Siemens S7 Plus Ethernet Driver

Table of Contents

Siemens S7 Plus Ethernet Driver 1
Table of Contents 2
Welcome to the Siemens S7 Plus Ethernet Driver Help Center 3
Overview 3
Setup 4
Channel Properties – General 5
Tag Counts 5
Channel Properties – Ethernet Communications 6
Channel Properties – Write Optimizations 6
Channel Properties – Advanced 7
Device Properties – General 7
Operating Mode 8
Tag Counts 8
Device Properties – Scan Mode 8
Device Properties – Timing 9
Device Properties – Auto-Demotion 10
Device Properties – Tag Generation 10
Device Properties – Communications 12
Device Properties – Redundancy 13
Optimizing Communications 14
Data Types Description 15
Data Type Mapping 16
Symbolic Address Descriptions 18
Event Log Messages 19
Unable to read tag. | Tag address = '<address>', 19
Read request failed. | 19
Unable to write to tag. | Tag address = '<address>', 19
Write request failed. | 19
No tag generated for the node. | Node address = '<address>', 20
Array definition updated to match change detected in the controller. | Tag address = '<address>'. 20
PLC details | IP = '<address>', Port = '<port>', PLC family = '<family>', Type = '<type>', MLFB = '<mlfb>', Firmware = '<firmware>'. 20
Siemens communication library | Version = '<version>', Build date = '<build date>'. 20
Loading symbols from PLC. | 20
Secure communication certificates loaded but invalid certificate(s) detected. | Loaded = '<count>', Expired = '<count>', Invalid signature = '<count>'. 21
Successfully loaded secure communication certificates. | Loaded = '<count>'. 21
Connection details | 21
Reason Explanations 21
Appendix – Reloading Symbols 30
Index 31

Welcome to the Siemens S7 Plus Ethernet Driver Help Center

This help center is the user documentation for Kepware Siemens S7 Plus Ethernet Driver. This help center is updated regularly to reflect the latest functionality and information.

Overview
What is the Siemens S7 Plus Ethernet Driver?

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

Optimizing Communications
How do I get the best performance from the driver?

Data Types Description
What data types does this driver support?

Address Descriptions
How do I address a data location on a Siemens device?

Event Log Messages
What messages does the Siemens S7 Plus Ethernet Driver produce?

Version 1.046

Overview

The Siemens S7 Plus Ethernet Driver provides a reliable way to connect Siemens Ethernet devices to OPC client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Siemens S7 1200 and 1500 PLCs supporting symbolic addressing and provides access to optimized blocks.

The driver requires no additional libraries or hardware. A standard Ethernet card is needed.

Setup

This section contains setup information for connecting to the S7-1200 and S7-1500 controllers.

Communication Protocol

S7 Comm Plus

Supported Devices

Device must support symbolic addressing.

- S7-1200
- S7-1500

These devices have a built-in Ethernet module.

The supported devices support the following firmware versions:

Kepware Release	S7-1200	S7-1500
6.14 or higher	FW 4.6, 4.5, 4.4, 4.2.1, SPS 4.2	FW 3.0.x, 2.9.x, 2.8.1, 2.6, 2.5, SPS 2.0, <1.7
6.11 - 6.13	FW 4.5, 4.4, 4.2.1, SPS 4.2	FW 2.9.2, 2.8.1, 2.6, 2.5, SPS 2.0, <1.7

Security Protocols

OpenSSL Version 3.0.8
TLS Version 1.3 (lower for older devices)

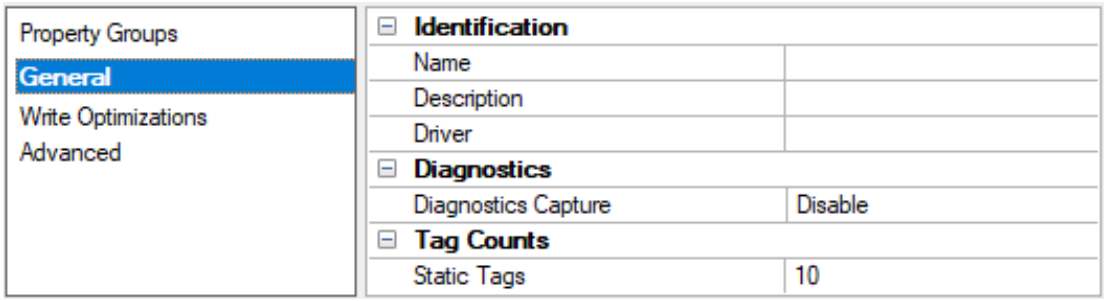
Channel and Device Limits

The maximum number of channels supported by this driver is 256. The maximum number of devices supported by this driver is 16 per channel.

See Also: *Channel Properties*, *Device Properties*, *Appendix – Siemens and Media-Level Redundancy Support*, *and the support Knowledge Base for the article about Migrating from Siemens TCP/IP to Siemens S7 Plus.*

Channel Properties – General

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



Identification

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

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

Driver: Specify the protocol / driver for this channel. Specify the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.
Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. Changes to the properties should not be made once a large client application has been developed. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.

Diagnostics

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

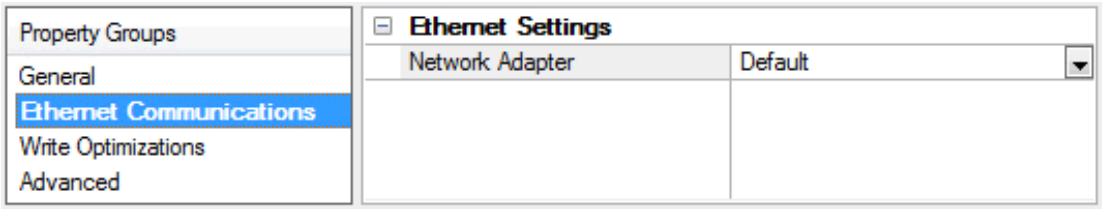
Tag Counts

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

Note: View the diagnostic information for this driver in ASCII.

Channel Properties – Ethernet Communications

Ethernet Communication can be used to communicate with devices.

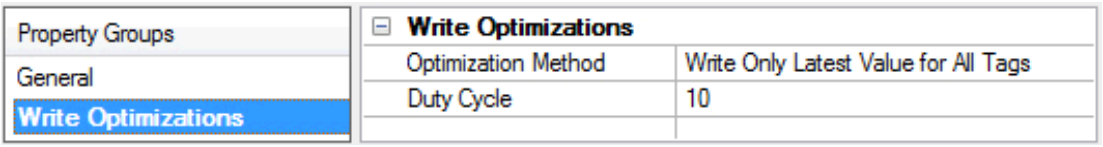


Ethernet Settings

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

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



Write Optimizations

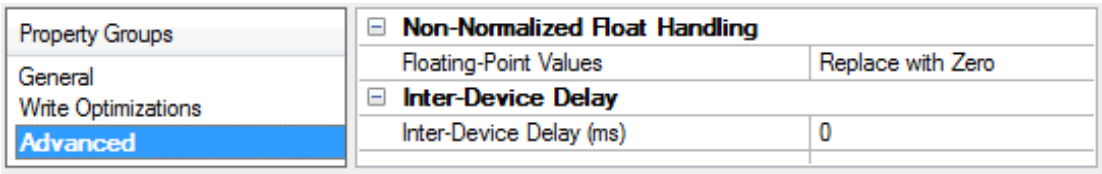
Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

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

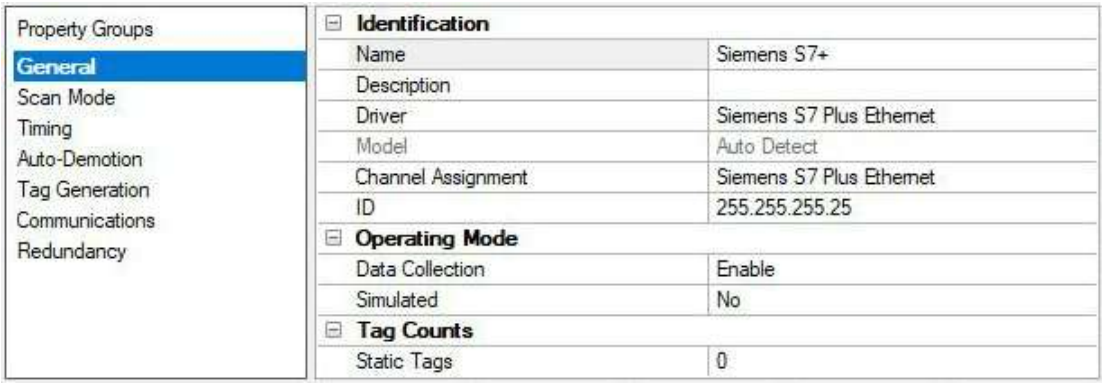
For more information on the floating-point values, refer to "How To ... Work with Non-Normalized Floating-Point Values" in the server help.

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

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

Device Properties – General

A device represents a single target on a communications channel.



Identification

Name: Specify the name of the device. It is a logical user-defined name that can be up to 256 characters long and may be used on multiple channels.
Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".
For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

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

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

Driver: Selected protocol driver for this device.

Model: The driver automatically detects the model.

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

Notes:

- 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.*
- The ID for this driver is a string representing the unique network address of the PLC, typically in the format of an IP address or a configured host name. Maximum length of this string is 63 characters.

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: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The 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.

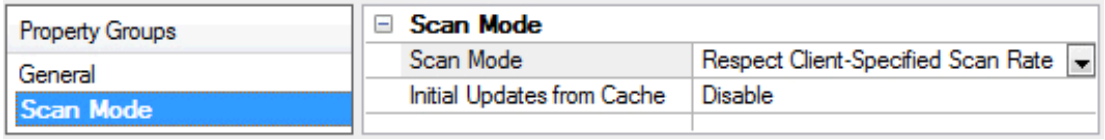
Simulation Mode is for test and simulation purposes only. It should never be used in a production environment.

Tag Counts

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

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.



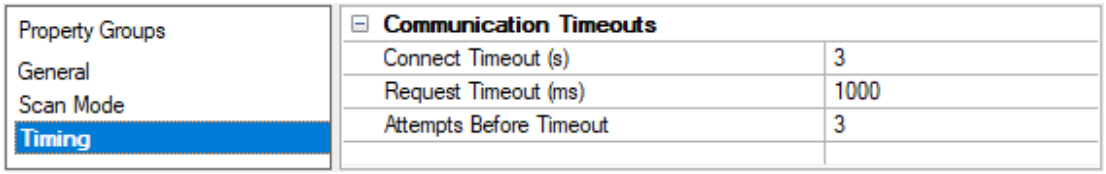
Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

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

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

Device Properties – Timing

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



Communications Timeouts

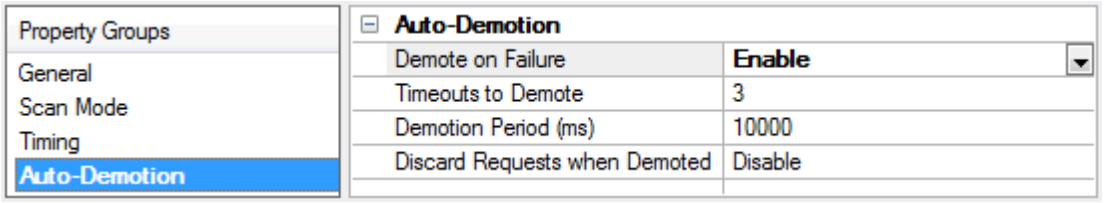
Connect Timeout: This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to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

Request Timeout: Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9999999 milliseconds (167 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

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

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

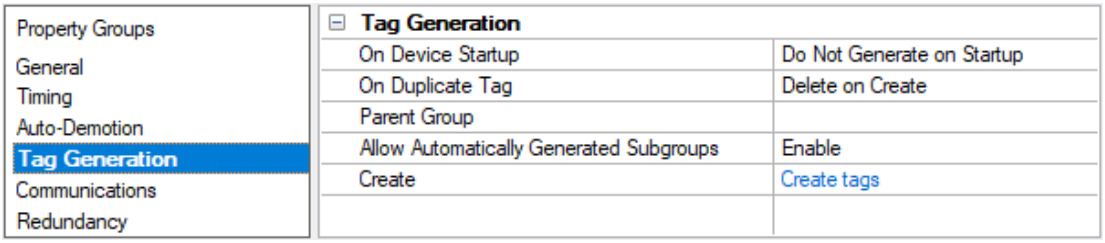
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the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

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

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

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

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

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

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

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

Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.
Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

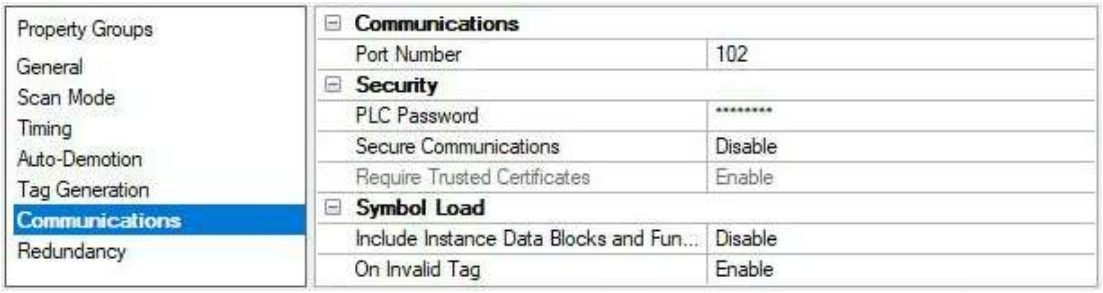
Create: Initiates the creation of automatically generated OPC tags. If the device's configuration has been modified, **Create tags** forces the driver to reevaluate the device for possible tag changes. Its ability to be accessed from the System tags allows a client application to initiate tag database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

Notes:

1. This driver replaces any backslash in a group name with an underscore.
2. Tags are generated only when all three HMI properties (Accessible, Writable, and Visible) are enabled or both Accessible and Visible are enabled in the TIA Portal programming software. Those Properties are:
   - Accessible from HMI/OPC UA/Web API.
   - Writable from HMI/OPC UA/Web API.
   - Visible in HMI Engineering
3. Symbols are reloaded from the controller on each attempt to generate tags.
4. Some Siemens data types are not supported by the driver. No tag is generated for those nodes. An event log message is posted to warn of nodes with no tag. No message is reported for structures not supported in a single tag.

Device Properties – Communications

If a protection-level password is configured in the PLC, the PLC Password property should be configured with a password that allows at least read access.
See the PLC programming software for more details on configuring access levels.



Communications

Port Number: Specify the TCP/IP port number configured for the device. The valid range is 1 to 65535. The default is 102.
Tip: It is recommended that the default port be used for most applications, where the server and the PLC exist on the same network. For an application using the Internet through firewalls and advanced routers, the port number can be changed to allow these operations to occur. In most cases, however, the PLC only accepts a connection on port 102 and may require router forwarding.

Security

PLC Password: Specify the password for the required access level configured in the PLC. The maximum length supported for a password is 256 characters. Wide characters are supported.

Secure Communications: Specify to establish a secure connection with the PLC. Secure PG / PC / HMI communication is available in PLCs configured with TIA Portal V17 or higher. The default is Disable. However, it is recommended to Enable secure communications if the PLC supports it. When enabled, all information is encrypted.

Require Trusted Certificates: Specify that the secure connection requires trusted certificates. This property is only available if the Secure Communications is enabled. The default is Enable.
Notes:

- If enabling Require Trusted Certificates, a Siemens PLC certificate must be manually imported into the Siemens S7 Plus Ethernet Trust Store.
- To manually import and configure certificates, use the **Certificate Store** tab in the Server Administration tool. Select the Siemens S7 Plus Ethernet feature. More details may be found in the server help document under the **Administration | Settings | Certificate Store** section.

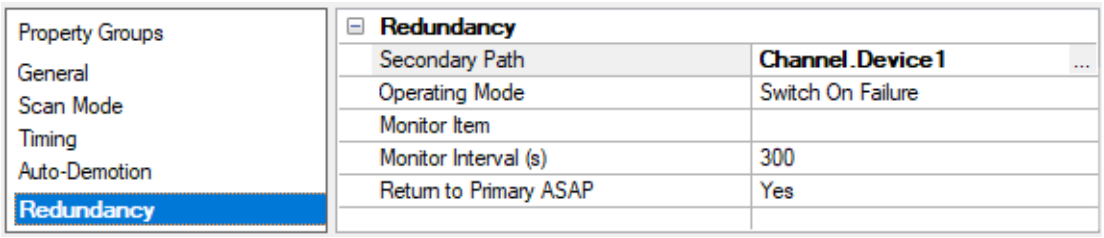
- The server runtime must be restarted when a new certificate is imported.

Symbol Load

Include Instance Data Blocks and Function Blocks: When enabled, symbol load includes symbols for instance data blocks and function blocks. The default is Disabled.

On Invalid Tag: When enabled, invalid tags cause a symbol download from the device when encountered. The default is Enabled.
Note: The device-level System tag "_ForceSymbolReload" forces a symbol reload from the device when written to with a non-zero value.

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.

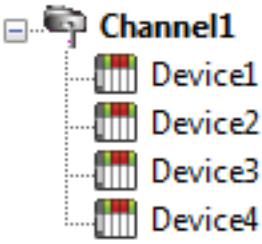
Note: Please note the following about the interaction between the Siemens S7 Plus Ethernet Driver and the Media-Level Redundancy Plug-In:

- The redundancy settings for Siemens devices allow a pair of S7-15xxR or S7-15xxH devices to provide their own redundancy system (independent of Kepware).
- If these redundancy settings are configured, the Media-Level Redundancy Plug-In does not work (and will not override the Siemens redundancy pair).
- If these settings are not configured and the PLC does not use the Siemens redundancy or IP address sharing, the controllers behave like other S7-15xx devices and should be compatible with the Media-Level Redundancy Plug-In.

Optimizing Communications

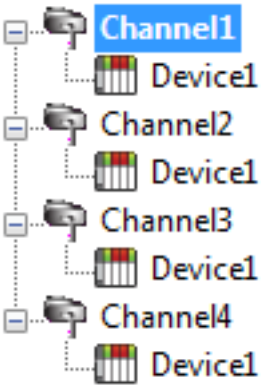
The Siemens S7 Plus Ethernet Driver was designed to provide the best performance with the least amount of impact on the system's overall performance. While the Siemens S7 Plus Ethernet Driver is fast, there are a couple of guidelines that can be used to optimize the application and gain maximum performance.

This server refers to communications protocols like Siemens S7 Plus Ethernet 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 can then be defined under that channel. It is not recommended for the series of devices to use the same device IP. It is preferred for projects to only have one physical connection per device. Each of these devices represents a single Siemens Ethernet controller from which data will be collected. Although this approach to defining the application provides a high level of performance, it does not take full advantage of the Siemens S7 Plus Ethernet Driver or the network. An example of how the application may appear when configured using a single channel is shown below.



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

If the Siemens S7 Plus Ethernet Driver could only define one channel, the example above would be the only option available; however, the driver can define up to 256 channels, each with support for 16 devices. 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 is shown below.



Each device can be defined under its own channel; however, it is not recommended to create multiple channels with devices using the same IP. It is preferred for projects to only have one physical connection per device. In this configuration, a single path of execution is dedicated to the task of gathering data from each device. 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 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.

Although the maximum number of channels is 256, the device ultimately determines the number of allowed connections. This constraint comes from the fact that some devices cannot support so many connections. For these devices, the maximum number of channels defined should equal the maximum number of connections allowed. For devices that support more connections, the maximum of 256 channels should be defined, with devices spread evenly over these channels. If a device's connection limit is exceeded, bad tag quality may occur. *Refer to Siemens documentation for maximum allowed connections.*

Tip: Using arrays can help improve performance.

Tip: For best performance, keep the total number of tags down. Remove tags in the server if the client is not using them. Memory consumption increases as the number of tags increase. It is recommended to keep server_runtime private bytes from approaching a size of 800 MB. Private bytes exceeding this can result in bad behavior and "Out of memory" event log error messages.

For more information on device connections, refer to Device Properties.

Data Types Description

The following data types are supported when creating tags and for tag reads and writes:

Data Type	Description
Default	The tag is assigned one of the data types below based upon the S7 data type. This data type is assigned upon the first successful read from the device. ***See Also***: *Data Type Mapping*
Boolean	Single bit Range: 0 to 1
Byte	Unsigned 8-bit value Range: 0 to 255
Char	Signed 8-bit value Range: -128 to 127
Word	Unsigned 16-bit value Range: 0 to 65,535
Short	Signed 16-bit value Range: 32,768 to 32,767
DWord	Unsigned 32-bit value Range: 0 to 4,294,967,295
Long	Signed 32-bit value Range: -2,147,483,648 to 2,147,483,647
QWord	Unsigned 64-bit value Range: 0 to 18,446,744,073,709,551,615
LLong	Signed 64-bit value Range: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Float	32-bit floating point value Range: ±1.17154943508222875E-38 to ±3.4028234663852886E+38 (normalized) 0 ±1.4012984643248170E-38 to ±1.1754942106924411E-38 (denormalized)
Double	64-bit floating point value Range: ±2.2250738585072014E-308 to ±1.7976931348623157+308 (normalized) 0 ±4.9406564584124654E-324 to ±2.2250738585072010E-308 (denormalized)
String	Null-terminated ASCII string The minimum size of strings is 1. The maximum size of strings is 254 characters. Strings are considered narrow (8 bits in length). This data type supports ASCII and extended-ASCII characters.

See Also: Data Type Mapping

Data Type Mapping

This table shows data types supported by this driver with data types on the controller on the left and their server equivalents on the right. If multiple server data types are supported for an S7 Data Type, the default data type is shown in bold. Unless stated otherwise, these data types are supported by both the S7-1200 and S7-1500 devices. All data types below are also supported as Read / Write unless otherwise indicated.

S7 Data Type	Server Data Types
Bool	Boolean, Boolean Array
Byte	Byte, Byte Array
Char	**String**, Byte, Char, String Array, Byte Array, Char Array **Notes**: • For the String data type, each Char tag is read in as a 1-character string (ex. 65 is 'A'). This means it is possible to write a string value to characters (ex. 'A' can be written instead of 65), assuming the data type is string. If a string of length greater than 1 is written to the char tag, only the first character is written to the device (ex. Writing 'foo' to a char tag truncates and writes 'f'). • For Byte data type, the value is the ASCII / Extended ASCII value in decimal format, ranging from 0 – 255 (ex. 'A' is 65). • For Char data type, the value is the ASCII / Extended ASCII value in decimal format, ranging from -128 – 127 (ex. 'A' is 65).
INT	Short, Short Array
DINT	Long, Long Array
LINT	LLong **Notes**: This type is only supported by S7-1500 devices. Array of the LINT data type is not supported (as of this release).
Word	Word, Word Array
DWord	DWord, DWord Array
LWord	QWord **Notes**: This type is only supported by S7-1500 devices. Array of the LWORD data type is not supported (as of this release).
REAL	Float, Float Array
LREAL	Double
SINT	Char, Char Array
UDINT	DWord, DWord Array
UINT	Word, Word Array
USINT	Byte, Byte Array
ULINT	QWord **Note**: This type is only supported by S7-1500 devices.Array of the ULINT data type is not supported (as of this release).
STRING	String, String Array
Date And Time (DT)	String **Note**: Date_And_Time is a read-only tag. It displays in standard time with the following format: mm/dd/yyyy hh/min/ss tt. Example of a read on this tag: 05/21/1991 05:30:21 PM. Not including leap years, the minimum value for this type is 01/01/1990 12:00:00 AM and the maximum value is 12/31/2089 11:59:59 PM. **Note**: This type is only supported by S7-1500 devices. Array of the DT data type is not supported (as of this release).
Time Of Day (TOD)	String **Note**: Time_Of_Day indicates the elapsed time from midnight. It displays in the following format: hh:mm:ss.msc. An example of a read on this tag: 23:31:21.999. The minimum value for this tag is: 00:00:00.000 and the maximum value for this tag

S7 Data Type	Server Data Types
	is 23:59:59.999. Array of the TOD data type is not supported (as of this release).
Time	**String**, Long **Note**: Time is displayed as a string in the following format: ddD_hhH_mmM_ssS_hhhMS. An example of a read on this tag: 21D_15H_12M_60S_333MS. The minimum value for this tag is -24D_20H_31M_23S_648MS and the maximum value for this tag is 24D_20H_31M_23S_647MS. **Note**: Array of the Time data type is not supported (as of this release).
S5Time	Long **Note**: The range of this type is 0 to 9990000. The value stored in the server represents S5Time in milliseconds. When writing the value to the server, it should be written as a millisecond value. Negative values cannot be written and any value above the maximum range is automatically set to 9990000. If the value in the PLC is an invalid S5Time value, the server returns bad quality for tags accessing this type. **Note**: This type is only supported by S7-1500 devices.
Date	String **Note**: Date is displayed in the following format: yyyy-mm-dd. An example of a read on this tag: 1991-02-03. The minimum value for this tag is 1990-01-01 and the maximum value for this tag is 2169-06-06. Array of the Date data type is not supported (as of this release).
OB TOD	Short

Notes:

- The number of elements for a client array tag is assigned on the first read of that tag and equals the total number of elements for that array node in the controller. For example, given a 2D array node [2,3], the number of elements assigned to the array tag is 6. If the number of elements for the array node changes in the controller, the driver updates the client array tag with the new element count. In the example above, if the array node changes from a 2D [2,3] to 1D [5], the array tag is assigned a new element count of 5. If instead the array node changed from 2D [2,3] to 1D [6], there is no change in element count so there is no change to the array tag.
- Arrays of server data types are supported for the data types indicated above. The server array maximum size is 65535 elements. The controller arrays can be defined with dimensions of 1D to 6D arrays; these are mapped to a 1-dimensional client array. If arrays in the PLC are defined with more than 65535 elements, automatic tag generation does not create the array tag itself, but all the array element tags are generated. An event log message is posted stating that arrays exceeding 65535 elements are not supported. A read or write on this same array tag also fails with an event log message.

Symbolic Address Descriptions

When manually creating a tag, some restrictions apply to the symbolic address being created. The rules and data types are below.

Examples:
PLC_1.Blocks.Data_block_1.Tag1
PLC_1.Blocks.Data_block_2.Tag2

Address Syntax Rules

- The length of the address must not exceed 1350 characters.
- The address length cannot be 0.
- The symbolic address cannot contain spaces only.
- The string for the address must come with a valid data type *(listed in the Data Types Mapping)*.
- Symbolic address strings cannot have leading and trailing spaces within the name.
- If the following characters are included in a single node name, the node name must be wrapped in double quotation marks:
  . decimal point
  (open parenthesis
) close parenthesis
  [open square bracket
  ] close square bracket

  Examples:
  PLC_1.Blocks.Data_block_1.“Tag(“
  PLC_1.Blocks.”Data_block.1”.Tag
  Tip: If the PLC node name contains a double quotation mark, the double quotation mark must be escaped with a double quotation mark and the node name must be wrapped in double quotation marks.

  Example:
  “PLC””Name”.Blocks.Data_block_1.Tag

Notes:

- Reading or writing to a symbolic address that does not exist in the PLC triggers the driver to load symbols from the PLC on every read or write request. For best performance, tags that reference invalid symbolic addresses must be removed to prevent them from being included in read or write requests and continuously loading symbols.
- There is no syntax for specifying the number of rows and columns for client array tags. Since 1D – 6D array tags in the controller are flattened to 1D (one dimension) client arrays, the number of rows is fixed at 1. The driver automatically determines and assigns the number of columns to the tag at runtime.

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the OPC 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.

Tip: Messages that originate from a data source (such as third-party software, including databases) are presented through the Event Log. Troubleshooting steps should include researching those messages online and in vendor documentation.

Unable to read tag. | Tag address = '<address>',

Error Type:
Error

Possible Cause:
An error occurred while reading the tag. See reason in event log message.

Possible Solution:
Review the given reason in the Reason Explanations Section.

See Also:
Reason Explanations

Read request failed. |

Error Type:
Error

Possible Cause:
An error occurred during a read request. See reason in event log message.

Possible Solution:
Review the given reason in the Reason Explanations Section.

See Also:
Reason Explanations

Unable to write to tag. | Tag address = '<address>',

Error Type:
Error

Possible Cause:
An error occurred while writing the tag. See reason in event log message.

Possible Solution:
Review the given reason in the Reason Explanations Section.

See Also:
Reason Explanations

Write request failed. |

Error Type:
Error

Possible Cause:

An error occurred during a write request. See reason in event log message.

Possible Solution:
Review the given reason in the Reason Explanations Section.

See Also:
Reason Explanations

No tag generated for the node. | Node address = '<address>',

Error Type:
Warning

Possible Cause:
An error occurred while browsing a node. See reason in event log message.

Possible Solution:
Review the given reason in the Reason Explanations Section.

See Also:
Reason Explanations

Array definition updated to match change detected in the controller. | Tag address = '<address>'.

Error Type:
Warning

Possible Cause:
The total number of elements changed for this array in the controller.

Possible Solution:
No action required; the tag <address> was automatically assigned the new element count.

PLC details | IP = '<address>', Port = '<port>', PLC family = '<family>', Type = '<type>', MLFB = '<mlfb>', Firmware = '<firmware>'.

Error Type:
Informational

Siemens communication library | Version = '<version>', Build date = '<build date>'.

Error Type:
Informational

Loading symbols from PLC. |

Error Type:
Informational

Possible Cause:
Symbol load from PLC required. See reason in event log message.

Possible Solution:
Review the given reason in the Reason Explanations Section.

See Also:
Reason Explanations

Secure communication certificates loaded but invalid certificate(s) detected. | Loaded = '<count>', Expired = '<count>', Invalid signature = '<count>'.

Error Type:
Security

Possible Cause:

1. A certificate in the certificate store is expired.
2. A certificate in the certificate store has an invalid signature.

Possible Solution:

1. Remove the expired certificate. Import a certificate with a future expiration date.
2. Remove the invalid signed certificate. Import a certificate with a valid signature.

Successfully loaded secure communication certificates. | Loaded = '<count>'.

Error Type:
Security

Connection details |

Error Type:
Security

Possible Cause:
Successful connection established. See details appended to this message in the event log.

Reason Explanations

Some event log messages include additional information under a reason field. Click on the link for a description of the reason.

Access to path is denied
Arrays exceeding 65535 elements are not supported
Array range invalid
Could not initialize SSL
Date string contains a syntax error. Expected yyyy-mm-dd format
Device is not responding
Device is not responding; Connection closed by device
Error code = <hex error code>
Failed to resolve host name
Internal driver error occurred
Invalid value encountered on read for S5Time
Invalid value encountered on write for S5Time
New connection
Node path is invalid
Out of connections
Out of memory
Password changed
Password required
Read from this Siemens data type is not supported
Secure communication required

Symbol load setting changed
The data type of the node is not supported
Time of Day string contains a syntax error. Expected hh:mm:ss.hhh format
Time string contains a syntax error. Expected ddD_hhH_mmM_ssS_hhhMS format
Unsupported communication configuration detected
Unsupported data type for this address
Untrusted certificate
User forced reload
Write to this Siemens data type is not supported
Wrong password

Reason = Array range invalid.

Possible Cause:

The array in the controller has fewer elements than the client array and the client attempted to write to the array tag.

Possible Solution:

1. Verify the client is writing the same or fewer number of elements as the array in the controller.
2. If the controller array changed, the client should read the tag before writing to it to synchronize the array configuration.

Reason = Could not initialize SSL.

Possible Cause:

PLC programmed with TIA Portal version that does not support secure communications. Secure PG/PC and HMI communication is available in PLCs configured with TIA Portal V17 or higher.

Possible Solution:

To use secure communications the PLC project must be V17 or later.

Reason = Device is not responding: ID = <IP address>.

Possible Cause:

1. The IP address is invalid.
2. The PLC is not configured to respond.
3. The configured connect timeout is too short.
4. The configured request timeout is too short.
5. The device refused the connection due to too many connections.

Possible Solution:

1. Verify the IP address.
2. Verify the PLC is in a state to respond and loaded with a valid program.
3. Increase the configured connect timeout to allow more time for the PLC to accept the connection request.

4. Increase the configured request timeout to allow more time for the PLC to respond to the request.
5. Verify the number of connections allowed by the PLC.

Reason = Device is not responding. Connection closed by device. ID = <IP address>.

Possible Cause:

The device refused the connection due to too many connections.

Possible Solution:

Verify the number of connections allowed by the PLC.

Reason = Failed to resolve host name. Host name = <host name>.

Possible Cause:

1. The format of IP address does not follow the expected IP address with four octets.
2. The host name provided is invalid and does not resolve to an IP address.

Possible Solution:

1. Verify or correct the format of the IP address.
2. Verify or correct the string is a valid host name that resolves to an IP address.

Reason = Node path is invalid.

Possible Cause:

1. Node path does not exist in the controller.
2. Array element does not exist in the array.
3. Node path is not a member of the UDT.
4. Node path is unavailable.
5. Node path is not applicable.
6. Part of the node path requires quotation marks due to special characters (. [ ] ( )).
7. Invalid characters in array element notation.
8. Node path is an array and missing the index syntax.
9. Node path is to a UDT member, where the UDT is not an instance.
10. Node path changed or was deleted since loading the symbols.
11. Node path data type changed since loading the symbols.

Possible Solution:

1. Verify the path to the node has the correct syntax.
2. Verify the node exists in the controller.

3. Verify the path to the node isn't a complex type, like an array or UDT.

Reason = Access to path is denied.

Possible Cause:

1. Node path is read-only.
2. Node path is not accessible to the HMI.
3. Node path is protected.

Possible Solution:

1. Verify the node is accessible by HMIs.
2. If unable to write, verify the node has write permissions.

Reason = Arrays exceeding 65535 elements are not supported.

Possible Cause:

The array in the controller exceeds 65535 elements.

Possible Solution:

Reduce the array in the controller to a maximum size of 65535 elements.

Reason = Read from this Siemens data type is not supported.

Possible Cause:

Driver does not support reading nodes defined with unsupported Siemens data type.

Possible Solution:

For a list of valid Siemens data types, see Data Type Mapping. If issue can't be resolved, contact technical support.

Reason = Write to this Siemens data type is not supported.

Possible Cause:

Driver does not support writing to nodes defined with the Siemens Date and Time data type.

Possible Solution:

For a list of valid Siemens data types, see Data Type Mapping. If issue can't be resolved, contact technical support.

Reason = Time of Day string contains a syntax error. Expected hh:mm:ss.hhh format.

Possible Cause:

1. The syntax of the value is not in the correct format.
2. The hours, minutes, seconds, or milliseconds part is out of range.

Possible Solution:

1. Format the value of the string to be written as hh:mm:ss.hhh and retry the write.
2. Verify the hours, minutes, seconds, and milliseconds are each within their expected range.

Reason = Unsupported data type for this address. Data type = <data type>.

Possible Cause:

The data type on the tag does not match the type of the node.

Possible Solution:

Correct the data type of the tag to match the data type of the node.

See Also: *For a list of valid Siemens data types, see Data Type Mapping*.

Reason = Date string contains a syntax error. Expected yyyy-mm-dd format.

Possible Cause:

1. The syntax of the value is not in the correct format.
2. The day, month, or year part is out of range.

Possible Solution:

1. Format the value of the string be written as yyyy-mm-dd and retry the write.
2. Verify the day, month, and year are each within their expected range.

Reason = Time string contains a syntax error. Expected ddD_hhH_mmM_ssS_hhhMS format.

Possible Cause:

1. The syntax of the value is not in the correct format.
2. The day, hour, minute, second, or milliseconds part is out of range.

Possible Solution:

1. Format the value of the string to be written as ddD_hhH_mmM_ssS_hhhMS and retry the write.
2. Verify the day, hours, minutes, seconds, and milliseconds are each within their expected range.

Reason = The data type of the node is not supported; Node type = <node type>.

Possible Cause:

The driver does not support the Siemens data type.

Possible Solution:

Contact technical support.

See Also: *For a list of valid S7 data types, see Data Type Mapping.*

Reason = Wrong password

Possible Cause:

The configured device password does not match the protection level password configured in the PLC.

Possible Solution:

Verify the configured password.

Reason = Password changed.

Possible Cause:

The password in the PLC or server project has changed.

Possible Solution:

Verify the password in the PLC and in the server project match, authentication succeeds, and updates continue.

Reason = Password required.

Possible Cause:

The PLC has a protection level password configured.

Possible Solution:

Configure a device password.

Reason = Invalid value encountered on read for S5Time. Valid range for this type is between 0 and 9990000.

Possible Cause:

The S5Time that is set in the device being accessed is incorrect.

Possible Solution:

Write a valid S5Time value to the device so the information can be read in correctly. If this does not work, contact technical support.

Reason = Invalid value encountered on write for S5Time. Valid range for this type is between 0 and 9990000.

Possible Cause:

A negative value is being written to an S5Time tag.

Possible Solution:

Enter a valid S5Time value. If this does not work, contact technical support.

Error Code = <hex error code>.

Possible Cause:

An uncommon error caused the request to fail.

Possible Solution:

1. Verify the item configuration and access rights.
2. Contact technical support.

Reason = Internal driver error occurred.

Possible Cause:

An unexpected error within the driver caused the request to fail.

Possible Solution:

1. Verify the item configuration and access rights.
2. Contact technical support.

Unsupported communication configuration detected, ID = <IP address>.

Possible Cause:

1. The PLC returned an error that is not currently handled.
2. The PLC device configuration is not supported.

Possible Solution:

If issue continues, contact technical support.

Reason = New connection.

Possible Cause:

1. A connection to the PLC is successfully established on a client read or write or tag generation request.
2. The previous connection disconnected due to network issues and a new connection to the PLC is successfully established.

Possible Solution:

Verify possible issues with network stability.

Reason = Out of connections.

Possible Cause:

The device refused the connection due to too many connections.

Possible Solution:

Verify the number of connections allowed by the PLC.

Reason = Out of memory.

Possible Cause:

The runtime process has reached the available memory capacity for an application on the system.

Possible Solution:

1. Restart the runtime.
2. Reduce the number of static tags in the project. All static tags in the project are loaded into the runtime memory regardless of whether a client accesses them. Therefore, remove any static tags that are not required.
3. Reduce the number of connections to each controller. Each connection must load the controller symbols into memory in order to read and write to them.
4. Reduce the number of channels and devices. After starting the runtime, connect a client and see if all tags with good quality. Verify the private byte size of the runtime to determine if it is near capacity for an application on the system. The size is dependent on the number connections as well as the number of tags in the project and the number of symbols in each controller.

Reason = Secure communication required.

Possible Cause:

PLC is configured to only allow secure PG/PC and HMI communication.

Possible Solution:

Enable Secure Communications Device property.

Reason = Untrusted certificate.

Possible Cause:

Missing a trusted certificate.

Possible Solution:

1. Add or replace certificate for the PLC in the Certificate Store
2. Disable **Require Trusted Certificates** device property to allow communication with PLC configured in mixed mode.

Reason = Symbol load setting changed.

Possible Cause:

The Include Instance Data Blocks and Function Blocks property has changed (located in the device properties Symbol Load group.

Possible Solution:

Symbol Load includes symbols for Instance Data Blocks and Function Blocks when the property is enabled.

Reason = User-forced reload.

Possible Cause:

A non-zero value was written to the device-level System tag "_ForceSymbolReload".

Possible Solution:

A symbol reload occurs.

Appendix – Reloading Symbols

If getting bad quality after controller project edits, reload symbols to update. Scenarios where a tag can change to bad quality because of controller project changes:

- When a tag's symbolic name in the controller is modified;
- When a tag's data type in the controller is modified;
- When a tag is removed; or
- If the Access Level of the device is changed to HMI access or below and the password in the server is not updated.

To make sure all other tags are reporting accurately, the server reloads tag information from the controller to maintain the accuracy of all other tags. When the server identifies that a tag has been modified, removed, or if an ATG occurs; it reloads tag information. All other scenarios require a manual reload of tag information. This is accomplished by executing a runtime reinitialize, which allows the server to reload tags. If a tag is still bad quality after the tag information update, consider manually correcting that tag or perform an ATG to correct tag information for many tags. In cases where tags are added in the controller, but not in the server; those tags must be manually entered into the server with correct information for it to read or write properly.

Note: Reading or writing to a symbolic address that does not exist in the PLC triggers the driver to load symbols from the PLC on every read or write request. For best performance, tags that reference invalid symbolic addresses must be removed to prevent them from being included in read or write requests and continuously loading symbols.

Index

A

Access Level 30
Address Descriptions 18
Address Syntax 18
Allow Sub Groups 11
Array definition updated to match change detected in the controller. | Tag address = '<address>'. 20
Attempts Before Timeout 9
Auto-Demotion 10

B

Boolean 15-16
Byte 15-16

C

Channel Assignment 8
Channel Properties – Advanced 7
Channel Properties – Ethernet Communications 6
Channel Properties – General 5
Channel Properties – Write Optimizations 6
Char 15-16
Communications 12
Communications Timeouts 9
Connect Timeout 9
Connection details | 21
Create 11

D

Data Collection 8
Data Type Mapping 16
Data Types Description 15
Date 17
Default 15
Delete 11
Demote on Failure 10
Demotion Period 10
Device Properties – Auto-Demotion 10

Device Properties – General 7
Device Properties – Redundancy 13
Device Properties – Tag Generation 10
Device Properties – Timing 9
Diagnostics 5
DINT 16
Discard Requests when Demoted 10
Do Not Scan, Demand Poll Only 9
Double 15-16
Driver 8
Duty Cycle 6
DWord 15-16

E

Ethernet card 3
Ethernet Settings 6
Event Log Messages 19

F

Float 15-16

G

General 7
Generate 11

I

ID 8
Identification 5, 7
Initial Updates from Cache 9
INT 16
Inter-Device Delay 7

L

LINT 16
LLong 15-16
Loading symbols from PLC. | 20
Long 15-16

LREAL 16

N

Name 7
Network Adapter 6
No tag generated for the node. | Node address = '<address>', 20
Non-Normalized Float Handling 7

O

On Device Startup 11
On Duplicate Tag 11
On Property Change 11
Operating Mode 8
Optimization Method 6
Optimizing Communications 14
Overview 3
Overwrite 11

P

Parent Group 11
PLC details | IP = '<address>', Port = '<port>', PLC family = '<family>', Type = '<type>', MLFB = '<mlfb>', Firmware = '<firmware>'. 20
PLC Password 12-13
Port Number 12
Protocol 4

Q

QWord 15-16

R

Read request failed. | 19
REAL 16
Reason = New connection 27
Reason = Out of connections 28
Reason = Out of memory 28
Reason = Secure communication required 28
Reason = Symbol load setting changed 28

Reason = Untrusted certificate 28
Reason = User forced reload 29
Reasons 21
Redundancy 13
Reloading Symbols 30
Replace with Zero 7
Request Timeout 9
Respect Tag-Specified Scan Rate 9

S

S5Time 17
S7-1200 4
S7-1500 4
S7 Comm Plus 4
Scan Mode 9
Secure communication certificates loaded but invalid certificate(s) detected. | Loaded = '<count>', Expired = '<count>', Invalid signature = '<count>'. 21
Security 12
Short 15-17
Siemens communication library | Version = '<version>', Build date = '<build date>'. 20
Simulated 8
SINT 16
String 15-16
Successfully loaded secure communication certificates. | Loaded = '<count>'. 21
Symbol Load 13
symbolic name 30

T

Tag Counts 5, 8
Tag Generation 10
Time 17
Timeouts to Demote 10
Timing 9
TOD 16

U

UDINT 16
UINT 16
ULINT 16
Unable to read tag. | Tag address = '<address>', 19

Unable to write to tag. | Tag address = '<address>', 19
Unmodified 7
Unsupported communication configuration detected, ID = '<IP address>', 27
USINT 16

W

Word 15-16
Write All Values for All Tags 6
Write Only Latest Value for All Tags 6
Write Only Latest Value for Non-Boolean Tags 6
Write request failed. | 19