WITS Level 0 Passive Driver

Table of Contents

WITS Level 0 Passive Driver ... **1**
Table of Contents ... **2**
WITS Level 0 Passive Driver ... 4
Overview ... 4
Setup ... **5**
Channel Properties — General ... 5
Channel Properties — Serial Communications ... 6
Channel Properties — Write Optimizations ... 9
Channel Properties — Advanced ... 10
Channel Properties — Timeout ... 10
Device Properties — General ... 11
Device Properties — Scan Mode ... 12
Device Properties — Tag Generation ... 13
Device Properties — Settings ... 15
Data Types Description ... **16**
Address Descriptions ... **17**
Error Descriptions ... **18**
Device '<device name>' item '<WITS ID>', Write value length exceeds maximum frame length. Truncating to 1024 bytes ... 18
Device '<device name>', Abandoning current message after <time in ms> ms of inactivity ... 18
Device '<device name>', Communications have timed out after <time in seconds> seconds of inactivity ... 19
Device '<device name>', Invalid delimiter. Expecting '<configured delimiter>'. Discarding remainder of message ... 19
Device '<device name>', Message parsing error. Discarding remainder of message ... 19
Device '<device name>', More than 4096 bytes in item '<WITS ID>' data. Discarding remainder of message ... 20
Device '<device name>', Non-numeric WITS ID found: '<WITS ID>'. Discarding remainder of message ... 20
Device '<device name>', Non-printable character '<hex byte>' found in WITS ID. Discarding remainder of message ... 20
Device '<device name>', Non-printable character '<hex byte>' in item '<WITS ID>' data. Discarding remainder of message ... 21
Device '<device name>', Received Null value '<Null value>' for item '<WITS ID>'. Setting tag to bad quality ... 21
Device '<device name>', Value '<value>' is invalid for item '<WITS ID>' with data type '<data ... 21

type>'. Setting tag to bad quality .....
Device '<device name>', Value '<value>' out of range for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality 21
Unable to write to address '<address>' on device '<device name>'. Invalid socket 22
Resources **23**
Index **24**

WITS Level 0 Passive Driver

Help version 1.008

CONTENTS

Overview
What is the WITS Level 0 Passive Driver?

Channel Setup
How do I specify timeout properties for the channel?

Device Setup
How do I configure devices for use with this driver?

Data Types Description
What data types does this driver support?

Address Descriptions
How do I address a data location on a WITS Level 0 Passive Driver device?

Error Descriptions
What error messages are produced by the WITS Level 0 Passive Driver?

Overview

The WITS Level 0 Passive Driver provides a reliable way to connect WITS Level 0 devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with devices that support Level 0 of the WITS protocol and transmit unsolicited data.

Setup

Supported Devices

All devices that transmit unsolicited WITS Level 0 data.

Communication Protocol

WITS Level 0.

Supported Communication Properties

Baud Rate: All major Baud rates.
Parity: Odd, Even, and None.
Data Bits: 8.
Stop Bits: 1 and 2.

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

Maximum Number of Channels and Devices

The maximum number of channels supported by this driver is 256. The maximum number of devices supported is 256 (1 device per channel).

Unsolicited Ethernet Encapsulation

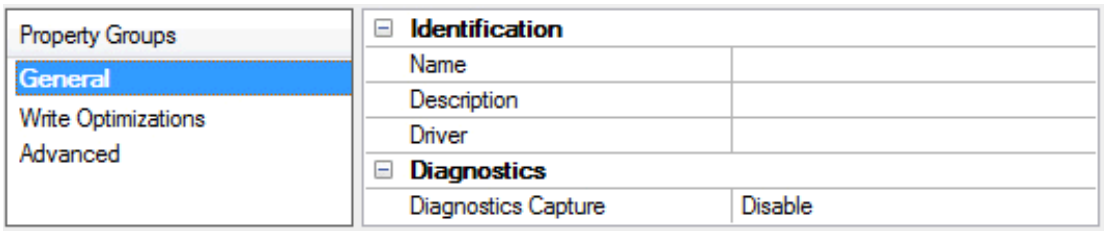
This driver supports Unsolicited Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server. It may be enabled for the channel in the Communications property group of Channel Properties. For more information, refer to the OPC server's help file.

Automatic Tag Database Generation

This driver supports the Automatic Tag Database Generation of all pre-defined WITS records. For more information on pre-defined WITS records, refer to **W.I.T.S. Wellsite Information Transfer Specification**.

Channel Properties — General

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



Identification

Name: User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information.

For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

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

Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties.
Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

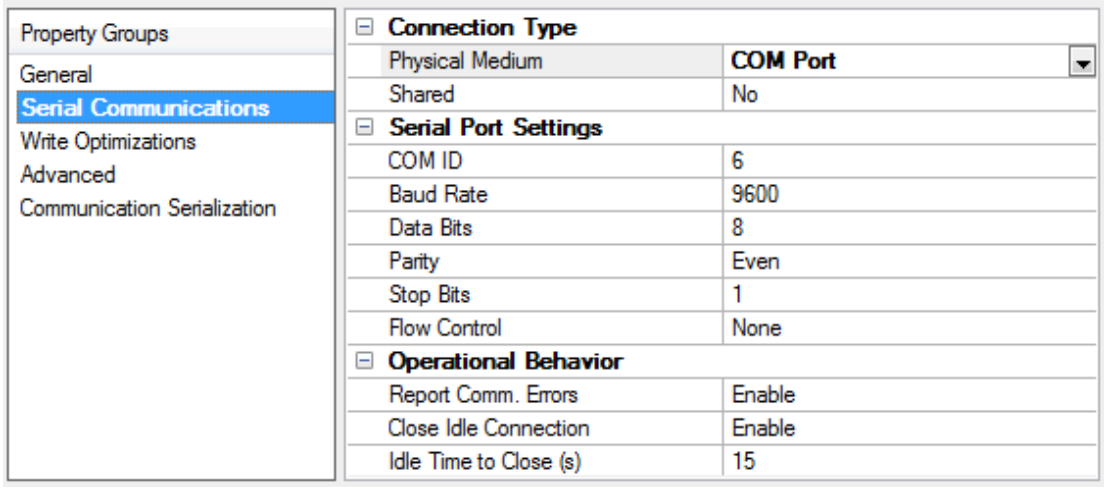
Diagnostics

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

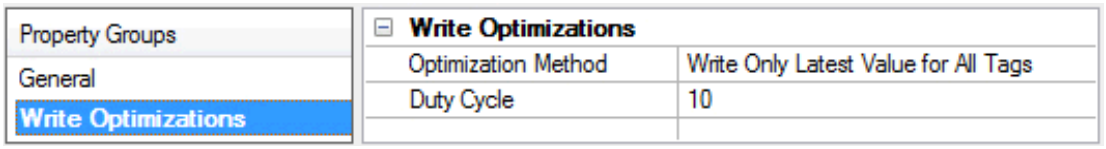
- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific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

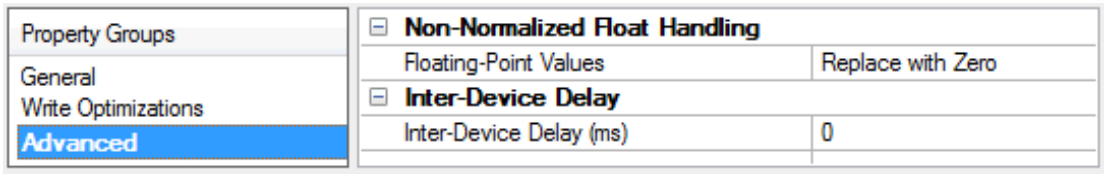
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing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

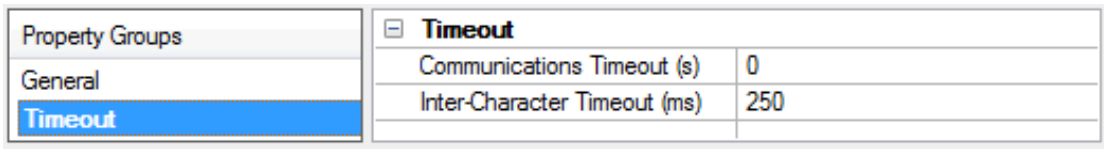
Note: This property is disabled if the driver does not support floating point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM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

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

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

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

Channel Properties — Timeout

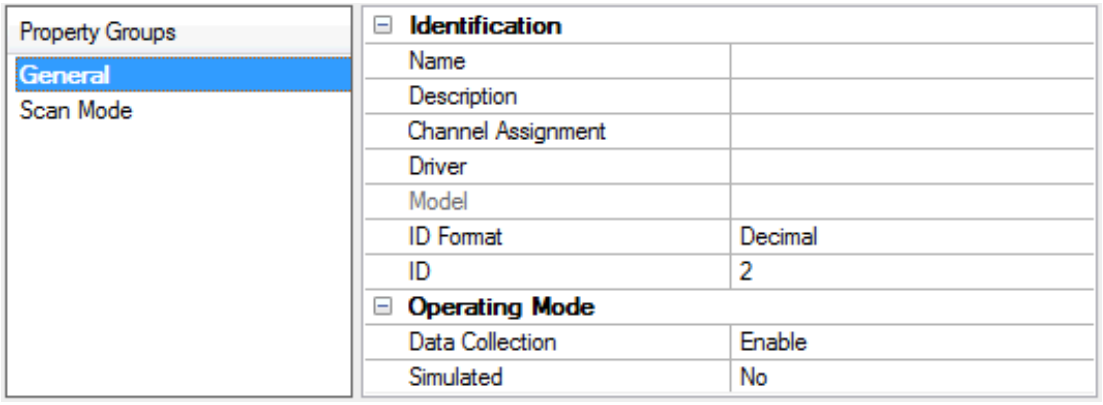


Communications Timeout: This property specifies the time in seconds that the driver will wait between receiving unsolicited messages. When the timeout expires, the device's error state will be set and all tags owned by the associated device will be set to Bad quality. A value of 0 will disable the timeout. The valid range is 0 to 3600 seconds. The default setting is 0 seconds.
Note: When left at the default disabled setting, users will not receive error messages if communications with the device stop.

Inter-Character Timeout: This property specifies the time in milliseconds that the driver will wait between receiving bytes within a message. When the timeout expires, the device's error state will be set, the current message will be abandoned, and the driver will interpret the next received byte as the beginning of a new message. The valid range is 50 to 1000 milliseconds. The default setting is 250 milliseconds.

Device Properties — General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.



Identification

Name: This property specifies the name of the device. It is a logical user-defined name that can be up to 256 characters long, and may be used on multiple channels.

Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".

For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

Description: User-defined information about this device.

Many of these properties, including Description, have an associated system tag.

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

Driver: Selected protocol driver for this device. This property specifies the driver selected during channel creation. It is disabled in the channel properties.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.

Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. For more information, refer to the driver help documentation.

ID: This property specifies the device's station / node / identity / address. The type of ID entered depends on the communications driver being used. For many drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The ID format can be Decimal, Octal, and Hexadecimal. If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver.

Operating Mode

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

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

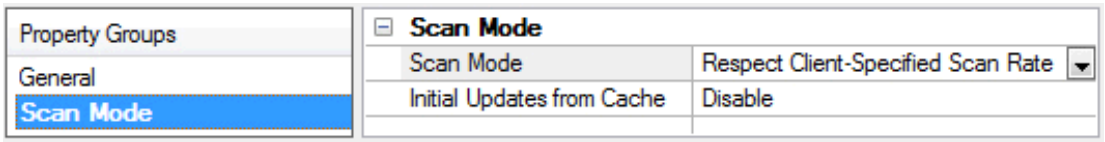
Notes:

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

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

Device Properties — Scan Mode

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



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

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

Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.

- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

Device Properties — Tag Generation

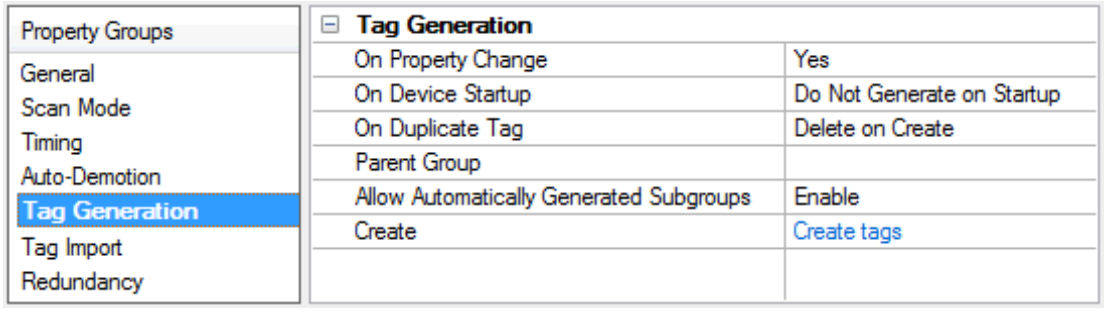
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the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

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

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

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.
2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

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



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

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

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

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

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

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

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

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

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

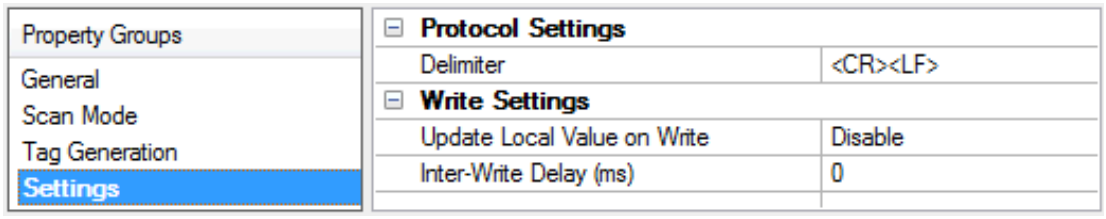
Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.

Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

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

Note: **Create tags** is disabled if the Configuration edits a project offline.

Device Properties — Settings



Delimiter: This property specifies the sequence of characters that the driver will expect between the lines of a WITS Level 0 message. Options include <CR><LF>, <LF><CR>, <CR>, and <LF>. The default setting is <CR><LF>. Descriptions of the options are as follows:

- **<CR><LF>:** This option is a carriage return followed by a line feed.
- **<LF><CR>:** This option is a line feed followed by a carriage return.
- **<CR>:** This option is a carriage return.
- **<LF>:** This option is a line feed.

Update local value on write: When enabled, this option will update the local value immediately when a tag is written. This option should be disabled when the WITS device is expected to echo the written item back to the driver. The default setting is disabled.
If the device is in an error state, the tag's value will not be updated until the error state is cleared.

Inter-write Delay: This property specifies the minimum amount of time that the driver will wait in between sending requests to the target device. The default setting is 0 milliseconds.

Data Types Description

Data Type	Description
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Float	32-bit floating point value bit 0 is the low bit bit 31 is the high bit
String	Null terminated ASCII string Strings up to a length of 4096 characters are supported.

Address Descriptions

The default data type is displayed in **bold**.

Address	Range	Data Type	Access
xxxx	0001-9999	Short, DWord, Long, Float, **String**	Read/Write

Note 1: The WITS Level 0 Passive Driver supports the full range of addresses between 0001 and 9999. If the address is pre-defined in the WITS specification, the default data type will be equal to the pre-defined item's data type. If the address is not pre-defined, the default data type will be String.

Note 2: For more information on pre-defined WITS records, refer to **W.I.T.S. Wellsite Information Transfer Specification**.

Error Descriptions

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

Device '<device name>' item '<WITS ID>', Write value length exceeds maximum frame length. Truncating to 1024 bytes
Device '<device name>', Abandoning current message after <time in ms> ms of inactivity
Device '<device name>', Communications have timed out after <time in seconds> seconds of inactivity
Device '<device name>', Invalid delimiter. Expecting '<configured delimiter>'. Discarding remainder of message
Device '<device name>', Message parsing error. Discarding remainder of message
Device '<device name>', More than 4096 bytes in item '<WITS ID>' data. Discarding remainder of message
Device '<device name>', Non-numeric WITS ID found: '<WITS ID>'. Discarding remainder of message
Device '<device name>', Non-printable character '<hex byte>' found in WITS ID. Discarding remainder of message
Device '<device name>', Non-printable character '<hex byte>' in item '<WITS ID>' data. Discarding remainder of message
Device '<device name>', Received Null value '<Null value>' for item '<WITS ID>'. Setting tag to bad quality
Device '<device name>', Value '<value>' is invalid for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality
Device '<device name>', Value '<value>' out of range for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality
Unable to write to address '<address>' on device '<device name>'. Invalid socket

Device '<device name>' item '<WITS ID>', Write value length exceeds maximum frame length. Truncating to 1024 bytes

Error Type:

Warning

Possible Cause:

The written value is greater than 1024 characters.

Solution:

If truncation is not acceptable, write values that are less than or equal to 1024 characters.

Device '<device name>', Abandoning current message after <time in ms> ms of inactivity

Error Type:

Warning

Possible Cause:

The connection with the device was lost partway through message reception.

Solution:

1. Ensure that the device is sending valid WITS Level 0 messages.
2. Ensure that the connection with the device is reliable.
3. Adjust the value of the Inter-Character Timeout property.

Device '<device name>', Communications have timed out after <time in seconds> seconds of inactivity

Error Type:
Warning

Possible Cause:
The connection with the device has not been established or has been lost.

Solution:

1. Verify the connection with the device.
2. Adjust the value of the Communication Timeout property.

Device '<device name>', Invalid delimiter. Expecting '<configured delimiter>'. Discarding remainder of message

Error Type:
Warning

Possible Cause:
The delimiter sent by the device is not equal to the delimiter that was configured.

Solution:
Ensure that the delimiter sent by the device is equal to the delimiter that was configured.

Device '<device name>', Message parsing error. Discarding remainder of message

Error Type:
Warning

Possible Cause:
An incorrect character was received when a header (&&) or trailer (!!) was expected.

Solution
Verify that the data being sent follows the WITS Level 0 protocol.

Device '<device name>', More than 4096 bytes in item '<WITS ID>' data. Discarding remainder of message

Error Type:
Warning

Possible Cause:
A line of data is greater than 4096 bytes long.

Solution:
Reduce the data sent by WITS Level 0 devices to 4096 bytes or fewer.

Device '<device name>', Non-numeric WITS ID found: '<WITS ID>'. Discarding remainder of message

Error Type:
Warning

Possible Cause:

1. A non-numeric character was received as part of a WITS ID. This may be due to noise on the transmission medium or an invalid WITS ID being sent by the device.
2. The configured delimiter is part of, but not equal to, the delimiter sent by the device.

Solution:

1. Ensure that the WITS device only transmits numeric WITS IDs.
2. Ensure that the delimiter is correctly configured.

Device '<device name>', Non-printable character '<hex byte>' found in WITS ID. Discarding remainder of message

Error Type:
Warning

Possible Cause:

1. A non-printable character was received in a WITS ID. This may be due to noise on the transmission medium or an invalid WITS ID being sent by the device.
2. The configured delimiter is part of, but not equal to, the delimiter sent by the device.

Solution:

1. Ensure the WITS device transmits only printable ASCII characters.
2. Ensure that the delimiter is correctly configured.

Device '<device name>', Non-printable character '<hex byte>' in item '<WITS ID>' data. Discarding remainder of message

Error Type:
Warning

Possible Cause:
A non-printable character was received in the value of an item. This may be due to noise on the transmission medium or an invalid value being sent by the device.

Solution:
Ensure that the WITS device only transmits printable ASCII characters.

Device '<device name>', Received Null value '<Null value>' for item '<WITS ID>'. Setting tag to bad quality

Error Type:
Warning

Possible Cause:
Either -8888 or -9999 was received as the value of an item.

Solution:
Ensure that the Null value transmission was expected.

Device '<device name>', Value '<value>' is invalid for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality

Error Type:
Warning

Possible Cause:
A tag with a numeric data type received a value from a device with a non-numeric character.

Solution:
Ensure that the data sent by the device is valid, and that the appropriate data type is configured.

Device '<device name>', Value '<value>' out of range for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality

Error Type:
Warning

Possible Cause:
The received data is outside the range of values allowed for the configured numeric data type.

Solution:
Ensure that the data sent by the device is valid, and that the appropriate data type is configured.

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

Error Type:
Error

Possible Cause:
When using unsolicited Ethernet Encapsulation, the WITS device's address will be unknown until the first message is received from the device. This error message occurs when a write is made to one of the device's items before its address is known.

Solution:
When the driver is configured for unsolicited Ethernet Encapsulation, users should ensure that writes are only made after the WITS device has sent a message to the driver.

Resources

In addition to this user manual, there are a variety of resources available to assist customers, answer questions, provide more detail about specific implementations, or help with troubleshooting specific issues.

Knowledge Base
Whitepapers
Connectivity Guides
Technical Notes
Training Programs
Training Videos
Kepware Technical Support
PTC Technical Support

Index

A

Address Descriptions 17
Advanced Channel Properties 10
Allow Sub Groups 15
Auto Dial 8

B

Baud Rate 7

C

Channel Assignment 11
Channel Properties — General 5
Channel Properties — Write Optimizations 9
Close Idle Connection 8
COM ID 7
Connection Type 6
Create 15

D

Data Bits 7
Data Collection 12
Data Types Description 16
Delete 14
Description 11
Device '<device name>' item '<WITS ID>', Write value length exceeds maximum frame length. Truncating to 1024 bytes 18
Device '<device name>', Abandoning current message after <time in ms> ms of inactivity 18
Device '<device name>', Communications have timed out after <time in seconds> seconds of inactivity 19
Device '<device name>', Invalid delimiter. Expecting '<configured delimiter>'. Discarding remainder of message 19
Device '<device name>', Message parsing error. Discarding remainder of message 19

Device '<device name>', More than 4096 bytes in item '<WITS ID>' data. Discarding remainder of message 20

Device '<device name>', Non-numeric WITS ID found '<WITS ID>'. Discarding remainder of message 20

Device '<device name>', Non-printable character '<hex byte>' found in WITS ID. Discarding remainder of message 20

Device '<device name>', Non-printable character '<hex byte>' in item '<WITS ID>' data. Discarding remainder of message 21

Device '<device name>', Received Null value '<Null value>' for item '<WITS ID>'. Setting tag to bad quality 21

Device '<device name>', Value '<value>' is invalid for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality 21

Device '<device name>', Value '<value>' out of range for item '<WITS ID>' with data type '<data type>'. Setting tag to bad quality 21

Device Properties — General 11

Device Properties — Tag Generation 13

Diagnostics 6

Do Not Scan, Demand Poll Only 13

Driver 6, 11

Duty Cycle 9

E

Error Descriptions 18

F

Flow Control 7

G

Generate 14

H

Help Contents 4

I

ID 12
Idle Time to Close 8
IEEE-754 floating point 10
Initial Updates from Cache 13

M

Model 11
Modem 8

N

Name 11
Network Adapter 8
Non-Normalized Float Handling 10

O

On Device Startup 14
On Duplicate Tag 14
On Property Change 14
Operational Behavior 8
Optimization Method 9
Overview 4
Overwrite 14

P

Parent Group 14
Parity 7
Physical Medium 7

R

Read Processing 9

Report Comm. Errors 8
Request All Data at Scan Rate 13
Request Data No Faster than Scan Rate 12
Resources 23
Respect Client-Specified Scan Rate 12
Respect Tag-Specified Scan Rate 13

S

Scan Mode 12
Serial Communications 6
Serial Port Settings 7
Settings 15
Setup 5
Simulated 12
Stop Bits 7

T

Tag Generation 13
Timeout 10

U

Unable to write to address '<address>' on device '<device name>'. Invalid socket 22

W

Write All Values for All Tags 9
Write Only Latest Value for All Tags 9
Write Only Latest Value for Non-Boolean Tags 9
Write Optimizations 9