ABB Totalflow Driver

Table of Contents

ABB Totalflow Driver

Help version 1.050

CONTENTS

Overview

The ABB Totalflow Driver is designed to work with ABB Totalflow devices that support the native DB1 and DB2 Serial Protocols (which are typically used by ABB's flow computers and analyzers). ABB's Totalflow Communication Interface (TCI) toolkit is used to implement the application stack for the driver.

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.

Setup

Supported Device Families

The following device families are supported under the "Totalflow FCU" model:

6000 Series MicroFLO
6000 XSeriesG4
6000 XSeriesG3
6000 Series FCU

Maximum Number of Channels and Devices

The maximum number of supported channels is 1024. The maximum number of supported devices per channel is 256.

Ethernet Encapsulation

This driver supports Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a serial-to-Ethernet terminal server. It also allows the driver to directly communicate with a device that is equipped with a TCP/IP port. It may be invoked through the Communications group in Channel Properties. For more information, refer to the server help file.

Channel Properties
Device Properties

Channel Properties

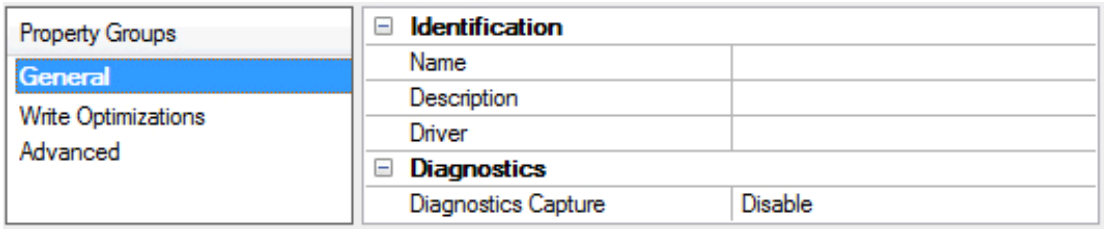
This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link.

The properties associated with a channel are broken in to logical groupings. While some groups are specific to a given driver or protocol, the following are the common groups:

General
Serial Communications
Write Optimization
Advanced
Communication Serialization

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.

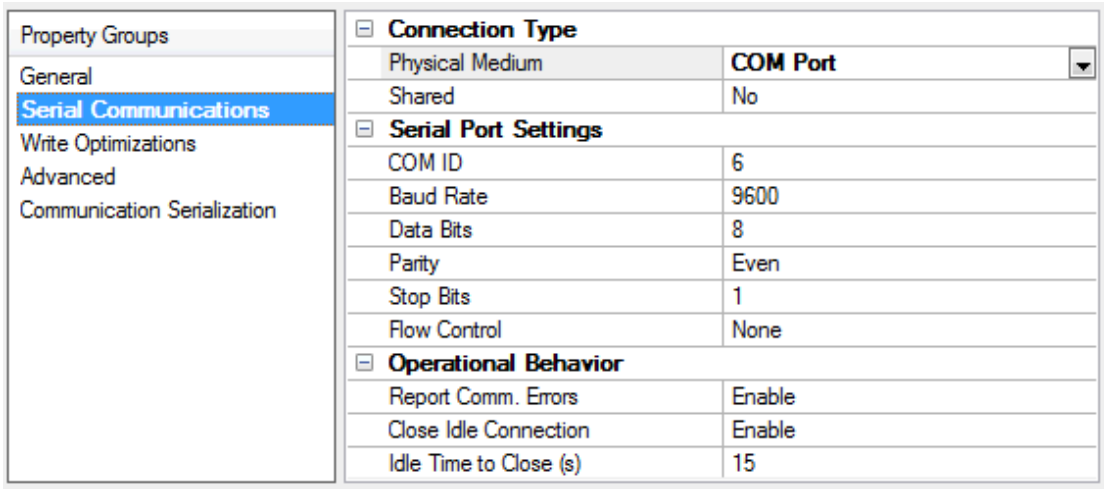
Diagnostics

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

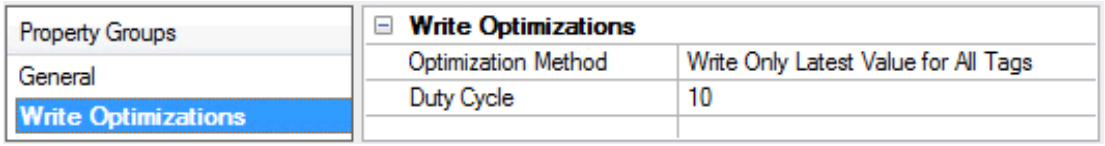
- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific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

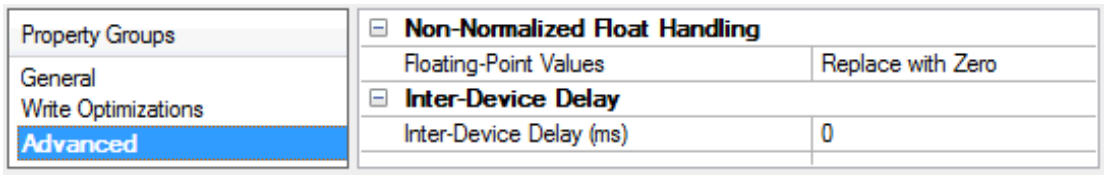
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write 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: Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. 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. 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.lin

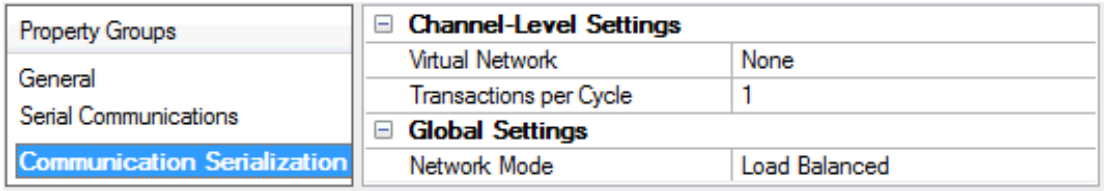
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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.

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 50. The default is None. Descriptions of the options are as follows:

- **None**: This option disables communication serialization for the channel.
- **Network 1 - Network 50**: 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 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:

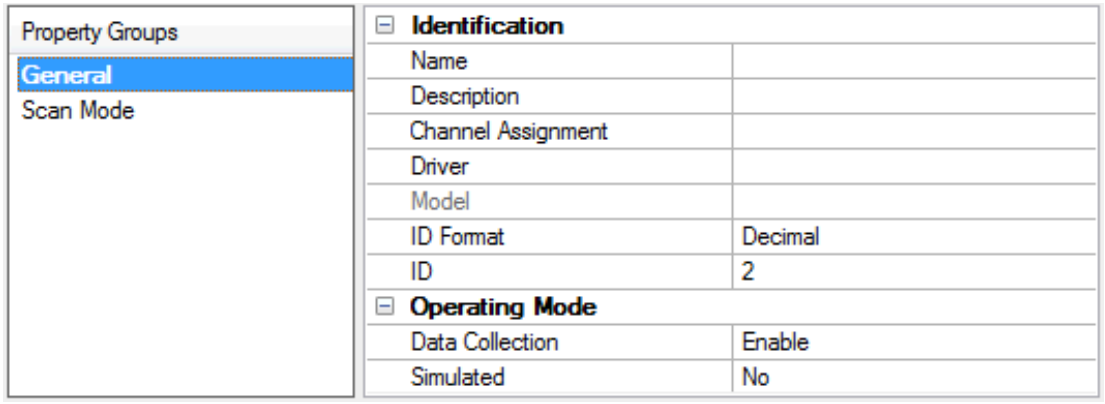
- How many tags must be read from each channel?
- How often is data written to each channel?
- Is the channel using a serial or Ethernet driver?
- Does the driver read tags in separate requests, or are multiple tags read in a block?
- Have the device's Timing properties (such as Request timeout and Fail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

Device Properties

Device properties are organized into the following groups. Click on a link below for details about the properties in that group.

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.

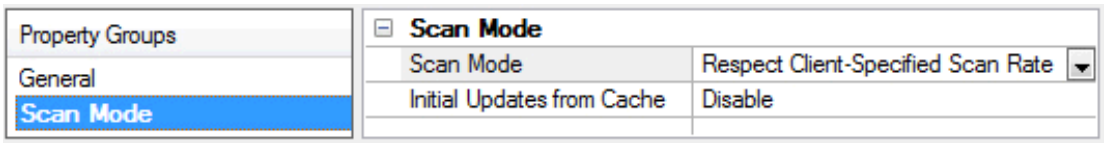
Notes:

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.



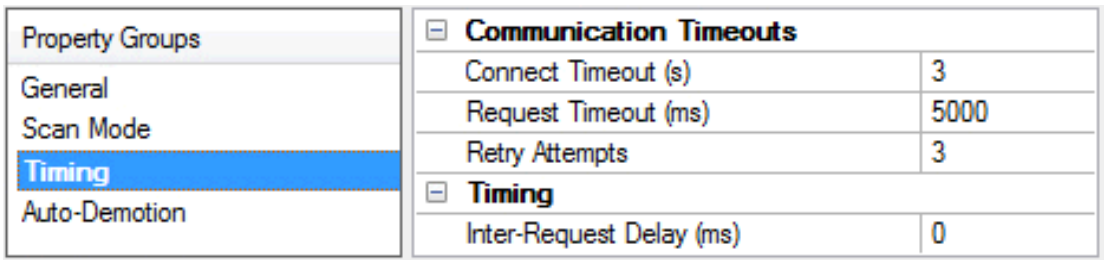
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribed clients. Descriptions of the options are:

- **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 — 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.

Retry Attempts: This property specifies how many times the driver retries 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 retries configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

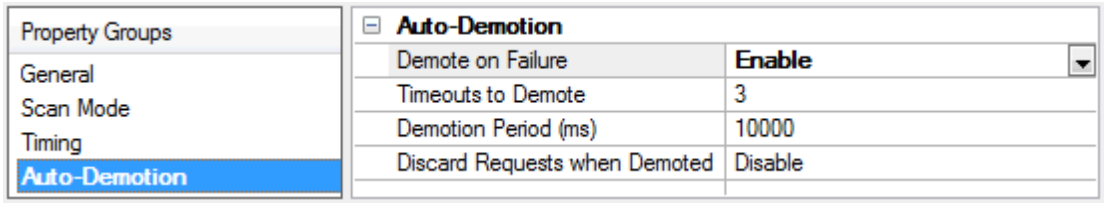
Timing

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.

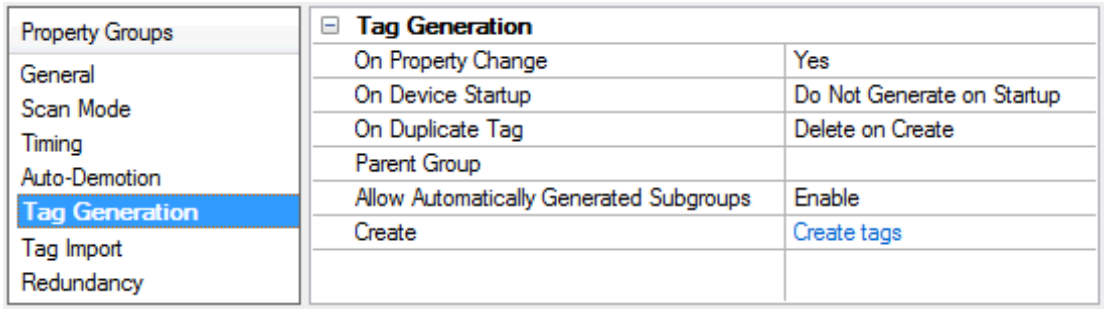
Device Properties — Tag Generation

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.

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.

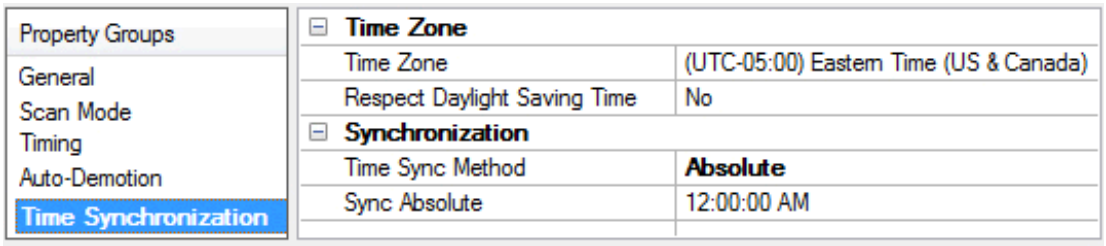
Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the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

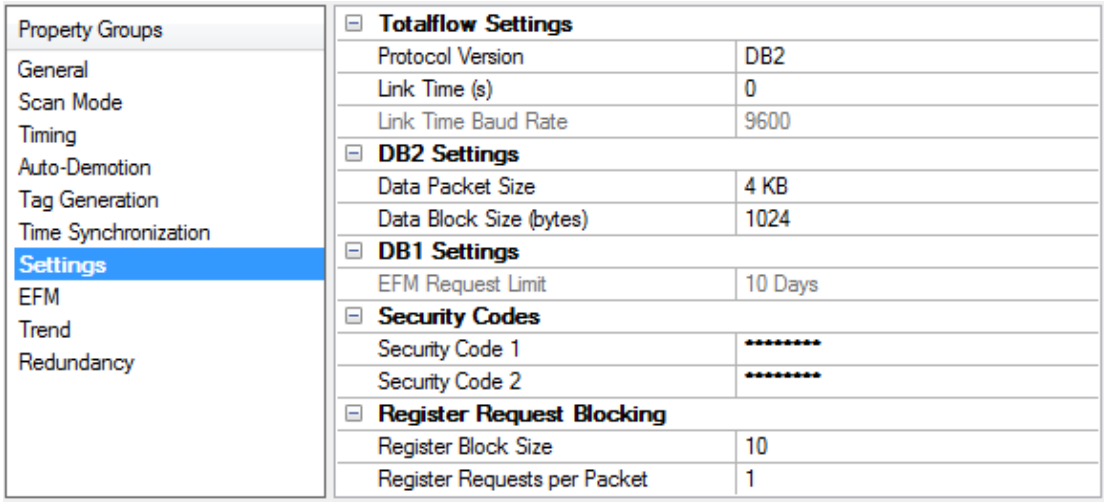
Time Zone: This property specifies the device's time zone. To ignore the time zone, select one of the first four options in the list (which do not have an offset). The default is the time zone of the local system.
Note: The driver uses this property both when synching the device time and when converting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

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).

Device Properties - Settings



Totalflow Settings

- **Protocol Version:** Specify the protocol version to be used for communications. Options include DB1 (not packet-based) and DB2 (packet-based). The default is DB2.
  Note: DB1 cannot communicate with liquid meters and can only communicate with one meter at a time. DB2 allows up to 20 meters to be created.
- **Link Time (s):** Specify the interval that the device opens the receive channel. It is used to determine how many supervisory frames to send so that a remote device's receive channel is on and the request is detected. Options include 0 seconds, 1 second, 2 seconds, and 4 seconds. The default is 0 seconds.
- **Link Time Baud Rate:** This property is required for the Link Time calculation when communicating with a serial device while Ethernet Encapsulation is enabled. When a COM port is being used, this property is fixed to the COM port's baud rate. The valid range is 300 to 256,000 bits per second. The default is 9600 bits per second.

DB2 and DB1 Settings

- **Data Packet Size:** Specify the size of each remote packet transmitted using packet control. Options include 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, and 32 KB. The default is 4 KB. This property affects EFM and trend uploads only and can be adjusted to optimize communications during uploads.
- **Data Block Size:** Specify the size of each remote data block transmitted that can be CRC checked. Options include 128 bytes, 256 bytes, 512 bytes, and 1024 bytes. The default is 1024 bytes. This

property affects EFM and trend uploads only and can be adjusted to optimize communications during uploads.

- **EFM Request Limit:** This property limits the amount of EFM data that is requested to the specified number of days. Options include 3 days, 10 days, and 35 days. The default is 10 days.

Security Codes

Users can configure security codes in the device to provide communications with additional security. The codes are stored securely.

- **Security Code 1:** Specify a four digit code that is used for read-only access. It has a maximum length of 4 characters. The default is 0000.
- **Security Code 2:** Specify a four digit code that is used for read/write access. It has a maximum length of 4 characters. The default is 0000.

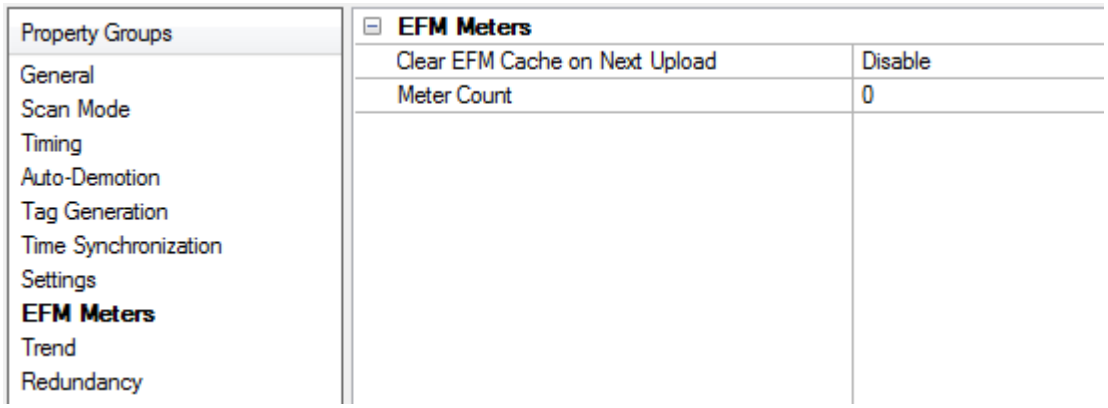
Register Request Blocking

- **Register Block Size:** Specify the maximum number of registers that can be read in a single request. The valid range is 1 to 100. The default is 10.
- **Register Requests per Packet:** Specify the maximum number of register blocks that can be read in a single packet. The valid range is 1 to 16. The default is 1.

Note: These properties are only available when the selected protocol version is DB2.

Device Properties - EFM Meters

EFM Properties

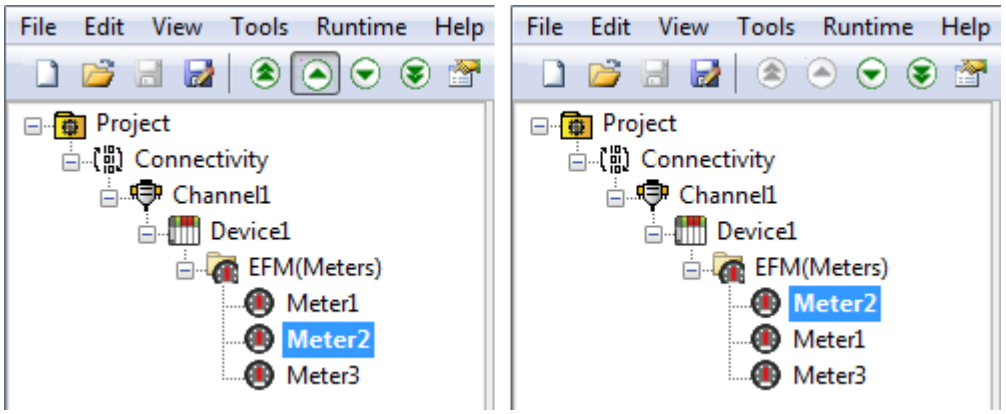


- **Clear EFM Cache on Next Upload:** When this property is enabled any cached EFM data will be cleared from the device during the next upload. Pointer files, which are used to track EFM uploads to prevent uploading the same records twice, will also be removed. All EFM data is re-uploaded. Once the cache is cleared, this property is automatically disabled. The default setting is disabled.
  Note: Changing a meter's order or removing it from the EFM Meter Configuration (see below) causes the EFM cache to be cleared on the next upload. Although this ensures data integrity, users can avoid it by disabling the **Clear Cache on Next Upload** property before applying any changes.
- **Meter Count:** This property displays the number of meters that have been added to the device.

Meter Configuration

Meters can be added, removed, and modified under **Device | EFM (Meters)** in the project tree. The meter order determines the association with the ABB Totalflow Flow Measurement application. The meter order in the project tree must coincide with the order of the Flow Applications in ABB's Portable Configuration and

Calibration Unit (PCCU) tool. To adjust the meter order, select a meter and use the arrow icons in the toolbar accordingly.



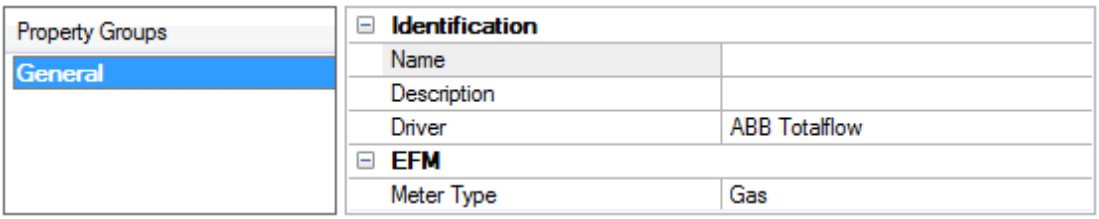
When creating meters, note that the DB1 protocol supports only a single meter and the DB2 protocol supports a maximum of 128 meters. This does not necessarily reflect the number of meters that can be created in the Totalflow device.

Notes:

1. Batch record data is not supported.
2. Liquid meters are not compatible with the DB1 protocol.

Adding a Meter

To add a meter, right-click on **EFM(Meters)** in the project tree and then select **New Meter**.



Description of the properties are as follows:

- **Name:** Specify the meter name. Each meter must be assigned a unique name to request EFM Configuration, Daily History, Hourly History, and Event records from an ABB Totalflow device.
- **Description:** Specify a description for the meter.
- **Driver:** Specify the driver associated with meter.
- **Meter Type:** Specify the meter type (gas or liquid) and must match the application type configured for the meter in PCCU.

Removing a Meter

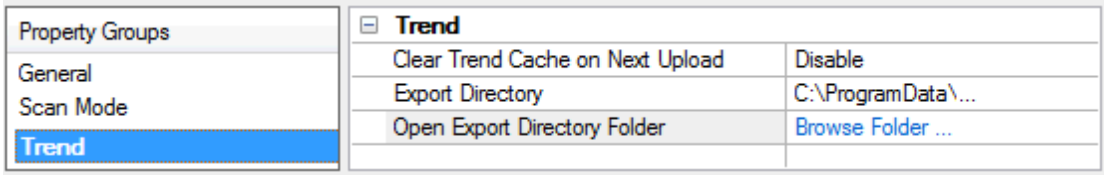
To remove a meter, select the meter and then press the **Delete** key. Alternatively, select **Edit | Delete** from the Edit menu or toolbar.

Displaying Meter Properties

To display the meter properties of a specific meter, double click the meter. Alternatively, select the meter and then select **Edit | Properties** from the Edit menu or toolbar.

Device Properties - Trend

The ABB Totalflow Driver has the ability to upload and export trend data from Totalflow devices via the DB2 protocol. Trend files are exported in comma-separated-value (CSV) file format with each cell contained within double quotations (") and using commas for the field delimiter. The CSV file name is the same as defined in the device with the device ID pre-pended to it. The format is <Device ID>-<Trend File Name>.csv. The device ID should be unique to prevent files from being overwritten.



Clear Trend Cache on Next Upload provides the option to delete the trend cache on the subsequent upload to cause all trend records to be retrieved from the device during the next upload. *See below for more information on trend cache.* Once the cache has been cleared, this option is set to Disable. The default is Disable.

Export Directory: Specifies the path to which all trend files are written when uploaded from this device. *See note below for information about the export directory.*

Open Export Directory Folder Browses the folder specified in Export Directory.

Tip: The default export directory is Documents\<company>\<product>\V<version_number>\Trend\ where the bracketed variables are replaced with the local installation information. The directory is created on first upload. The directory can be changed by browsing to a different folder or typing in a new path. If the folder does not exist, it will be created on the next upload.

Trend Cache

The trend cache is used to track the timestamp of the latest upload for each trend file. It allows the ability to upload only new records from the device.

Note: the trend cache is stored on disk in the form of pointer files so that the cache data is maintained across a server restart. Clearing the trend cache also removes these files from disk.

Reasons for Export Failure

Trend file uploads and exports can fail for various reasons. Failures during upload from a device are generally caused by device configuration or communication problems noted in the event log *(see **Error Descriptions** for more information in these cases)*. Failures during the trend file export can occur in the following ways:

- The file to be written exists and is locked.
- The file to be written is located in a directory for which the server does not have write permissions.
- The server does not have write permissions for the file to be written or the file is read-only.

In these cases, the server cannot export the uploaded data. On the next upload, the same data is uploaded and another export attempted. If problem still exists, the export fails again. All export failures are noted in the event log with a message indicating why the export failed. *See **Error Descriptions** for more information on these failures.*

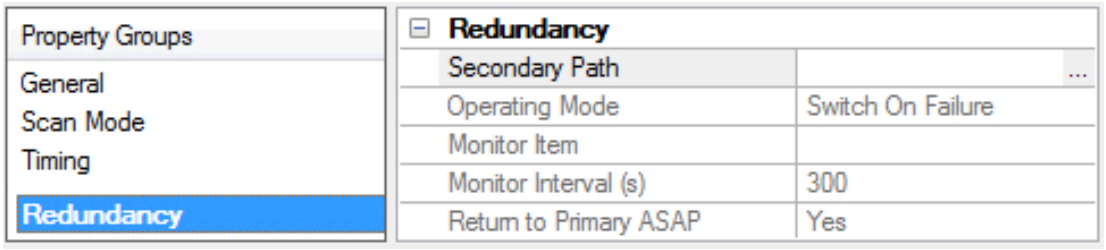
Notes on Export Directory

The following restrictions are placed on the export path.

- The path must be either a valid UNC (\\server\share\) or drive letter (C:\path\) path.
- The root of the path must not be a mapped network drive *(see below for more information)*.
- The path must not contain any characters not supported in Windows folder names.
- The path must be less than 256 characters long.
  Note: The operating system may place more strict restrictions on the path length than 256. On many Windows operating systems, the maximum path length also includes the file name and extension. Therefore, character space must be reserved when creating the path for all trend file names and extensions to be written to disk.

The driver supports exporting trend CSV files to multiple types of media, including drives with removable storage (flash drives or external hard drives) and network drives. Due to the nature of the server runtime, the driver does not have access to mapped network drives. To export data to a network drive, the path must be specified with the UNC format (\\server\share\) and not a mapped drive letter. Any circumstances in which the media to be exported to is not accessible (cases such as network communication difficulties or removable media removed) data is re-uploaded and another attempt at exporting the data made on the next upload.

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.

Automatic Tag Database Generation

The ABB Totalflow Driver can be configured to automatically generate tags for characteristic items and trend files in the device, as well as tags that indicate the status of a trend upload.

The Characteristic Items, created when the protocol version is configured for DB1, are described by three functional groups: Last Volume Period (LVP) Tags, Current Measurement Tags, and Device Setup Tags. All tags are read only. These tags are created in the Device Setup and Measurement groups.

Tags related to trend file uploads are created in a tag group at the device level named "Trend". The trend file upload tags created include one tag per file defined in the device, as well as *TF_UploadAll* and *TF_LastModified* tags. Trend status are also generated: *TS_UploadingTrend*, *TS_UploadingTrendFile*, and *TS_Error (see Address Descriptions for more information on these tags)*.

To generate tags from the device:

1. In the Configuration, select the device for which to generate tags.
2. Right-click and select **Properties...** to open the Device Properties.
3. Select the Tag Generation group.
4. Click the **Create tags** button to initiate tag database creation.
5. Click the **Close** button to exit the grid view.
6. In the Event Log, verify messages confirming successful generation.

Note: An automatic tag generation creates the characteristic items regardless of the server's ability to connect to the device. If the server is not connected to the device or communications are lost during the tag generation, the trend file tags are not created and a warning message is posted in the event log.

For more information about automatic tag generation properties, see the server help file.

See Also:
Address Descriptions
Characteristic Address Descriptions
Trend Address Descriptions
DB1 and DB2 Settings

Data Types Description

Data Type	Description
Bool	Single bit
Char	Signed 8-bit value
Byte	Unsigned 8-bit value
Short	Signed 16-bit value
Word	Unsigned 16-bit value
Long	Signed 32-bit value
DWord	Unsigned 32-bit value
Float	32-bit floating point value
String	Null-terminated character array

Address Descriptions

ABB Totalflow devices organize data by application, array, and register. Although the meaning and type of data is specific to the application type, a fully-qualified ABB Totalflow address requires all three of these parameters. Its syntax is as follows: <*application #*>.<*array #*>.<*register #*>.

Each application is usually designed to perform a specific task, calculation, or function in the ABB Totalflow device. The application numbers are not standardized, meaning that the application number for a Flow Measurement application AGA3-1 can differ from device-to-device. The application numbers are based on the application category and the order in which they are added to the device. The ABB Totalflow Driver follows this addressing convention, supporting the following syntax:

<*Application*>.<*Array*>.<*Register*>/<*bit*>
<*Application*>.<*Array*>.<*Register*>[*row*][*column*]

Note: The valid range for the register identifier fields are 0-255, 0-255, and 0-65535 respectively. The bit index is validated against the specified OPC type. In the event that the register type does not support the specified bit index, the read fails and an appropriate error message is logged once per tag. The product of rows multiplied by columns cannot exceed the maximum register block size of 100. A 1-length row/column is allowed.

For more information, select a link from the list below.

Characteristic Address Descriptions
DB2 Protocol Address Descriptions
Trend Address Descriptions
Statistics Items

Characteristic Address Descriptions

Data Blocking

G1 and G2 devices do not use registers. Data can be requested from a number of categories: the processing is the same for all of them. Data is requested in two blocks based on the method of data request. Characteristics items cannot be requested individually. The driver supports the Current Measurement and/or Device Setup tag categories. Due to blocking limitations, it is recommended that data be retrieved by register for DB2 devices. Only the Characteristic Address Tags are available for DB1 devices (because they do not use a register-based architecture). For more information, refer to **DB2 Protocol Address Descriptions**.

The Characteristic Items are described by three functional groups: Last Volume Period (LVP) Tags, Current Measurement Tags, and Device Setup Tags. All tags are read only.

Last Volume Period (LVP) Tags

ABB Totalflow devices calculate volume in a user-configurable interval between 1 and 60 minutes.

Address	Data Type	Description
LVPALARMS	Byte	Alarms
LVPACF	Float	Accumulated Flow
LVPDP	Float	DP
LVPAP	Float	AP

Address	Data Type	Description
LVPTF	Float	Temperature
LVPVOLUME	Float	Volume
LVPQV	Float	Qv
LVPIRANGE	Float	Turbine Range
LVPICOUNTS	Float	Turbine Counts
LVPEXT	Float	Extension
LVPCP	Float	Cp
LVPFB	Float	Fb
YSTACFLOW	Float	Yesterday's Low ACF
LVPFR	Float	F(r)
YSTACFHIGH	Float	Yesterday's High ACF
LVPY	Float	Y
YSTACFTOTAL	Float	Yesterday's Total
LVPFPB	Float	Pressure Base Factor
LVPFTB	Float	Temperature Base Factor
LVPFTF	Float	Flowing Temperature Factor
LVPFG	Float	Gravity Factor
LVPFPV	Float	Supercompressibility Factor
LVPFA	Float	Auxiliary
LVPFW	Float	Water Vapor Factor
LVPFAUX	Float	Auxiliary Factor
LVPBTU	Float	BTU/SPC
LVPGRAVITY	Float	Specific Gravity
LVPN2	Float	Mole %
LVPCO2	Float	Mole %
LVPH2S	Float	Mole %
LVPH2O	Float	Mole %
LVPHELIUM	Float	Mole %
LVPMETHANE	Float	Mole %
LVPETHANE	Float	Mole %
LVPPROPANE	Float	Mole %
LVPNBUTANE	Float	Mole %
LVPIBUTANE	Float	Mole %
LVPNPENTANE	Float	Mole %
LVPIPENTANE	Float	Mole %
LVPNHEXANE	Float	Mole %
LVPNHEPTANE	Float	Mole %
LVPNOCTANE	Float	Mole %
LVPNNONANE	Float	Mole %

Address	Data Type	Description
LVPNDECANE	Float	Mole %
LVPOXYGEN	Float	Mole %
LVPCO	Float	Mole %
LVPHYDROGEN	Float	Mole %
LVPARGON	Float	Mole %
LVPFIP	Float	FIP
LVPS	Float	F(pv)^2
LVPEV	Float	Ev
LVPFPM	Float	FPM
LVPCORORIF	Float	Corrected orifice diameter for the effect of temperature
LVPFTM	Float	Ftm
LVPCORPIPE	Float	Corrected pipe diameter (ID) for the effect of temperature
LVPPULSECOUNT	Float	Pulse Count
LVPRHOB	Float	Rhob
FLOWPERIOD	Float	Flow Period (seconds)
LVPQM	Float	Qm
LVPCD	Float	Cd

Current Measurement Tags

Address	Data Type	Description
CURACCVOL	Float	Current Accumulated Volume
CURACCVOLTURBINE	Float	Current Accumulated Volume (Turbine)
CURAP	Float	Current Static Pressure
CURDP	Float	Current Differential Pressure
CURMACF	Float	Current Actual MCF
CURTF	Float	Current Temperature
CURFLOW	Float	Current Flow Rate (SCF)
CURVOLTAGE	Float	Current Voltage
YSTACCVOL	Float	Yesterday's Accumulated Volume
YSTMCFTOTAL	Float	Yesterday's Total MCF

Device Setup Tags

Address	Data Type	Description
CONTRACTHOUR	Byte	Contract Hour (0-23)
STREAMID	Uint32	4 Bytes. In ASCII, the Stream ID is *iiii-aa-ss*. Bytes 0 & 1: AIU number (*iiii*); 0000 to 9999, with the bytes reversed from Microsoft Byte 2: Analyzer number (*aa*); 1-32 Byte 3: Stream number in the analyzer (*ss*); 1-32

Address	Data Type	Description
FIXEDAXONERR	Bool	If STREAMATTACHED is 1 and the received analysis is not correct or, if no analysis is received, then: False = Use last known good analysis True = Use fixed analysis
STREAMATTACHED	Bool	False = Always use fixed analysis True = Attached to AIU
RTDINSTALLED	Bool	False = No True = Yes
USEMEASTEMP	Bool	False = Fixed Temp True = Live
CHECKSEC	Bool	Use Security
MONELORIF	Bool	False = Stainless True = Monel
USEFB	Bool	Use FB in Calculations
USES	Bool	Use S (F(pv)^2); Turbine Only
USEFR	Bool	Use Flow Rate
USEFTC	Bool	Use Temperature Correction Factor
USEY	Bool	Use Expansion Factor
USEFPC	Bool	Use Pressure Correction Factor
USEFTB	Bool	Use Temperature Base Factor
USEFPB	Bool	Use Pressure Base Factor
USEFTF	Bool	Use Flowing Temperature Factor
USEFG	Bool	Use Gravity Factor
USEFPV	Bool	Use Super Compressibility Factor
USEFA	Bool	Auxiliary
LCCONTACT	Bool	Charger Low
DPLOWCONTACT	Bool	DP Low Contact
DPHICONTACT	Bool	DP High Contact
APLOWCONTACT	Bool	AP Low Contact
APHICONTACT	Bool	AP High Contact
REMSENSECONTAC	Bool	Use Remote Sense
AUTORESET	Bool	Automatic Reset
VOLSPCONTACT	Bool	Use Volume Set Point
USEFW	Bool	Use Water Vapor Factor
USEFAUX	Bool	Use Auxiliary Factory
FIXEDAP	Float	Fixed AP
USEMEASAP	Float	Use Measured AP
FB	Float	Basic Orifice Factor
ORIFICE	Float	Orifice Diameter (inches)

Address	Data Type	Description
PIPE	Float	Pipe Diameter (inches)
GRAVITY	Float	Specific Gravity
DPLOWLIM	Float	DP Low Limit
DPHILIM	Float	DP High Limit
APLOWLIM	Float	AP Low Limit
APHILIM	Float	AP High Limit
CO2	Float	Mole % CO2
N2	Float	Mole % N2
APLOWCAL	Float	AP Calibration: Low Point
APMIDCAL	Float	AP Calibration: Mid Point
APHICAL	Float	AP Calibration: High Point
DPLOWCAL	Float	DP Calibration: Low Point
DPMIDCAL	Float	DP Calibration: Mid Point
DPHICAL	Float	DP Calibration: High Point
ZEROCUTOFF	Float	DP Zero Cutoff
TBASE	Float	Temperature Base
PBASE	Float	Pressure Base
FIXEDTF	Float	Fixed Temperature
TEMPBIAS	Float	Temperature Bias
VISC	Float	Centipose
RSPH	Float	Ratio of Specific Heats
FT	Float	Mass Flow Calibration Temperature Coefficient
FP	Float	Pressure
BTU	Float	BTU/SCF
ACFLOWLIM	Float	Turbine: Low Limit
ACFHILIM	Float	Turbine: High Limit
FAUX	Float	Auxiliary Factory
K	Float	Turbine
CALCUNITS	Byte	Bit Definitions 0 = US Units 1 = IP Units 2 = MT Units 3 = SI Units
ACFLOWLIM	Float	Accumulated Flow Limit
ZMETHOD	Float	Bit Definition 0 = NX19, fixed Ft, Fp 1 = NX19 GCN or GCNM 2 = NX19 GCN

Address	Data Type	Description
		3 = NX19 GCNM 7 = AGA-8 HGCN 8 = AGA-8 GCN Limited Number of High Pressure FCUs 10= AGA-8 GCNM 11= AGA-8 Gross 1992 (Gross 2) 12= AGA-8 Detail 1992
TAPLOCATION	Bool	0 = Downstream 1 = Upstream
TAPTYPE	Bool	0 = Flange 1 = Pipe
VOLCALCMETHOD	Byte	Value Definition 0= No Volume Calculation 1 = AGA-3 1985 2= AGA-3 1992, API 14.3
FIXEDCD	Bool	0 = Fixed 1 = Calculated CD
PIPEREFTEMP	Float	Reference Temperature (F)
ORIFREFTEMP	Float	Orifice Temperature (F)
ZBA	Float	Z of Air at Base Conditions
VOLCALCPER	Uint16	Volume Calculation Period in Seconds
VOLLOGPER	Uint32	Volume Log Period in Seconds
H2S	Float	Mole %
H2O	Float	Mole %
HELIUM	Float	Mole %
METHANE	Float	Mole %
ETHANE	Float	Mole %
PROPANE	Float	Mole %
NBUTANE	Float	Mole %
IBUTANE	Float	Mole %
NPENTANE	Float	Mole %
IPENTANE	Float	Mole %
NHEXANE	Float	Mole %
NHEPTANE	Float	Mole %
NOCTANE	Float	Mole %
NNONANE	Float	Mole %
NDECANE	Float	Mole %
OXYGEN	Float	Mole %
CO	Float	Mole %
ORIFEXP	Float	Orifice Plate Coefficient of Expansion
PIPEEXP	Float	Pipe Coefficient of Expansion

Address	Data Type	Description
BAROP	Float	Barometric Pressure
USECALCCD	Bool	Use Calculated Cd
HYDROGEN	Float	Mole %
ARGON	Float	Mole %
APMIDLOWCAL	Float	AP Calibration: Mid-Low
APMIDHICAL	Float	AP Calibration: Mid-High
DPMIDLOWCAL	Float	DP Calibration: Mid-Low
DPMIDHICAL	Float	DP Calibration: Mid-High
SQRTAVG	Float	Square Root Mean
VOLSP	Float	Volume Set Point
TFLOWLIM	Float	Temperature Low Limit
TFHILIM	Float	Temperature High Limit
FLOWLOWLIM	Float	Flow Low Limit
FLOWHILIM	Float	Flow High Limit
PASSWORDMODE	Bool	0 = No Password Mode 1 = Mode Enabled
FIRSTAX	Bool	At Least One Good Analyzer
PRIMARYELEM	Byte	0= Orifice 1= Turbine
REPORTUNITS	Byte	Report Unite Bit Definition: 0 = US 1 = IP 2 = MT 3 = SI

DB2 Protocol Address Descriptions

DB2 Protocol addressing information applies to G2, G3, and G4 devices.

Data Blocking

Due to blocking limitations, it is recommended that data be retrieved by register for DB2 device. A DB2 register request consists of a base address (in the form of <*application*>.<*array*>.<*register*>) and a register count. The registers retrieved in a request are always in the same Totalflow array and have the same data type. For example, if 10 registers are requested beginning at "9.3.0," the response contains data for "9.3.0" through "9.3.9" with a uniform data type. The driver blocks data in the same manner.

Register data is retrieved from the TCI toolkit as an array of bytes. The data type is inferred from the size of the array and the number of elements that were requested. The element size within a data block is fixed based on the largest atomic type (because the data type cannot be accurately deduced until data is retrieved from the device). Encoding the data length provides greater flexibility for the client tag data type. This means that users do not need to know the Totalflow data type when defining client tags in the driver.

Data type mismatches result in truncation. There is no special handling for floating point values. Examples are as follows:

1. A Word tag is defined with an address of "9.3.0". The ABB Totalflow device has this register defined as an int32. The driver reads the tag and sets its quality to Bad because a Word cannot properly represent all int32 values. An error message is posted to the Event Log.
2. A DWord tag is defined with an address of "9.5.0". The ABB Totalflow device has this register defined as a Float. The driver reads the tag, receives the value "1.523," and stores it in the block memory. When the tag is updated, its value is 1069740458 (the binary equivalent of 1.523).

Strings are not blocked by the driver due to their fixed size. A runtime error (including an Event Log message) is posted when a string register is assigned a blockable type. Strings are not converted to any other data type. Furthermore, multiple register requests can be included in a single DB2 transaction. This means that the driver can service multiple block tag reads with a single device request/response. The number of register requests per packet is user-configurable. The driver pools tags based on the following criteria:

- All tags will be read or write.
- All tags must be for the same device.
- The number of tags is limited to the number specified by the user.

Bit-within-Word Booleans

Bit-within-Word Booleans provide a mechanism for interpreting register data as a bit field. For reads, this involves using a mask to determine the Boolean value for a desired bit (which is specified in the tag's address). In order to write bit values, the driver must perform a Read/Modify/Write operation to ensure that only a single bit is being set.

Trend Address Descriptions

Trend tags apply to all ABB Totalflow devices that support the DB2 protocol and trend logging.

Trend File Tags

Syntax Example: Channel.Device.TF_UploadAll

Tag	Data Type	Description	Access
TF_<trend_file_name>	Boolean	This tag is used to upload a single file specified by <trend_file_name> where trend_file_name must match the name of a trend file defined in the device. Writing any value to this tag initiates an upload. This tag always reads 0.	Write Only
TF_LastModified	Boolean	This tag is used to upload the trend file that was last modified in the device. This tag reserves the name LastModified; therefore trend files should not be created on the device with this name. Writing any value to this tag initiates an upload. This tag always reads 0.	Write Only
TF_UploadAll	Boolean	This tag is used to upload all trend files on the device. This tag reserves the name UploadAll; therefore trend files should not be created on the device with this name. Writing any value to this tag initiates an upload. This tag always reads 0.	Write Only
TF_UploadLastHours	Word	This tag is defines how many hours of records, from the current time, will be uploaded. It only specifies the number of	Read/Write

Tag	Data Type	Description	Access
		hours and does not initiate the upload. The value is used only for the next upload initiated by a single trend file tag, last modified tag, or upload all tag and returns to zero when the upload is complete. This tag reserves the name UploadLastHours; therefore trend files should not be created on the device with this name. **Notes:** • The clear cache property has no effect on uploads that use this tag. Upon completion of an upload that uses this tag, the trend cache is updated with the timestamp of the newest record uploaded. • If an existing CSV file isn't consumed before using this tag, the file could be missing data or have duplicate data. • This tag uses the time zone/DST properties in the Time Synchronization group in the Device Properties. These properties must match the device's time zone/DST settings for this tag to work properly.	

Trend File Tag Validation Requirements:

The following address requirements must be met when creating trend file tags:

- Length (including the "TF_") must be at least four (4) characters.
- Length (including the "TF_") must be less than 66 characters. This is due to the 63 character restriction enforced by the TCI toolkit.
- The address must be valid ASCII characters.
- The file name should not contain white space other than the standard space character.
- Prohibited characters are double quotes (") and 'at' symbol (@).

Notes:

- The file's Scan Status in the device must be set to "On" to upload the trend file from the device.
- These addresses are not case sensitive.

Trend Status Tags

Syntax Example: Channel.Device.TS_Uploading

Tag	Data Type	Description	Access
TS_Uploading	Boolean	This tag indicates whether the driver is currently retrieving a trend file from the device and exporting trend data to disk.	Read Only
TS_UploadingFile	String	This tag indicates the name of the trend file that the driver is currently uploading from the device.	Read Only
TS_ErrorOnLastUpload	Boolean	This tag indicates that an error has occurred during the	Read Only

Tag	Data Type	Description	Access
		most recent trend file upload. If an error occurs during a single file or upload all request, this tag is set to TRUE. The tag is reset upon completion of the next successful upload, including uploads that result in no new data.	
TS_LastUploadStart	Date	This tag indicates the time that the most recent trend file upload began. All times are reported in local time. If no upload has started, the tag reports 01/01/1601 00:00:00.000.	Read Only
TS_LastUploadEnd	Date	This tag indicates the time that the most recent trend file upload finished. All times are reported in local time. If no upload has completed, the tag reports 01/01/1601 00:00:00.000.	Read Only
TS_LastUp-loadDurationSec	Double	This tag indicates the amount of time (in seconds) the last trend file upload took to complete. This tag is updated on upload completion whether the upload succeeds or fails.	Read Only
TS_ErrorCount	DWord	This tag increments every time a trend file upload fails due to a communication or export error. Writing any value to this tag resets the counter to zero.	Read/Write

Blocking Trend File Uploads

When possible, trend files are uploaded in multiple small blocks of records instead of one large block to optimize data throughput over noisy communication links. The Totalflow firmware provides the information required to block uploads for up to 15 trend files via registers *(see Notes below)*. If there are more than 15 trend files configured on a device, some of the trend file uploads will not be uploaded in small blocks. One exception is that all trend file uploads initiated with a non-zero TF_UploadLastHours value are blocked regardless of the number of trend files configured on the device.

Notes:

- The driver records the timestamp of the newest record in the file at the beginning of a trend file upload. If records are being logged at a high rate and the file has wrapped, it is possible that the driver may upload a few records less than the full trend file size when uploading from a cleared cache state. This is expected behavior as the driver will upload those records on the next upload.
- The following register addresses are used when determining if a trend file upload can be blocked. If a trend file's name is in one of these 15 registers when an upload for the file is initiated, the file's upload is blocked. If the trend file's "Scan Status" is set to "Off" in the device, the trend file's name will not appear in these registers.

  <trend app num>.241.0 through <trend app num>.241.14

  where <trend app num> is the application slot number where the trend system application is instantiated.

Statistics Items

Statistical items use data collected through additional diagnostics information, which is not collected by default. To use statistical items, Communication Diagnostics must be enabled. To enable Communication Diagnostics, right-click on the channel in the Project View and click **Properties | Enable Diagnostics**. Alternatively, double-click on the channel and select **Enable Diagnostics**.

Channel-Level Statistics Items

The syntax for channel-level statistics items is <*channel*>.*_Statistics*.

Note: Statistics at the channel level are the sum of those same items at the device level.

Item	Data Type	Access	Description
_CommFailures	DWord	Read/Write	The total number of times communication has failed (or has run out of retries).
_ErrorResponses	DWord	Read/Write	The total number of valid error responses received.
_ExpectedResponses	DWord	Read/Write	The total number of expected responses received.
_LastResponseTime	String	Read Only	The time at which the last valid response was received.
_LateData	DWord	Read/Write	The total number of times that a driver tag's data update occurred later than expected (based on the specified scan rate).
_MsgResent	DWord	Read/Write	The total number of messages sent as a retry.
_MsgSent	DWord	Read/Write	The total number of messages sent initially.
_MsgTotal	DWord	Read Only	The total number of messages sent (both _MsgSent + _MsgResent).
_PercentReturn	Float	Read Only	The proportion of expected responses (Received) to initial sends (Sent) as a percentage.
_PercentValid	Float	Read Only	The proportion of total valid responses received (_TotalResponses) to total requests sent (_MsgTotal) as a percentage.
_Reset	Bool	Read/Write	Resets all diagnostic counters. Writing to the _Reset Tag causes all diagnostic counters to be reset at this level.
_RespBadChecksum	DWord	Read/Write	The total number of responses with checksum errors.
_RespTimeouts	DWord	Read/Write	The total number of messages that failed to receive any kind of response.
_RespTruncated	DWord	Read/Write	The total number of messages that received only a partial response.
_TotalResponses	DWord	Read Only	The total number of valid responses received (_ErrorResponses + _ExpectedResponses).

Statistical items are not updated in simulation mode *(see device general properties)*.

Device-Level Statistics Items

The syntax for device-level statistics items is <*channel*>.<*device*>.*_Statistics*.

Item	Data Type	Access	Description
_CommFailures	DWord	Read/Write	The total number of times communication has failed (or has run out of retries).
_ErrorResponses	DWord	Read/Write	The total number of valid error responses received.
_Expec-tedResponses	DWord	Read/Write	The total number of expected responses received.
_LastResponseTime	String	Read Only	The time at which the last valid response was received.
_LateData	DWord	Read/Write	The total number of times that a driver tag's data update occurred later than expected (based on the specified scan rate).
_MsgResent	DWord	Read/Write	The total number of messages sent as a retry.
_MsgSent	DWord	Read/Write	The total number of messages sent initially.
_MsgTotal	DWord	Read Only	The total number of messages sent (both _MsgSent + _ MsgResent).
_PercentReturn	Float	Read Only	The proportion of expected responses (Received) to initial sends (Sent) as a percentage.
_PercentValid	Float	Read Only	The proportion of total valid responses received (_TotalRe-sponses) to total requests sent (_MsgTotal) as a percentage.
_Reset	Bool	Read/Write	Resets all diagnostic counters. Writing to the _Reset Tag causes all diagnostic counters to be reset at this level.
_RespBadCheck-sum	DWord	Read/Write	The total number of responses with checksum errors.
_RespTimeouts	DWord	Read/Write	The total number of messages that failed to receive any kind of response.
_RespTruncated	DWord	Read/Write	The total number of messages that received only a partial response.
_TotalResponses	DWord	Read Only	The total number of valid responses received (_ErrorRe-sponses + _ExpectedResponses).

Statistical items are not updated in simulation mode *(see device general properties)*.

Error Descriptions

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

Communication Error Messages

A communication error occurred while <reading/writing> address <tag address> on device <device name>: <verbose communication error>.
A communication error occurred while reading address <tag address> on device <device name>: Device did not respond.
A communication error occurred while reading address <tag address> on device <device name>: General remote NACK error.
A communication error occurred while reading address <tag address> on device <device name>: Illegal register read or write.
A communication error occurred while reading address <tag address> on device <device name>: Invalid data structure.
A communication error occurred while reading address <tag address> on device <device name>: Received data had a CRC error.
A communication error occurred while reading address <tag address> on device <device name>: Transaction ID did not match.

Device-Specific Error Messages

Bit index out of range for register address <tag address> on device <device name>.
Could not write data for trend file <trend file> on device <device> to disk. <reason>.
<Device name> - Failed to read EFM pointer file. <Extended error>.
<Device name> - Failed to write EFM pointer file. <Extended error>.
<Device name> - Failed to read trend pointer file. <Extended error>.
<Device name> - Failed to write trend pointer file. <Extended error>.
Device <device name> is not responding.
EFM type mismatch for meter <meter name> on device <device name>. The meter is configured for <meter type> in the server but it is a <meter type> meter in the device.
Extra data revision not supported. The EFM output file for <meter name> on device <device name> will be missing some data. Extra data revision = <extra data revision>, extra data size = <extra data size>.
Extra data size is 0 bytes. The EFM output file for <meter name> on device <device name> will be missing some data. Extra data revision = <extra data revision>, extra data size = <extra data size>.
Failed to send communication request for address <tag address> on device <device name>
Invalid address for register block <App>.<Array>.<Register X> - <App>.<Array>.<Register Y> on device <device name>.
Invalid data type <data type> for address <tag address> for device <device name>
Register blocking error, verify that data type <data type> is valid for address <tag address> on device <device name>.
Serialization of EFM data to temporary file <file name> failed. Reason: <file I/O error>.

Trend file configuration changed for trend file <trend file> on device <device>. Old data moved to <file name>. New data written to <file name>.
Trend file <trend file name> does not exist on device <device name>.
Trend upload expected new records for trend file <trend file> on device <device name>, but no records were retrieved.
Unable to load TCI Toolkit extradata.ini file, EFM feature may not work properly.
Unable to write to <address> on device <device name>.

A communication error occurred while <reading/writing> address <tag address> on device <device name>: <verbose communication error>.

Error Type:
Warning

Possible Cause:
There was an error in communications.

Solution:
1. For more information, refer to the verbose error message.
2. Verify the ABB Totalflow device's configuration.

A communication error occurred while reading address <tag address> on device <device name>: Device did not respond.

Error Type:
Warning

Possible Cause:
The protocol version is not set correctly.

Solution:
Update the protocol version to match that of the device. *For more information, refer to* ***Settings***.

A communication error occurred while reading address <tag address> on device <device name>: General remote NACK error.

Error Type:
Warning

Possible Cause:
There is a response mismatch.

Solution:
Increase the Request Timeout to allow time for the response to complete.

A communication error occurred while reading address <tag address> on device <device name>: Illegal register read or write.

Error Type:
Warning

Possible Cause:
The device does not recognize the register address.

Solution:
Verify that the register address is valid.

A communication error occurred while reading address <tag address> on device <device name>: Invalid data structure.

Error Type:
Warning

Possible Cause:

1. The data that was requested is not supported by the destination device.
2. The meter for which data was requested does not exist in the destination device.

Solution:

1. Verify that the device supports the DB2 protocol.
2. Verify that, for each meter in the server, a meter also exists on the destination device. Ensure that the order of meters in the server matches the order of meters in the device.

A communication error occurred while reading address <tag address> on device <device name>: Received data had a CRC error.

Error Type:
Warning

Possible Cause:

1. The response's integrity could not be validated.
2. There is a response mismatch resulting in an invalid CRC check.

Solution:

1. Investigate for sources of noise or errors in the transmission media.
2. Increase the Request Timeout to prevent unnecessary request retries.

A communication error occurred while reading address <tag address> on device <device name>: Transaction ID did not match.

Error Type:
Warning

Possible Cause:

1. The specified tag data type and Totalflow register data type do not match. This prevents the tag from being blocked with other Totalflow registers.
2. There is a response mismatch.

Solution:

1. Ensure that the tag data type and the register data type match.
2. Increase the Request Timeout to allow time for the response to complete.

Bit index out of range for register address <tag address> on device <device name>.

Error Type:
Warning

Possible Cause:
The bit index specified in the tag address is incompatible with the register data type.

Solution:
Verify the register data type. Then, adjust the bit index accordingly.

Could not write data for trend file <trend file> on device <device> to disk: <reason>.

Error Type:
Warning

Possible Cause:
The source of the problem is indicated by the <reason> message. Common <reason> messages:

- Unable to create path <path>.
  - Access denied.
  - Permission denied.
- Unable to open file <file name>.
  - Access denied.
  - Permission denied.
  - No such file or directory.
- Trend file configuration changed, unable to move old data to another file.

Solution:

- The path specified for export does not exist and the server was unable to create it. There are two common reasons this can occur:
  - The server does not have permission to the drive or path to create a new folder. Ensure that the server runtime service has permissions for the drive and path to which it must write.
  - The specified path is too long for the current operating system. Choose a shorter path.
- The server was not able to open for read or create the output file.
  - Verify that the file is not locked by another program and is not specified as read only.
  - Verify that the server has permission to read/write to the directory and file.
  - “No such file or directory” appears if the specified path, combined with the output file name, exceeds the system maximum path length. To resolve this problem, use a shorter path to export data.

- When the configuration of the trend file (number of variables, variable names, and units) changes and the trend file already exists on disk, the server attempts to move the old trend data to another file so that the new configuration is not written into the old file with the old configuration. This message is posted if the server is unable to complete this move. Ensure that the file to be moved does not already exist and is not locked, that the disk is not full, and that the server has write permissions in the export directory.

<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 ABB Totalflow 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.

For more information, refer to the extended error.

<Device name> - Failed to read trend 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 trend pointer cache was read.
2. The trend pointer cache file is corrupt.

Solution:
The ABB Totalflow Driver automatically generates a new trend pointer file; however, the server re-polls (uploading all trend data) during the next poll for trend files in the device.

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 attempts to update the EFM pointer file periodically, in addition to when the server is shutdown. If the pointer file cannot be written, the server re-polls (uploading all EFM data) during the next EFM poll for meters in the device.

For more information, refer to the extended error.

<Device name> - Failed to write trend 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 trend pointer cache was written.

Solution:
The server attempts to update the trend pointer file periodically, in addition to when the server is shutdown. If the pointer file cannot be written on server shutdown, the next upload after restart begins uploading records with the timestamp from the last successful pointer file write.

Note:
For more information, refer to the extended error.

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 configured incorrectly.
3. The named device may have been assigned an incorrect device ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device property.

Solution:

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

EFM meter <meter name> on device <device name> is invalid.

Error Type:
Warning

Possible Cause:
The specified meter name or type was modified or the meter was removed while being referenced by an active EFM Exporter meter.

Solution:
Update the EFM Exporter's meters to reference the latest EFM meters for the affected device.

EFM type mismatch for meter <meter name> on device <device name>. The meter is configured for <meter type> in the server but it is a <meter type> meter in the device.

Error Type:
Warning

Possible Cause:
The meter is configured as a liquid meter in the server, but it is a gas meter in the device or vice versa.

Solution:
Change the value of the meter type property in the server to match the meter type in the device.

Extra data revision not supported. The EFM output file for <meter name> on device <device name> will be missing some data. Extra data revision = <extra data revision>, extra data size = <extra data size>.

Error Type:
Warning

Possible Cause:
The device is running a firmware version that is not supported by the driver.

Solution:
Verify that the device is running a firmware version that is supported by the driver. Contact Technical Support.

Extra data size is 0 bytes. The EFM output file for <meter name> on device <device name> will be missing some data. Extra data revision = <extra data revision>, extra data size = <extra data size>.

Error Type:
Warning

Possible Cause:
The device is running a firmware version that is not supported by the driver.

Solution:
Verify that the device is running a firmware version that is supported by the driver. Contact Technical Support.

Failed to send communication request for address <tag address> on device <device name>.

Error Type:
Warning

Possible Cause:
Failed to send the request for the specified tag and device.

Solution:
Verify the values of the Totalflow protocol and communication port properties.

Invalid address for register block <App>.<Array>.<Register X> - <App>.<Array>.<Register Y> on device <device name>.

Error Type:
Warning

Possible Cause:

1. An attempt has been made to reference a nonexistent location in the specified device.
2. There is a response mismatch.

Solution:

1. Verify the tags that are assigned to the addresses in the specified range on the device. Then, eliminate ones that reference invalid locations.
2. Increase the Request Timeout to allow time for the response to complete.

Invalid data type <data type> for address <tag address> for device <device name>.

Error Type:
Warning

Possible Cause:
A tag data type has been selected that cannot properly represent the Totalflow register data type.

Solution:
Select a tag data type that matches the Totalflow register data type.

Register blocking error, verify that data type <data type> is valid for address <tag address> on device <device name>.

Error Type:
Warning

Possible Cause:
The specified tag data type and Totalflow register data type do not match. This prevents the tag from being blocked with other Totalflow registers.

Solution:
Ensure that the tag data type and the register data type match.

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.

Trend file configuration changed for trend file <trend file> on device <device>. Old data moved to <file name>. New data written to <file name>.

Error Type:
Warning

Possible Cause:
The configuration of the trend file on the Totalflow device changed after the data was written to disk for that file. Changes detected include the number of records being logged to the file, the order in which they appear, as well as the description or units associated with each variable. Rather than continuing to write data to a file with a header that does not match, the old file is renamed so that the new data can be written to a new file with that name.

Solution:
Before modifying trend file configuration in the device, consume and move or delete the old file.

Trend file <trend file name> does not exist on device <device name>.

Error Type:
Warning

Possible Cause:
The requested trend file is not configured on the device.

Solution:
Verify that the requested trend file is configured on the device or correct the file or path.

Trend upload expected new records for trend file <trend file> on device <device name>, but no records were retrieved.

Error Type:
Warning

Possible Cause:
Records are being logged at a rate that overwrites the entire file before the driver can upload the file.

Solution:
Adjust the logging rate and trend file upload interval so that records in the file are not overwritten between uploads.

Unable to load TCI Toolkit extradata.ini file, EFM feature may not work properly.

Error Type:
Warning

Possible Cause:
The extradata.ini file is corrupt or not located in the same directory as the TCI Toolkit dll (tcidll.dll).

Solution:
Verify that a valid extradata.ini file exists in the same directory as the TCI Toolkit dll (tcidll.dll). If the extradata.ini file cannot be found or installed in the directory, contact Technical Support.

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 configured incorrectly.
3. The named device may have been assigned an incorrect Device ID.

Solution:

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

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W