Enron Modbus Driver

Table of Contents

Enron Modbus Driver

Help version 1.060

CONTENTS

Overview

The Enron Modbus Driver provides real-time and EFM data access. The driver configuration retrieves historical and alarm/event archive data, and also maps data in the device to the server's EFM Model (which consists of various EFM attributes such as pressure, temperature, and so forth).

Important: EFM functionality is not available in all server versions. To determine whether support is available, refer to the "Server Summary Information" topic located in the server help file.

Channel Setup

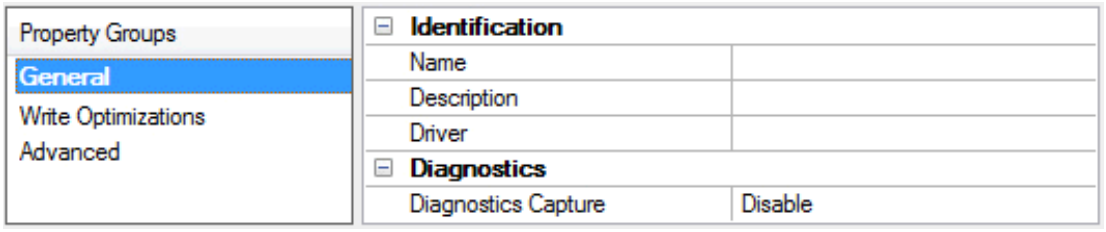
The maximum number of channels supported by this driver is 1024.

Communication Serialization

The Enron Modbus Driver supports Communication Serialization, which specifies whether data transmissions should be limited to one channel at a time.

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.

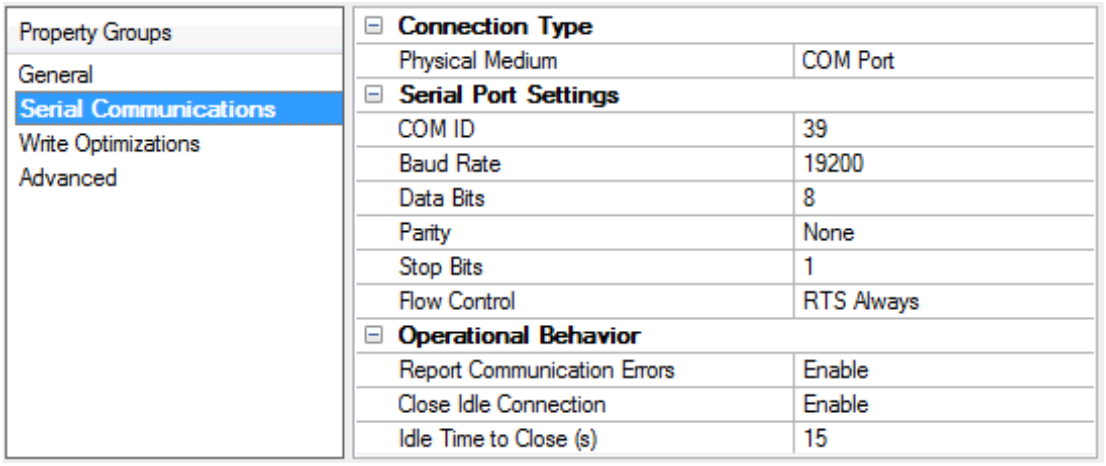
Diagnostics

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 — Serial Communications

Serial communication properties are available to serial drivers and vary depending on the driver, connection type, and options selected. Below is a superset of the possible properties.
Click to jump to one of the sections: Connection Type, Serial Port Settings or Ethernet Settings, and Operational Behavior.

Note: With the server's online full-time operation, these properties can be changed at any time. Utilize the User Manager to restrict access rights to server features, as changes made to these properties can temporarily disrupt communications.



Connection Type

Physical Medium: Choose the type of hardware device for data communications. Options include COM Port, None, Modem, and Ethernet Encapsulation. The default is COM Port.

- **None**: Select None to indicate there is no physical connection, which displays the Operation with no Communications section.
- **COM Port**: Select Com Port to display and configure the Serial Port Settings section.
- **Modem**: Select Modem if phone lines are used for communications, which are configured in the Modem Settings section.
- **Ethernet Encap.**: Select if Ethernet Encapsulation is used for communications, which displays the Ethernet Settings section.
- **Shared**: Verify the connection is correctly identified as sharing the current configuration with another channel. This is a read-only property.

Serial Port Settings

COM ID: Specify the Communications ID to be used when communicating with devices assigned to the channel. The valid range is 1 to 9991 to 16. The default is 1.

Baud Rate: Specify the baud rate to be used to configure the selected communications port.

Data Bits: Specify the number of data bits per data word. Options include 5, 6, 7, or 8.

Parity: Specify the type of parity for the data. Options include Odd, Even, or None.

Stop Bits: Specify the number of stop bits per data word. Options include 1 or 2.

Flow Control: Select how the RTS and DTR control lines are utilized. Flow control is required to communicate with some serial devices. Options are:

- **None**: This option does not toggle or assert control lines.
- **DTR**: This option asserts the DTR line when the communications port is opened and remains on.
- **RTS**: This option specifies that the RTS line is high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line is low. This is normally used with RS232/RS485 converter hardware.

- **RTS, DTR**: This option is a combination of DTR and RTS.
- **RTS Always**: This option asserts the RTS line when the communication port is opened and remains on.
- **RTS Manual**: This option asserts the RTS line based on the timing properties entered for RTS Line Control. It is only available when the driver supports manual RTS line control (or when the properties are shared and at least one of the channels belongs to a driver that provides this support).
  RTS Manual adds an **RTS Line Control** property with options as follows:
  - **Raise**: This property specifies the amount of time that the RTS line is raised prior to data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Drop**: This property specifies the amount of time that the RTS line remains high after data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Poll Delay**: This property specifies the amount of time that polling for communications is delayed. The valid range is 0 to 9999. The default is 10 milliseconds.

Tip: When using two-wire RS-485, "echoes" may occur on the communication lines. Since this communication does not support echo suppression, it is recommended that echoes be disabled or a RS-485 converter be used.

Operational Behavior

- **Report Communication Errors**: Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection**: Choose to close the connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close**: Specify the amount of time that the server waits once all tags have been removed before closing the COM port. The default is 15 seconds.

Ethernet Settings

Note: Not all serial drivers support Ethernet Encapsulation. If this group does not appear, the functionality is not supported.

Ethernet Encapsulation provides communication with serial devices connected to terminal servers on the Ethernet network. A terminal server is essentially a virtual serial port that converts TCP/IP messages on the Ethernet network to serial data. Once the message has been converted, users can connect standard devices that support serial communications to the terminal server. The terminal server's serial port must be properly configured to match the requirements of the serial device to which it is attached. *For more information, refer to "How To... Use Ethernet Encapsulation" in the server help.*

- **Network Adapter**: Indicate a network adapter to bind for Ethernet devices in this channel. Choose a network adapter to bind to or allow the OS to select the default.
  Specific drivers may display additional Ethernet Encapsulation properties. For more information, refer to Channel Properties — Ethernet Encapsulation.

Modem Settings

- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific modem properties.

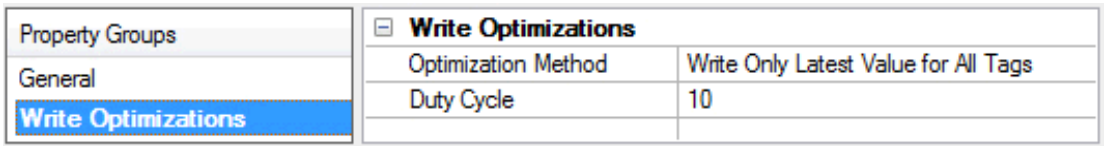
- **Auto-Dial**: Enables the automatic dialing of entries in the Phonebook. The default is Disable. *For more information, refer to "Modem Auto-Dial" in the server help.*
- **Report Communication Errors**: Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection**: Choose to close the modem connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close**: Specify the amount of time that the server waits once all tags have been removed before closing the modem connection. The default is 15 seconds.

Operation with no Communications

- **Read Processing**: Select the action to be taken when an explicit device read is requested. Options include Ignore and Fail. Ignore does nothing; Fail provides the client with an update that indicates failure. The default setting is Ignore.

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:

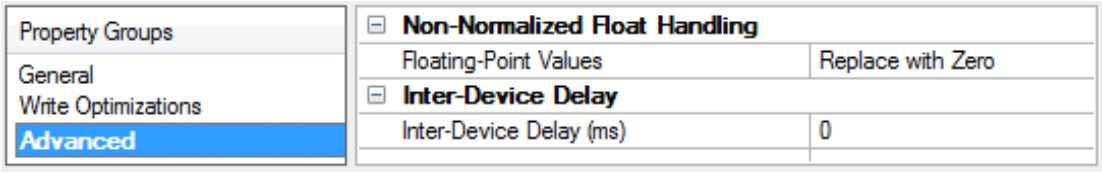
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing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

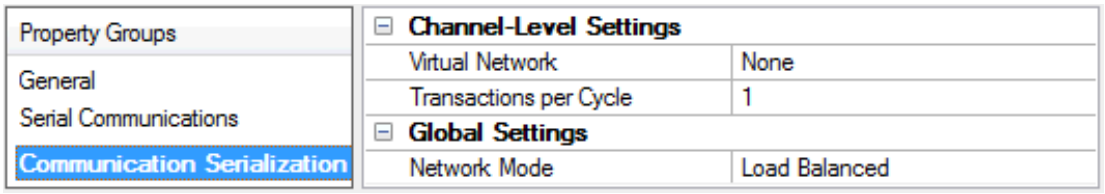
Note: This property is not available for all drivers, models, and dependent settings.

Channel Properties — Communication Serialization

The server's multi-threading architecture allows channels to communicate with devices in parallel. Although this is efficient, communication can be serialized in cases with physical network restrictions (such as

Ethernet radios). Communication serialization limits communication to one channel at a time within a virtual network.

The term "virtual network" describes a collection of channels and associated devices that use the same pipeline for communications. For example, the pipeline of an Ethernet radio is the master radio. All channels using the same master radio associate with the same virtual network. Channels are allowed to communicate each in turn, in a "round-robin" manner. By default, a channel can process one transaction before handing communications off to another channel. A transaction can include one or more tags. If the controlling channel contains a device that is not responding to a request, the channel cannot release control until the transaction times out. This results in data update delays for the other channels in the virtual network.



Channel-Level Settings

Virtual Network: This property specifies the channel's mode of communication serialization. Options include None and Network 1 - Network 500. The default is None. Descriptions of the options are as follows:

- **None**: This option disables communication serialization for the channel.
- **Network 1 - Network 500**: This option specifies the virtual network to which the channel is assigned.

Transactions per Cycle: This property specifies the number of single blocked/non-blocked read/write transactions that can occur on the channel. When a channel is given the opportunity to communicate, this is the number of transactions attempted. The valid range is 1 to 99. The default is 1.

Global Settings

- **Network Mode**: This property is used to control how channel communication is delegated. In **Load Balanced** mode, each channel is given the opportunity to communicate in turn, one at a time. In **Priority** mode, channels are given the opportunity to communicate according to the following rules (highest to lowest priority):
  - Channels with pending writes have the highest priority.
  - Channels with pending explicit reads (through internal plug-ins or external client interfaces) are prioritized based on the read's priority.
  - Scanned reads and other periodic events (driver specific).

  The default is Load Balanced and affects *all* virtual networks and channels.

Devices that rely on unsolicited responses should not be placed in a virtual network. In situations where communications must be serialized, it is recommended that Auto-Demotion be enabled.

Due to differences in the way that drivers read and write data (such as in single, blocked, or non-blocked transactions); the application's Transactions per cycle property may need to be adjusted. When doing so, consider the following factors:

- How many tags must be read from each channel?
- How often is data written to each channel?
- Is the channel using a serial or Ethernet driver?
- Does the driver read tags in separate requests, or are multiple tags read in a block?
- Have the device's Timing properties (such as Request timeout and Fail after *x* successive timeouts) been optimized for the virtual network's communication medium?

Device Setup

Supported Communication Properties

Baud Rate: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, 115200, 128000, and 256000
Parity: Odd, Even, and None
Data Bits: 5, 6, 7, and 8
Stop Bits: 1 and 2

Note: Not all of the listed configurations may be supported in every device.

Maximum Number of Devices

The maximum number of devices supported per channel is 1024.

Models

Name	Address Bytes	Station ID Range
Standard	1	0-255
Extended Station ID	2	0-65535
Enron Modbus[1]	1	0-255

1: Deprecated. Older projects using model Enron Modbus will be converted to Standard when opened.

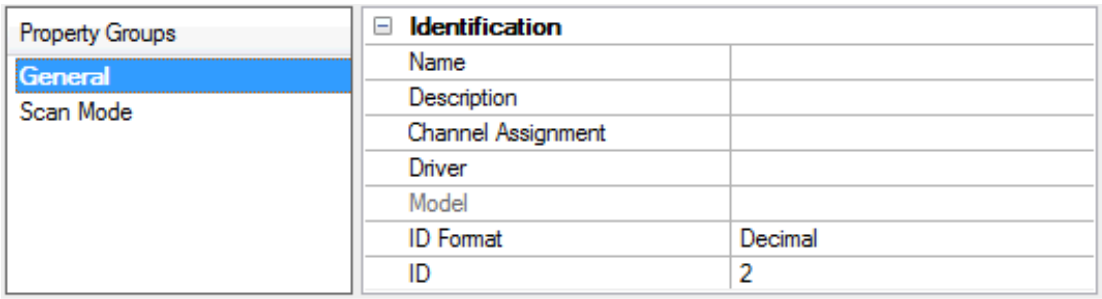
Ethernet Encapsulation

This driver supports Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server. It may be enabled through the Communications dialog in Channel Properties. *For more information, refer to the main server's help file.*

See Also: Using Ethernet Encapsulation

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.

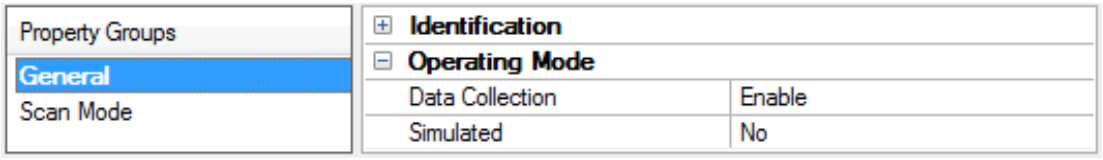
Channel Assignment: User-defined name of the channel to which this device currently belongs.

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 attemp-

ted 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.

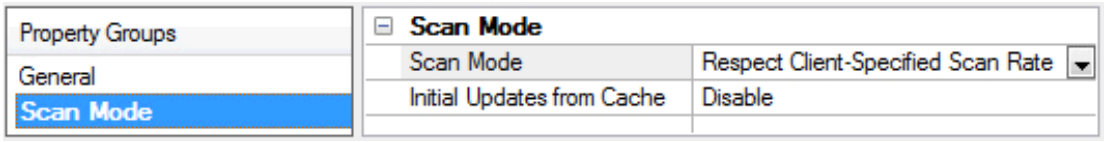
Notes:

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:

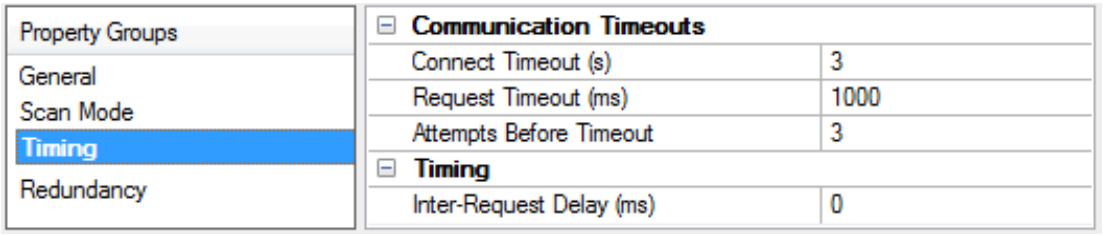
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server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

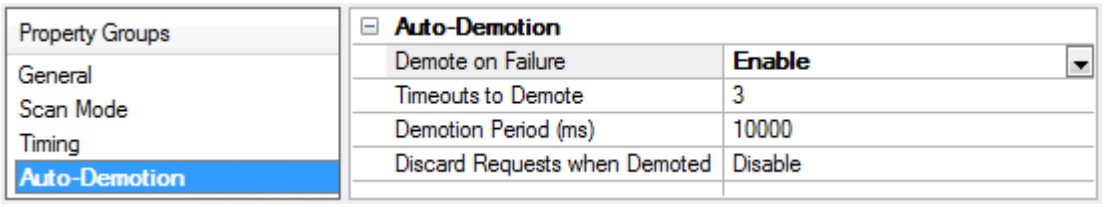
Timing

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 — Tag Generation

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

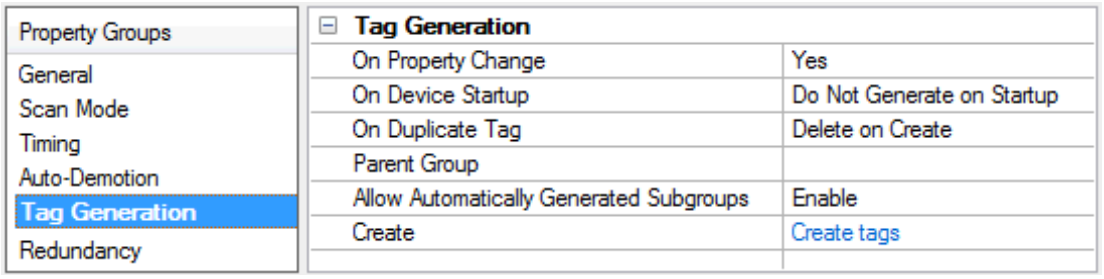
Not all devices and drivers support full automatic tag database generation and not all support the same data types. Consult the data types descriptions or the supported data type lists for each driver for specifics.

If the target device supports its own local tag database, the driver reads the device's tag information and uses the data to generate tags within the server. If the device does not natively support named tags, the driver creates a list of tags based on driver-specific information. An example of these two conditions is as follows:

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.

2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

Note: Automatic tag database generation's mode of operation is completely configurable. *For more information, refer to the property descriptions below.*



On Property Change: If the device supports automatic tag generation when certain properties change, the **On Property Change** option is shown. It is set to **Yes** by default, but it can be set to **No** to control over when tag generation is performed. In this case, the **Create tags** action must be manually invoked to perform tag generation.

On Device Startup: This property specifies when OPC tags are automatically generated. Descriptions of the options are as follows:

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

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

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

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

- **Delete on Create**: This option deletes any tags that were previously added to the tag space before any new tags are added. This is the default setting.
- **Overwrite as Necessary**: This option instructs the server to only remove the tags that the communications driver is replacing with new tags. Any tags that are not being overwritten remain in the server's tag space.

- **Do not Overwrite**: This option prevents the server from removing any tags that were previously generated or already existed in the server. The communications driver can only add tags that are completely new.
- **Do not Overwrite, Log Error**: This option has the same effect as the prior option, and also posts an error message to the server's Event Log when a tag overwrite would have occurred.

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

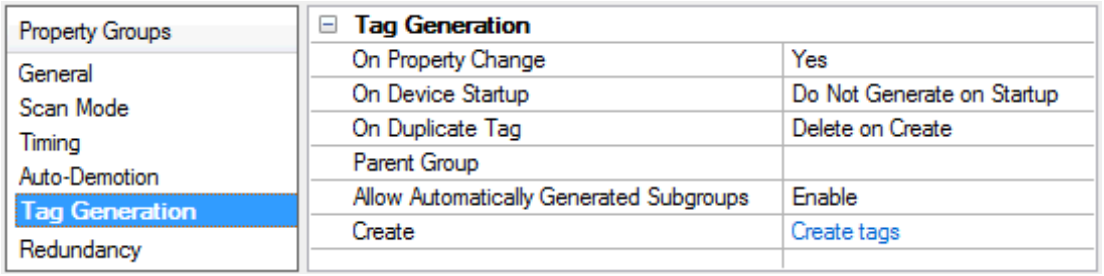
Parent Group: This property keeps automatically generated tags from mixing with tags that have been entered manually by specifying a group to be used for automatically generated tags. The name of the group can be up to 256 characters. This parent group provides a root branch to which all automatically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

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

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

Automatic Tag Database Generation

The Enron Modbus Driver supports the server's Automatic Tag Database Generation feature. When enabled, a list of tags will be built in the server for registers in the device that are mapped to the EFM Configuration Attributes for each enabled meter. To access and configure the Automatic Tag Database Generation settings, right-click on the device and select **Properties**. Then, open the **Tag Generation** property group.

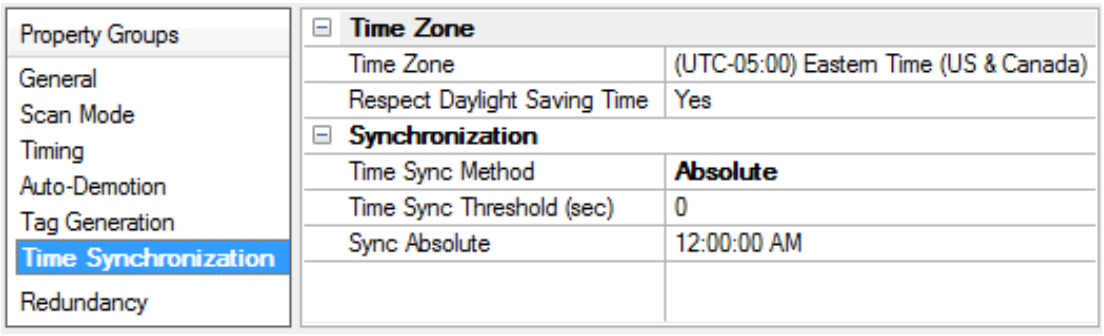


For more information on EFM Configuration Attribute Mapping, refer to Configuration Mapping.

Device Properties — Time Synchronization

This group is used to specify the device's time zone and time synchronization properties. It primarily applies to time stamped data or information from battery-powered devices at remote locations where the device

time may deviate (causing issues with the time-stamped data). To prevent this problem from occurring, users can specify that the server synchronize the device time.



Note: Not all drivers and models support all options.

Time Zone: This property specifies the device's time zone. To ignore the time zone, select one of the first four options in the list (which do not have an offset). The default is the time zone of the local system.
Note: The driver uses this property both when synching the device time and when converting EFM timestamps from the device to UTC time.

Respect Daylight Saving Time: Select Yes to follow Daylight Saving Time offset when synching the device time. Select No to ignore Daylight Saving Time. Only time zones that observe Daylight Saving Time will be affected. The default is No (disabled).
Note: When enabled, the time of the device is adjusted by +1 hour for Daylight Saving Time (in the spring), and adjusted by -1 hour after Daylight Saving Time (in the fall).

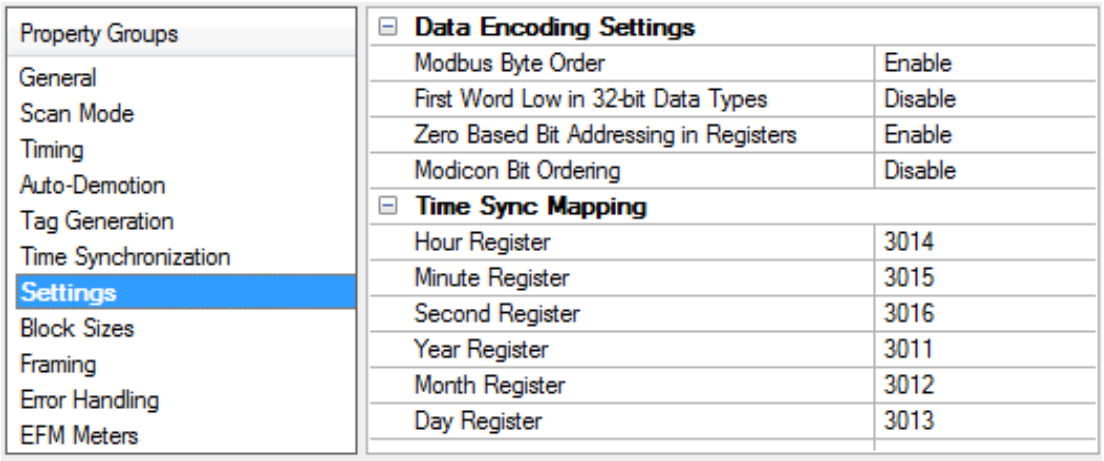
Time Sync Method: This property specifies the method of synchronization. Options include Disabled, Absolute, and Interval. The default is Disabled. Descriptions of the options are as follows:

- **Disabled**: No synchronization.
- **Absolute**: Synchronizes to an absolute time of day specified through the Time property (appears only when Absolute is selected).
- **Interval**: Synchronizes on startup and every number of minutes specified through the Sync Interval property (appears only when Interval is selected). The default is 60 minutes.
- **OnPoll**: Synchronizes when poll is completed (applicable only to EFM devices).

Time Sync Threshold: This property specifies the maximum allowable difference, in seconds, between the device time and the system time before syncing the device time to the system time. If the threshold is set to 0, a time synchronization occurs every time. The default is 0 seconds. The maximum allowable threshold is 600 seconds.

Device Properties — Settings

This group specifies the device's data bit settings.



Modbus Byte Order: When disabled, this option allows users to change the driver's byte order from the default Modbus byte ordering to Intel byte ordering. The default setting is enabled, which is the normal setting for Modbus compatible devices.

Note: If the device uses Intel byte ordering, disabling this option will allow the Enron Modbus Driver to read Intel formatted data properly.

First Word Low in 32-bit Data Types: Users can specify whether the driver should assume the first word is the low or high word of a 32-bit value. First word low follows the convention of the Modicon Modsoft programming software. The default setting is disabled.

Zero-Based Bit Addressing in Registers: When enabled, this option will use zero-based bit addressing within registers and start the first bit at 0. The default setting is enabled.

Use Modicon Bit Ordering (bit 0 is MSB): When enabled, the driver will reverse the bit order on reads and writes to registers to follow the convention of the Modicon Modsoft programming software. For example, when enabled, a write to address 40001.0/1 will affect bit 15/16 in the device. The default setting is disabled. *For more information, refer to the "Modicon Bit Ordering" subtopic below.*

Zero- vs. One-Based Addressing in Registers

Memory types that allow bits within Words can be referenced as a Boolean. The addressing notation for doing this is <*address*>.<*bit*>, where <*bit*> represents the bit number within the Word. Bit level addressing within registers provides two ways of addressing a bit within a given Word; Zero Based and One Based. Zero Based Bit Addressing within registers simply means that the first bit begins at 0. One Based Bit Addressing within registers means that the first bit begins at 1.

Modicon Bit Ordering

For the following example, the 1st through 16th bit signifies either 0 to 15 bits or 1 to 16 bits, depending on whether the driver is set at zero based addressing within registers. In the tables below, MSB is the Most Significant Bit and LSB is the Least Significant Bit.

MSB								LSB							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Use Modicon Bit Ordering Unchecked

MSB								LSB							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Time Synchronization Mapping

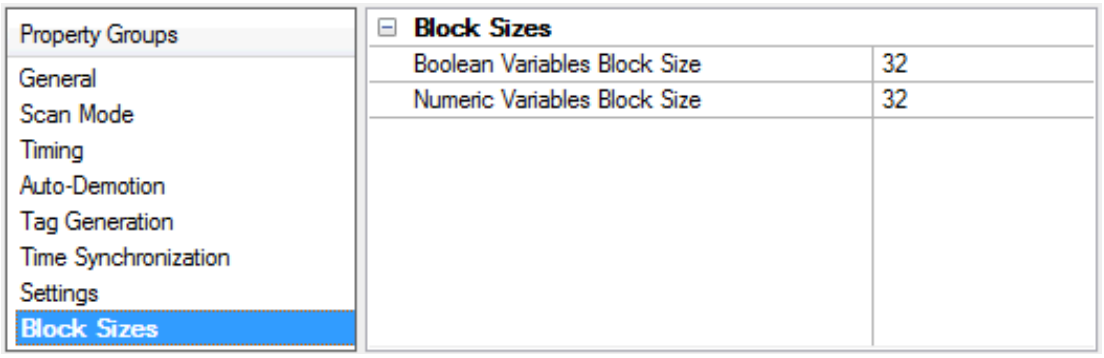
This group is a mapping of the device's real-time clock register addresses. It is only used when synchronizing the device clock. For more information, refer to "Device Properties — Time Synchronization" in the server help file.

Descriptions of the registers are as follows:

- **Hour**: Specify the address of the register containing the RTC hour.
- **Minute**: Specify the address of the register containing the RTC minute.
- **Second**: Specify the address of the register containing the RTC second.
- **Year**: Specify the address of the register containing the RTC year.
- **Month**: Specify the address of the register containing the RTC month.
- **Day**: Specify the address of the register containing the RTC day.

Important: Addresses must fall within their defined ranges. For more information, refer to Address Ranges.

Device Properties — Block Sizes

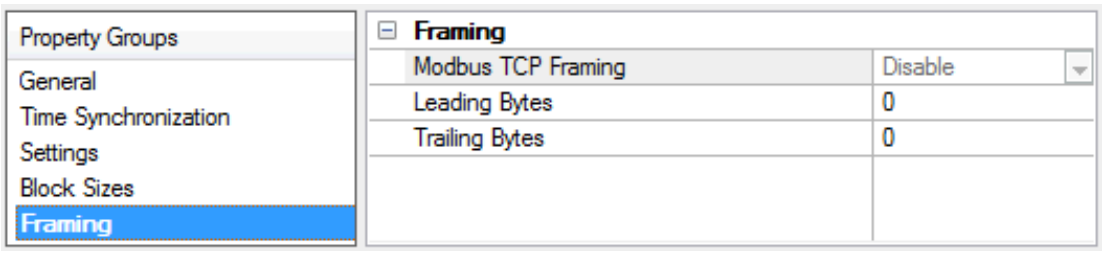


Boolean Variables Block Size: Specify the output and input coils. Coils can be read from 8 to 2000 points (bits) at a time. A higher block size means more points will be read from the device in a single request. The block size can be reduced in order to read data from non-contiguous locations within the device. The default setting is 32.

Numeric Variables Block Size: Specify the internal and holding registers. Registers can be read from 1 to 125 locations (words) at a time. A higher block size means more register values will be read from the device in a single request. The block size can be reduced in order to read data from non-contiguous locations within the device. The default setting is 32.

Device Properties — Framing

Because some terminal server devices add additional data to Modbus frames, this dialog may be used to configure the driver to ignore the additional bytes in response messages.



Modbus TCP Framing: When enabled, this option communicates with native Modbus TCP devices using Ethernet Encapsulation. For more information, refer to "Using Ethernet Encapsulation" below.

Leading bytes: Specify the number of bytes that will be attached to the beginning of Modbus responses. The valid range is 0 to 8. The default setting is 0.

Trailing bytes: Specify the number of bytes that will be attached to the end of Modbus responses. The valid range is 0 to 8. The default setting is 0.

Using Ethernet Encapsulation

Ethernet Encapsulation must be enabled for both Framing and the Use Modbus TCP Framing option to be enabled. For information on enabling Ethernet Encapsulation, refer to the instructions below.

1. To start, right-click on the channel and select **Properties**.
2. Next, open the **Communications** property group. In the **Connection Type** drop-down menu, select **Use Ethernet Encapsulation**. Then, click **Apply** | **Close**.
3. Next, right-click on the device and select **Properties**.
4. Open the **Ethernet Encapsulation** property group. Descriptions of the properties are as follows:
    - **IP Address**: Specify the device's IP address. The default setting is 255.255.255.255.
    - **Port Number**: Specify the port number. Modbus TCP devices typically use 502.
    - **Protocol**: Specify the protocol. Options include UDP and TCP/IP. The default setting is TCP/IP.
5. Configure the Ethernet Encapsulation settings as desired. Once finished, click **OK**.

Device Properties — Error Handling

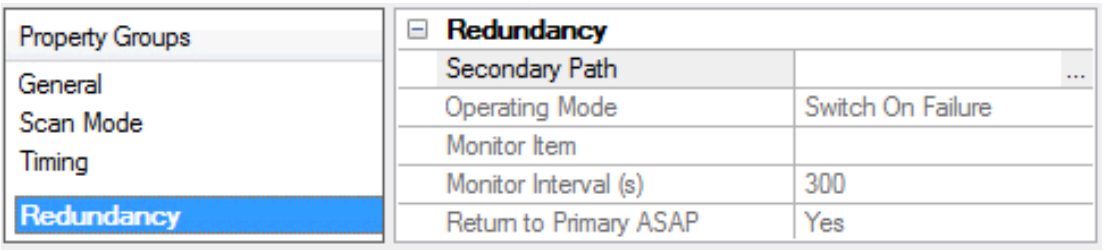
This dialog determines how the Enron Modbus Driver handles errors from the device.

Deactivate Tags On Illegal Address Exception: When enabled, this option causes the driver to stop polling for a block of data if the device returns Modbus exception code 2 (illegal address) or 3 (illegal data, such as number of points) in response to a read of that block. To read addresses that are accessible dynamically in the device, disable this option. The default setting is enabled.

Reject repeated messages: When enabled, the driver rejects repeated messages. When disabled, the driver interprets a repeated message as an invalid response and retries the request. The default setting is disabled.

Note: Some message-relay equipment echoes Modbus requests back to the driver.

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.

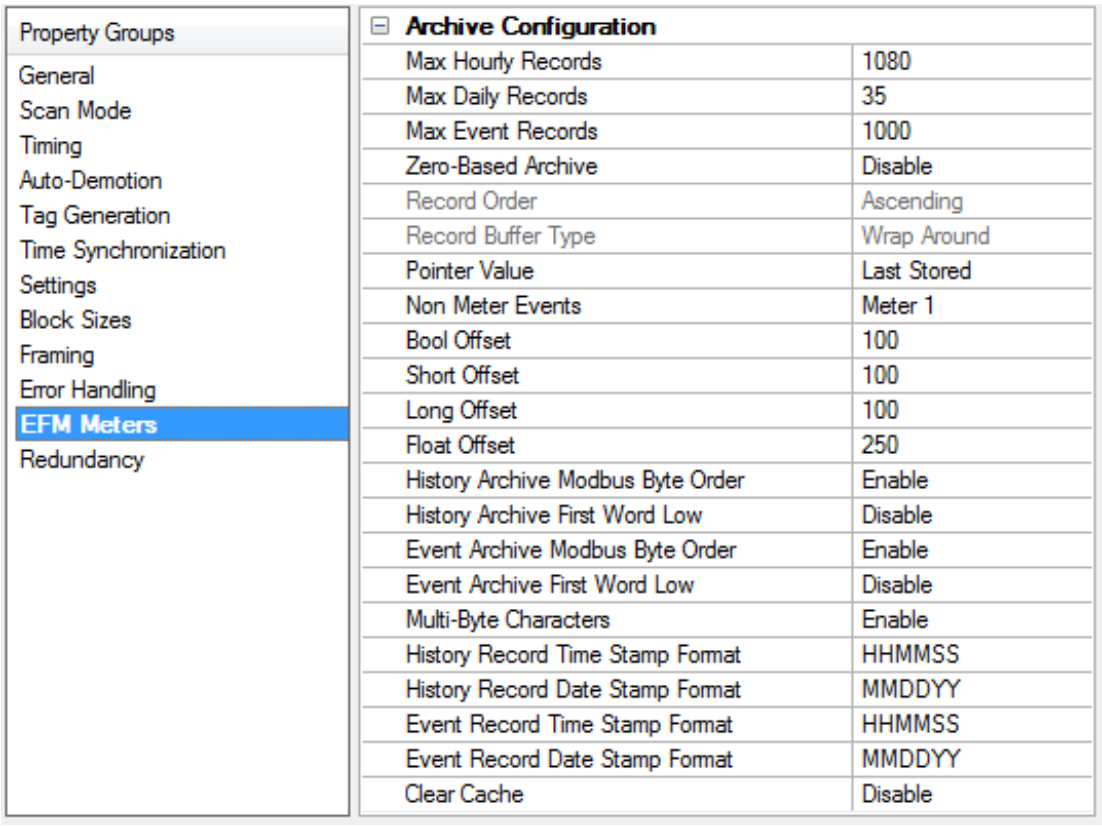
Device Properties — EFM Meters

This dialog contains meter-specific configuration options, and includes a grid control with the following device and per-meter settings. Up to twelve meters are supported.

Correctly configuring the EFM Meters group is essential to exporting all records from the meter's historical archives. If the driver detects a misconfiguration, it stops the poll for the affected archive immediately and logs appropriate messages to the event log. Depending on the type of misconfiguration, the driver may also prevent any further poll requests on the archive until it is reconfigured. Review the event log for errors as well as exported archive data for consistency when performing the first few polls on a newly configured device.

Some Enron devices support different archive sizes on a per-meter basis; this driver does not. If meters have different archive sizes, the driver may not be able to collect the archives for all the meters. Keep this in mind while defining the Maximum Record properties below.

Archive Configuration



Max Hourly Records: Specify the maximum number of hourly records that the device will store. The valid range is 1 to 65535. The default setting is 1080.

Max Daily Records: Specify the maximum number of daily records that the device will store. The valid range is 1 to 65535. The default setting is 35.

Max Event Records: Specify the maximum number of event records that the device will store. The valid range is 1 to 65535. The default setting is 1000.

Record Order: Specify the order that the records will be stored in the device. The default setting is Ascending. This setting is currently not supported.

Zero-Based Archive: Enable if the first entry in the archive is located at index zero (vs. index one, the default). This allows archives from different manufacturers to be read without skipping the first record in the archive.

Tip: This should not be enabled for native Enron Modbus devices. FisherROC devices that support the Enron Modbus protocol use zero-based addressing; enable if communicating with a FisherROC device.

Record Buffer Type: Specify the type of buffer that the device will use to store the records in the device. The default setting is Wrap Around. This setting is currently not supported.

Pointer Value: Specify whether the value in the hourly and daily pointer registers will be the location of the last stored record or the next record that is available. The default setting is Last Stored.

Non Meter Events: Specify the meter number for non-meter events. Utilize the drop-down menu to ignore non-meter events, log non-meter events to a specific meter, or log non-meter events to all meters. The default setting is Meter_1. For more information, refer to **Event Mapping**.

Bool Offset: Specify the offset of the Bool configuration register range for each successive meter. The default setting is 100. *For more information, refer to* ***Configuration Mapping****.*

Short Offset: Specify the offset of the Short configuration register range for each successive meter. The default setting is 100. *For more information, refer to* ***Configuration Mapping****.*

Long Offset: Specify the offset of the Long configuration register range for each successive meter. The default setting is 100. *For more information, refer to* ***Configuration Mapping****.*

Float Offset: Specify the offset of the Float configuration register range for each successive meter. The default setting is 250. *For more information, refer to* ***Configuration Mapping****.*

History Archive Modbus Byte Order: Specify whether the data in History archives will be in standard Modbus byte order. The default setting is enabled.
Note: This property can be overridden per record element. *For more information, refer to* ***History Mapping****.*

History Archive First Word Low: Specify whether the data in History archives will be stored with the first word low. The default setting is disabled.
Note: This property can be overridden per record element. *For more information, refer to* ***History Mapping****.*

Event Archive Modbus Byte Order: Specify whether data in Alarm/Event archives will be in standard Modbus byte order. The default setting is enabled.

Event Archive First Word Low: Specify whether the data in the Alarm/Event archives will be stored with the first word low. The default setting is disabled.

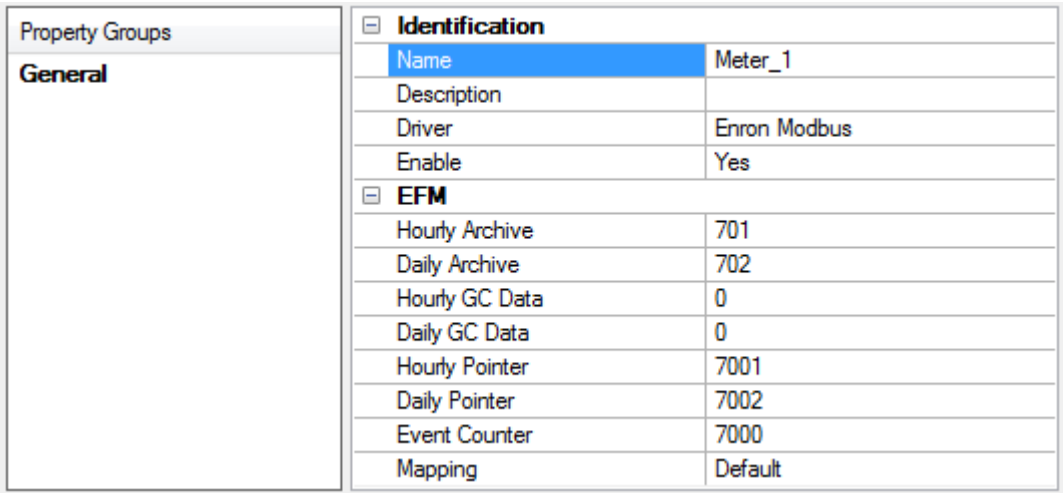
Multi-Byte Characters: Specify whether string configuration data will be stored in two bytes per character format. The default setting is enabled.

History Record Time Stamp Format: Enron Modbus time stamps for history records are floats. Specify whether the time stamps will be in HHMMSS or HHMM.SS (with the seconds after the decimal) format. The default setting is HHMMSS.
Note: The driver assumes that the first and second values in a history record are Date and Time, respectively.

- **History Record Date Stamp Format**: Specify the date stamp format used in the History archive. The default setting is MMDDYY.
- **Event Record Time Stamp Format**: Specify the time stamp format used in the Event archive. The default setting is HHMMSS.
- **Event Record Date Stamp Format**: Specify the date format used in the Event archive. The default setting is MMDDYY.
- **Clear Cache**: Specify whether to clear the device's EFM cache, which is maintained by the server and stores history, alarms, and events data for each meter. When enabled, the cache will be cleared on the next poll. This feature will also remove pointer files, which are used to track EFM uploads in order to prevent uploading the same records twice. All EFM data in the device will be requested again on the next poll. Once the cache is cleared, this property will automatically be set back to disabled. The default setting is disabled.

Note: This option should be used during testing, if the EFM mappings are not configured correctly, or in situations where it is beneficial to re-request all EFM data from the device.

Meter *N*



Meter Name: Specify a descriptive name for the meter. The valid range is 1 to 128 characters. It can neither begin nor end with a trailing blank space. It also cannot begin with an underscore or contain an '@' character, period, or quotation mark.
Note: Multiple meters may not use the same meter name.

Description: This property further describes the meter for identification.

Driver: Specify the driver in use.

Enable: Specify whether the meter is enabled. The default setting for Meter 1 is Yes. The default setting for meters 2 through 12 is No.

Hourly Archive: Specify the meter / run's hourly archive address. The default setting is 701.

Daily Archive: Specify the meter / run's daily archive address. The default setting is 702.

Hourly GC Data: Specify the meter / run's hourly archive address for gas chromatography. The default setting is 0.
Note: When set to 0, GC data will not be requested from the device for hourly archives.

Daily GC Data: Specify the meter / run's daily archive address for gas chromatography. The default setting is 0.
Note: When set to 0, GC data will not be requested from the device for daily archives.

Hourly Pointer: Specify the register that the device will use to indicate the position of the current (or last) hourly record in the buffer. The default setting is 7001.

Daily Pointer: Specify the register that the device will use to indicate the position of the current (or last) daily record in the buffer. The default setting is 7002.

Event Counter: Specify the register that the device will use to indicate the number of alarm / event records in the buffer. The default setting is 7000.

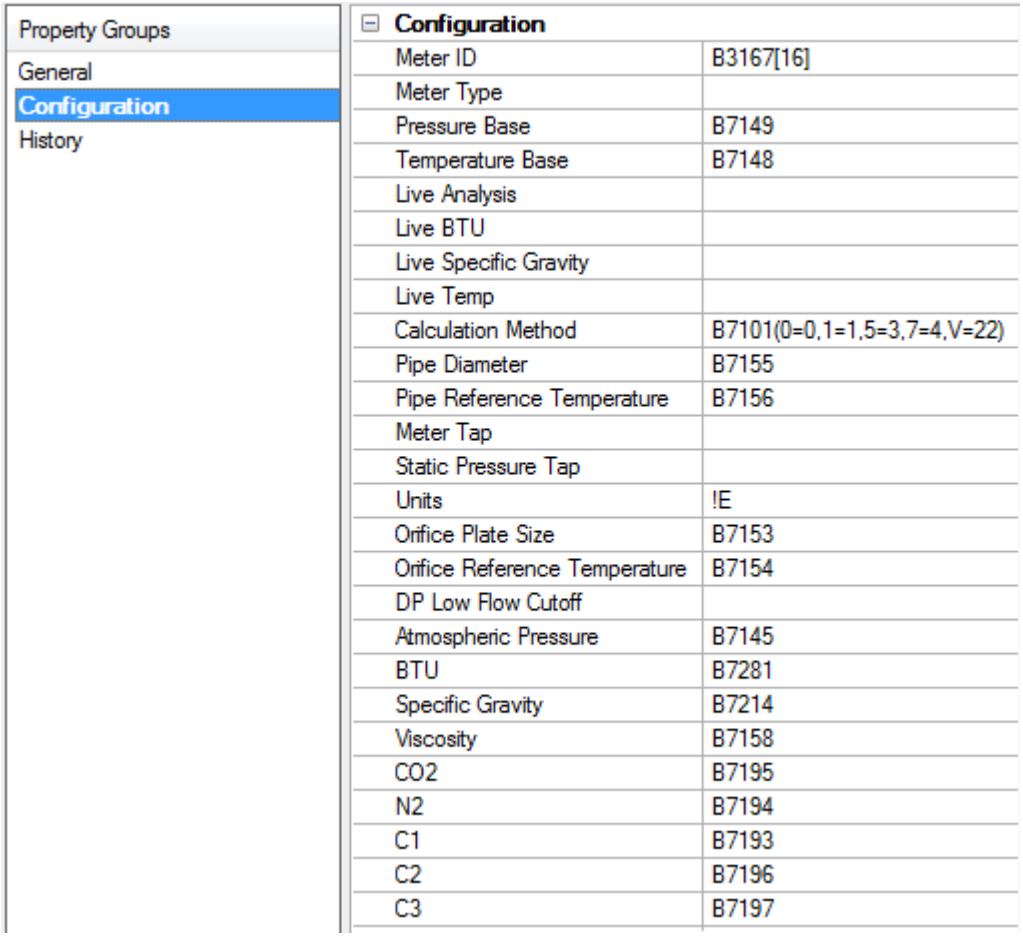
Mapping: Specify the type of mapping that will be used for Configuration, History, and Alarm data. The default setting is Default. *For information on creating a new mapping, refer to Creating a New Mapping.*

Important: The Hourly Pointer, Daily Pointer, and Event Counter addresses must fall within the defined address ranges. *For more information, refer to Address Ranges.*

Configuration Mapping

The addresses that are defined in the Configuration Mapping will be read from the device per meter on each EFM poll. Addresses that are left blank or static will not be read from the device.

Note: Spaces are not allowed in mappings. They are considered unexpected characters and will cause errors.



Configuration	
Meter ID	B3167[16]
Meter Type	
Pressure Base	B7149
Temperature Base	B7148
Live Analysis	
Live BTU	
Live Specific Gravity	
Live Temp	
Calculation Method	B7101(0=0,1=1,5=3,7=4,V=22)
Pipe Diameter	B7155
Pipe Reference Temperature	B7156
Meter Tap	
Static Pressure Tap	
Units	!E
Orifice Plate Size	B7153
Orifice Reference Temperature	B7154
DP Low Flow Cutoff	
Atmospheric Pressure	B7145
BTU	B7281
Specific Gravity	B7214
Viscosity	B7158
CO2	B7195
N2	B7194
C1	B7193
C2	B7196
C3	B7197

Configuration Syntax

Dynamic values that are read from the device will use the following syntax: *B1234[LL](E1=x1,E2=x2,...)* where:

- **B**: The base address. The address is the base address used for Meter 1: an offset will be added for each subsequent meter. The offset depends on the data type, and will be defined by the Bool, Short, Long, and Float offsets specified in EFM Meters. No offset will be used for the attribute if B is omitted from the address.

  Note: The base address syntax allow meters to share a common Configuration Mapping. For

example, with a Bool Offset of 100, "B1000" would be "B1000" for Meter 1, "B1100" for Meter 2, and so forth.

- **LL**: The specifier that is used for data that spans more than one register. LL is the number of subsequent registers to span. This is generally only used for string data, and should only be used for Meter ID.
- **(E1=x1,E2=x2,...)**: The enumeration mapping. The first value (E1, E2, and so forth) is the server's enumeration, and the second value (x1, x2, and so forth) is the equivalent numeric value in the device.

  Note: For example, 4000(O=1,T=2) for Meter Type. If the value at address 4000 is 1, the Meter Type is Orifice. If the value is 2, the Meter Type is Turbine. 'O' and 'T' are defined by the server. For more information on the enumerated types, refer to "Configuration Attributes and Mappings" below.

  Important: All configuration addresses must fall within the defined address ranges. Dynamic value addresses that do not fall within the defined ranges will be skipped when the driver uploads configuration data from the device. For more information, refer to **Address Ranges**.

Static values use the following syntax: *!<static>* where:

- **!**: This character indicates that the subsequent entry is static for the associated attribute.
- **static**: Static can be a string, float, int, or enumeration character depending on the configuration attribute's data type.

Note: Static values are not read from the device. If the Configuration Mapping contains all static values, no device communications or polls will be performed when collecting configuration data.

Configuration Attributes and Mappings

The table below lists all attributes available in the Configuration Mapping, and includes their CSV Name, data type, address syntax, and description.

Attribute	CSV Name	Data Type	Address Syntax	Description
Meter ID	meter_id	String	B1234[10] 1234[10] !Meter1	ID that uniquely identifies the meter.
Meter Type	meter_type	Enumeration	B1234(O=1,- ,P=2,...) 1234(O=1,- ,P=2,...) !O	Enumeration for the meter type. Server enumerations are as follows: O = Orifice P = Positive Displacement T = Turbine U = Ultrasonic L = Liquids V = Vcone C = Coriolis I = Line Pack
Pressure Base	pressure_ base	Float	B1234 1234 !1.2	Pressure base for measured gas volume or gas equivalent volume.**
Temperature Base	temp_base	Float	B1234 1234	Temperature base for measured gas volume or gas equivalent volume.*

Attribute	CSV Name	Data Type	Address Syntax	Description
			!1.2	
Live Analysis	live_analysis	Bool	B1234 1234 !1	I/O connected for live analysis.
Live BTU	live_btu	Bool	B1234 1234 !1	I/O connected for live heat/energy measurement.
Live Specific Gravity	live_specific_gravity	Bool	B1234 1234 !1	I/O connected for live specific gravity measurement.
Live Temp	live_temp	Bool	B1234 1234 !1	I/O connected for live temperature readings.
Calculation Method	calculation_method	Enumeration	B1234(0=2,-,1=3,...) 1234(0=2,-,1=3,...) !V	The method of flow calculation. Options are as follows: 0 = AGA3 1985 1 = AGA3 1992 5= AGA5 7 = AGA7 V = VCone C = AGA11 I = Line Pack
Pipe Diameter	pipe_diameter	Float	B1234 1234 !1.2	Diameter of the pipe. Inches for English units, millimeters for metric.
Pipe Reference Temperature	pipe_ref_temp	Float	B1234 1234 !1.2	Reference temperature of the pipe.*
Meter Tap	meter_taps	Enumeration	B1234(F=1,-,P=2) 1234(F=1,-,P=2) !F	Placement of the meter taps. Options are as follows: F = Flange P = Pipe
Static Pressure Tap	static_pressure_taps	Enumeration	B1234(U=1,-,D=2) 1234(U=1,-,D=2) !U	Placement of the static pressure taps. Options are as follows: U = Upstream D = Downstream
Units	units	Enumeration	B1234(E=1,-,M=2) 1234(E=1,-,M=2) !E	Default units for most of the properties. More granular options are available for Volume Units and K Factor Units. E = English

Attribute	CSV Name	Data Type	Address Syntax	Description
				M = Metric **Note**: This design assumes a device does not mix English and Metric units.
Orifice Plate Size	orifice_plate_size	Float	B1234 1234 !1.2	Size of the Orifice plate. This is only used for Orifice meters. Inches for English Units and Millimeters for Metric.
Orifice Reference Temperature	orifice_ref_temp	Float	B1234 1234 !1.2	Reference Temperature for the Orifice plate. Used for Orifice meters only.*
DP Low Flow Cutoff	dp_low_flow_cutoff	Float	B1234 1234 !1.2	Low flow alarm cutoff.**
Atmospheric Pressure	atmospheric_pressure	Float	B1234 1234 !1.2	Atmospheric Pressure is in PSI for English and Kilopascal for Metric.
BTU	btu	Float	B1234 1234 !1.2	Heating value. BTU/scfc foot for English and MJcubic meter for Metric.
Specific Gravity	specific_gravity	Float	B1234 1234 !1.2	Specific Gravity of real gas.
Viscosity	viscosity	Float	B1234 1234 !1.2	lb/(feet * sec) for English and Centipoises for Metric
CO2	co2	Float	B1234 1234 !90	%
N2	n2	Float	B1234 1234 !90	%
C1	c1	Float	B1234 1234 !90	%
C2	c2	Float	B1234 1234 !90	%
C3	c3	Float	B1234 1234 !90	%
ISO-C4	iso_c4	Float	B1234 1234	%

Attribute	CSV Name	Data Type	Address Syntax	Description
			!90	
NC4	nc4	Float	B1234 1234 !90	%
ISO-C5	iso_c5	Float	B1234 1234 !90	%
NC5	c5	Float	B1234 1234 !90	%
NEO C5	neo_c5	Float	B1234 1234 !90	%
C6	c6	Float	B1234 1234 !90	%
C7	c7	Float	B1234 1234 !90	%
C8	c8	Float	B1234 1234 !90	%
C9	c9	Float	B1234 1234 !90	%
C10	c10	Float	B1234 1234 !90	%
O2	o2	Float	B1234 1234 !90	%
H2O	h2o	Float	B1234 1234 !90	%
H2S	h2s	Float	B1234 1234 !90	%
He	he	Float	B1234 1234 !90	%
H2	h2	Float	B1234 1234 !90	%
CO	co	Float	B1234	%

Attribute	CSV Name	Data Type	Address Syntax	Description
			1234 !90	
Ar	ar	Float	B1234 1234 !90	%
Hourly Record Span	hourly_ record_span	Enumeration	B1234(L=1,-,T=2) 1234(L=1,-,T=2) !L	The meaning of the contract hour. Leading, or Data Preceding, means a contract hour of 8:00 is from 8:00 to 8:59. Trailing, or Data Following, means a contract hour of 8:00 is from 7:01 to 8:00. L = Leading T = Trailing **Note**: Enron Modbus leading / trailing mappings are user-entered based on device configuration for informational purposes only. They do not control driver or exporter behavior.
Contract Hour	contract_hour	Int	B1234 1234 !0	The start of a new billing day. Valid options are 0 to 23.
Contract Day	contract_day	Int	B1234 1234 !0	The start of a new billing month. Valid options are 0 to 31. **Note**: For most fields, this is usually 0.
BTU Base	btu_base	Enumeration	B1234(W=1,-,D=2,...) 1234(W=1,-,D=2,...) !W	Options are as follows: W = Wet D = Dry A = As Delivered
Factor FA	factor_fa	Bool	B1234 1234 !1	Orifice expansion factor due to temperature.
Factor FB	factor_fb	Bool	B1234 1234 !1	Basic Orifice Factor. True if used.
Factor FG	factor_fg	Bool	B1234 1234 !1	Specific Gravity Factor SQRT(1/G). True if used.
Factor FPV	factor_fpv	Bool	B1234 1234 !1	Super compressibility factors.
Factor FR	factor_fr	Bool	B1234	Reynolds factor.

Attribute	CSV Name	Data Type	Address Syntax	Description
			1234 !1	
Factor FT	factor_ft	Bool	B1234 1234 !1	Temperature factor.
Factor FWV	factor_fwv	Bool	B1234 1234 !1	Water Vapor Correction factor.
Factor FY	factor_fy	Bool	B1234 1234 !1	Expansion factor.
Fixed Factor	fixed_factor	Float	B1234 1234 !1.2	Calibration multiplier.
Pipe Material	pipe_material	Enumeration	B1234(S=1,-,C=2,...) 1234(S=1,-,C=2,...) !S	Possible values are as follows: S = Stainless Steel C = Carbon Steel M = Monel
Plate Material	plate_material	Enumeration	B1234(S=1,-,C=2,...) 1234(S=1,-,C=2,...) !S	Possible values are as follows: S = Stainless Steel C = Carbon Steel M = Monel
FPV Method	fpv_method	Enumeration	B1234(N=1,-,A=2,...) 1234(N=1,-,A=2,...) !N	Super compressibility method. Possible values are as follows: N = NX19 A = AGA8 Detail 1 = AGA8 Gross 1 2 = AGA8 Gross 2
Static Pressure Type	static_pressure_type	Enumeration	B1234(G=1,-,A=2) 1234(G=1,-,A=2) !G	Possible values are as follows: G = Gauge A = Absolute
DP Calibration Range High	dp_calib_range_high	Float	B1234 1234 !1.2	High calibration range for dynamic pressure.**
DP Transducer Range High	dp_transd_range_high	Float	B1234 1234 !1.2	High transducer range for dynamic pressure.**
SP Calibration High	sp_calib_high	Float	B1234 1234 !1.2	High calibration range for static pressure.**

Attribute	CSV Name	Data Type	Address Syntax	Description
SP Calibration Low	sp_calib_low	Float	B1234 1234 !1.2	Low calibration range for static pressure.**
SP Transducer Range High	sp_transd_range_high	Float	B1234 1234 !1.2	High transducer range for static pressure.**
Temp Calibration Range High	temp_calib_range_high	Float	B1234 1234 !1.2	High calibration range for temperature.*
Temp Calibration Range Low	temp_calib_range_low	Float	B1234 1234 !1.2	Low calibration range for temperature.*
Temp Transducer Range High	temp_transd_range_high	Float	B1234 1234 !1.2	High transducer range for temperature.*
Temp Transducer Range Low	temp_transd_range_low	Float	B1234 1234 !1.2	Low transducer range for temperature.*
DP Low Alarm	dp_low_alarm	Float	B1234 1234 !1.2	Dynamic pressure low alarm.**
DP Backflow Alarm	dp_backflow_alarm	Float	B1234 1234 !1.2	Dynamic pressure backflow alarm.**
DP High Alarm	dp_high_alarm	Float	B1234 1234 !1.2	Dynamic pressure high alarm.**
SP Low Alarm	sp_low_alarm	Float	B1234 1234 !1.2	Static pressure low alarm.**
SP High Alarm	sp_high_alarm	Float	B1234 1234 !1.2	Static pressure high alarm.**
Temp Low Alarm	temp_low_alarm	Float	B1234 1234 !1.2	Temperature low alarm.*
Temp High Alarm	temp_high_alarm	Float	B1234 1234 !1.2	Temperature high alarm.*
K Factor	k_factor	Float		Units are controlled by K Factor Units Setting.
Meter Factor	meter_factor	Float	B1234 1234 !1.2	The actual volume of gas passing through the meter over the meter indicated volume.
Accumulated	accumulated_	Float	B1234	Units are controlled by Volume Units setting.

Attribute	CSV Name	Data Type	Address Syntax	Description
Volume	volume		1234 !1.2	
Volume Units	unit_volume	Enumeration	B1234(0=1,-,1=2,...) 1234(0=1,-,1=2,...) !0	0= Million Cubic Meters (CM) 1 = Thousand CM 2= Hundred CM 3 = CM 4 = Million Cubic Feet (CF) 5 = Thousand CF 6 = Hundred CF 7 = CF
K Factor Units	unit_k_factor	Enumeration	B1234(0=1,-,1=2,...) 1234(0=1,-,1=2,...) !0	0 = Million Cubic Meters (CM) 1 = Thousand CM 2= Hundred CM 3 = CM 4 = Million Cubic Feet (CF) 5 = Thousand CF 6 = Hundred CF 7 = CF
Ratio of Specific Heats	specific_heats	Float	B1234 1234 !1.2	The heat capacity ratio, adiabatic index, or ratio of specific heats.

* Units of Fahrenheit for English and Celsius for Metric.
* * Units of Inches of water for English and Kilopascal for Metric.

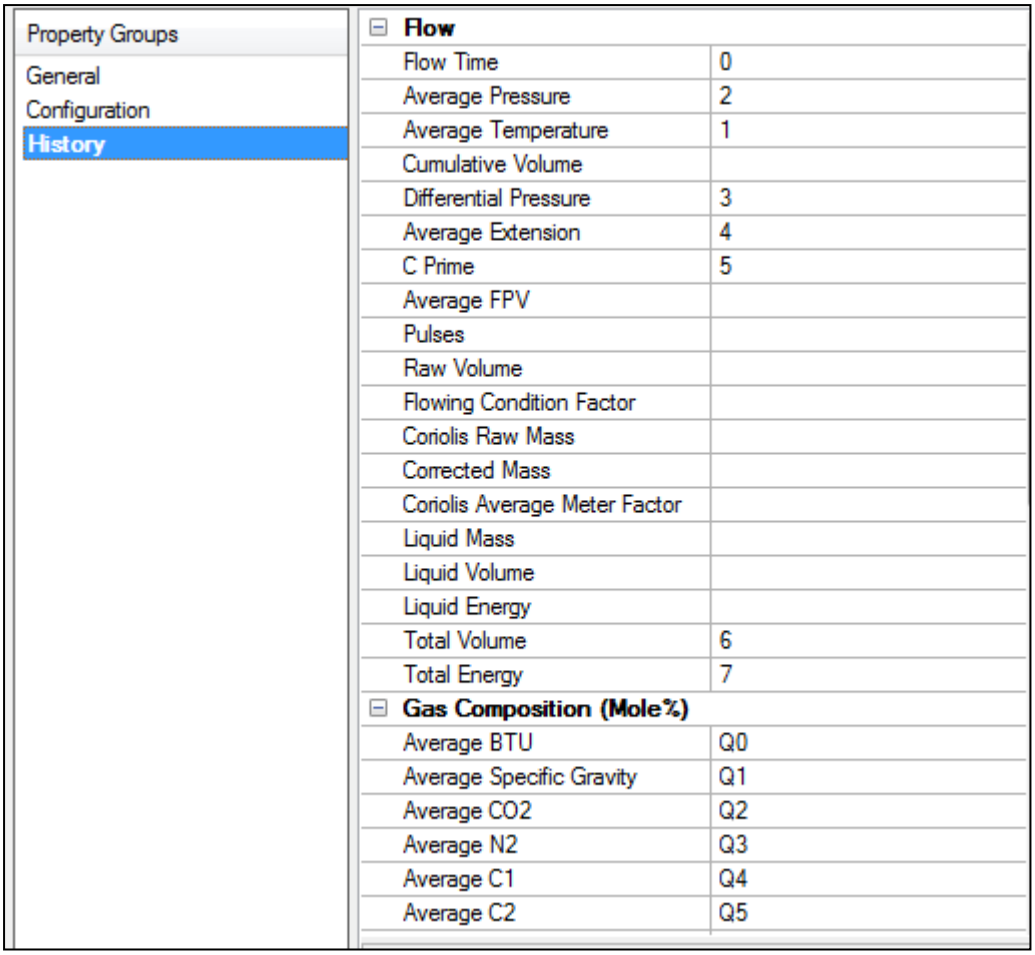
History Mapping

History data pulled from the device is in record form, with each record containing an array of four byte floats. Each float has a unique index or position in the array. The valid range is 0 to *N*, where *N* is the maximum number of floats in the record.

The History dialog is used to assign each float to an EFM attribute using the float's unique index. Records that are retrieved from the device are parsed using this mapping, which applies to both Hourly and Daily History data and can include data from the Gas Chromatograph (GC) archives. When data is not included from the GC archives, they will not be uploaded.

Notes:

1. The mapping assumes that the first float in the record is Date, and that the second float in the record is Time. As a result, Index 0 actually refers to the third value in the record. The Time format is specified in the "History Record Time Stamp Format" property. *For more information, refer to* ***EFM Meters***.
2. Spaces are not allowed in mappings. They are considered unexpected characters and cause errors.



History Syntax

A History Index uses the following syntax: *QN,BO=0/1,WO=0/1* where:

- **Q**: This optional index indicates that the element comes from the Gas Chromatograph archive.
- **N**: This index in the record associates with an attribute. The valid range is 0 to 60.
- **BO=0/1**: This optional syntax allows the value's Byte Order to differ from the "History Archive Modbus Byte Order" setting located in EFM Meters. "BO=0" means Modbus Byte Order with Big Endian (or the most significant bit) first. "BO=1" means Little Endian (or the least significant byte) first.
- **WO=0/1**: This optional syntax allows the value's Word Order to differ from the "History Archive First Word Low" setting located in EFM Meters. "WO=0" means the first word is low. "WO=1" means the first word is high.

Static values use the following syntax: *!<static>* where:

- **!**: This character indicates that the subsequent entry is static for the associated attribute.
- **static**: Static values are always considered floats.

See Also: EFM Meters

History Attributes and Mappings

The table below lists all the attributes available in the History Mapping, and includes their CSV name, data type, and description. Attributes that are left blank will be ignored.

Tips:

- The index syntax (*QN,BO=0/1,WO=0/1*) is available for all attributes.
- The scale factor syntax (*N/<scale factor>* or *S<N>/<scale factor>*) is available for the Flow Time attribute only. Scale factors are always considered floats. Specify word/byte ordering with syntax *N/<Scale factor>,BO=<x>,WO=<y>*.

Flow

Attribute	CSV Name	Data Type	Description
Flow Time	flow_time	Float	Flow time for this record in minutes
Average Pressure	avg_pressure	Float	Average pressure*
Average Temperature	avg_temp	Float	Average temperature over the flow time Fahrenheit for English and Celsius for Metric
Cumulative Volume	cumulative_volume	Float	Volume added during this interval for orifice and turbine meters Units are controlled by the Volume Units in the Configuration Mapping
Differential Pressure	diff_pressure	Float	Average differential pressure for orifice meters*
Average Extension	avg_extension	Float	Average extension for orifice meters*
C Prime	c_prime	Float	Orifice flow constant
Average FPV	avg_fpv	Float	Average Super Compressibility Factor
Pulses	pulses	Float	Pulses for turbine meters
Raw Volume	raw_volume	Float	Raw volume for turbine meters Units are controlled by the Volume Units in the Configuration Mapping
Flowing Condition Factor	flowing_condition_factor	Float	Flowing Condition Factor for turbine meters
Coriolis Raw Mass	coriolis_raw_mass	Float	Raw mass for coriolis meters Units are pounds for English and KG for Raw Mass
Corrected Mass	corrected_mass	Float	Corrected mass for coriolis meters Units are pounds for English and KG for Metric
Coriolis Average Meter Factor	coriolis_avg_meter_factor	Float	Average meter factor for coriolis meters
Liquid Mass	liquid_mass	Float	Mass for liquid meters Units are pounds for English and KG for Metric
Liquid Volume	liquid_volume	Float	Volume for liquid meters Units are controlled by the Volume Units in the Configuration Mapping
Liquid Energy	liquid_energy	Float	Energy for liquid meters

Attribute	CSV Name	Data Type	Description
			Units are BTU/cubic foot for English and MJcubic meter for Metric
Total Volume	total_volume	Float	Total volume Units are controlled by the Volume Units in the Configuration Mapping
Total Energy	total_energy	Float	Total energy Units are BTU/cubic foot for English and MJcubic meter for Metric

* Inches of Water for English and Kilopascals for Metric.

Gas Composition (Mole%)

Attribute	CSV Name	Data Type	Description
Average BTU	avg_btu	Float	Average heating value Units are Dekatherms for English and Gigajoules for Metric
Average Specific Gravity	avg_specific_gravity	Float	Average specific gravity
Average CO2	avg_co2	Float	%
Average N2	avg_n2	Float	%
Average C1	avg_c1	Float	%
Average C2	avg_c2	Float	%
Average C3	avg_c3	Float	%
Average ISOC4	avg_isoc4	Float	%
Average NC4	avg_nc4	Float	%
Average ISOC5	avg_isoc5	Float	%
Average NC5	avg_nc5	Float	%
Average NEOC5	avg_neoc5	Float	%
Average C6	avg_c6	Float	%
Average C7	avg_c7	Float	%
Average C8	avg_c8	Float	%
Average C9	avg_c9	Float	%
Average C10	avg_c10	Float	%
Average O2	avg_o2	Float	%
Average H2O	avg_h2o	Float	%
Average H2S	avg_h2s	Float	%
Average HE	avg_he	Float	%

Attribute	CSV Name	Data Type	Description
Average H2	avg_h2	Float	%
Average CO	avg_co	Float	%
Average AR	avg_ar	Float	%
Specific Heat Ratio	specific_heat_ratio	Float	Ratio of specific heat
Viscosity	viscosity	Float	Viscosity Units of Pounds/Mass per Foot/Second for English and Centipoises for Metric

Alarm Mapping

The Alarms dialog is used to assign alarms received from the device to specific meters, alarm types, and states. Alarms can apply to one or more meters depending on how the alarm address is specified.

Notes:

1. When an alarm is received from a device that does not match an address in any of the Alarm Mappings being used, the server will check if the alarm matches an address in the Event Mappings. If it does, it will be handled by the Event Mapping. If it does not, the alarm will be logged as a user string event.
2. Spaces are not allowed in mappings. They are considered unexpected characters and will cause errors.

See Also: EFM Mapping

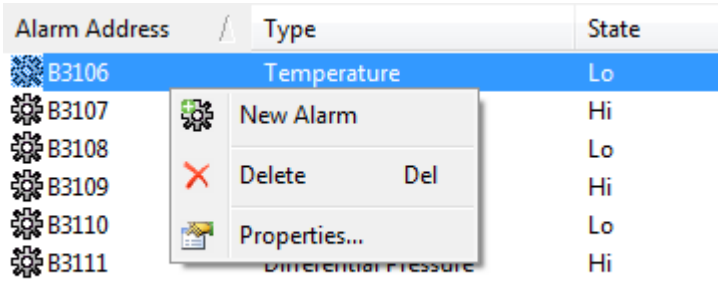
Alarm	
Address	
Type	Differential Pressure
State	Off

Descriptions of the properties are as follows:

- **Address**: Specify the Enron address that generates the alarm. The default setting is blank.
- **Type**: Specify the type of alarm. The default setting is Differential Pressure. Options are as follows:
  - Differential Pressure
  - Static Pressure
  - Temperature
  - Cutoff
  - Backflow
  - Battery
- **State**: Specify the alarm state. The default setting is Off. Options are as follows:

- Off
- On
- Hi
- Lo

To access the following properties, right-click an existing alarm and click the appropriate selection.



Descriptions of the properties are as follows:

- **Delete**: When clicked, this removes the selected alarm from the mapping.
- **Properties...** : When clicked, this launches the Alarm dialog that contains the selected alarm's properties.

Alarm Syntax

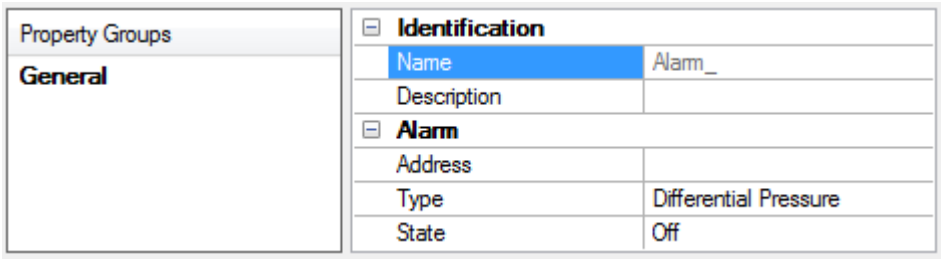
An alarm address may use one of the following syntactic forms:

- **B1234**: This is a base address, and makes the alarm meter-specific. The offset depends on the address data type and the Bool, Short, Long, and Float Offsets specified in EFM Meters.
  Note: For example, an address is "B1234" and the data type is Bool. If the Bool Offset is 10, an alarm received from device address "1234" will be assigned to Meter 1. An alarm received from device address "1244" will be assigned to Meter 2.
- **1234**: This is a static address, and makes the alarm non-meter specific. An alarm that is received from address 1234 will be sent to all meters that use the Alarm Mapping.

Important: All alarm addresses must fall within the defined address ranges. Dynamic value addresses that do not fall within the defined ranges will be skipped when the driver uploads alarm data from the device. For more information, refer to Address Ranges.

Adding a New Alarm

1. To start, right-click **Default** under **EFM(Mappings)**, and select **New Alarm**.
2. Specify the new alarm's Name, Description, Address, Type, and State.



3. Once finished, click **OK**.

Event Mapping

Event Mappings are not user-configurable: they depend on the Configuration Mapping.

When an event is received from an address that matches an address in any Configuration Mapping, it will be converted to an audit event. For example, if a Configuration Mapping with address "B7500" is set to BTU and Meter 1 is using the mapping, an event will be generated from address "7500" when the user changes the BTU in the device. The event will be converted to an audit event for the BTU field on Meter 1. Both the old and new values will be displayed.

When an event is received from an address that does not match an address in any Configuration Mapping, it will be considered a non-meter event. The event will be converted to a string event, and then handled as defined in the "Non Meter Events" setting located in the EFM Meters property group.

EFM Cache

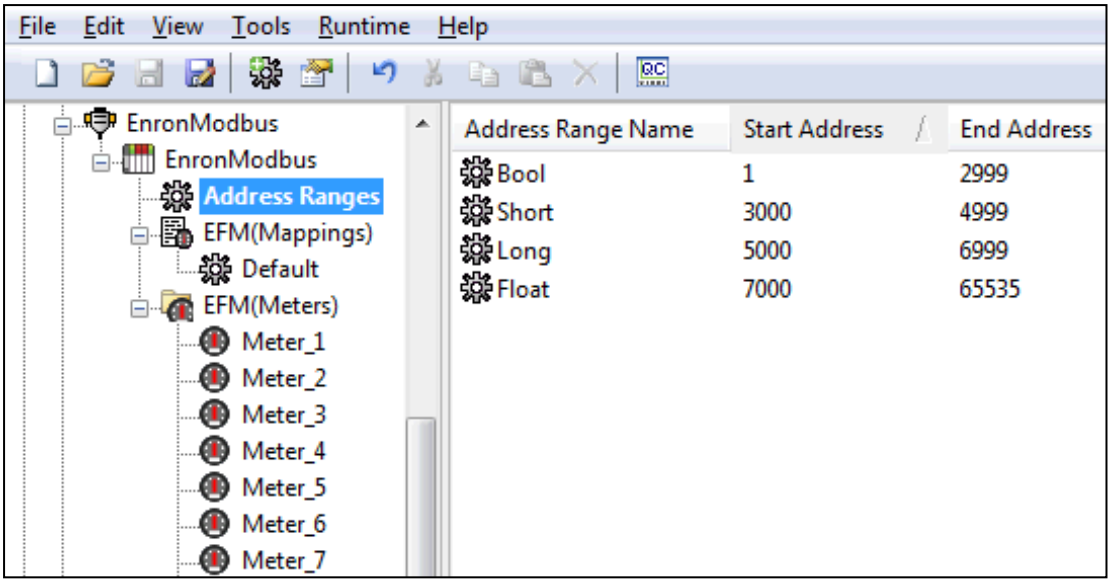
The Enron Modbus Driver caches EFM data per device. During polls, the driver will only request new data from the device and then add it to its local cache. This minimizes communication between the physical device and the driver. The cache that is maintained by the driver will be cleared under the following scenarios:

1. The server is reinitialized, restarted, or a new project is loaded.
2. The channel or device is deleted.
3. The cache is cleared manually through the "Clear Cache" setting located in EFM Meters.
4. A meter's Hourly or Daily GC or History archives change.

See Also: EFM Meters

Address Ranges

This group specifies the address ranges that are available in the device.



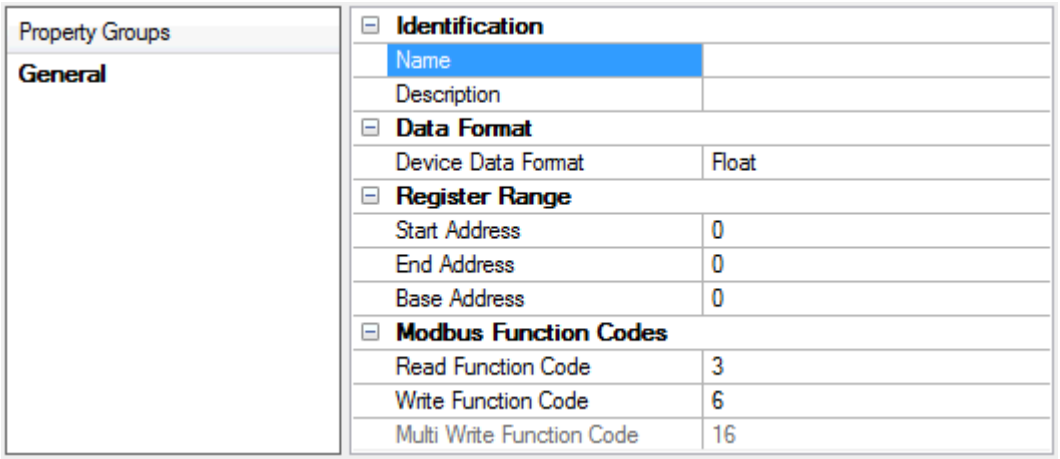
Right-click **Address Ranges** to access the following properties:

- **New Address Range**: This property launches the Address Range group, which is used to define a new address range. For more information, refer to "Address Range" below.
- **Import CSV...**: This property launches the Import from CSV dialog, which is used to import address ranges defined in a CSV file.
- **Export: CSV...** This property launches the Export to CSV dialog, which is used to export the address ranges to a CSV file (where they can be edited and imported).

Notes: Imported address ranges can neither overlap nor have the same descriptions as existing address ranges.For more information, refer to CSV Import/Export.

Address Range

This group is used to create a new address range or modify an existing address range.



Descriptions of the properties are as follows:

- **Name**: This property identifies the address range.
- **Description**: Specify a descriptive name for the address range.

- **Device Data Format**: Specify the data format of the address range in the device. Valid data types include Boolean, Short, Long, Float, Long(2x16), and Float(2x16). The default is Float.
  For more information, refer to Data Types Description.
  Note: Float and Long data types read/write to one register; Long(2x16) and Float(2x16) read/write to two registers.
- **Start Address**: Specify the starting address for the address range. The default is 0.
- **End Address**: Specify the ending address for the address range. The default is 0.
- **Base Address**: Specify an offset that may be applied to the address range to map to registers in the device. This should be used if the device expects the address in read and write requests to be a data address rather than a coil/register number. For example, if the start address is 40001, the end address is 49999, and the device expects a read request for the first register to be address 0, then the base address would be 40001. The default setting is 0.
- **Read Function Code**: Specify the read function code for the address range. The valid values are 1 and 2 for Boolean ranges, and 3 and 4 for all other device data formats. The default is 3.
- **Write Function Code**: Specify the write function code for the address range. If the device does not support the single write function code, this property may be set to the same value as Multi Write. The valid values are 5 and 15 for Boolean ranges, and 6 and 16 for all other device data formats. The default setting is 6.
- **Multi Write Function Code**: Specify the multi-write function code for the address range. The valid values are 15 for Boolean ranges, and 16 for all other device data formats. The default is 16.

CSV Import/Export

The EFM Mappings support the import and export of data in a Comma Separated Variable (CSV) file. CSV import and export supports the efficient configuration of many devices. For more information on a specific aspect of CSV Import/Export, select a link from the list below.

Creating a Template

The easiest way to create an import CSV file is to create a template. For more information, refer to the instructions below.

1. To start, create a new device using the default settings. Then, click **OK**.
2. Next, right-click on **Default**, under **EFM Mapping**.
3. Select **Export CSV**.
4. Save the file to an accessible location.
5. Use the exported template in a spreadsheet application that supports CSV files, and then modify the file as desired.

Note: Microsoft Excel is an excellent tool for editing large groups of tags outside the server. Once a template CSV file has been exported, it can be loaded directly into Excel for editing.

Exporting EFM Mappings

Exporting an EFM Mapping will generate a CSV text file that contains sections for Configuration, History, and Alarms. Each section has a heading record followed by a record for each item. Column names must match those listed; however, columns may be in any order.

Configuration Mapping
The required columns are listed in **bold**.

Column Name	Description
Attribute	This is the name of the Configuration Mapping attribute. Attributes can be in any order. Attributes that are not included in an import will be left blank in the mapping. **Note**: All possible attribute names are listed in the Configuration dialog. For more information, refer to Configuration Mapping.
Value	This is the address syntax for the attribute. It can be blank, static, or take the form *B1234[LL](E1=1,E2=2)*. **Note**: For information on each attribute's value limitations, refer to Configuration Mapping.

History Mapping
The required columns are listed in **bold**.

Column Name	Description
Attribute	This is the name of the History Mapping attribute. Attributes can be in any order. Attributes that are not included on an import will be left blank in the mapping. **Note**: All possible attribute names are listed in the History dialog. For more information, refer to History Mapping.
Value	This is the address syntax for the attribute. It can be blank, static, or take the form *QN,BOO=0/1,WO=0/1*. **Note**: For information on each attribute's value limitations, refer to History Mapping.

Alarm Mapping
The required columns are listed in **bold**.

Column Name	Description
Address	This is the address of the alarm. It can take the form *B1234*. For more information, refer to Alarm Mapping.
Alarm Type	This is the type of the alarm. The default setting is Differential Pressure. The valid types are as follows: 1 = Differential Pressure 2 = Static Pressure 3 = Temperature

Column Name	Description
	4 = Cutoff 5 = Backflow 6 = Battery
Alarm State	This is the state of the alarm. The default setting is Off. 1 = Off 2 = On 3 = Hi 4 = Lo

Importing EFM Mappings

Once the CSV file has been created and exported, it may be re-imported into an EFM Mapping. To do so, open **EFM Mapping** and then click **Import CSV**.

Note: For Configuration, History, and Alarms, importing will replace all existing settings with the settings specified in the CSV file. When the import is complete, the configured mapping should match one for one with the file.

Using Other Characters as the Delimiter

When utilizing a CSV file that does not use a comma or semi-colon delimiter, users should do one of the following:

- Save the project in XML. Then, perform mass configuration on the XML file instead of using CSV.
- Perform a search-and-replace on the delimiter in the CSV file and then replace the delimiter with a comma or semicolon. The delimiter being used by the OPC server (either comma or semicolon) must be set to the replacement character.

Note: For information on specifying which character to use as the variable (comma or semicolon), refer to "Options - General" in the server help file.

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

Data Type	Description
	bit 0 is the low bit bit 31 is the high bit
Long / Long (2x16)	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit *See Note 2*
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Float / Float (2x16)	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 *See Note 2*
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.

Notes:

1. The descriptions assume the default first word low data handling of 32-bit data types.
2. Float and Long data types can read/write to one register or two registers based on how they are specified by **Address Range** when setting up the device.

Address Descriptions

The Enron Modbus Driver supports the default address ranges listed in the table below. The default data types are shown in **bold**.

See Also: **Address Ranges**

Address	Default Range	Data Type	Access
Boolean Variables	0001-2999	**Boolean**	Read/Write
16-bit Short Variables	3001-4999 3xxx.0/1-4xxx.15/16*	**Short**, Word, BCD Boolean	Read/Write
32-bit Long Variables	5001-6999 5xxx.0/1-6xxx.31/32*	**Long**, DWord, LBCD, Float Boolean	Read/Write
32-bit Float Variables	7000-65535	**Float**, Long, DWord, LBCD	Read/Write

Address	Default Range	Data Type	Access
	7xxx.0/1-6xxxx.31/32*	Boolean	

Notes:

1. 32-bit Float/Long variables can span either one register (1x32) or two registers (2x16), depending on how the address range is defined.
2. * For more information, refer to "Zero vs. One Based Addressing Within Registers" in **Data Encoding Settings**.

Array Support

Arrays are supported for all data types. There are two methods of addressing an array. The following examples use holding register locations:

7*xxx* [rows] [cols]
7*xxx* [cols] with assumed row count of one.

For arrays, rows multiplied by cols cannot exceed the block size that has been assigned to the device for the register/coil type. For register arrays of 32-bit data types, rows multiplied by cols multiplied by 2 cannot exceed the block size.

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>.
Data Type <type> is not valid for device address <address>.
Device address <address> contains a syntax error.

Device Status Messages

Device <device name> is not responding.
Unable to write to <address> on device <device name>.

Enron Modbus Specific Messages

<Device Name> - Failed to read EFM pointer file. <Extended Error>.
<Device Name> - Failed to write EFM pointer file. <Extended Error>.
Alarm mapping for address <address> is invalid and will be ignored.
Archive address <address> is used in Meter <number> for <archive> archive and in Meter <number> for <archive> archive in device <device name>. Duplicate archive addresses are not allowed.
Bad address in block [<start address> to <end address>] on device <device name>.
Bad array spanning [<address> to <address>] on device <device>.
Block address [<start address> to <end address>] on device <device> responded with exception code <code>.
Config attribute <attribute index> is unknown and will be ignored.
Config mapping for attribute <attribute name> is invalid and will be ignored.
Error parsing alarm/event record. The record size is incorrect.
Error parsing history record. History mapping does not match record read from device, record will not be logged.
Error reading date and time, alarm/event record will not be logged.
Error reading date and time, history record will not be logged.
Failure to load <mapping name> mapping from CSV. The header contains a duplicate field name <name>.
Failure to load <mapping name> mapping from CSV. The header contains an unrecognized field name <name>.
Failure to load <mapping name> mapping from CSV. There is no header in the CSV file.
History attribute <attribute index> is unknown and will be ignored.
History mapping for attribute <attribute name> is invalid and will be ignored.
Meter <number> has an invalid EFM Mapping (<mapping name>). Defaulting the mapping to <mapping name>.
Meter name <name> is used in Meter <number> and in Meter <number> in device <device name>. Duplicate meter names are not allowed.
Serialization of EFM data to temporary file <file name> failed. Reason: <file I/O error>.

The configuration map address <address> for meter <meter name> is beyond the maximum address allowed by the Enron Modbus protocol. This address will be ignored.
The EFM Meter Daily GC data value <value> in device <device name>' is not valid. Valid range is 0 or <min> to <max>.
The EFM Meter Event Counter value <value> in device <device name> is not valid. Valid range is 0 or <min> to <max>.
The EFM Meter Hourly GC data value <value> in device <device name> is not valid. Valid range is 0 or <min> to <max>.
Unable to create tag for EFM configuration attribute <attribute> with address <address> on meter <meter name>.
Unable to read <address> from device <device name>. The device is configured for broadcast writes only.
Unable to read <address> from device <device name>. The device is not responding.
Unable to read block address [<start address> to <end address>] on device <device name>. Unexpected characters in response.
Unable to read from address <address> on device <device name>: Device responded with exception code <code>.
Unable to read from address <address> on device <device name>. Response is not the correct size.
Unable to read from address <address> on device <device name>. Unexpected characters in response.
Unable to synchronize time with device <device name>. The device is not responding.
Unable to write to address <address> on device <device>. Device responded with exception code <code>.
Unable to write to address <address> on device <device name>. Unexpected characters in response.
Value for attribute <attribute name> retrieved from Configuration read could not be associated with a valid enumerable value.
Warning loading <mapping name> mapping from CSV. Alarm state for address <address> is invalid. Setting the state to off.
Warning loading <mapping name> mapping from CSV. Alarm type for address <address> is invalid. Setting the type to differential pressure alarm.
Warning loading <mapping name> mapping from CSV. Ignoring alarm with no address.
Warning loading <mapping name> mapping from CSV. Ignoring record with no address.
Warning loading <mapping name> mapping from CSV. No records were imported.
Warning loading <mapping name> mapping from CSV. The attribute <name> is unknown, and will be ignored.

Serial Communications

Communications error on <channel name> [<error mask>].
COMn does not exist.
COMn is in use by another application.
Error opening COMn.
Unable to set comm properties on COMn.

See Also: Modbus Exception Codes

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 a smaller value for the array or a different starting point.

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

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

Device <device name> is not responding.

Error Type:
Serious

Possible Cause:

1. The serial connection between the device and the Host PC is broken.
2. The communications properties for the serial 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 (ms)" property.

Solution:

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

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

Error Type:
Serious

Possible Cause:

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

Solution:

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

<Device Name> - Failed to read EFM pointer file. <Extended Error>.

Error Type:
Warning

Extended Error:
When supplied by the operating system, this describes the file error that occurred.

Possible Cause:

1. A permission error was encountered when the EFM pointer cache was read.
2. The EFM pointer cache file is corrupt.

Solution:

The Enron Modbus Driver automatically generates a new EFM pointer file; however, the server re-polls (uploading all EFM data) during the next EFM poll for meters in the device.

Note:

For more information, refer to the extended error.

<Device name> - Failed to write EFM pointer file. <Extended error>.

Error Type:

Warning

Extended Error:

When supplied by the operating system, this describes the file error that occurred.

Possible Cause:

1. The disk is full.
2. A permission error was encountered when the EFM pointer cache was written.

Solution:

The server will attempt to update the EFM pointer file periodically, in addition to when the server is shut-down. If the pointer file cannot be written, the server will re-poll (uploading all EFM data) during the next EFM poll for meters in the device.

Note:

For more information, refer to the extended error.

Alarm mapping for address <address> is invalid and will be ignored.

Error Type:

Warning

Possible Cause:

An invalid Alarm Mapping was imported from a CSV file or loaded from an XML project file.

Solution:

Correct the Alarm Mapping in the CSV import file or the XML project file.

See Also:

Alarm Mapping

Archive address <address> is used in Meter <number> for <archive> archive and in Meter < number> for <archive> archive in device <device name>. Duplicate archive addresses are not allowed.

Error Type:
Serious

Possible Cause:
The XML project file contains duplicate archive addresses.

Solution:
Ensure that the XML project file does not contain duplicate archive addresses.

Note:
Daily and hourly GC archive addresses may be shared across meters.

Bad address in block [<start address> to <end address>] on device <device name>.

Error Type:
Serious

Possible Cause:

1. An attempt has been made to reference a nonexistent location in the specified device.
2. An attempt has been made to read more registers than allowed by the protocol.

Solution:

1. Verify the tags assigned to addresses in the specified range on the device and eliminate ones that reference invalid locations.
2. Decrease the register **block size** value to 125.

See Also:
Error Handling
Block Sizes

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

Error Type:
Serious

Possible Cause:

1. An attempt has been made to reference a nonexistent location in the specified device.
2. An attempt has been made to read more registers than allowed by the protocol.

Solution:

1. Verify that all the register addresses requested in the array exist in the device and reduce the array size such that only valid addresses (that exist in the device) are requested by the array.
2. Reduce the array size value to 125.

See Also:
Error Handling
Block Sizes

Block address [<start address> to <end address>] on device <device> responded with exception code <code>.

Error Type:
Warning

Possible Cause:
For a description of the exception codes, refer to Modbus Exception Codes.

Solution:
For a description of the exception codes, refer to Modbus Exception Codes.

Config attribute <attribute index> is unknown and will be ignored.

Error Type:
Warning

Possible Cause:
An invalid Configuration Mapping was imported from a CSV file or loaded from an XML project file.

Solution:
Correct the Configuration Mapping in the CSV import file or the XML project file.

See Also:
Configuration Mapping

Config mapping for attribute <attribute name> is invalid and will be ignored.

Error Type:
Warning

Possible Cause:
An invalid Configuration Mapping was imported from a CSV file or loaded from an XML project file.

Solution:
Correct the Configuration Mapping in the CSV import file or the XML project file.

See Also:
Configuration Mapping

Error parsing alarm/event record. The record size is incorrect.

Error Type:
Warning

Possible Cause:
An EFM Alarm/Event archive record returned by the device is not a complete Enron Modbus Historical or Alarm/Event archive record.

Solution:
Verify that the EFM Archive settings are correct.

See Also:
EFM Meters

Error parsing history record. History mapping does not match record read from device, record will not be logged.

Error Type:
Warning

Possible Cause:
The History Mapping does not match the History Record that was returned from the device.

Solution:
Verify that the History Mapping is correct for the device.

Error reading date and time, alarm/event record will not be logged.

Error Type:
Warning

Possible Cause:
The date and time format in the Alarm/Event record returned by the device could not be read.

Solution:
Verify that the EFM Meter Event's Word and Byte order are correct.

See Also:
EFM Meters

Error reading date and time, history record will not be logged.

Error Type:
Warning

Possible Cause:
The date and time format in the History Record returned by the device could not be read.

Solution:

Verify that the EFM Meter History's Word and Byte order are correct.

See Also:
EFM Meters

Failure to load <mapping name> mapping from CSV. The header contains a duplicate field name <name>.

Error Type:
Fatal

Possible Cause:
The CSV file header contains a duplicate field name.

Solution:
Verify that the CSV file is a valid EFM Mapping CSV import file.

Failure to load <mapping name> mapping from CSV. The header contains an unrecognized field name <name>.

Error Type:
Fatal

Possible Cause:
The CSV file header contains an invalid field name.

Solution:
Verify that the CSV file is a valid EFM Mapping CSV import file.

Failure to load <mapping name> mapping from CSV. There is no header in the CSV file.

Error Type:
Fatal

Possible Cause:
The CSV file does not contain a valid header.

Solution:
Verify that the CSV file is a valid EFM Mapping CSV import file.

History attribute <attribute index> is unknown and will be ignored.

Error Type:
Warning

Possible Cause:
An invalid History Mapping was imported from a CSV file or loaded from an XML project file.

Solution:

Correct the History Mapping in the CSV import file or the XML project file.

See Also:
History Mapping

History mapping for attribute <attribute name> is invalid and will be ignored.

Error Type:
Warning

Possible Cause:
An invalid History Mapping was imported from a CSV file or loaded from an XML project file.

Solution:
Correct the History Mapping in the CSV import file or the XML project file.

See Also:
History Mapping

Meter <number> has an invalid EFM Mapping (<mapping name>). Defaulting the mapping to <mapping name>.

Error Type:
Warning

Possible Cause:
The EFM Mapping specified for a meter/run in the project file is missing or invalid.

Solution:
Verify that the project file contains the specified EFM Mapping.

Meter name <name> is used in Meter <number> and in Meter <number> in device <device name>. Duplicate meter names are not allowed.

Error Type:
Serious

Possible Cause:
The XML project file contains duplicate meter names.

Solution:
Ensure that the XML project file does not contain duplicate meter names.

Serialization of EFM data to temporary file <file name> failed. Reason: <file I/O error>.

Error Type:
Warning

Possible Cause:

1. The driver was unable to create the specified file directory.
2. The driver was unable to access the specified file.

Solution:

1. Verify that the disk has sufficient disk space.
2. Verify user permissions for the specified file directory.

The configuration map address <address> for meter <meter name> is beyond the maximum address allowed by the Enron Modbus protocol. This address will be ignored.

Error Type:
Warning

Possible Cause:
When the offset is applied for the specified meter, the base address of an EFM configuration attribute extends beyond the maximum address that is allowed by the Enron Modbus Protocol.

Solution:
Verify that the specified meter's base address and offset are correct.

See Also:
Configuration Mapping
EFM Meters

The device archive index is larger than the archive size configured in the server. Please reconfigure the device with the correct archive size.

Error Type:
Warning

Possible Cause:

1. The **Max Records** property for the archive does not match the archive size in the device.
2. The device uses a different base for the start/end of the archive.

Solution:

1. Ensure the archive size is correctly configured for the affected archive.
2. Verify that the **Zero-Based Archive** property is correctly configured for the device.

The EFM Meter Daily GC data value <value> in device <device name>' is not valid. Valid range is 0 or <min> to <max>.

Error Type:
Serious

Possible Cause:
The XML project file contains a Daily GC data value that is out of range.

Solution:
Ensure that the Daily GC data value is within the specified range.

The EFM Meter Event Counter value <value> in device <device name> is not valid. Valid range is 0 or <min> to <max>.

Error Type:
Serious

Possible Cause:
The XML project file contains an Event Counter value that is out of range.

Solution:
Ensure that the Event Counter value is within the specified range.

The EFM Meter Hourly GC data value <value> in device <device name> is not valid. Valid range is 0 or <min> to <max>.

Error Type:
Serious

Possible Cause:
The XML project file contains an Hourly GC data value that is out of range.

Solution:
Ensure that the Hourly GC data value is within the specified range.

The requested record does not exist or is invalid for <archive tag> on <device>. Aborting the poll. Please verify the archive configuration settings.

Error Type:
Warning

Possible Cause:

1. The **Max Records** property for the archive does not match the archive size in the device.
2. The device uses a different base for the start/end of the archive.

Solution:

1. Ensure the archive size is correctly configured for the affected archive.
2. Verify that the **Zero-Based Archive** property is correctly configured for the device.

Unable to create tag for EFM configuration attribute <attribute> with address <address> on meter <meter name>.

Error Type:
Warning

Possible Cause:
The calculated address for the attribute is out of range when given the meter number.

Solution:
1. Ensure that the address mapped to the attribute is correct.
2. Ensure that the offset is correct for the data type.
3. Ensure that the address ranges are properly configured for the device.

See Also:
Configuration Mapping
EFM Meters
Address Ranges

Unable to read <address> from device <device name>. The device is configured for broadcast writes only.

Error Type:
Warning

Possible Cause:
The device is configured for broadcast writes only, and an EFM Poll was triggered. The Device ID is set to 0.

Solution:
1. Disable EFM polling for broadcast devices.
2. Do not use a Device ID of 0 for EFM-enabled devices.

Unable to read block address [<start address> to <end address>] on device <device name>. Unexpected characters in response.

Error Type:
Warning

Possible Cause:

The calculated CRC did not match the CRC that was sent by the device.

Solution:

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

Unable to read from address <address> on device <device>: Device responded with exception code <code>.

Error Type:
Warning

Possible Cause:
For a description of the exception code, refer to **Modbus Exception Codes**.

Solution:
For a description of the exception code, refer to **Modbus Exception Codes**.

Unable to read from address <address> on device <device>. The configured device ID did not match the value retrieved from the device <deviceID>.

Error Type:
Warning

Possible Cause:
The model property is not set correctly.

Solution:
Set the model correctly for the device in the configuration in the OPC server. Currently, the options are Standard and Extended Station ID.

Unable to read from address <address> on device <device name>. Response is not the correct size.

Error Type:
Warning

Possible Cause:
An EFM upload request response did not contain a complete Enron Modbus Historical or Alarm/Event archive record.

Solution:
Verify that the EFM Archive settings are correct.

See Also:
EFM Meters

Unable to read from address <address> on device <device name>. Unexpected characters in response.

Error Type:
Warning

Possible Cause:
The calculated CRC did not match the CRC that was sent by the device.

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

Unable to read <address> from device <device name>. The device is not responding.

Error Type:
Warning

Possible Cause:
The device is not responding to a read request.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications 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 setting so that the entire response can be handled.

Unable to synchronize time with device <device name>. The device is not responding.

Error Type:
Warning

Possible Cause:
The device is not responding to a time synchronization write.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications 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 setting so that the entire response can be handled.

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

Error Type:
Warning

Possible Cause:
For a description of the exception code, refer to **Modbus Exception Codes**.

Solution:
For a description of the exception code, refer to **Modbus Exception Codes**.

Unable to write to address <address> on device <device name>. Unexpected characters in response.

Error Type:
Warning

Possible Cause:
The calculated CRC did not match the CRC that was sent by the device.

Solution:

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

Value for attribute <attribute name> retrieved from Configuration read could not be associated with a valid enumerable value.

Error Type:
Warning

Possible Cause:
The device returned an attribute value that could not be mapped to a valid enumeration value.

Solution:
Verify that the project configuration mapping for the attribute matches the data device should return.

Warning loading <mapping name> mapping from CSV. Alarm state for address <address> is invalid. Setting the state to off.

Error Type:
Warning

Possible Cause:
The alarm state in the CSV import file for the specified address is invalid.

Solution:
Verify that the alarm state in the CSV import file is correct.

Warning loading <mapping name> mapping from CSV. Alarm type for address <address> is invalid. Setting the type to differential pressure alarm.

Error Type:
Warning

Possible Cause:
The alarm type in the CSV import file for the specified address is invalid.

Solution:
Verify that the alarm type in the CSV import file is correct.

Warning loading <mapping name> mapping from CSV. Ignoring alarm with no address.

Error Type:
Warning

Possible Cause:
The CSV import file contains an alarm that does not specify an address.

Solution:
Verify that an alarm address is present in the CSV import file and is correct.

Warning loading <mapping name> mapping from CSV. Ignoring record with no address.

Error Type:
Warning

Possible Cause:
The CSV import file contains a configuration or history attribute that does not specify an address.

Solution:
Verify that an attribute address is present in the CSV import file and is correct.

Warning loading <mapping name> mapping from CSV. No records were imported.

Error Type:
Warning

Possible Cause:
The CSV import file did not contain any valid records.

Solution:
Verify that the CSV import file contains valid records.

Warning loading <mapping name> mapping from CSV. The attribute <name> is unknown, and will be ignored.

Error Type:
Warning

Possible Cause:
The specified attribute in the CSV import file is unknown.

Solution:
Verify that the attribute in the CSV import file is correct.

Communications error on <channel name> [<error mask>].

Error Type:
Serious

Error Mask Definitions:
B = Hardware break detected.
F = Framing error.
E = I/O error.
O = Character buffer overrun.
R = RX buffer overrun.
P = Received byte parity error.
T = TX buffer full.

Possible Cause:
1. The serial connection between the device and the Host PC is bad.
2. The communications properties for the serial connection are incorrect.

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

COMn does not exist.

Error Type:
Fatal

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

Solution:
Verify that the proper COM port has been selected.

COMn is in use by another application.

Error Type:

Fatal

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

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

Error opening COMn

Error Type:
Fatal

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

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

Unable to set comm properties on COMn

Error Type:
Fatal

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

Solution:
Verify the serial properties and make any necessary changes.

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 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.

For this driver, the terms Slave and Unsolicited are used interchangeably.

Index

A

B

C

D

E

F

G

H

I

M

N

O

P

R

S

T

U

V

W