two tabs, General and Settings.

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

General Tab



Name
ControlLogix Device 0

Description

☐ Disable Writes

☑ DirectAccess

☑ Disable Writes

Protocol
ControlLogix over Ethernet

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.

When DirectAccess is enabled, a _Status folder will appear under this network connection in the client browser window. For more information on this folder and the status items it contains, refer to the Cyberlogic OPC Server Help.

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.

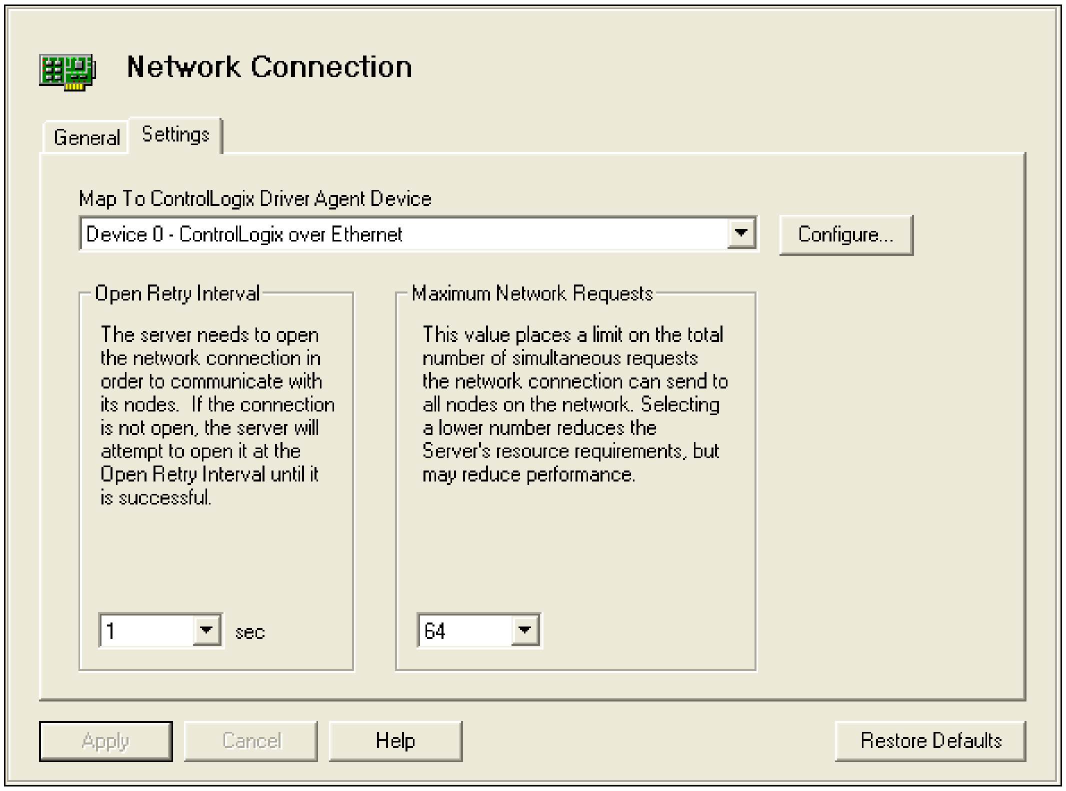
Note If the DirectAccess Disable Writes checkbox is grayed-out, it indicates that DirectAccess writes have already been disabled at a higher level.

Protocol

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

The choices are ControlLogix over Ethernet and ControlLogix over DHX.

Settings Tab



Network Connection

Map To ControlLogix Driver Agent Device: Device 0 - ControlLogix over Ethernet

Open Retry Interval

The server needs to open the network connection in order to communicate with its nodes. If the connection is not open, the server will attempt to open it at the Open Retry Interval until it is successful.

1 sec

Maximum Network 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.

64

Map To ControlLogix Driver Agent Device

Click the drop-down button to select the CLX device this network connection will use to communicate.

The data field will tell you if the device is not configured or if it is configured for a different protocol than you have selected for the network connection.

Configure...

Click this button to run the DHX Driver Configuration Editor, allowing you to modify the configuration of the CLX devices and to create new ones. For more information on configuring CLX devices, refer to Editing the ControlLogix Driver Agent Device.

Open Retry Interval

The server will try to open the CLX device shown in the Map To ControlLogix Driver Agent 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.

The valid range for the Open Retry Interval is 1-60 seconds, and the default is 1 second.

Maximum Network Requests

This value defines the maximum number of simultaneous transactions that the server will allow for the nodes on this network connection. The number of concurrent requests may also be limited for each network node on its Optimizations Tab.

The optimum value to use depends upon the network type. If the messages will pass through a bridge or other device to a different type of network, then the slowest network in the path to the network node should dictate the value to use.

Caution!

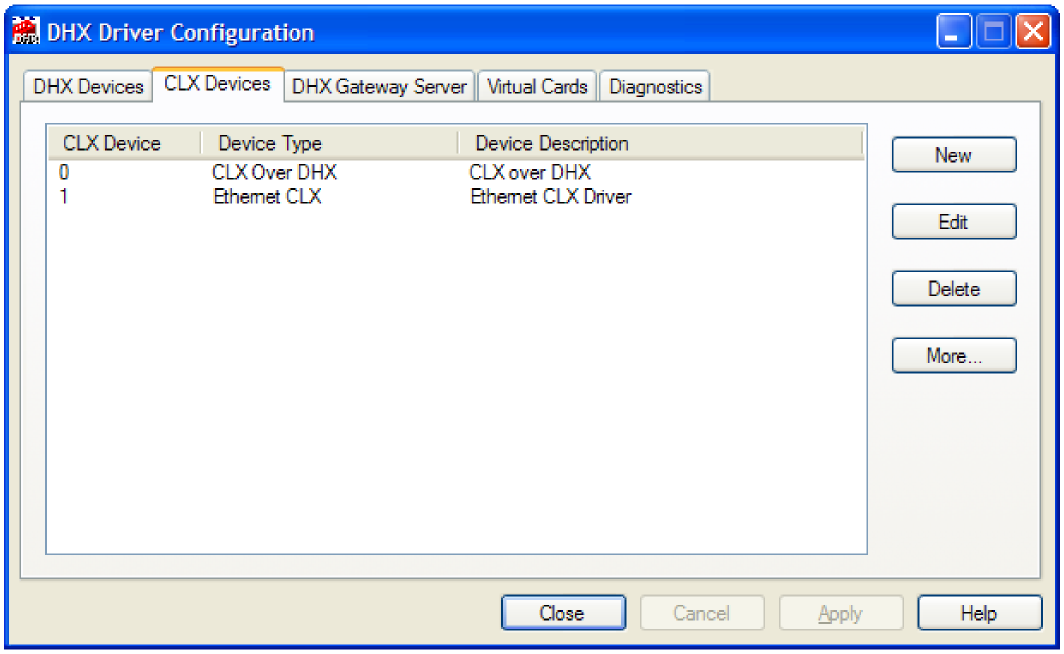
Lower values reduce the server's resource requirements, but may reduce performance.

Selecting excessively high values consumes more system resources, but typically does not improve the performance of the server, and may actually harm performance.

Refer to Appendix E: Configuring Maximum Concurrent Requests for more information and examples.

Editing the ControlLogix Driver Agent Device

When you click the Configure… button on the Settings tab, the DHX Driver Configuration editor opens.

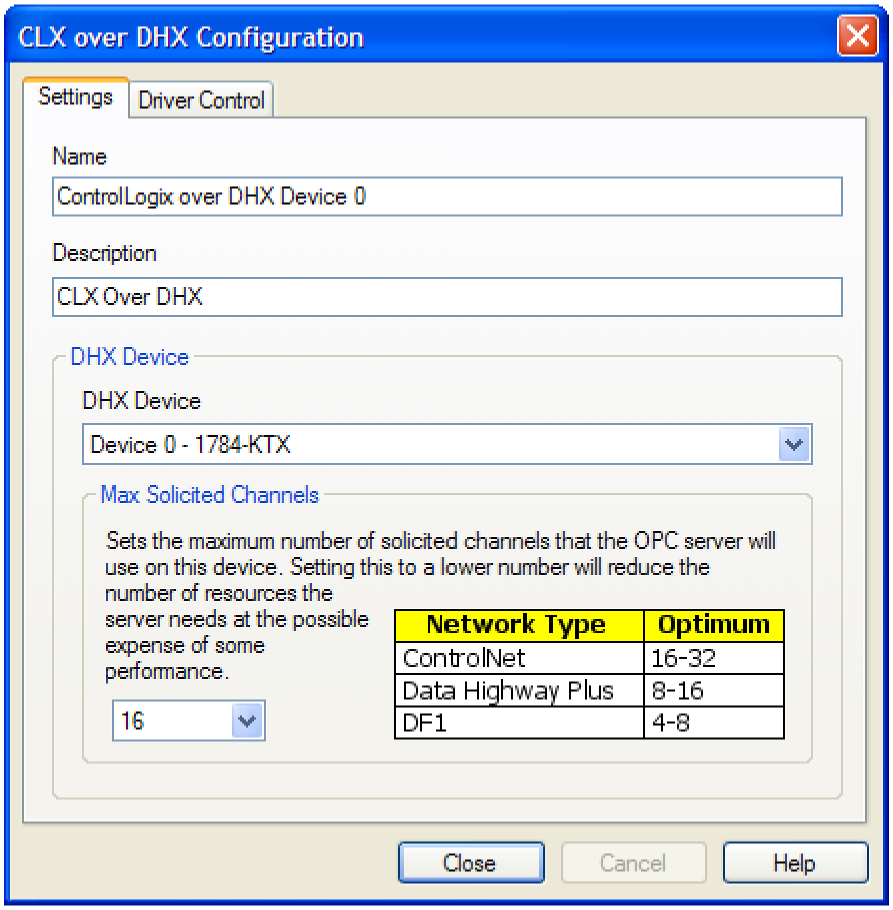


CLX Device	Device Type	Device Description
0	CLX Over DHX	CLX over DHX
1	Ethernet CLX	Ethernet CLX Driver

The CLX Devices tab lists all of the ControlLogix Driver Agent devices and allows you to add new devices. Select the desired CLX device and click ***Edit***. The appropriate editor dialog appears.

Settings Tab

The two editors are similar, so we will use the CLX over DHX Configuration Editor as an example, noting where the Ethernet CLX Editor differs.



Name: ControlLogix over DHX Device 0

Description: CLX Over DHX

DHX Device

DHX Device: Device 0 - 1784-KTX

Max Solicited Channels

Sets the maximum number of solicited channels that the OPC server will use on this device. Setting this to a lower number will reduce the number of resources the server needs at the possible expense of some performance.

16

Network Type	Optimum
ControlNet	16-32
Data Highway Plus	8-16
DF1	4-8

Name

Edit the name of the device.

Description

Here, you can edit the device's description.

DHX Device

This section does not appear for Ethernet CLX devices.

For CLX Over DHX devices, you must select the DHX device that the CLX device will use.

Max Solicited Channels

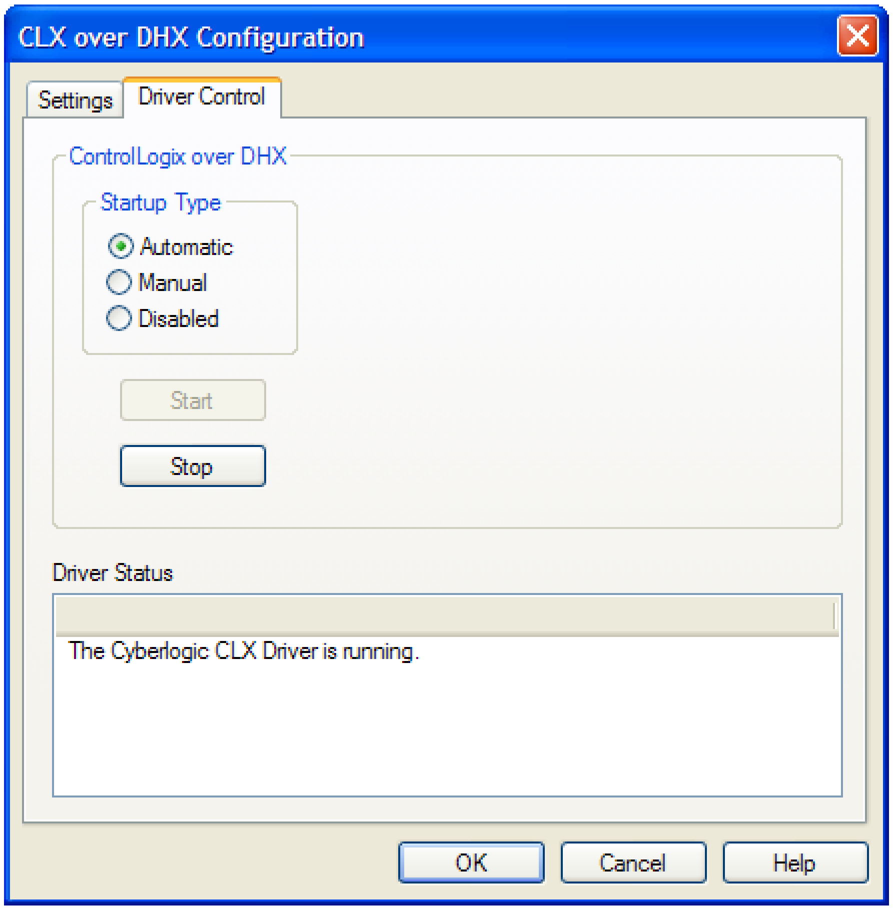
This field does not appear for Ethernet CLX devices.

This selection sets the maximum number of solicited channels the OPC Server will use for the device. 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
ControlNet	16-32
Data Highway Plus	8-16
DF1	4-8

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

Driver Control Tab



Startup Type: [✓ Automatic] Manual Disabled

The Cyberlogic CLX Driver is running.

Automatic

When this option is selected, the CLX Driver will start when Windows boots.

Manual

When this option is selected, the CLX Driver will not start when Windows boots, but you can control it manually using the Start and Stop buttons.

Disabled

When this option is selected, the CLX Driver will not run.

Start

In Automatic or Manual mode, click this button to start the CLX Driver.

Stop

In Automatic or Manual mode, click this button to stop the CLX Driver.

Driver Status

This tells you whether or not the CLX Driver is running, stopped, starting or stopping.

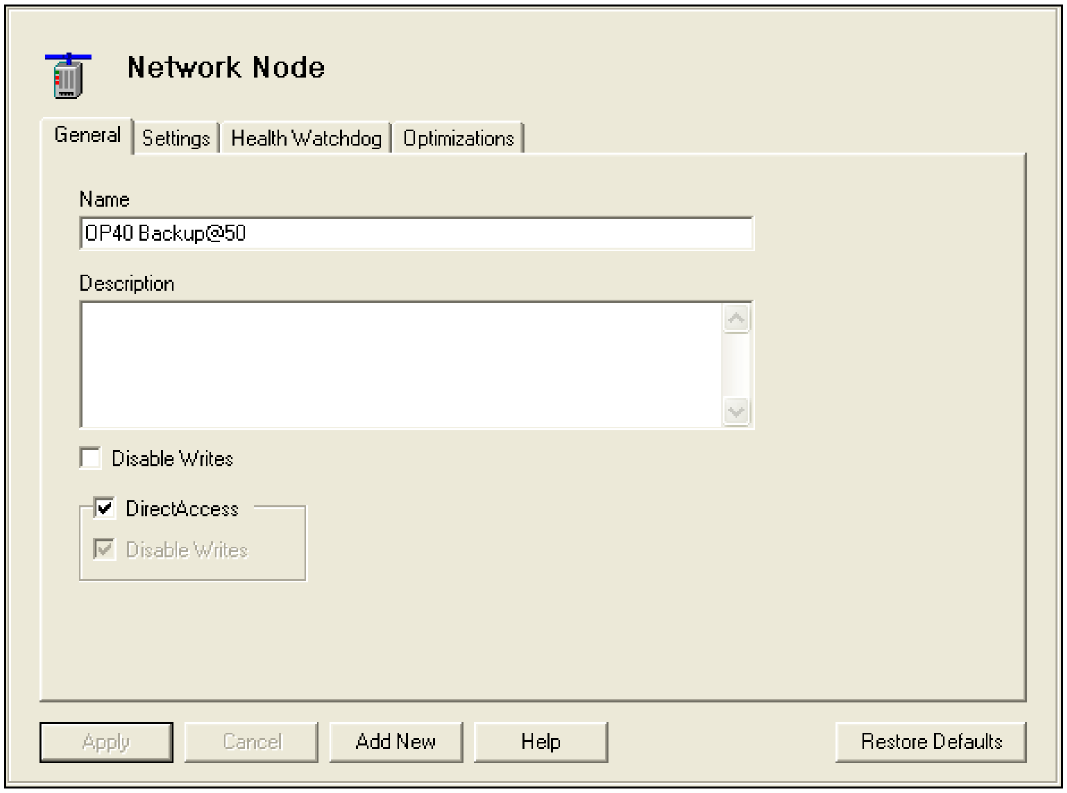
Editing Network Nodes

Once you have created a ControlLogix network node, simply select it and the configuration screen will appear on the right side of the editor.

The ControlLogix network node configuration has four tabs, General, Settings, Health Watchdog and Optimizations.

Caution! After you edit the configuration, you must open the ***File*** menu and select ***Save & Update Server***, or click the ***Save & Update Server*** toolbar button, 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.

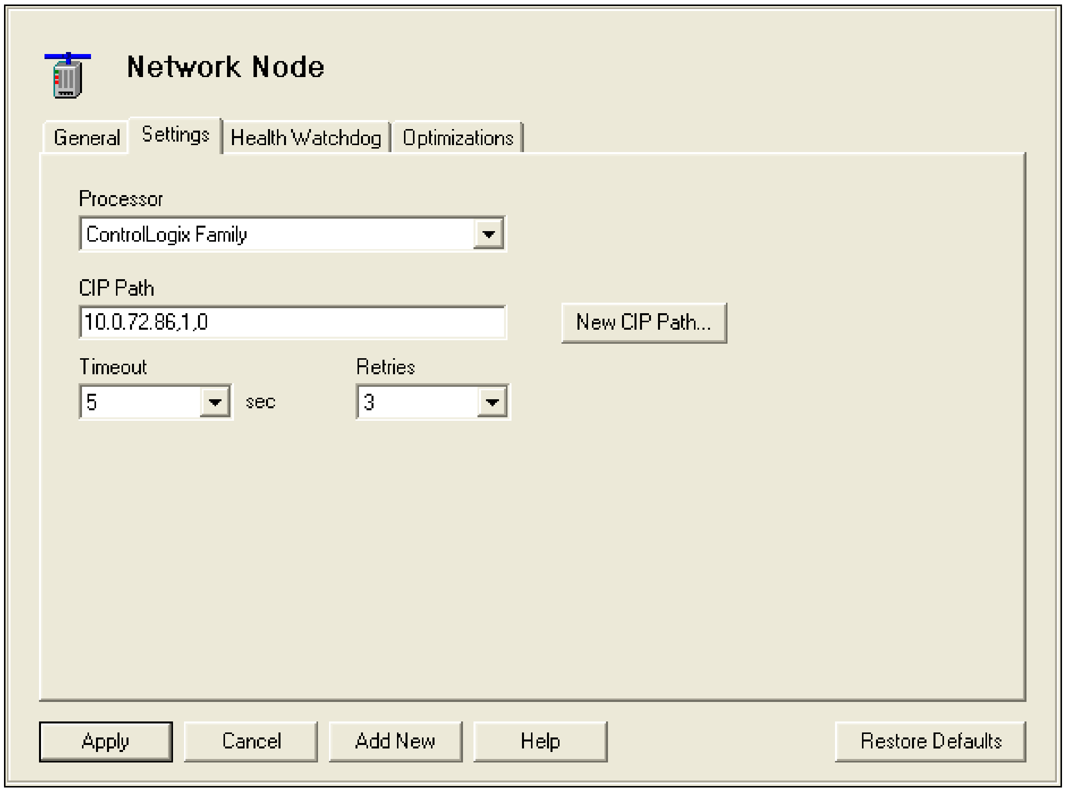
When DirectAccess is enabled, a _Status folder will appear under this network node in the client browser window. For more information on this folder and the status items it contains, refer to the Cyberlogic OPC Server Help.

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



Network Node

Processor: ControlLogix Family

CIP Path: 10.0.72.86,1,0

Timeout: 5 sec

Retries: 3

Processor

This field identifies the physical device type of this network node. You may select a specific PLC model or a PLC family. Choose a family if you are unsure of the exact model in use. The server uses this information to optimize network communications.

CIP Path

This is the node's network address. The interpretation of this address varies according to the type of network. For more information about CIP paths, refer to Appendix C: CIP Paths. The CIP Path Wizard will help you to create the correct CIP path for your installation.

New CIP Path

Click this path to launch the CIP Path Wizard. It will guide you through the configuration of a new CIP path.

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.

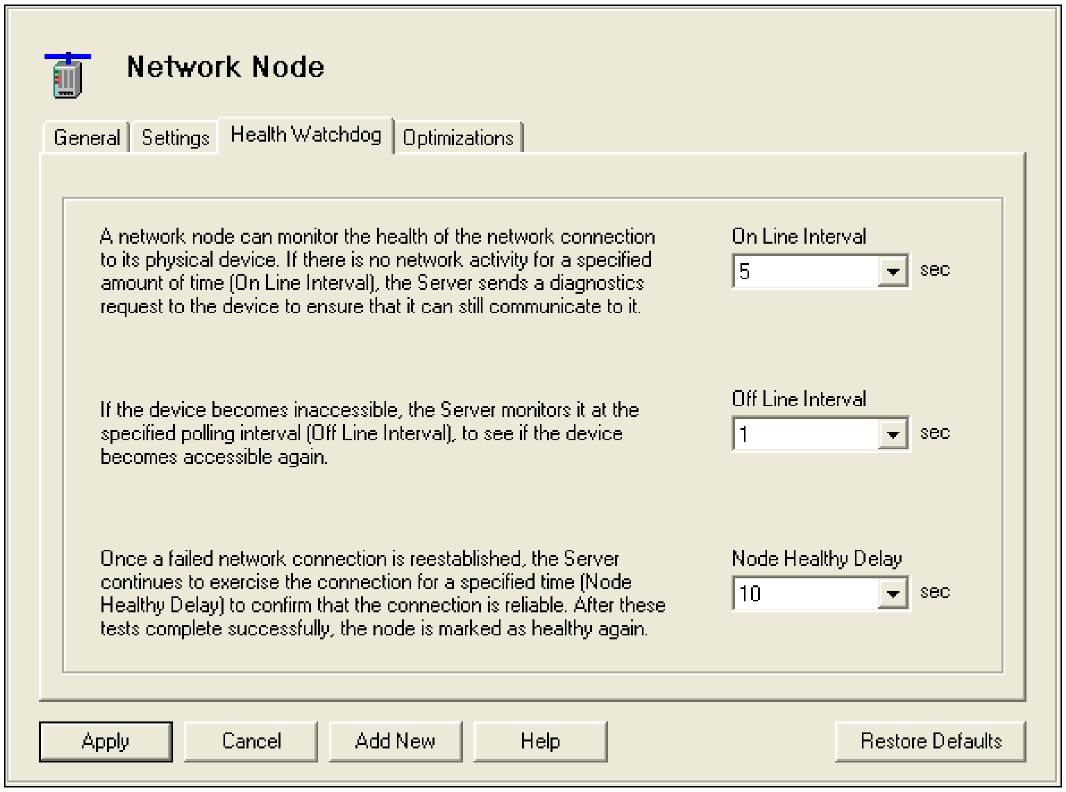
Retries

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. Refer to the Health Watchdog Tab section for more information on this feature.

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 physical 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.

The Health Watchdog tab allows you to configure the time intervals associated with these tests.



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 sec

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 sec

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 sec

On Line Interval

You may select a value in the range of 1-60 seconds or None.

If there is no traffic to a healthy node for the specified length of time, the server will send a status request to the node to verify that it is still online. Selecting None disables the on-line health monitoring.

If you are using a slow network and you are not using redundant access paths, you might want to disable health monitoring to reduce some of the network traffic. In that case, the OPC server will always report the status of the node as healthy.

Off Line Interval

You may select a value in the range of 1-60 seconds.

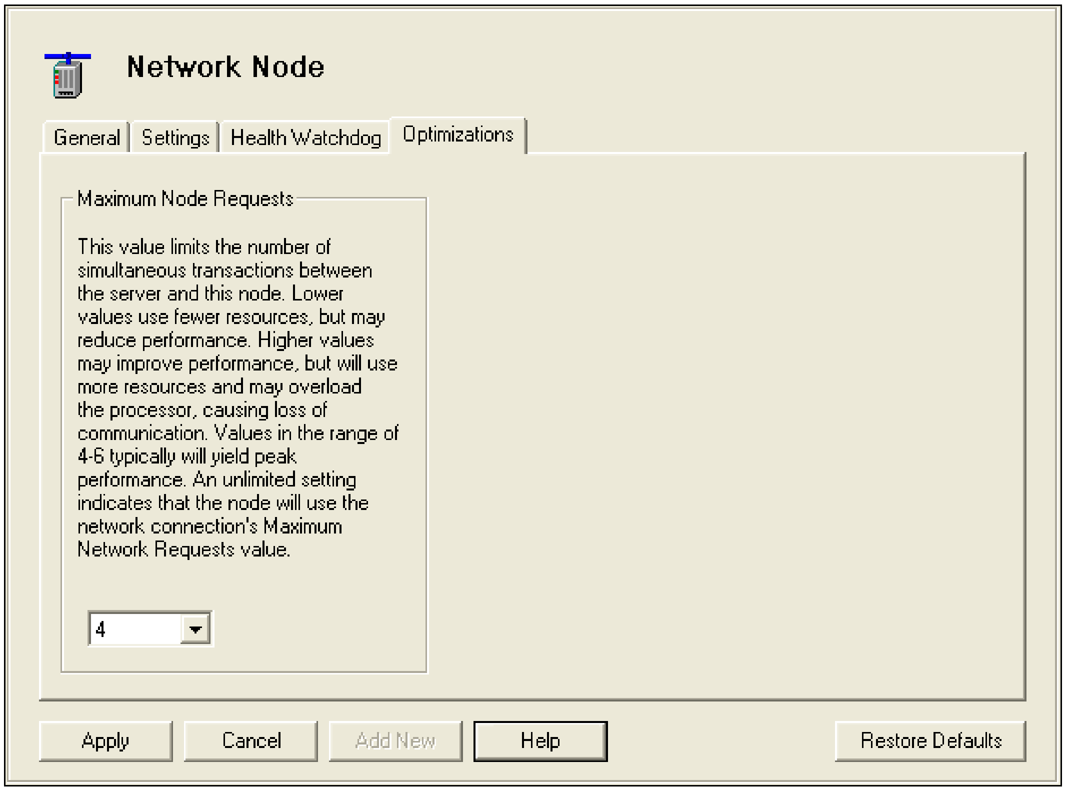
Once communication to a node has failed, the server will send status requests to the node at this interval to determine whether it can communicate.

Node Healthy Delay

You may select a value in the range of 1-60 seconds.

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 to be healthy. The server then returns the node to service.

Optimizations Tab



Maximum Node Requests

This value limits the number of simultaneous transactions between the server and this node. Lower values use fewer resources, but may reduce performance. Higher values may improve performance, but will use more resources and may overload the processor, causing loss of communication. Values in the range of 4-6 typically will yield peak performance. An unlimited setting indicates that the node will use the network connection's Maximum Network Requests value.

4

Maximum Node Requests

This value defines the maximum number of simultaneous transactions that the server will allow for this node. The number of concurrent requests is also limited for the network connection on its Settings Tab.

Lower values reduce the resource requirements but may reduce performance. Higher values use more resources, but typically improve performance. However, an excessively high setting may reduce performance or overload the processor, possibly causing loss of communication. Setting this selection to Unlimited causes the node to use the network connection's Maximum Network Requests value.

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

Refer to Appendix E: Configuring Maximum Concurrent Requests for more information and examples.

Address Space

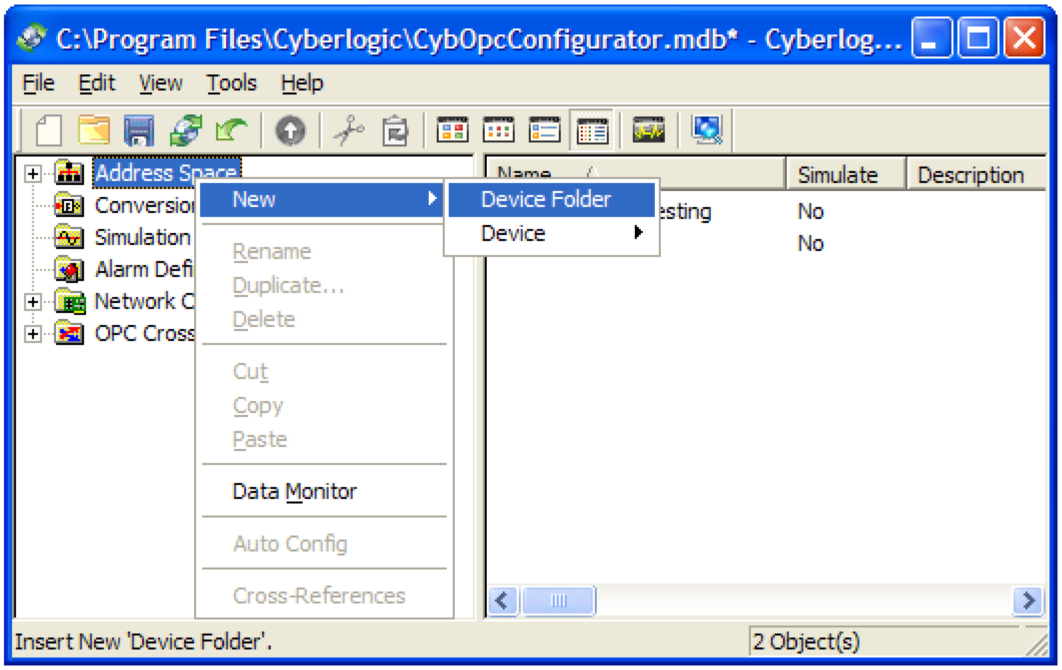
The Address Space Tree describes the hierarchical address structure of the Cyberlogic OPC Server. The branches of the tree are Device Folders, Devices and Folders. Its "leaves" are Data Items. The intent of this structure is to permit the user to organize the data items into logical groups.

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.

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

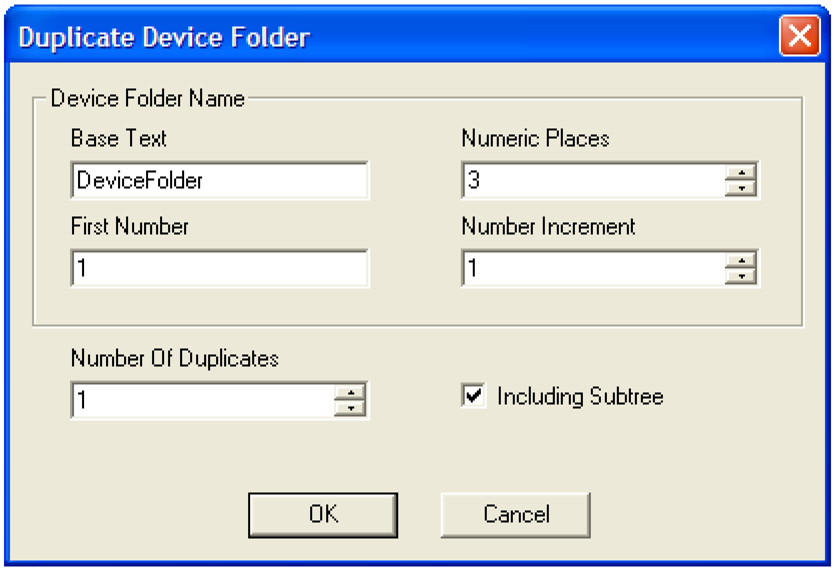
Creating a New Device Folder



Right-click on the Address Space root folder or an existing device folder. Select ***New*** and then ***Device Folder*** from the context menu.

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 one. To do this, right-click on an existing device folder and select ***Duplicate...*** from the context menu.



Device Folder Name

Base Text: DeviceFolder

Numeric Places: 3

First Number: 1

Number Increment: 1

Number Of Duplicates: 1

[✓] Including Subtree

The above dialog box opens. You must specify how the duplicates are to be named by entering values for the ***Base Text***, ***First Number***, ***Numeric Places*** and ***Number Increment*** fields. To generate names for the duplicated device folders, the editor begins with the base text and appends a number to it. The first duplicate uses the selected First Number value with the specified number of digits. This number is then incremented by the specified number for each of the remaining duplicates.

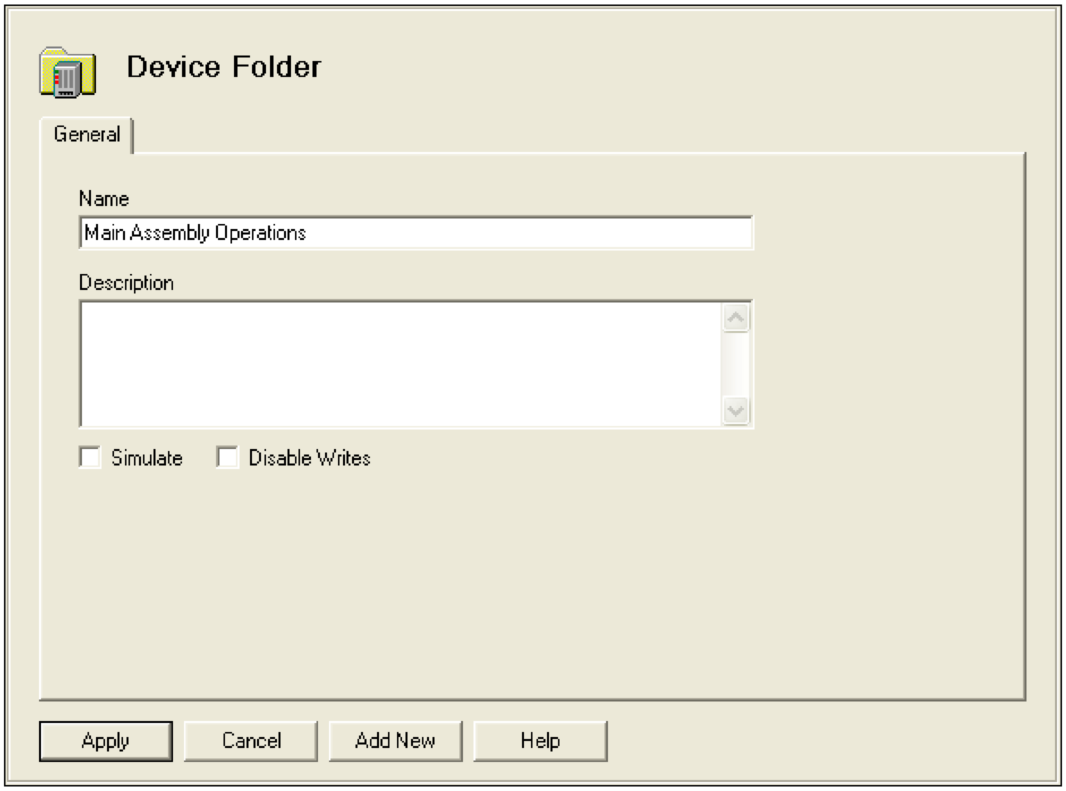
As an example, if Numeric Places is 3 and First Number is 2, the number 002 will be appended to the base text.

Use the ***Number Of Duplicates*** field to specify the number of device folders you wish to create. If you want to duplicate all branches within the original device folder, check the ***Including Subtree*** checkbox.

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.

General Tab



Name: Main Assembly Operations

Description:

[ ] Simulate [ ] Disable Writes

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 the Cyberlogic OPC Server Help for a full discussion about 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 a higher level.

Devices

A device in the address space 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 a device folder. In addition to its device-specific functionality, a device operates as a folder. It can contain folders and data items.

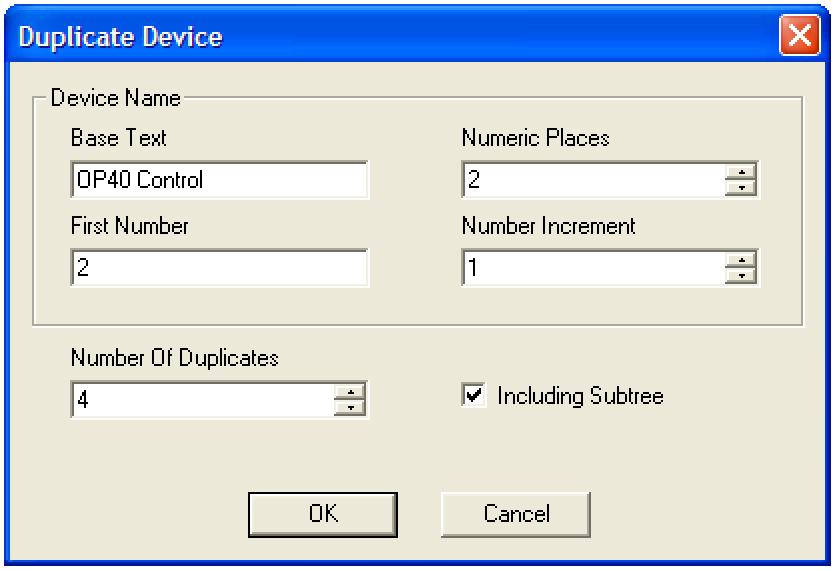
Caution! After you edit the configuration, you must open the ***File*** menu and select ***Save & Update Server***, or click the ***Save & Update Server*** toolbar button, for the changes you have made to take effect. Otherwise, the server will still be running with the old configuration.

Creating a New Device

Right-click on the Address Space root folder or an existing device folder. Select ***New*** and then ***Device*** from the context menu.

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 one. To do this, right-click on an existing device and select ***Duplicate…*** from the context menu.


Duplicate Device

Device Name

Base Text: OP40 Control

Numeric Places: 2

First Number: 2

Number Increment: 1

Number Of Duplicates: 4

☑ Including Subtree

The above dialog box opens. You must specify how the duplicates are to be named by entering values for the ***Base Text***, ***First Number***, ***Numeric Places*** and ***Number Increment*** fields. To generate names for the duplicated devices, the editor begins with the base text and appends a number to it. The first duplicate uses the selected First Number value with the specified number of digits. This number is then incremented by the specified number for each of the remaining duplicates.

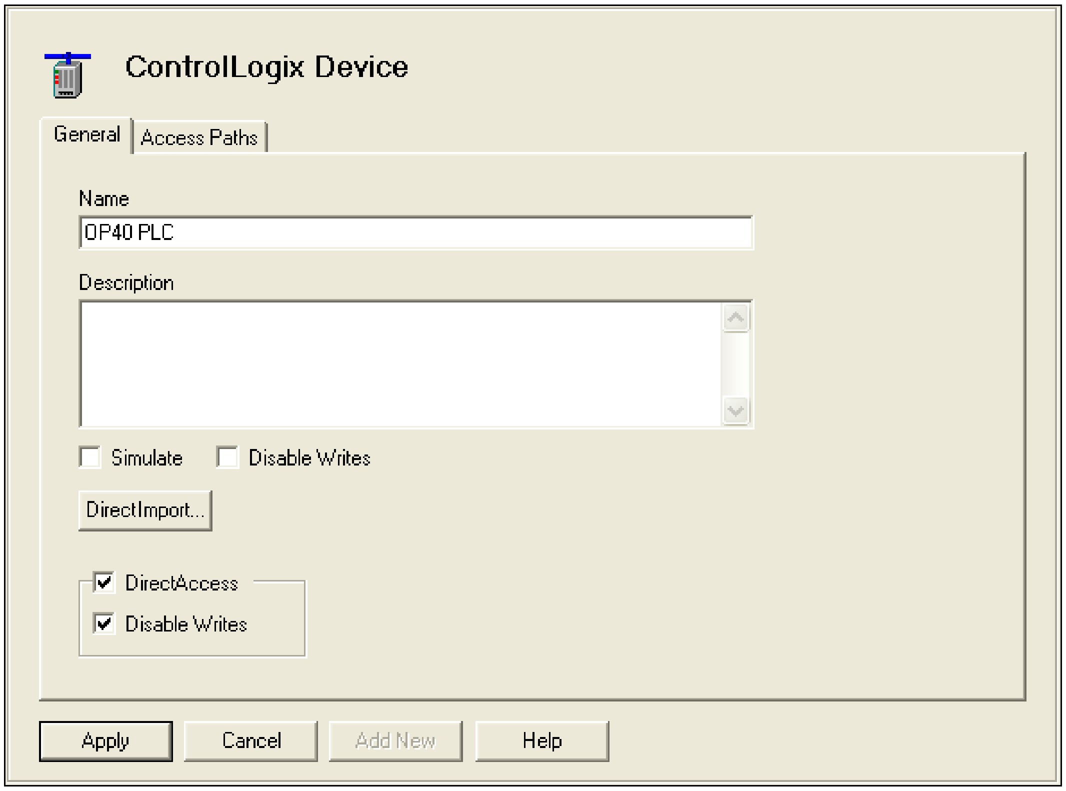
As an example, if Numeric Places is 3 and First Number is 2, the number 002 will be appended to the base text.

Use the ***Number Of Duplicates*** field to specify the number of devices you wish to create. If you want to duplicate all branches within the original device, check the ***Including Subtree*** checkbox.

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.

General Tab


ControlLogix Device

General | Access Paths

Name: OP40 PLC

Description:

☐ Simulate ☐ Disable Writes

DirectImport...

☑ DirectAccess
☑ Disable Writes

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 the Cyberlogic OPC Server Help for a full discussion about simulating data.

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

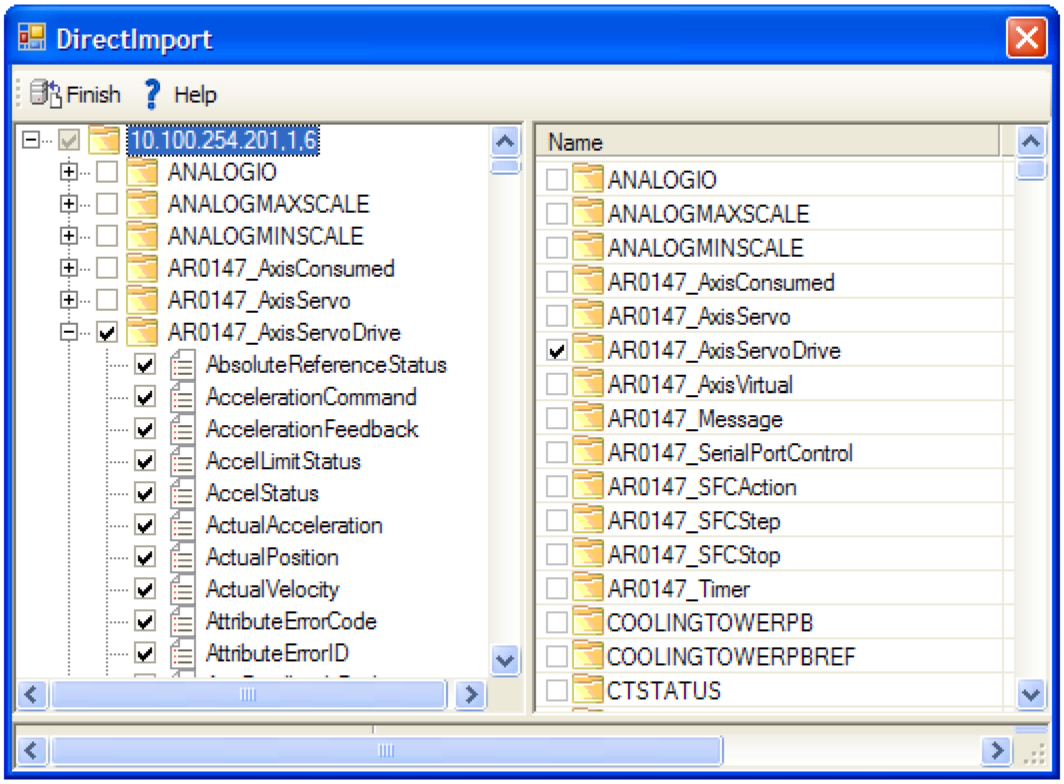
Disable Writes

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 a higher level.

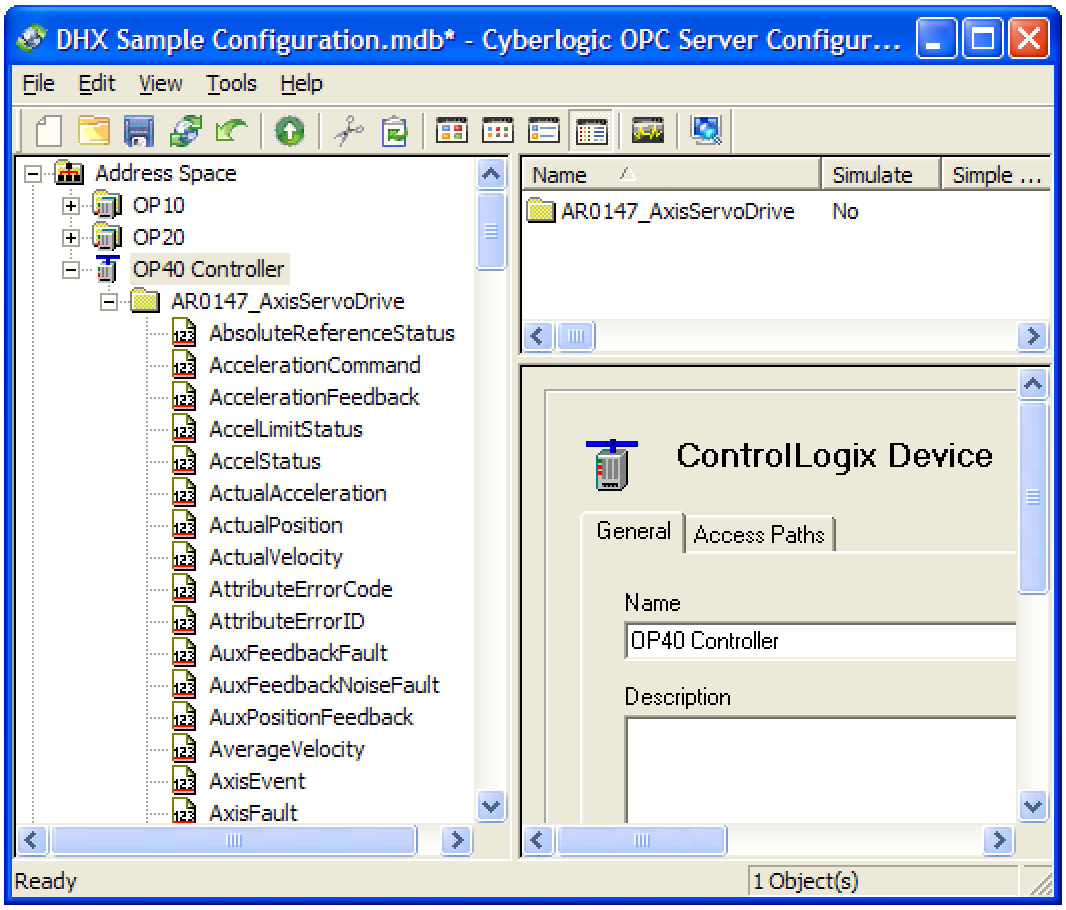
DirectImport

Click this button to import data items from one of the controllers listed on the Access Paths Tab.



After you choose the controller you wish to import from, the DirectImport window will show you all of the data items in the controller, with arrays and structures arranged in folders. You can browse through it, using the check boxes to select the items you wish to import.

After you have made all of your selections, click ***Finish***.



The items you selected will be imported into the Address Space tree, ready for use.

Caution!

DirectImport will try to arrange the folders and data items using the same structure as they have in the controller from which you are importing.

However, the Cyberlogic OPC Server allows no more than four levels of folders within a device. If the DirectImport would result in more than four levels of folders, then the imported structure will be flattened to fit it into the available number of levels.

DirectAccess

If this box is checked, the user is permitted to configure DirectAccess to the nodes associated with this device. The default state is checked, allowing DirectAccess.

When DirectAccess is enabled, a _Status folder will appear under this device in the client browser window. For more information on this folder and the status items it contains, refer to the Cyberlogic OPC Server Help.

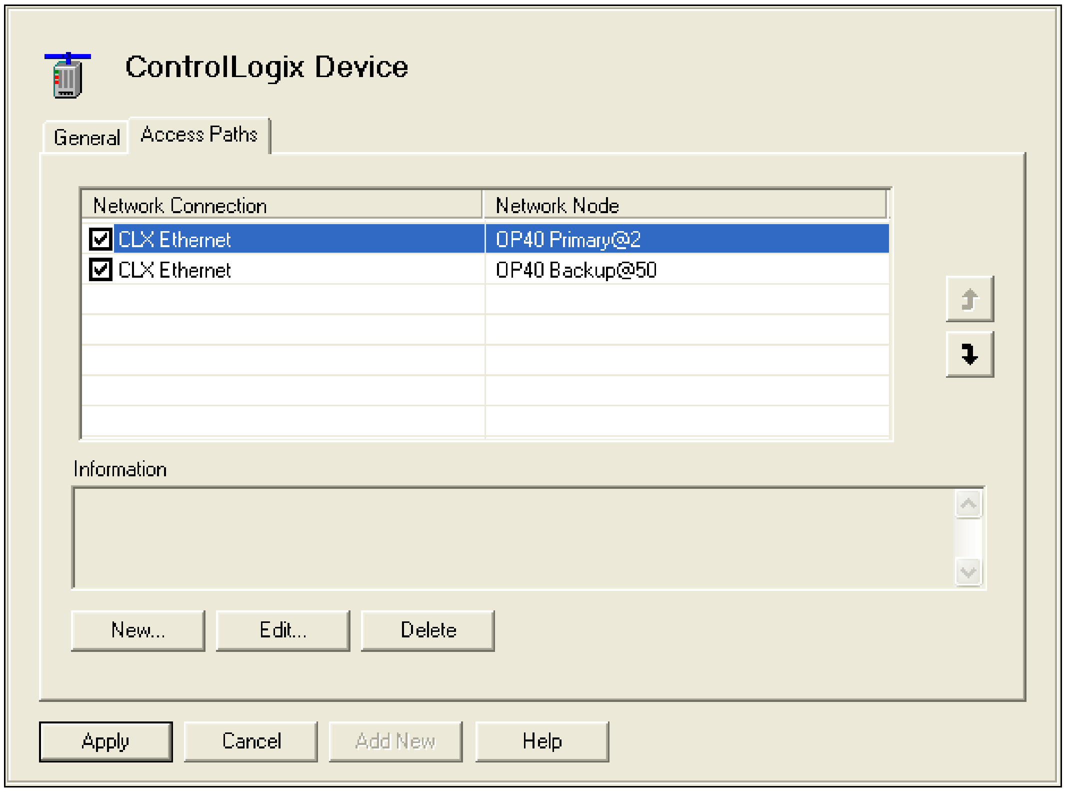
DirectAccess Disable Writes

If this box is checked, the server will not write data via DirectAccess to any of the nodes associated with this device. This does not affect writes through configured data items. The default state is checked, disabling DirectAccess writes.

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 or is disabled, the server switches to the highest access path that is available and enabled.

Note If you wish to use DirectImport, you must first configure one or more access paths for the device.



Network Connection	Network Node
☑ CLX Ethernet	OP40 Primary@2
☑ CLX Ethernet	OP40 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 connection associated with each of the access paths.

Network Node

The Network Node column displays the network node associated with each of the access paths.

Up and Down Arrows

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.

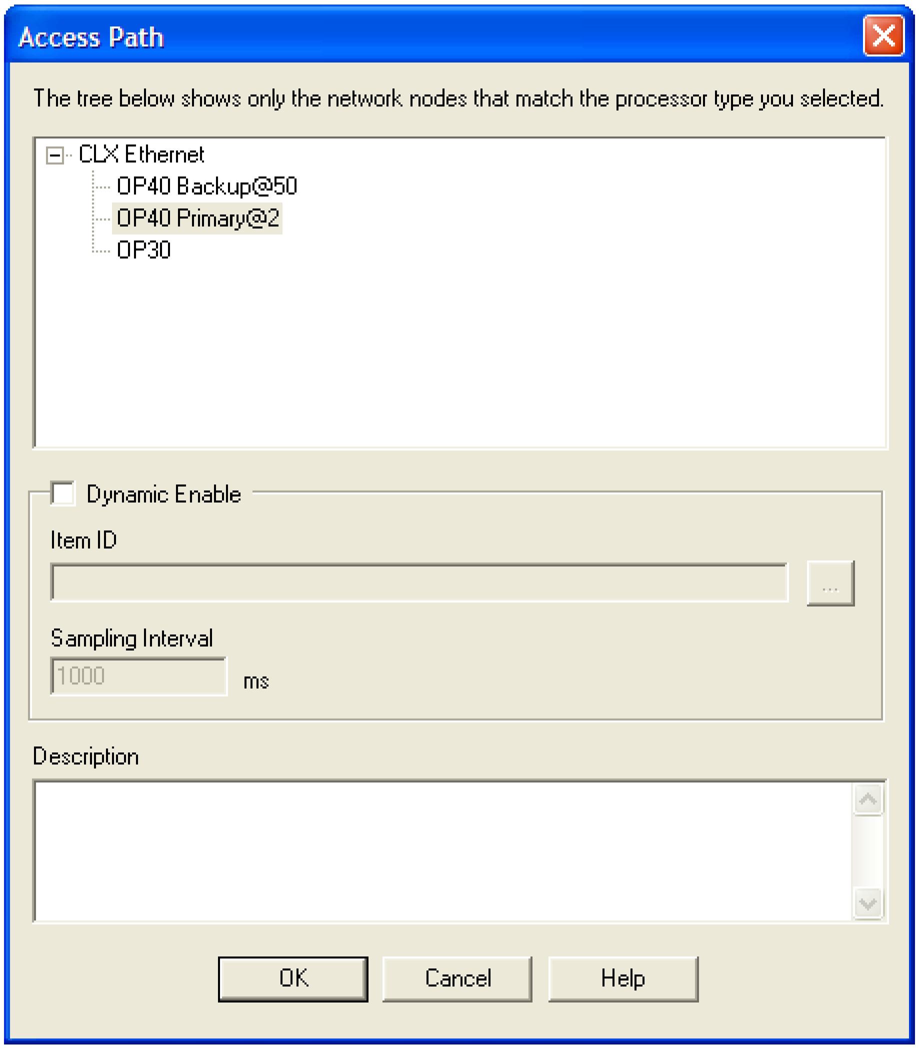
Information

If dynamic enable is configured for the highlighted access path, the enable data item will be shown here. In addition, if a description was entered for the access path, it will be displayed here.

This information is edited on the Access Path dialog box, described below.

New...

Click the ***New...*** button (or right-click inside the list window and select ***New...*** from the context menu) to create a new access path. The Access Path dialog box opens.



Select the network node for this access path.

If you check the Dynamic Enable box, you can specify an Item ID that will be used to control the enable status of the access path. Enter a data item or DirectAccess item ID in the box, or click the browse button to the right of it to browse for the desired item. You can use a Math & Logic item, if you wish. The Sampling Interval allows you to specify how often the enable item should be checked.

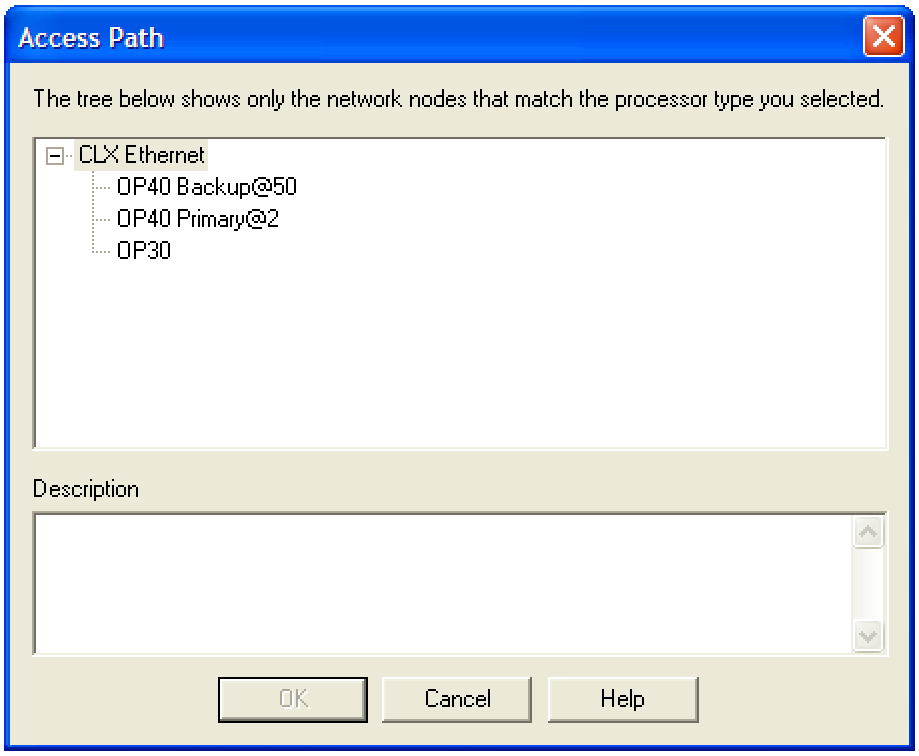
If the value of the enable item is false (Boolean false or a register value of 0), then the access path will be disabled. If the value is true (Boolean true or a nonzero register value), then the access path will be enabled.

You can also enter an optional description text of up to 255 characters.

Click ***OK*** when you are done.

Edit...

To modify an existing access path, select it and click the ***Edit...*** button (or right-click on the access path and select ***Edit...*** from the context menu). The Access Path dialog box opens.


Access Path
The tree below shows only the network nodes that match the processor type you selected.
CLX Ethernet
OP40 Backup@50
OP40 Primary@2
OP30
Description
OK
Cancel
Help

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

Delete

To delete an existing access path, select it and click the ***Delete*** button (or right-click and select ***Delete*** from the context menu).

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.

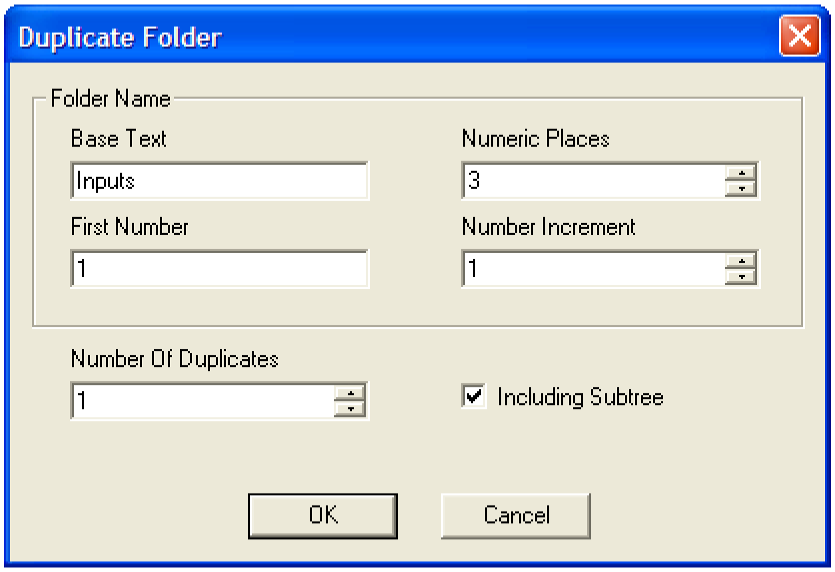
Caution! After you edit the configuration, you must open the ***File*** menu and select ***Save & Update Server***, or click the ***Save & Update Server*** toolbar button, for the changes you have made to take effect. Otherwise, the server will still be running with the old configuration.

Creating a new Folder

Right-click on an existing device or folder, and select ***New*** and then ***Folder*** from the context menu.

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 one. To do this, right-click on an existing folder and select ***Duplicate...*** from the context menu.



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

The above dialog box opens. You must specify how the duplicates are to be named by entering values for the ***Base Text***, ***First Number***, ***Numeric Places*** and ***Number Increment*** fields. To generate names for the duplicated folders, the editor begins with the base text and appends a number to it. The first duplicate uses the selected First Number value with the specified number of digits. This number is then incremented by the specified number for each of the remaining duplicates.

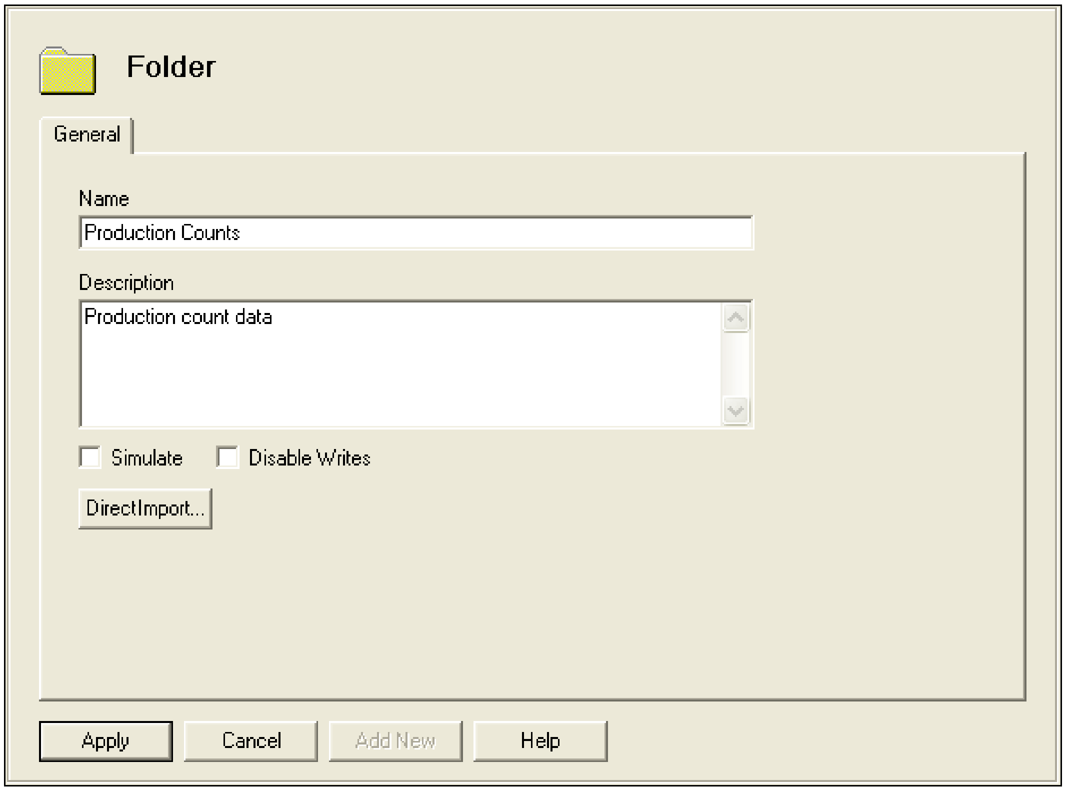
As an example, if Numeric Places is 3 and First Number is 2, the number 002 will be appended to the base text.

Use the ***Number Of Duplicates*** field to specify the number of folders you wish to create. If you want to duplicate all branches within the original folder, check the ***Including Subtree*** checkbox.

Deleting a Folder

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

General Tab


Folder

General

Name

Production Counts

Description

Production count data

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 or below. This provides a quick way to switch between real and simulated data for a large number of data items. Refer to the Cyberlogic OPC Server Help for a full discussion about 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 a higher level.

DirectImport

Click this button to import data items from a PLC. For a detailed description of how to use this function, refer to the DirectImport discussion in the Devices section.

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. The specific types of registers available vary from one PLC type to another, but all of the register types for all PLC models are available for configuration.

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

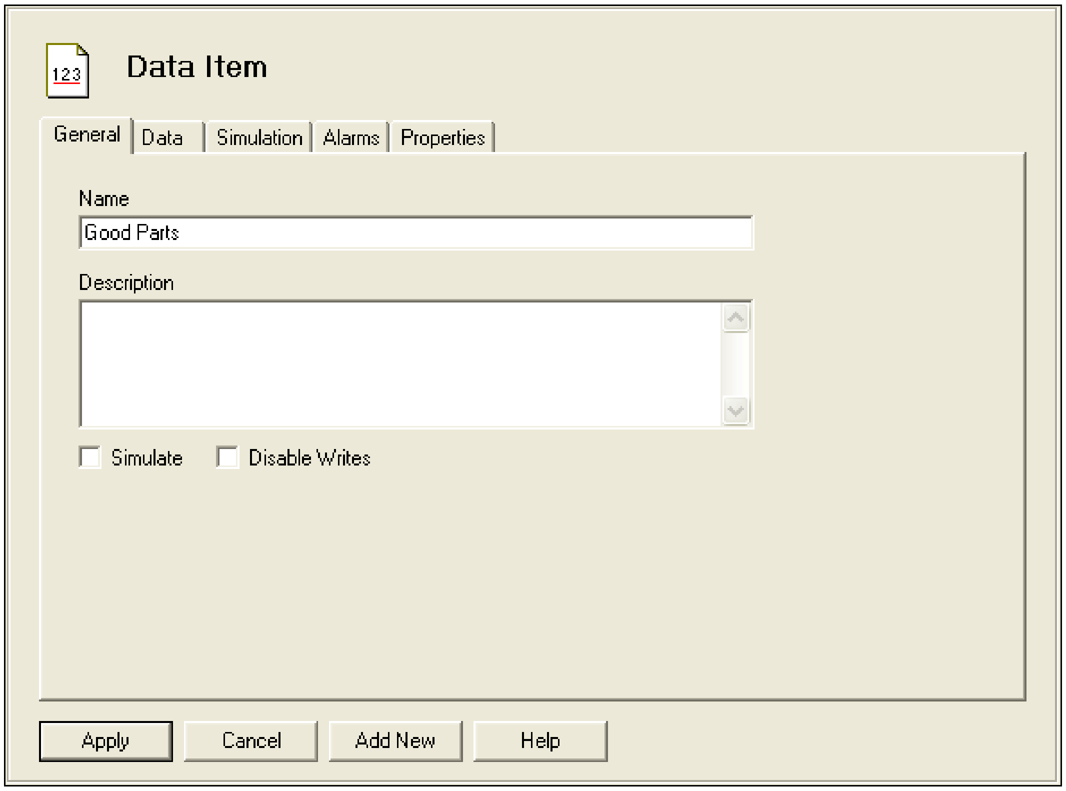
Creating a New Data Item

Right-click on an existing device or folder, and select ***New*** and then ***Data Item*** from the context menu.

Deleting a Data Item

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

General Tab



Name: Good Parts

Description:

☐ Simulate ☐ Disable Writes

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 the Cyberlogic OPC Server Help for a full discussion about simulating data.

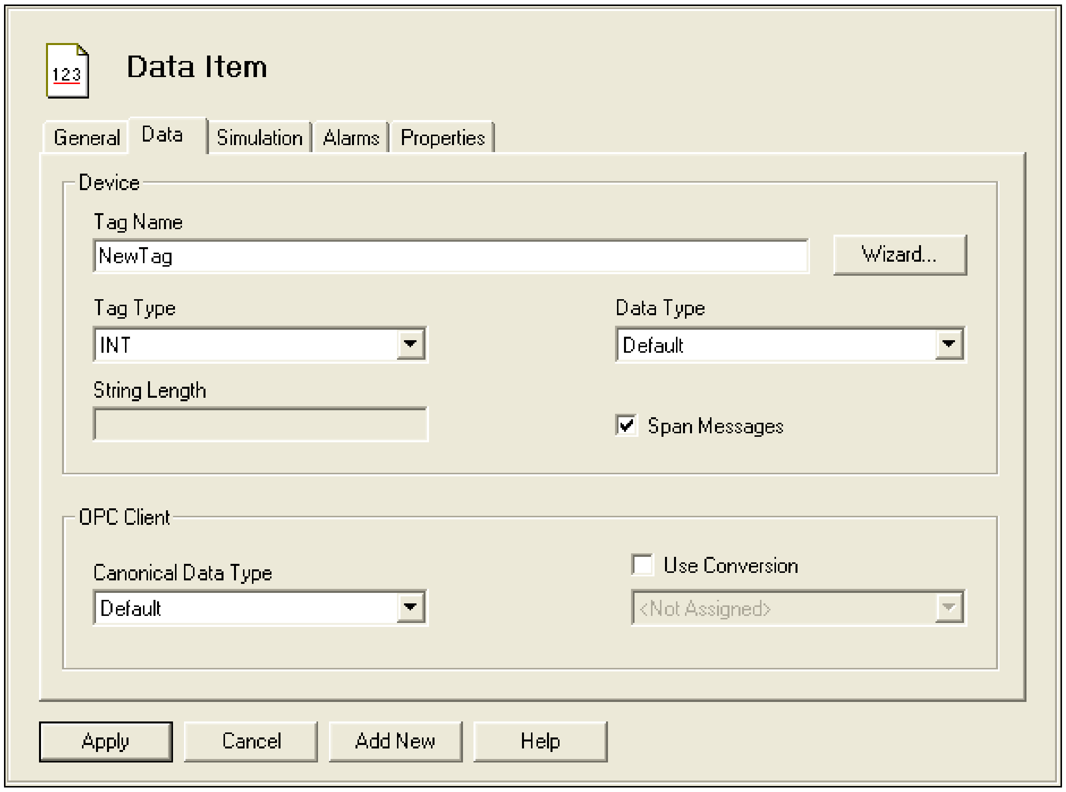
Note If the Simulate checkbox is grayed-out, it indicates that simulation has already been selected at a higher level.

Disable Writes

Checking this box disables 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 a higher level.

Data Tab



Device
Tag Name: NewTag
Tag Type: INT
Data Type: Default
String Length:
☑ Span Messages

OPC Client
Canonical Data Type: Default
☐ Use Conversion
<Not Assigned>

Tag Name

This is the tag name used for the data in the physical device.

In general, all addresses, as documented in Allen-Bradley manuals, are acceptable. However, the ControlLogix Driver Agent extends this syntax when it comes to specifying arrays, array elements and bit fields. For a complete list of all valid tag name syntaxes, refer to Appendix A: PLC Tag Names.

Wizard...

If you have difficulties with specifying valid addresses, click this button and the Address Wizard will help you to define the address correctly.

For details on running the wizard, refer to Appendix D: Address Wizard.

Tag Type

This is the type of data specified by the item identified in Tag Name.

All of the ControlLogix pre-defined structures are supported, along with user-defined types. Refer to Appendix B: Predefined ControlLogix Structures for more information on these structures.

Data Type

This drop-down allows you to select 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 characters 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 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 which case the Span Messages box must be checked.

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.

Canonical Data Type

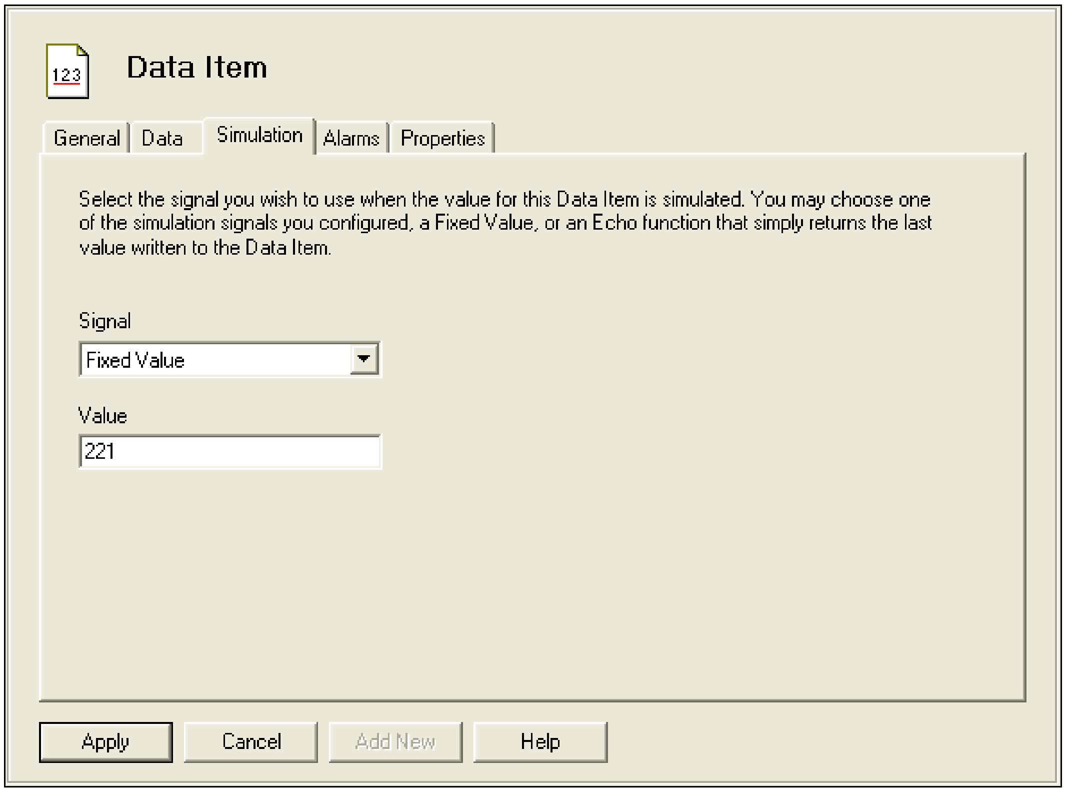
This selection specifies the default variant format (VT_xxx) in which the data will be sent to OPC clients. Each OPC client can request data in any variant format and consequently override the Canonical Data Type setting.

Type	Size in bits	Default Canonical Data Type	.NET Data Type	Description
Default				Default type based on the data item address
BIT	1	VT_BOOL	bool	1-bit boolean
SINT8	8	VT_I1	sbyte	Signed 8-bit integer -128 to 127
UINT8	8	VT_UI1	byte	Unsigned 8-bit integer 0 to 255
SINT16	16	VT_I2	short	Signed 16-bit integer -32,768 to 32,767
UINT16	16	VT_UI2	ushort	Unsigned 16-bit integer 0 to 65,535
SINT32	32	VT_I4	int	Signed 32-bit integer -2,147,483,648 to 2,847,483,647
UINT32	32	VT_UI4	uint	Unsigned 32-bit integer 0 to 4,294,967,295
SINT64	64	VT_I8	long	Signed 64-bit integer −9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
UINT64	64	VT_UI8	ulong	Unsigned 64-bit integer 0 to 18,446,744,073,709,551,615
FLOAT32	32	VT_R4	float	IEEE 32-bit floating point number ±3.4e±38
FLOAT64	64	VT_R8	double	IEEE 64-bit floating point number ±1.7e±308
BCD16	16	VT_UI2	ushort	BCD value; 0 to 9,999
BCD32	32	VT_UI4	uint	BCD value; 0 to 99,999,999
STRING	String size * 8	VT_BSTR	string	Zero terminated ASCII string of 8-bit characters
WSTRING	String size * 16	VT_BSTR	string	Zero terminated UNICODE string of 16-bit characters
FIELD	Field size	Best fitting VT_UIx or array of VT_UI1 if size > 64	Best fitting unsigned type or byte[ ] if size > 64	Multiple bit field

Use Conversion

Check this box to indicate 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



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: Fixed Value

Value: 221

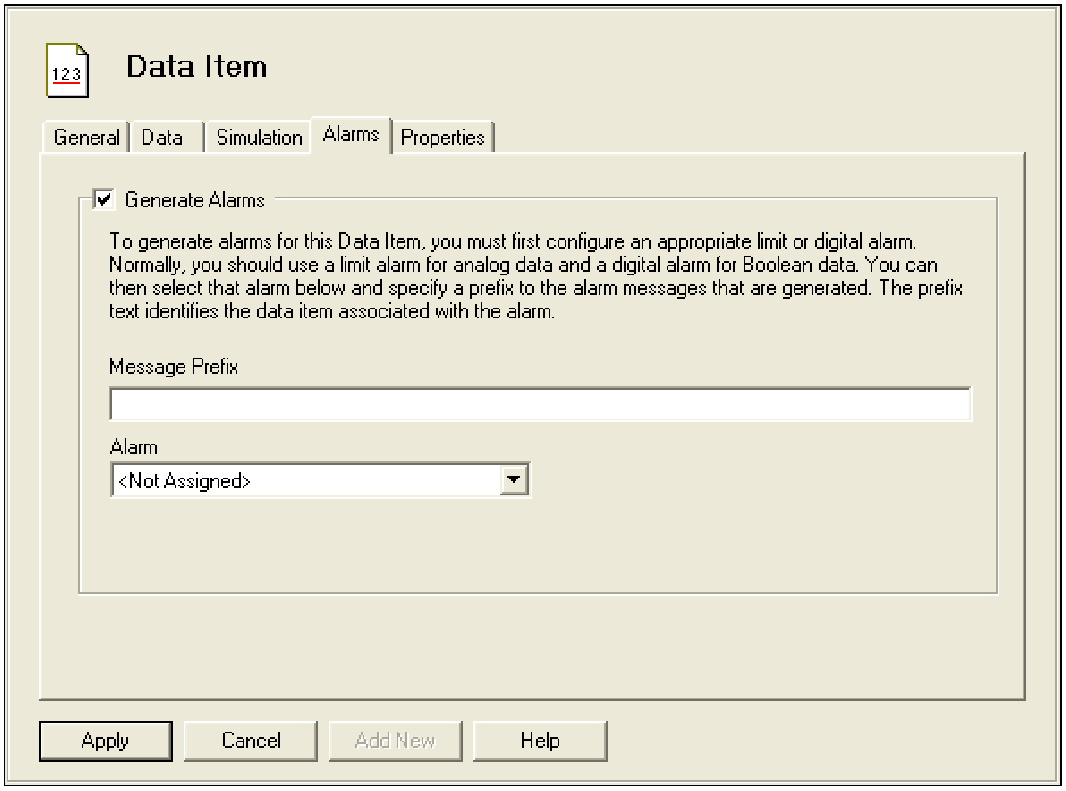
Signal

If you enabled simulation on the General tab or at a higher level, 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 simulation is enabled and the Signal field is set to Fixed Value, the data item will be set to this value.

Alarms Tab



[✓] Generate Alarms

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, generating alarms as appropriate.

Refer to the Cyberlogic OPC Server Help for a full discussion on creating and using alarms.

Message Prefix

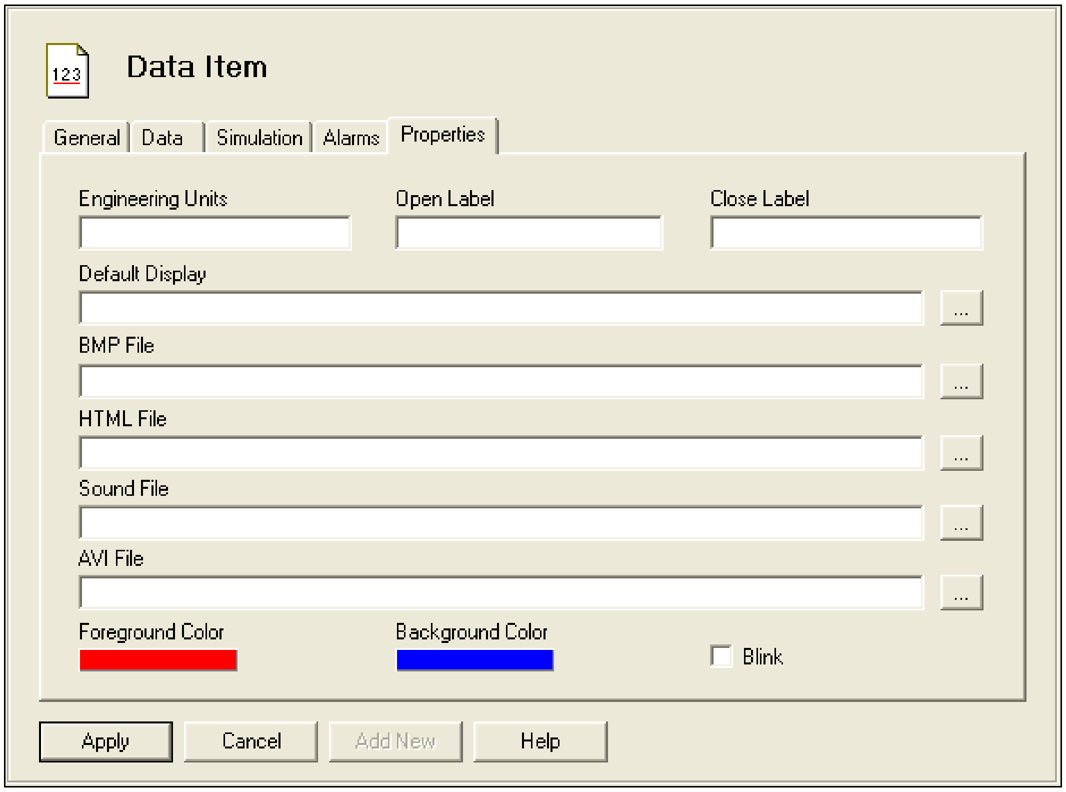
Enter the text for the first part of the alarm message. The second part will be the body text of the specific alarm that is generated.

Alarm

Select one of the previously-defined Alarm Definitions to serve as 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.HTML. 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 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, permitting it to generate alarms based on the value of data items.

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 also supports the OPC Alarms & Events specification. Consult your client software documentation to see what specifications it supports.

Database Operations

In addition to providing data to OPC clients in real time, the Cyberlogic OPC Server can store it in a database. The feature that does this is called Data Logger. Once the data is logged, it is available to any application that can access that database. It need not be an OPC client application.

Refer to the Data Logger Help for a full discussion.

OPC Crosslinks

OPC Crosslinks allow you to transfer data from an OPC server or PLC to other OPC servers or PLCs. The data item you read from is called the crosslink input. You may write its value to any number of data items, and these are called crosslink outputs.

Refer to the OPC Crosslink Help for a full discussion.

Saving and Undoing Configuration Changes

The Cyberlogic OPC Server Configuration Editor keeps track of recent configuration changes. Until you save these changes, you can revert to the previously-saved configuration. The editor supports two types of save operations. The standard Save operation saves the changes without updating the server or the connected clients. The Save & Update Server operation saves the changes and also updates the server and all connected clients.

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

Saving Configuration Changes Without Updating the Server

To save the configuration without updating the server, open the ***File*** menu and select ***Save***, or click the ***Save*** button on the toolbar. The changes will be saved but the server will still be running with the old configuration.

Saving Configuration Changes and Updating Server

To save the configuration and update the server, open the ***File*** menu and select ***Save & Update Server***, or click the ***Save & Update Server*** button on the toolbar.

Undoing Configuration Changes

To undo configuration changes and revert to the previously saved configuration, open the ***File*** menu and select ***Undo Changes***, or click the ***Undo Changes*** button on the 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 also from comma separated values files from other vendors' OPC servers and programming software.

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 ControlLogix 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 OPC 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

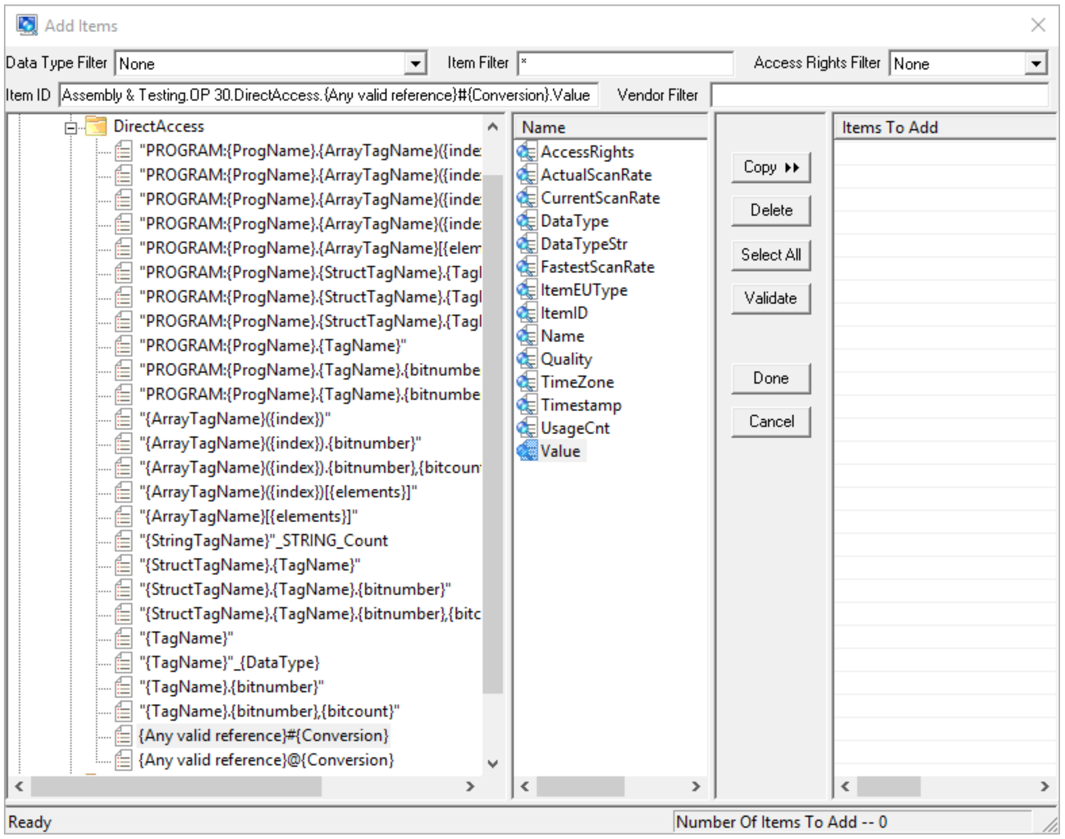
The Performance Monitor is a Microsoft diagnostic tool that the Cyberlogic drivers support. Its use is described in detail in the Cyberlogic OPC Server Help.

DirectAccess

At run time, in addition to the user-configured branches, the Cyberlogic OPC Server dynamically creates DirectAccess branches in its address space. These are created for both network nodes and devices.

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 network node configuration, and on a device-by-device basis as part of the device configuration.

Device DirectAccess



Data Type Filter: None
Item Filter: *
Access Rights Filter: None
Item ID: Assembly & Testing.OP 30.DirectAccess.{Any valid reference}#{Conversion}.Value
Vendor Filter:

DirectAccess
"PROGRAM:{ProgName}.{ArrayTagName}({inde
"PROGRAM:{ProgName}.{ArrayTagName}({inde
"PROGRAM:{ProgName}.{ArrayTagName}({inde
"PROGRAM:{ProgName}.{ArrayTagName}({inde
"PROGRAM:{ProgName}.{ArrayTagName}[{elem
"PROGRAM:{ProgName}.{StructTagName}.{Tagl
"PROGRAM:{ProgName}.{StructTagName}.{Tagl
"PROGRAM:{ProgName}.{StructTagName}.{Tagl
"PROGRAM:{ProgName}.{TagName}"
"PROGRAM:{ProgName}.{TagName}.{bitnumbe
"PROGRAM:{ProgName}.{TagName}.{bitnumbe
"{ArrayTagName}({index})"
"{ArrayTagName}({index}).{bitnumber}"
"{ArrayTagName}({index}).{bitnumber},{bitcoun
"{ArrayTagName}({index})[{elements}]"
"{ArrayTagName}[{elements}]"
"{StringTagName}"_STRING_Count
"{StructTagName}.{TagName}"
"{StructTagName}.{TagName}.{bitnumber}"
"{StructTagName}.{TagName}.{bitnumber},{bitc
"{TagName}"
"{TagName}"_{DataType}
"{TagName}.{bitnumber}"
"{TagName}.{bitnumber},{bitcount}"
{Any valid reference}#{Conversion}
{Any valid reference}@{Conversion}

Name
AccessRights
ActualScanRate
CurrentScanRate
DataType
DataTypeStr
FastestScanRate
ItemEUType
ItemID
Name
Quality
TimeZone
Timestamp
UsageCnt
Value

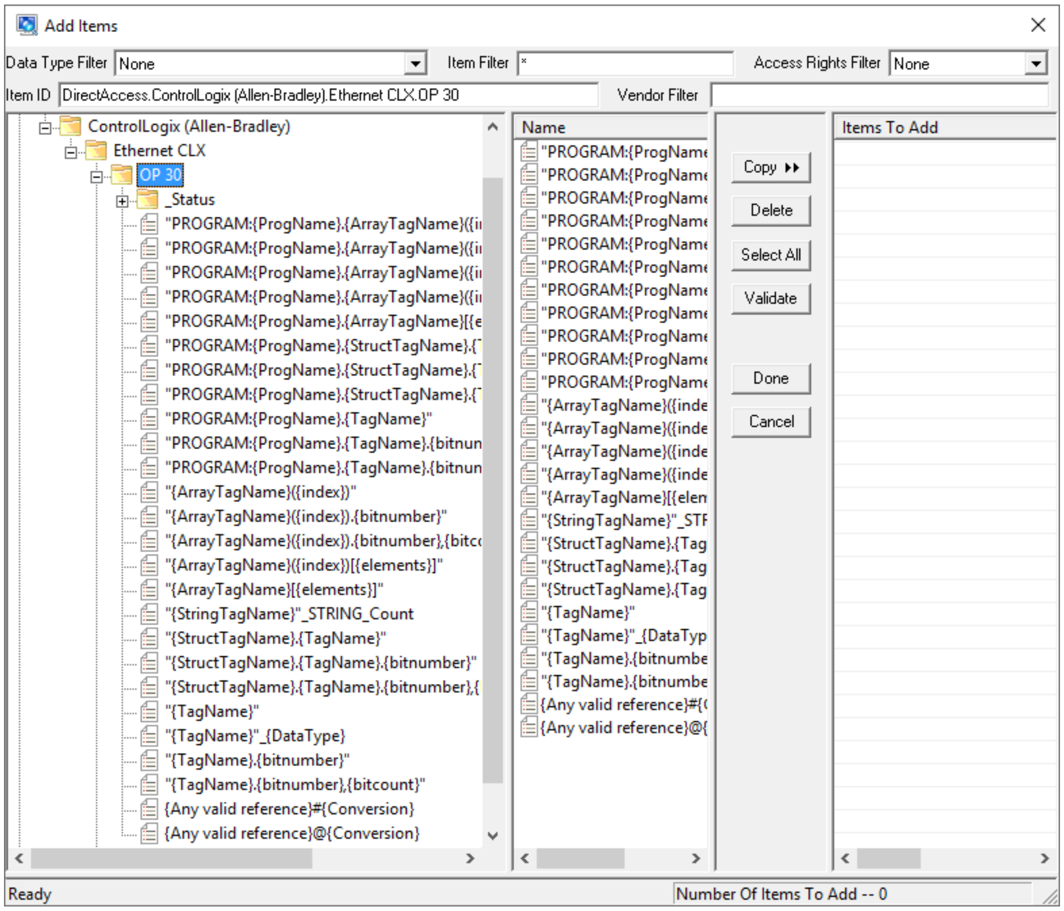
Items To Add

Ready
Number Of Items To Add -- 0

Each device in the address space will contain all of its configured data items, plus a DirectAccess branch, as shown in the example above. This branch will appear only for devices that have DirectAccess enabled. OPC clients can then use this branch to access any register in the device by directly specifying the register address.

In the DirectAccess branch for a device, the Cyberlogic OPC Server reports a list of hints about the types of data items that may exist on the selected device. These are not valid item addresses. Rather, they are just hints to help the user to specify a proper address. For more details on using these hints, refer to the Address Hints section.

Network Node DirectAccess



Data Type Filter: None
Item Filter: *
Access Rights Filter: None
Item ID: DirectAccess.ControlLogix (Allen-Bradley).Ethernet CLX.OP 30
Vendor Filter:

ControlLogix (Allen-Bradley)
Ethernet CLX
OP 30
_Status
"PROGRAM:{ProgName}.{ArrayTagName}({i
"PROGRAM:{ProgName}.{ArrayTagName}({i
"PROGRAM:{ProgName}.{ArrayTagName}({i
"PROGRAM:{ProgName}.{ArrayTagName}({i
"PROGRAM:{ProgName}.{ArrayTagName}[{e
"PROGRAM:{ProgName}.{StructTagName}.{
"PROGRAM:{ProgName}.{StructTagName}.{
"PROGRAM:{ProgName}.{StructTagName}.{
"PROGRAM:{ProgName}.{TagName}"
"PROGRAM:{ProgName}.{TagName}.{bitnun
"PROGRAM:{ProgName}.{TagName}.{bitnun
"{ArrayTagName}({index})"
"{ArrayTagName}({index}).{bitnumber}"
"{ArrayTagName}({index}).{bitnumber},{bitc
"{ArrayTagName}({index})[{elements}]"
"{ArrayTagName}[{elements}]"
"{StringTagName}"_STRING_Count
"{StructTagName}.{TagName}"
"{StructTagName}.{TagName}.{bitnumber}"
"{StructTagName}.{TagName}.{bitnumber},{
"{TagName}"
"{TagName}"_{DataType}
"{TagName}.{bitnumber}"
"{TagName}.{bitnumber},{bitcount}"
{Any valid reference}#{Conversion}
{Any valid reference}@{Conversion}

Ready
Number Of Items To Add -- 0

DirectAccess to network nodes is achieved through a branch called DirectAccess at the root of the address space. This branch acts like a device folder that contains all of the configured network connections.

As you can see in the example above, each network connection branch contains its configured network nodes. However, only network nodes that enable DirectAccess are present. OPC clients can then use this branch to access any bit or register in any configured network node by directly specifying its address.

For network nodes in the DirectAccess branch, the Cyberlogic OPC Server reports a list of hints about the types of data items that may exist on the selected node. These are not valid item addresses. Rather, they are just hints to help the user to specify a proper address.

Address Hints

In the ControlLogix example above, "{ArrayTagName}({index})".Value is an address hint. The {ArrayTagName} portion indicates the tag name for the data array that you wish to address. If, for example, this is an array of temperature values, then {ArrayTagName} might be TempSensor.

The ({index}) portion of the hint indicates the item within the array. If we have five temperature sensors, 0-4, and we want to view the third one, ({index}) would become (2).

The last part of the hint is the OPC property. If none is specified, the default is *.Value*. Here, we explicitly request the Value.

Therefore, to access the value of the third item in this array of temperature sensors using DirectAccess to the network node, we would edit the Item ID field to read:

DirectAccess.ControlLogix (Allen-Bradley).Ethernet CLX.OP30."TempSensor(2)".Value

To access the same register using DirectAccess to the device, you would edit the Item ID to read:

Assembly & Testing.OP30.DirectAccess."TempSensor(2)".Value

Here are some additional examples of valid DirectAccess item addresses for a ControlLogix processor:

"control.2,4"

A bit field four bits long starting at the third bit (bit 2) of a value named *control*

"recipe (3) [10]"

Ten elements, starting at element 3, from an array named *recipe*

"goodparts"_UINT16

Unsigned 16-bit integer named *goodparts*. (The _UINT16 portion is a data type override.)

You may want to use DirectImport to import some of the data items, to see how they are handled by the ControlLogix Driver Agent. This can help you to get the correct syntax when using DirectAccess.

Here is a brief explanation of the other address hints:

{Any valid reference}#{Conversion}

This form allows you to apply a previously configured data conversion to a raw register value in order to convert it into a form that is more useful to the client. When the conversion name is preceded by a # sign, the canonical data type for the requested data will always be VT_R8 (64-bit floating point).

{Any valid reference}@{Conversion}

This form allows you to apply a previously configured data conversion to a raw register value in order to convert it into a form that is more useful to the client. When the conversion name is preceded by an @ sign, the canonical data type will match the data type of the requested register or the specified {Data Type Override}.

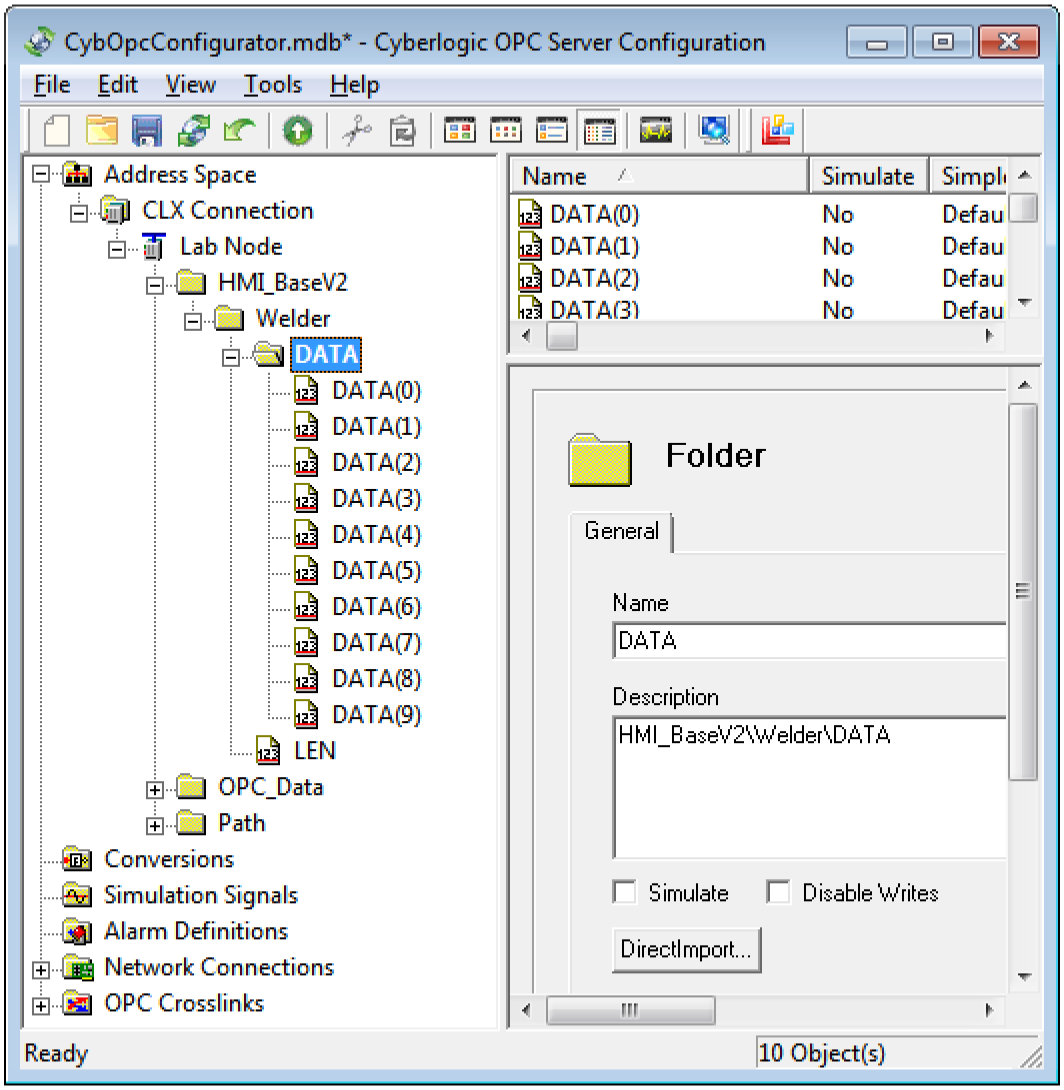
Note

The address hints are shown enclosed in double-quotes, and the item address you specify in place of the hint must also be enclosed in double-quotes. If a data type override is used, it is not enclosed in the double-quotes.

Previous versions of the Cyberlogic OPC Server did not require the double-quotes, but had the requirement that any periods (.) in the address had to be replaced with a forward slash (/). This format is still valid, for compatibility with existing configurations. However, the double-quote format is preferred for new configurations.

Working with Strings

When you DirectImport a string, the import tool treats it as an array of characters and a register containing the string's length. This allows you to work with the individual characters, if you wish. However, you can also work with the entire string.



Name	Simulate	Simpl
DATA(0)	No	Defau
DATA(1)	No	Defau
DATA(2)	No	Defau
DATA(3)	No	Defau

Name: DATA

Description: HMI_BaseV2\Welder\DATA

Simulate [ ] Disable Writes [ ]

In the example shown, ***Welder*** is the string we want to use. Select the folder called ***DATA*** and look at the description on the ***General*** tab. DirectImport inserts the path to the string as the description, and this provides you with the tag name to use in place of the hint in the DirectAccess Item ID.

However, you must replace the backslash characters with periods, and you must explicitly declare the type and length. In this example, the item address would be:

"HMI_BaseV2.Welder.DATA"_STRING_10

If you omit the _STRING data type override, you will get the ASCII numeric value of the first character. If you omit the length, it defaults to one, and you will get only the first character. If you are not sure of the length, you can use a large value. The maximum is 82.

If you want a single character, address it as:

"HMI_BaseV2.Welder.DATA(2)"_STRING

to access the third character as an ASCII character, or

"HMI_BaseV2.Welder.DATA(2)"

to access the third character as an ASCII numeric value.

DirectAccess Address Formats

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

- {Tag Name}
- {Tag Name}_{Data Type Override}
- {Tag Name}#{Conversion}
- {Tag Name}_{Data Type Override}#{Conversion}
- {Tag Name}@{Conversion}
- {Tag Name}_{Data Type Override}@{Conversion}

Tag Name

The Tag Name portion may be any tag name that is acceptable in the Tag Name field of a data item. This portion is enclosed in double-quotes.

The PLC item tag requirements are discussed in Appendix A: PLC Tag Names and Appendix B: Predefined ControlLogix Structures.

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. The data type override portion is optional, and if used, is not enclosed in double-quotes.

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.

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 table below shows all supported data types.

Caution! The Data Type Override field requires you to use the form in the Type column, not the Canonical Data Type. For example, if you want 32-bit floating point format, you must specify ***FLOAT32***. The canonical form ***VT_R4*** will not work.

Type	**Size in bits**	**Default Canonical Data Type**	**.NET Data Type**	**Description**
Default				Default type based on the data item address
BIT	1	VT_BOOL	bool	1-bit boolean
SINT8	8	VT_I1	sbyte	Signed 8-bit integer -128 to 127
UINT8	8	VT_UI1	byte	Unsigned 8-bit integer 0 to 255
SINT16	16	VT_I2	short	Signed 16-bit integer -32,768 to 32,767
UINT16	16	VT_UI2	ushort	Unsigned 16-bit integer 0 to 65,535
SINT32	32	VT_I4	int	Signed 32-bit integer -2,147,483,648 to 2,847,483,647
UINT32	32	VT_UI4	uint	Unsigned 32-bit integer 0 to 4,294,967,295
SINT64	64	VT_I8	long	Signed 64-bit integer −9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
UINT64	64	VT_UI8	ulong	Unsigned 64-bit integer 0 to 18,446,744,073,709,551,615
FLOAT32	32	VT_R4	float	IEEE 32-bit floating point number ±3.4e±38
FLOAT64	64	VT_R8	double	IEEE 64-bit floating point number ±1.7e±308
BCD16	16	VT_UI2	ushort	BCD value; 0 to 9,999
BCD32	32	VT_UI4	uint	BCD value; 0 to 99,999,999
STRING	String size * 8	VT_BSTR	string	Zero terminated ASCII string of 8-bit characters
WSTRING	String size * 16	VT_BSTR	string	Zero terminated UNICODE string of 16-bit characters
FIELD	Field size	Best fitting VT_UIx or array of VT_UI1 if size > 64	Best fitting unsigned type or byte[ ] if size > 64	Multiple bit field

Conversion

This feature allows you to apply a previously configured data conversion to a raw register value in order to convert it into a form that is more useful to the client. For example, you can change a transducer's voltage value into a pressure value in psi.

The conversion name must be preceded by either a # sign or an @ sign. With the # sign, the canonical data type for the requested data will always be VT_R8 (64-bit floating point). With the @ sign, the canonical data type will match the data type of the requested register or the specified {Data Type Override}. Here are a couple of examples:

DirectAccess.ControlLogix (Allen-Bradley).Ethernet CLX.OP30."Sensor(2)".Value#100%
Assembly & Testing.OP30.DirectAccess."TempSensor(2)".Value_SINT16@P200

In the first example, the canonical data type is VT_R8, while in the second example, it is VT_I2. The *100%* and *P200* are the names of the conversions.

Event Viewer

During startup and operation, the Cyberlogic OPC Server may detect problems or other significant events. When a noteworthy event is detected, the server sends an appropriate message to the Windows Event Logger. You can view these messages using the Windows Event Viewer. Its use is described in detail in the Cyberlogic OPC Server Help.

For an explanation of the error messages that may be logged by the ControlLogix Driver Agent, refer to the ControlLogix Driver Agent Messages section.

ControlLogix Driver Agent Messages

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

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

A necessary registration DLL could not be loaded. This may indicate a corrupted installation. Repair the existing installation, or remove and reinstall the software.

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

A registration check indicated that the software's evaluation time has expired. Run the Activation Wizard to authorize further use of the software.

This is a <Number of Hours>-hour promotional copy of the ControlLogix Driver Agent. The Cyberlogic OPC Server started at <Start Time> and the agent will stop at <Stop Time>.

This is a time-limited installation of the software. After the stop time, the driver agent will not allow any further I/O operations.

This is a promotional copy of the ControlLogix Driver Agent. The allowed operation time has expired. The I/O operations of the server have been disabled.

This is a time-limited installation of the software. The stop time has been reached or exceeded, so the driver agent will not allow any further I/O operations.

The Cyberlogic License Server failed to respond with valid license information. The I/O operations of the ControlLogix Driver Agent have been disabled. Contact the manufacturer's technical support.

The driver agent experienced a problem when it tried to contact the Cyberlogic License Server. If the license server is not running, start it and then try restarting the driver. If the license server is already running, contact Cyberlogic Tech Support.

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

The driver agent failed to allocate needed memory. This is a fatal error. Close other open applications or add more memory to the system, and then try to restart the OPC server.

Unexpected error in <Function Name>. Please contact the manufacturer's technical support.

Indicates a possible programming bug in the OPC server. Contact Cyberlogic Tech 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 OPC server. Contact Cyberlogic Tech Support for more information on a possible solution.

Frequently Asked Questions

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

I have a 1784-PKTX card connected to the Data Highway Plus network. I successfully installed the DHX OPC Server, but when I try the auto configuration, the editor fails to detect any network connections. What is the problem?

The ControlLogix driver agent does not support the auto configuration feature. You must configure the network connections and network nodes manually.

I have a Plug and Play adapter card for Data Highway (or Data Highway Plus) installed in my system. There is no communication and the DHX demo software doesn't show any of the existing nodes.

The Plug and Play cards for Allen-Bradley networks all default to node address 0. 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 DHX Configuration Editor to change the adapter's node address to a different value.

I have ControlLogix controllers on a Data Highway Plus network. When I try the auto configuration, the editor fails to detect any of them, but does "detect" a bunch of nodes on the DHX Network Connections for Allen-Bradley branch and shows them as controllers of "Unknown" type. What is the problem?

ControlLogix nodes will not show up under the ControlLogix Network Connections branch because the ControlLogix Driver Agent does not support the auto configuration feature.

The DHX Driver Agent will detect the nodes, but because Logix processors are not supported by that driver agent, it will show them as "Unknown". That is, they are not of a type that the DHX Driver Agent knows how to support. Auto configuration will do the same for any unknown controller type. For example, if Allen-Bradley introduces a new type of SLC, auto configuration will not be able to identify it and will show it as "Unknown". This would also be true for any third-party Data Highway device that the utility detects.

What is the optimum setting for the System Overhead Time Slice (SOTS)?

The SOTS allocates time between the communication tasks (such as OPC) and the controller tasks (such as solving ladder logic). The SOTS is set in RSLogix5000 and is expressed as the percentage of time to be allocated to the communication tasks. Its default value is 10%. Thus, 10% of the scan time will go to communication tasks and 90% will go to servicing controller tasks.

The value you should use will depend on the priority you place on communication tasks as compared with the priority you place on controller tasks. SOTS values above 50% are appropriate for applications where communication has the higher priority. Values below 50% should be used when controller tasks have the higher priority. A reasonable value to start with would be in the range of 30-60%. From there, you should observe the system performance and adjust the value as needed to find the optimum for your application.

I had another brand of OPC server installed on the same system as Cyberlogic's OPC Server. After I uninstalled the other server, the Cyberlogic

Server stopped working. What happened?

There are certain OPC Foundation common components that are installed and used by all OPC servers. If the OPC server you removed had a poorly-written uninstall program, it may have removed these common components even though they were still in use by the Cyberlogic OPC Server. To correct this, run the DHX OPC Server installation program and follow the procedure for repairing the installation.

APPENDIX A: PLC TAG NAMES

The allowed syntax for the Tag Name field on the Data Tab of a data item depends upon the PLC type. In general, the Cyberlogic OPC Server supports the standard naming conventions as used by Allen-Bradley. This appendix describes this syntax for the Logix family.

However, the ControlLogix Driver Agent extends this syntax when it comes to specifying arrays, array elements and bit fields.

Arrays

Single-dimensional arrays are supported for data types other than 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 one, in which case you would set the Lower Bound value to a 1.

Array Elements

Some complex data types may contain built-in arrays of certain type. To specify an element of such array, use the following syntax:

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

Bit Fields

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

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

Examples

Sta3PartPresent	Tag named *Sta3PartPresent*
goodparts	Tag named *goodparts*
current_value	Tag named *current_value*
control.2,4	A bit field four bits long starting at the third bit (bit 2) of a value named *control*
control.2 [4]	A bit array four bits long starting at the third bit (bit 2) of a value named *control*
control.1	A bit field one bit long starting at the second bit (bit 1) of a value named *control*
PartCount.ACC	The *ACC* field (accumulated value) for a counter instance named *PartCount*
sta2a_watchdog.DN	The *DN* field (timer done bool) for a timer instance named *sta2a_watchdog*
Blob.abc	The *abc* field for a user-defined structure instance named *Blob*
recipe (3) [10]	Ten elements, starting at element 3, from an array named *recipe*
Counts (7)	Element 7 from an array named *Counts*. The implied element count is 1.
line_pressures (0) [2]	Two elements, starting at element 0, from an array named *line_pressures*
Sta7Pids (1).SP	*SP* field (setpoint) from element 1 from a PID array named *Sta7Pids*
Sta7Pids (2).DATA (1) [3]	Three elements, starting at element 1, from the *DATA* field of element 2 of a PID array named *Sta7Pids*
ThreeDArray (3)(6)(1)	Element 3,6,1 of a three-dimensional array named *ThreeDArray*
MyTimerArray (2).MyTimers (6).PRE.6	Element 2 of *MyTimerArray* which contains a field named *MyTimers*. Access element 6 of *MyTimers*. Access bit 6 of the *PRE* field of that timer.
MyComplexData.MyDintArray (3).0 [128]	Structure called *MyComplexData* which contains a field named *MyDintArray*. Access

element 3 of *MyDintArray*. Bit array 128 bits long starting at bit 0.

Syntax

The following tables define the syntax used for the PLC tag names.

Notation

{ }	An optional part of the syntax.
[dimension,firstindex]	Identifies an array. The number of elements in the array is specified by dimension. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Both atomic and complex data arrays are supported.
(index)	Used to identify a specific element in an array.
atomic data	A piece of data that is not an entire structure or array. It must be of type BOOL, SINT, INT, DINT or REAL.
structure	A data map consisting of one or more fields. Each field can be atomic data, another structure, an array of atomic data or an array of structures. It is not valid to access an entire structure. It is only valid to access fields in the structure.
structure.field	Identifies a specific field within a structure. A field can be atomic data, another structure, an array of atomic data or an array of structures.
adTag	Atomic data that has global scope
PROGRAM:progname.adTag	Atomic data that has program scope.
cdTag	A complex piece of data that has global scope. It is not valid to attempt to access an entire structure. It is only valid to access fields in the structure.
cdTag.field	A field in a structure. The field can be atomic data, another structure, an array of atomic data or an array of structures. Global scope.
PROGRAM:progname.cdTag	A complex piece of data that has program scope. It is not valid to attempt to access an entire structure. It is only valid to access fields in the structure.
PROGRAM:progname.cdTag.field	A field in a structure. The field can be atomic data, another structure, an array of atomic data or an array of structures. Program scope.

Accepted Syntaxes

adTag	An atomic piece of data. Global scope.
adTag.bitnumber	A bit in either an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). Global scope.
adTag.bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Global scope.
adTag.bitnumber [dimension1{,firstindex1}]	A bit array in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array cannot extend past the end of SINT, INT or DINT. Global scope.
aradTag [dimension1{,firstindex1}] {[dimension2{,firstindex2}] {[dimension3{,firstindex3}]}}	An array of atomic data. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Logix processors support 1, 2 and 3-dimensional arrays. The server will convert arrays of 2 or more dimensions into single dimensional arrays. Global scope.
aradTag (index1){(index2){(index3)}}	Identifies a specific element in an array of atomic data. In ControlLogix, the first index of each dimension is zero (0). Global scope.
aradTag (index1){(index2){(index3)}} .bitnumber	Identifies a bit in a specific element in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). In ControlLogix, the first index of each dimension is zero (0). Global scope.
aradTag (index1){(index2){(index3)}} .bitnumber,bitcount	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Global scope.

aradTag (index1){(index2){(index3)}} .bitnumber [dimension1{,firstindex1}]	Identifies a bit array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array can extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Global scope.
PROGRAM:progname.adTag	An atomic piece of data. Program scope.
PROGRAM:progname.adTag .bitnumber	A bit in either an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). Program Scope.
PROGRAM:progname.adTag .bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Program scope.
PROGRAM:progname.adTag .bitnumber [dimension1{,firstindex1}]	A bit array in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array cannot extend past the end of SINT, INT or DINT. Program scope.
PROGRAM:progname.aradTag [dimension1{,firstindex1}] {[dimension2{,firstindex2}] {[dimension3{,firstindex3}]}}	An array of atomic data. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Logix processors support 1, 2 and 3-dimensional arrays. The server will convert arrays of 2 or more dimensions into single dimensional arrays. Program Scope.
PROGRAM:progname.aradTag (index1){(index2){(index3)}}	Identifies a specific element in an array of atomic data. In ControlLogix, the first index of each dimension is zero (0). Program Scope.
PROGRAM:progname.aradTag (index1){(index2){(index3)}} .bitnumber	Identifies a bit in a specific element in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). In ControlLogix, the first index of each dimension is zero (0). Program Scope.

PROGRAM:progname.aradTag (index1){(index2){(index3)}} .bitnumber,bitcount	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Program Scope.
PROGRAM:progname.aradTag (index1){(index2){(index3)}} .bitnumber [dimension1{,firstindex1}]	Identifies a bit array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array can extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Program scope.

Structures with atomic fields

cdTag.adField	An atomic piece of data. Global scope.
cdTag. adField.bitnumber	A bit in either an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). Global scope.
cdTag. adField.bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Global scope.
cdTag. adField.bitnumber [dimension1{,firstindex1}]	A bit array in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array cannot extend past the end of SINT, INT or DINT. Global scope.
cdTag. adField.bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Global scope.
cdTag.aradField [dimension1{,firstindex1}] {[dimension2{,firstindex2}] {[dimension3{,firstindex3}]}}	An array of atomic data. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Logix processors support 1, 2 and 3-dimensional arrays. The server will convert arrays of 2 or more dimensions into single dimensional arrays. Global scope.
cdTag.aradField (index1){(index2){(index3)}}	Identifies a specific element in an array of atomic data. In ControlLogix, the first index of each dimension is zero (0). Global scope.
cdTag.aradField (index1){(index2){(index3)}} .bitnumber	Identifies a bit in a specific element in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). In ControlLogix, the first index of each dimension is zero (0). Global scope.

cdTag.aradField (index1){(index2){(index3)}} .bitnumber,bitcount	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Global scope.
cdTag.aradField (index1){(index2){(index3)}} .bitnumber [dimension1{,firstindex1}]	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array can extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Global scope.
PROGRAM:progname .cdTag.adField	An atomic piece of data. Program scope.
PROGRAM:progname .cdTag.adField.bitnumber	A bit in either an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). Program Scope.
PROGRAM:progname .cdTag.adField .bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Program Scope.
PROGRAM:progname .cdTag.adField .bitnumber [dimension1{,firstindex1}]	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array cannot extend past the end of SINT, INT or DINT. Program Scope.
PROGRAM:progname .cdTag. aradField [dimension1{,firstindex1}] {[dimension2{,firstindex2}] {[dimension3{,firstindex3}]}}	An array of atomic data. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Logix processors support 1, 2 and 3-dimensional arrays. The server will convert arrays of 2 or more dimensions into single dimensional arrays. Program Scope.
PROGRAM:progname .cdTag.aradField (index1){(index2){(index3)}}	Identifies a specific element in an array of atomic data. In ControlLogix, the first index of each dimension is zero (0). Program Scope.

PROGRAM:progname .cdTag.aradField (index1){(index2){(index3)}}.bit number	Identifies a bit in a specific element in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). In ControlLogix, the first index of each dimension is zero (0). Program Scope.
PROGRAM:progname .cdTag.aradField (index1){(index2){(index3)}} .bitnumber,bitcount	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Program Scope.
PROGRAM:progname .cdTag.aradField (index1){(index2){(index3)}} .bitnumber [dimension1{,firstindex1}]	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array can extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Program Scope.

Array of structures with atomic fields

arcdTag (index1){(index2){(index3)}} .adField	An atomic piece of data. Global scope.
arcdTag (index1){(index2){(index3)}} .adField.bitnumber	A bit in either an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). Global scope.
arcdTag (index1){(index2){(index3)}} .adField.bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Global scope.
arcdTag (index1){(index2){(index3)}} .adField.bitnumber [dimension1{,firstindex1}]	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array cannot extend past the end of SINT, INT or DINT. Global scope.
arcdTag (index1){(index2){(index3)}} .aradField [dimension1{,firstindex1}] {[dimension2{,firstindex2}] {[dimension3{,firstindex3}]}}	An array of atomic data. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Logix processors support 1, 2 and 3-dimensional arrays. The server will convert arrays of 2 or more dimensions into single dimensional arrays. Global scope.
arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}}	Identifies a specific element in an array of atomic data. In ControlLogix, the first index of each dimension is zero (0). Global scope.
arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}} .bitnumber	Identifies a bit in a specific element in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). In ControlLogix, the first index of each dimension is zero (0). Global scope.
arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}} .bitnumber,bitcount	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Global scope.

arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}} .bitnumber [dimension1{,firstindex1}]	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array can extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Global scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .adField	An atomic piece of data. Program scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .adField.bitnumber	A bit in either an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). Program Scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .adField.bitnumber,bitcount	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. Program Scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .adField.bitnumber [dimension1{,firstindex1}]	A bit field in an SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array cannot extend past the end of SINT, INT or DINT. Program Scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}}. aradField [dimension1{,firstindex1}] {[dimension2{,firstindex2}] {[dimension3{,firstindex3}]}}	An array of atomic data. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. Logix processors support 1, 2 and 3-dimensional arrays. The server will convert arrays of 2 or more dimensions into single dimensional arrays. Program Scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}}	Identifies a specific element in an array of atomic data. In ControlLogix, the first index of each dimension is zero (0). Program Scope.

PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}} .bitnumber	Identifies a bit in a specific element in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). In ControlLogix, the first index of each dimension is zero (0). Program Scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}} .bitnumber,bitcount	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). bitcount specifies the number of bits in the bit field and must be one (1) or larger. A bit field cannot extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Program Scope.
PROGRAM:progname .arcdTag (index1){(index2){(index3)}} .aradField (index1){(index2){(index3)}} .bitnumber [dimension1{,firstindex1}]	Identifies a bit field in an array. This is only valid for arrays of SINT (bitnumber 0-7), INT (bitnumber 0-15) or DINT (bitnumber 0-31). dimension1 specifies the number of bits in the bit array and must be one (1) or larger. The index of the first element is specified by firstindex. If no firstindex is given, a zero (0) is assumed. A bit array can extend past the end of SINT, INT or DINT. In ControlLogix, the first index of each dimension is zero (0). Program Scope.

APPENDIX B: PREDEFINED CONTROLLOGIX STRUCTURES

The Cyberlogic OPC Server now features DirectImport of the PLC tags. We recommend that you use this feature, rather than attempting to set up the structures manually. Because these structures may vary from one controller firmware version to another, DirectImport is the best way to be sure of using the correct syntax for your revision.

APPENDIX C: CIP PATHS

A key part of configuring network communications is specifying the Control and Information Protocol (CIP) path. This appendix discusses the syntax of the CIP Path and the use of the CIP Path Wizard to simplify the path configuration.

For additional information on CIP path configuration, refer to Allen-Bradley's *ControlLogix Data Highway Plus-Remote I/O Communication Interface Module 1756-DHRIO User Manual*, available as A-B publication number 1756-UM514B-EN-P.

General Syntax

This section explains the general syntax used for all CIP paths. The specific format used for a given configuration will depend on the driver, hardware and network setup. Refer to the example sections for details.

The CIP path is an addressing method used to identify a target device by specifying each step of the route to that device. By default, all numeric values in the CIP path are assumed to be decimal, unless specified otherwise. Path fields are separated by commas.

A CIP path can have two forms:

- address, port, address, etc.
- port, address, port, address, etc.

Extra <port, address> pairs can be added as necessary to either form.

In the second form, the leading port number is disregarded by the Cyberlogic drivers. This form is supported for compatibility with RSLinx/RSLogix 5000.

Address Fields

The address fields can take many forms, depending on the type of network used. These are shown below with examples of each.

Ethernet

Form	Example
<IP Address>	192.168.1.12
<IP Address>:<IP Port>	192.168.1.12:32767
<DNS Name>	sta3r
<DNS Name>:<IP Port>	sta3r:16385

Note In the table above, the forms that include <IP Port> are to be used when it is necessary to override the default IP port. The specified IP port is part of the address field and does not take the place of the port field in the CIP path syntax.

Data Highway Plus Node

Form	**Example**
<Octal station number>	8#17
<Decimal station number>	15

DF1 Network

Form	**Example**
<Decimal station number>	254

ControlNet Network

Form	**Example**
<Decimal station number>	99

ControlLogix backplane

Form	**Example**
<Decimal slot number>	1

Port Fields

PLCs, modules and other components may have ports associated with them, through which the message can be routed to another address. The most common port assignments are shown in the table below.

In the case of Data Highway Plus ports, an alternative method is to use .A or .B appended to the slot number instead of using the number to specify the port. Thus, to address channel A of a 1756-DHRIO module in slot 3, you could use either 3, 2 or 3.A. For channel B, your choices would be 3, 3 or 3.B.

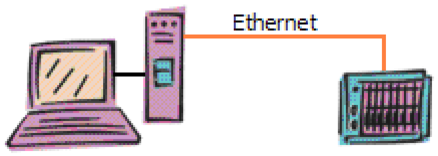
DH-485 ports are always addressed using the .A or .B notation.

Component	Port	Assigned Number
All ControlLogix chassis	Backplane	1
Processor	DF1 Port	2
1756-ENET or 1756-ENBT	Ethernet Port	2
1756-CNB	ControlNet Port	2
1756-DHRIO	DH+ Port (Channel A)	2
1756-DHRIO	DH+ Port (Channel B)	3
1761-NET-ENI	DF1 Port	3
1756-DH485	DH-485 Ports	Use .A or .B notation

Ethernet CLX Examples

These are examples of how to configure CIP paths for an Ethernet CLX device. This type of device is available only on systems that have the DHX OPC Server Suite, DHX OPC Premier Suite or DHX OPC Enterprise Suite installed.

Ethernet to ControlLogix Processor



In this example, the Ethernet CLX device sends a message directly to a ControlLogix chassis that is the final destination.

192.168.1.2, 1, 0

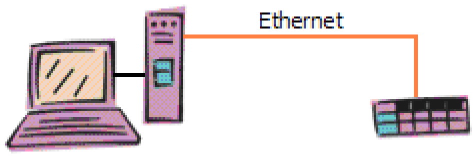
The message goes to an Ethernet module at the specified IP address. It must then go across the backplane to the processor module.

192.168.1.2: Addresses an Ethernet module at this IP address

1: Uses the ControlLogix backplane port

0: Addresses the processor module in slot 0

Ethernet to CompactLogix Processor



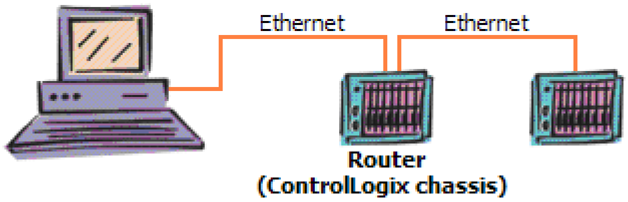
Here, the Ethernet CLX device sends a message directly to its final destination, a CompactLogix controller.

192.168.54.40,1,0

The message goes to a CompactLogix controller at the specified IP address. The CompactLogix hardware does not have a literal slot for the processor, but you must specify a CIP path as though it does, with the processor residing in slot 0.

> *192.168.54.40:* Addresses the CompactLogix controller at this IP address
>
> *1:* Specifies the virtual "backplane"; always 1 for CompactLogix
>
> *0:* Specifies the virtual "slot" for the processor; always 0 for CompactLogix

ControlLogix Chassis Ethernet Router



This example has the Ethernet CLX device send the message to a ControlLogix chassis, which routes it to another Ethernet network, and on to its final destination.

10.8.2.124, 1, 3, 2, 10.9.2.78, 1, 0

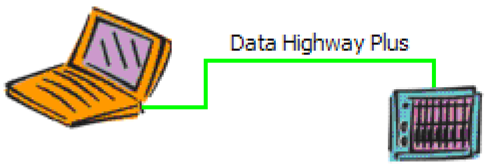
The message goes to an Ethernet module in a ControlLogix chassis, then goes across its backplane to another Ethernet module. From there, the message goes to another ControlLogix chassis, where it crosses the backplane to a processor module.

> *10.8.2.124:* Addresses an Ethernet module at this IP address
>
> *1:* Uses the ControlLogix backplane port
>
> *3:* Addresses the second Ethernet module, which is in slot 3
>
> 2: Uses the module's Ethernet port
>
> *10.9.2.78:* Addresses an Ethernet module at this IP address
>
> *1:* Uses the ControlLogix backplane port
>
> *0:* Addresses the processor module in slot 0

CLX over DHX Examples

These are examples of how to configure CIP paths for a CLX over DHX device. This type of device is available only on systems that have the DHX OPC Server Suite, DHX OPC Premier Suite or DHX OPC Enterprise Suite installed.

Data Highway Plus to ControlLogix Processor – Decimal Address



This is a simple example in which the CLX over DHX device directly addresses a ControlLogix processor on a DH+ network.

31, 1, 2

Here we address a Data Highway Plus module in a chassis, then go across the backplane to the processor module.

> *31:* Addresses a DH+ module at station address 31
>
> *1:* Uses the ControlLogix backplane port
>
> *2:* Addresses the processor module in slot 2

Data Highway Plus to ControlLogix Processor – Octal Address

This is identical to the previous example; only the addressing method is different.

8#37, 1, 2

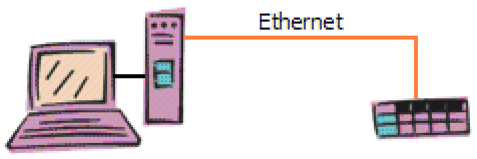
This is functionally the same as the previous example, except here we specify the address in octal.

> *8#37:* Addresses a DH+ module at octal station address 37
>
> *1:* Uses the ControlLogix backplane port
>
> *2:* Addresses the processor module in slot 2

Ethernet DHX/CIP Examples

These are examples of how to configure CIP paths for an Ethernet DHX/CIP device.

Ethernet to MicroLogix 1100, SLC5/05 or PLC-5



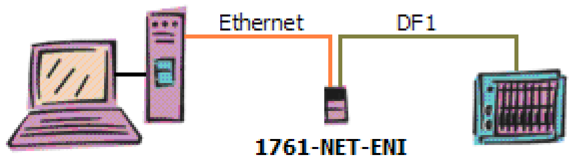
In this example, we are addressing a MicroLogix 1100, SLC5/05 or PLC-5 processor using an Ethernet DHX/CIP device.

192.168.33.117

There is no backplane port or slot to address, so the CIP path is simply the IP address.

Caution! Only the more recent versions of SLC5/05 and PLC-5 controllers can use the Ethernet DHX/CIP Driver. The older versions use the Ethernet DHX Driver.

1761-NET-ENI Module



The Allen-Bradley 1761-NET-ENI module is used to bridge from an Ethernet network to the serial port on a PLC. In this example, an Ethernet DHX/CIP device communicates to the 1761-NET-ENI module, which then passes the information through its serial DF1 port to the serial port on the processor.

The CIP path you must use with this module depends on its firmware revision level.

Firmware Rev. A

192.168.1.77

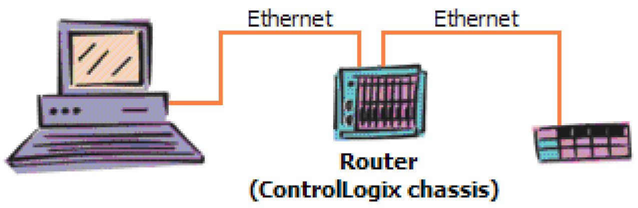
For firmware revision A, simply use the IP address of the module.

Firmware Rev. B-D

192.168.1.77, 3, 1

For firmware revisions B-D, the IP address alone may work, but it may be necessary to use *<IP address>, 3, 1*. In this case, the 3 is the port number for the DF1 port and 1 is the default address for the device connected via the serial link.

ControlLogix Chassis Ethernet Router



For this example, we have created a router by installing two Ethernet modules in a ControlLogix chassis, and will use this to pass messages from an Ethernet DHX/CIP device on one Ethernet network to a logic controller on another Ethernet network.

Caution! When used with the Ethernet DHX/CIP Driver, this type of router can be used only for configurations in which the final destination device is a MicroLogix 1100, SLC5/05 or PLC-5.

10.3.54.101, 1, 5, 2, 192.168.65.6

The message is sent to one of the modules, which passes it along the ControlLogix backplane to the other module, which then sends it along to the final destination.

10.3.54.101: Addresses an Ethernet module at this IP address

1: Uses the ControlLogix backplane port

5: Addresses the Ethernet module in slot 5

2: Specifies the module's Ethernet port

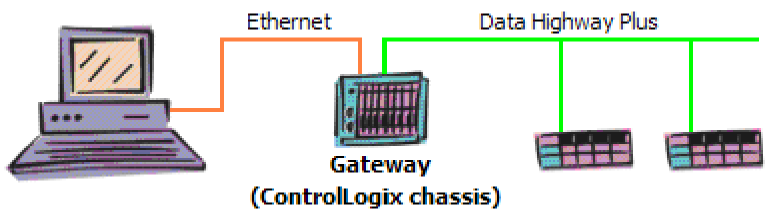
192.168.65.6: The IP address of the final destination of the message

ControlLogix Gateway Driver Examples

The ControlLogix Gateway Driver uses a special syntax in its configuration. It is of the form *address, port, address, port*. Note that this is an incomplete CIP path, because it ends with a port, rather than an address.

When you configure the CIP path for the ControlLogix Gateway Driver, you are really configuring only the beginning of the path. That is, you are specifying the path only up to the port on the 1756-DHRIO or 1756-DH485 module in the gateway chassis, omitting the final address. This address is the destination node for the message, and it will be appended to the configured CIP path by the driver, as it processes the message.

IP Address and Typical Port Syntax



In this example, a ControlLogix Gateway Driver device is configured to send messages to controllers on a Data Highway Plus network by passing them through a 1756-DHRIO module in a ControlLogix chassis.

192.168.9.2, 1, 4.B

The message is addressed to the Ethernet module in the chassis used as the gateway, then it goes across the backplane to the Data Highway Plus module. Notice that the path does not specify the DH+ node address of the destination device.

192.168.9.2: Addresses an Ethernet module at this IP address

1: Uses the ControlLogix backplane port

4.B: Addresses the 1756-DHRIO module in slot 4, and uses its DH+ port B

(The destination DH+ node address is not specified, but will be appended by the driver at runtime.)

IP Address and Alternative Port Syntax

This is the same as the previous example, but here we use the alternative method of specifying the DH+ channel.

192.168.9.2, 1, 4, 3

The message routing is identical to the previous example, because the 4, 3 form is exactly the same as the 4.B form.

192.168.9.2: Addresses an Ethernet module at this IP address

1: Uses the ControlLogix backplane port

4: Addresses the 1756-DHRIO module in slot 4

3: Uses DH+ port B of the 1756-DHRIO module

(The destination DH+ node address is not specified, but will be appended by the driver at runtime.)

DNS Name

In this example, the message goes to an Ethernet module with the DNS name AssemblyLineA, and is routed through the gateway chassis in the same manner as in the previous example.

AssemblyLineA, 1, 2.A

The routing in this case is handled the same way as the previous examples, but this time using the DNS name. Notice also that in this example we use port A of the DH+ module, which is in slot 2.

AssemblyLineA: Addresses an Ethernet module with this DNS name

1: Uses the ControlLogix backplane port

2.A: Addresses the 1756-DHRIO module in slot 2, and uses its DH+ port A

(The destination DH+ node address is not specified, but will be appended by the driver at runtime.)

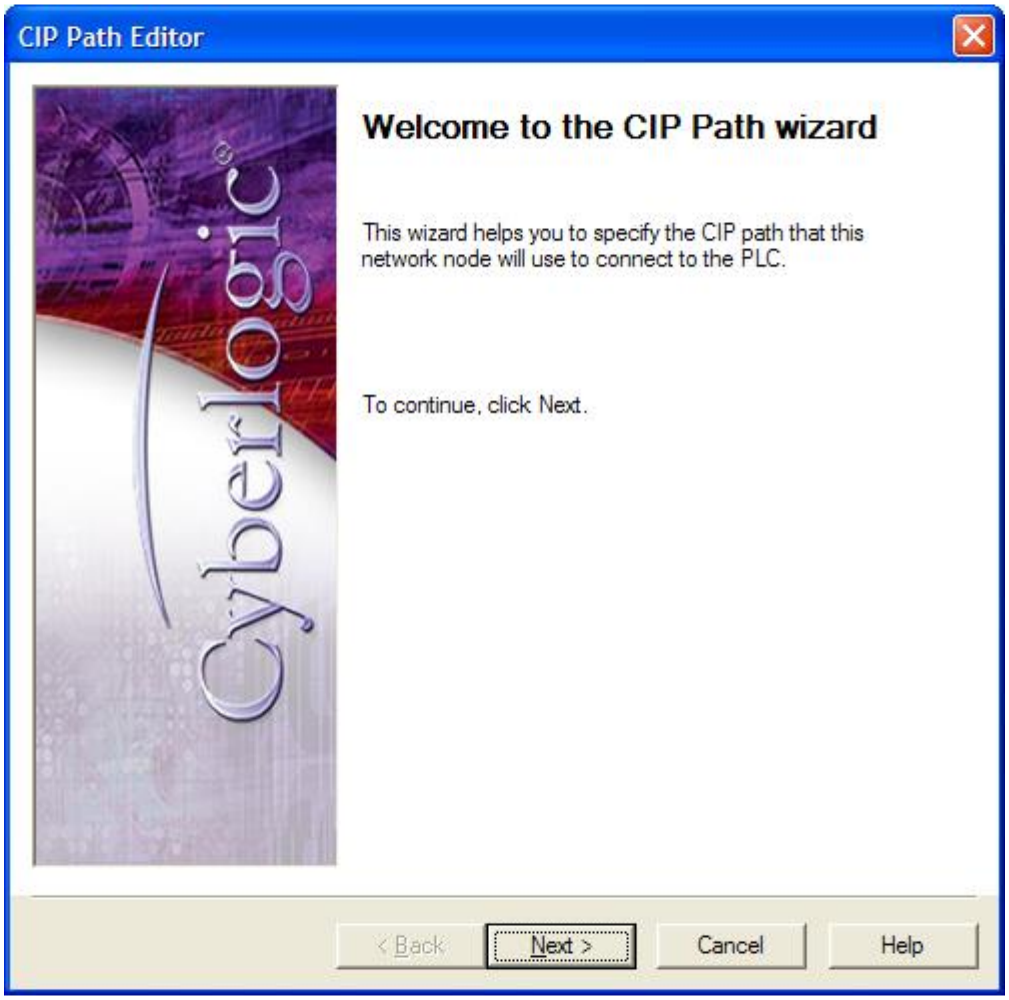
CIP Path Wizard

The CIP Path Wizard will guide you in specifying the Control and Information Protocol (CIP) path that the network node will use to connect to the PLC. The screens you will see will vary depending upon the CIP path you are building. We will show three examples.

- Example 1: Simple Ethernet covers a typical Ethernet application.
- Example 2: Simple Data Highway Plus covers Data Highway Plus and is similar to the DF1 configuration process.
- Example 3: Complex Addressing involves both Ethernet and Data Highway Plus.

Example 1: Simple Ethernet

In this example, we want to set up communications over Ethernet to a ControlLogix chassis at IP address 10.207.55.221. The controller is in slot 9 of that chassis.

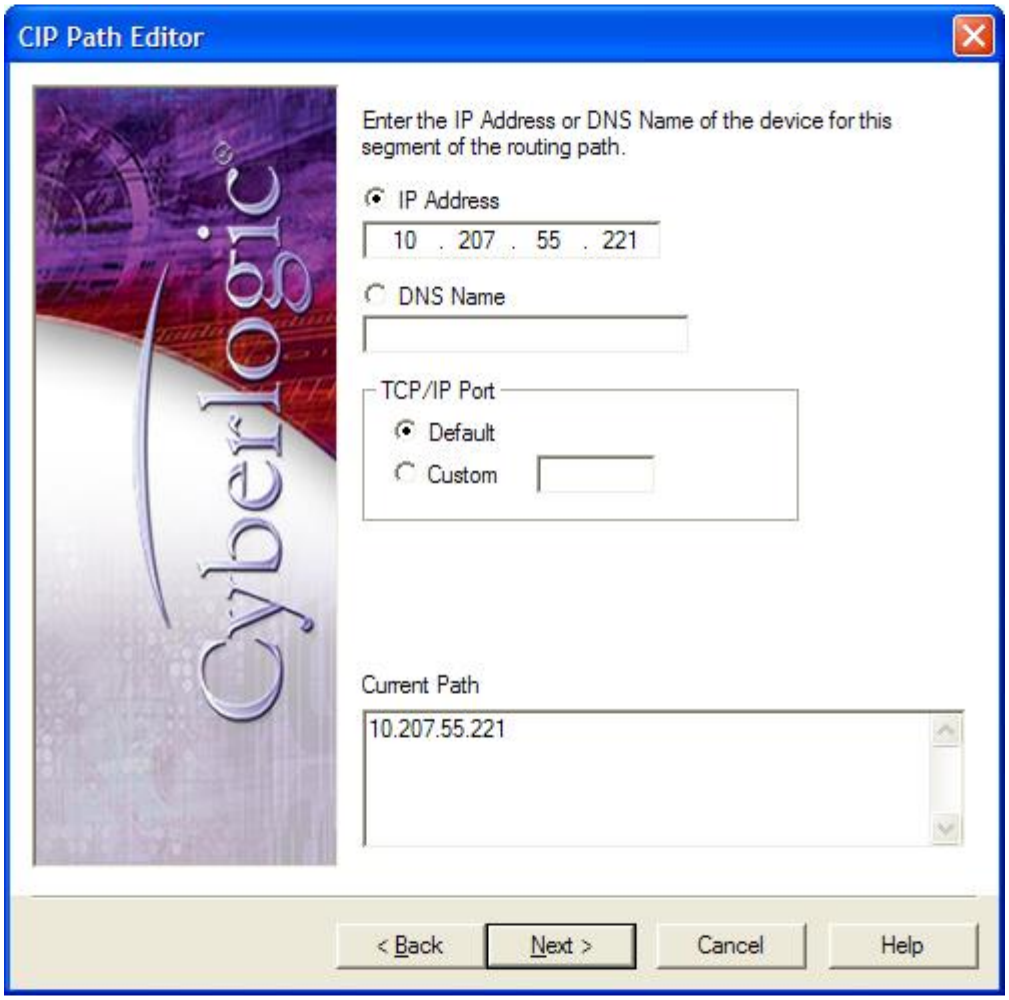


Welcome to the CIP Path wizard

This wizard helps you to specify the CIP path that this network node will use to connect to the PLC.

To continue, click Next.

1. From the Welcome screen, click ***Next*** to continue.



Enter the IP Address or DNS Name of the device for this segment of the routing path.

[✓ IP Address] 10 . 207 . 55 . 221

DNS Name

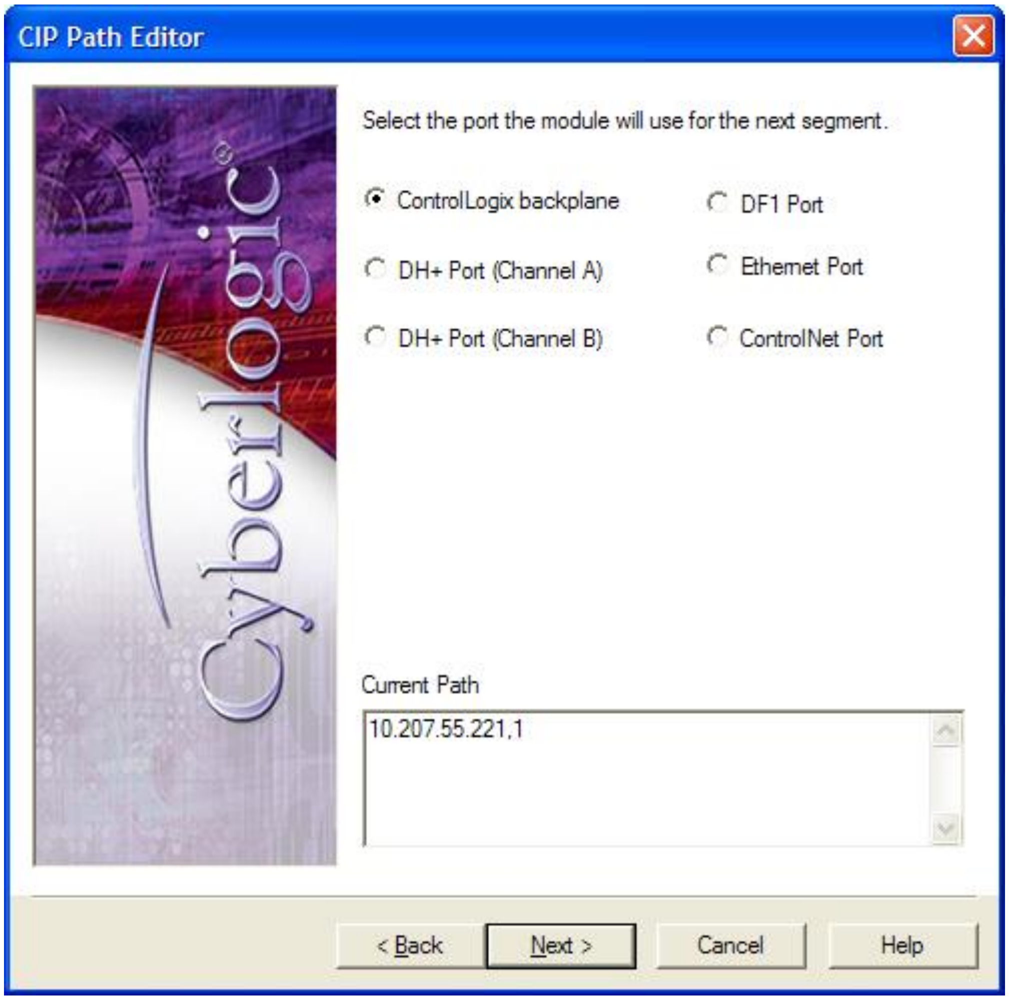
TCP/IP Port: [✓ Default] Custom

Current Path: 10.207.55.221

2. Enter the IP address or DNS name of the node. For this example, select ***IP Address*** and enter ***10.207.55.221***.

3. Optionally, you may enter an IP port number if you do not wish to use the default of 44818. For this example, leave this field blank and select ***Default***.

4. Click ***Next*** to continue.



Select the port the module will use for the next segment.

[✓ ControlLogix backplane] DF1 Port

DH+ Port (Channel A) Ethernet Port

DH+ Port (Channel B) ControlNet Port

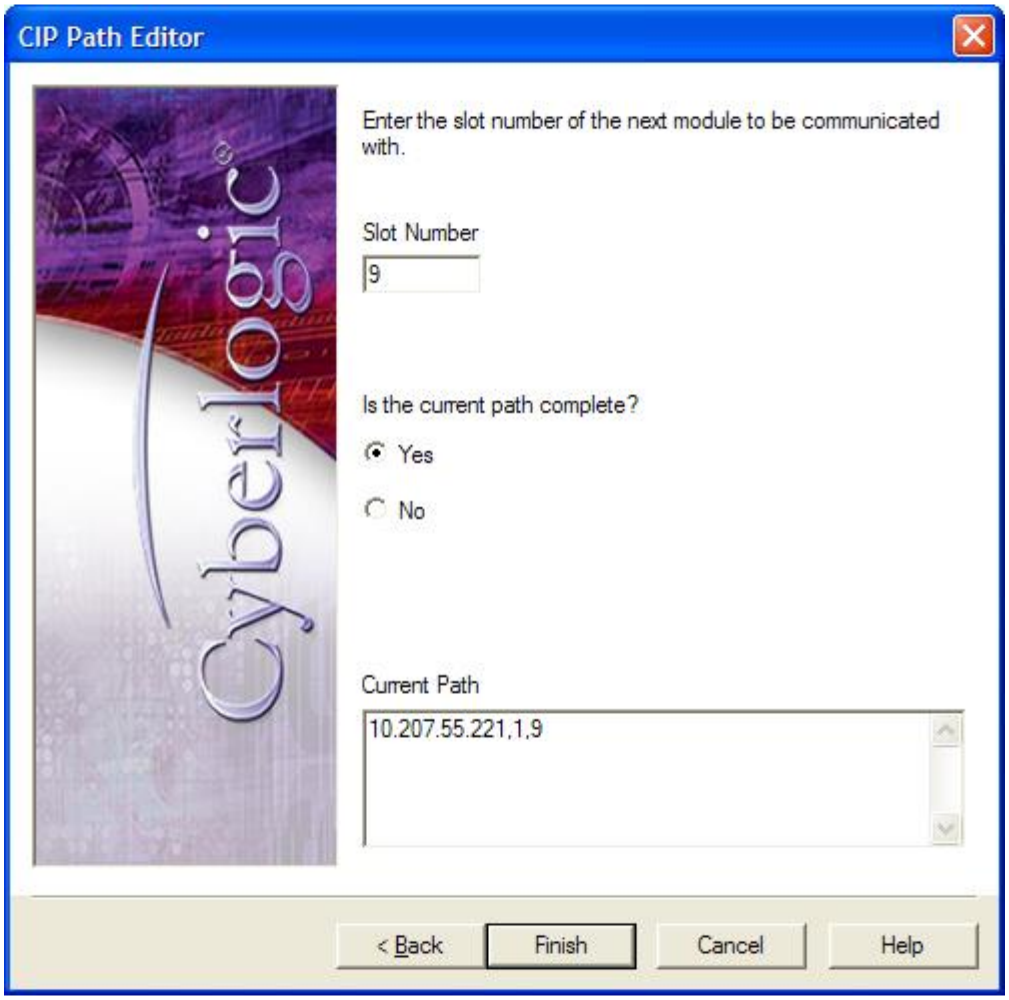
Current Path

10.207.55.221,1

You have specified the Ethernet interface in the target system. Now you must communicate over that system's backplane.

5. Select ***ControlLogix backplane***.

6. Click ***Next*** to continue.



Enter the slot number of the next module to be communicated with.

Slot Number: 9

Is the current path complete?

(● Yes) ○ No

Current Path: 10.207.55.221,1,9

Finally, you must specify the slot number for the controller module.

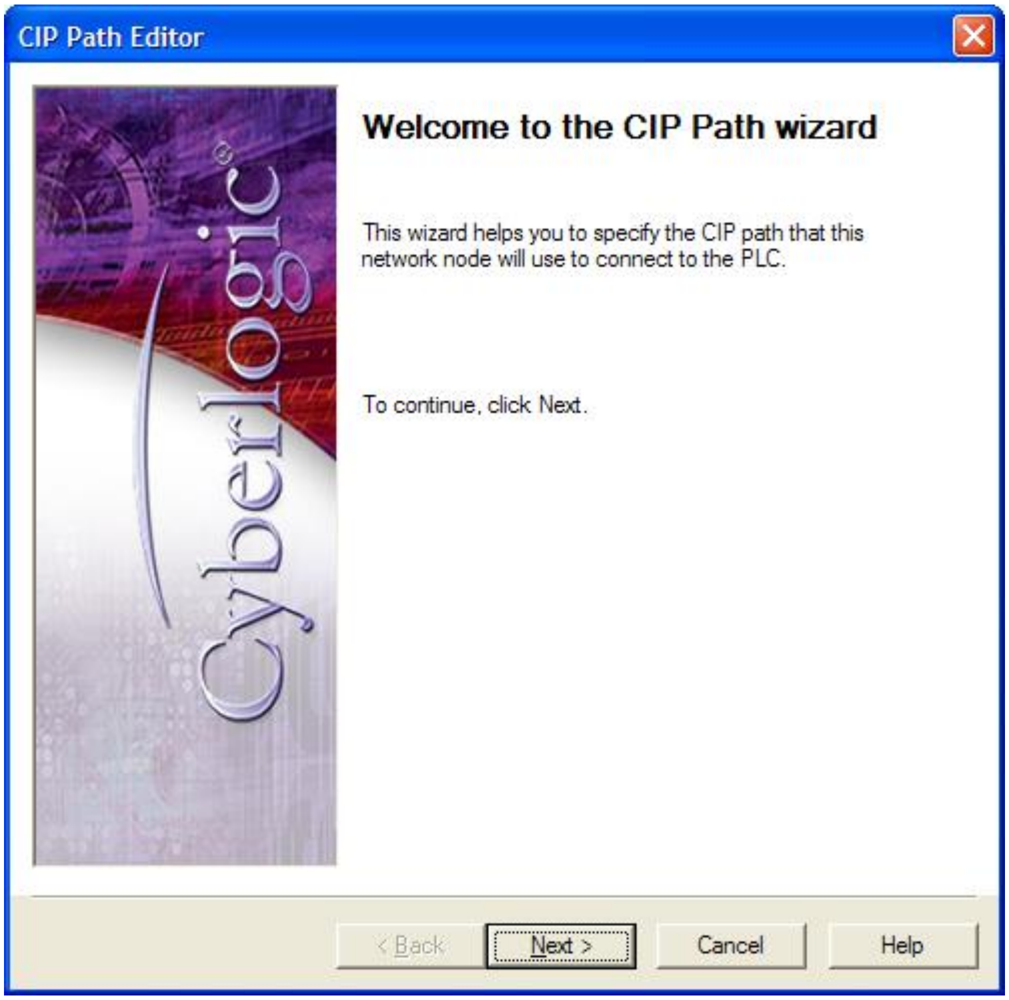
7. Enter ***9*** in the ***Slot Number*** field.

8. Select ***Yes*** to indicate that you have specified the entire path.

9. Click ***Finish*** to complete the wizard operation and exit.

Example 2: Simple Data Highway Plus

In this example, we want to set up communications over Data Highway Plus to a ControlLogix chassis at node 15 of the Data Highway Plus network. The controller is in slot 9 of that chassis.

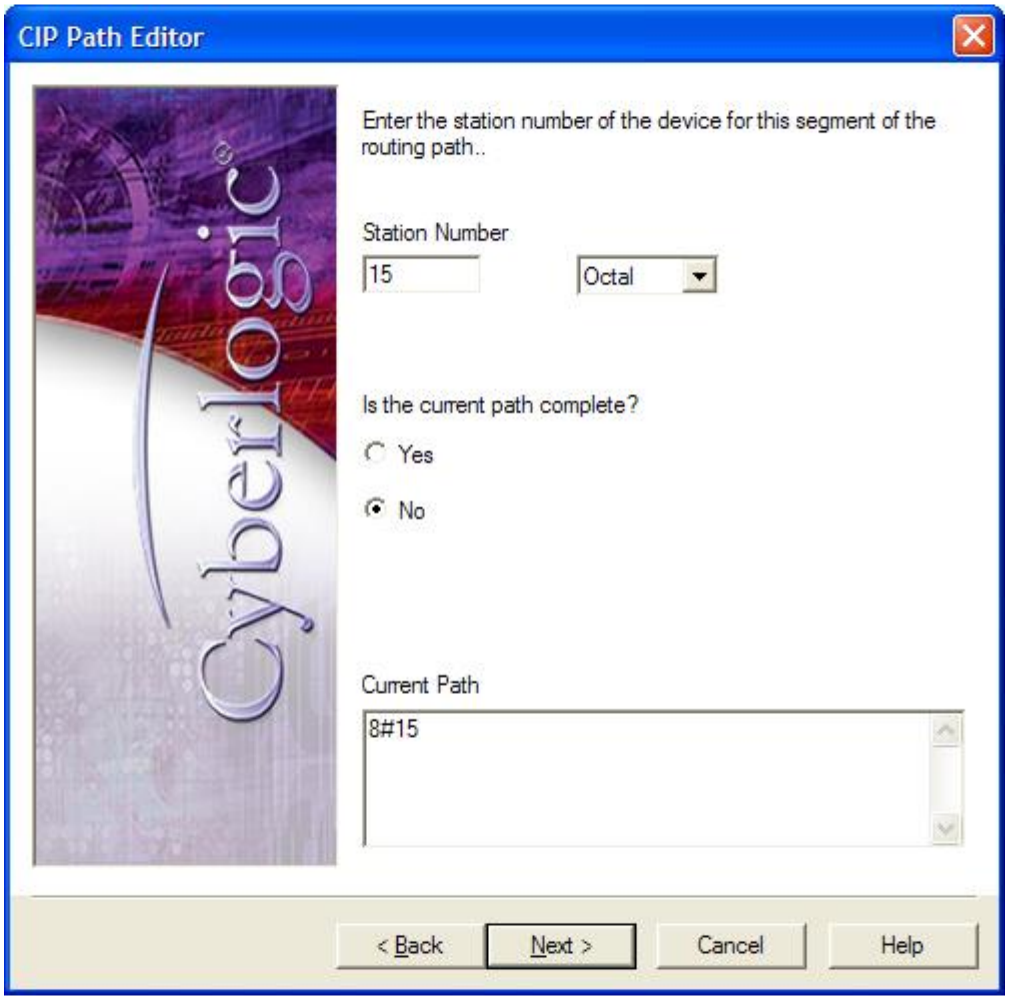


Welcome to the CIP Path wizard

This wizard helps you to specify the CIP path that this network node will use to connect to the PLC.

To continue, click Next.

1. From the Welcome screen, click ***Next*** to continue.



Enter the station number of the device for this segment of the routing path..

Station Number: 15 — Octal

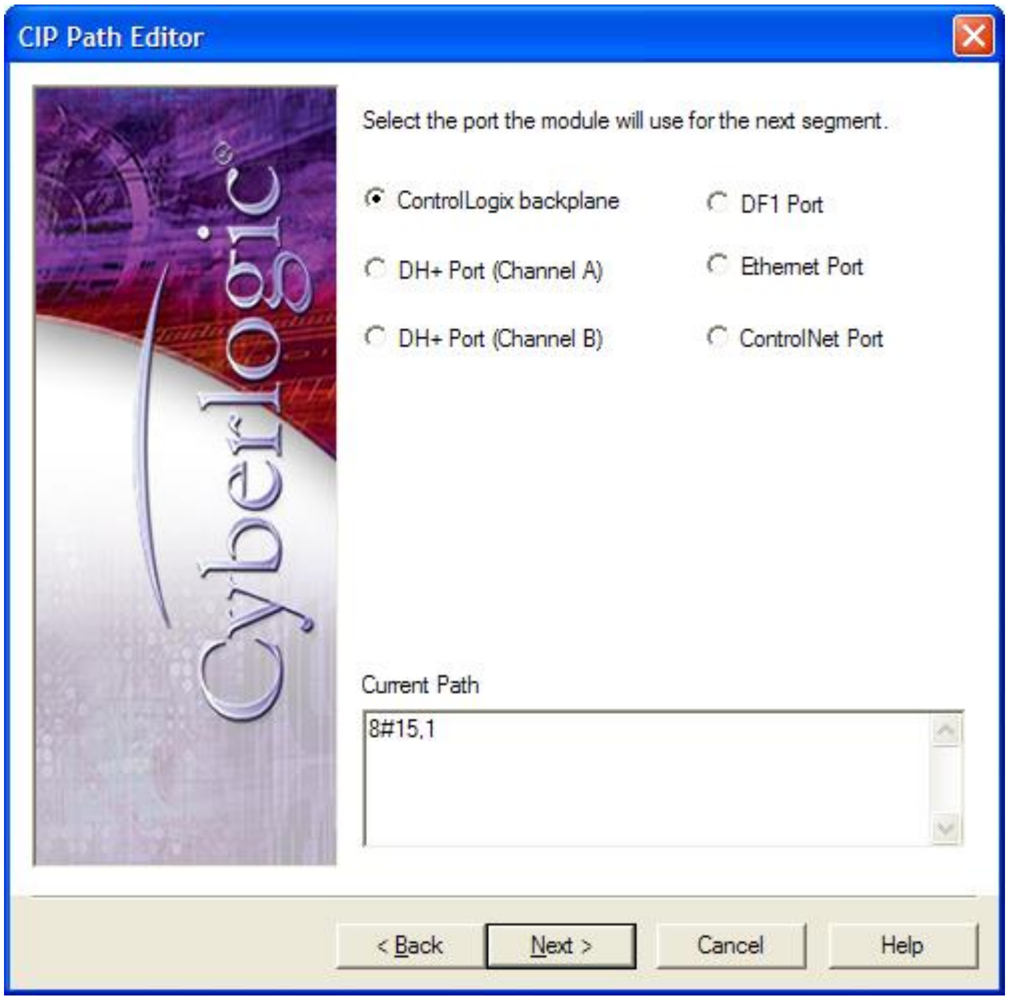
Is the current path complete?

Yes [✓ No]

Current Path: 8#15

2. Enter the octal Data Highway Plus node number of the target node. For this example, enter ***15*** in the ***Station Address*** field.

3. Click ***Next*** to continue.



Select the port the module will use for the next segment.

[✓ ControlLogix backplane] DF1 Port
DH+ Port (Channel A) Ethernet Port
DH+ Port (Channel B) ControlNet Port

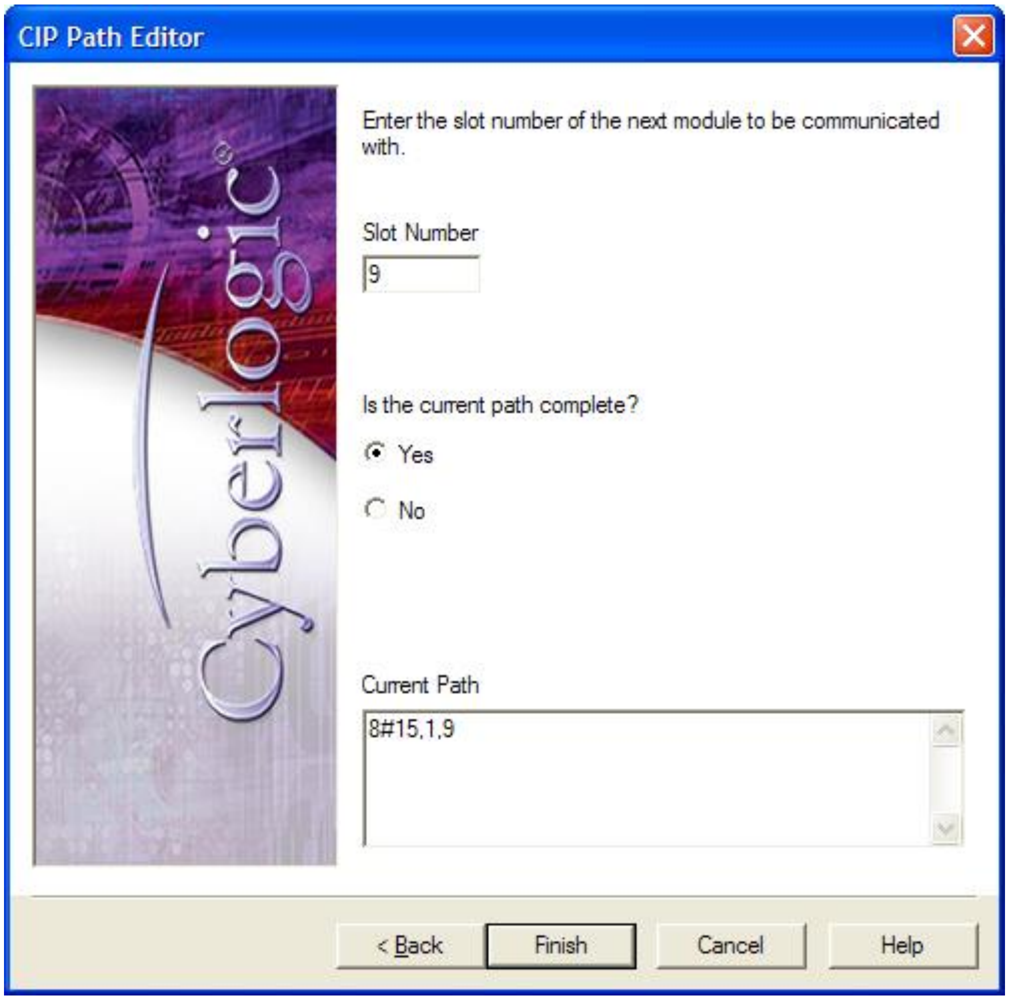
Current Path

8#15,1

You have specified the Data Highway Plus interface in the target system. Now you must communicate over that system's backplane.

4. Select ***ControlLogix backplane***.

5. Click ***Next*** to continue.



Enter the slot number of the next module to be communicated with.

Slot Number
9

Is the current path complete?
(●) Yes
( ) No

Current Path
8#15,1,9

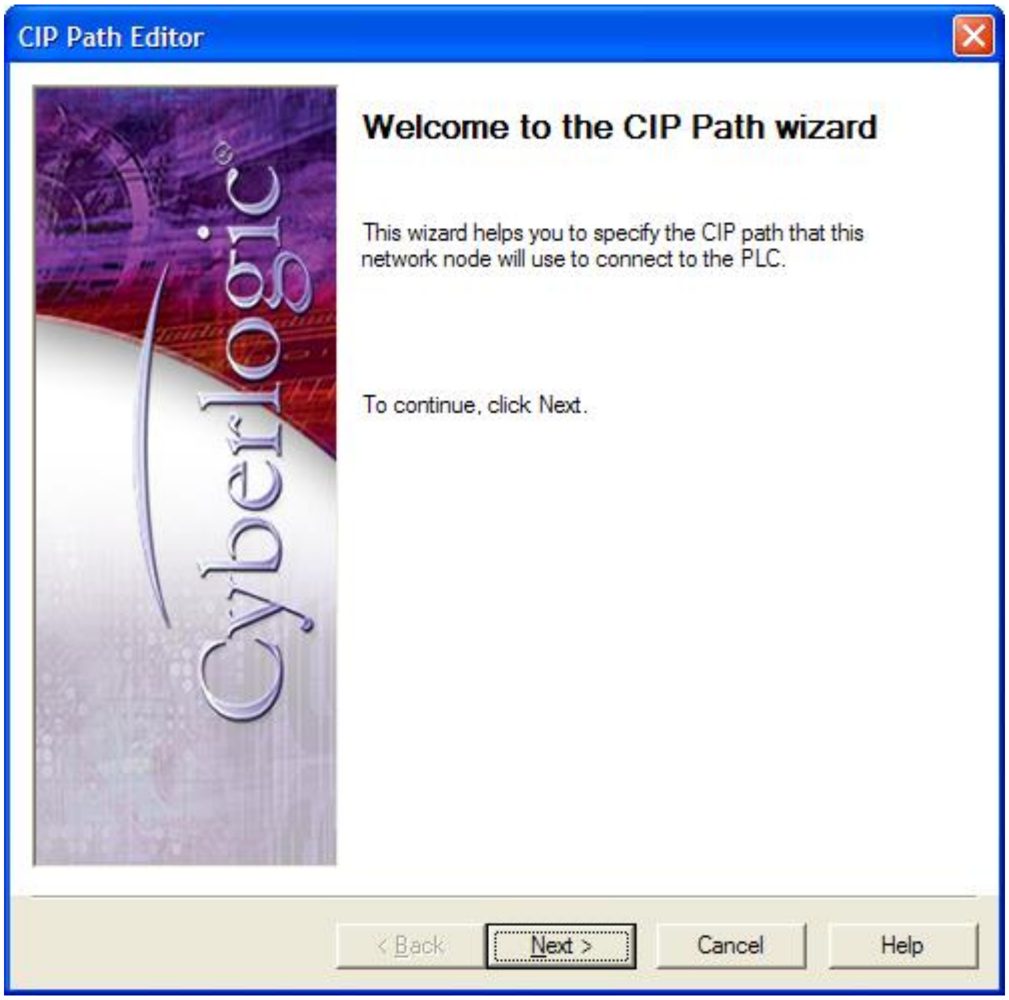
6. Enter ***9*** in the ***Slot Number*** field to specify the slot number for the controller module.

7. Select ***Yes*** to indicate that you have specified the entire path.

8. Click ***Finish*** to complete the wizard operation and exit.

Example 3: Complex Addressing

In this example, we want to set up communications over Ethernet to a ControlLogix chassis at IP address 10.207.55.221. That chassis has a Data Highway Plus module in slot 4. We will use channel A of that module to communicate to another chassis at Data Highway Plus node 15. The controller is in slot 9 of that chassis.

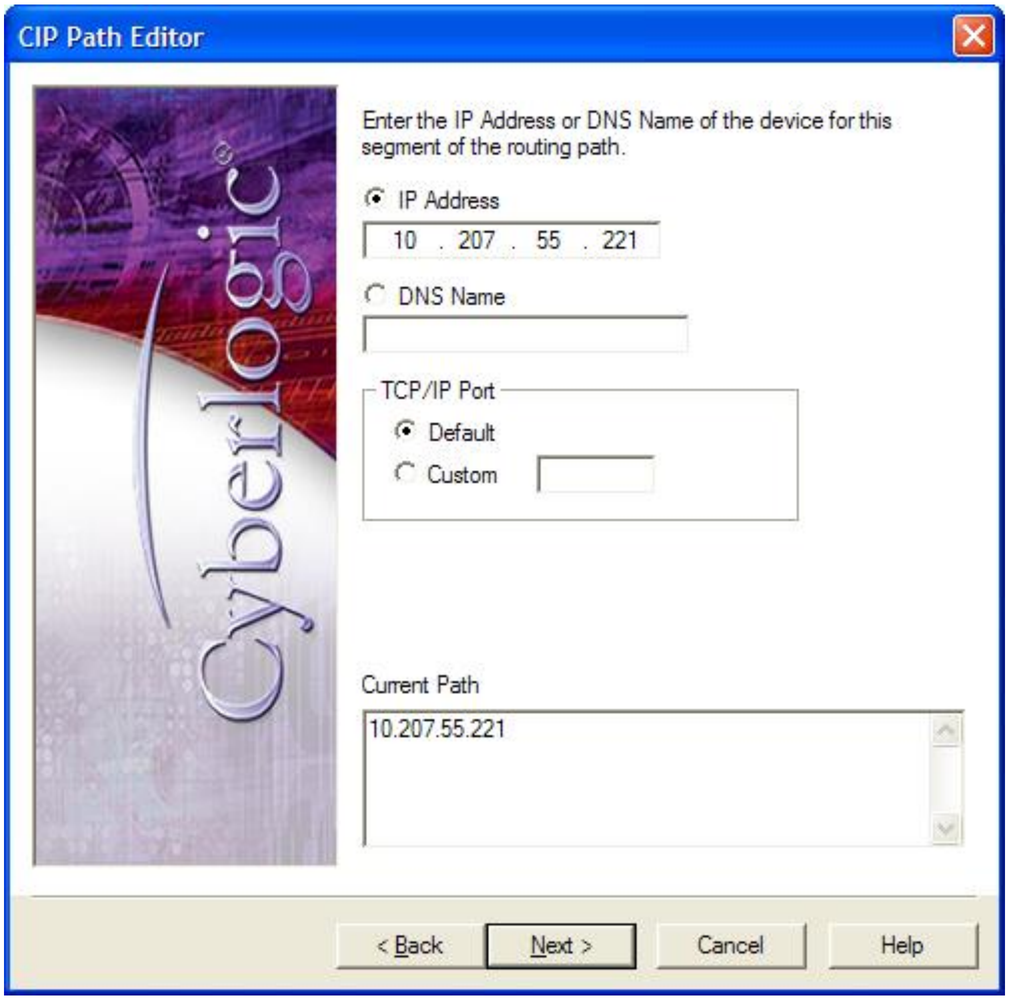


Welcome to the CIP Path wizard

This wizard helps you to specify the CIP path that this network node will use to connect to the PLC.

To continue, click Next.

1. From the Welcome screen, click ***Next*** to continue.



Enter the IP Address or DNS Name of the device for this segment of the routing path.

[✓ IP Address] 10 . 207 . 55 . 221

DNS Name

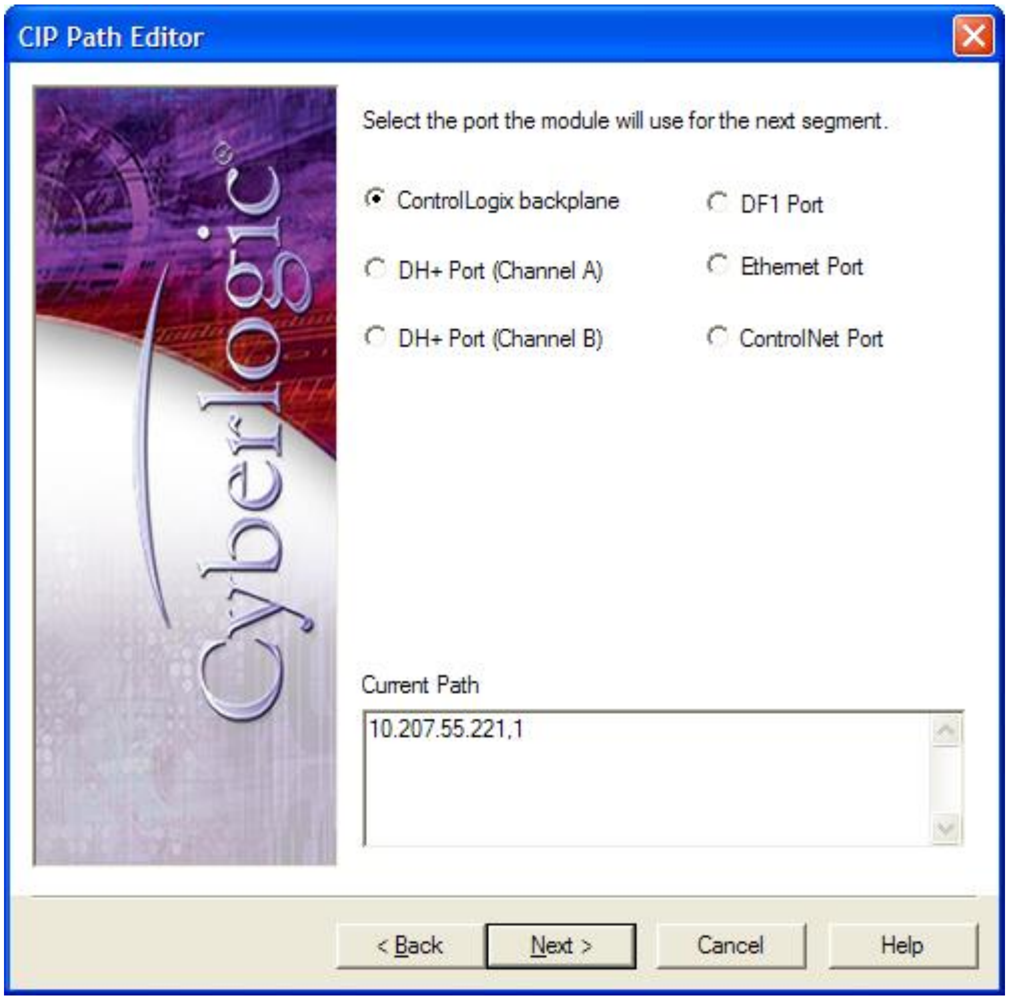
TCP/IP Port: [✓ Default] Custom

Current Path: 10.207.55.221

2. Enter the IP address or DNS name of the node. For this example, select ***IP Address*** and enter ***10.207.55.221***.

3. Optionally, you may enter an IP port number if you do not wish to use the default of 44818. For this example, leave this field blank and select ***Default***.

4. Click ***Next*** to continue.



Select the port the module will use for the next segment.

[✓ ControlLogix backplane] DH+ Port (Channel A) DH+ Port (Channel B) DF1 Port Ethernet Port ControlNet Port

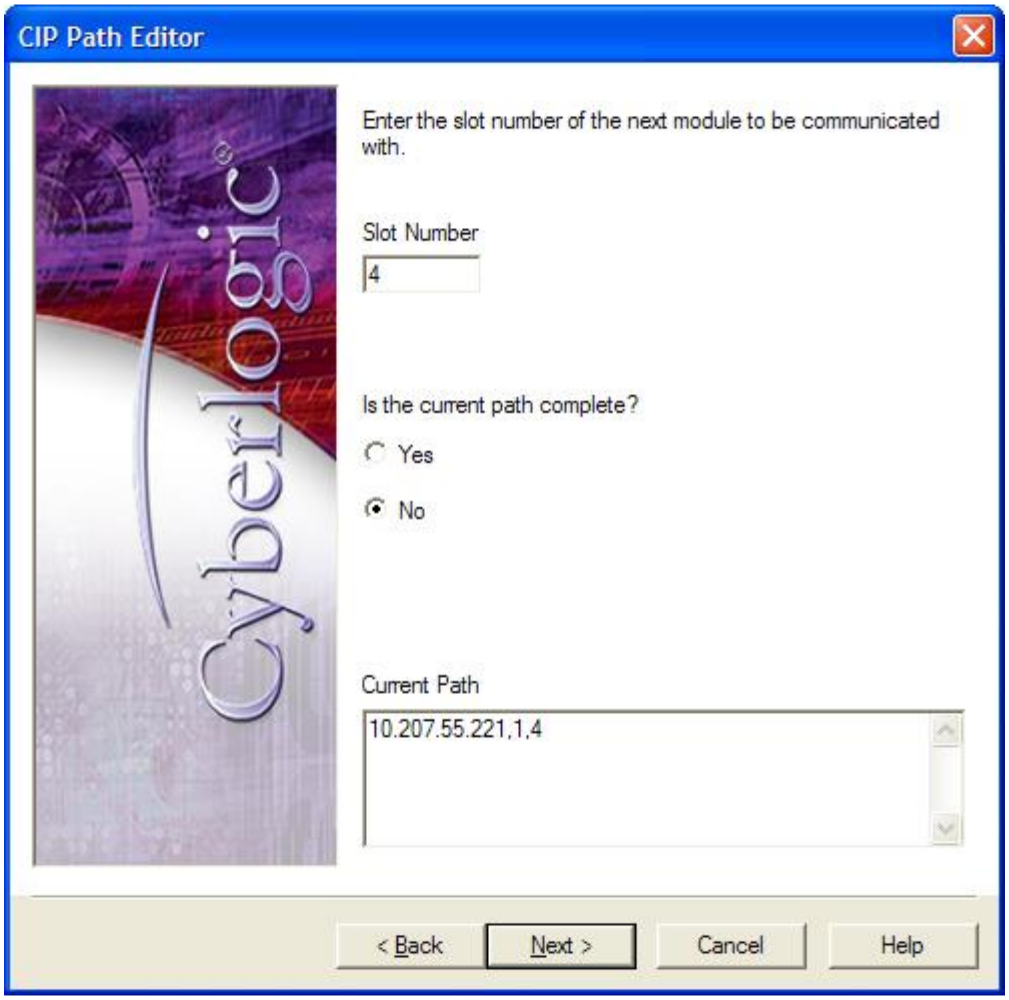
Current Path

10.207.55.221,1

You have specified the Ethernet interface in the first chassis. To reach its Data Highway Plus module, you must communicate over the backplane.

5. Select ***ControlLogix backplane***.

6. Click ***Next*** to continue.



Enter the slot number of the next module to be communicated with.

Slot Number

4

Is the current path complete?

Yes [✓ No]

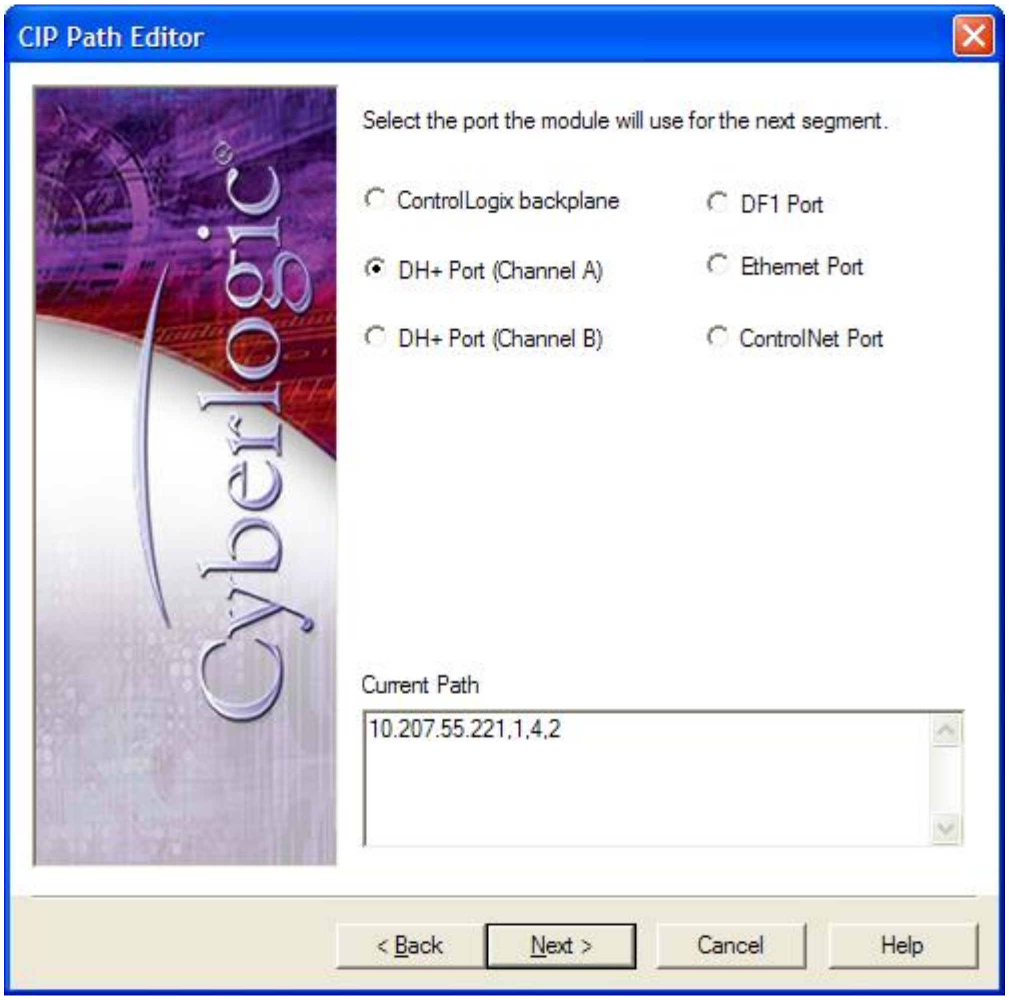
Current Path

10.207.55.221,1,4

7. Enter ***4*** in the ***Slot Number*** field to specify the location of the Data Highway Plus module.

8. You are not finished with the addressing path, so select ***No***.

9. Click ***Next*** to continue.



Select the port the module will use for the next segment.

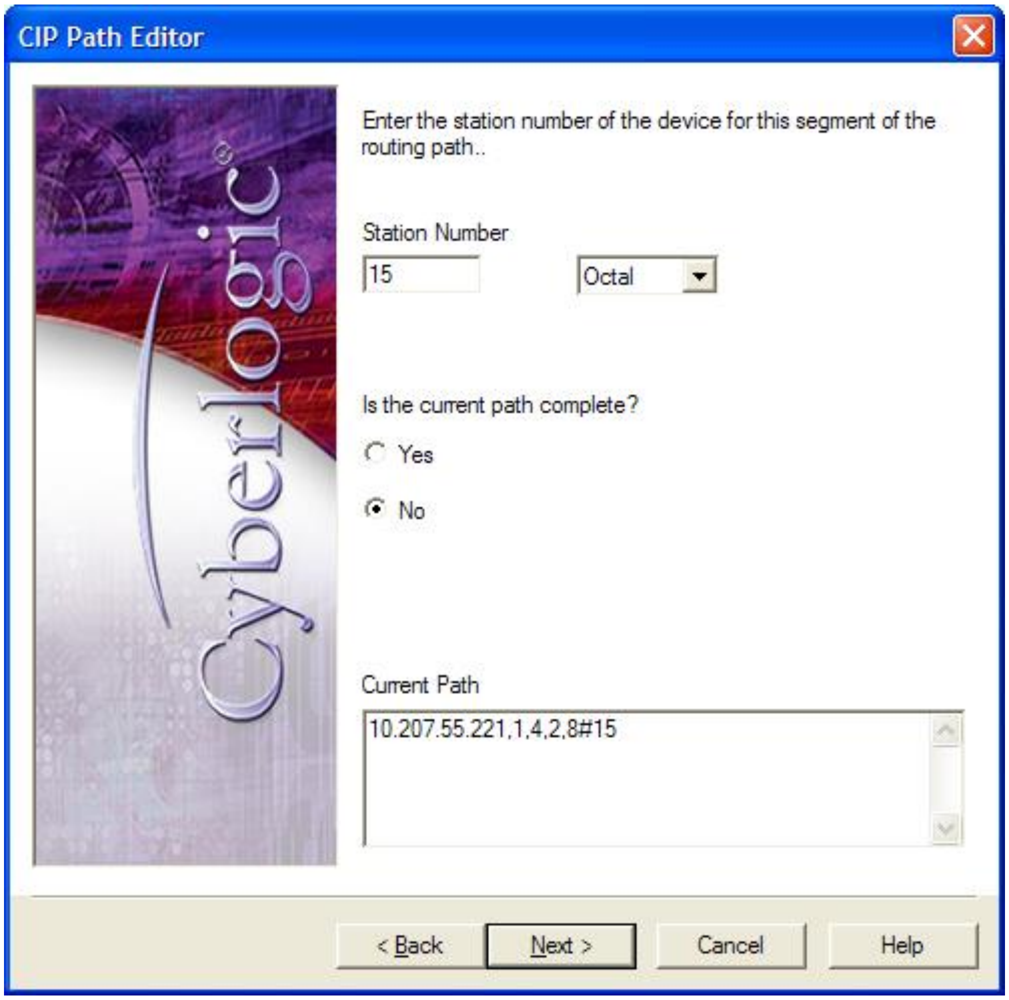
( ) ControlLogix backplane
( ) DF1 Port
[✓ DH+ Port (Channel A)]
( ) Ethernet Port
( ) DH+ Port (Channel B)
( ) ControlNet Port

Current Path

10.207.55.221,1,4,2

10. Select ***Data Highway Plus port (Channel A)***.

11. Click ***Next*** to continue.



Enter the station number of the device for this segment of the routing path..

Station Number
15
Octal

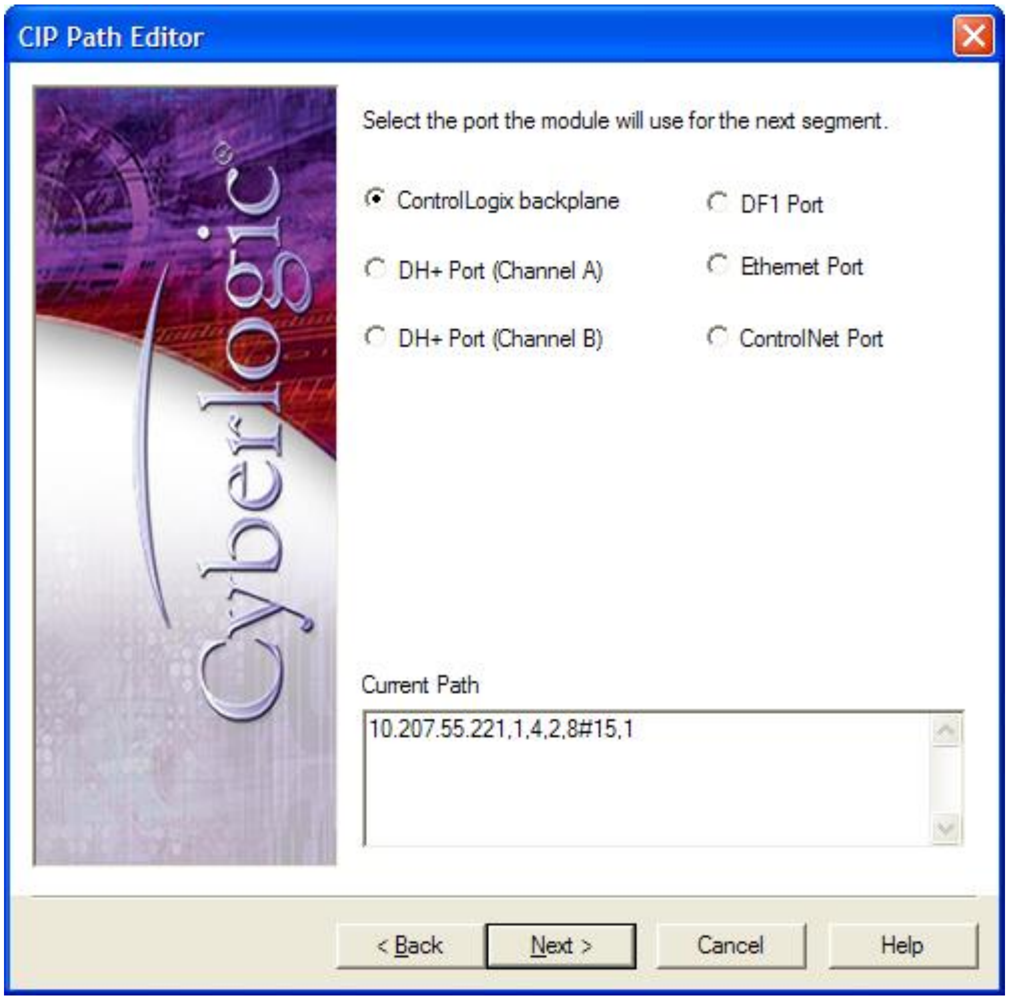
Is the current path complete?
Yes [✓ No]

Current Path
10.207.55.221,1,4,2,8#15

12. Enter ***15*** in the ***Station Address*** field. This is the octal DH+ node number of the target node.

13. You have not yet reached the controller in the target chassis, so select ***No*** to continue specifying the path.

14. Click ***Next*** to go to the next screen.



Select the port the module will use for the next segment.

[✓ ControlLogix backplane] DF1 Port

DH+ Port (Channel A) Ethernet Port

DH+ Port (Channel B) ControlNet Port

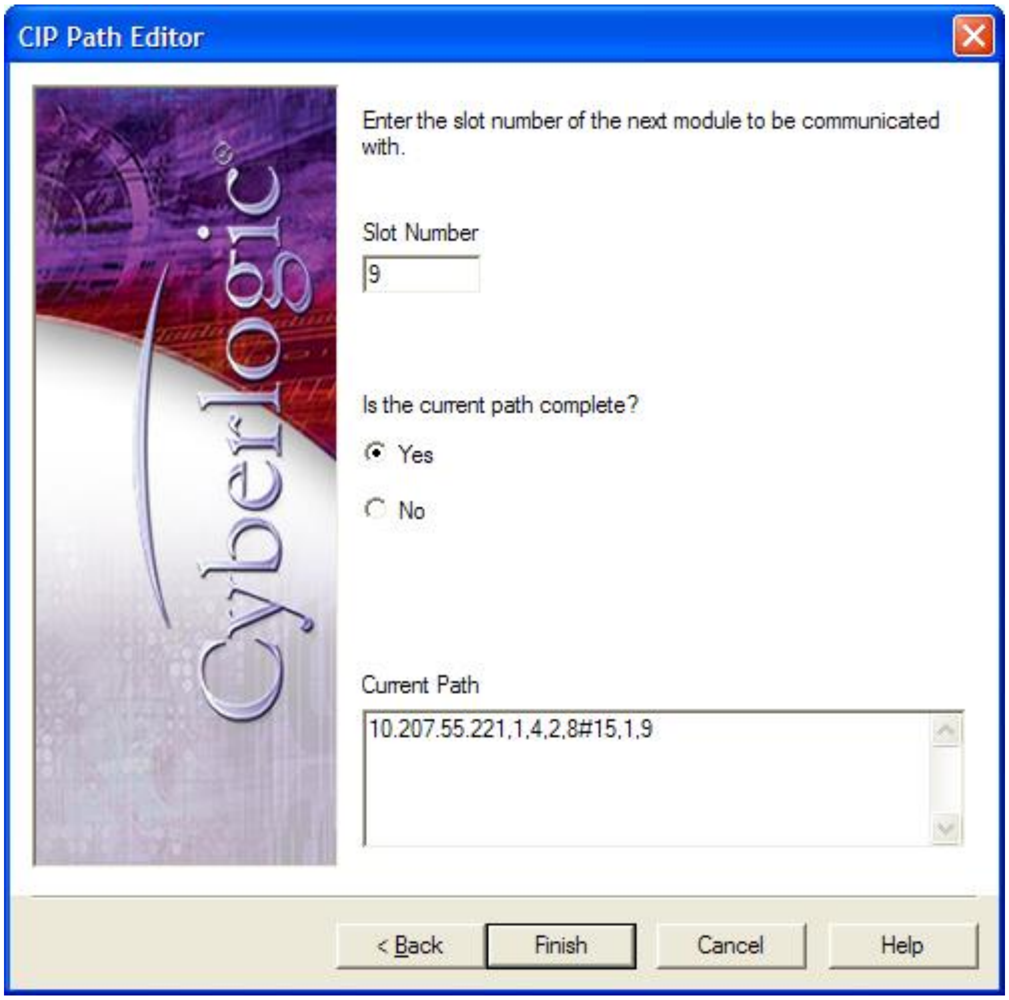
Current Path

10.207.55.221,1,4,2,8#15,1

You have specified the DH+ interface in the target system. Now you must communicate over that system's backplane.

15. Select ***ControlLogix backplane***.

16. Click ***Next*** to continue.



Enter the slot number of the next module to be communicated with.

Slot Number: 9

Is the current path complete?

[✓ Yes] No

Current Path: 10.207.55.221,1,4,2,8#15,1,9

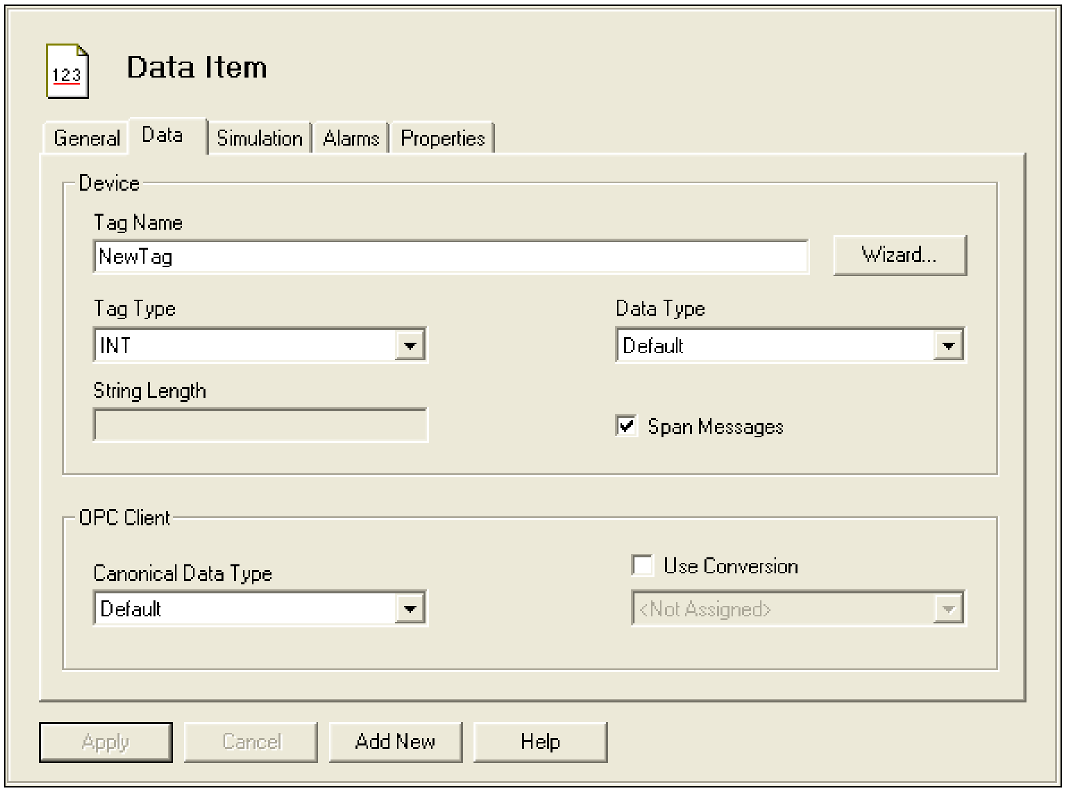
17. Enter ***9*** in the ***Slot Number*** field to specify the location of the controller module.

18. You have now specified the entire path, so select ***Yes*** to indicate that you are done.

19. Click ***Finish*** to complete the wizard operation and exit.

APPENDIX D: 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.



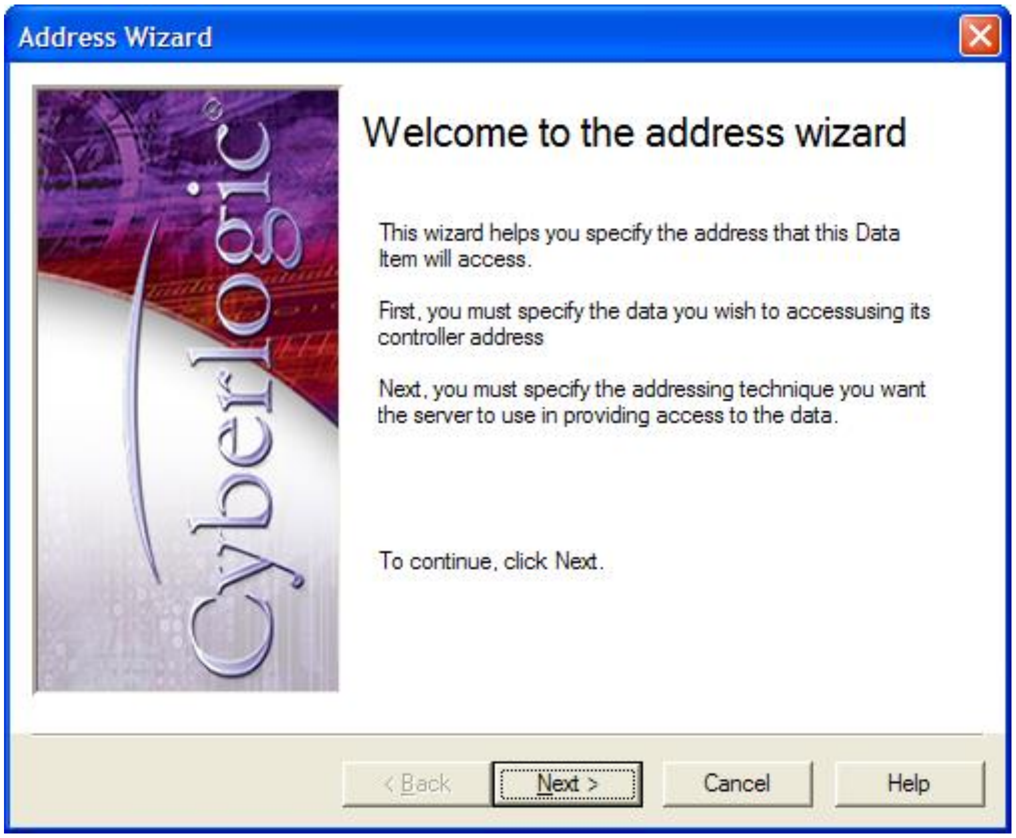
Tag Name
NewTag
Tag Type
INT
Data Type
Default
String Length
☑ Span Messages
Canonical Data Type
Default
☐ Use Conversion
<Not Assigned>

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

- Example 1: Simple Addressing covers setting up the addressing for a single register.
- Example 2: Accessing Bits Within a Register shows how to address bits within a register as a bit array.

Example 1: Simple Addressing

This example shows how to set up the addressing for a single register. You will first specify the controller address where the data is located, then you will specify the addressing method the OPC server will use to provide access to the data.



Welcome to the address wizard

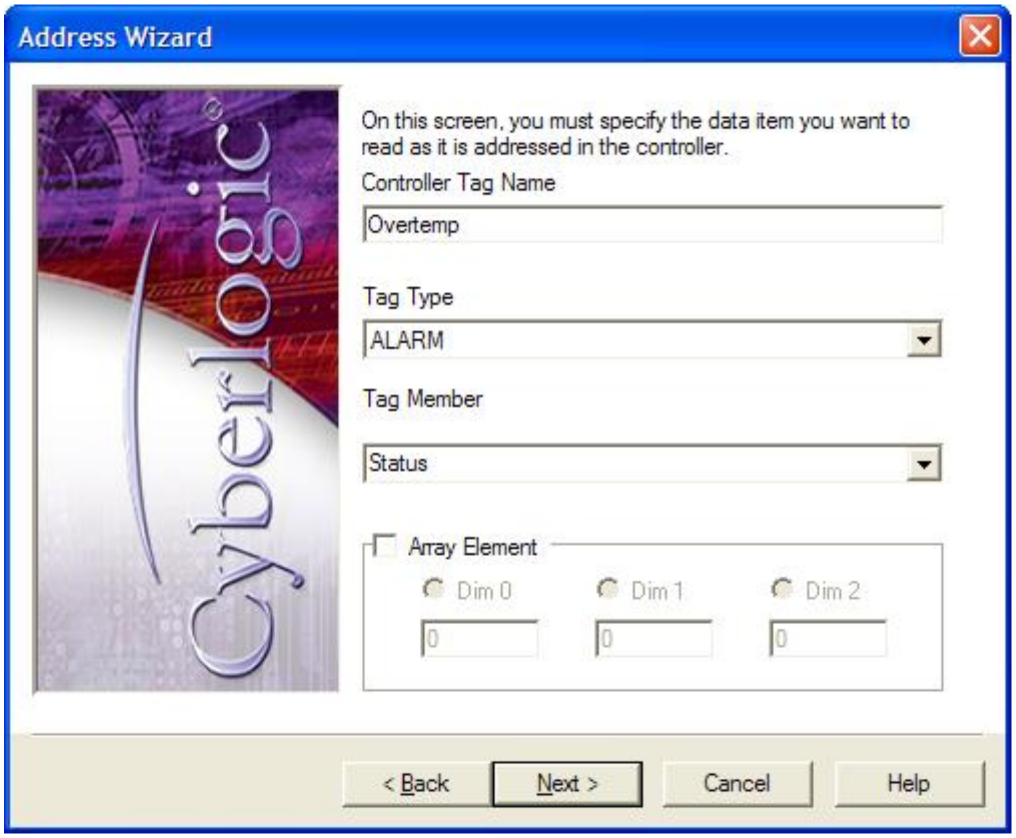
This wizard helps you specify the address that this Data Item will access.

First, you must specify the data you wish to accessusing its controller address

Next, you must specify the addressing technique you want the server to use in providing access to the data.

To continue, click Next.

1. From the Welcome screen, click ***Next*** to continue.



On this screen, you must specify the data item you want to read as it is addressed in the controller.

Controller Tag Name

Overtemp

Tag Type

ALARM

Tag Member

Status

Array Element

Dim 0 Dim 1 Dim 2

0 0 0

On this screen, you must specify the data you want to access.

2. Enter ***Overtemp*** in the ***Controller Tag Name*** field.

   This is the tag name used by the controller.

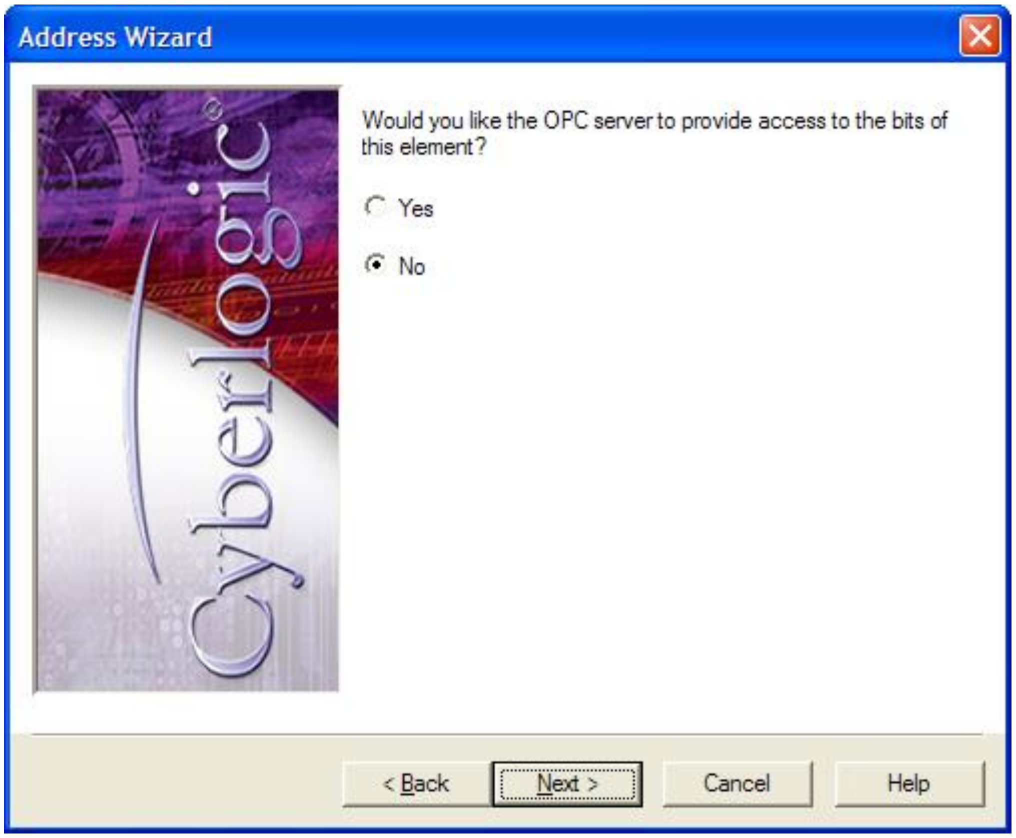
3. Select ***Alarm*** for the ***Tag Type***.

4. Select ***Status*** for the ***Tag Member***.

5. Leave ***Array Element*** unchecked.

   If the data were an element of an array, you would need to check this box and specify the element of the array here.

6. Click ***Next*** to continue.



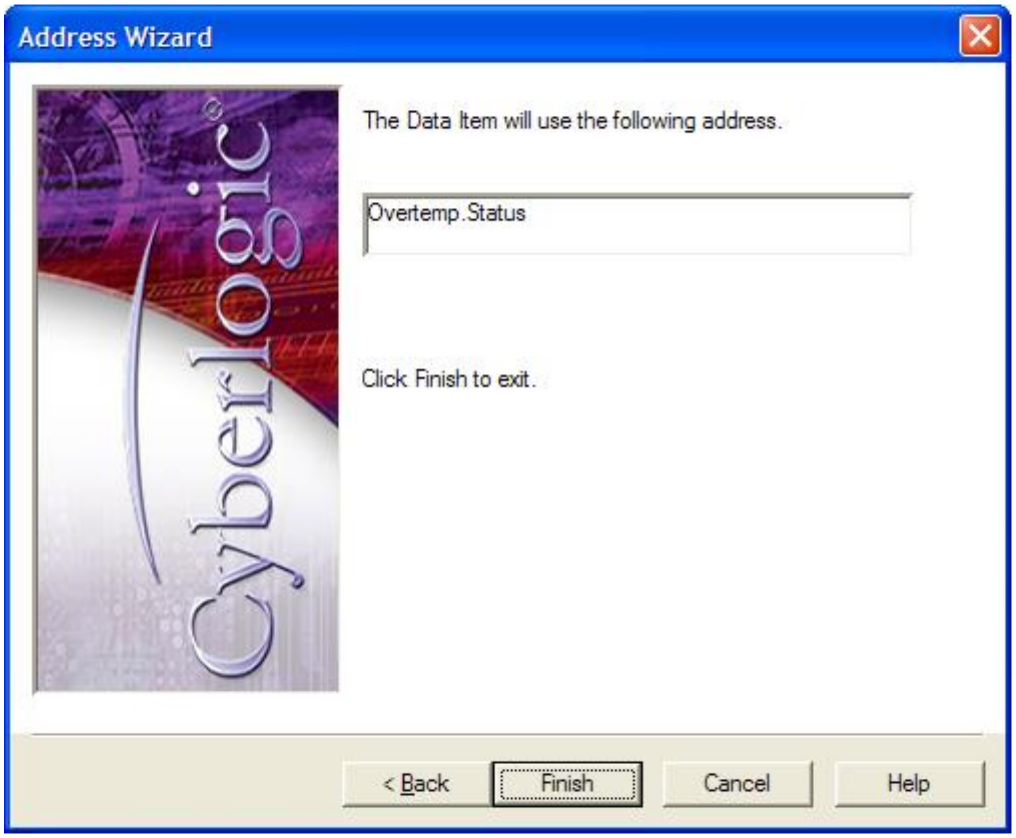
Would you like the OPC server to provide access to the bits of this element?

Yes [✓ No]

It is possible to access individual bits or groups of bits within the register, but for this example we will access the register as a word.

7. Select ***No***.

8. Click ***Next***.


Address Wizard

The Data Item will use the following address.

Overtemp.Status

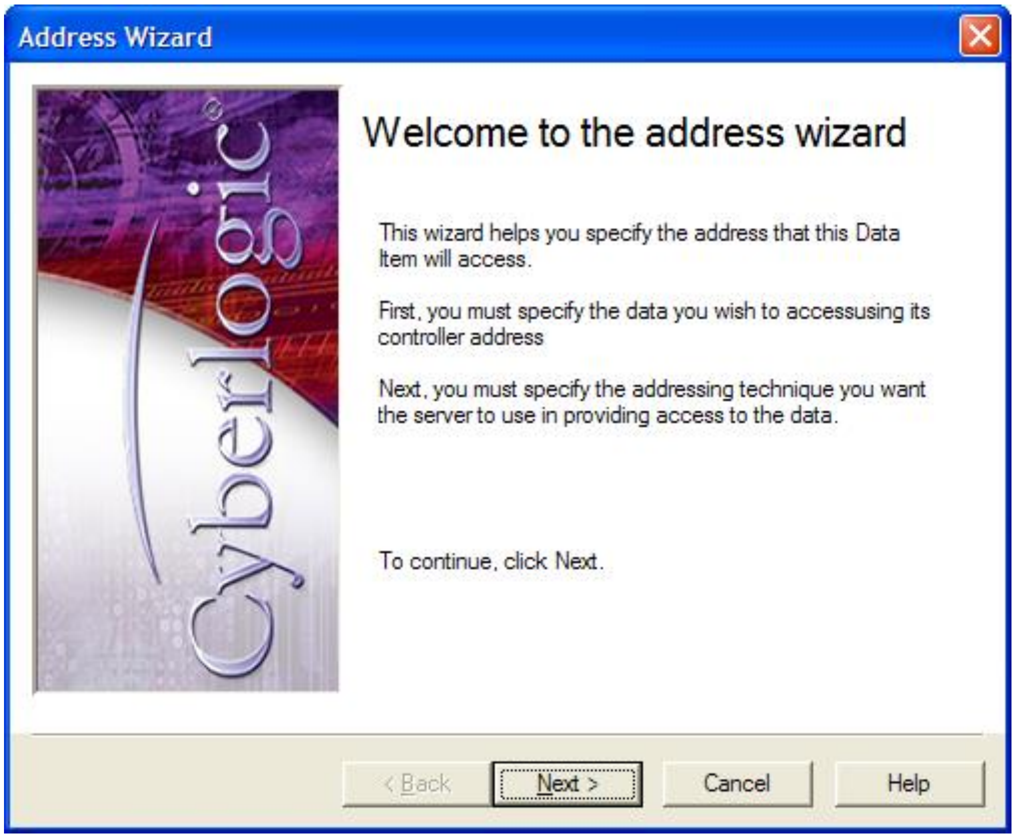
Click Finish to exit.

This screen shows the result of your selections.

9. Click ***Finish*** to allow the wizard to complete and exit.

Example 2: Accessing Bits Within a Register

This example shows how to set up the addressing for a bit field or group of bits within a register.



Welcome to the address wizard

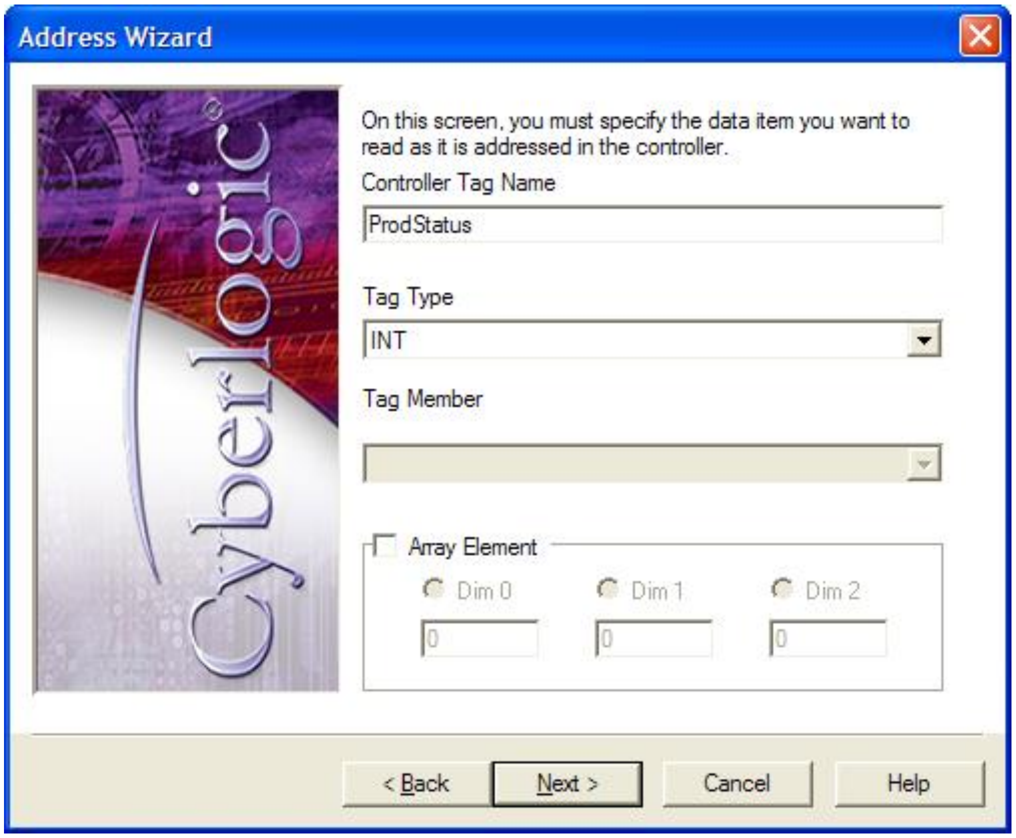
This wizard helps you specify the address that this Data Item will access.

First, you must specify the data you wish to accessusing its controller address

Next, you must specify the addressing technique you want the server to use in providing access to the data.

To continue, click Next.

1. From the Welcome screen, click ***Next*** to continue.



On this screen, you must specify the data item you want to read as it is addressed in the controller.

Controller Tag Name: ProdStatus

Tag Type: INT

Tag Member:

☐ Array Element

Dim 0: 0 | Dim 1: 0 | Dim 2: 0

On this screen, you must specify the data you want to access.

10. Enter ***ProdStatus*** in the ***Controller Tag Name*** field.

    This is the tag name used by the controller.

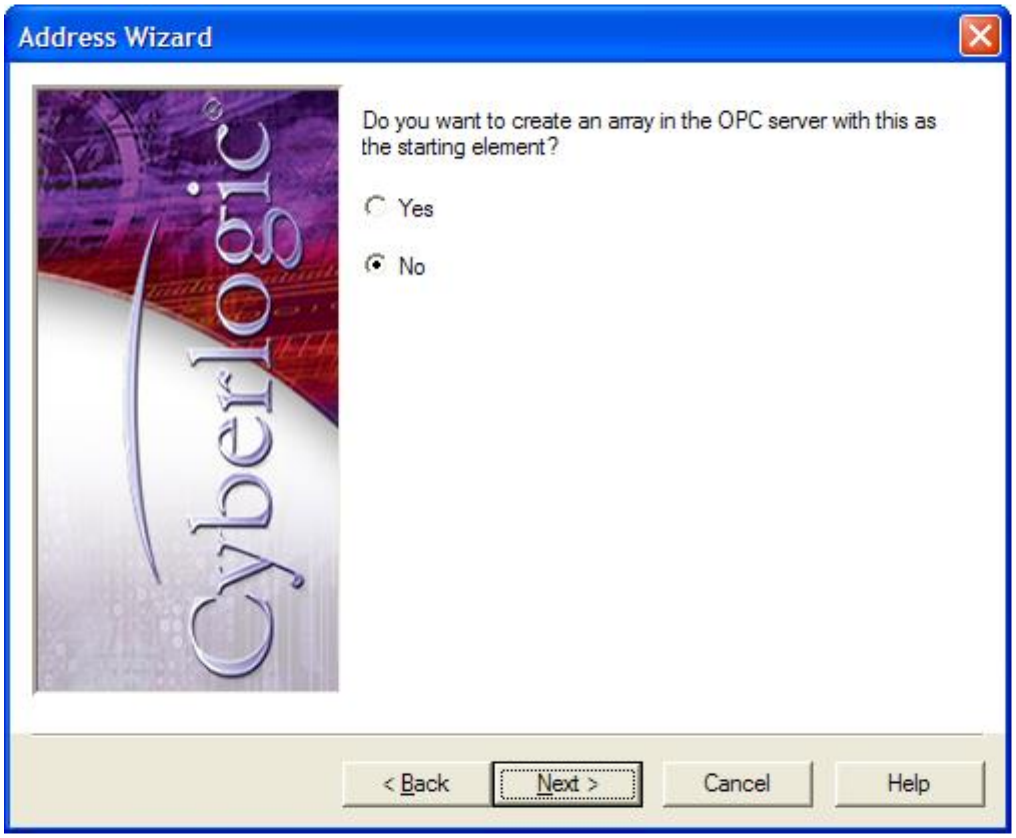
11. Select ***INT*** for the ***Tag Type***.

12. The ***Status*** selection is not available for this tag type.

13. Leave ***Array Element*** unchecked.

    If the data were an element of an array, you would need to check this box and specify the element of the array here.

14. Click ***Next*** to continue.



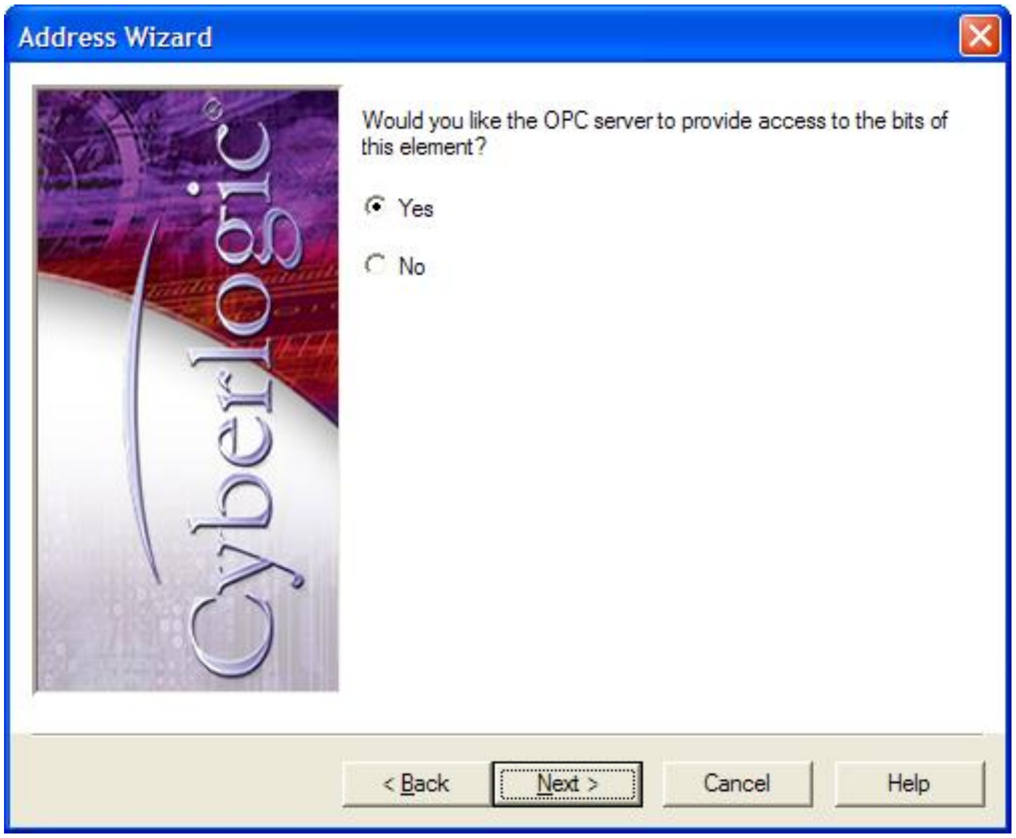
Do you want to create an array in the OPC server with this as the starting element?

Yes [✓ No]

Now you will begin defining the addressing scheme to be used by the OPC Server. You can create an array within the OPC server, but we do not want to do this.

15. Select ***No***.

16. Click ***Next***.

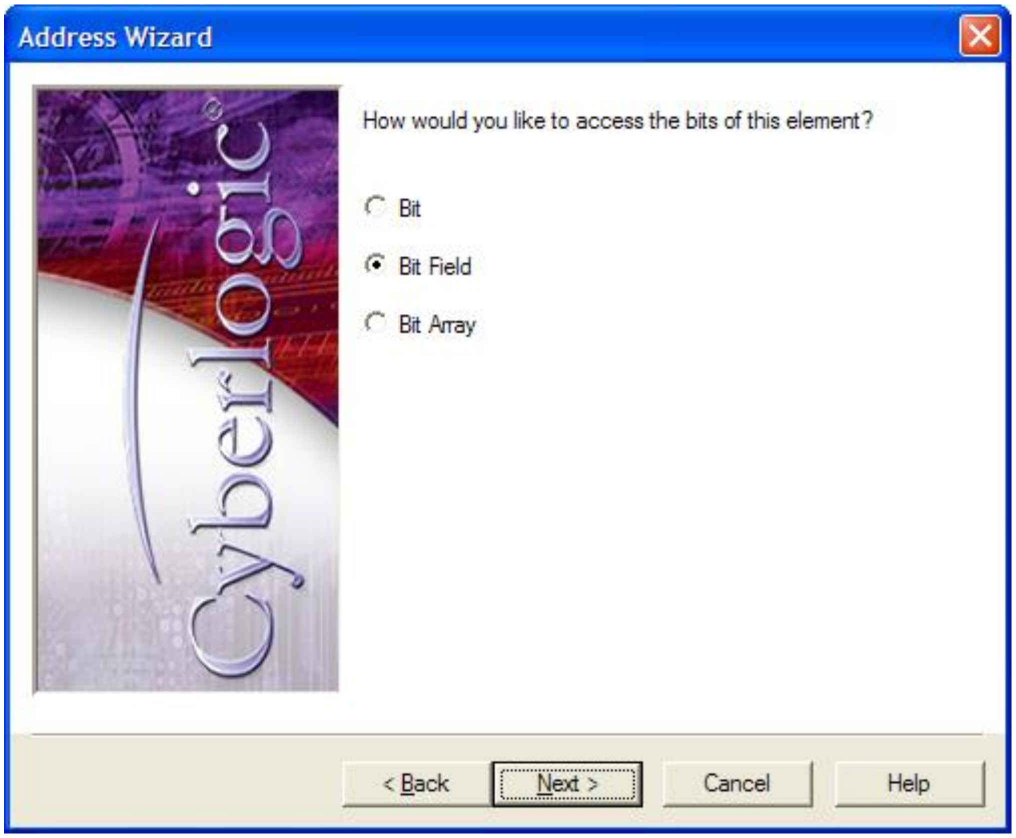

Address Wizard

Would you like the OPC server to provide access to the bits of this element?

[✓ Yes] No

17. Select ***Yes*** to indicate that we want to access only some of the bits within the element.

18. Click ***Next***.



How would you like to access the bits of this element?

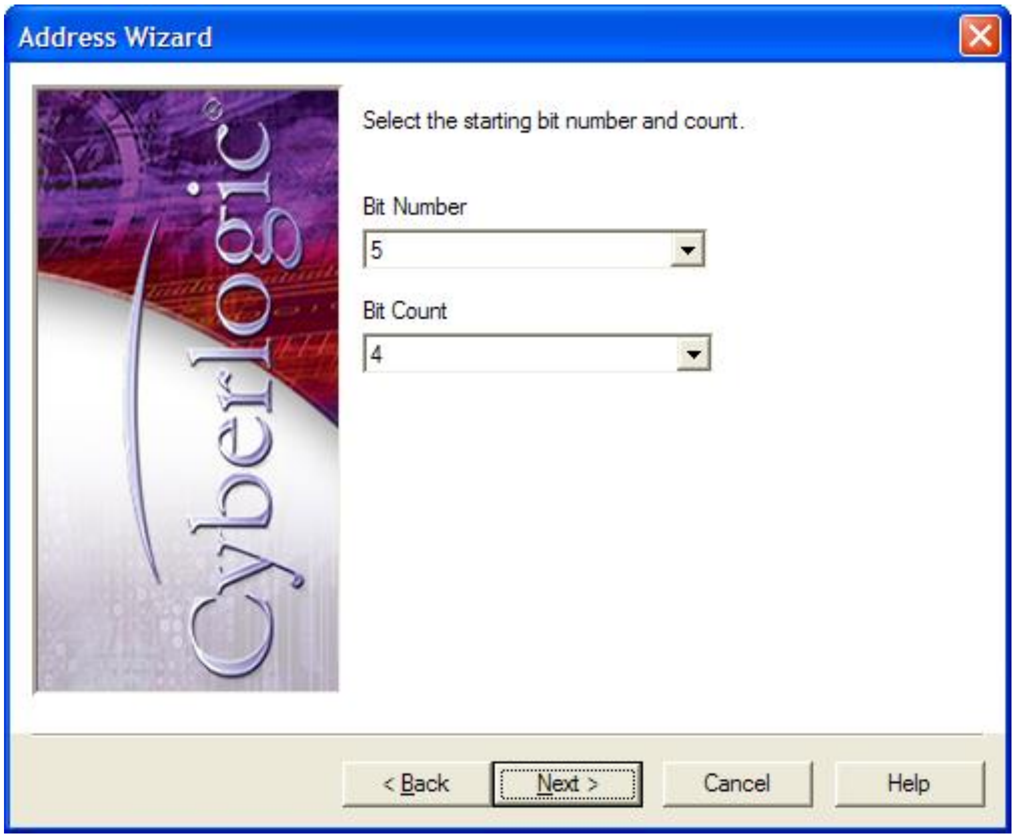
( ) Bit

(•) Bit Field

( ) Bit Array

19. Select ***Bit Field***.

20. Click ***Next*** to continue.



Select the starting bit number and count.

Bit Number: 5

Bit Count: 4

On this screen, you specify the exact bits you want to address.

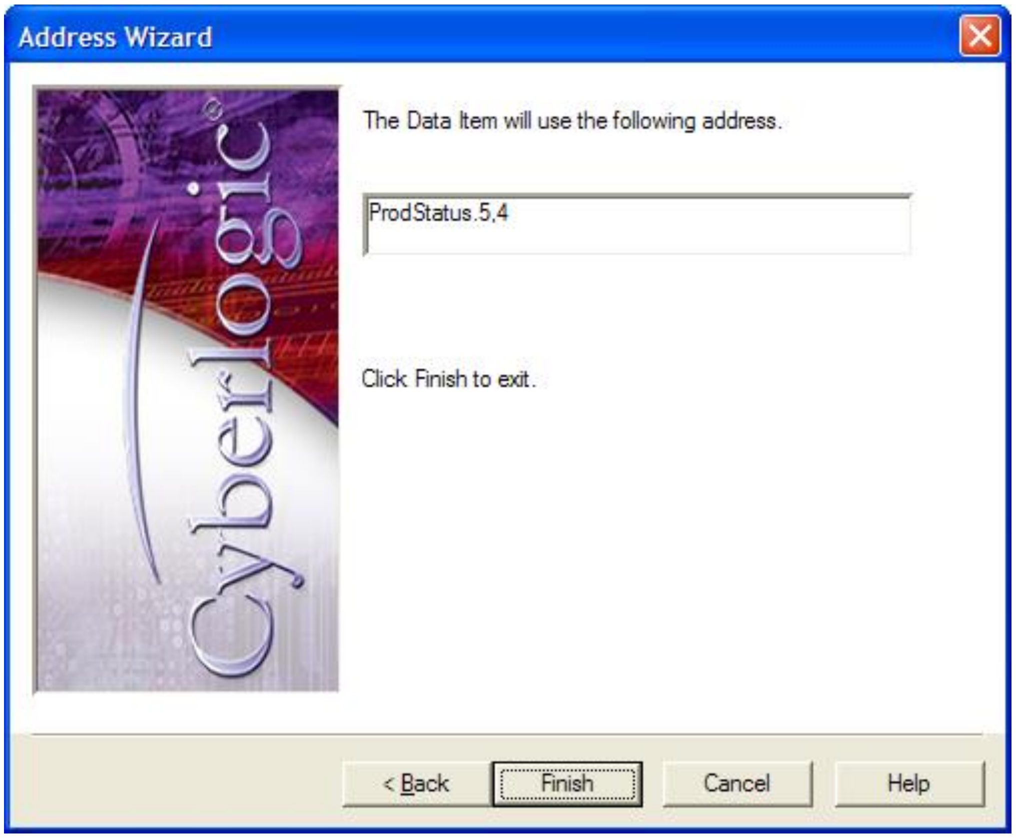
21. Select ***5*** for the ***Bit Number***.

    This is the first bit in the range.

22. Select ***4*** for the ***Bit Count***.

    This specifies that you want a field of four bits.

23. Click ***Next*** to continue.



The Data Item will use the following address.

ProdStatus.5,4

Click Finish to exit.

This screen shows the result of your selections.

24. Click ***Finish*** to allow the wizard to complete and exit.

APPENDIX E: CONFIGURING MAXIMUM CONCURRENT REQUESTS

The Maximum Network Requests and Maximum Node Requests settings limit the number of transactions that the network connection or network node will process simultaneously. Selecting the proper values for these limits can be crucial in obtaining peak performance from your OPC server. The server uses these limits to determine the best way to balance the network load while ensuring an optimum level of communication with each network node.

Although the default values will work well for most common network layouts, some cases may require a few adjustments. In this appendix, we will discuss how to decide on the proper settings to use, and will provide examples of some typical kinds of configurations.

Two Levels of Limits

Maximum request values are set at both the network connection level and at the network node level.

The maximum network request limit is entered on the Settings Tab of the network connection. This parameter limits the total number of simultaneous transactions for all network nodes on that network connection.

The maximum node request limit is entered on the Optimizations Tab of the network node. This parameter allows you to limit the number of simultaneous transactions for that individual network node. If Unlimited is selected, the number of transactions is limited only by the value set for the network connection.

These parameters interact, so each must be taken into consideration when setting the other. For example, the number of simultaneous transactions actually handled by a network node will never be greater than the lower of the two numbers. Furthermore, the number of simultaneous transactions handled by a network connection will never be greater than the total of the limits set on the nodes on that network.

In addition, you must remember that the resources available to process transactions through the network connection are shared among the network nodes. This further restricts the number of transactions that may be available for communication to a given network node. As the client applications request data from the various network nodes, the server arbitrates these competing requests, allowing them to be processed as resources become available. Thus, the number of transactions being processed by each network node will constantly be changing, but each network node will never process more than its limit, and all of them combined will never process more than the limit for the network connection.

Resources and Performance

The purpose of these parameter settings is to allow the user to allocate system resources and network bandwidth in a way that will yield the best performance. If the concurrent request limit is set too low, throughput will suffer even though there are unused resources available to the server.

However, choosing the maximum setting doesn't always improve performance, and may actually make things worse. There are a number of situations in which you should lower these limits.

- *Bridging to other networks*. If you have two or more different networks bridged together, and the messages must pass through more than one network, you should choose values consistent with the slowest network.
- *Other software using the network*. If you have applications other than the OPC server communicating on the network, you may find that the OPC server consumes so much bandwidth that the other applications cannot run satisfactorily. For example, with the OPC server running, your programming software may not be able to connect to the PLC, or uploads and downloads may take excessively long. You may wish to lower the OPC server's limits to prevent the network from saturating and permit the programming software to run.

The goal is to adjust these limits to a level that will allocate the system resources most effectively.

Logix Family of Controllers

Controllers in the Logix family are particularly sensitive to the maximum concurrent request settings. With other controller families, setting the value too high may consume system resources without improving performance. But with the Logix family, a setting that is too high not only wastes resources, but will result in poorer performance or complete failure to update the data. This happens because the controller can become overwhelmed with update requests, causing repeated errors and retries. When this occurs, the OPC server will report the data items as having bad quality, an indication that you should reduce the setting.

The recommended range for Logix family controllers is 4-8, with 6 being the default setting. If you wish to change the setting, the recommended procedure is to make a small change and check to see how that affects the data quality. If you increase the limit and start to see bad quality being reported, you must lower the setting until the bad quality disappears.

Example 1: Network Node Set to Lower Limit

A network connection is configured for a maximum of 16 network requests and one of its nodes is configured for a maximum of 4.

The limit for that node will be 4 simultaneous transactions, regardless of the fact that the network connection can handle more. This means that there will always be at least 12 concurrent requests available to other network nodes using that network connection.

Example 2: Network Node Set to Unlimited

A network connection is configured for a maximum of 8 network requests and one of its nodes is configured as Unlimited.

The maximum number for that node would be 8, that is, everything the network connection can handle. If the network connection limit is increased to 16, then the limit for that network node would increase as well.

Example 3: Interaction Between Multiple Network Nodes

The network connection is configured for a maximum of 20 network requests. It serves five network nodes, each with a limit of 8.

The total of the limits for the five network nodes is then 40, twice what the network connection can handle. This simply means that it will not be possible run the limit of 8 requests on all five network nodes at the same time.

In this configuration, no more than two network nodes could run at their limit of 8 simultaneous transactions, and occasionally one or two might reach that limit. However, the server will always try to spread the load equally among all nodes, so most of the time all nodes would be running at about 4 simultaneous transactions.

Example 4: Network Nodes with Varying Speeds

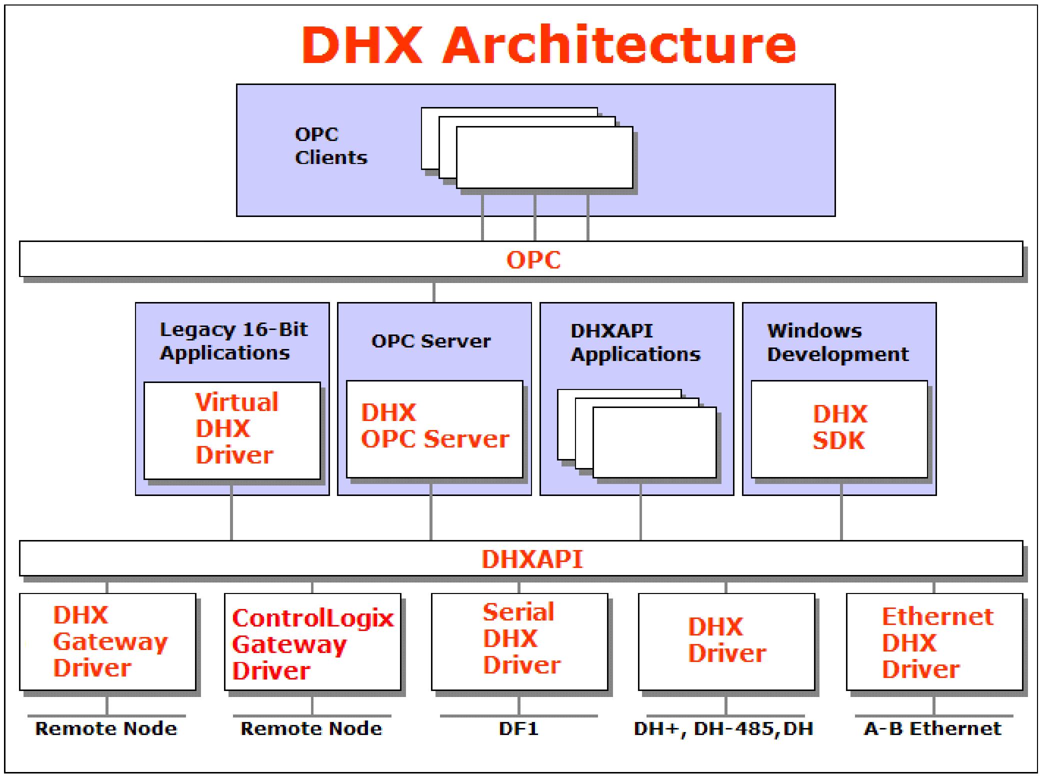
The network connection is configured for a maximum of 10 network requests. It serves three network nodes, one of which is limited to a maximum of 8 concurrent requests, and the other two limited to 2.

The benefit of this configuration is that it avoids the situation where the slower nodes are overwhelmed with transactions they cannot handle, while the faster node is deprived of transactions that it could handle. When the slower nodes are each handling two transactions, the remaining 6 that the network can handle will be available to the faster node.

Appendix F: DHX Architecture and Companion Products

The ControlLogix Driver Agent is part of the Cyberlogic DHX family. This family consists of several well-integrated products that provide connectivity for Data Highway, Data Highway Plus, DH-485, ControlNet and Ethernet networks in distributed environments.

This section illustrates the layout of the DHX architecture. It includes a description of each DHX component along with suggested methods for employing them to support Allen-Bradley networks.



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

DHX Driver

The DHX Driver provides connectivity between Windows-based applications and interface adapter cards from Allen-Bradley and SST. A few of the many cards supported are the 1784-PKTX and 1784-PCMK from Allen-Bradley, as well as the SST DHP-PCI and 5136-SD-PCI from SST. These provide communication over Data Highway, Data Highway Plus and DH 485.

The kernel mode device driver of the DHX Driver has exceptional performance and stability. It operates in either interrupt or polled mode and fully implements all network features, including solicited and unsolicited communication. The high performance native API (DHXAPI) of the DHX Driver takes full advantage of the event-driven, multitasking, multithreaded features of Windows operating systems.

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

The DHX Driver is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite
- DHX Driver Suite

Ethernet DHX Driver

The Cyberlogic Ethernet DHX Driver emulates Data Highway Plus over the Ethernet TCP/IP protocol. It supports most DHXAPI and 6001-F1E-compatible software, providing instant access to Ethernet TCP/IP compatible devices without code modifications. It is compatible with all Ethernet cards supported by Windows.

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

The Ethernet DHX Driver is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite
- DHX Driver Suite

Serial DHX Driver

The Cyberlogic Serial DHX Driver provides connectivity to full-duplex DF1-compatible devices through standard serial COM ports. These devices include the 1770-KF2, 1785-KE, 1770-KF3 and 1770-KFC15 interface modules for Data Highway, Data Highway Plus, DH-485 and ControlNet, as well as direct connection to devices with full-duplex DF1-compatible ports. The Serial DHX Driver supports both the DF1 BCC and DF1 CRC-16 protocols.

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

The Serial DHX Driver is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite
- DHX Driver Suite

DHX Gateway Driver

The DHX Gateway Driver lets applications use DHX devices on remote DHX Gateway Server nodes as though they were on the local system. The client system running the DHX Gateway Driver must be a Windows node connected over a standard LAN to another system running the DHX Gateway Server. It can then access the Data Highway, Data Highway Plus, DH-485 and ControlNet networks that are connected to the server node.

For example, the DHX Gateway Driver provides complete DHX Driver functionality to the client node applications. An interface adapter, such as a 1784-PCMK card, is not required on the client node. DHX Gateway Driver nodes can communicate with multiple remote servers and all Windows-compatible TCP/IP networks are supported.

The DHX Gateway Driver is compatible with all other components of the DHX family.

The DHX Gateway Driver is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite
- DHX Driver Suite

ControlLogix Gateway Driver

The ControlLogix Gateway Driver lets applications access Data Highway Plus networks from a remote location through a ControlLogix gateway module. With this driver, a remote system can communicate over a standard Ethernet network to a ControlLogix chassis containing a 1756-DHRIO module. That module then acts as a gateway to a Data Highway Plus network. This allows the remote system to access the PLC-5s, SLC-500s and any other devices on the Data Highway Plus network as though it were connected directly to that network.

The ControlLogix Gateway Driver is fully compatible with all other components of the DHX family.

The ControlLogix Gateway Driver is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite
- DHX Driver Suite

Virtual DHX Driver

The Virtual DHX Driver allows 16-bit DOS and Windows applications using 1784-KT/KTX interface adapters 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 latest Windows operating systems. By emulating the physical 1784-KT/KTX adapters, the Virtual DHX Driver will work with all legacy software, regardless of which DOS driver is used.

If your computer uses Windows 7 or the 64-bit edition of any Windows version, refer to Cyberlogic Knowledge Base article *KB2010-02 Running 16-Bit Applications* for important information on using the Virtual DHX Driver on your system.

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

- DHX Driver
- Ethernet DHX Driver
- Serial DHX Driver
- DHX Gateway Driver
- ControlLogix Gateway Driver

The Virtual DHX Driver is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite
- DHX Driver Suite

DHX OPC Server

The Cyberlogic DHX OPC Server connects OPC-compliant clients to Data Highway, Data Highway Plus, DH-485, ControlNet and Ethernet networks. It supports the latest OPC Data Access and OPC Alarms and Events specifications and uses the DHX drivers for connectivity to Allen-Bradley networks.

The DHX 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 DHX OPC Server will automatically detect and configure the attached networks and node devices. Other noteworthy features include DirectAccess, Data Write Protection and Health Watchdog.

The DHX OPC Server is included in the following products:

- DHX OPC Enterprise Suite
- DHX OPC Premier Suite
- DHX OPC Server Suite

DHX SDK

Software developers can use the DHX Software Development Kit to provide connectivity to Data Highway, Data Highway Plus, DH-485, Ethernet and ControlNet networks from their 32-bit and 64-bit C/C++/C# applications.

The SDK supports 6001-F1E and Cyberlogic's high-performance DHXAPI and DHXAPI.Net interfaces. The 6001-F1E interface is an excellent bridge for developers who would like to port their 16-bit applications to the latest Windows environments. Developers of new applications can use any of the three interfaces. For a complete 6001-F1E specification, contact any Allen-Bradley distributor.

Since all DHX family drivers are built on the same DHX architecture, applications developed with the DHX SDK can be used with all DHX family drivers and can execute under all current Windows operating systems.