

# AutomationDirect Productivity Series Ethernet Driver

© 2025 PTC Inc. All Rights Reserved.

# Table of Contents

<b>AutomationDirect Productivity Series Ethernet Driver</b>	<b>1</b>
<b>Table of Contents</b>	<b>2</b>
Welcome to the AutomationDirect Productivity Series Ethernet Driver Help Center	4
Overview	4
<b>Setup</b>	<b>4</b>
Channel Properties – General	5
Tag Counts	5
Channel Properties – Ethernet Communications	5
Channel Properties – Write Optimizations	6
Channel Properties – Advanced	6
Device Properties – General	8
Operating Mode	8
Tag Counts	9
Device Properties – Scan Mode	9
Device Properties – Timing	10
Device Properties – Auto-Demotion	11
Device Properties – Tag Generation	11
Device Properties – Communication	13
Device Properties – Data Handling	13
Device Properties – Tag Import File Path	13
Device Properties – Redundancy	14
<b>Automatic Tag Database Generation</b>	<b>15</b>
<b>Optimizing Communications</b>	<b>17</b>
<b>Data Types Description</b>	<b>18</b>
<b>Address Descriptions</b>	<b>19</b>
Ordering of Array Data	21
Address Formats	21
<b>Error Descriptions</b>	<b>23</b>
Tag name <tag name> encountered validation error and will not be imported.	23
Unable to generate a tag database for device <device name>. Reason: Auto tag generation cancelled.	24
Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt.	24
Unable to generate a tag database for device <device name>. Reason: Import file not found.	24
Unable to generate a tag database for device <device name>. Reason: Low memory resources.	25
Unable to bind to adapter: <network adapter>. Connect failed. Winsock Err# <error number>.	25
Winsock initialization failed (OS Error = <OS error code>).	25
Winsock shut down failed (OS Error = <OS error code>).	25
Winsock V1.1 or higher must be installed to use the Productivity Series Ethernet device driver.	26
Cannot read <tag count> items starting at tag <tag address>: address does not exist in device <device name>.	26
Cannot read <tag count> items starting at tag <tag address>: device <device name> returned error code <error code>.	26
Cannot read <tag count> items starting at tag <tag address>: error receiving response frame from	26

device <device name> .....	
Cannot read <tag count> items starting at tag <tag address>: System ID <System ID> does not exist in device <device name> .....	26
Cannot read <tag count> items starting at tag <tag address>: value is invalid for data type <data type> in device <device name> .....	27
Cannot read tag <tag address>: address does not exist in device <device name> .....	27
Cannot read tag <tag address>: device <device name> returned with error code <error code> .....	27
Cannot read tag <tag address>: error receiving response frame from device <device name> .....	27
Cannot read tag <tag address>: System ID <System ID> does not exist in device <device name> .....	27
Cannot read tag <tag address>: value is invalid for data type <data type> in device <device name> .....	28
Cannot write to tag <tag address>: address does not exist in device <device name> .....	28
Cannot write to tag <tag address>: device <device name> returned with error code <error code> .....	28
Cannot write to tag <tag address>: error receiving response frame from device <device name> .....	28
Cannot write to tag <tag address>: System ID <System ID> does not exist in device <device name> .....	28
Cannot write to tag <tag address>: value is invalid for data type <data type> in device <device name> ..	29
Modbus Exception Codes .....	30
<b>Appendix: Exporting a CSV File from Productivity Suite .....</b>	<b>30</b>
<b>Index .....</b>	<b>32</b>

## Welcome to the AutomationDirect Productivity Series Ethernet Driver Help Center

This help center is the user documentation for Kepware AutomationDirect Productivity Series Ethernet Driver. This help center is updated regularly to reflect the latest functionality and information.

### ["Overview" below](#)

What is the AutomationDirect Productivity Series Ethernet Driver?

### [Setup](#)

How do I configure a device for use with this driver?

### [Automatic Tag Database Generation](#)

How can I easily configure tags for the AutomationDirect Productivity Series Ethernet Driver?

### [Optimizing Communications](#)

How do I get the best performance from the AutomationDirect Productivity Series Ethernet Driver?

### [Data Types Descriptions](#)

What data types does the driver support?

### [Address Descriptions](#)

How do I reference a data location in a Productivity Series Ethernet device?

### [Error Descriptions](#)

What messages does the AutomationDirect Productivity Series Ethernet Driver produce?

Version 1.050

© 2025 PTC Inc. All Rights Reserved.


## Overview

The AutomationDirect Productivity Series Ethernet Driver provides a reliable way to connect AutomationDirect Productivity Series Ethernet devices to OPC client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications.

## Setup

### Supported Devices

P3-550

 **Important:** The PS-550 device cannot be used as a MODBUS TCP server when DHCP IP addressing is being used.

### Firmware Versions

P3-550: ver.1.0.7.2

### Hardware Setup


It is recommended that users keep the default setting "No exception response for non-existing MODBUS address requests". For more information, refer to the device's programming software under "Project Properties: MODBUS Server Settings."

### Channel and Device Limits

The maximum number of channels supported by this driver is 100. The maximum number of devices supported by this driver is 256 per channel.

### Device ID

**Device ID** specifies the IP address of the device with the extension :#, where # is the server ID.

 **Tip:** The server ID can typically be left as 255 if unknown.

## Channel Properties – General

This server supports the use of multiple simultaneous 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.

Property Groups	<input type="checkbox"/> <b>Identification</b> Name Description Driver	
<b>General</b>	<input type="checkbox"/> <b>Diagnostics</b> Diagnostics Capture      Disable	
Write Optimizations	<input type="checkbox"/> <b>Tag Counts</b> Static Tags      10	
Advanced		

### Identification

**Name:** Specify the 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. The property is required for creating a channel.

• For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

**Description:** Specify user-defined information about this channel.

• Many of these properties, including Description, have an associated system tag.

**Driver:** Specify the protocol / driver for this channel. Specify the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.

• **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. Changes to the properties should not be made once a large client application has been developed. Utilize proper user role and privilege management to prevent operators from changing properties or accessing 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 not available if the driver or operating system does not support diagnostics.

• For more information, refer to *Communication Diagnostics and Statistics Tags* in server help.

### Tag Counts

**Static Tags:** Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

## Channel Properties – Ethernet Communications

Ethernet Communication can be used to communicate with devices.

Property Groups	Ethernet Settings	
General	Network Adapter	Default
<b>Ethernet Communications</b>		
Write Optimizations		
Advanced		

## Ethernet Settings

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

## Channel Properties – Write Optimizations

The server must ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties to meet specific needs or improve application responsiveness.

Property Groups	Write Optimizations	
General	Optimization Method	Write Only Latest Value for All Tags
<b>Write Optimizations</b>	Duty Cycle	10

## 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.

Property Groups	<input type="checkbox"/> <b>Non-Normalized Float Handling</b>	
General	Floating-Point Values	Replace with Zero
Write Optimizations	<input type="checkbox"/> <b>Inter-Device Delay</b>	
<b>Advanced</b>	Inter-Device Delay (ms)	0

**Non-Normalized Float Handling:** 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. Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. 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.

Property Groups	<input type="checkbox"/> Identification	
General	Name	
	Description	
	Channel Assignment	
	Driver	
	Model	
	ID Format	Decimal
	ID	2


### Identification

**Name:** Specify 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:** Specify the user-defined information about this device.

 Many of these properties, including Description, have an associated system tag.


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

**Driver:** Selected protocol driver for this device.

**Model:** Specify the type of device that is associated with this ID. The contents of the drop-down menu depend 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 documentation.

**ID:** Specify 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 format is set by the driver by default. Options include 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.

### Operating Mode

Property Groups	<input type="checkbox"/> Identification <input checked="" type="checkbox"/> Operating Mode	
General	Data Collection	Enable
	Simulated	No



**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:** Place the device into or out of 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 data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

● **Notes:**

1. Updates are not applied until clients disconnect and reconnect.
2. The 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.
3. 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.
4. When a device is simulated, updates may not appear faster than one (1) second in the client.

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

## Tag Counts

Property Groups	[-] Identification	
General	[-] Operating Mode	
	[-] Tag Counts	
	Static Tags	130

**Static Tags:** Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

## 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.

Property Groups	[-] Scan Mode	
General	Scan Mode	Respect Client-Specified Scan Rate ▼
Scan Mode	Initial Updates from Cache	Disable

**Scan Mode:** Specify how tags in the device are scanned for updates sent to subscribing 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 value set as the maximum scan rate. 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 OPC 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.

Property Groups	<input checked="" type="checkbox"/> <b>Communication Timeouts</b>	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	1000
<b>Timing</b>	Attempts Before Timeout	3

### 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:** Specify 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 9999999 milliseconds (167 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.

**Attempts Before Timeout:** Specify how many times the driver issues 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 attempts configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

### Timing

**Inter-Request Delay:** Specify how long the driver waits before sending the next request to the target device after receiving the response to the previous request. 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 turn-around 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.


Property Groups	<input checked="" type="checkbox"/> <b>Timing</b>	
General	Inter-Request Delay (ms)	0
Scan Mode		
<b>Timing</b>		

## Device Properties – Auto-Demotion

The Auto-Demotion properties can temporarily place a device off-scan in the event that a device is not responding. By placing a non-responsive device offline for a specific time period, the driver can continue to optimize its communications with other devices on the same channel. After the time period has been reached, the driver re-attempts to communicate with the non-responsive device. If the device is responsive, the device is placed on-scan; otherwise, it restarts its off-scan time period.

Property Groups	<input checked="" type="checkbox"/> <b>Auto-Demotion</b>	
General	Demote on Failure	Enable <input type="button" value="v"/>
Scan Mode	Timeouts to Demote	3
Timing	Demotion Period (ms)	10000
<b>Auto-Demotion</b>	Discard Requests when Demoted	Disable

**Demote on Failure:** When enabled, the device is automatically taken off-scan until it is responding again.

 **Tip:** Determine when a device is off-scan by monitoring its demoted state using the `_AutoDemoted` system tag.


**Timeouts to Demote:** Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

**Demotion Period:** Indicate how long the device should be placed off-scan when the timeouts value is reached. During this period, no read requests are sent to the device and all data associated with the read requests are set to bad quality. When this period expires, the driver places the device on-scan and allows for another attempt at communications. The valid range is 100 to 3600000 milliseconds. The default is 10000 milliseconds.

**Discard Requests when Demoted:** Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post a message to the Event Log.


## 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.

 **Not all devices and drivers support full automatic tag database generation and not all support the same data types. Consult the data types descriptions or the supported data type lists for each driver for specifics.**

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.*

Property Groups	<b>Tag Generation</b>	
General	On Device Startup	Do Not Generate on Startup
Timing	On Duplicate Tag	Delete on Create
Auto-Demotion	Parent Group	
<b>Tag Generation</b>	Allow Automatically Generated Subgroups	Enable
Communications	Create	Create tags
Redundancy		

**On Property Change:** If the device supports automatic tag generation when certain properties change, the **On Property Change** option is shown. It is set to **Yes** by default, but it can be set to **No** to control over when tag generation is performed. In this case, the **Create tags** action must be manually invoked to perform tag generation. To invoke via the Configuration API service, access `/config/v1/project/channels/{name}/devices/{name}/services/TagGeneration`.

**On Device Startup:** Specify 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 – Communication

---

Communication	
Port Number	502

**Port Number:** This property is used to specify the port number. The default setting is 502.

● **Note:** The default communication protocol is MODBUS® TCP/IP.

## Device Properties – Data Handling

---

Data Handling	
First Word High	Enable

**First Word High:** In an AutomationDirect Productivity Series Ethernet device, the addresses of two consecutive registers are used for 32-bit data types. When this option is enabled, the driver will assume the first word is high for the 32-bit value. When this option is disabled, the driver will assume that the first word is low for the 32-bit value. The default setting is enabled.

## Device Properties – Tag Import File Path

---

Tag Import File Path	
Tag Import File	*.csv

**Tag Import File:** This property specifies the exact location of the tag import file. The "\*.csv" file must be created in the Productivity Suite Programming Software. The tag import file will be used by the Automatic Tag Database Generation feature to create the tag database. All tags will be imported and expanded according to their respective data types.

● **Note:** For more information, refer to [Automatic Tag Database Generation](#).

## Device Properties – Redundancy

Property Groups General Scan Mode Timing Auto-Demotion <b>Redundancy</b>	<b>Redundancy</b>	
	Secondary Path	Channel Device 1 ...
	Operating Mode	Switch On Failure
	Monitor Item	
	Monitor Interval (s)	300
	Return to Primary ASAP	Yes

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

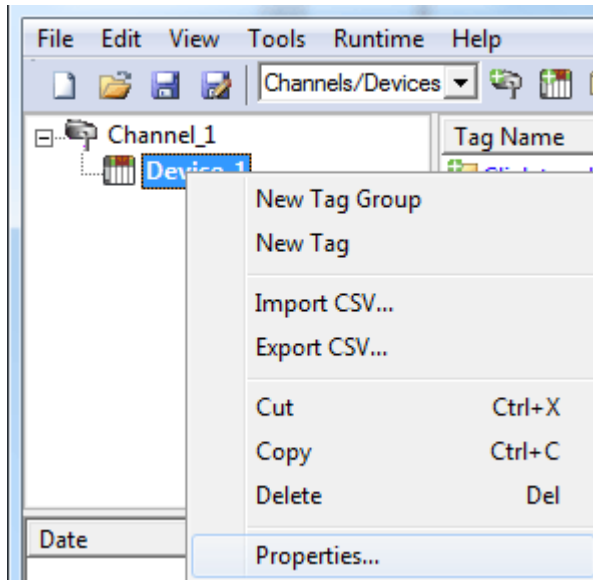
• Consult the website, a sales representative, or the [user manual](#) for more information.

## Automatic Tag Database Generation

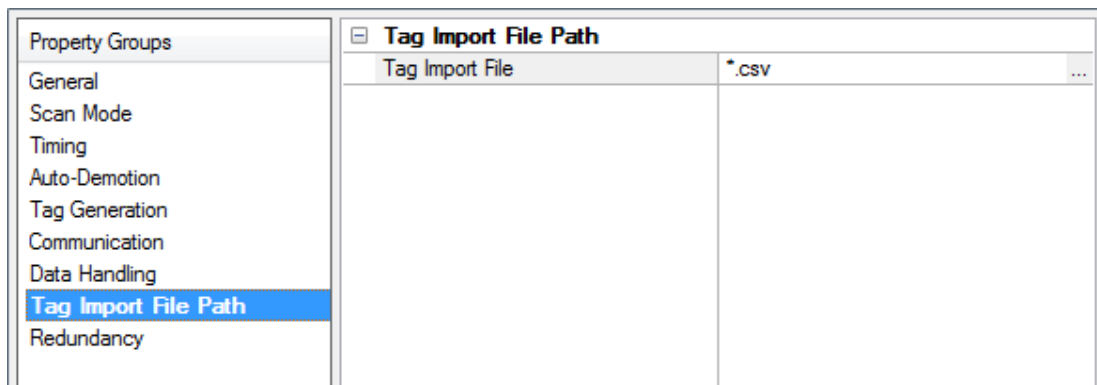
The AutomationDirect Productivity Series Ethernet Driver supports the OPC server's Automatic Tag Database Generation feature, which allows the driver to automatically create tags that access data points used in a device configuration. The OPC server uses the tag import file to create the tag database. The tag import file (\*.csv) must be created in the Productivity Suite Programming Software. For more information, refer to [Tag Import File Path](#).

For information on using automatic tag database generation, refer to the following instructions.

1. In the OPC server project, right-click on the device and then select **Properties**.



2. Next, open the **Tag Import File Path** property group.



3. Click **Browse** to locate the folder containing the import file (\*.csv).

● **Note:** Productivity Software Suite Versions 1.4 to 1.9 add a field to exported .csv files that is not supported and causes tag import to fail. To use these files, delete the last field or column. The modified .csv file can be imported. Starting with version 1.10 2 files are created with \_basic and \_extended appended to the file name. The \_basic file is correctly formatted to be used for Auto Tag Generation.

4. Next, select the import file and then click **OK** to start the import and tag generation process.
5. The OPC server's Event Log will show when the tag generation process started, any errors that occurred during processing, and when the process completed.

● **Note:** For more information, refer "Automatic Tag Database Generation" in the server help documentation.

### See Also:

[Exporting a CSV File from Productivity Suite](#)

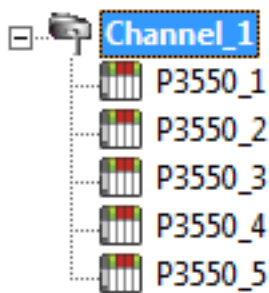




## Optimizing Communications

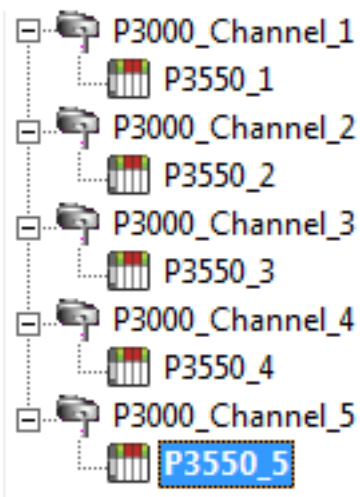
The AutomationDirect Productivity Series Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the AutomationDirect Productivity Series Ethernet Driver is fast, there are a couple of guidelines that can be used to optimize the application and gain maximum performance.

This server refers to communications protocols like AutomationDirect Productivity Series Ethernet Driver as a channel. Each channel defined in the application represents a separate path of execution in the server. Once a channel has been defined, a series of devices must then be defined under that channel. Each of these devices represents a single controller from which data will be collected. While this approach to defining the application will provide a high level of performance, it won't take full advantage of the AutomationDirect Productivity Series Ethernet Driver or the network. An example of how the application may appear when configured using a single channel is shown below.



Each device appears under a single channel. In this configuration, the driver must move from one device to the next as quickly as possible to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the AutomationDirect Productivity Series Ethernet Driver could only define one single channel, then the example shown above would be the only option available; however, the AutomationDirect Productivity Series Ethernet Driver can define up to 256 devices per channel. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device can be defined under its own channel. In this configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 100 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 100 devices. While fewer devices may be ideal, the application will benefit from additional channels. Although spreading the device load across all channels will cause the server to move from device to device again, it can now do so with far less devices to process on a single channel.

• See Also: [Data Handling](#)

## 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
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 registers as a floating point value by making the second register the high word and the first register the low word.
String	Null-terminated ASCII string Support includes HiLo LoHi byte order selection.

## Address Descriptions

The default data types are shown in **bold**.

### System Index Addresses

Device Type	Range	Data Type	Access
Analog Input Float 32	AIF32-00.00.00.01-AIF32-99.05.15.32	<b>Float</b>	Read Only
Analog Input Signed 32	AIS32-00.00.00.01-AIS32-99.05.15.32	<b>Long</b>	Read Only
Analog Output Float 32	AOF32-00.00.00.01-AOF32-99.05.15.32	<b>Float</b>	Read/Write
Analog Output Signed 32	AOS32-00.00.00.01-AOS32-99.05.15.32	<b>Long</b>	Read/Write
Discrete Inputs	DI-00.00.00.01-DI-99.05.15.128	<b>Boolean</b>	Read Only
Discrete Outputs	DO-00.00.00.01-DO-99.05.15.128	<b>Boolean</b>	Read/Write
I/O Module Status Bits	MST-00.00.00.01-MST-99.05.15.128	<b>Boolean</b>	Read Only

### Continuous Addresses

Device Type	Range	Data Type	Access
Internal BCD 16	BCD16-000001-BCD16-999999	<b>BCD</b>	Read/Write
Internal BCD 32	BCD32-000001-BCD32-999999	<b>LBCD</b>	Read/Write
Internal Bits	C-000001-C-999999	<b>Boolean</b>	Read/Write
Internal Float 32	F32-000001-F32-999999	<b>Float</b>	Read/Write
Internal Unsigned 8	US8-000001-US8-999999	<b>Byte</b>	Read/Write
Internal Signed 16	S16-000001-S16-999999	<b>Short</b>	Read/Write
Internal Signed 32	S32-000001-S32-999999	<b>Long</b>	Read/Write
String	STR-000001:1-STR-999999:128	<b>String</b>	Read/Write
System String	SSTR-000001:1-SSTR-999999:50	<b>String</b>	Read Only
System Read Only Bit	SBR-000001-SBR-999999	<b>Boolean</b>	Read Only
System Read/Write Bit	SBRW-000001-SBRW-999999	<b>Boolean</b>	Read/Write
System Read Only Word	SWR-000001-SWR-999999	<b>Word</b>	Read Only
System Read/Write Word	SWRW-000001-SWRW-999999	<b>Word</b>	Read/Write
Internal Unsigned 16	US16-000001-US16-999999	<b>Word</b>	Read/Write

### One-Dimensional Array Elements

Device Type	Range	Data Type	Access
1 Dimensional Internal BCD 16	AR1BCD16-00001(1)-AR1BCD16-999999(65535)	<b>BCD</b>	Read/Write
1 Dimensional Internal BCD 32	AR1BCD32-00001(1)-AR1BCD32-999999 (65535)	<b>LBCD</b>	Read/Write
1 Dimensional Internal Bits	AR1C-000001(1)-AR1C-999999(65535)	<b>Boolean</b>	Read/Write
1 Dimensional Internal Float 32	AR1F32-00001(1)-AR1F32-999999(65535)	<b>Float</b>	Read/Write
1 Dimensional Internal Unsigned 8	AR1US8-00001(1)-AR1US8-999999(65535)	<b>Byte</b>	Read/Write
1 Dimensional Signed 16	AR1S16-00001(1)-AR1S16-999999(65535)	<b>Short</b>	Read/Write
1 Dimensional Signed 32	AR1S32-00001(1)-AR1S32-999999(65535)	<b>Long</b>	Read/Write

Device Type	Range	Data Type	Access
1 Dimensional String	AR1STR-00001:1(1)-AR1STR-999999:128(65535)	String	Read/Write
1 Dimensional Internal Unsigned 16	AR1US16-00001(1)-AR1US16-999999(65535)	Word	Read/Write

### Two-Dimensional Array Elements

Device Type	Range	Data Type	Access
2 Dimensional Internal BCD 16	AR2BCD16-00001(1),(1)-AR2BCD16-999999 (65535),(65535)	BCD	Read/Write
2 Dimensional Internal BCD 32	AR2BCD32-00001(1),(1)-AR2BCD32-999999 (65535),(65535)	LBCD	Read/Write
2 Dimensional Internal Bits	AR2C-000001(1),(1)-AR2C-999999 (65535),(65535)	Boolean	Read/Write
2 Dimensional Internal Float 32	AR2F32-00001(1),(1)-AR2F32-999999 (65535), (65535)	Float	Read/Write
2 Dimensional Unsigned Hex 8	AR2US8-00001(1),(1)-AR2US8-999999 (65535), (65535)	Byte	Read/Write
2 Dimensional Signed 16	AR2S16-00001(1),(1)-AR2S16-999999 (65535), (65535)	Short	Read/Write
2 Dimensional Signed 32	AR2S32-00001(1),(1)-AR2S32-999999 (65535), (65535)	Long	Read/Write
2 Dimensional String	AR2STR-00001:1(1),(1)-AR2STR-999999:128 (65535),(65535)	String	Read/Write
2 Dimensional Internal Unsigned 16	AR2US16-00001(1),(1)-AR2US16-999999 (65535), (65535)	Word	Read/Write

• See Also: [Ordering of Array Data](#)

## Ordering of Array Data

---

### 1. Dimensional Arrays - array [dim1]

1 dimensional array data is passed to and from the controller in ascending order:

```
for (dim1=0; dim1<dim1_max; dim1++)
```

**Example:** 3 element array

```
array [0]
array [1]
array [2]
```

### 2. Dimensional Arrays - array [dim1, dim2]

2 dimensional array data is passed to and from the controller in ascending order:

```
for (dim1=0; dim1<dim1_max; dim1++)
for (dim2=0; dim2<dim2_max; dim2++)
```

**Example:** 3X3 element array

```
array [0, 0]
array [0, 1]
array [0, 2]
array [1, 0]
array [1, 1]
array [1, 2]
array [2, 0]
array [2, 1]
array [2, 2]
```

## Address Formats

---

### Tag Names

The default tag names are assigned when hardware is added. Tag names for I/O modules generally follow the AAAAA-B.C.D.E format, where:

- **AAAAA:** Category
- **B:** Base Group Number
- **C:** Base Number
- **D:** Slot Number
- **E:** Point or Channel Number

● **Note:** For example, "DI-0.1.1.1" is Discrete Input P3-550.

### Category

The category of data may also include the data type. Descriptions of the categories are as follows:

- **Discrete I/O points:** "DI" is the Discrete Input Point. "DO" is the Discrete Output Point .
- **Boolean:** This category includes data items other than I/O points. For example, a blown fuse or out-of-range error bits. "MST" is the Module Status.
- **Analog points:** This category uses five digits. The first two digits designate the type of point and the last three digits describe the data type being used. "AIxxx" is the Analog Input Point. "AOxxx" is the Analog Output Point. "xxF32" is the Floating Point 32-bit data type. "xxS32" is the Signed 32-bit data type.

### Base Group Number

The base group number is where the module resides. For the P3-550, the base group is group 0.

### Base Number

The base number within the group is where the module resides. For the P3-550 or P3-RS, the base is base 1.

### Slot Number

The slot number in the base is where the module resides. The slot to the right of the CPU is slot 1.

**Point or Channel Number**

If the tag name is for an I/O point of a module, this digit will be the channel number. If the tag name is for a status indication, this digit will differentiate them.

For example, a P3-16TR has one fuse on both commons and a blown fuse indicator bit for each. Their default tag names are "MST-x.x.x.1" and "MST-x.x.x.2". "MST-x.x.x.1" is the blown fuse indicator for the fuse on the common of channels 1 through 8.

## Error Descriptions

---

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

### Automatic Tag Database Generation Error Messages

[Tag name <tag name> encountered validation error and will not be imported.](#)

[Unable to generate a tag database for device <device name>. Reason: Auto tag generation cancelled.](#)

[Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt.](#)

[Unable to generate a tag database for device <device name>. Reason: Import file not found.](#)

[Unable to generate a tag database for device <device name>. Reason: Low memory resources.](#)

### Driver Error Messages

[Unable to bind to adapter: <network adapter>. Connect failed. Winsock Err # <Error number>.](#)

[Winsock initialization failed \(OS Error = <error code>\).](#)

[Winsock shut down failed \(OS Error = <error code>\).](#)

[Winsock V1.1 or higher must be installed to use the Productivity Series Ethernet device driver.](#)

### Read Errors

[Cannot read <tag count> items starting at tag <tag address>: address does not exist in device <device name>.](#)

[Cannot read <tag count> items starting at tag <tag address>: device <device name> returned error code <error code>.](#)

[Cannot read <tag count> items starting at tag <tag address>: error receiving response frame from device <device name>.](#)

[Cannot read <tag count> items starting at tag <tag address>: System ID <System ID> does not exist in device <device name>.](#)

[Cannot read <tag count> items starting at tag <tag address>: value is invalid for data type <data type> in device <device name>.](#)

[Cannot read tag <tag address>: address does not exist in device <device name>.](#)

[Cannot read tag <tag address>: device <device name> returned with error code <error code>.](#)

[Cannot read tag <tag address>: error receiving response frame from device <device name>.](#)

[Cannot read tag <tag address>: System ID <System ID> does not exist in device <device name>.](#)

[Cannot read tag <tag address>: value is invalid for data type <data type> in device <device name>.](#)

### Write Errors

[Cannot write to tag <tag address>: address does not exist in device <device name>.](#)

[Cannot write to tag <tag address>: device <device name> returned with error code <error code>.](#)

[Cannot write to tag <tag address>: error receiving response frame from device <device name>.](#)

[Cannot write to tag <tag address>: System ID <System ID> does not exist in device <device name>.](#)

[Cannot write to tag <tag address>: value is invalid for data type <data type> in device <device name>.](#)

### See Also:

[Modbus Exception Codes](#)

**Tag name <tag name> encountered validation error and will not be imported.**

---

### Error Type:

Warning

### Possible Cause:

The tag import file contains tag(s) that have invalid character(s). This error message will be returned for each tag that contains an invalid character.

**Solution:**

Remove the invalid character(s).

---

**Unable to generate a tag database for device <device name>. Reason: Auto tag generation cancelled.**

---

**Error Type:**

Warning

**Possible Cause:**

The Automatic Tag Generation process was cancelled.

**Solution:**

Retry the Automatic Tag Generation process.

**See Also:**

[Automatic Tag Database Generation](#)

---

**Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The Tag Import File is a corrupt project file.
2. The .csv file was created with Productivity Software Suite Versions 1.4 to 1.9, which adds a field that is not supported for import.

**Solution:**

1. In the OPC server project, right-click on the device and then select **Properties** from the context menu. Next, click on the **Tag Import File Path** tab. Select a valid, properly formatted **Productivity Suite Programming Software Import File** or produce a new import file by retrying the tag export process in the application.
2. Delete the last field or column from the Versions 1.4 to 1.9 .csv file and try the import again.

**See Also:**

[Automatic Tag Database Generation](#)  
[Exporting a CSV File from Productivity Suite](#)

---

**Unable to generate a tag database for device <device name>. Reason: Import file not found.**

---

**Error Type:**

Warning

**Possible Cause:**

The Tag Import File could not be found.

**Solution:**

In the OPC server project, right-click on the device and then select **Properties** from the context menu. Next, click on the **Tag Import File Path** tab. Select a valid, properly formatted **Productivity Suite Programming Software Import File**, or produce a new import file by retrying the tag export process in the application.



### Unable to generate a tag database for device <device name>. Reason: Low memory resources.

#### Error Type:

Warning

#### Possible Cause:

Memory required for database generation could not be allocated. The process is cancelled.

#### Solution:

Close unused applications and/or increase the amount of virtual memory and try again.

### Unable to bind to adapter: <network adapter>. Connect failed. Winsock Err# <error number>.

#### Error Type:

Fatal

#### Possible Cause:

1. The operating system could not find an unused port to use for communication with this device.
2. Network system failure, such as Winsock or network adapter failure.
3. 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 shut them down.

### Winsock initialization failed (OS Error = <OS error code>).

#### 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 shut down failed (OS Error = <OS error code>).

#### Error Type:

Fatal

OS Error	Possible Solution
10036	The network subsystem is still busy with unfinished processing. Wait a few seconds and restart the driver.
10050	The network subsystem has failed. Refer to the Network Administrator.
10093	The network subsystem was not initialized before the shutdown was attempted. Wait a few seconds and try again.

---

**Winsock V1.1 or higher must be installed to use the Productivity Series 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.

---

**Cannot read <tag count> items starting at tag <tag address>: address does not exist in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x02 (illegal data address) for a blocked Read transaction.

**Solution:**

Replace the current data address with one within the valid range (as listed in the device protocol).

---

**Cannot read <tag count> items starting at tag <tag address>: device <device name> returned error code <error code>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC has returned an error code.

**Solution:**

Refer to the list of error code descriptions located in the PLC's manual.

---

**Cannot read <tag count> items starting at tag <tag address>: error receiving response frame from device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The response frame received from the PLC contains an error.

**Solution:**

Check the connection to the PLC and then resend the data.

---

**Cannot read <tag count> items starting at tag <tag address>: System ID <System ID> does not exist in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x01 (illegal function) for a blocked Read transaction.

**Solution:**

Replace the current System ID with a valid one (as listed in the device protocol).

**Cannot read <tag count> items starting at tag <tag address>: value is invalid for data type <data type> in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x03 (illegal data value) for a blocked Read transaction.

**Solution:**

Replace the current data with valid values (depending on the data type).

**Cannot read tag <tag address>: address does not exist in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x02 (illegal data address) for an unblocked Read transaction.

**Solution:**

Replace the current data address with one within the valid range (as listed in the device protocol).

**Cannot read tag <tag address>: device <device name> returned with error code <error code>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC has returned an error code.

**Solution:**

Refer to the list of error code descriptions located in the PLC's manual.

**Cannot read tag <tag address>: error receiving response frame from device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The response frame received from the PLC contains an error.

**Solution:**

Check the connection to the PLC and then resend the data.

**Cannot read tag <tag address>: System ID <System ID> does not exist in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x01 (illegal function) for an unblocked Read transaction.

**Solution:**

Replace the current System ID with a valid one (as listed in the device protocol).

**Cannot read tag <tag address>: value is invalid for data type <data type> in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x03 (illegal data value) for an unblocked Read transaction.

**Solution:**

Replace the current data with valid values (depending on the data type).

**Cannot write to tag <tag address>: address does not exist in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x02 (illegal data address) for a Write transaction.

**Solution:**

Replace the current data address with one within the valid range (as listed in the device protocol).

**Cannot write to tag <tag address>: device <device name> returned with error code <error code>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC has returned an error code.

**Solution:**

Refer to the list of error code descriptions located in the PLC's manual.

**Cannot write to tag <tag address>: error receiving response frame from device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The response frame received from the PLC contains an error.

**Solution:**

Check the connection to the PLC and then resend the data.

**Cannot write to tag <tag address>: System ID <System ID> does not exist in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x01 (illegal function) for a Write transaction.

**Solution:**

Replace the current System ID with a valid one (as listed in the device protocol).

**Cannot write to tag <tag address>: value is invalid for data type <data type> in device <device name>.**

---

**Error Type:**

Warning

**Possible Cause:**

The PLC returned an error code of 0x03 (illegal data value) for a Write transaction.


**Solution:**

Replace the current data with valid values (depending on the data type).

## Modbus Exception Codes

The following data is from Modbus Application Protocol Specifications documentation.

Code Dec/Hex	Name	Meaning
01/0x01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server is in the wrong state to process a request of this type, for example, because it is unconfigured and is being asked to return register values.
02/0x02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed. A request with offset 96 and length 5 generates exception 02.
03/0x03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.
04/0x04	SERVER DEVICE FAILURE	An unrecoverable error occurred while the server was attempting to perform the requested action.
05/0x05	ACKNOWLEDGE	The server has accepted the request and is processing it, but a long duration of time is required to do so. This response is returned to prevent a timeout error from occurring in the client. The client can next issue a Poll Program Complete message to determine if processing is completed.
06/0x06	SERVER DEVICE BUSY	The server is engaged in processing a long-duration program command. The client should retransmit the message later when the server is free.
07/0x07	NEGATIVE ACKNOWLEDGE	The server cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 decimal. The client should request diagnostic or error information from the server.
08/0x08	MEMORY PARITY ERROR	The server attempted to read extended memory, but detected a parity error in the memory. The client can retry the request, but service may be required on the server device.
10/0x0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. This usually means that the gateway is misconfigured or overloaded.
11/0x0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways indicates that no response was obtained from the target device. This usually means that the device is not present on the network.

 **Note:** For this driver, the terms server and unsolicited are used interchangeably.

## Appendix: Exporting a CSV File from Productivity Suite

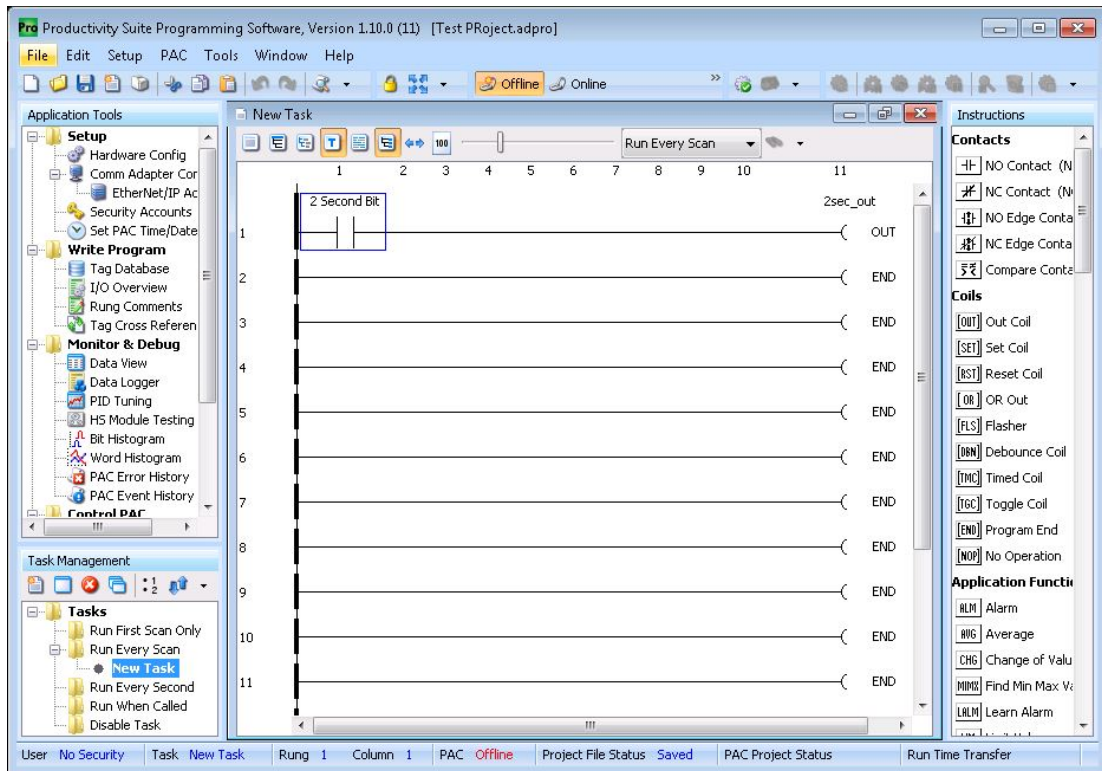
Tag export from the Productivity Suite software produces a CSV (Comma Separated Value) file. Prior to Productivity Suite Version 1.10.0.11, it created a single CSV file. Version v1.10.0.11 and newer create two CSV files: basic and extended. Use the basic file for Automatic Tag Generation.

### Example

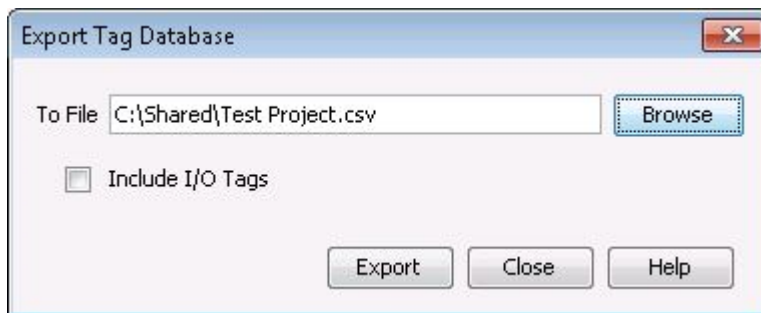
TestProject\_basic.csv  
TestProject\_extended.csv

To export the tag database:

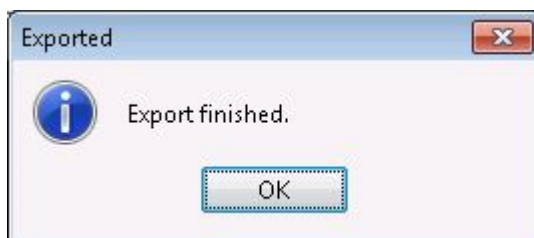
1. In the Productivity Suite main menu, select **File | Export | Tags...**



2. Browse to and select the location in which to save the exported tag CSV file.



3. Click **Export** to create the .csv export files.



4. Click **OK**.

#### See Also:

[Automatic Tag Database Generation](#)

# Index

## A

Address Descriptions 19  
Address Formats 21  
Allow Sub Groups 12  
Attempts Before Timeout 10  
Auto-Demotion 11  
Automatic Tag Database Generation 15

## C

Cannot read <tag count> items starting at tag <tag address>: address does not exist in device <device name>. 26  
Cannot read <tag count> items starting at tag <tag address>: device <device name> returned error code <error code>. 26  
Cannot read <tag count> items starting at tag <tag address>: error receiving response frame from device <device name>. 26-27  
Cannot read <tag count> items starting at tag <tag address>: value is invalid for data type <data type> in device <device name>. 27  
Cannot read tag <tag address>: address does not exist in device <device name>. 27  
Cannot read tag <tag address>: device <device name> returned with error code <error code>. 27  
Cannot read tag <tag address>: System ID <system ID> does not exist in device <device name>. 27  
Cannot read tag <tag address>: value is invalid for data type <data type> in device <device name>. 28  
Cannot write to tag <tag address>: address does not exist in device <device name>. 28  
Cannot write to tag <tag address>: device <device name> returned with error code <error code>. 28  
Cannot write to tag <tag address>: error receiving response frame from device <device name>. 28  
Cannot write to tag <tag address>: System ID <system ID> does not exist in device <device name>. 28  
Cannot write to tag <tag address>: system ID <system ID> does not exist in device <device name>. 26  
Cannot write to tag <tag address>: value is invalid for data type <data type> in device <device name>. 29  
Channel Assignment 8  
Channel Properties – Advanced 6  
Channel Properties – Ethernet Communications 5  
Channel Properties – General 5  
Channel Properties – Write Optimizations 6  
Communication 13  
Communications Timeouts 10  
Connect Timeout 10  
Create 13

## D

Data Collection 9



- Data Handling 13
- Data Types Description 18
- Delete 12
- Demote on Failure 11
- Demotion Period 11
- Device ID 4
- Device Properties – Auto-Demotion 11
- Device Properties – General 8
- Device Properties – Redundancy 14
- Device Properties – Tag Generation 11
- Device Properties – Timing 10
- Diagnostics 5
- Discard Requests when Demoted 11
- Do Not Scan, Demand Poll Only 10
- Driver 8
- Duty Cycle 6

## **E**

- Error Descriptions 23
- Ethernet Settings 6
- Exporting a CSV File from Productivity Suite 30

## **G**

- General 8
- Generate 12

## **I**

- ID 8
- Identification 5, 8
- Initial Updates from Cache 10
- Inter-Device Delay 7

## **M**

- Modbus Exception Codes 30
- Model 8

## N

Name 8

Network Adapter 6

Non-Normalized Float Handling 7

## O

On Device Startup 12

On Duplicate Tag 12

On Property Change 12

Operating Mode 8

Optimization Method 6

Optimizing AutomationDirect Productivity Series Ethernet Communications 17

Ordering of Array Data 21

Overview 4

Overwrite 12

## P

Parent Group 12

## R

Redundancy 14

Replace with Zero 7

Request Timeout 10

Respect Tag-Specified Scan Rate 10

## S

Scan Mode 9

Setup 4

Simulated 9

## T

Tag Counts 5, 9

Tag Generation 11

Tag Import File Path 13

Tag name <tag name> encountered validation error and will not be imported. 23

Timeouts to Demote 11

Timing 10

## U

Unable to bind to adapter:<network adapter>. Connect failed. Winsock Err# <error number>. 25

Unable to generate a tag database for device <device name>. Reason: Auto tag generation cancelled. 24

Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt. 24

Unable to generate a tag database for device <device name>. Reason: Import file not found. 24

Unable to generate a tag database for device <device name>. Reason: Low memory resources. 25

Unmodified 7

## W

Winsock initialization failed (OS Error = <OS error code>). 25

Winsock shut down failed (OS Error = <OS error code>). 25

Winsock V1.1 or higher must be installed to use the Productivity Series Ethernet device driver. 26

Write All Values for All Tags 6

Write Only Latest Value for All Tags 6

Write Only Latest Value for Non-Boolean Tags 6