MTConnect Driver

Table of Contents

MTConnect Driver

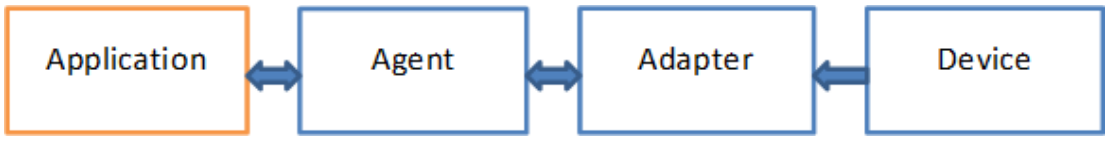
Help version 1.023

CONTENTS

Overview

The MTConnect Driver provides a reliable way to connect MTConnect agents to OPC Client applications, including HMI, SCADA, Historian, MES, ERP, and many custom applications.

MTConnect is a lightweight, open, and extensible protocol designed to exchange data between shop-floor equipment and software applications used for monitoring and data analysis. The standard is owned and maintained by the MTConnect Institute. The basic architecture of a MTConnect system is:



The agent, adapter, and device can be combined or separate and implemented in a device, a PC, or a combination. The MTConnect Driver runs at the application level. TCP/IP is used to communicate with both the adapter and the agent by addressing their IP and port numbers. The default port number for an adapter is 7878 and the default port for an agent is 5000, though 80, the default HTTP port is frequently used. An agent responds to an application request with well-formed XML, where the adapter sends pipe-separated data to an agent.

The MTConnect Driver works with all V1.1 and V1.2 compliant MTConnect agents and provides partial support for V1.3. The driver currently supports the following MTConnect commands:

- **PROBE:** This command reads the device's address space (which is represented by the agent and used for automatic tag generation).
- **CURRENT:** This command reads a data item's snapshot value.

Best Practices

The MTConnect Driver always connects to an agent; whatever port number is assigned to the target agent must be specified as the Port Number in the Device Properties | Settings. It is possible to connect the MTConnect Driver to an adapter port, which fails because the driver requires XML format and cannot interpret pipe-separated data. It is a good practice to verify the ports by connecting to them with a browser to confirm valid data results.

Example:
If both the agent and adapter are running on a PC with IP address 10.20.21.73, connect using these addresses:

http://10.20.21.73:5000
http://10.20.21.73:7878

Port 5000 returns well-formed XML.
Port 7878 returns pipe-separated data; e.g. 2015-01-15T16:40:58.616Z|Xloadc|NORMAL||||

An XML validation error in the Event Log means the XML sent from the agent failed some facet or restriction in a schema file in the folder driver/MTConnectClient or the version of the schema file specified in the XML stream from the agent isn't in the driver/MTConnectClient folder. To temporarily bypass all schema validation for the agent stream, remove all schemas from the driver/MTConnectClient folder. The MTConnect version used to stream data between agent and driver is determined by the agent and can be set in the agent configuration file.

Notes:

1. Restart or reinitialize the server runtime to load schema changes.
2. MTConnect version 1.3, released in November 2014, added support for XML schema 1.1. This MTConnect Driver doesn't support enhancements that rely on XSD 1.1.

See Also:
Error Descriptions

Setup

Supported Devices

Any V1.1 compliant MTConnect Agent.

Note: Users can import additional schemas into the server's root *\Drivers\MTConnectClient* folder.

Maximum Number of Channels and Devices

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

Device ID

The Device ID specifies the IP address/host name of the agent in addition to the name of the device being referenced in the agent. The format is *<IPAddress\HostName>/<Device-ID>*. The device name has to be of NMTOKEN XML type.

Example

173.45.237.40/VMC-3Axis

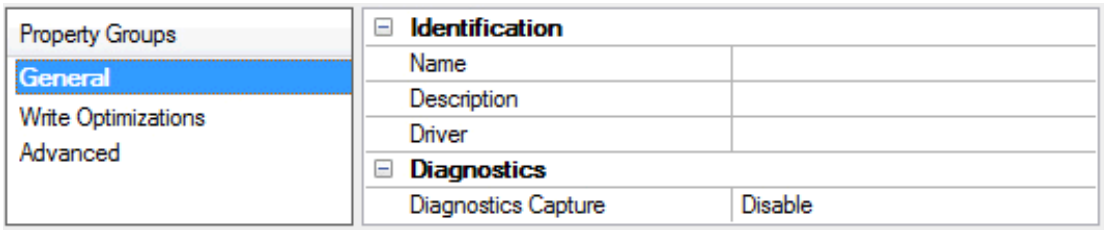
Note: For more information, refer to **Extensible Markup Language (XML) 1.0 (Fifth Edition)**.

Auto-Demotion

A non-responsive agent with many tags may impede communications with other agents on the same channel due to the timeout period used on each successive query to the non-responsive agent. As such, it is recommended that auto-demotion be used for each agent when communications are unreliable. For more information on auto-demotion, refer to the server help file.

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.

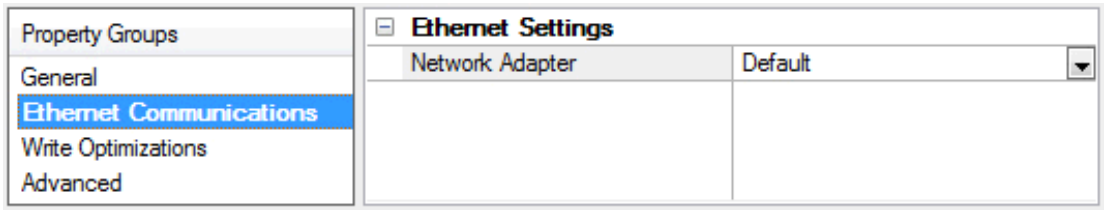
Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are 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 - Ethernet Communications

Ethernet Communication can be used to communicate with devices.

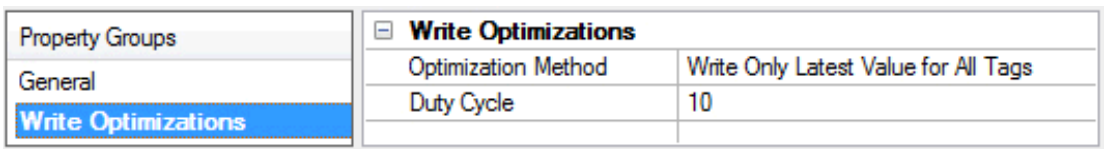


Ethernet Settings

Network Adapter: Specify the network adapter to bind. When Default is selected, the operating system selects the default adapter.

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:

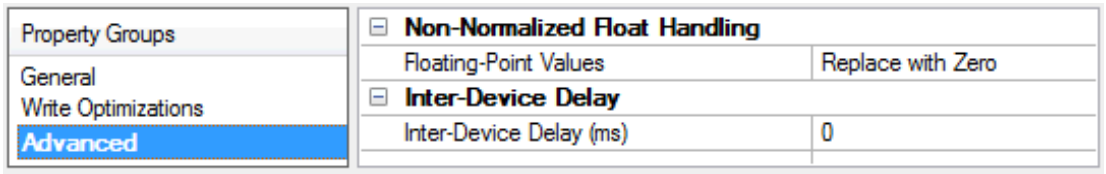
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing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

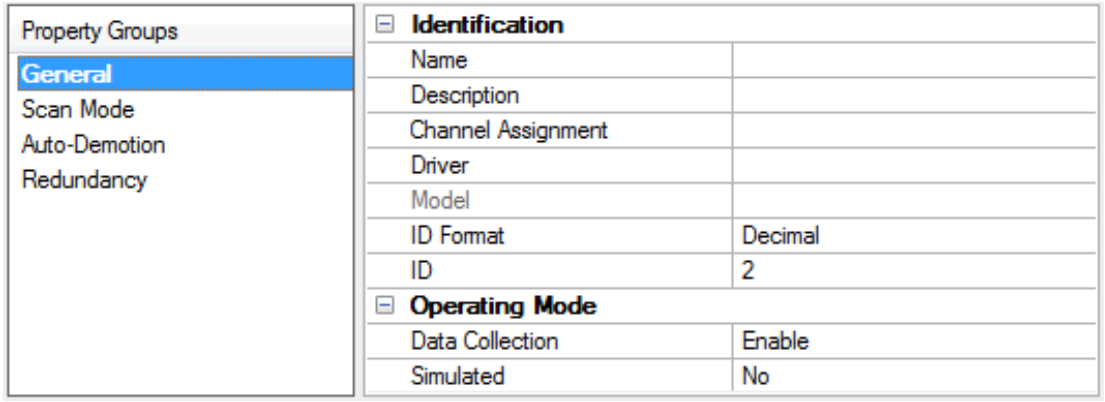
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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

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

Device Properties - General

A device represents a single target on a communications channel. 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 driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The ID format can be Decimal, Octal, and Hexadecimal.

Note: If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver. For more information, refer to the driver's help documentation.

Operating Mode

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

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

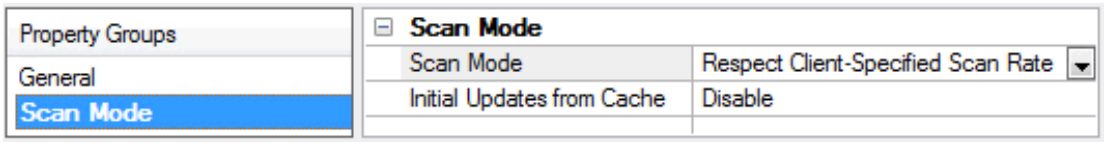
Notes:

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

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

Device Properties - Scan Mode

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



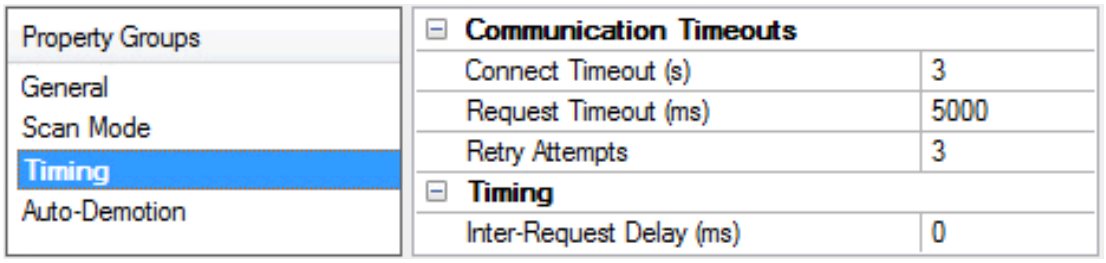
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribed clients. Descriptions of the options are:

- **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.

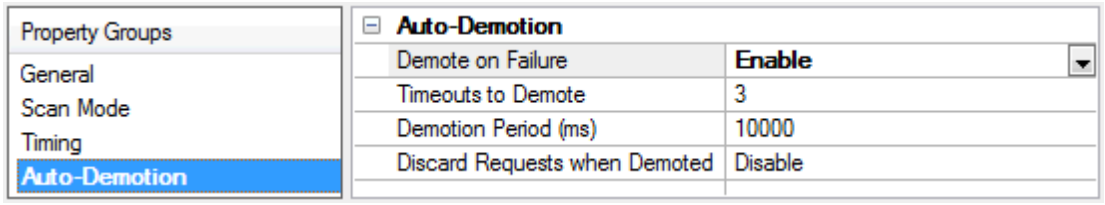
Timing

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.

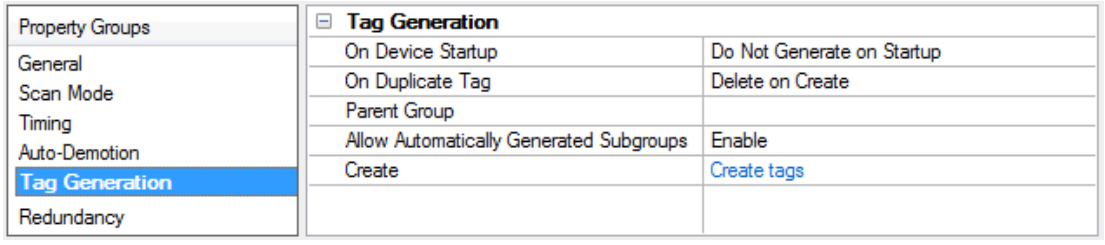
Device Properties - Tag Generation

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

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

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

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

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

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

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

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

Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server

generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.

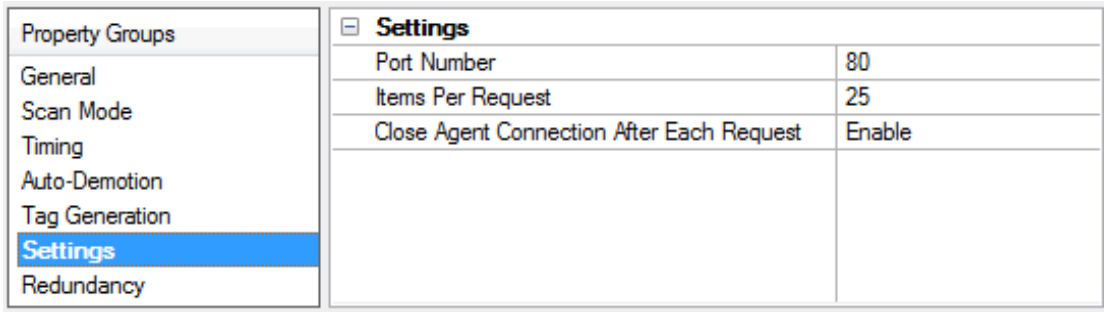
Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the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

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

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

Device Properties - Settings

These properties are used when communicating with the agent.

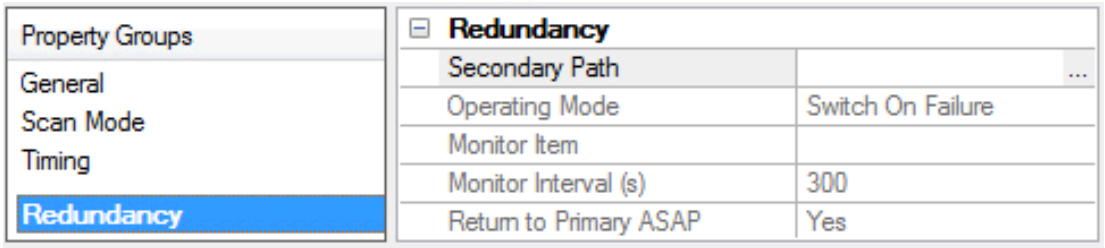


Port Number: This property specifies the port number. The valid range is 1 to 65535. The default setting is 80.

Items Per Request: This property specifies how many data items will be bundled together in each read request. The valid range is 1 to 100. The default setting is 25.

Close Agent Connection After Each Request: When enabled, this option will close the socket connection with the agent after each read request. A new connection will be opened for the subsequent read. The default setting is enabled.

Device Properties - Redundancy

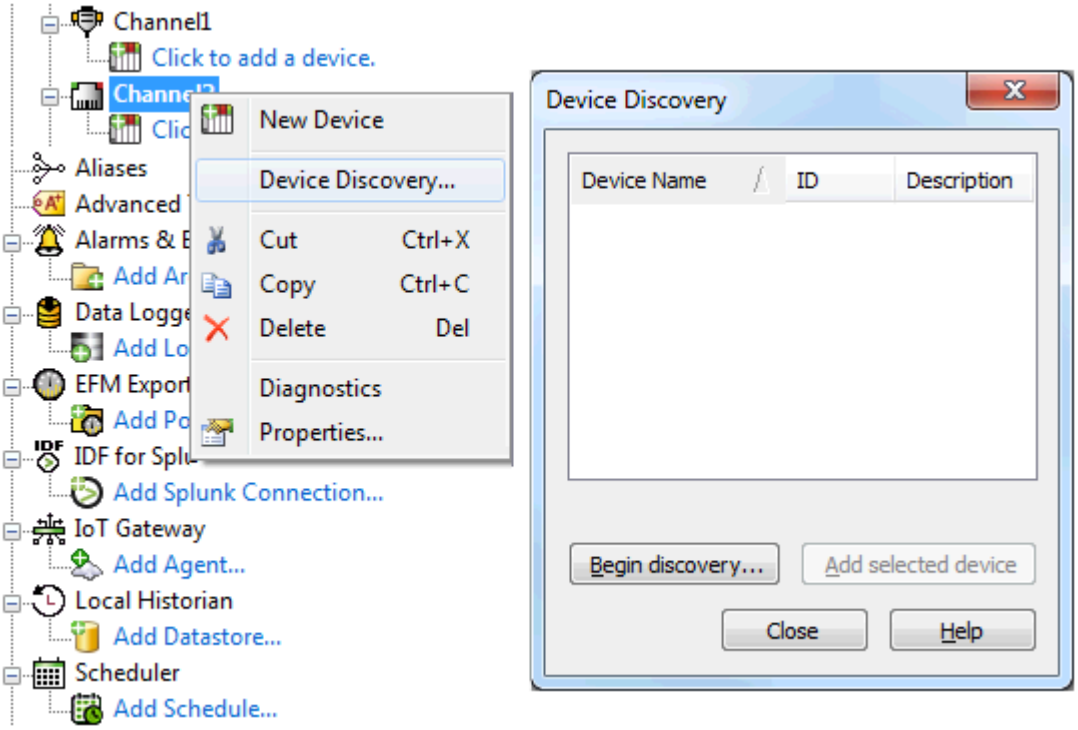


Redundancy is available with the Media-Level Redundancy Plug-In.

Consult the website, a sales representative, or the user manual for more information.

Device Discovery Procedure

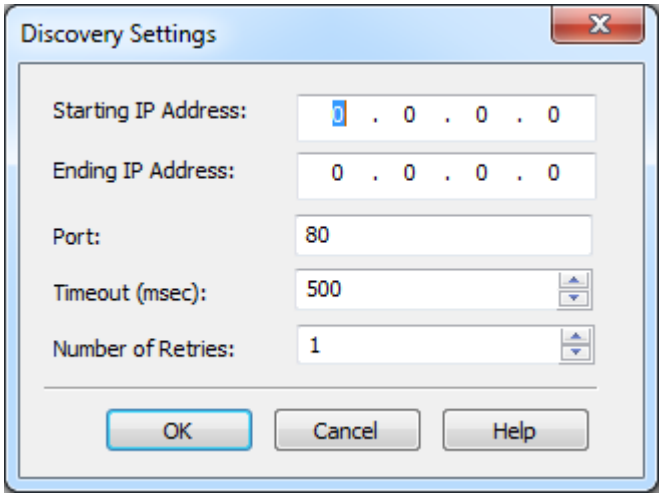
Device Discovery is available for drivers that support locating devices on the network. Once devices are found, they may be added to a channel. The maximum number of devices that can be discovered at once is 65535.



1. Select the channel in which devices should be discovered and added.
2. Right click on the channel node and select **Device Discovery...**
3. Click the **Begin discovery...** button to start the discovery process.
4. Specify the discovery properties, which are driver-specific, such as address range, timeout, discovery scope.
5. Click **OK**.
6. Devices discovered populate the dialog with the following information / headings **Name**, **ID**, **Description**.
7. If any discovered device is of interest, select that device and click **Add selected device...**.
8. Click **Close**.

Device Discovery Settings

The MTConnect Driver can locate agents on the network. Once agents are located, they can be added to the channel as a server device. The maximum number of agents that can be discovered at once is 65535. For more information on device discovery, refer to the server help file.



Descriptions of the properties are as follows:

- **Starting IP Address:** This property specifies the starting IP address. The default setting is 0.0.0.0.
- **Ending IP Address:** This property specifies the ending IP address. The default setting is 0.0.0.0.
- **Port:** This property specifies the port number. The valid range is 1 to 65535. The default setting is 80.
- **Timeout (msec):** This property specifies the time that the driver will wait for a connection to be made with an agent, as well as the time that the driver will wait on a response from the agent before giving up and going on to the next request. The default setting is 500 milliseconds.
- **Number of Retries:** This property specifies the number of times that the driver will retry a message before giving up and going on to the next message. The default setting is 1.

Automatic Tag Database Generation

The MTConnect Driver has the ability to automatically generate tags from the MTConnect agent for the device. When doing so, the driver will use the PROBE command to receive all existing data items for the device. Tags will then be created that correspond to each received data item.

An MTConnect device is made up of components that may have further sub-components. Data items are organized inside components. The automatically generated tags will be organized in the same hierarchy; that is, in tag groups that correspond to each component or sub-component.

Note: For more information on automatic tag database generation, refer to the server help file.

Data Types Description

Data Type	Description
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
String	ASCII text string
Double	64-bit Floating point value bit 0 is the low bit bit 63 is the high bit

Assigning Data Types

Tags that are created manually must also assign the data type manually. Users must assign the correct data type because the MTConnect Driver does not have any information based on the tag address (or Data Item ID). The default data type for manually created tags is String.

Automatically generated tags will automatically assign the data type based on the MTConnect standard. For more information, refer to *MTConnect® Standard Part 2 – Components and Data Items Version 1.1.0* and *MTConnect® Standard Part 3 – Streams, Events, Samples, and Condition Version 1.1.0*.

The following data type mapping tables are for automatically generated tags.

SAMPLE Data Items

Data Item Type	Data Type
ACCELERATION	Double
AMPERAGE	Double
ANGLE	Double
ANGULAR_ACCELERATION	Double

Data Item Type	Data Type
ANGULAR_VELOCITY	Double
AXIS_FEEDRATE	Double
DISPLACEMENT	Double
FREQUENCY	Double
LOAD	Double
PATH_FEEDRATE	Double
PATH_POSITION	String
PH	String
POSITION	Double
PRESSURE	Double
SPINDLE_SPEED	Double
TEMPERATURE	Double
TORQUE	Double
VELOCITY	Double
VOLTAGE	Double
WATTAGE	Double

EVENT Data Items

Data Item Type	Data Type
ACTIVE_AXES	String
AVAILABILITY	String
BLOCK	String
CODE	String
CONTROLLER_MODE	String
COUPLED_AXES	String
AXIS_COUPLING	String
DIRECTION	String
DOOR_STATE	String
EXECUTION	String
EMERGENCY_STOP	String
LINE	DWord
MESSAGE	String
PART_COUNT	Integer
PART_ID	String
PATH_MODE	String
POWER_STATUS	String
POWER_STATE	String
PROGRAM	String
ROTARY_MODE	String

Data Item Type	Data Type
TOOL_ID	String
WORKHOLDING_ID	String

Note: All CONDITION type data items are assigned the String data type.

Address Descriptions

The MTConnect Driver specifies addresses using the Data Item ID. This ID is case sensitive, and must adhere to the W3C Standard ID type. As such, the ID must start with a colon, comma, underscore, or letter (A-Z or a-z). It can then be followed with numbers, letters, a hyphen, or a period. For more information, refer to **Extensible Markup Language (XML) 1.0 (Fifth Edition)**.

Note: All addresses are read only.

Error Descriptions

The following categories of messages may be generated. Click on the link for a list of related messages.

Address Validation

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

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

Error Type:

Warning

Possible Cause:

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

Solution:

Modify the requested data type in the client application.

Device address <address> contains a syntax error.

Error Type:

Warning

Possible Cause:

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

Solution:

Re-enter the address in the client application.

See Also:

Address Descriptions

Device address <address> is read only.

Error Type:

Warning

Possible Cause:

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

Solution:
Change the access mode in the client application.

Device Status Messages

The following messages may be generated. Select a link from the list below for a description of the message.

Device <device name> is not responding.

Error Type:
Serious

Possible Cause:

1. The connection between the driver (Host PC) and the agent is broken.
2. The connection's communication parameters are incorrect.
3. 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 Host PC and the agent.
2. Verify that the specified communication parameters match those of the agent.
3. Increase the Request Timeout property so that the entire response can be handled.

Read for the data item <address> on device <device> failed. Reason: Agent reported data is UNAVAILABLE.

Error Type:
Warning

Possible Cause:
The value of the data item received from the agent is UNAVAILABLE. This indicates that the agent was not able to read the latest value of the data item from the device. The value is currently indeterminate. UNAVAILABLE is a valid value when the data source is not connected or the data source is unable to retrieve information. The UNAVAILABLE value persists until the connection is restored and a new value can be retrieved. This state does not imply the device is no longer operational; it only implies that the state cannot be determined.

Solution:
Wait for some time so that the agent is able to read the data item from the device. It is possible that a data item will remain in this state indefinitely (such as a temperature sensor that becomes permanently disconnected from the agent).

Read for the data item <address> on device <device> failed. Reason: Response from agent failed schema validation.

Error Type:
Serious

Possible Cause:
The XML response sent by the agent for the CURRENT request is not valid as per the MTConnectStreams_ n.n.xsd/ MTConnectError_n.n.xsd schema published by MTConnect (where n.n is the version of the agent, i.e. 1.1, 1.2, 1.3).

Solution:
Ensure that the MTConnect schemas are in the path that the driver expects them *(see the MTConnect section in the **Overview**).*

Note:
If the solution fails, contact the agent's Technical Support.

Read for the data items <address> on device <device> failed. Reason: Error in creating MTConnect packet.

Error Type:
Serious

Possible Cause:
An unexpected internal error occurred when creating the MTConnect CURRENT packet.

Solution:
Contact Technical Support for help.

Read for the data items <address> on device <device> failed. Reason: Internal error.

Error Type:
Serious

Possible Cause:
An unexpected driver error occurred.

Solution:
Contact Technical Support for help.

Read for the data items <address> on device <device> failed. Reason: Invalid data item ID.

Error Type:
Serious

Possible Cause:
The requested Data Item ID does not exist in the agent.

Solution:
Correct the Data Item ID and ensure that it exists in the agent.

Read for the data items <address> on device <device> failed. Reason: Invalid response from agent.

Error Type:
Serious

Possible Cause:

1. A misbehaving agent sent an invalid MTConnect response.
2. The entity that the driver is assuming to be an agent is not an MTConnect agent.

Solution:
Contact the agent's Technical Support.

Read for the data items <address> on device <device> failed. Reason: Received HTTP error <error-code>.

Error Type:
Serious

Possible Cause:
The cause depends on the HTTP error code.

Solution:
The solution depends on the HTTP error code.

See Also:
HTTP Response Codes

Read for the data items <address> on device <device> failed. Reason: Received MTConnect error <error-string>.

Error Type:
Serious

Possible Cause:
The cause depends on the MTConnect error.

Solution:
The solution depends on the MTConnect error.

See Also:
MTConnect Error Codes

Read for the data items <address> on device <device> failed. Reason: Unable to connect to the agent.

Error Type:
Serious

Possible Cause:
1. The agent that the IP address/host name pointed to in the device ID is not running.
2. An incorrect IP address/host name has been specified in the device ID.
3. An incorrect port number has been specified.

Solution:
1. Ensure that the agent is running.
2. Correct the IP address/host name so that it points to a valid agent.
3. Correct the port number to match the agent at the specified IP address.

Read for the data items <address> on device <device> failed. Reason: Unable to transmit request to the agent.

Error Type:
Serious

Possible Cause:
The agent disconnected (abruptly or otherwise).

Solution:
The agent will attempt to re-connect with the agent on the subsequent read transaction. Ensure that the agent is up and running.

Read for the data items <address> on device <device> failed. Reason: XML Validation Error: The node is neither valid nor invalid because no DTD/schema declaration was found.

Error Type:
Serious

Possible Cause:
The specified MTConnectDevices_x.x.xsd XML file is a different version than the one installed with the server. The server installs:

- MTConnectDevices_1.1.xsd
- MTConnectStreams_1.1.xsd
- MTConnectError_1.1.xsd
- MTConnectDevices_1.2.xsd
- MTConnectStreams_1.2.xsd
- MTConnectAssets_1.2.xsd (new in version 1.2)
- MTConnectError_1.2.xsd
- MTConnectDevices_1.3.xsd
- MTConnectStreams_1.3.xsd
- MTConnectAssets_1.3.xsd
- MTConnectError_1.3.xsd

Solution:
Use a browser to connect to the MTConnect device directly, then find the XML code being used. This code will be listed in the first few lines of the XML code. Follow the link to the specified XSD location and download the specified versions of the MTConnect*.xsd files. Copy the files to the *\Drivers\MTConnect* folder located in the server's root folder, leaving the default V1.1 XSD files in the folder as well.

Note:
If this solution fails, contact the agent's Technical Support.

<Schema file> failed to load; may cause validation errors.

Error Type:
Serious

Possible Cause:

1. The schema file is not well formed or is invalid.
2. The schema file contains features beyond XML schema Version 1.0.

Solution:

1. Verify that the file passes XSD validation checks.
2. This driver supports XML schema Version 1.0. Remove references to newer features and re-validate.

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

Error Type:
Warning

Possible Cause:
The specified network adapter cannot be bound to for communications because it cannot be located in the system device list. This usually occurs when a project has been moved from one PC to another, and when the project specifies a network adapter instead of using the default. The server will fall back to the default adapter.

Solution:
Either change the Network Adapter property to Default or select a new adapter. Then, save the project and try connecting.

Automatic Tag Database Generation Messages

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

Unable to generate a tag database for device <device>. Reason: Agent did not respond.
Unable to generate a tag database for device <device>. Reason: Agent returned MTConnect error: <error-string>.
Unable to generate a tag database for device <device>. Reason: Error in creating MTConnect packet.
Unable to generate a tag database for device <device>. Reason: HTTP error: <error-code>.
Unable to generate a tag database for device <device>. Reason: Internal error.
Unable to generate a tag database for device <device>. Reason: Invalid response from agent.
Unable to generate a tag database for device <device>. Reason: Response from agent failed schema validation.
Unable to generate a tag database for device <device>. Reason: Unable to connect to the agent.
Unable to generate a tag database for device <device>. Reason: Unable to transmit request to the agent.

Unable to generate a tag database for device <device>. Reason: Agent did not respond.

Error Type:
Serious

Possible Cause:
The agent that the IP address/host name pointed to in the Device ID is not running.

Solution:
Ensure that the agent is running.

Unable to generate a tag database for device <device>. Reason: Agent returned MTConnect error: <error-string>.

Error Type:
Serious

Possible Cause:
1. The device name is incorrect and does not exist in the agent.
2. The cause is as indicated by the MTConnect error string.

Solution:
1. Change the device name so that the new name exists in the agent. Users should also ensure that case is correct, because the device name is case sensitive in MTConnect.
2. The solution depends on the MTConnect error string that was received.

Unable to generate a tag database for device <device>. Reason: Error in creating MTConnect packet.

Error Type:
Serious

Possible Cause:
An unexpected internal error occurred during the creation of the MTConnect PROBE packet.

Solution:
Contact Technical Support for help.

Unable to generate a tag database for device <device>. Reason: HTTP error: <error-code>.

Error Type:
Serious

Possible Cause:
The cause depends on the HTTP error code.

Solution:
The solution depends on the HTTP error code.

See Also:
HTTP Response Codes

Unable to generate a tag database for device <device>. Reason: Internal error.

Error Type:
Serious

Possible Cause:
An unexpected driver error occurred.

Solution:
Contact Technical Support for help.

Unable to generate a tag database for device <device>. Reason: Invalid response from agent.

Error Type:
Serious

Possible Cause:

1. A misbehaving agent sent an invalid MTConnect response.
2. The entity that the driver is assuming to be an agent is not an MTConnect agent.

Solution:
Consult the agent's Technical Support.

Unable to generate a tag database for device <device>. Reason: Response from agent failed schema validation.

Error Type:
Serious

Possible Cause:
The XML response sent by the agent for the PROBE request is not valid as per the MTConnectDevices_ n.n.xsd/ MTConnectError_n.n.xsd schema published by MTConnect (where n.n is the version of the agent, i.e. 1.1, 1.2, 1.3).

Solution:

1. Ensure that the MTConnect schemas are in the path where the driver expects them *(see the MTConnect section in the **Overview**).*
2. Consult the agent's Technical Support.

Unable to generate a tag database for device <device>. Reason: Unable to connect to the agent.

Error Type:
Serious

Possible Cause:

1. The agent that the IP address/host name pointed to in the Device ID is not running.
2. An incorrect IP address/host name has been specified in the Device ID.

Solution:

1. Ensure that the agent is running.
2. Ensure that the IP address/host name is correct.

Unable to generate a tag database for device <device>. Reason: Unable to transmit request to the agent.

Error Type:
Serious

Possible Cause:
The agent disconnected (abruptly or otherwise).

Solution:
The agent will attempt to re-connect with the agent on the subsequent auto generate transaction. Users should ensure that the agent is up and running.

MTConnect Error Codes

Error Code	Description
UNAUTHORIZED	The request did not have sufficient permissions to perform the request.
NO_DEVICE	The device specified in the URI could not be found.
INVALID_URI	The URI provided was incorrect.
INVALID_ REQUEST	The request was not one of the three specified requests.
INTERNAL_ ERROR	The agent did not behave correctly. Contact the software provider.
INVALID_PATH	The path could not be parsed. The syntax is invalid.
UNSUPPORTED	A valid request was provided but the agent does not support the feature or request type.

HTTP Response Codes

Response Code	Title	Description
400	Bad Request	The request could not be interpreted.
500	Internal Error	There was an internal error in processing the request. This will require Technical Support to resolve.
501	Not Implemented	The request could not be handled on the server because the specified functionality was not implemented.

Index

A

C

D

E

G

H

I

M

N

O

P

R

S

T

U

W