Opto 22 Ethernet Driver

Table of Contents

Opto 22 Ethernet Driver

Help version 1.023

CONTENTS

Overview

The Opto 22 Ethernet Driver provides an easy and reliable way to connect Opto 22 Ethernet devices to OPC Client applications, including HMI, SCADA, Historian, MES, ERP and countless custom applications.

Setup

Supported Devices

Supported devices include the following:

SNAP Industrial Controllers
SNAP PAC S-Series
SNAP PAC R-Series
SNAP-LCE

SNAP Brains
SNAP PAC EB-Series
SNAP Ultimate I/O
SNAP Ethernet I/O
SNAP Simple I/O

E1 and E2 Brain Boards

Communications Protocols

MMIO over Ethernet TCP/IP or UDP
CONT over Ethernet TCP/IP

Notes:

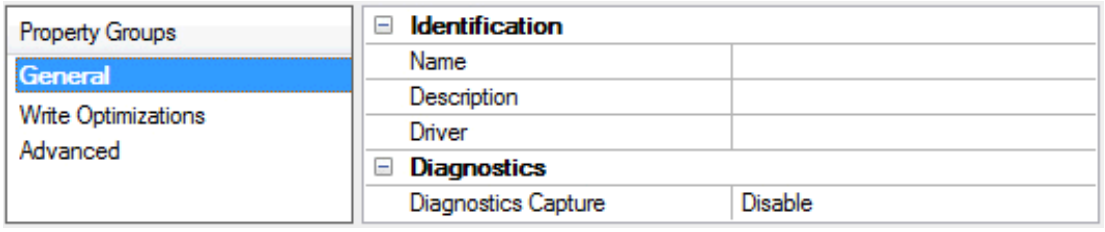
1. This driver requires Winsock V1.1 or higher.
2. Firmware version 8.0 or higher is required for some features. For more information, refer to **Address Descriptions**.

Maximum Number of Channels and Devices

The maximum number of channels is 256. The maximum number of devices is 65535 per channel.

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.

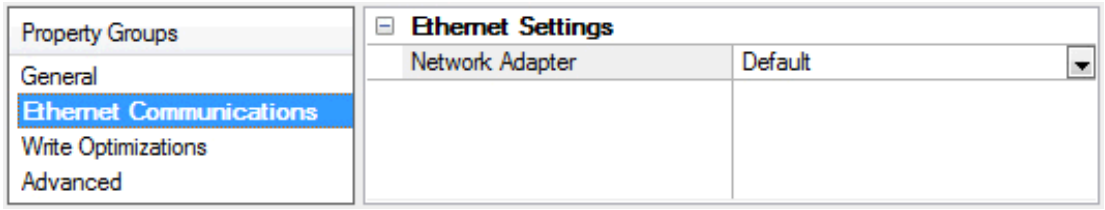
Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties.
Note: With the server's online full-time operation, these properties can be changed at any time. This includes 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 not available 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.

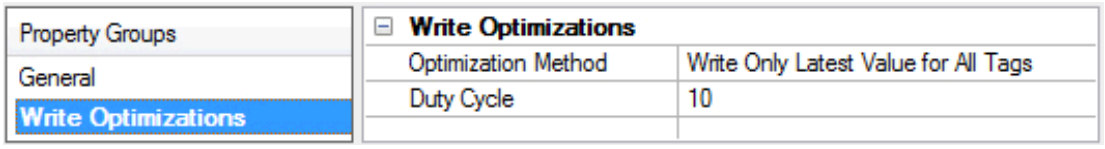


Ethernet Settings

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 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:

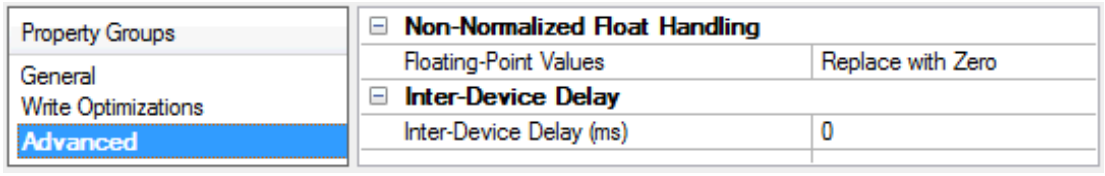
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing 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: 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 not available 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.

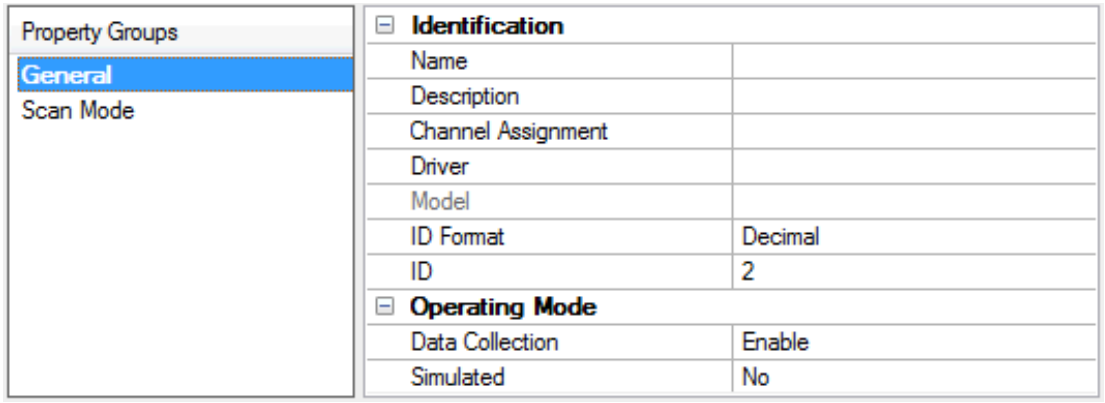
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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 station / node / identity / address. The type of ID entered depends on the communications driver being used. For many 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. 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

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.

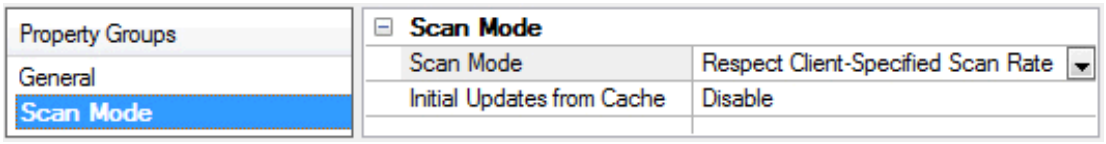
Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client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



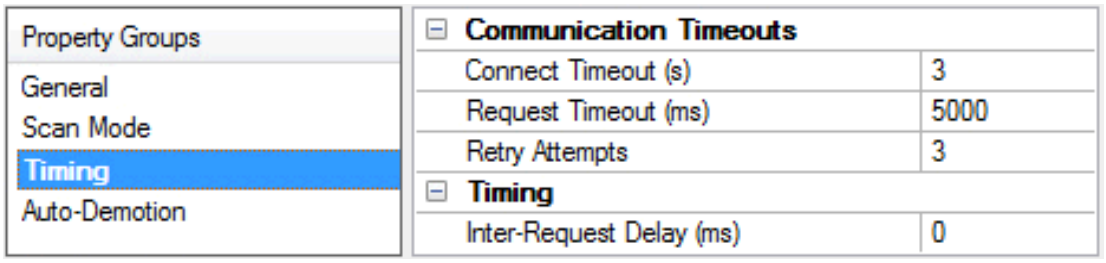
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **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 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.

Attempts Before Timeout: This property specifies 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

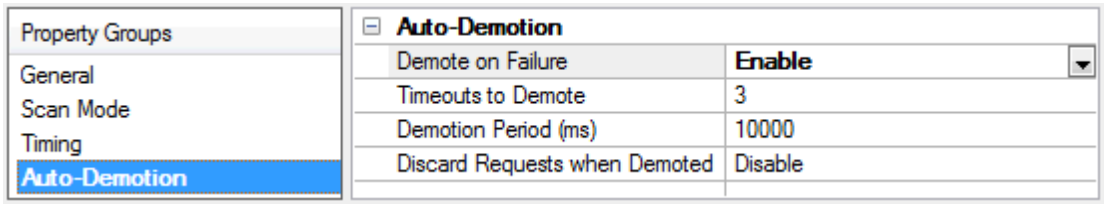
Inter-Request Delay: This property specifies how long the driver waits before sending the next request to the target device. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow 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 — 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.



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

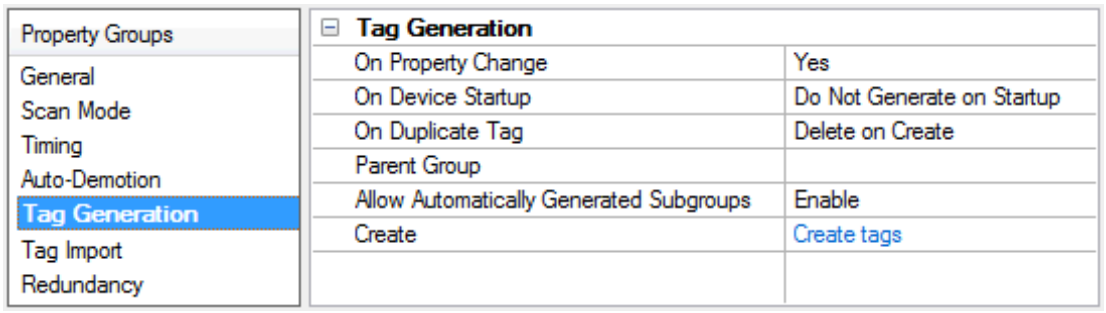
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the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



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.

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 - Communications Parameters

I/O Unit (MMIO)	
I/O Unit IP Protocol	TCP/IP
I/O Unit Port Number	2001
Control Engine (CONT)	
Control Engine Port Number	22001

I/O Unit (MMIO)

The I/O Unit properties are used to access memory mapped data using the MMIO protocol. Descriptions of the properties are as follows:

- **I/O Unit IP Protocol** Options include TCP/IP or UDP. The default setting is TCP/IP.

  Note: Although TCP is a more reliable protocol, it requires more network overhead. Devices will accept a limited number of TCP connections. Once the limit is reached, the driver will not be able to communicate with the device using TCP.
- **I/O Unit Port Number** This property specifies the port number that the device has been configured to use for MMIO communications. The default setting is 2001.

Control Engine (CONT)

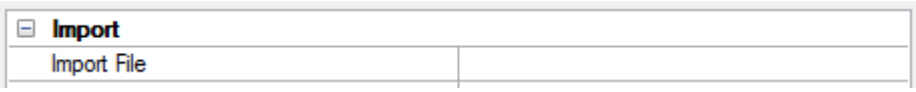
The Control Engine properties are used to access strategy variables using the CONT protocol. Description of the property is as follows:

- **Control Engine Port Number** This property specifies the port number that the device has been configured to use for CONT communications. The default setting is 22001.

Note: The PAC Control (or other similar tool) can be used to configure devices for MMIO and CONT communications. For instructions and additional information on setting up each device, refer to the Opto 22 User Guides.

Device Properties - Import

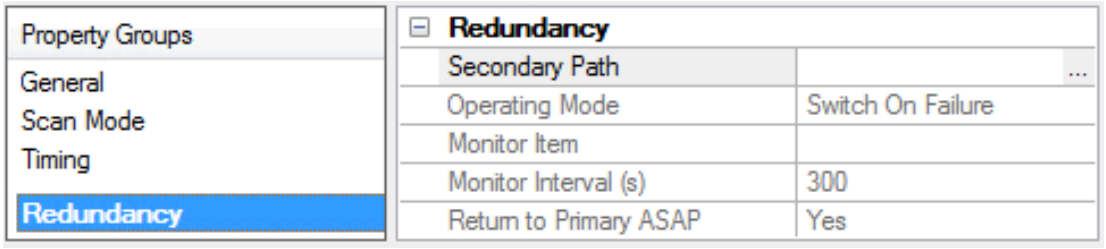
The Import property is used to specify the browser database file (*.bdb) from which tags will be imported. To browse the file system, click the Browse button.



A browser database file can be created from one or more device configuration files using the Opto Browser Configurator tool. The driver will generate tags for all supported Item IDs in the file with a location IP address that matches the driver's Device ID. The Database Creation options located in Device Properties can be used to initiate the import process. For more information on the tag import procedure, refer to **Automatic Tag Database Generation**.

See Also: **Address Descriptions** and **Device Setup**.

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.

Data Types Description

The Opto 22 Ethernet Driver supports the following data types.

Data Type	Description
Boolean	Single 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
Float	32-bit floating point value
String	Null terminated ASCII string

Address Descriptions

In order to address a data location on an Opto 22 Ethernet device, follow the information and instructions below.

Prefix

All addresses may include the following prefix: *[Device|Protocol|Location]*.

Where:
Device = "MMIO" or "CONT"
Protocol = "ip"
Location = "tcp:<IP Address>:<Port>"

Note: This prefix is optional. The ability to recognize the prefix can be useful when migrating OPC client applications from other servers that use the full Opto 22 Item ID syntax. Item IDs in these applications must be modified to either replace the prefix with the appropriate path (<*channel name*>.<*device name*>) or to prepend the prefix with the path. The modified Item IDs will be recognized as valid dynamic tag addresses. For a description of dynamic tags, refer to the OPC Server Help documentation.

Memory Mapped Data (MMIO Protocol)

Basic Address Syntax
ItemName[p]

For I/O:
p = (Module) (PointsPerModule) + Point

where:
Module = zero-based module number.
Point = zero-based point within module.
PointsPerModule = 64 for xxx_4096 ItemNames (see below), and 4 for all others.

For Alarm Addresses
p = zero-based alarm index.

For Scratch Pad Addresses
p = zero-based scratch pad value index.

For PID Addresses
p = zero based PID index.

Array Address Syntax

ItemName[pStart-pEnd]

pStart = start I/O point or scratch pad value index.
pEnd = end I/O point or scratch pad value index.

Note: Arrays are limited to 1024 bytes.

HDD (High Density Digital) Address Syntax

ItemName(Module)[Point]

Module = zero-base module number.
Point = zero-based point within module.

HDD (High Density Digital) Bank Address Syntax

ItemName(Module)

Analog Point

Item Name	Description	Range	Data Type	Access	Arrays
EU[p]	Engineering units value	0-63	Float	Read/Write	Yes
COUNTS[p]	Counts	0-63	Float	Read/Write	Yes
MIN[p]	Minimum value	0-63	Float	Read	Yes
MAX[p]	Maximum value	0-63	Float	Read	Yes
MIN_READCLEAR[p]	Read then clear minimum value*	0-63	Float	Read	Yes
MAX_READCLEAR[p]	Read then clear maximum value*	0-63	Float	Read	Yes
EU_4096[p]	Engineering units value**	0-4095	Float	Read/Write	No
COUNTS_4096[p]	Counts**	0-4095	Float	Read/Write	No
MIN_4096[p]	Minimum value**	0-4095	Float	Read	No
MAX_4096[p]	Maximum value**	0-4095	Float	Read	No
MIN_READCLEAR_4096[p]	Read then clear minimum value*,**	0-4095	Float	Read	No
MAX_READCLEAR_4096[p]	Read then clear maximum value*,**	0-4095	Float	Read	No

*The device clears these values after each read. OPC clients should add these items as inactive so the driver does not poll them.

**These items require firmware 8.0 or higher.

Digital Point (4-Channel Digital Modules)

Item Name	Description	Range	Data Type	Access	Arrays
STATE[p]	On/Off state	0-63	Boolean	Read/Write	No
ONLATCH[p]	On-latch state	0-63	Boolean	Read	No
OFFLATCH[p]	Off-latch state	0-63	Boolean	Read	No
ACTIVECOUNTER[p]	Active state of counter*	0-63	Boolean	Read/Write	No
COUNTERDATA[p]	Counter value*	0-63	**Long**, DWord	Read	Yes
ONLATCH_READCLEAR[p]	Read then clear On-latch state**	0-63	Boolean	Read	Yes
OFFLATCH_READCLEAR[p]	Read then clear Off-latch state**	0-63	Boolean	Read	Yes
COUNTERDATA_READCLEAR[p]	Read then clear counter value**	0-63	**Long**, DWord	Read	Yes

*To use a digital point as a counter, it must be configured as a counter and the counter must be active.

**The device clears these values after each read. OPC clients should add these items as inactive so the driver does not poll them.

High Density Digital (HDD) Point

Item Name	Description	Range	Data Type	Access	Arrays
HDD_STATE(m)[p]	On/Off state	m: 0-15 p: 0-31	Boolean	Read/Write	No
HDD_ONLATCH(m)[p]	On-latch state	m: 0-15 p: 0-31	Boolean	Read	No
HDD_OFFLATCH(m)[p]	Off-latch state	m: 0-15 p: 0-31	Boolean	Read	No
HDD_ONLATCH_CLEAR (m)[p]	Clear On-latch state*	m: 0-15 p: 0-31	Boolean	Write	No
HDD_OFFLATCH_CLEAR (m)[p]	Clear Off-latch state*	m: 0-15 p: 0-31	Boolean	Write	No
HDD_COUNTER(m)[p]	Counter value	m: 0-15 p: 0-31	**Long**, DWord	Read	No
HDD_COUNTER_ READCLEAR (m)[p]	Read then clear counter value**	m: 0-15 p: 0-31	**Long**, DWord	Read	No
HDD_BANK_STATE(m)	On/Off state for all 32 points in module***	m: 0-15 p: n/a	**Long**, DWord	Read/Write	No
HDD_BANK_ONLATCH(m)	On-latch state for all 32 points in module***	m: 0-15 p: n/a	**Long**, DWord	Read	No
HDD_BANK_OFFLATCH(m)	Off-latch state for all 32 points in module***	m: 0-15 p: n/a	**Long**, DWord	Read	No
HDD_BANK_ONLATCH_ CLEAR (m)	Clear On-latch state of all 32 points of module*,****	m: 0-15 p: n/a	**Long**, DWord	Write	No
HDD_BANK_OFFLATCH_ CLEAR (m)	Clear Off-latch state of all 32 points of module*,****	m: 0-15 p: n/a	**Long**, DWord	Write	No

*These items are Write Only and not a "Read and Clear." Item may be set active by client. Driver will always return zero on client read requests. Write TRUE (non-zero) to clear respective latch.

**Unlike latches, the device clears these values after each read. OPC clients should add these items as inactive so the driver does not poll them.

***The value sent is a 32-bit mask where high bits equal On and low bits equal Off.

****The value sent is a 32-bit mask where high bites equal "Clear respective latch" and low bits equal "Do nothing."

Point Configuration

Item Name	Description	Range	Data Type	Access	Arrays
MODULETYPE[p]	Module type	0-63	**Long**, DWord	Read	No
POINTTYPE[p]	Point type	0-63	**Long**, DWord	Read/Write	No
FEATURE[p]	Point feature	0-63	**Long**, DWord	Read/Write	No
OFFSET[p]	Offset value (analog calibration)	0-63	Float	Read/Write	No
GAIN[p]	Gain value (analog calibration)	0-63	Float	Read/Write	No
HISCALE[p]	Analog high scaling factor	0-63	Float	Read/Write	No
LOSCALE[p]	Analog low scaling factor	0-63	Float	Read/Write	No
MODULETYPE_4096[p]	Module type*	0-4095	**Long**, DWord	Read	No
POINTTYPE_4096[p]	Point type*	0-4095	**Long**, DWord	Read/Write	No
FEATURE_4096[p]	Point feature*	0-4095	**Long**, DWord	Read/Write	No
OFFSET_4096[p]	Offset value (analog calibration) *	0-4095	Float	Read/Write	No
GAIN_4096[p]	Gain value (analog calibration)*	0-4095	Float	Read/Write	No
HISCALE_4096[p]	Analog high scaling factor*	0-4095	Float	Read/Write	No
LOSCALE_4096[p]	Analog low scaling factor*	0-4095	Float	Read/Write	No

*These items require firmware 8.0 or higher.

Alarm

Item Name	Description	Range	Data Type	Access	Arrays
ALARM_HI_STATE[p]	High alarm state	0-63	Boolean	Read	No
ALARM_HI_ENABLE[p]	High alarm enabled state	0-63	Boolean	Read/Write	No
ALARM_HI_SETPOINT[p]	High alarm setpoint	0-63	Float	Read/Write	No
ALARM_HI_DEADBAND[p]	High alarm deadband	0-63	Float	Read/Write	No
ALARM_LO_STATE[p]	Low alarm state	0-63	Boolean	Read	No
ALARM_LO_ENABLE[p]	Low alarm enabled state	0-63	Boolean	Read/Write	No
ALARM_LO_SETPOINT[p]	Low alarm setpoint	0-63	Float	Read/Write	No
ALARM_LO_DEADBAND[p]	Low alarm deadband	0-63	Float	Read/Write	No

Scratch Pad (SNAP PAC Controllers and Ultimate Brains Only)

Item Name	Description	Range	Data Type	Access	Arrays
SP_BIT[p]	Scratch pad bit	0-63	Boolean	Read/Write	Yes
SP_INTEGER[p]	Scratch pad integer	0-1023	**Long**, DWord	Read/Write	Yes
SP_INTEGER_EXT[p]	Scratch pad integer	1024-10239	**Long**, DWord	Read/Write	Yes

Item Name	Description	Range	Data Type	Access	Arrays
SP_FLOAT[p]	Scratch pad float	0-1023	Float	Read/Write	Yes
SP_FLOAT_EXT[p]	Scratch pad float	1024-10239	Float	Read/Write	Yes
SP_STRING[p]	Scratch pad string*	0-63	String	Read/Write	No

*Scratch pad string values are limited to 128 characters. The driver will truncate write values that exceed this length.

PID

Item Name	Description	Range	Data Type	Access	Arrays
PID_CV_IN[p]	Current value: Input	0-127	Float	Read	No
PID_CV_SP[p]	Current value: Setpoint	0-127	Float	Read	No
PID_CV_OUT[p]	Current value: Output	0-127	Float	Read/Write	No
PID_CV_FF[p]	Current value: Feed forward	0-127	Float	Read/Write	No
PID_CV_ERROR[p]	Current value: Error	0-127	Float	Read	No
PID_CV_P[p]	Current value: Gain contribution	0-127	Float	Read	No
PID_CV_I[p]	Current value: Integral contribution	0-127	Float	Read	No
PID_CV_D[p]	Current value: Derivative contribution	0-127	Float	Read	No
PID_CV_INTEGRAL[p]	Current value: Integral	0-127	Float	Read	No
PID_LSV_IN[p]	Last scanned value: Input	0-127	Float	Read/Write	No
PID_LSV_SP[p]	Last scanned value: Setpoint	0-127	Float	Read/Write	No
PID_STATUS[p]	Status flags	0-127	**Long**, DWord	Read/Write	No
PID_STATUS_ON[p]	Status flags On mask	0-127	**Long**, DWord	Read/Write	No
PID_STATUS_OFF[p]	Status flags Off mask	0-127	**Long**, DWord	Read/Write	No
PID_TUNE_P[p]	Tuning: Proportional value	0-127	Float	Read/Write	No
PID_TUNE_I[p]	Tuning: Integral value	0-127	Float	Read/Write	No
PID_TUNE_D[p]	Tuning: Derivative value	0-127	Float	Read/Write	No
PID_TUNE_FF[p]	Tuning: Feed forward gain	0-127	Float	Read/Write	No
PID_CFG_MAX_OUT [p]	Configuration: Max output change allowed	0-127	Float	Read/Write	No
PID_CFG_MIN_OUT[p]	Configuration: Min output change allowed	0-127	Float	Read/Write	No
PID_CFG_SCAN_TIME [p]	Configuration: Scan time in seconds	0-127	Float	Read/Write	No
PID_CFG_LOW_ RANGE[p]	Configuration: Output when input is low	0-127	Float	Read/Write	No

Item Name	Description	Range	Data Type	Access	Arrays
PID_CFG_HI_RANGE [p]	Configuration: Output when input is high	0-127	Float	Read/Write	No
PID_CFG_ALG[p]	Configuration: Algorithm	0-127	**Long**, DWord	Read/Write	No
PID_CFG_MAN_ MODE[p]	Configuration: Manual mode 1=Yes, 0=No	0-127	**Long**, DWord	Read/Write	No
PID_CFG_FLAGS[p]	Configuration: Flags	0-127	**Long**, DWord	Read/Write	No
PID_CFG_FLAGS_ON [p]	Configuration: Flags On mask	0-127	**Long**, DWord	Read/Write	No
PID_CFG_FLAGS_OFF [p]	Configuration: Flags Off mask	0-127	**Long**, DWord	Read/Write	No
PID_CFG_MM_IN[p]	Configuration: Input mem map address	0-127	**Long**, DWord	Read/Write	No
PID_CFG_MM_SP[p]	Configuration: Setpoint mem map address	0-127	**Long**, DWord	Read/Write	No
PID_CFG_MM_OUT[p]	Configuration: Output mem map address	0-127	**Long**, DWord	Read/Write	No
PID_SCALE_IN_LOW [p]	Scaling: Input low range	0-127	Float	Read/Write	No
PID_SCALE_IN_HI[p]	Scaling: Input high range	0-127	Float	Read/Write	No
PID_SCALE_OUT_ LOW[p]	Scaling: Output lower clamp	0-127	Float	Read/Write	No
PID_SCALE_OUT_HI [p]	Scaling: Output upper clamp	0-127	Float	Read/Write	No
PID_SCAN_COUNTER [p]	Scan counter	0-127	**Long**, DWord	Read/Write	No

Examples

Address	Description
[MMIO|ip|tcp:10.10.110.126:2001] EU[4]	EU value of point 0 in module 1. Address includes optional prefix.
EU[4]	EU value of point 0 in module 1.
EU_4096[64]	EU value of point 0 in module 1.
EU[4-7]	Array with EU value of points 0, 1, 2, 3 in module 1.
STATE[9]	State of point 1 in module 2.
HDD_STATE(3)[0]	State of point 0 in HDD module 3.
HDD_BANK_STATE(3)	State of points 0 through 31 in HDD module 3. Value returned as a 32-bit integer, LSB = point 0.
ALARM_HI_STATE[5]	High state of alarm 5.
SP_INTEGER[4]	Scratch pad integer value 4.
PID_CV_IN[10]	Current value input of PID 10.

Strategy Variables (CONT Protocol)

Basic Address Syntax
DataType;Property;VariableName

Table Element Address Syntax
DataType;Property[ElementNumber]

Table Element Array Address Syntax
DataType;Property[StartElementNumber-EndElementNumber]

Note: Arrays are limited to 1024 bytes.

Bit Address Syntax
DataType;Property.BitNumber;VariableName

Note: This syntax may be used for 32-bit integer values (I32) and single 32-bit integer table elements (I32T) only. Arrays of bit in integer values are not allowed.

Variables

Data Type (Mnemonic)	Property	Description	Data Type	Access
I32	VALUE	32-bit integer value	**Long**, DWord	Read/Write
I64	VALUE	64-bit integer value*	**Long**, DWord	Read/Write
F	VALUE	Float value	Float	Read/Write
S	VALUE	String value**	String	Read/Write
T	VALUE	Timer value	Float	Read/Write

*64-bit integer values are represented as a 2-element Long or DWord array.

**String values are limited to 1024 characters. The driver will truncate write values that exceed this length.

Tables

Data Type (Mnemonic)	Property	Description	Data Type	Access
I32T	VALUE	Table of 32-bit integer values	**Long**, DWord	Read/Write
I64T	VALUE	Table of 64-bit integer values*	**Long**, DWord	Read/Write
FT	VALUE	Table of float values	Float	Read/Write
ST	VALUE	Table of string values**,***	String	Read/Write

*64-bit integer values are represented as a 2-element Long or DWord array. An n-element array of 64-bit integer values is represented as a 2n-element Long or DWord array.

**String values are limited to 1024 characters. The driver will truncate write values that exceed this length.

***This driver does not support arrays of string table elements.

Analog Points

Data Type (Mnemonic)	Property	Description	Data Type	Access
APOINT	EU	Engineering Units value*	Float	Read/Write

*Read/Write to XVAL (external value) if communications are enabled for I/O unit and point. Read/Write to IVAL (internal value) otherwise.

Digital Points

Data Type (Mnemonic)	Property	Description	Data Type	Access
DPOINT	STATE	On/Off State*	Boolean	Read/Write

*Read/Write to XVAL (external value) if communications are enabled for I/O unit and point. Read/Write to IVAL (internal value) otherwise.

Pointers

The driver can read/write data referenced by pointer variables by pre-pending "PTR_" to the DataType. Pointer tables are not supported. Supported pointer types include the following:

PTR_I32
PTR_I64
PTR_F
PTR_S
PTR_T
PTR_APOINT
PTR_DPOINT

Examples

Address	Description
[CONT|ip|tcp:10.10.110.126:22001] F;VALUE;MyFloatVariable	Read/Write value of a floating point variable named "MyFloatVariable." Address includes optional prefix.
F;VALUE;MyFloatVariable	Read/Write value of a floating point variable named "MyFloatVariable."
I32;VALUE;MyIntegerVariable	Read/Write value of a 32-bit integer variable named "MyIntegerVariable."
I32;VALUE.0;MyIntegerVariable	Read/Write bit 0 (LSB) of 32-bit integer variable named "MyIntegerVariable."
I32T;VALUE[0];MyIntegerTable	Read/Write value of element 0 of a 32-bit integer table called "MyIntegerTable."
I32T;VALUE[1].0;MyIntegerTable	Read/Write bit 0 of table element 1 value.
I32T;VALUE[0-3];MyIntegerTable	Read/Write value of table elements 0, 1, 2, and 3.
PTR_I32;VALUE;pMyIntegerPointer	Read/Write value of 32-bit integer referenced by the pointer variable named "pMyIntegerPointer."

Automatic Tag Database Generation

The Opto 22 Ethernet Driver can import items from a browser database file (*.bdb). Items must meet the following criteria for import:

- The IP must match the Device ID.
- The item must have supported protocol (MMIO or CONT).
- The address must be supported.*
- The array size must be compatible.*
- The data type must be compatible.**
- Read and Clear MMIO addresses are not imported, but can be created manually.

*For more information, refer to **Address Descriptions**.

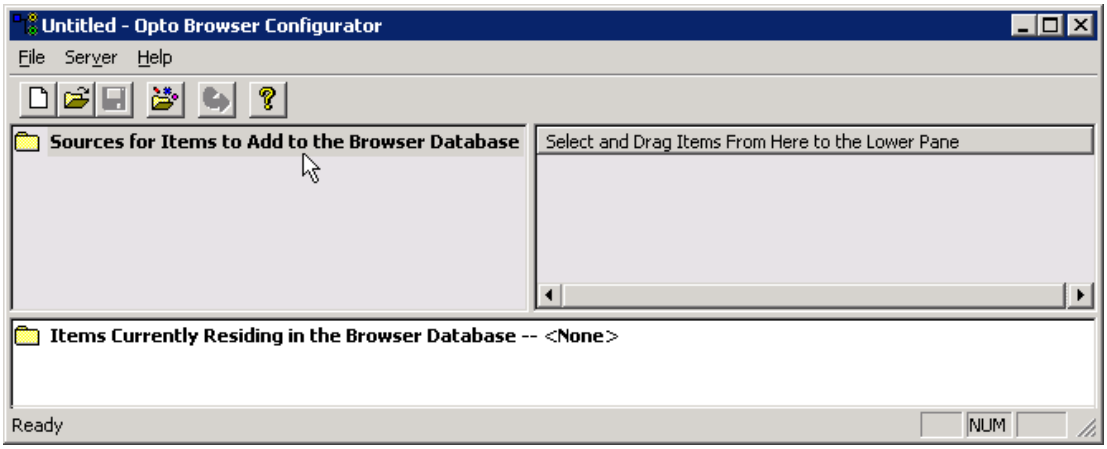
**For example, the driver does not support string arrays.

Creating a Browser Database File

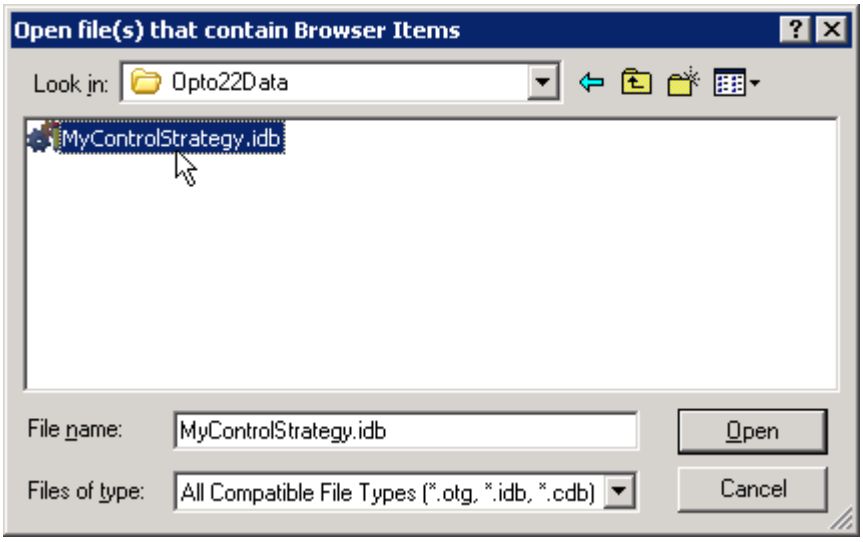
A browser database file can be created from one or more device configuration files using the Opto Browser Configurator tool. Follow the instructions below for directions on how to create a browser database file.

Note: The images below are from the Opto Browser Configurator tool, version R8.2a.

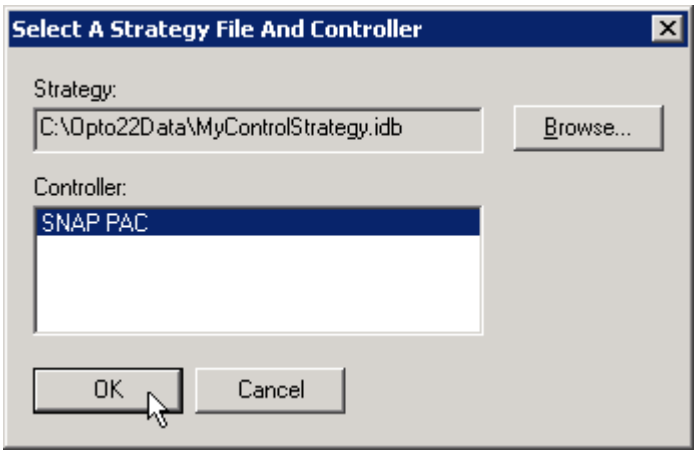
1. Start a new project in the **Browser Configurator**.



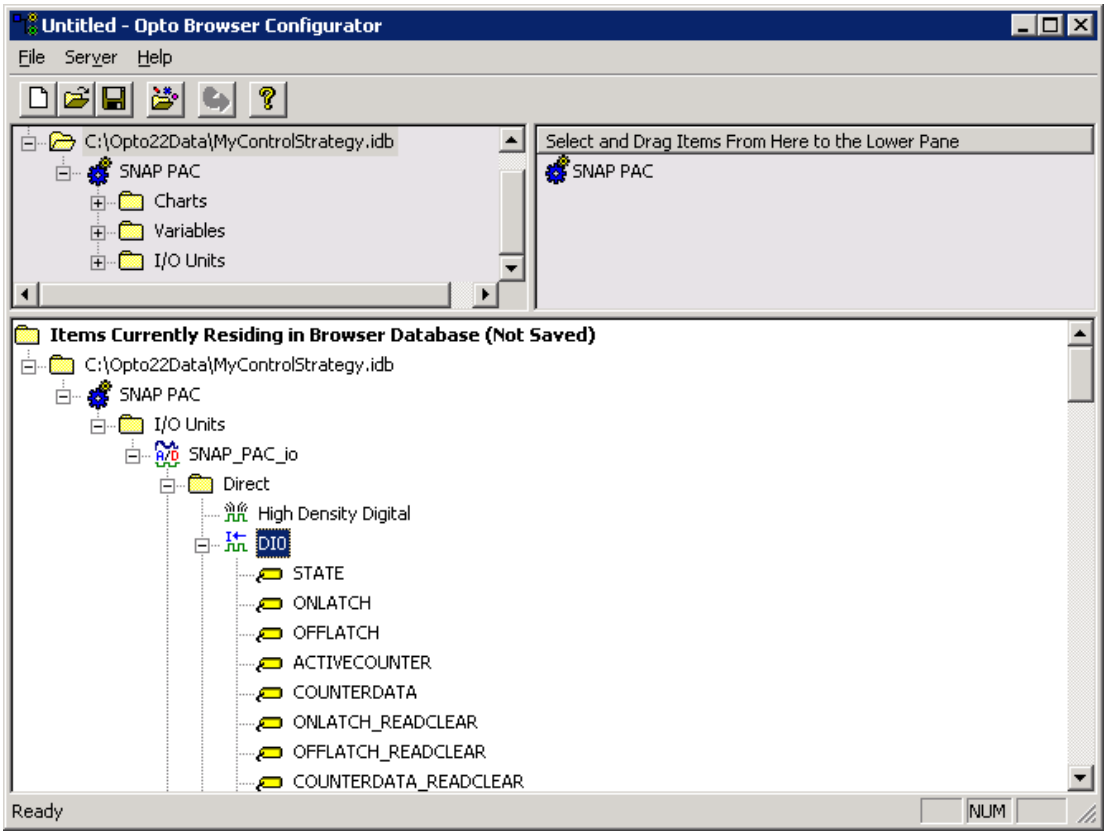
2. Open one or more Browser Item Files created with **PAC Manager** (*.otg), **PAC Control** (*.idb), or **OptoControl** (*.cdb).



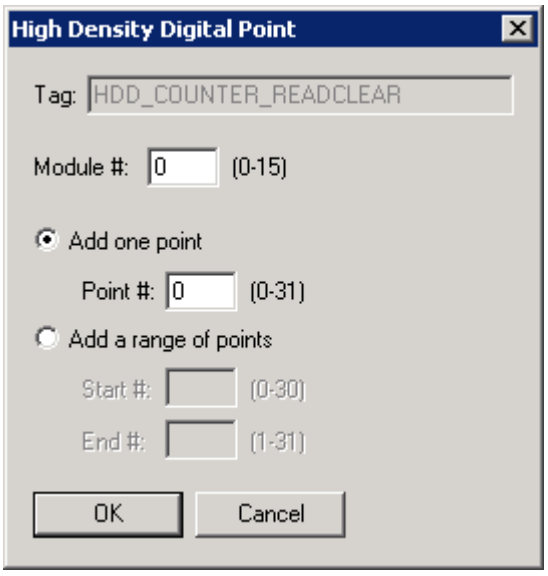
3. If an .idb or .cdb file was opened, the controllers will need to be specified.



4. At this point, the browse items should be visible in the upper-left panel. The right panel will show which specific tags are associated with the browser items selected on the left panel. To include tags in the database, drag and drop them from the right panel to the bottom panel.



5. If including High Density Digital (HDD) module items, additional information must be specified. Right-click on the item and select **Add to Browser Database...**.



6. Next, save the browser database.

Importing Browser Items

The driver can automatically generate tags for items in the browser database file. To do so, follow the instructions below.

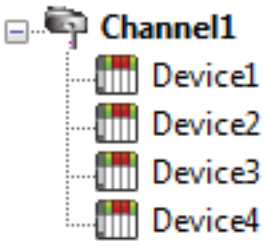
1. In the specific device's Device Properties, click on the **Import** tab.
2. Specify the name and path to the database file. Then, click **Apply**.
3. In Device Properties, click on the **Database Creation** tab. The driver can be configured to import these tags every time the driver starts, thus automatically keeping the driver configuration up to date with changes made to the database file. It is, however, usually sufficient to import the tags once by clicking **Auto Create** | **OK**. For more information on database creation options, refer to the OPC Server Help documentation.
4. Once the automatic tag generation feature is triggered, the driver will open the specified browser database file and create a tag for each supported item with location IP addresses that match the driver's Device ID.

See Also: Address Descriptions

Optimizing Communications

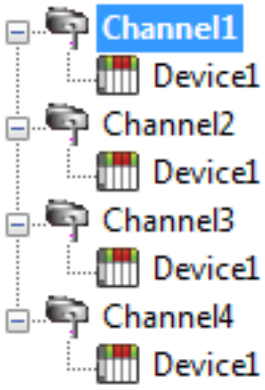
The Opto 22 Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Opto 22 Ethernet Driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application and gain maximum performance.

This server refers to communications protocols like Opto 22 Ethernet device 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 Ethernet device 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 Opto 22 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 Opto 22 Ethernet Device channel. In this configuration, the driver must move from one device to the next as quickly as possible in order 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 Opto 22 Ethernet Driver could only define one single channel, then the example shown above would be the only option available; however, the Opto 22 Ethernet Driver can define up to 256 channels. 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 has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 256 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 256 devices. While 256 or fewer devices may be ideal, the application will still benefit from additional channels. Although by 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.

Error Descriptions

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

Address Validation

Missing Address
Device's address '<address>' contains a syntax error
Address '<address>' is out of range for the specified device or register
Device address '<address>' is not supported by model '<model name>'
Array support is not available for the specified address: '<address>'
Data Type'<type>' is not valid for the device address '<address>
Device address '<address>' is Read Only

Device Status Messages

Device '<channel.device>' not responding to requests on I/O Unit (MMIO) port
Device '<channel.device>' not responding to requests on Control Engine (CONT) port
Unable to write to '<address>' on device '<device name>'

Device Specific Messages

Winsock initialization failed (OS Error = <n>)
Winsock V1.1 or higher must be installed to use the Opto 22 Ethernet Driver
Unable to bind to adapter: '<adapter name>'. Connect failed
Powerup clear failed for device '<device name>'. <Protocol> response code: <n>
Write failed for tag '<tag name>' on device '<device name>'. <Protocol> response code: <n>
Read failed for tag '<tag name>' on device '<device name>'. <Protocol> response code: <n>
Block read failed for <n> bytes starting at <memory map offset> on device '<device name>'. <Protocol> response code: <n>
Read failed for tag '<tag name>' on device '<device name>'. Object appears to be invalid in current strategy
Read failed for tag '<tag name>' on device '<device name>'. Unexpected CONT data format
Read request failed for multiple tags on device '<device name>'. CONT response code: <n>

Import Error Messages

Did not import one or more items with incompatible protocol or IP
No compatible items found in import file
Did not import item '<address>' at record <record> - arrays not supported for specified address
Did not import item '<address>' at record <record> - Read and Clear tags must be manually created
Error parsing import file record number n, field f
Did not import item '<address>' at record <record> - unsupported array size

Response Codes

MMIO Response Codes

CONT Response Codes

Missing address

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically has no length.

Solution:

Re-enter the address in the client application.

Device address '<address>' contains a syntax error

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically contains one or more invalid characters.

Solution:

Re-enter the address in the client application.

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

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically 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.

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

Error Type:

Warning

Possible Cause:

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

Solution:

Verify 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.

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 dynamically has been assigned an invalid data type.

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

Device address '<address>' is Read Only

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically 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.

Device '<channel.device>' not responding to requests on I/O Unit (MMIO) port

Error Type:
Serious

Possible Cause:

1. The MMIO port number configured in the device may not match the port configured in the hardware.
2. The device cannot accept any more TCP connections.

Solution:

1. Verify the port configuration and change as needed.
2. Eliminate other applications and hardware that have made unnecessary TCP connections to the device. Alternatively, configure the driver to UDP.

See Also:
Communications Parameters

Device '<channel.device>' not responding to requests on Control Engine (CONT) port

Error Type:
Serious

Possible Cause:
The CONT port number configured in the device may not match the port configured in the hardware.

Solution:
Verify the port configuration and change as needed.

See Also:
Communications Parameters

Unable to write to '<address>' on device '<device>'

Error Type:
Serious

Possible Cause:

1. The network connection between the device and the host PC is broken.
2. The communications parameters configured for the device and driver do not match.

Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.

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 communications.	Wait a few seconds and restart the driver.

OS Error	Indication	Possible Solution
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 Opto 22 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.

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

Error Type:
Fatal

Possible Cause:
The driver was unable to bind to the specified network adapter, which is necessary for communications with the device.

Reasons:

1. Adapter is disabled or no longer exists.
2. Network system failure, such as Winsock or network adapter failure.
3. No more available ports.

Solution:

1. Check the Channel Properties | Network Interface | Network Adapter list in the Communications Server application for network adapters available on the system. If '<adapter>' is not in this list, steps should be taken to make it available to the system. This includes but is not limited to: verifying that the network connection is enabled and connected in the PC's Network Connections.
2. Determine how many channels are using the same '<adapter>' in the communications server application. Reduce this number so that only one channel is referencing '<adapter>'. If the error still occurs, check to see if other applications are using that adapter and shut down those applications

Powerup clear failed for device'<device name>'. <Protocol> response code: <n>

Error Type:
Serious

Possible Cause:
The device was restarted and requires the driver to send a "Powerup Clear' message before data access is permitted.

Solution:
The problem may correct itself as the driver issues new requests. Additional actions will depend on the reported error code. Refer to the response code for the specific reason of failure.

Note:
If the Protocol is MMIO, refer to **MMIO Response Codes**. If Protocol is CONT, refer to **CONT Response Codes**.

Write failed for tag '<tag name>' on device '<device name>'. <Protocol> response code: <n>

Error Type:
Warning

Possible Cause:
The device rejected a write request. Refer to the response code for the specific reason why.

Solution:
The problem may correct itself as the driver issues new requests. Additional actions will depend on reported error code.

Note:
If Protocol is MMIO, see **MMIO Response Codes**. If Protocol is CONT, see **CONT Response Codes**.

Read failed for tag '<tag name>' on device '<device name>'. <Protocol> response code: <n>

Error Type:
Warning

Possible Cause:
The device rejected a read request for a single tag. Refer to the response code for the specific reason why.

Solution:
The problem may correct itself as the driver issues new requests. Additional actions will depend on reported error code.

Note:
If Protocol is MMIO, see **MMIO Response Codes**. If Protocol is CONT, see **CONT Response Codes**.

Block read failed for <n> bytes starting at <memory map offset> on device '<device name>'. <Protocol> response code: <n>

Error Type:
Warning

Possible Cause:
The device rejected a read request for a block of data that would update multiple tags. Refer to the response code for the specific reason why.

Solution:
The problem may correct itself as the driver issues new requests. Additional actions will depend on reported error code.

Note:
If Protocol is MMIO, see **MMIO Response Codes**. If Protocol is CONT, see **CONT Response Codes**.

Read failed for tag '<tag name>' on device '<device name>'. Object appears to be invalid in current strategy

Error Type:
Warning

Possible Cause:
The tag addresses a strategy variable that is not defined in the device control strategy.

Solution:
1. Correct the address or remove the tag.
2. Define a new strategy variable.

Read failed for tag '<tag name>' on device '<device name>'. Unexpected CONT data format

Error Type:
Serious

Possible Cause:
The driver received a response to a strategy variable Read or Write that did not have the expected length or format.

Solution:
Contact Technical Support.

Read request failed for multiple tags on device '<device name>'. CONT response code: <n>

Error Type:
Warning

Possible Cause:
The device rejected a read request for multiple tags that address strategy variables. Refer to the response code for specific reason why.

Solution:
The problem may correct itself as the driver issues new requests. Additional actions will depend on the reported error code.

See Also:
CONT Response Codes

Did not import one or more items with incompatible protocol or IP

Error Type:
Warning

Possible Cause:
1. The import file contained items for a device with an IP other than that configured for the Device ID.
2. The import file contained items for a protocol that is not supported by this driver.

Solution:
This driver will only import those items with a matching IP address. Make sure the IP specified as the Device ID matches the desired items in the import file.

Note:
It is possible for bdb files to contain data for more than one device. In these cases, it is normal for the driver to log this message.

See Also:
Device Setup

No compatible items found in import file

Error Type:
Warning

Possible Cause:
An item must meet the following criteria for import:

1. The IP must match the Device ID.
2. The item must have supported protocol (MMIO or CONT).
3. The address must be supported.
4. The array size must be compatible.
5. The data type must be compatible.
6. Read and Clear MMIO addresses are not imported, but can be created manually.

Solution:
Check each of the import criteria above.

Note:
The most common reason when tags are not imported is an IP mismatch.

See Also:
Address Descriptions
Automatic Tag Database Generation

Did not import item '<address>' at record <record> - arrays not supported for specified address

Error Type:
Warning

Possible Cause:
The driver does not support arrays for certain types (such as strings).

Solution:
Create tags for individual array elements. For example, although the driver would not allow an array of CONT string table elements, individual tags for each table element are permitted.

Did not import item '<address>' at record <record> - Read and Clear tags must be manually created

Error Type:
Warning

Possible Cause:
The driver will not import Read and Clear MMIO items since they must be used cautiously and must not be set active in OPC clients. These include the following:

MIN_READCLEAR
MAX_READCLEAR
MIN_READCLAR_4096
MAX_READCLEAR_4096
ONLATCH_READCLEAR
OFFLATCH_READCLEAR
COUNTERDATA_READCLEAR
HDD_COUNTER_READCLEAR

Solution:
Tags for these addresses may be created manually.

Error parsing import file record number n, field f

Error Type:
Warning

Possible Cause:

1. The import file may have been modified and an invalid entry was created.
2. The import file format may have changed since the driver was created.

Solution:

1. Correct or recreate the import file.
2. Contact Technical Support.

Did not import item '<address>' at record <record> - unsupported array size

Error Type:
Warning

Possible Cause:
The driver does not support arrays larger than 1024 bytes.

Solution:
Create multiple arrays that are smaller in size.

MMIO Response Codes

Code	Meaning
1	Undefined command
2	Invalid point type
3	Invalid float value
4	Powerup Clear expected
5	Invalid memory address or invalid data for the memory address
6	Invalid command length
7	Reserved
8	Busy
9	Cannot erase flash
10	Cannot program flash
11	Downloaded images too small
12	Image CRC mismatch
13	Image length mismatch
14	Feature is not yet implemented
15	Communications watchdog timeout

CONT Response Codes

Code	Meaning
-3	Buffer overrun or invalid length
-4	Powerup clear expected
-5	Operation failed
-6	Data field error
-7	Communications watchdog timeout
-8	Invalid data
-12	Invalid table index
-14	Invalid number
-17	Port locked - strategy download in progress
-20	Device busy
-22	Command not valid on specified I/O unit
-28	Object not found
-29	Wrong object type – most likely pointer type mismatch
-30	Pointer not initialized

Index

A

B

C

D

E

G

H

I

M

N

O

P

R

S

T

U

W