Alstom Redundant Ethernet Driver

Table of Contents

Alstom Redundant Ethernet Driver

Help version 1.010

CONTENTS

Overview

The Alstom Redundant Ethernet Driver provides a reliable way to connect Non-Vital Custom Modbus TCP/IP Protocol with Alstom Redundancy Management Interface devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP and countless custom applications.

Setup

Supported Models

Alstom iVPI

Channel and Device Limits

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

Connect Timeout

Specify the amount of time that the driver will wait for a connection to be made with a device. The connect time may vary with each connection attempt (depending on network load). The default setting is 3 seconds. The valid range is 1 to 30 seconds.

Request Timeout

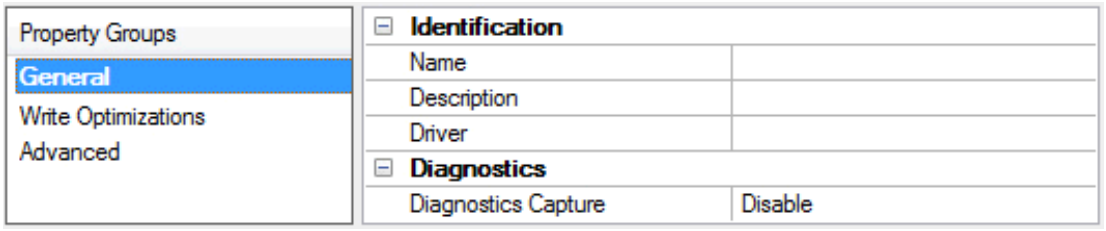
Specify the amount of time that the driver will wait for a response from the device before giving up and going on to the next request. Long timeouts will only affect performance if a device is not responding. The valid range is 100 to 30000 milliseconds. The default setting is 1000 milliseconds.

Retry Attempts

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

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.

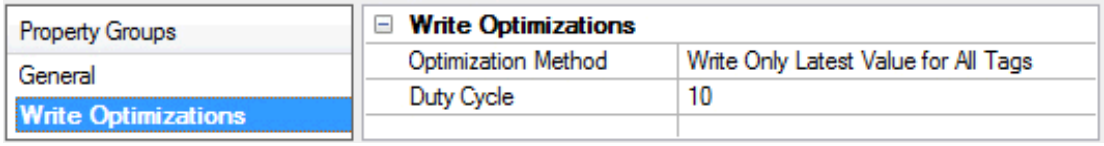
Note: This property is not available if the driver does not support diagnostics.

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

Channel Properties — 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

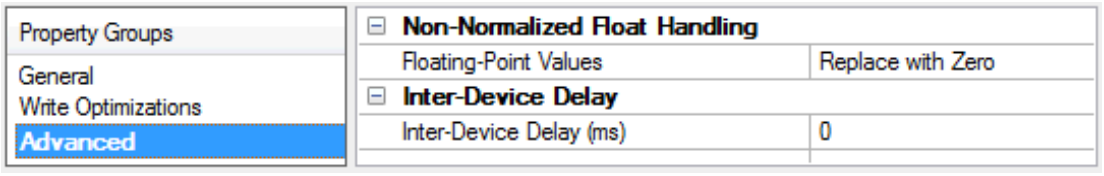
Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

- **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.

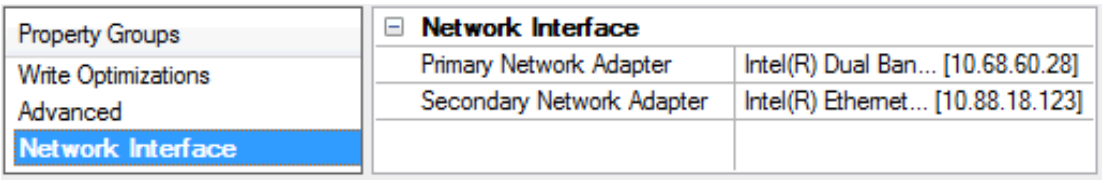
Inter-Device Delay: Specify the amount of time the communications channel waits to send new requests to the next device after data is received from the current device on the same channel. Zero (0) disables the delay.

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

Channel Properties — Network Interface

When configuring the Alstom Redundant Ethernet Driver, it is recommended that two physical network adapters be used to properly utilize the redundant system. The Network Interface property group specifies the network adapters that will be used for the primary and the secondary networks. The selections available depend on the network configuration settings, the number of unique network adapters that are installed in the PC, and the number of unique IPs that are assigned to the adapters.

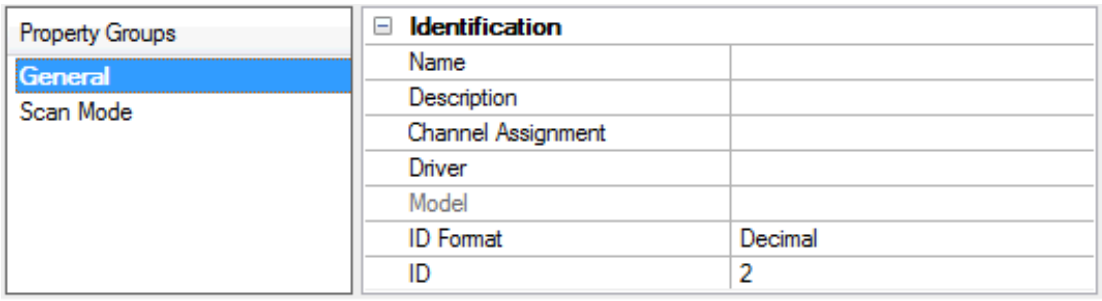
Note: If more than one network adapter is available, the primary and secondary network adapters must be unique.



Note: Changes made to the network configuration are respected by the server on startup.

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.

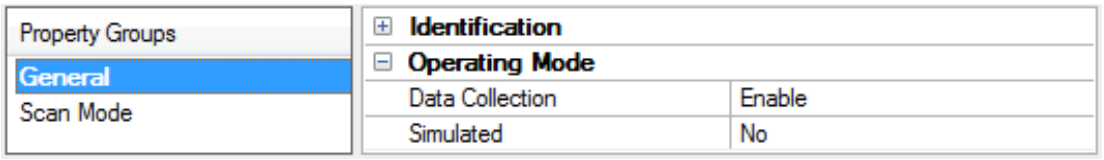
Channel Assignment: User-defined name of the channel to which this device currently belongs.

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.

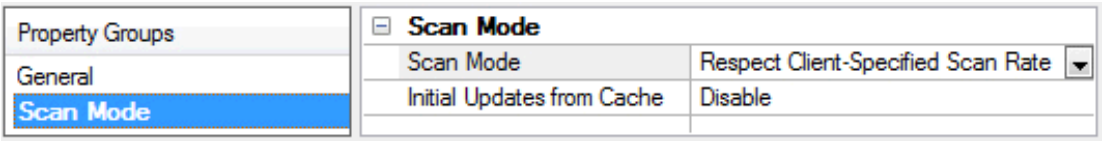
Notes:

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:

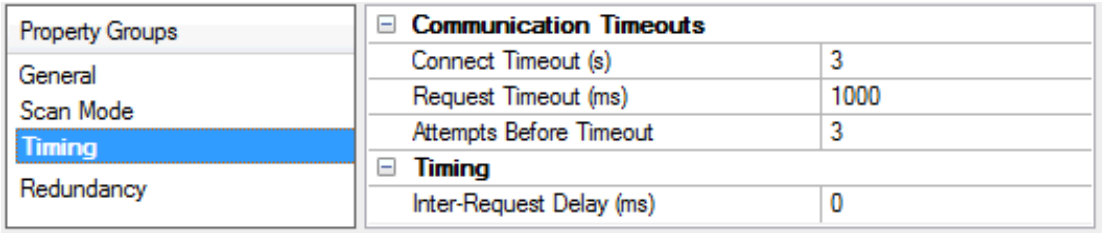
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server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

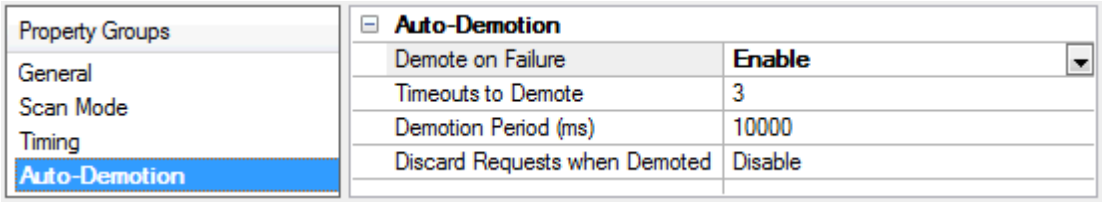
Timing

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.

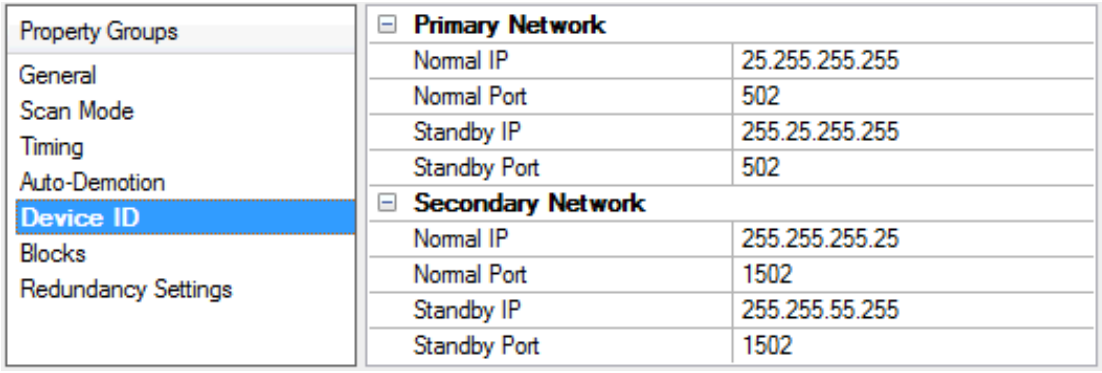
Timeouts to Demote: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

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 — Device ID

The Device ID property group specifies the primary and secondary networks' IP and Port. It consists of four unique IP addresses that are configured in the Non-Vital System Processor (NVSP) boards. The device contains two NVSP boards: Normal and Standby. Each NVSP board has an Ethernet connection to both the primary and secondary networks.



The following properties are specified for both the **Primary Network** and **Secondary Network**:

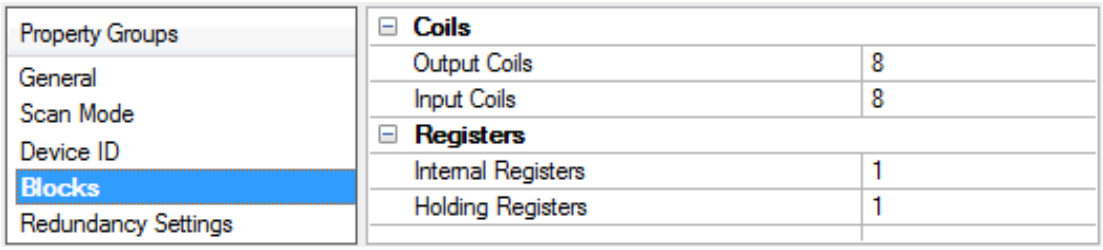
Normal IP: Specify the unique IP address the remote device is configured to use. IPs consist of four values that are separated by periods. The valid range of each value is 0 to 255.

Normal Port: Specifies the port number the remote device is configured to use. The valid range is 0 to 65535. The default setting for the primary network is 502.

Standby IP: Specify the unique IP address the remote device is configured to use. IPs consist of four values that are separated by periods. The valid range of each value is 0 to 255.

Standby Port: Specifies the port number the remote device is configured to use. The valid range is 0 to 65535. The default setting for the secondary network is 1502.

Device Properties — Blocks



Output Coils: Specify the coil block size value. Coils can be read from 8 to 2000 points (bits) at a time. The default setting is 8.

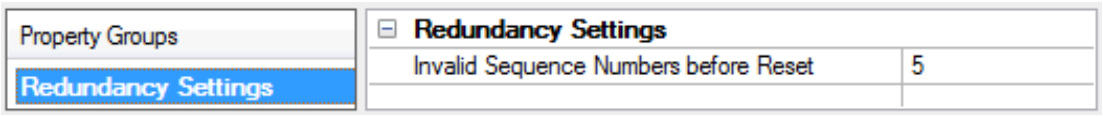
Input Coils: Specify the coil block size value. Coils can be read from 8 to 2000 points (bits) at a time. The default setting is 8.

Internal Registers: Specify the register value. Registers can be read from 1 to 120, standard registers (16 bit) at a time. The default setting is 1.

Holding Registers: Specify the register value. Registers can be read from 1 to 120, standard registers (16 bit) at a time. The default setting is 1.

Device Properties — Redundancy Settings

The communication header utilizes sequence numbers to determine whether the data received is valid.



Invalid Sequence Numbers before Reset Specify the number of invalid sequences received before the device resets. When an unexpected sequence number is received, subsequent attempts are made until the reset value is reached. At that point, a request is issued by the driver to the device to reset the sequence number. Separate sequence numbers are used for normal and standby boards. The valid range is 1 to 5 attempts. The default setting is 5.

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 bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
String	Null terminated ASCII string Includes Hi-Lo Lo-Hi byte order selection.
Double*	64-bit floating point value

Data Type	Description
	The driver interprets four consecutive registers as a double precision value by making the last two registers the high DWord and the first two registers the low DWord.
Double Example	If register 40001 is specified as a double, bit 0 of register 40001 would be bit 0 of the 64-bit data type and bit 15 of register 40004 would be bit 63 of the 64-bit data type.
Float*	32-bit floating point value The driver interprets two consecutive registers as a single precision value by making the last register the high word and the first register the low word.
Float Example	If register 40001 is specified as a float, bit 0 of register 40001 would be bit 0 of the 32-bit data type and bit 15 of register 40002 would be bit 31 of the 32-bit data type.

* The descriptions assume the default; that is, first DWord low data handling of 64-bit data types and first word low data handling of 32-bit data types.

Address Descriptions

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

Decimal Addressing

Address	Range	Data Type	Access	Address Limit
0xxxx	0-9999	**Boolean**	Read/Write	800 Bits
1xxxx	0-9999	**Boolean**	Read Only	800 Bits
3xxxx	0-9999 0-9998 0-9996	**Word**, Short, BCD Float, DWord, Long, LBCD Double	Read Only Read Only Read Only	62 Registers 62 Registers 62 Registers
4xxxx	0-9999 0-9998 0-9996	**Word**, Short, BCD Float, DWord, Long, LBCD Double	Read/Write Read/Write Read/Write	62 Registers 62 Registers 62 Registers

Hexadecimal Addressing

Address	Range	Data Type	Access	Address Limit
H0xxxx	0-270F	**Boolean**	Read/Write	800 Bits
H1xxxx	0-270F	**Boolean**	Read Only	800 Bits
H3xxxx	0-270F 0-270E 0-270C	**Word**, Short, BCD Float, DWord, Long, LBCD Double	Read Only Read Only Read Only	62 Registers 62 Registers 62 Registers
H4xxxx	0-270F 0-270E 0-270C	**Word**, Short, BCD Float, DWord, Long, LBCD Double	Read/Write Read/Write Read/Write	62 Registers 62 Registers 62 Registers

Error Descriptions

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

Address Validation

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

Device Status Messages

Device <device name> is not responding
Unable to bind to adapter: <network adapter name>. Connect failed
Unable to create a socket connection for device <device name>
Unable to write to address <address> on device <device>: Device responded with exception code <code>

Device Specific Messages

Bad address in block [x to y] on device <device name>
Bad array spanning [<address> to <address>] on device <device name>
Bad received length [x to y] on device <device name>
Failure to initiate 'winsock.dll' [Error: <winsock error>]

See Also: Modbus Exception Codes

Modbus Exception Codes

The following data is from Modbus Application Protocol Specifications documentation.

Code Dec/Hex	Name	Meaning
01/0x01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type; for example, because it is unconfigured and is being asked to return register values.
02/0x02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 will generate exception 02.
03/0x03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04/0x04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
05/0x05	ACKNOWLEDGE	The slave has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed.
06/0x06	SLAVE DEVICE BUSY	The slave is engaged in processing a long duration program command. The master should retransmit the message later when the slave is free.
07/0x07	NEGATIVE ACKNOWLEDGE	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 decimal. The master should request diagnostic or error information from the slave.
08/0x08	MEMORY PARITY ERROR	The slave attempted to read extended memory but detected a parity error in the memory. The master can retry the request, but service may be required on the

Code Dec/Hex	Name	Meaning
		slave device.
10/0x0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. This usually means that the gateway is misconfigured or overloaded.
11/0x0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways indicates that no response was obtained from the target device. This usually means that the device is not present on the network.

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

Error Type:
Warning

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

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

Array size is out of range for address <address>

Error Type:
Warning

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

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

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

Error Type:
Warning

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

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

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

Error Type:
Warning

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

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

Device address <address> contains a syntax error

Error Type:

Warning

Possible Cause:
An invalid tag address has been specified in a dynamic request.

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

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

Error Type:
Warning

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

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

Device address <address> is Read Only

Error Type:
Warning

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

Solution:
Change the access mode in the client application.

Missing address

Error Type:
Warning

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

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

Device <device name> is not responding

Error Type:
Serious

Possible Cause:

1. The connection between the device and the Host PC is broken.
2. The communication properties for the connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device property.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the specified communication properties match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.
4. Increase the Request Timeout property so that the entire response can be handled.

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

Error Type:
Warning

Possible Cause:
Since the specified network adapter cannot be located in the system device list, it cannot be bound to for communications. This usually occurs 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 falls back to the default adapter.

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

Unable to create a socket connection for device <device name>

Error Type:
Warning

Possible Cause:
The server was unable to establish a TCP/IP socket connection to the specified device. It will continue to attempt connection.

Solution:

1. Verify that the device is online.
2. Verify that the device IP is within the subnet of the IP to which the server is bound. Alternatively, verify that a valid gateway is available that allows a connection the other network.

Unable to write to address <address> on device <device>: Device responded with exception code <code>

Error Type:

Warning

Possible Cause:
See **Modbus Exception Codes** for a description of the exception code.

Solution:
See **Modbus Exception Codes**.

Bad address in block [x to y] on device <device name>

Error Type:
Fatal addresses falling in this block.

Possible Cause:
This error is reported when the driver attempts to read a location in a PLC that does not exist. For example, in a PLC that only has holding registers 40001 to 41400, requesting address 41405 would generate this error. Once this error is generated, the driver will not request the specified block of data from the PLC again. Any other addresses being requested that are in this same block will also go invalid.

Solution:
The client application should be modified to ask for addresses within the range of the device.

Bad array spanning [<address> to <address>] on device <device name>

Error Type:
Fatal

Possible Cause:
An array of addresses was defined that extends past the end of the address space.

Solution:
Verify the size of the device's memory space and then redefine the array length accordingly.

Bad received length [x to y] on device <device name>

Error Type:
Fatal addresses falling in this block.

Possible Cause:
The driver attempted to read a block of memory in the PLC. The PLC responded with no error, but did not provide the driver with the requested block size of data.

Solution:
Ensure that the range of memory exists for the PLC.

Failure to initiate 'winsock.dll' [Error: <winsock error>]

Error Type:
Fatal

Possible Cause:

Could not negotiate with the operating system's Winsock 1.1 functionality.

Solution:

Verify that the winsock.dll is properly installed on the system.

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W