Keyence Ethernet Driver

Table of Contents

Keyence Ethernet Driver

Help version 1.017

CONTENTS

Overview

The Keyence Ethernet Driver provides a reliable way to connect Keyence KV Ethernet devices to client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Keyence KV series devices.

With Keyence Ethernet Driver, Keyence PLCs can provide real-time read/write connectivity with various devices and applications, offering a single-source for users to monitor quality data for errors in production and calculate overall equipment efficiency for machines or lines connected via Keyence PLCs. Users can connect Keyence PLCs to an MES system to send recipes to devices and/or read batch information back from that PLC. Often, these PLCs are in control of an automated line, but there are some instances where MES information is sent to a stand-alone machine. Keyence Ethernet Driver offers both of these capabilities.

Setup

Supported Devices

KV Series (KV-7500, KV-7300, KV-5500, KV-5000, KV-3000, KV-1000, KV-700, KV-Nano)

Communication Protocol

Host Link

Supported Communication Parameters

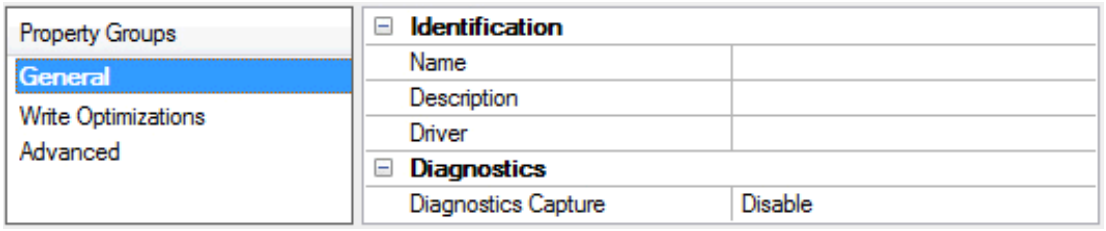
IP Address: 0.0.0.0 – 255.255.255.255
Protocol Mode: TCP/IP, UDP
Port: 1 – 65535
Receive Time Out: 0 – 59s

Maximum Number of Channels and Devices

The maximum number of channels supported by this driver is 1024. The maximum number of supported devices is 256 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. 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: 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. 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. 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

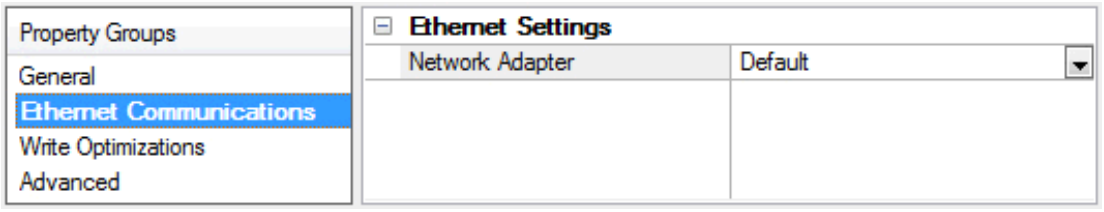
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

Note: This property is not available if the driver does not support diagnostics.

For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

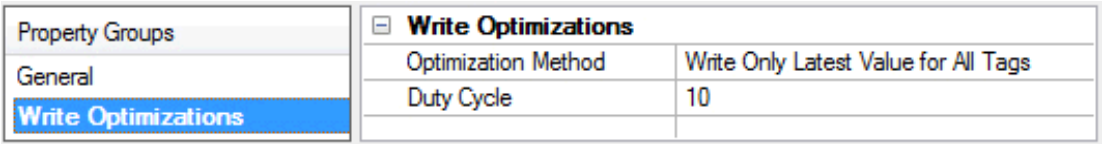


Ethernet Settings

Network Adapter: Specify the network adapter to bind. When left blank or 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:

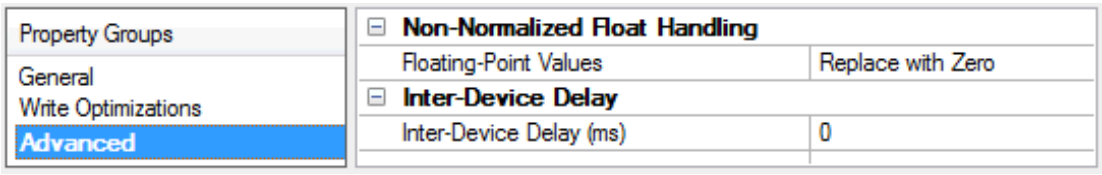
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing 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.

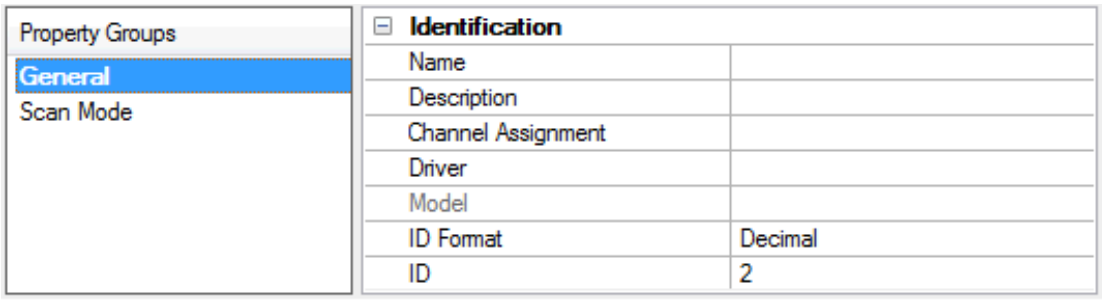
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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.

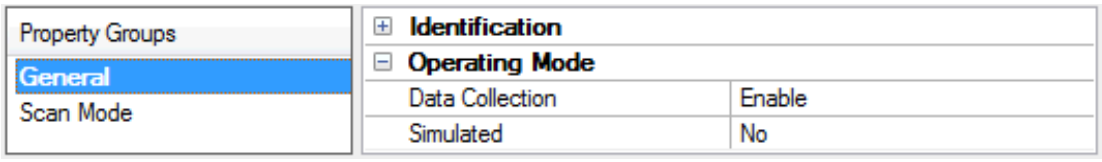
Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.
Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. For more information, refer to the driver help documentation.

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

Operating Mode



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

Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

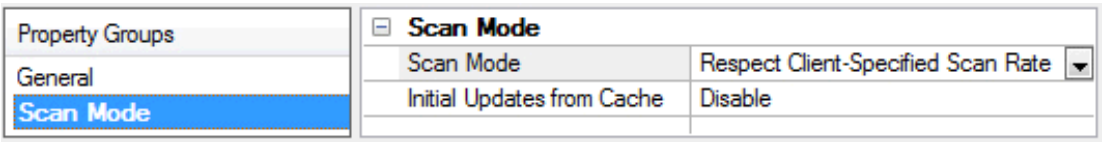
Notes:

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

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

Device Properties — Scan Mode

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



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

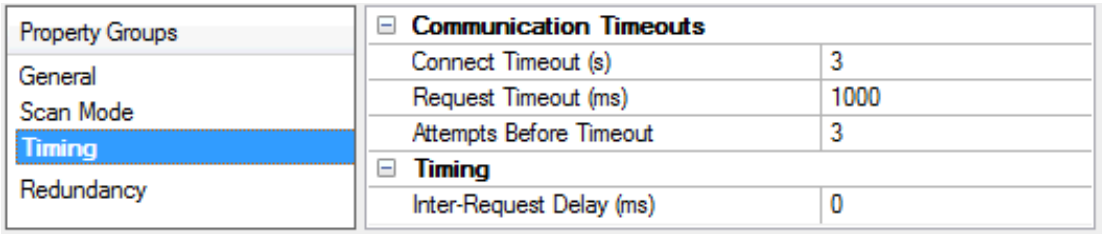
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server 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 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.

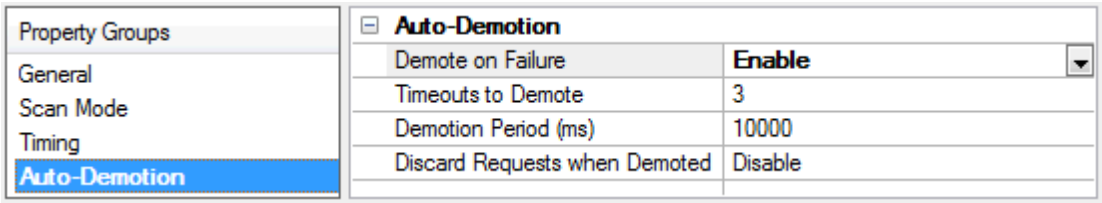
Timing

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.

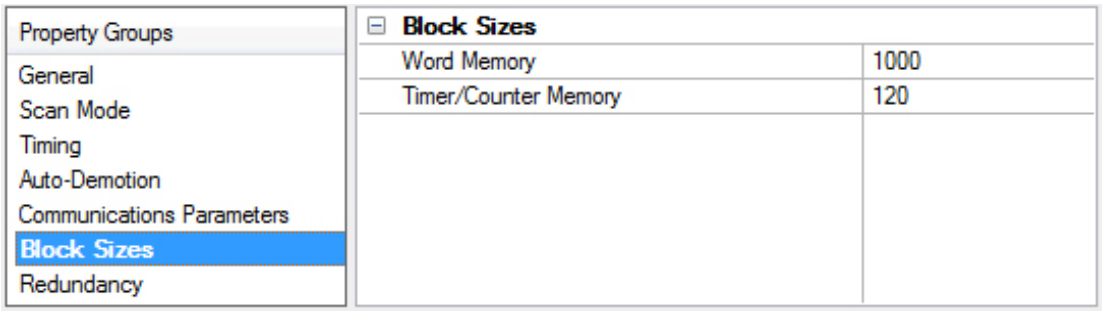
Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

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 — Block Sizes



Word Memory: The block size for reading Word memory registers. These include device types B, CM, CR, DM, EM, FM, LR, MR, R, TM, VB, VM, W, and ZF. Word memory can be read from 1 to 1000 points (memory devices) at a time. The default is 1000.

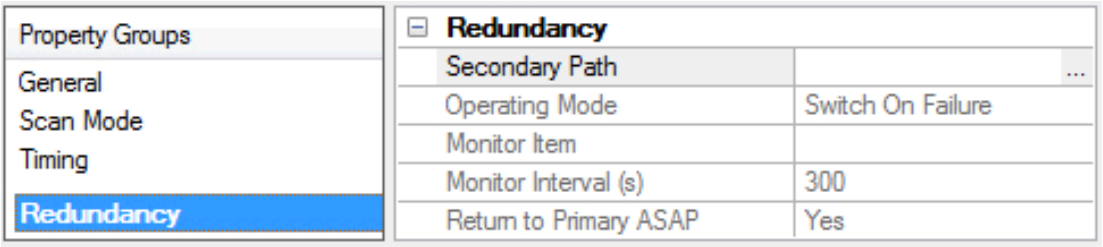
Timer / Counter Memory: The block size for reading Timer and Counter memory registers. These include device types T, TC, TS, C, CC, CS. Timer and Counter memory can be read from 1 to 120 points (memory devices) at a time. The default is 120.

Notes:

1. A higher block size means more register values are read from the device in a single request and can help reduce bandwidth. A lower block size means more requests are issued to the device and can be useful when reading data from non-contiguous locations within the device.
2. Block size changes take effect while a client is connected.
3. All other Device Types have fixed block sizes as follows:

Device Type	Fixed Block Size
Index Register (Z)	12
High-Speed Counter (CTH)	1
High-Speed Counter Comparator (CTC)	1
Digital Trimmer (AT)	8

Device Properties — Redundancy



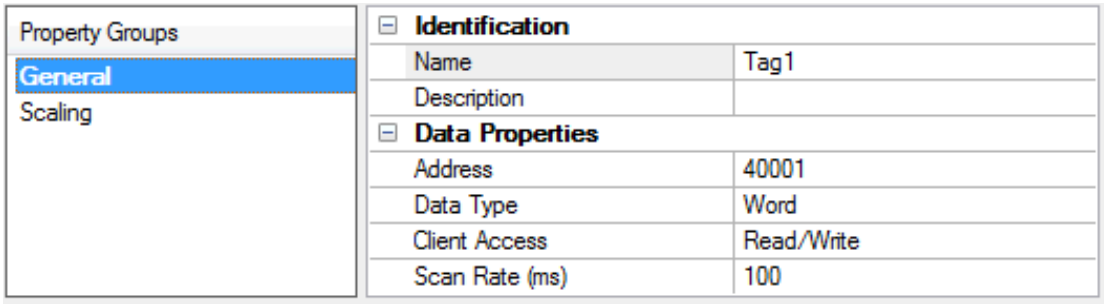
Redundancy is available with the Media-Level Redundancy Plug-In.
Consult the website, a sales representative, or the user manual for more information.

Tag Properties — General

A tag represents addresses in the PLC or other hardware device with which the server communicates. The server allows both Dynamic tags and user-defined Static tags. Dynamic tags are entered directly in the client

and specify device data. User-defined Static tags are created in the server and support tag scaling. They can be browsed from clients that support tag browsing.

For more information, refer to ***Dynamic Tags*** *and* ***Static User-Defined Tags****.*



Name: Enter a string to represent this tag. The tag name can be up to 256 characters in length. While using descriptive names is generally a good idea, some OPC client applications may have a limited display window when browsing the tag space of an OPC UA server. The tag name is part of the OPC browse data tag names must be unique within a given device branch or tag group branch. *For information on reserved characters, refer to* ***How To... Properly Name a Channel, Device, Tag, and Tag Group****.*

Tip: If the application is best suited for using blocks of tags with the same names, use tag groups to segregate the tags. *For more information, refer to* ***Tag Group Properties****.*

Description: Apply a comment to the tag. A string of up to 255 characters can be entered for the description. When using an OPC client that supports Data Access 2.0 tag properties, the description property is accessible from the tag's item Description properties.

Address: Enter the target tag's driver address. The address's format is based on the driver protocol. The address can be up to 128 characters.

Tip: For hints about how an address should be entered, click the browse (...) button. If the driver accepts the address as entered, no messages are displayed. A popup informs of any errors. Some errors are related to the data type selection and not the address string.

Data Type: Specify the format of this tag's data as it is found in the physical device. In most cases, this is also the format of the data as it returned to the client. The data type setting is an important part of how a communication driver reads and writes data to a device. For many drivers, the data type of a particular piece of data is rigidly fixed and the driver knows what format needs to be used when reading the device's data. In some cases, however, the interpretation of device data is largely in the user's hands. An example would be a device that uses 16-bit data registers. Normally this would indicate that the data is either a Short or Word. Many register-based devices also support values that span two registers. In these cases the double register values could be a Long, DWord or Float. When the driver being used supports this level of flexibility, users must tell it how to read data for this tag. By selecting the appropriate data type, the driver is being told to read one, two, four, eight, or sixteen registers or possibly a Boolean value. The driver governs the data format being chosen.

- **Default** - Uses the driver default data type
- **Boolean** - Binary value of true or false
- **Char** - Signed 8-bit integer data
- **Byte** - Unsigned 8-bit integer data
- **Short** - Signed 16-bit integer data
- **Word** - Unsigned 16-bit integer data
- **Long** - Signed 32-bit integer data

- **DWord** - Unsigned 32-bit integer data
- **LLong** - Signed 64-bit integer data
- **QWord** - Unsigned 64-bit integer data
- **Float** - 32-bit real value IEEE-754 standard definition
- **Double** - 64-bit real value IEEE-754 standard definition
- **String** - Null-terminated Unicode string
- **BCD** - Two byte-packed BCD value range is 0-9999
- **LBCD** - Four byte-packed BCD value range is 0-99999999
- **Date -** See Microsoft® Knowledge Base.

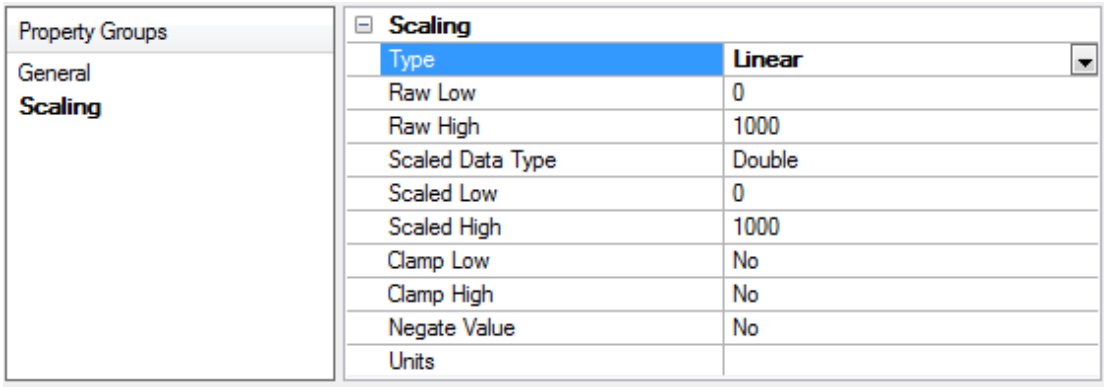
Client Access: Specify whether the tag is **Read Only** or **Read / Write**. By selecting Read Only, users can prevent client applications from changing the data contained in this tag. By selecting **Read / Write**, users allow client applications to change this tag's value as needed. The **Client Access** selection also affects how the tag appears in the browse space of an OPC UA client. Many client applications allow filtering tags based on attributes. Changing the access method of this tag may change how and when the tag appears in the browse space of the client.

Scan Rate: Specify the update interval for this tag when used with a non-OPC client. OPC clients can control the rate at which data is scanned by using the update rate that is part of all OPC groups. Normally non-OPC clients don't have that option. The server is used to specify an update rate on a tag per tag basis for non-OPC clients. Using the scan rate, users can tailor the bandwidth requirements of the server to suit the needs of the application. If, for example, data that changes very slowly needs to be read, there is no reason to read the value very often. Using the scan rate this tag can be forced to read at a slower rate reducing the demand on the communications channel. The valid range is 10 to 99999990 milliseconds (ms), with a 10 ms increment. The default is 100 milliseconds.

With the server's online full-time operation, these properties can be changed at any time. Changes made to tag properties take effect immediately; however, client applications that have already connected to this tag are not affected until they release and attempt to reacquire it. Utilize the User Manager to restrict access rights to server features and prevent operators from changing the properties.

Tag Properties — Scaling

This server supports tag Scaling, which allows raw data from the device to be scaled to an appropriate range for the application.



Type: Select the method of scaling raw values. Select **Linear**, **Square Root**, or **None** to disable. The formulas for scaling types are shown below.

Type	Formula for Scaled Value
Linear	(((ScaledHigh - ScaledLow)/(RawHigh - RawLow))*(RawValue - RawLow)) + ScaledLow
Square root	(Square root ((RawValue - RawLow)/(RawHigh - RawLow))*(ScaledHigh - ScaledLow)) + ScaledLow

Raw Low: Specify the lower end of the range of data from the device. The valid range depends on the raw tag data type. For example, if the raw value is Short, the valid range of the raw value would be from -32768 to 32767.

Raw High: Specify the upper end of the range of data from the device. The Raw High value must be greater than the Raw Low value. The valid range depends on the raw tag data type.

Scaled Data Type: Select the data type for the tag being scaled. The data type can be set to any valid OPC data type, including a raw data type, such as Short, to an engineering value with a data type of Long. The default scaled data type is Double.

Scaled Low: Specify the lower end of the range of valid resulting scaled data values. The valid range depends on the tag data type.

Scaled High: Specify the upper end of the range of valid resulting scaled data values. The valid range depends on the tag data type.

Clamp Low: Select **Yes** to prevent resulting data from exceeding the lower end of the range specified. Select **No** to allow data to fall outside of the established range.

Clamp High: Select **Yes** to prevent resulting data from exceeding the upper end of the range specified. Select **No** to allow data to fall outside of the established range.

Negate Value: Select **Yes** to force the resulting value to be negated before being passed to the client. Select **No** to pass the value to the client unmodified.

The server supports the OPC tag properties available in the 2.0 Data Access specifications. If the OPC client being used supports these properties, it can automatically configure the range of objects (such as user input objects or displays) by using the Scaling settings. Utilize the User Manager to restrict access rights to server features to prevent any unauthorized operator from changing these properties.

Data Types Description

Data Type	Description
Boolean	Single 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

Data Type	Description
	bit 15 is the sign bit
Examples of Word, Short -16-bit (two byte)	If register DM100 is specified as a 16-bit data type, bit 0 of register DM100 would be bit 0 of the 16-bit value, and bit 15 of register DM100 would be bit 15 of the 16-bit value. If register R99800 is specified as a 16-bit data type, R99800 would be bit 0 of the 16-bit value, and R99815 would be bit 15 of the 16-bit value.
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
Examples of DWord, Long, Float - 32-bit (four byte)	If register DM100 is specified as a 32-bit data type, bit 0 of register DM100 would be bit 0 of the 32-bit value, and bit 15 of register DM101 would be bit 31 of the 32-bit value. If register R99800 is specified as a 32-bit data type, R99800 would be bit 0 of the 32-bit value, and R99915 would be bit 31 of the 32-bit value.
QWord	Unsigned 64-bit value Bit 0 is the low bit Bit 63 is the high bit
LongLong	Signed 64-bit value Bit 0 is the low bit Bit 62 is the high bit Bit 63 is the sign bit
Double	64-bit floating-point value
Examples of QWord, LongLong, Double - 64-bit (eight byte)	If register DM100 is specified as a 64-bit data type, bit 0 of register DM100 would be bit 0 of the 64-bit value, and bit 15 of register DM103 would be bit 63 of the 32-bit value. If register R99800 is specified as a 64-bit data type, R99800 would be bit 0 of the 64-bit value, and R10115 would be bit 63 of the 64-bit value.
String	Null-terminated ASCII string support includes HiLo and LoHi byte order selection and string lengths up to the configured block size for the device type times the number of bytes per device type.

KV Series Addressing

The following memory map is open for all memory types to support newer devices. Consult the manufacturer's documentation for device-specific address ranges. The default data types are shown in **bold**.

Use caution when modifying tags of 32-bit data types (DWord, Long, and Float) and 64-bit data types (Qword, LongLong, and Double). Since a majority of the device types are 16-bit address spaces, 32-bit and 64-bit data type tags overlap adjacent Word addresses. When using DWord data type or larger, for example,

use DM0, DM2, DM4, and so on to prevent overlapping Words. When using Qword data type, for example, use Z1, Z3, Z5, and so on to prevent overlapping DWords.

*For more information, refer to **Bit-Within-Word Support**, **String Support**, and **Array Support**.*

Device Type	Range	Data Type	Access
Input / Output Relay	RXXXXX00 – RXXXXX15 R0000000 – R6553500 R0000000 – R6553400 R0000000 – R6553200 R0000000[1] – R6553500[1000] R0000000[1] – R6553400[500] R0000000[1] – R6553200[250]	**Boolean** Word, Short DWord, Long, Float QWord, LongLong, Double **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array	Read / Write
Control Relay	CRXXXXX00 – CRXXXXX15 CR0000000 – CR6553500 CR0000000 – CR6553400 CR0000000 – CR6553200 CR0000000[1] – CR6553500 [1000] CR0000000[1] – CR6553400 [500] CR0000000[1] – CR6553200 [250]	**Boolean** Word, Short DWord, Long, Float QWord, LongLong, Double **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array	Read / Write
Latch Relay	LRXXXXX00 – LRXXXXX15 LR0000000 – LR6553500 LR0000000 – LR6553400 LR0000000 – LR6553200 LR0000000[1] – LR6553500 [1000] LR0000000[1] – LR6553400 [500] LR0000000[1] – LR6553200 [250]	**Boolean** Word, Short DWord, Long, Float QWord, LongLong, Double **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array	Read / Write
Internal Auxiliary Relay	MRXXXXX00 – MRXXXXX15 MR0000000 – MR6553500 MR0000000 – MR6553400 MR0000000 – MR6553200 MR0000000[1] – MR6553500 [1000] MR0000000[1] – MR6553400 [500] MR0000000[1] – MR6553200 [250]	**Boolean** Word, Short DWord, Long, Float QWord, LongLong, Double **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array	Read / Write
Link Relay	BXXXX0 – BXXXXF B00000 – BFFFF0 B00000 – BFFFE0 B00000 – BFFFC0 B00000[1] – BFFFF0[1000]	**Boolean** Word, Short DWord, Long, Float QWord, LongLong, Double **Word**, Short **Array**	Read / Write

Device Type	Range	Data Type	Access
	B00000[1] – BFFFE0[500] B00000[1] – BFFFC0[250]	DWord, Long, Float Array QWord, LongLong, Double Array	
Work Relay	VBXXXX0 – VBXXXXF VB00000 – VBFFFF0 VB00000 – VBFFFE0 VB00000 – VBFFFC0 VB00000[1] – VBFFFF0[1000] VB00000[1] – VBFFFE0[500] VB00000[1] – VBFFFC0[250]	**Boolean** Word, Short DWord, Long, Float QWord, LongLong, Double **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array	Read / Write
Control Memory	CM0.0 – CM1048575.15 CM0 – CM1048575 CM0 – CM1048574 CM0 – CM1048572 CM0[1] – CM1048575[2000] CM0[1] – CM1048575[1000] CM0[1] – CM1048574[500] CM0[1] – CM1048572[250] CM0:1 – CM1048575:2000 CM0:1H – CM1048575:2000H CM0:1L – CM1048575:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Data Memory	DM0.0 – DM1048575.15 DM0 – DM1048575 DM0 – DM1048574 DM0 – DM1048572 DM0[1] – DM1048575[2000] DM0[1] – DM1048575[1000] DM0[1] – DM1048574[500] DM0[1] – DM1048572[250] DM0:1 – DM1048575:2000 DM0:1H – DM1048575:2000H DM0:1L – DM1048575:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Extended Data Memory	EM0.0 – EM1048575.15 EM0 – EM1048575 EM0 – EM1048574 EM0 – EM1048572 EM0[1] – EM1048575[2000] EM0[1] – EM1048575[1000] EM0[1] – EM1048574[500] EM0[1] – EM1048572[250] EM0:1 – EM1048575:2000 EM0:1H – EM1048575:2000H EM0:1L – EM1048575:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write

Device Type	Range	Data Type	Access
Temporary Memory	TM0.0 – TM1048575.15 TM0 – TM1048575 TM0 – TM1048574 TM0 – TM1048572 TM0[1] – TM1048575[2000] TM0[1] – TM1048575[1000] TM0[1] – TM1048574[500] TM0[1] – TM1048572[250] TM0:1 – TM1048575:2000 TM0:1H – TM1048575:2000H TM0:1L – TM1048575:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Work Memory	VM0.0 – VM1048575.15 VM0 – VM1048575 VM0 – VM1048574 VM0 – VM1048572 VM0[1] – VM1048575[2000] VM0[1] – VM1048575[1000] VM0[1] – VM1048574[500] VM0[1] – VM1048572[250] VM0:1 – VM1048575:2000 VM0:1H – VM1048575:2000H VM0:1L – VM1048575:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Timer	T0 – T1048575	**Boolean**	Read / Write
Timer Current Value	TC0 – TC1048575 TC0[1] – TC1048575[120]	**DWord**, Long **DWord**, Long **Array**	Read / Write
Timer Setting Value	TS0 – TS1048575 TS0[1] – TS1048575[120]	**DWord**, Long **DWord**, Long **Array**	Read / Write
Counter	C0 – C1048575	**Boolean**	Read / Write
Counter Current Value	CC0 – CC1048575 CC0[1] – CC1048575[120]	**DWord**, Long **DWord**, Long, **Array**	Read / Write
Counter Setting Value	CS0 – CS1048575 CS0[1] – CS1048575[120]	**DWord**, Long **DWord**, Long **Array**	Read / Write
High-Speed Counter	CTH0 – CTH1048575	**DWord**, Long	Read / Write
High-Speed Counter Comparator	CTC0 – CTC1048575	**DWord**, Long	Read / Write
File Register Current Bank	FM0.0 – FM1048575.15 FM0 – FM1048575 FM0 – FM1048574 FM0 – FM1048572 FM0[1] – FM1048575[2000] FM0[1] – FM1048575[1000]	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array**	Read / Write

Device Type	Range	Data Type	Access
	FM0[1] – FM1048574[500] FM0[1] – FM1048572[250] FM0:1 – FM1048575:2000 FM0:1H – FM1048575:2000H FM0:1L – FM1048575:2000L	DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	
File Register Consecutive Mode	ZF0.0 – ZF1048575.15 ZF0 – ZF1048575 ZF0 – ZF1048574 ZF0 – ZF1048572 ZF0[1] – ZF1048575[2000] ZF0[1] – ZF1048575[1000] ZF0[1] – ZF1048574[500] ZF0[1] – ZF1048572[250] ZF0:1 – ZF1048575:2000 ZF0:1H – ZF1048575:2000H ZF0:1L – ZF1048575:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Link Register	W0.0 – WFFFF.15 W0 – WFFFF W0 – WFFFE W0 – WFFFC W0[1] – WFFFF[2000] W0[1] – WFFFF[1000] W0[1] – WFFFE[500] W0[1] – WFFFC[250] W0:1 – WFFFF:2000 W0:1H – WFFFF:2000H W0:1L – WFFFF:2000L	**Boolean** **Word**, Short DWord, Long, Float QWord, LongLong, Double Byte Array **Word**, Short **Array** DWord, Long, Float Array QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Index Register	Z1.0 – Z1048575.31 Z1 – Z1048575 Z1 – Z1048574 Z1[1] – Z1048575[48] Z1[1] – Z1048575[12] Z1[1] – Z1048574[6] Z1:1 – Z1048575:48 Z1:1H – Z1048575:48H Z1:1L – Z1048575:48L	**Boolean** **DWord**, Long, Float QWord, LongLong, Double Byte Array **DWord**, Long, Float **Array** QWord, LongLong, Double Array **String** **String** **String**	Read / Write
Digital Trimmer	AT0.0 – AT1048575.31 AT0 – AT1048575 AT0 – AT1048574 AT0[1] – AT1048575[8] AT0[1] – AT1048574[4]	**Boolean** **DWord**, Long, Float QWord, LongLong, Double **DWord**, Long, Float **Array** QWord, LongLong, Double Array	Read / Write

Bit-Within-Word Support

Certain Memory, Register and Digital Trimmer address spaces support Bit-within-Word tag syntax (as indicated in the table above). Bit syntax allows reading from and writing to only a specific bit in physical Word memory or DWord memory of the PLC. Tags created with Bit syntax default to Boolean data type. Writing to a Bit syntax tag modifies only the targeted Bit within the Word or DWord memory space by performing a read-modify-write sequence. A read will be performed before the write of the targeted Bit so that the other Bits within the Word or DWord memory are not overwritten.

Examples

- To address Bit 0 at DM100 (16-Bit memory), enter: DM100.0
- To address Bit 15 at DM100 (16-Bit memory), enter: DM100.15
- To address Bit 31 at AT0 (32-Bit memory), enter: AT0.31

Note: A read-modify write is not performed on every write to a Bit syntax tag. Only one previous read of the address is required in order to write to a Bit syntax tag. If writing to a Bit syntax tag where the Word or DWord memory is changing frequently it is recommended to add a separate Word or DWord tag for that address to handle the reads in order to maintain the expected data.

String Support

The Open model supports reading and writing Memory device types: CM, DM, EM, TM, and VM and Register device types: FM, W, ZF and Z as an ASCII or multi-byte string. When using any of these device types for string data, each address can contain up to four bytes of character data for 32-bit device types and up to two bytes of character data for 16-bit device types. The allowable string length in bytes for any applicable device type is from 1 to ([BlockSize]*[DeviceTypeBytes]). The length of the string in bytes cannot exceed the allowable address space of the intended device type. The byte order of the character data can be selected when the string is defined by appending H or L to the string address syntax. HiLo byte ordering is the default if not specified. Strings of odd byte length or string byte lengths defined to be less than the number of bytes of the address space utilized will write NUL characters (0) to the remaining bytes of the address space. If a string is within a block the string will be read as part of the block read. If a string crosses a block boundary then reading the string will result in an independent read request to the device for that string.

Strings support the single byte or multi-byte character set defined by the OS system locale.

Examples

- To address a string starting at DM0 with a length of 2000 bytes of character data and HiLo byte order, enter: DM0:2000
- To address a string starting at WCE21 with a length of 1 byte of character data and HiLo byte order, enter: WCE21:1H
- To address a string starting at DM0 with a length of 7 bytes of character data and LoHi byte order, enter: DM0:7L
- HiLo versus LoHi ASCII string “To” at address DM0:
    1. DM0:2H (HiLo byte ordering) – DM0 = “To” (0x546F)
    2. DM0:2L (LoHi byte ordering) – DM0 = “oT” (0x6F54)
- Maximum string length for 16-bit addresses using default maximum block size of 1000 is: 1000 * 2bytes = 2000 bytes
- Maximum string length for 32-bit address Z with hard-coded block size of 12 is: 12 * 4bytes = 48 bytes

Array Support

Data Types: Arrays are supported for all data types except Boolean and String.
Device Types: Arrays are supported for Relay device types: B, CR, LR, MR, R, and VB; 16-bit Memory device types: CM, DM, EM, TM and VM; and Register device types: FM, W, and ZF; 32-bit Memory device types: AT and Z and 32-bit Timer/Counter device types: TC, TS, CC, and CS.

Array syntax is formatted by entering the device type and number followed by square brackets containing the array length specification. The tag data type will default to Word Array for 16-bit device types and DWord Array for 32-bit device types, if not specified when using this syntax. See examples below:

- *R40000[2] Word Array* – This tag reads from and write to addresses R40000 and R40100.
- *DM0[3] DWord Array* – This tag reads from and write to addresses DM0 & DM1, DM2 & DM3, and DM4 & DM5.
- *W3C[8] Word Array* – This tag reads from and write to addresses W3C, W3D, W3E, W3F, W40, W41, W42, and W43.
- *AT0[4] DWord Array* – This tag reads from and write to addresses AT0, AT1, AT2, and AT3.
- *Z1[3] Float Array* – This tag reads from and write to addresses Z1, Z2, and Z3.
- *Z1[4] Double Array* – This tag reads from and write to addresses Z1 & Z2, Z3 & Z4, Z5 & Z6, and Z7 & Z8.

The specified array length cannot exceed the configured Memory block size.

If an array is within a block the array will be read as part of the block read. If an array crosses a block boundary then reading the array will result in an independent read request to the device for that array.

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

Unable to read address on device. | Address = '<address>'.

Error Type:
Error

Possible Cause:

1. Communication with device succeeded, but the response packet is malformed or of unexpected length.
2. The driver could not allocate the resources required to read from the device.

Possible Solution:

1. Improve connection to reduce chances of lost data. Try read again.
2. Shut down unnecessary applications, restart the server, and try again.

Unable to read address block on device. Device returned an error. | Address block = '<address>' to '<address>', Error code = <code>.

Error Type:
Error

Possible Cause:
The device returned an error code in the response indicating a problem with the request.

Possible Solution:
The solution depends on the error code returned.

See Also:
Device Response Error Codes

Unable to read address block on device. | Address block = '<address>' to '<address>'.

Error Type:
Error

Possible Cause:

1. Communication with device succeeded, but the response packet is malformed or of unexpected length.
2. The driver could not allocate the resources required to read from the device.

Possible Solution:

1. Improve connection to reduce chances of lost data. Try read again.
2. Shut down unnecessary applications, restart the server, and try again.

Unable to read address on device. Device returned an error. | Address = '<address>', Error code = <code>.

Error Type:
Error

Possible Cause:
The device returned an error code in the response indicating a problem with the request.

Possible Solution:
The solution depends on the error code returned.

See Also:
Device Response Error Codes

Unable to write to address on device. | Address = '<address>'.

Error Type:
Error

Possible Cause:

1. Communication with device succeeded, but the response packet is malformed or of unexpected length.
2. The driver could not allocate the resources required to write to the device.

Possible Solution:

1. Improve connection to reduce chances of lost data. Try write again.
2. Shut down unnecessary applications and try again.

Unable to write to address on device. Device returned an error. | Address = '<address>', Error code = <code>.

Error Type:
Error

Possible Cause:

The device returned an error code in the response indicating a problem with the request.

Possible Solution:

The solution depends on the error code returned.

See Also:

Device Response Error Codes

Device Response Error Codes

Error Code	Description	Possible Cause	Possible Solution
0	Device Number Error	The specified device number or address is out of range.	Change the address to one that is within range for the device.
1	Instruction / Command Error	Unsupported command is sent. Using the Open model in this driver allows addresses that may not be supported in the device.	Change the address to one that is within range for the device.
2	Program not registered	No program is registered in the CPU. The RUN/PROG switch of CPU unit is not in RUN mode.	Add a program to the CPU. Switch the RUN/PROG switch of the CPU unit to the correct RUN mode
3	Reserved		
4	Write Protected	The device number of the device type is protected against writes.	Change the address to one that allows writes or configure the address in the device to allow writes.
5	PLC Error	CPU unit is in error.	Correct CPU unit error and retry the request.
6	No Comment	The device number of the device type does not have comments.	If desired, register the comments at the device number in the device.

Com port is in use by another application. | Port = '<port>'.

Error Type:

Error

Possible Cause:

The serial port assigned to a device is being used by another application.

Possible Solution:

1. Verify that the correct port has been assigned to the channel.
2. Verify that only one copy of the current project is running.

Unable to configure com port with specified parameters. | Port = COM<number>, OS error = <error>.

Error Type:

Error

Possible Cause:
The serial parameters for the specified COM port are not valid.

Possible Solution:
Verify the serial parameters and make any necessary changes.

Driver failed to initialize.

Error Type:
Error

Unable to create serial I/O thread.

Error Type:
Error

Possible Cause:
The server process has no resources available to create new threads.

Possible Solution:
Each tag group consumes a thread. The typical limit for a single process is about 2000 threads. Reduce the number of tag groups in the project.

Com port does not exist. | Port = '<port>'.

Error Type:
Error

Possible Cause:
The specified COM port is not present on the target computer.

Possible Solution:
Verify that the proper COM port is selected.

Error opening com port. | Port = '<port>', OS error = <error>.

Error Type:
Error

Possible Cause:
The specified COM port could not be opened due an internal hardware or software problem on the target computer.

Possible Solution:
Verify that the COM port is functional and may be accessed by other applications.

Connection failed. Unable to bind to adapter. | Adapter = '<name>'.

Error Type:

Error

Possible Cause:
Since the specified network adapter cannot be located in the system device list, it cannot be bound to for communications. This can occur when a project is moved from one PC to another (and when the project specifies a network adapter rather than using the default). The server reverts to the default adapter.

Possible Solution:
Change the Network Adapter property to Default (or select a new adapter), save the project, and retry.

Winsock shut down failed. | OS error = <error>.

Error Type:
Error

Winsock initialization failed. | OS error = <error>.

Error Type:
Error

Possible Solution:

1. The underlying network subsystem is not ready for network communication. Wait a few seconds and restart the driver.
2. The 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 this driver.

Error Type:
Error

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

Possible Solution:
Upgrade Winsock to version 1.1 or higher.

Socket error occurred binding to local port. | Error = <error>, Details = '<information>'.

Error Type:
Error

Device is not responding.

Error Type:
Warning

Possible Cause:

1. The connection between the device and the host PC is broken.
2. The communication parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect device ID.
4. The response from the device took longer to receive than allowed by the Request Timeout device setting.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.
3. Verify that the device ID for the named device matches that of the actual device.
4. Increase the Request Timeout setting to allow the entire response to be handled.

Device is not responding. | ID = '<device>'.

Error Type:
Warning

Possible Cause:

1. The network connection between the device and the host PC is broken.
2. The communication parameters configured for the device and driver do not match.
3. The response from the device took longer to receive than allowed by the Request Timeout device setting.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.
3. Increase the Request Timeout setting to allow the entire response to be handled.

Serial communications error on channel. | Error mask = <mask>.

Error Type:
Warning

Possible Cause:

1. The serial connection between the device and the host PC is broken.
2. The communications parameters for the serial connection are incorrect.

Possible Solution:

1. Investigate the error mask code and the related information.
2. Verify the cabling between the PC and the PLC device.
3. Verify that the specified communication parameters match those of the device.

See Also:
Error Mask Codes

Unable to write to address on device. | Address = '<address>'.

Error Type:
Warning

Possible Cause:

1. The connection between the device and the host PC is broken.
2. The communications parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect device ID.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communication parameters match those of the device.
3. Verify that the device ID given to the named device matches that of the actual device.

Items on this page may not be changed while the driver is processing tags.

Error Type:
Warning

Possible Cause:
An attempt was made to change a channel or device configuration while data clients were connected to the server and receiving data from the channel/device.

Possible Solution:
Disconnect all data clients from the server before making changes.

Specified address is not valid on device. | Invalid address = '<address>'.

Error Type:
Warning

Possible Cause:
A tag address has been assigned an invalid address.

Possible Solution:

Modify the requested address in the client application.

Address '<address>' is not valid on device '<name>'.

Error Type:
Warning

This property may not be changed while the driver is processing tags.

Error Type:
Warning

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

Error Type:
Warning

Possible Cause:

1. The connection between the device and the host PC is broken.
2. The communications parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect device ID.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communication parameters match those of the device.
3. Verify that the device ID given to the named device matches that of the actual device.

Socket error occurred connecting. | Error = <error>, Details = '<information>'.

Error Type:
Warning

Possible Cause:
Communication with the device failed during the specified socket operation.

Possible Solution:
Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

Socket error occurred receiving data. | Error = <error>, Details = '<information>'.

Error Type:
Warning

Possible Cause:

Communication with the device failed during the specified socket operation.

Possible Solution:
Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

Socket error occurred sending data. | Error = <error>, Details = '<information>'.

Error Type:
Warning

Possible Cause:
Communication with the device failed during the specified socket operation.

Possible Solution:
Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

Socket error occurred checking for readability. | Error = <error>, Details = '<information>'.

Error Type:
Warning

Possible Cause:
Communication with the device failed during the specified socket operation.

Possible Solution:
Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

Socket error occurred checking for writability. | Error = <error>, Details = '<information>'.

Error Type:
Warning

Possible Cause:
Communication with the device failed during the specified socket operation.

Possible Solution:
Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

%s |

Error Type:
Informational

<Name> Device Driver '<name>'

Error Type:
Informational

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

W