GE EGD Driver

Table of Contents

GE EGD Driver

Help version 1.040

CONTENTS

Overview

The GE Ethernet Global Data (EGD) Driver provides a reliable way to connect GE Ethernet Global Data (EGD) devices to OPC Client applications; including HMI, SCADA, Historian, MES, ERP and countless custom applications.

Setup

Supported Devices

GE devices that support EGD transactions. These devices require CPU-364 or higher.
Any device that supports the EGD protocol.

Ethernet Global Data

Ethernet Global Data is a communications protocol developed by GE in 1998. It allows a device (the Producer) to transfer data to other devices (the Consumers) on the network. Applications that require high-speed networking and coordination from PLC to PLC (or PLC to GE Variable Frequency Drives) can take advantage of EGD. It is up to ten times more efficient on bandwidth utilization than the normal polling system.

Server Project Channel Setup

When setting up a server project for this driver, it is recommended that users only include one channel. If more than one channel is used in a server project, each channel must be configured to bind to a unique network adapter.

Communication Protocol

Ethernet using Winsock V1.1 or higher.

Device ID

This arbitrary field is not used by the driver.

Connection Timeout

This property specifies the time that the driver will wait for a connection to be made with a device. Depending on network load, the connect time may vary with each connection attempt. The valid range is 1 to 60 seconds. The default setting is 3 seconds.

Request Timeout

This property specifies the time that the driver will wait on a response from the device before giving up and going on to the next request. Longer timeouts only affect performance if a device is not responding. The valid range is 100 to 30000 milliseconds. The default setting is 1000 milliseconds.

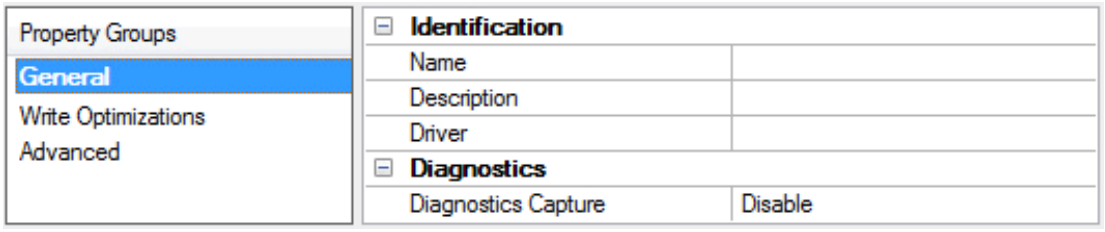
Retry Attempts

This property specifies the number of times the driver will retry a message before giving up and going on to the next message. The valid range is 1 to 10. The default setting is 3.

See Also: Name Resolution and Exchange Configuration.

Channel Properties - General

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.



Identification

Name: User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information.
For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

Description: User-defined information about this channel.
Many of these properties, including Description, have an associated system tag.

Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties.

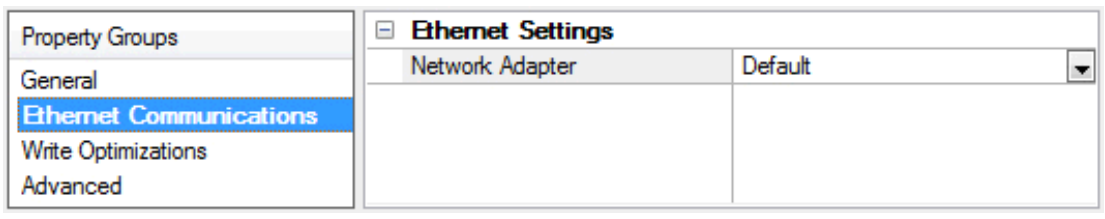
Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

Diagnostics

Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.
Note: This property is disabled if the driver does not support diagnostics.
For more information, refer to "Communication Diagnostics" in the server help.

Channel Properties - Ethernet Communications

Ethernet Communication can be used to communicate with devices.

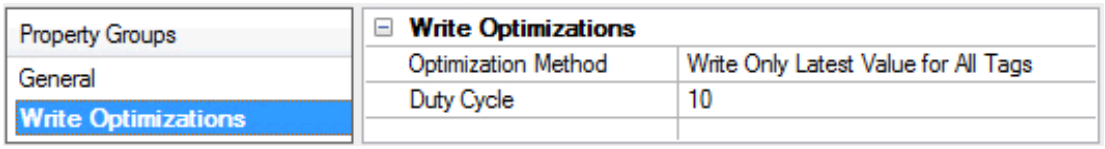


Ethernet Settings

Network Adapter: Specify the network adapter to bind. When Default is selected, the operating system selects the default adapter.

Channel Properties - Write Optimizations

As with any OPC server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used to meet specific needs or improve application responsiveness.



Write Optimizations

Optimization Method: controls how write data is passed to the underlying communications driver. The options are:

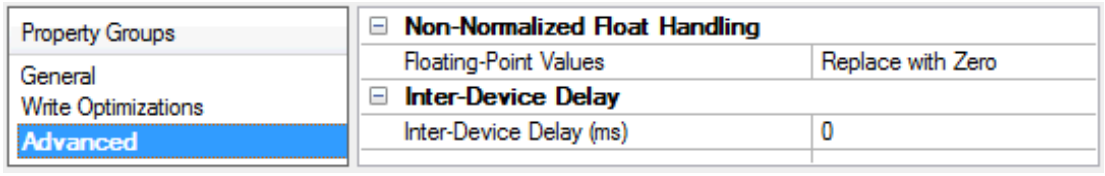
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

Duty Cycle: is used to control the ratio of write to read operations. The ratio is always based on one read for every one to ten writes. The duty cycle is set to ten by default, meaning that ten writes occur for each read operation. Although the application is performing a large number of continuous writes, it must be ensured that read data is still given time to process. A setting of one results in one read operation for every write operation. If there are no write operations to perform, reads are processed continuously. This allows optimization for applications with continuous writes versus a more balanced back and forth data flow.

Note: It is recommended that the application be characterized for compatibility with the write optimization enhancements before being used in a production environment.

Channel Properties - Advanced

This group is used to specify advanced channel properties. Not all drivers support all properties; so the Advanced group does not appear for those devices.



Non-Normalized Float Handling: Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. A non-normalized value is defined as Infinity, Not-a-Number (NaN), or as a Denormalized Number. The default is Replace with Zero. Drivers that have native float handling may default to Unmodified. Descriptions of the options are as follows:

- **Replace with Zero**: This option allows a driver to replace non-normalized IEEE-754 floating point values with zero before being transferred to clients.
- **Unmodified**: This option allows a driver to transfer IEEE-754 denormalized, normalized, non-number, and infinity values to clients without any conversion or changes.

Note: This property is disabled if the driver does not support floating point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM data is not affected by this setting.

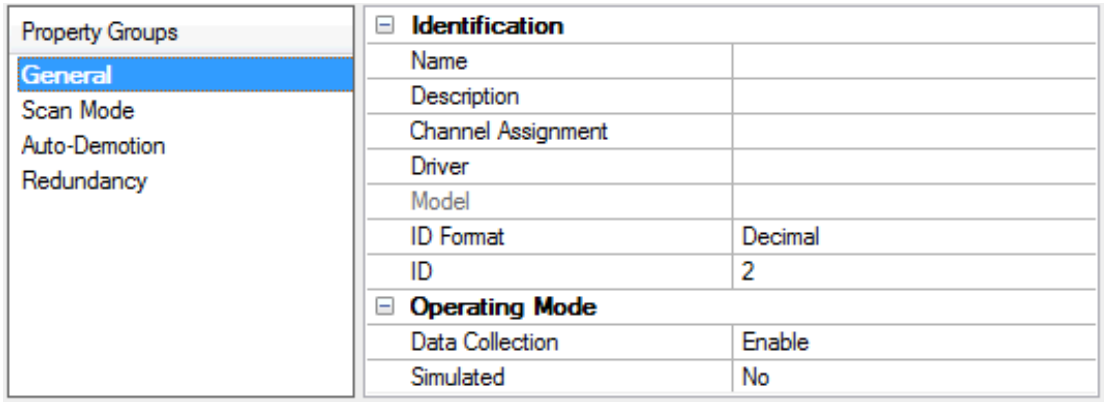
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

Inter-Device Delay: Specify the amount of time the communications channel waits to send new requests to the next device after data is received from the current device on the same channel. Zero (0) disables the delay.

Note: This property is not available for all drivers, models, and dependent settings.

Device Properties - General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.



Identification

Name: This property specifies the name of the device. It is a logical user-defined name that can be up to 256 characters long, and may be used on multiple channels.

Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".
For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

Description: User-defined information about this device.
Many of these properties, including Description, have an associated system tag.

Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device. This property specifies the driver selected during channel creation. It is disabled in the channel properties.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.

Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. For more information, refer to the driver help documentation.

ID: This property specifies the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The ID format can be Decimal, Octal, and Hexadecimal.

Note: If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver. For more information, refer to the driver's help documentation.

Operating Mode

Data Collection: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attempted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.

Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops

physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

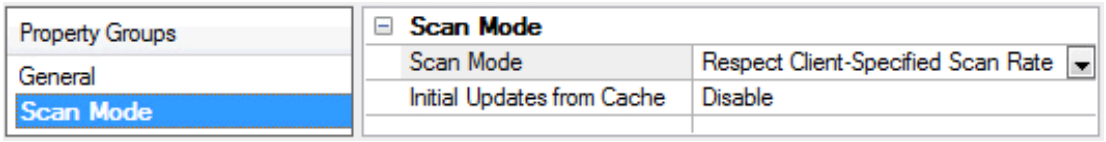
Notes:

1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.
2. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

Simulation Mode is for test and simulation purposes only. It should never be used in a production environment.

Device Properties - Scan Mode

The Scan Mode specifies the subscribed-client requested scan rate for tags that require device communications. Synchronous and asynchronous device reads and writes are processed as soon as possible; unaffected by the Scan Mode properties.



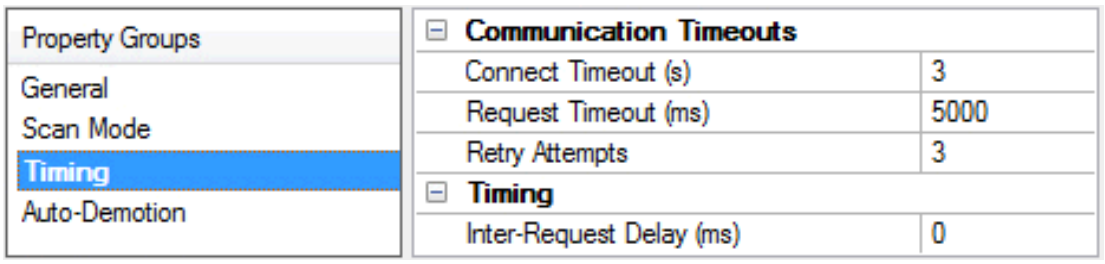
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribed clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the maximum scan rate to be used. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

Initial Updates from Cache: When enabled, this option allows the server to provide the first updates for newly activated tag references from stored (cached) data. Cache updates can only be provided when the new item reference shares the same address, scan rate, data type, client access, and scaling properties. A device read is used for the initial update for the first client reference only. The default is disabled; any time a client activates a tag reference the server attempts to read the initial value from the device.

Device Properties - Timing

The device Timing properties allow the driver's response to error conditions to be tailored to fit the application's needs. In many cases, the environment requires changes to these properties for optimum performance. Factors such as electrically generated noise, modem delays, and poor physical connections can influence how many errors or timeouts a communications driver encounters. Timing properties are specific to each configured device.



Communications Timeouts

Connect Timeout: This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to 30 seconds. The default is typically 3 seconds, but can vary depending on the driver's specific nature. If this setting is not supported by the driver, it is disabled.

Note: Due to the nature of UDP connections, the connection timeout setting is not applicable when communicating via UDP.

Request Timeout: This property specifies an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9,999,999 milliseconds (167.6667 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

Retry Attempts: This property specifies how many times the driver retries a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number of retries configured for an application depends largely on the communications environment.

Timing

Inter-Request Delay: This property specifies how long the driver waits before sending the next request to the target device. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turnaround times and in cases where network load is a concern. Configuring a delay for a device affects communications with all other devices on the channel. It is recommended that users separate any device that requires an inter-request delay to a separate channel if possible. Other communications properties (such as communication serialization) can extend this delay. The valid range is 0 to 300,000 milliseconds; however, some drivers may limit the maximum value due to a function of their particular design. The default is 0, which indicates no delay between requests with the target device.

Note: Not all drivers support Inter-Request Delay. This setting does not appear if it is not available.

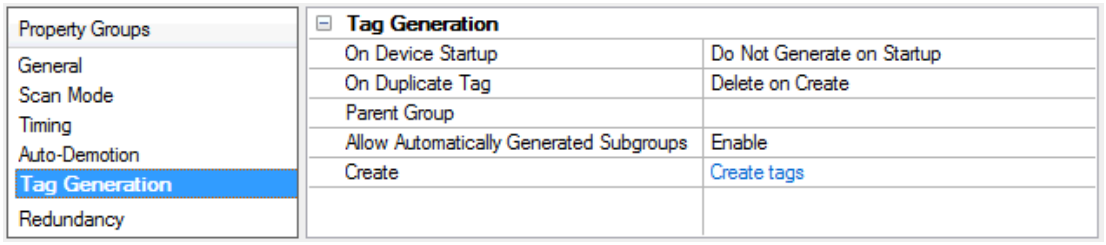
Device Properties - Tag Generation

The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the supporting driver) can be browsed from the clients.

If the target device supports its own local tag database, the driver reads the device's tag information and uses the data to generate tags within the server. If the device does not natively support named tags, the driver creates a list of tags based on driver-specific information. An example of these two conditions is as follows:

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.
2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

Note: Automatic tag database generation's mode of operation is completely configurable. For more information, refer to the property descriptions below.



On Device Startup
This property specifies when OPC tags are automatically generated. Descriptions of the options are as follows:

- **Do Not Generate on Startup**: This option prevents the driver from adding any OPC tags to the tag space of the server. This is the default setting.
- **Always Generate on Startup**: This option causes the driver to evaluate the device for tag information. It also adds tags to the tag space of the server every time the server is launched.
- **Generate on First Startup**: This option causes the driver to evaluate the target device for tag information the first time the project is run. It also adds any OPC tags to the server tag space as needed.

Note: When the option to automatically generate OPC tags is selected, any tags that are added to the server's tag space must be saved with the project. Users can configure the project to automatically save from the **Tools** | **Options** menu.

On Duplicate Tag
When automatic tag database generation is enabled, the server needs to know what to do with the tags that it may have previously added or with tags that have been added or modified after the communications driver since their original creation. This setting controls how the server handles OPC tags that were automatically generated and currently exist in the project. It also prevents automatically generated tags from accumulating in the server.

For example, if a user changes the I/O modules in the rack with the server configured to **Always Generate on Startup**, new tags would be added to the server every time the communications driver detected a new I/O module. If the old tags were not removed, many unused tags could accumulate in the server's tag space. The options are:

- **Delete on Create**: This option deletes any tags that were previously added to the tag space before any new tags are added. This is the default setting.
- **Overwrite as Necessary**: This option instructs the server to only remove the tags that the communications driver is replacing with new tags. Any tags that are not being overwritten remain in the server's tag space.
- **Do not Overwrite**: This option prevents the server from removing any tags that were previously generated or already existed in the server. The communications driver can only add tags that are completely new.
- **Do not Overwrite, Log Error**: This option has the same effect as the prior option, and also posts an error message to the server's Event Log when a tag overwrite would have occurred.

Note: Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags. Users should avoid adding tags to the server using names that may match tags that are automatically generated by the driver.

Parent Group: This property keeps automatically generated tags from mixing with tags that have been entered manually by specifying a group to be used for automatically generated tags. The name of the group can be up to 256 characters. This parent group provides a root branch to which all automatically generated tags are added.

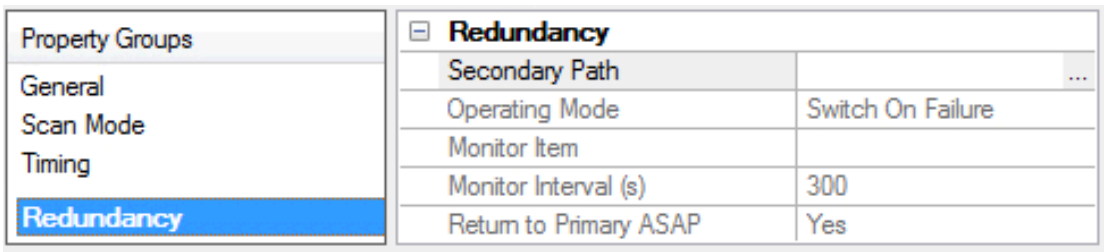
Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.

Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

Create: Initiates the creation of automatically generated OPC tags. If the device's configuration has been modified, **Create tags** forces the driver to reevaluate the device for possible tag changes. Its ability to be accessed from the System tags allows a client application to initiate tag database creation.

Note: **Create tags** is disabled if the Configuration edits a project offline.

Device Properties - Redundancy



Redundancy is available with the Media-Level Redundancy Plug-In.

Consult the website, a sales representative, or the user manual for more information.

Exchange Configuration

Consumer and producer exchanges can be configured within the GE EGD Driver. They must match the exchanges configured in the producing / consuming GE EGD Devices.

Consumer Exchanges

A consumer exchange defines an exchange that will be consumed by the GE EGD Driver. It is configured in a GE EGD device as a producer exchange with a consumed address equal to the IP address of the machine running the device driver. To edit a consumer exchange, highlight Consumer Exchanges and double-click on the desired entry in the detail view.

For information on Device Properties, refer to the server help file.

Descriptions of the properties are as follows:

- **Exchange ID:** This property specifies a user-defined Exchange ID number that will be used when defining tags to reference the exchange. It allows users to change the exchange properties without reconfiguring the pre-existing tags. The default setting is 0. For more information, refer to **Address Descriptions**.
- **Exchange Number:** This property specifies the exchange number. It must match the exchange number of the producing node. The default setting is 0.
- **Producer ID:** This property specifies the IP address of the producing node. The default setting is blank.
- **Group ID:** This property specifies the multicast group ID for when there is more than one consumer receiving the same exchange. The default setting is 0.
- **Consumed Period:** This property specifies the time interval in which the consuming node should receive updates from the producing node. The default setting is 10 milliseconds.
- **Update Timeout:** This property specifies the timeout after which the consumer will declare an error if a consumed exchange is not received. The default setting is 10 milliseconds.

Defining a New Consumer Exchange

To define a new consumer exchange, right-click on **Consumer Exchanges** and click **New Consumer Exchange**.

Defining Range Properties

To define a new range, select an exchange, right-click and select **New Range**. To edit a previously created range, simply double-click on it.

Descriptions of the properties are as follows:

- **Reference:** This property specifies the memory type mnemonic corresponding to the type of data being transferred. The default setting is %AI.
- **Low Point:** This property specifies the starting address offset. The default setting is 0.
- **High Point:** This property specifies the ending address offset. The default setting is 0.

Note: The total amount of range data for an exchange cannot exceed 1400 bytes.

Producer Exchange

A producer exchange defines an exchange that will be produced by the GE EGD Driver. It will be configured in a GE EGD device as a consumer exchange with a producer address equal to the IP address of the machine running the device driver. To edit a consumer exchange, highlight **Producer Exchanges** and double-click on the desired entry in the detail view

Descriptions of the properties are as follows:

- **ID:** This property specifies a user-defined Exchange ID number that will be used when defining tags to reference the exchange. It allows users to change the exchange properties without reconfiguring pre-existing tags. The default setting is 0. For more information, refer to **Address Descriptions**.
- **Exchange Number:** This property specifies the exchange number. It must match the exchange number of the consuming node. The default setting is 0.
- **Consumed Type:** This property specifies the type of consumer address defined in the following field. Options include Group ID, IP, and Name. The default setting is IP.
- **Consumed Address:** This property specifies the consumer address. When the Consumed Type is Name, the address will be a name defined in the Name Resolution dialog. When the Consumed Type is IP, the address will be an IP address in dotted decimal form (*XXX.XXX.XXX.XXX*). When the Consumed Type is Group ID, the address will specify the multicast Group ID.
- **Producer Interval:** This property specifies the time interval in which the GE EGD Driver should produce data and send it to the consuming nodes. The default setting is 10 milliseconds.
- **Reply Rate:** This property is currently not used.

See Also: **Name Resolution**

Defining a New Producer Exchange

To define a new producer exchange, right-click on **Producer Exchanges** and click **New Producer Exchange**.

Name Resolutions

These properties are used to assign logical names to IP addresses for EGD Producer Exchanges. To add a Name Resolution, right-click on Name Resolutions and select New Name Resolution. To edit an existing Name Resolution, select Name Resolutions in the Tree View, and double-click on the desired entry in the detail view.

Note: To remove an entry from the list, right-click the entry and click Delete.

See Also: ***Exchange Configuration***

Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8 bit value bit 0 is the low bit bit 7 is the high bit
Char	Signed 8 bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign bit
Word	Unsigned 16 bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16 bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32 bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32 bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Float	32 bit floating point value. The driver interprets two consecutive 16 bit registers as a floating point value by making the second register the high word and the first register the low word.
Double	64 bit floating point value

Address Descriptions

The Consumer Exchange format is *Cx:r:MMyyyyy*. The Producer Exchange format is *Px:r:MMyyyyy*. Descriptions of the syntax are as follows:

- **C:** This defines a consumer exchange.
- **P:** This defines a producer exchange.
- **x:** This defines the Exchange ID.
- **r:** This defines the Range Number.
- **MM:** This defines the Memory Type Mnemonic.*
- **yyyyy:** This defines the Memory Address.*

*For more information, refer to "Addressing" below.

See Also: Exchange Configuration

Addressing

This driver supports the following memory type, mnemonics, and address ranges. The actual number of addresses for each type depends on the GE Ethernet Global Data Exchange Configuration. The default data types for dynamic tags are shown in **bold**. Consumer exchange references are Read Only, whereas Producer exchange references are Read/Write.

Note: The total amount of range data for an exchange cannot exceed 1400 bytes.

Description	Memory Type and Range	Data Type*
Discrete Inputs	I0001 to I32760 I0001 to I32753 (every 8th bit) I0001 to I32745 (every 8th bit) I0001 to I32729 (every 8th bit)	**Boolean** Byte Word, Short, BCD DWord, Long, Float, LBCD
Discrete Outputs	Q0001 to Q32760 Q0001 to Q32753 (every 8th bit) Q0001 to Q32745 (every 8th bit) Q0001 to Q32729 (every 8th bit)	**Boolean** Byte Word, Short, BCD DWord, Long, Float, LBCD
Discrete Globals	G0001 to G32760 G0001 to G32753 (every 8th bit) G0001 to G32745 (every 8th bit) G0001 to G32729 (every 8th bit)	**Boolean** Byte Word, Short, BCD DWord, Long, Float, LBCD
Internal Coils	M0001 to M32760 M0001 to M32753 (every 8th bit) M0001 to M32745 (every 8th bit) M0001 to M32729 (every 8th bit)	**Boolean** Byte Word, Short, BCD DWord, Long, Float, LBCD
Temporary Coils	T0001 to T32760 T0001 to T32753 (every 8th bit) T0001 to T32745 (every 8th bit) T0001 to T32729 (every 8th bit)	**Boolean** Byte Word, Short, BCD DWord, Long, Float, LBCD
Status References (Same for SB, SC)	SA0001 to SA32760 SA0001 to SA32753 (every 8th bit) SA0001 to SA32745 (every 8th bit) SA0001 to SA32729 (every 8th bit)	**Boolean** Byte Word, Short, BCD DWord, Long, Float, LBCD

Description	Memory Type and Range	Data Type*
Register References	R0000 to R32767 R0000 to R32766 R0000 to R32764	**Word**, Short, BCD DWord, Long, LBCD, Float Double
Analog Inputs	AI0000 to AI32767 AI0000 to AI32766 AI0000 to AI32764	**Word**, Short, BCD DWord, Long, LBCD Float Double
Analog Outputs	AQ0000 to AQ32767 AQ0000 to AQ32766 AQ0000 to AQ32764	**Word**, Short, BCD DWord, Long, LBCD Float Double

*The default data type Boolean will become a Byte when an array specification is given.

Array Support

An array is a collection of contiguous elements of a given data type. The maximum array size is 16 Doubles, 32 DWords (Longs, Floats), 64 Words (Shorts), or 128 Bytes for a total of 1024 bits. The following data types support arrays: Byte, Word, Short, DWord, Long, Float, and Double. For examples on how to specify an array, refer to the table below.

'P50:2:M1' references internal coil 1 which is defined in range 2 of producer exchange 50.	
'C100:3:R2' references register 2 which is defined in range 3 of consumer exchange 100.	
'C1:2:G1 [4]' includes the following **byte** addresses: **Note:** G25 indicates the fourth byte beginning at bit 25.	G1,G9,G17,G25 1 row implied = 4 bytes 4 x 8 (byte) = 32 total bits
'P2:1:R16 [3][4] includes the following **Word** addresses:	R16,R17,R18,R19 R20,R21,R22,R23 R24,R25,R26,R27 3 rows x 4 columns = 12 words 12 x 16 (word) = 192 total bits

Error Descriptions

The following categories of messages may be generated. Click on the link for a list of related messages.

Address Validation

The following error/warning messages may be generated. Click on the link for a description of the message.

Address Validation

Address '<address>' is out of range for the specified device or register

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically references a location that is beyond the range of supported locations for the device.

Solution:

Verify the address is correct; if it is not, re-enter it in the client application.

Array size is out of range for address '<address>'

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically is requesting an array size that is too large for the address type or block size of the driver.

Solution:

Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

Array support is not available for the specified address: '<address>'

Error Type:

Warning

Possible Cause:
A tag address that has been specified statically contains an array reference for an address type that doesn't support arrays.

Solution:
Re-enter the address in the client application to remove the array reference or correct the address type.

Data Type '<type>' is not valid for device address '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically has been assigned an invalid data type.

Solution:
Modify the requested data type in the client application.

Device address '<address>' contains a syntax error

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically contains one or more invalid characters.

Solution:
Re-enter the address in the client application.

Device address '<address>' is not supported by model '<model name>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically references a location that is valid for the communications protocol but not supported by the target device.

Solution:
Verify that the address is correct; if it is not, re-enter it in the client application. Also verify that the selected model name for the device is correct.

Device address '<address>' is Read Only

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically has a requested access mode that is not compatible with what the device supports for that address.

Solution:
Change the access mode in the client application.

Missing address

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically has no length.

Solution:
Re-enter the address in the client application.

GE Ethernet Global Data Device-Specific Messages

The following error/warning messages may be generated. Click on the link for a description of the message.

Consumer exchange (<Exchange ID>) on channel '<channel>' contains an invalid signature from producer (<producer IP>)

Error Type:
Fatal

Possible Cause:
The producing node has sent an exchange which does not conform to the standards expected by the GE EGD Device Driver.

Solution:
Contact Technical Support.

Consumer exchange (<Exchange ID>) on channel '<channel>' is not receiving production data from producer (<producer IP>)

Error Type:
Serious

Possible Cause:
The producing node's producer exchange is configured with no range data.

Solution:
Make sure the producer exchange is configured properly.

Consumer exchange (<Exchange ID>) on channel '<channel>' is not receiving updates within the configured period (elapsed time <time>ms)

Error Type:
Warning

Possible Cause:
Depending on network configuration, it could be taking too long to receive an update from the producer, or a bad network cable could be slowing communications down.

Solution:
Increase the configured time period. Check the network connections to make sure they are functioning at the highest speed possible.

Consumer exchange (<Exchange ID>) on channel '<channel>' is receiving an unsynchronized timestamp from producer (<producer IP>)

Error Type:
Serious

Possible Cause:
The producing node has flagged the produced timestamp to be unsynchronized with the data.

Solution:
Consult the GE EGD PLC documentation.

Consumer exchange (<Exchange ID>) on channel '<channel>' is receiving out of date data from producer (<producer IP>)

Error Type:
Serious

Possible Cause:
The producing node has flagged the produced data as either old or invalid.

Solution:
Consult the GE EGD PLC documentation.

No exchanges have been configured. Tags not generated

Error Type:
Warning

Possible Cause:
No exchanges have been configured.

Solution:
Add one or more exchanges to the device.

Non-production exchange (<Exchange ID>) received from producer (<producer IP>)

Error Type:
Serious

Possible Cause:
The producing node is producing exchanges which are not valid.

Solution:
Call Technical Support.

Received exchange (<Exchange ID>) with an unsupported protocol version (<protocol version received>) from producer (<producer IP>)

Error Type:
Fatal

Possible Cause:
The GE EGD Device Driver does not support the protocol version number received from the producing node.

Solution:
Call Technical Support.

The driver has not been properly configured to receive exchange (<Exchange ID>) from producer (<producer IP>)

Error Type:
Serious

Possible Cause:
The consumer exchange configuration does not match the exchange received from the producing node, or the exchange is not configured.

Solution:
Make sure that the consumer exchange is properly configured.

The xml file contains an alias name which has illegal characters. Only alphanumeric and underscore characters are valid

Error Type:
Fatal

Possible Cause:
The project file has become corrupted.

Solution:
Try saving the project in .opf format, and then resave in .xml format.

Unable to bind consumer socket on '<device>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
The driver was unable to bind to the network adapter or IP Address specified.

Solution:
Try specifying a different network adapter and/or IP address.

Unable to bind producer socket required for multicasting on '<device>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
The driver was unable to bind to the network adapter specified.

Solution:
Try specifying a different network adapter and/or IP address.

Unable to bind to adapter: '<adapter>'. Connect failed

Error Type:
Fatal

Possible Cause:
The operating system could not find an unused port to use for communication with this device.

1. Network system failure, such as Winsock or network adapter failure.
2. Other applications have claimed all available ports (possible but unlikely).

Solution:

1. Reboot the computer and check the network adapter.
2. Check for applications that could be causing conflicts and then shut them down.

Unable to create a producer socket on '<device>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
The operating system could not find an unused port to use for communication with this device.

1. Network system failure, such as Winsock or network adapter failure.
2. Other applications have claimed all available ports (possible but unlikely).

Solution:

1. Reboot the computer and check the network adapter.
2. Check for applications that could be causing conflicts and then shut them down.

Unable to create a consumer socket on '<device>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
The operating system could not find an unused port to use for communication with this device.

1. Network system failure, such as Winsock or network adapter failure.
2. Other applications have claimed all available ports (possible but unlikely).

Solution:
1. Reboot the computer and check the network adapter.
2. Check for applications that could be causing conflicts and then shut them down.

Unable to determine host name for producing node. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
Network system failure or network adapter failure.

Solution:
Check to make sure that the network connection is working and active.

Unable to get host address for producing node. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
Network system failure or network adapter failure.

Solution:
Check to make sure that the network connection is working and active.

Unable to join multicast group on '<IP address>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
The operating system could not find an unused port to use for communication with this device.

1. Network system failure, such as Winsock or network adapter failure.
2. Other applications have claimed all available ports (possible but unlikely).

Solution:

1. Reboot the computer and check the network adapter.
2. Check for applications that could be causing conflicts and then shut them down.

Unable to read producer exchange (<Exchange ID>) configuration data on channel '<channel>'. Terminating producer exchange

Error Type:
Fatal

Possible Cause:
The system may be running low on memory.

Solution:
Make sure the system contains the amount memory required by the server.

Unable to reuse consumer socket on '<device>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
The operating system could not find an unused port to use for communication with this device.

1. Network system failure, such as Winsock or network adapter failure.
2. Other applications have claimed all available ports (possible but unlikely).

Solution:

1. Reboot the computer and check the network adapter.
2. Check for applications that could be causing conflicts and then shut them down.

Unable to set time to live for multicasted socket on '<device>'. (Error Code = <error code>)

Error Type:
Warning

Possible Cause:
Network system failure or network adapter failure.

Solution:
Check to make sure that the network connection is working and active.

Winsock initialization failed (OS Error = n)

Error Type:
Fatal

OS Error:	Indication	Possible Solution
10091	Indicates that the underlying network subsystem is not ready for network communication.	Wait a few seconds and restart the driver.
10067	Limit on the number of tasks supported by the Windows Sockets implementation has been reached.	Close one or more applications that may be using Winsock and restart the driver.

Winsock V1.1 or higher must be installed to use the GE Ethernet device driver

Error Type:
Fatal

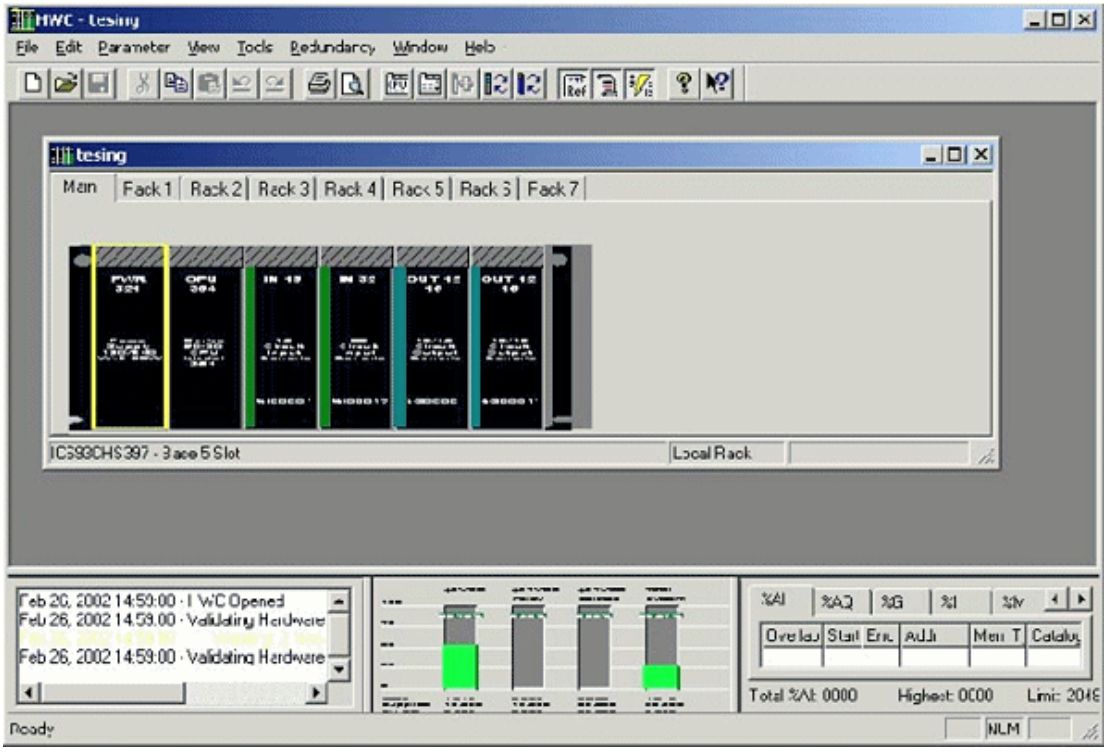
Possible Cause:
The version number of the Winsock DLL found on the system is less than 1.1.

Solution:
Upgrade Winsock to version 1.1 or higher.

Appendix: Configuring EGD in VersaPro and the OPC Server

The example below uses VersaPro v2.01.

1. To start, launch VersaPro. Then, click **File** | **New Folder**.
2. Next, name the project and then click **Finish**. Alternatively, if a project already exists, click **File** | **Open Folder**. Then, locate and open the project.
3. The hardware modules on the GE 90-30 PLC must be configured first in the **Hardware Configuration Utility** that is included with VersaPro. To do so, double-click on **Hardware Configuration** in the **Folder Browser**.

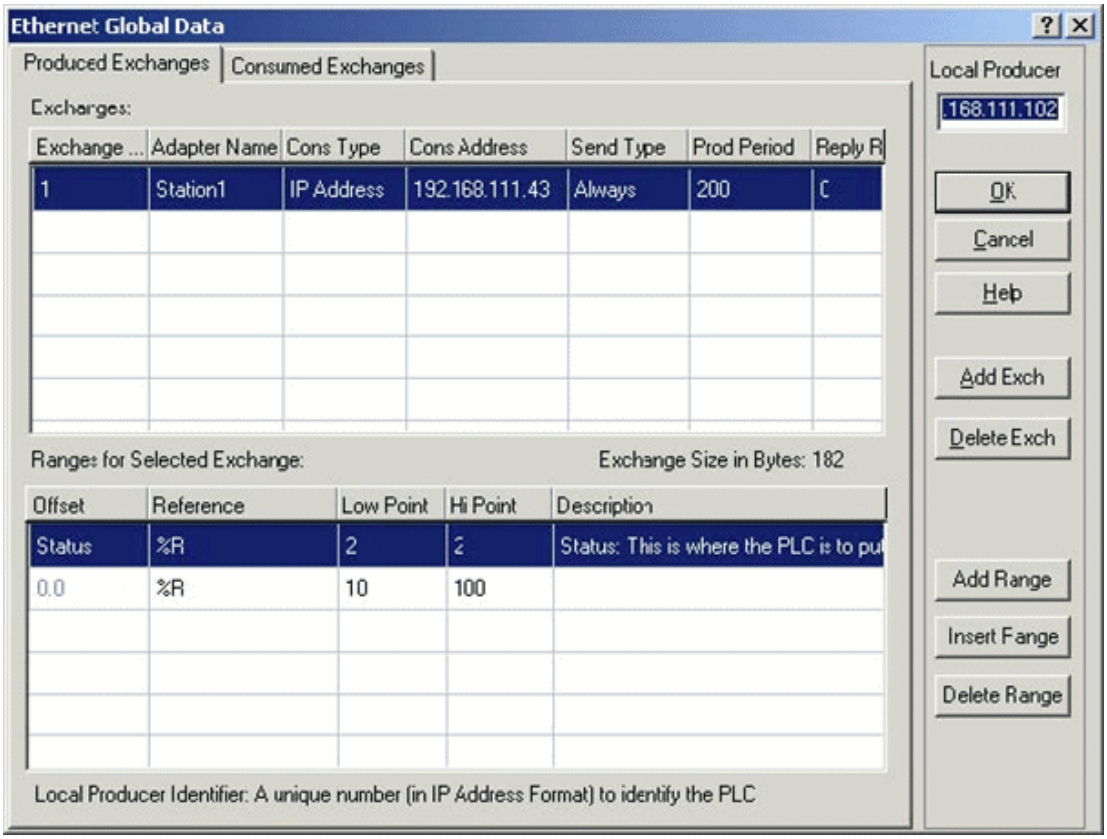


4. Once finished, open the **Ethernet** tab in the CPU's configuration window. Then, set the following parameters:

   - **Adapter Name:** This parameter specifies the PLC during EGD configuration in VersaPro. It may be named as desired.
   - **IP Address**
   - **Subnet Mask**
   - **Gateway IP Address**
   - **Status Address:** This parameter specifies the start of an 80 bit global status area. It may be 80 consecutive single-bit locations or 5 consecutive 16 bit registers, and may kept at the default setting.

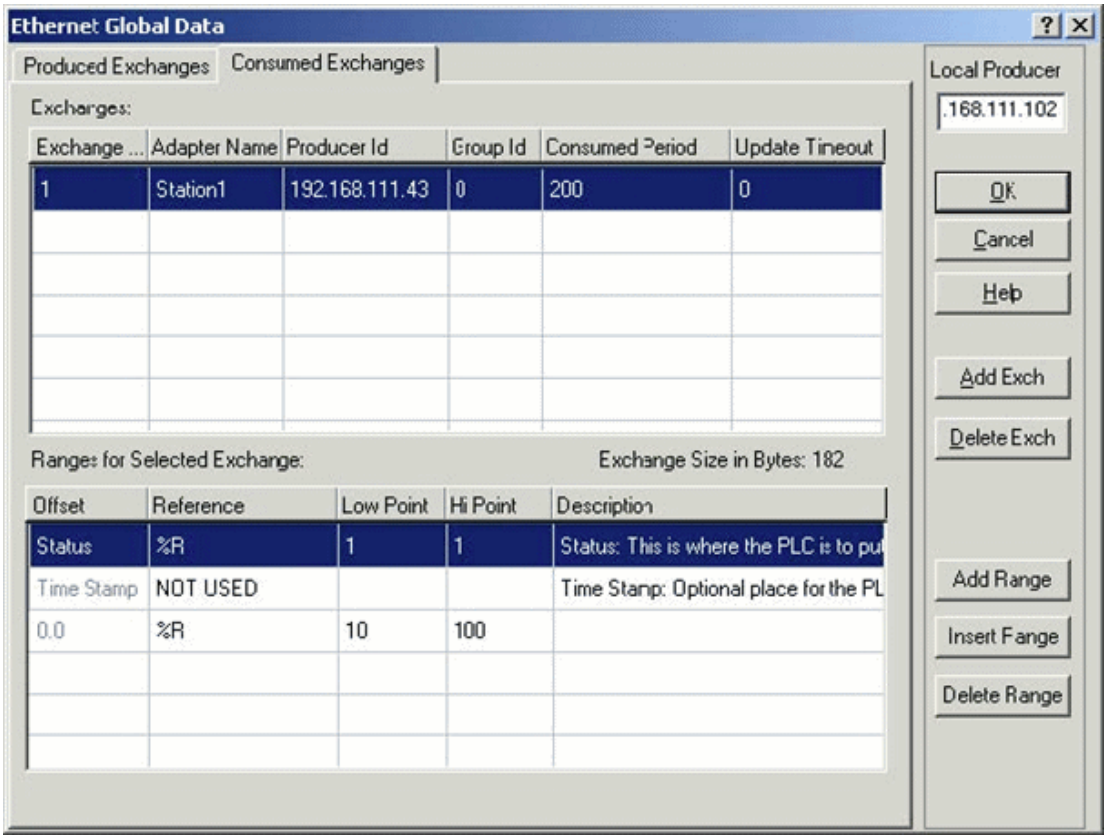
Note: Users should be aware of this address so that there are no later attempts to allocate something else in that same register space.

5. Once the hardware configuration is complete, save it to disk. Then, return to VersaPro and select **Tools** | **Communications Setup** to invoke the **Communication Configuration Utility**.
6. Ensure that the PLC is listed in this utility with the correct IP address. If it is not, add the PLC using **Port ENET**. Then, click **OK** to save the changes and exit the communication configuration utility.
7. In VersaPro, click **PLC** | **Connect** to test the PLC's connectivity. Once established, click **Edit** | **Rack Operations** | **Ethernet Global Data**.



8. In **Produced Exchanges**, complete the following:
   - Enter the PLC's IP address in **Local Producer**.
   - Click on **Add Exch** to make a new exchange.
   - In **Exchange ID**, enter 1.
   - In **Cons Address**, enter the IP address of the PC running the OPC server.
   - In **Prod Period**, specify how often the PLC will send the data to the consumers in milliseconds. The consumers (or OPC Server) must be set to receive data at this same period.
   - Define a 16 bit Exchange Status Area. In this example, %R2 has been selected. Users must ensure that this status area does not overlap the global status area already defined. After the exchange status area is defined, add the ranges of data that the PLC will produce.

9. When all is finished, click the **Consumed Exchanges** tab.



10. In **Consumed Exchanges**, complete the following:

    - Match the Producer ID with the IP address of the PC running the OPC Server.
    - In **Consumed Period**, specify how often the PLC will expect the data from the Producer. The Producer (or OPC Server) must be set to send data at this same period.
    - Define a 16 bit Status Area. In this example, %R1 has been selected.

      Note: In this example, the data ranges are set to match those in the Produced Exchanges screen. That is, a bi-directional transfer of this register range is set up over EGD. This causes the Hardware Configuration Utility to display non-fatal warnings that the ranges in the producer and consumer exchanges are overlapping. These warnings may be ignored because they would not appear if the register range was only set to be produced or consumed.

11. Once the parameters are set, click **OK**.
12. In VersaPro, click **PLC** | **Store** to download the configuration to the PLC. Then, turn on **Store Hardware Configuration and Motion to the PLC** and click **OK**.
13. Next, click **PLC** | **Run** to start the program in the PLC.

Note: In order to communicate using EGD, the PLC must be in "Enabled" mode (outputs enabled).

Configuring EGD in the OPC Server

1. To start, create a new OPC Server project.
2. Then, add a new channel. In **Device Driver**, select **GE Ethernet Global Data** from the drop-down menu. Continue through the Channel Wizard, specifying the channel settings as desired. Once complete, click **Finish**.
3. Next, add a new device. Accept the default settings until reaching the **Exchange Configuration** properties.
4. Open the **Consumer Exchanges** tab.
5. Click **Add Exchange** and specify the following:
    - **ID:** 1.
    - **Exchange Number:** 1.
    - **Producer ID:** The IP address of the PLC.
    - **Consumed Period:** This value must match the "Prod. Period" value entered in VersaPro.
    - **Update Timeout:** A minimum setting of 100 milliseconds is suggested. In this example, 200 milliseconds is used.
6. Select the newly added exchange and then click **Add Range**. Add the address range that was specified in VersaPro. The status area does not need to be defined.
7. Open the **Producer Exchanges** tab.
8. Click **Add Exchange** and specify the following:
    - **ID:** 1.
    - **Exchange Number:** 1.
    - **Consumed Type:** The IP.
    - **Producer Interval:**This value must match the Consumed Period value entered in VersaPro.
    - **Consumed Address:** IP address of the PLC.
9. Next, select the newly added exchange and then click **Add Range**. Add the address range that was specified in VersaPro. The status area does not need to be defined.
10. Click **Next** | **Finish**.
11. Then, save the project.
12. In the OPC Server, "Channel1" and "Device1" should now be visible. Right-click on "Device1," and then select **Properties** | **Database Creation**.
13. Next, click **Auto Create**. Then, press **OK**.

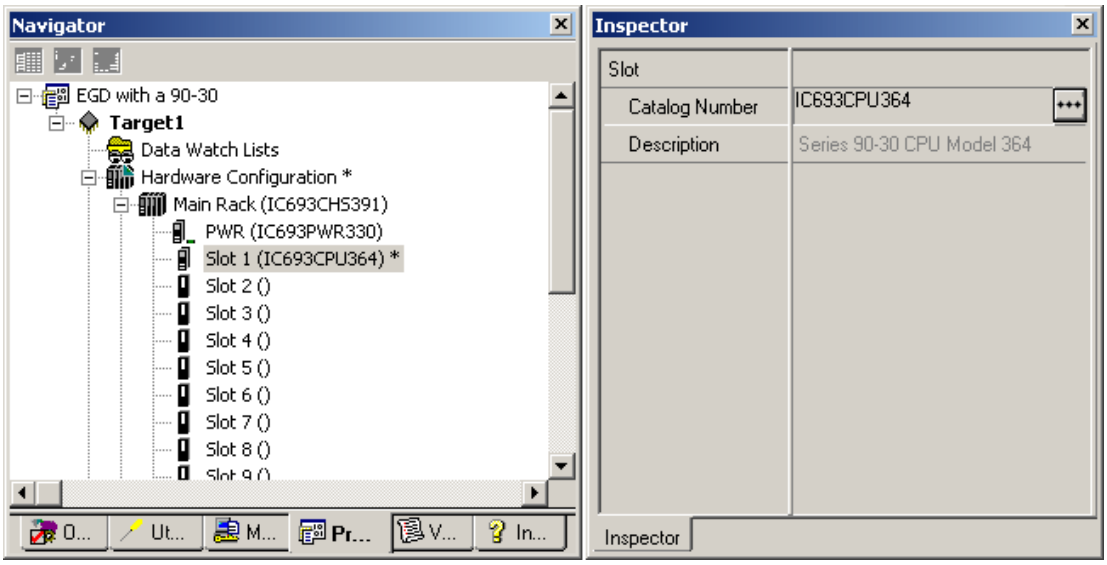
    Note: To display the newly-added tag groups for the consumer and producer exchanges, click on the plus sign located to the left of "Device1" to expand it. The tag groups will contain tags that correspond to the ranges that were set in VersaPro.
14. Launch the OPC Quick Client. Good values should be displayed in both tag groups.

Appendix: Configuring EGD in GE Proficy Machine Edition

The example below uses GE Proficy version 6.0. The PLC's IP address is 192.168.0.0 and the OPC server's IP address is 192.168.0.99.

1. To start, open GE Proficy. Then, use the project templates to create a new, generic project for a GE Series 90-30 PLC.
2. In the **Navigator** window, open the **Project** tab to display the Project tree and select a CPU.
3. Under **Hardware Configuration**, select **Main Rack (IC693CHS391)**. Then, click **Slot1 (IC693CPU3xx)** to display the CPU's properties in the **Inspector** window.

   Note: When an item is selected in the Navigator's Project tree, its properties will become visible in the Inspector window. Both the Navigator and Inspector windows float in the main GE Proficy window.

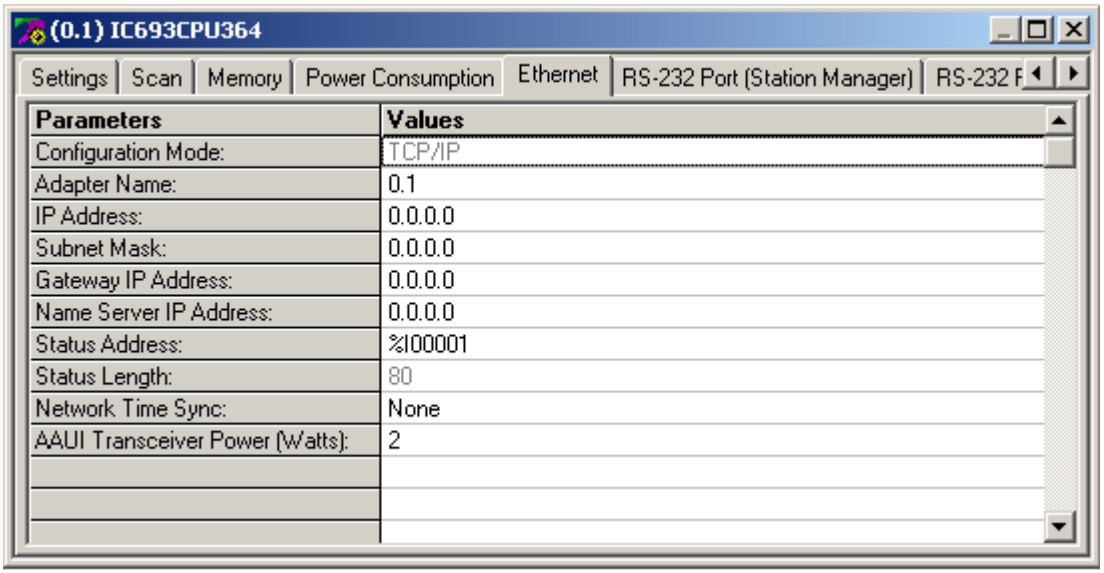


   Note: A default catalog number will be displayed (such as "IC693CPU374"). For more options, click the **Selection** icon and then choose the desired CPU. In this example, "IC693CPU364" is used.

4. Next, select a power supply for the project. To do so, locate **Main Rack (IC693CHS391)** and then click **PWR (IC693PWR3xx)** to display the power supply's properties in the Inspector window.

   Note: A default catalog number will be displayed (such as "IC693PWR330"). For more options, click **Selection** and then choose the desired power supply. In this example, "IC693PWR321" is used.

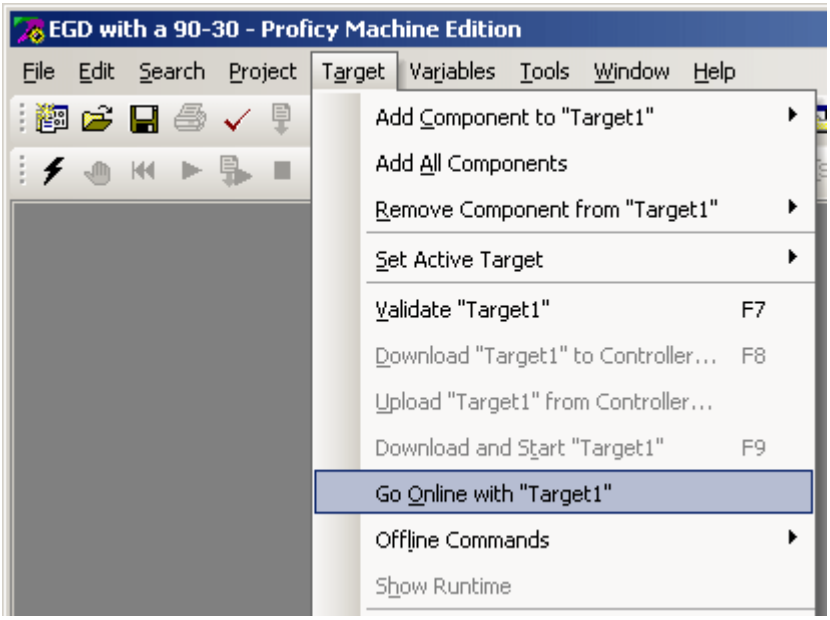
5. Select modules as needed for the remaining slots under Main Rack. Then, configure the CPU. In the project tree, under **Hardware Configuration** | **Main Rack (IC693CHS391)**, double-click on **Slot1 (IC693CPU364)** to display the CPU's configuration window.



Note: Almost all of the entries in the **Ethernet** tab will be defaulted to zeros.

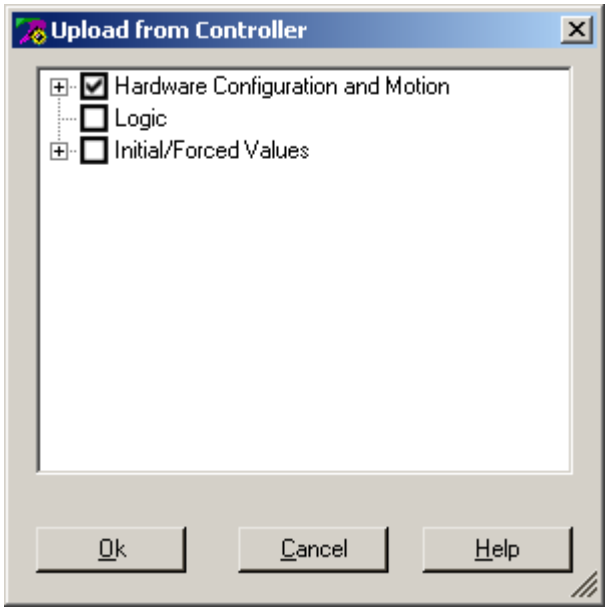
Important: It is strongly recommended that the PLC's pre-existing CPU configuration be uploaded into Proficy instead of entered manually. To do so, a serial connection is required between the PLC and the computer that is running Proficy. Use the PLC's SNP port and an SNP cable for the upload. For more information on uploading the CPU configuration, refer to the instructions below.

1. To get online with the PLC, click the lightning bolt icon in the menu bar. Alternatively, open **Target** at the top of the Proficy screen and then select **Go online with "Target1"**.



Note: A connection will be established after several seconds. If the PLC's configuration does not match the Proficy project's configuration, a red cross will appear over "Target1" in the project tree. This cross is informational and should not cause concern.

2. In the **Target** drop-down menu, select **Upload "Target1" from controller**.
3. In **Upload from Controller**, check **Hardware Configuration and Motion**. Leave the remaining two options unchecked. Once finished, click **OK**.



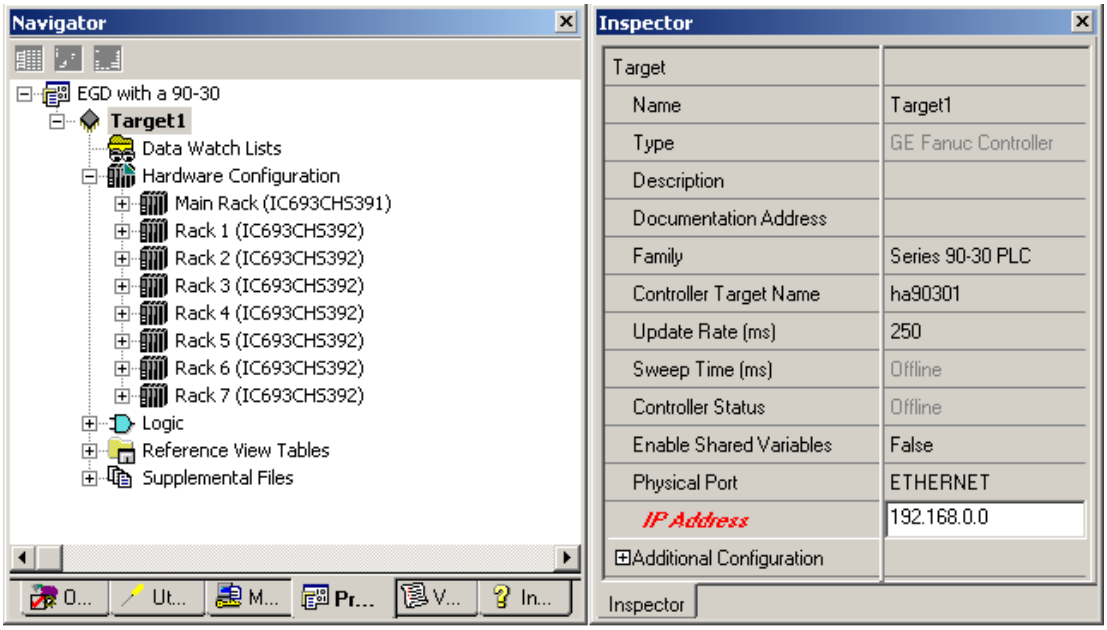
4. Next, re-open the CPU configuration and examine the **Ethernet** tab. An Ethernet configuration should have been uploaded from the PLC, and all properties (except for the Adapter Name) should now contain values. Users must enter a valid Adapter Name at a later time, and note the IP Address for later use.

   Note: The red cross should still be displayed over the **Target1** icon because a partial upload of the PLC was performed.

5. Next, open the **Target** drop-down menu and select **Go offline with Target1**. This will disconnect the serial connection between Proficy and the PLC, remove the red cross from Target1.

   Note: The serial connection can also be used to download a new CPU configuration.

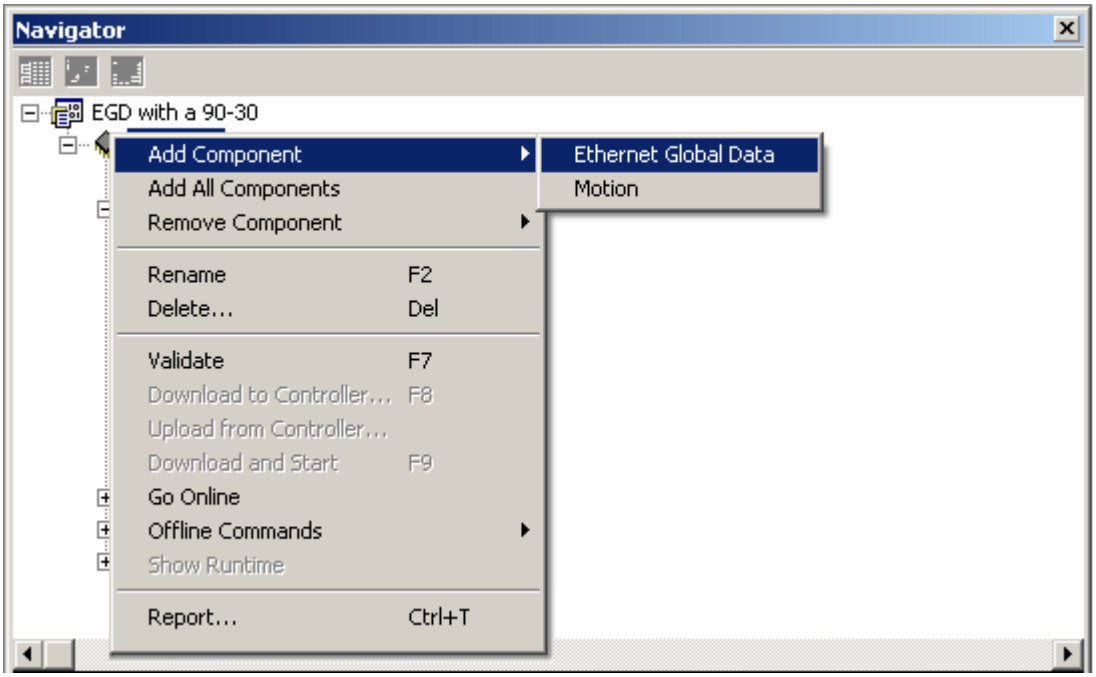
6. Next, select **Target1** at the top of the project tree. In the Inspector window, locate the **Physical Port** parameter and change it to "ETHERNET". In the **IP Address** parameter, enter the same IP address that appears in the CPU's Ethernet configuration.



7. Next, enter an Adapter Name in the CPU's configuration. To do so, double-click on **Slot1 (IC693CPU364)** and then select the **Ethernet** tab.
8. Double-click in the value field of **Adapter Name** and type "0.1."

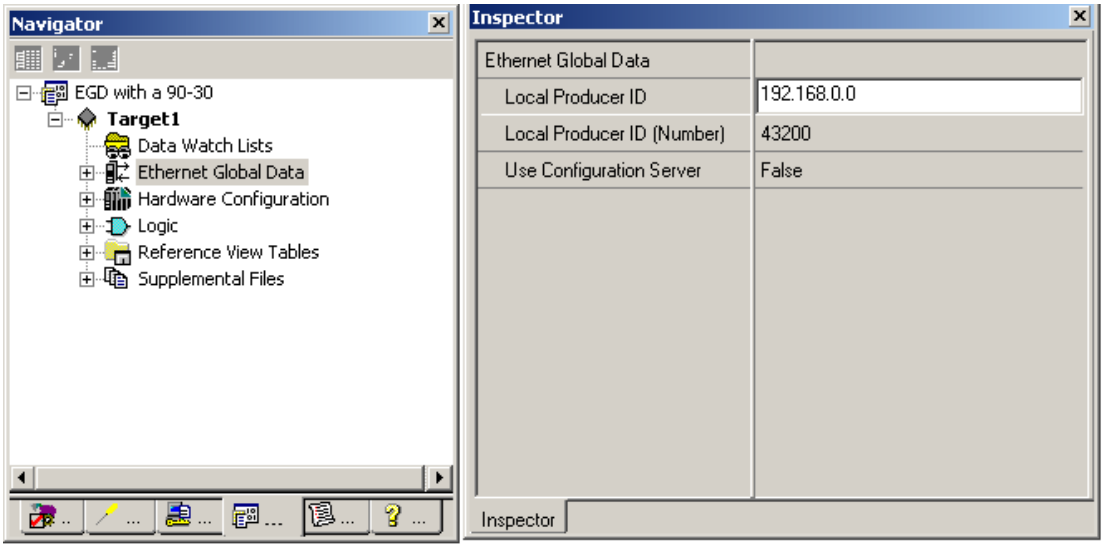
Note: At this point, a serial connection is not necessary.

9. Next, add EGD components to the project by right-clicking on **Target1** and then selecting **Add Component** | **Ethernet Global Data**.



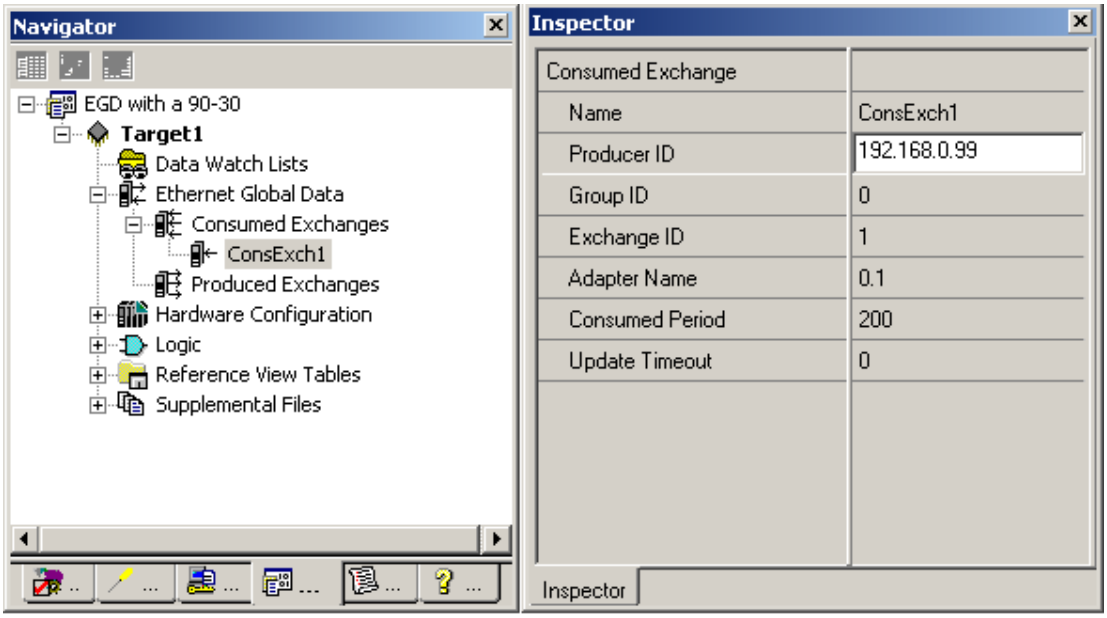
Note: The **Ethernet Global Data** icon should now appear in the project tree. Open its properties in

the Inspector window, and then locate the **Local Producer ID** parameter and enter the EGD Producer's IP address. In this example, the PLC's IP address is used.

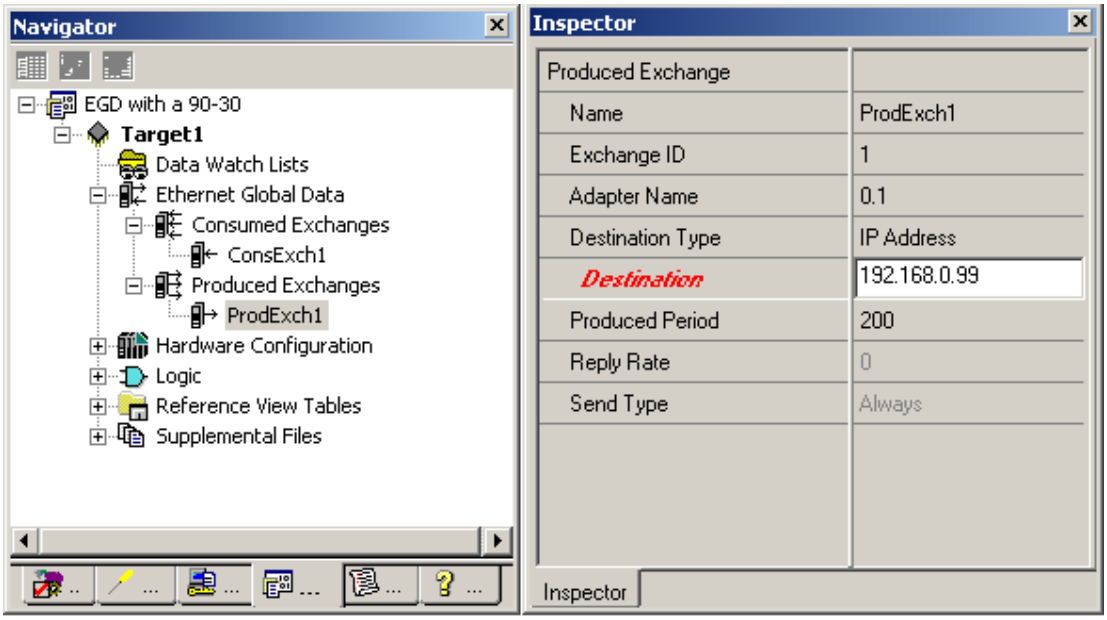


10. Expand the Ethernet Global Data icon to reveal the **Consumed Exchanges** and **Produced Exchanges** icons.

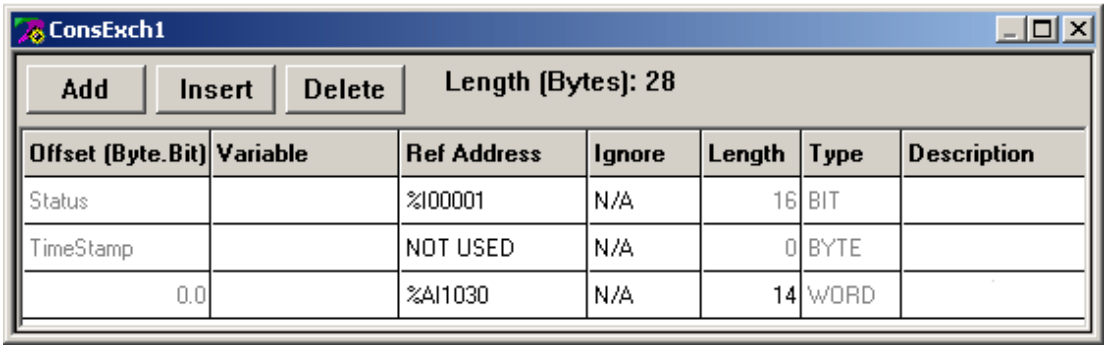
11. Next, right-click on **Consumed Exchanges** and select **New**. The icon **ConsExch1** will appear. Display its properties in the Inspector window and then locate the **Producer ID** parameter. Enter the IP address of the computer on which the OPC server will be running.



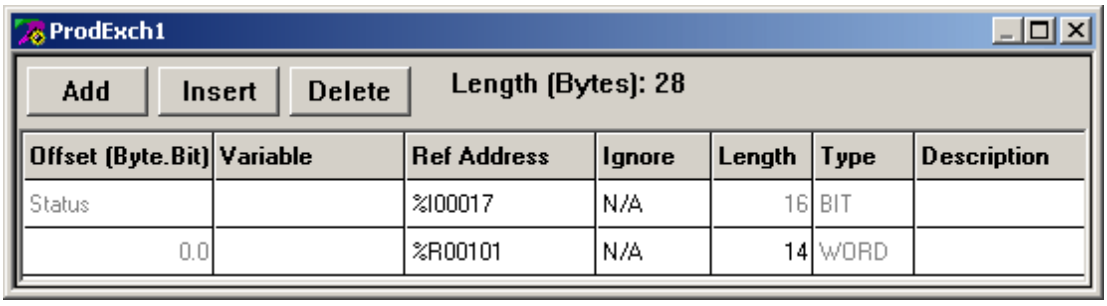
12. Next, right-click on Produced Exchanges and select **New**. The icon **ProdExch1** will appear. Display its properties in the Inspector window and then locate the **Destination Type** and **Destination** properties. In **Destination Type**, set the parameter to "IP Address." In **Destination**, enter the IP address of the computer on which the OPC server will be running.



13. Next, double-click on **ConsExch1** to invoke the "ConsExch1" window, which is used to add register addresses to the Consumer Exchange. Then, click **Add** to insert a new row. In this demonstration, the new row's **Ref Address** is set to "%AI1030." The **Length** is set to 14.



14. Then, double-click on **ProdExch1** to invoke the "ProdExch1" window, which is used to add register addresses to the Producer Exchange. Then, click **Add** to insert a new row. In this demonstration, the new row's **Ref Address** is set to "%R00101." The **Length** is set to 14.



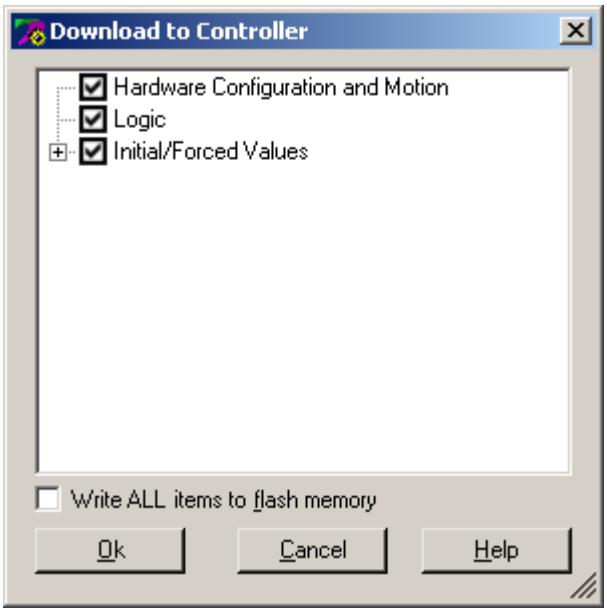
15. Next, create ladder logic for the project by expanding the **Logic** icon in the Project tree and then double-clicking on "_Main." This will invoke a window for creating and editing ladder logic.

Important: In order to place the PLC into the Run state, a ladder must be present in the PLC's

memory. As such, ladder logic must be included in an EGD configuration for the 90-30 PLC even if the logic is not directly relevant.

Downloading the Project to the PLC

1. To start, save the Proficy project and then get online with the PLC by clicking the lightning bolt icon. Alternatively, click **Target** | **Go online with Target1**.
2. The project must be validated before a download can be attempted. To do so, open the **Target** menu and then select **Validate "Target1"**. Then, check for validation errors by opening the **Feedback Zone** window. If there are no errors, continue on.
3. Next, toggle Proficy into Programmer Mode by clicking the green hand icon (located in the Proficy toolbar). Then, click **Target** | **Download** and select **Target1**.
4. In the **Download to Controller** window, ensure that all three items in the window are checked. Then, click **OK**.



Note: If the download was successful, the Feedback Zone window should report no errors. The red cross should no longer be over Target1 because the PLC's configuration will match the Proficy project's configuration.

Configuring an OPC Server Project

1. To start, open a new server project and then create a channel. In **Device Driver**, specify **GE Ethernet Global Data**. Continue through the Channel Wizard, specifying the channel settings as desired. Once complete, click **Finish**.
2. Next, add a new device. Accept the default settings until reaching the **Exchange Configuration** dialog. Then, select the **Consumer Exchanges** tab and click **Add Exchange**. Set the parameters to the following:

   - **ID:** 1.
   - **Exchange Number:** 1.

- **Producer ID:** The 90-30 PLC's IP Address.
- **Group ID:** 0.
- **Consumed Period:** 200.
- **Update Timeout:** 200.

3. Then, click **OK**.
4. Next, select the newly-created Consumer Exchange and then click **Add Range**. Set the parameters to the following:

   - **Reference:** %R.
   - **Low Point:** 101.
   - **High Point:** 114.

5. Click **OK**.
6. Select the **Producer Exchanges** tab and click **Add Exchange**. Set the parameters to the following:

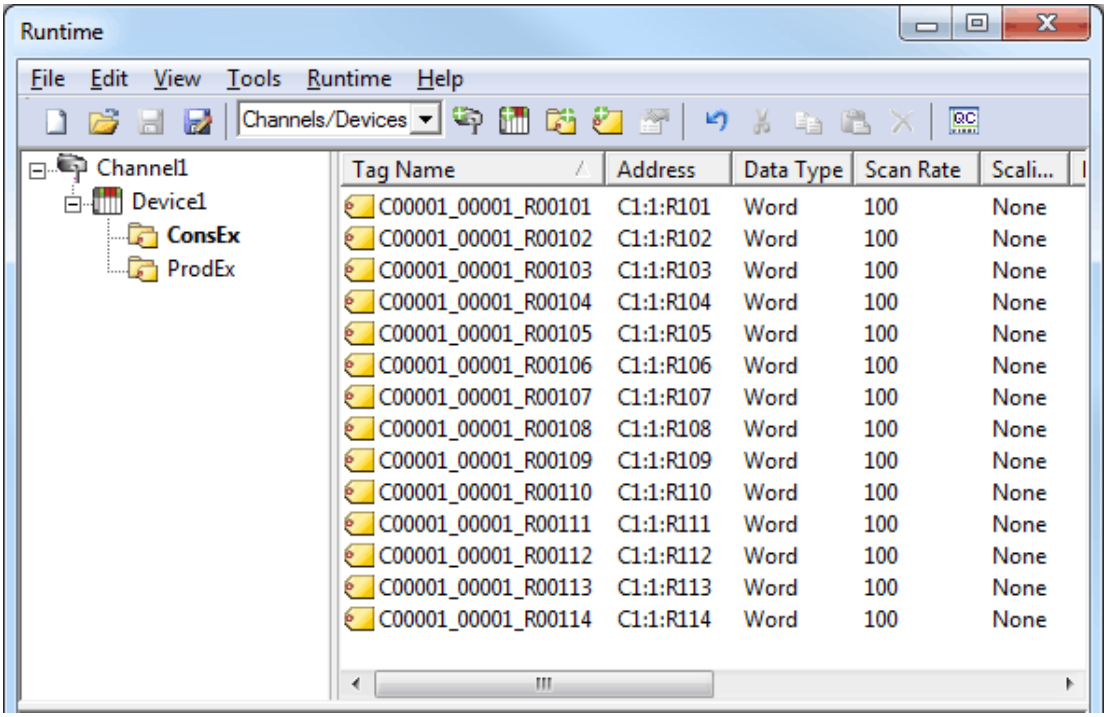
   - **ID:** 1.
   - **Exchange Number:** 1.
   - **Consumed Type:** IP.
   - **Consumed Address:** The 90-30 PLC's IP address.
   - **Producer Interval:** 200.

7. Click **OK**.
8. Select the newly-created Producer Exchange and then click **Add Range**. Set the parameters to the following:

   - **Reference:** %AI.
   - **Low Point:** 1030.
   - **High Point:** 1043.

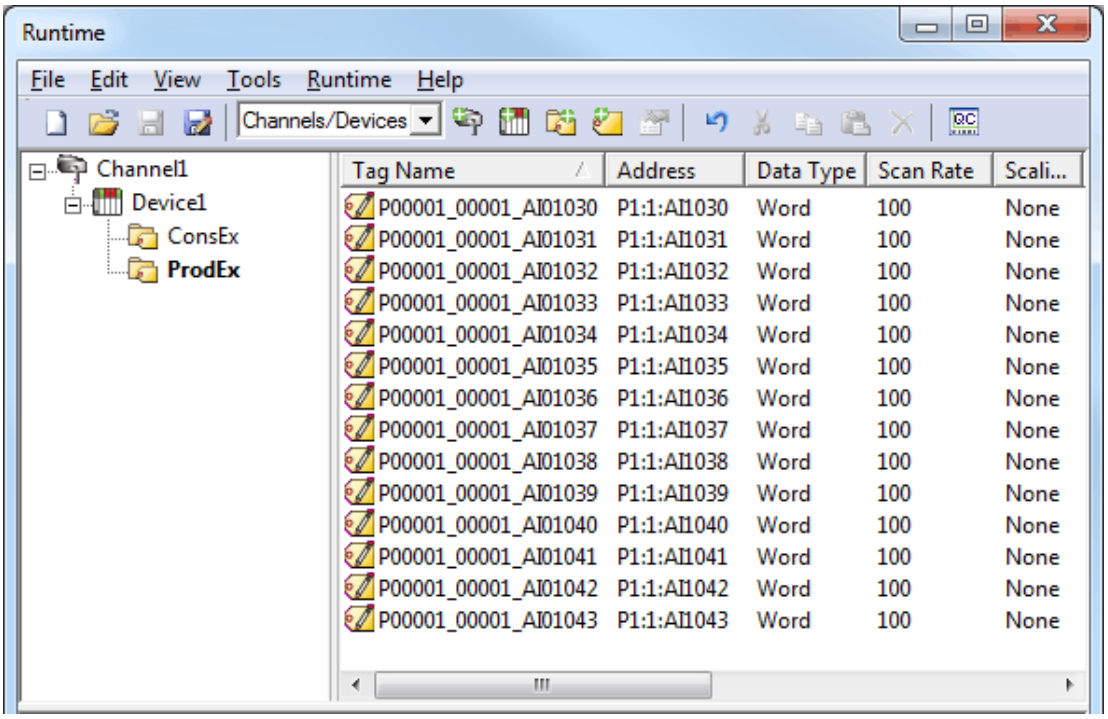
9. Then, click **OK**.
10. Next, add a new tag group beneath the device. To do so, right-click on the device and then select **New Tag Group**. In **Name**, enter "ConsEx" and click **OK**.

11. Open the folder and create new tags at the addresses "C1:1:R101" through "C1:1:R114".



12. Next, add a new tag group beneath the device. To do so, right-click on the device and then select **New Tag Group**. In **Name**, enter "ProdEx". Then, click **OK**.

13. Open the folder and create new tags at the addresses "P1:1:AI1030" through "P1:1:AI1043".



14. Next, start the OPC Quick Client. If the PLC is online and the Proficy project created earlier has been downloaded to the PLC, the tags should show Good quality. Users should also be able to write to the Producer Tags.
15. By going online with Proficy, users can check the values held by PLC registers at the addresses %AI1030 to %AI1043 and %R101 to %R114. To find the registers, return to the project tree and then click **Reference View Tables** | **Default Tables**.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W