

MQTT Client Driver

© 2025 PTC Inc. All Rights Reserved.

Table of Contents

MQTT Client Driver	1
Table of Contents	2
Welcome to the MQTT Client Driver Help Center	4
Overview	5
Setup	6
Channel Properties – General	6
Tag Counts	6
Channel Properties – Advanced	7
Channel Properties – MQTT Server	7
Channel Properties – Timing	8
Device Properties – General	9
Operating Mode	10
Tag Counts	10
Device Properties – Scan Mode	11
Device Properties – Tag Generation	11
Data Types Supported through Automatic Tag Generation	12
Data Types Description	14
Timestamp Behavior	15
Quality Behavior	15
Tag Address Descriptions	16
Tag Addressing Examples	16
Event Log Messages	19
Datatype conversion failed. Device = '<device>', Address = '<address>', Expected = '<datatype>', Actual = '<datatype>'	19
Unexpected character found while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	19
Invalid Unicode character found while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	19
Missing matching quote while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	19
Invalid value found while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	20
Invalid number found while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	20
Unexpected token found within an array while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	20
Unexpected token found within an object while parsing MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	20
Unexpected token found while looking for the end of the MQTT payload. Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.	20
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>'	21
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = '<non-localized reason>'	21
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Error Code = '<code>'	22

Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = '<reason>', Error Code = '<code>'.	22
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = 'Connection refused; MQTT Version not supported'.	22
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = 'Connection refused; Client ID rejected'.	23
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = 'Connection refused; MQTT server unavailable'.	23
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = 'Connection refused; Bad Username or Password'.	23
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = 'Connection refused; Not authorized to connect'.	23
Failed to connect to MQTT server. Channel = '<channel>', Server = '<hostname:port>', Reason = 'Connection refused', Error Code = '<code>'.	23
Connection with MQTT server lost. Channel = '<channel>', Server = '<hostname:port>'.	24
Unable to subscribe to topic. Channel = '<channel>', Topic = '<topic>'.	24
Unable to unsubscribe from topic. Channel = '<channel>', Topic = '<topic>'.	24
MQTT server is not able to support the requested QoS. Subscribed with a lower QoS. Channel = '<channel>', Topic = '<topic>', QoS Requested = <QoS>, QoS Subscribed With = <QoS>.	24
Operation cancelled by user.	25
Operation cancelled due to a system event.	25
Operation cancelled due to a property change.	25
No publishes received from MQTT server on specified topic.	25
MQTT version automatically selected by server. Channel = '<channel>', Server = '<hostname:port>', MQTT Version = <version>.	26
Unable to generate a tag database for '<channel>'.<device>': <Reason>.	26
Index	27

Welcome to the MQTT Client Driver Help Center

This help center is the user documentation for Kepware MQTT Client Driver. This help center is updated regularly to reflect the latest functionality and information.

[Overview](#)

What is the MQTT Client Driver?

[Setup](#)

How do I configure a device for use with this driver?

[Data Types Description](#)


What data types does this driver support?

[Address Descriptions](#)

How do I address a data location on an MQTT client?

[Event Log Messages](#)

What messages are produced by the MQTT Client Driver?

 *Some of the messages displayed in the Event Log are forwarded from other protocols. Consult the vendor document for additional information.*

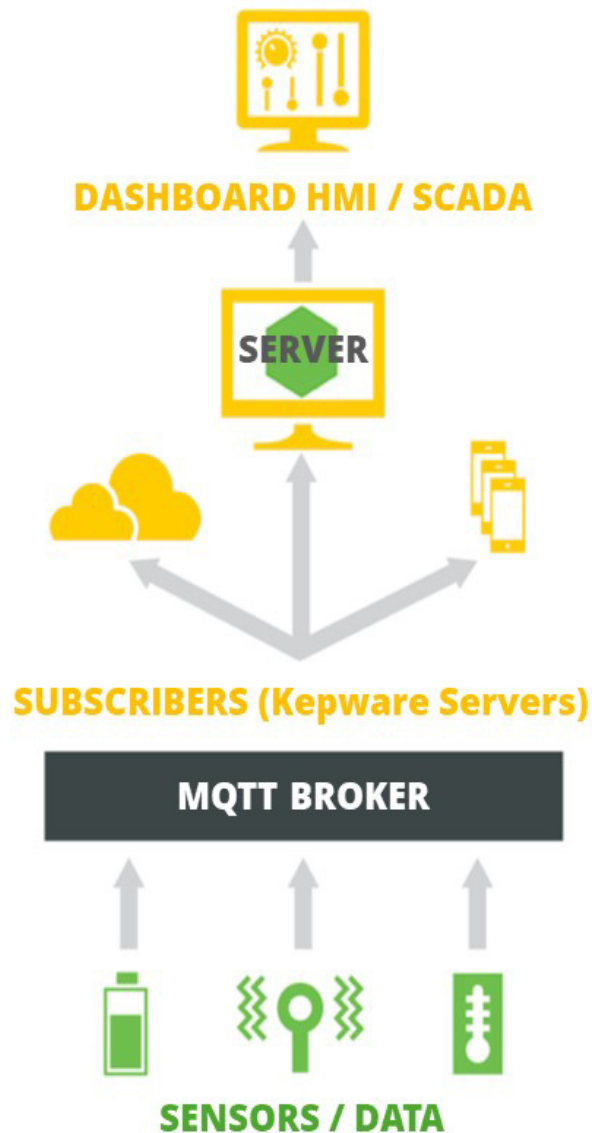
Version 1.037

© 2025 PTC Inc. All Rights Reserved.

Overview

The MQTT Client Driver provides a way to get data from devices sent through the MQTT protocol. Many devices and sensors use diverse or proprietary communication protocols. Using MQTT can unify data collection and publication. This driver connects to an MQTT server (broker) and subscribes to specified topics. When updates on those topics are received, the payload is parsed and the values are set on OPC tags. Those tags can then be accessed through OPC clients to build meaningful data models and dashboards. The MQTT Client Driver supports:

- Parsing valid JSON data format
- Reading data from a MQTT server (writes are currently not supported)
- Authentication and TLS/SSL encryption
- Connecting to MQTT servers using MQTT 3.1 and 3.1.1 protocols
- Integration with ThingWorx platform and other HMI, SCADA, IOT, and cloud-based solutions
- User-level access based on the User Manager and Security Policies Plug-In



Setup

Channel and Device Limits

The maximum number of channels supported by this driver is 60. The maximum number of devices supported by this driver is 1024 per channel.

• If the environment is secured with trusted certificates, establish those relationships through the server administration settings.

Channel Properties – General

This server supports the use of multiple simultaneous 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.

Property Groups General Write Optimizations Advanced	<table border="1"> <tr> <td colspan="2">[-] Identification</td> </tr> <tr> <td>Name</td> <td></td> </tr> <tr> <td>Description</td> <td></td> </tr> <tr> <td>Driver</td> <td></td> </tr> <tr> <td colspan="2">[-] Diagnostics</td> </tr> <tr> <td>Diagnostics Capture</td> <td>Disable</td> </tr> <tr> <td colspan="2">[-] Tag Counts</td> </tr> <tr> <td>Static Tags</td> <td>10</td> </tr> </table>	[-] Identification		Name		Description		Driver		[-] Diagnostics		Diagnostics Capture	Disable	[-] Tag Counts		Static Tags	10
[-] Identification																	
Name																	
Description																	
Driver																	
[-] Diagnostics																	
Diagnostics Capture	Disable																
[-] Tag Counts																	
Static Tags	10																

Identification

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

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

Description: Specify user-defined information about this channel.

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

Driver: Specify the protocol / driver for this channel. Specify the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.

• **Note:** With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. Changes to the properties should not be made once a large client application has been developed. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.

Diagnostics

Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

• **Note:** This property is not available if the driver or operating system does not support diagnostics.

• For more information, refer to *Communication Diagnostics and Statistics Tags* in server help.

Tag Counts

Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

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.

Property Groups	<input type="checkbox"/> Non-Normalized Float Handling	
General	Floating-Point Values	Replace with Zero
Write Optimizations	<input type="checkbox"/> Inter-Device Delay	
Advanced	Inter-Device Delay (ms)	0

Non-Normalized Float Handling: A non-normalized value is defined as Infinity, Not-a-Number (NaN), or as a Denormalized Number. The default is Replace with Zero. Drivers that have native float handling may default to Unmodified. Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. Descriptions of the options are as follows:

- **Replace with Zero:** This option allows a driver to replace non-normalized IEEE-754 floating point values with zero before being transferred to clients.
- **Unmodified:** This option allows a driver to transfer IEEE-754 denormalized, normalized, non-number, and infinity values to clients without any conversion or changes.

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

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

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

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

Channel Properties – MQTT Server

Property Groups	<input type="checkbox"/> Connection	
General	Host	localhost
Write Optimizations	Port	1883
Advanced	SSL/TLS	Disable
MQTT Server	<input type="checkbox"/> Security	
Timing	Client ID	J36TKVtG9QOdHQtl9CaeaU
	Username	
	Password	*****
	Client Certificate	Disable
	<input type="checkbox"/> Optional Configuration	
	MQTT Server Version	Auto
	Subscription QoS	0 (At most once)
	Suppress Parsing Warnings	Disable

Host: The IP address or host name of the MQTT server to connect. The default is localhost.

Port: The port to use to connect to the MQTT server. The valid range is 1 to 65535. The default is 1883.

SSL/TLS: Enable to use a secure connection when connecting to the MQTT server. When enabled, all information is encrypted; this usually requires additional setup.

● **Notes:**

- If enabling SSL/TLS, an MQTT server certificate must be manually uploaded into the MQTT Client Trust Store.

- To manually upload and otherwise configure SSL/TLS certificates, use the Certificate Store tab in the Server Administration tool. Select the MQTT Client feature. More details may be found in the server help document under the **Administration | Settings | Certificate Store** section.
- The server runtime must be restarted when a new certificate is imported.

Client ID: The client identifier defines this MQTT client to the MQTT server. It defaults to a 22-character, randomly generated value. All characters accepted.

Tip: If this value is left blank, the MQTT server assigns a unique value; this depends on the MQTT server.

Caution: This property value must be unique for each MQTT client connecting to a specific MQTT server. Sharing projects without changing the Client ID can lead to connection issues, including disconnections and missing updates.

User Name: Enter a UTF-8 string for the authorized user to connect to the MQTT server. This cannot be blank if the password property has a value.

Password: The password to use when connecting to the MQTT server with the specified user name.

Caution: If SSL/TLS is not enabled, the password sent to the MQTT server can be viewed using a packet sniffing tool.

Client Certificate: Enable to allow client-side certificate validation with the MQTT server.

Notes:

- To configure SSL/TLS certificates, use the **Certificate Store** tab in the Server Administration tool. Select the **MQTT Client** feature. More details may be found in the server help document under the **Administration | Settings | Certificate Store** section.
- The server runtime must be restarted when a new certificate is imported.

MQTT Server Version: Select the MQTT protocol version to use when connecting to the MQTT server. Selecting **Auto** attempts to use version 3.1.1 first, then 3.1.0 if 3.1.1 does not succeed. An event log message shows the version used for connection only if **Auto** is selected. The default is **Auto**.

Subscription QoS: Select the Quality of Service (QoS) to request when subscribing to topics. If the MQTT server doesn't support the selected QoS, an event log message is posted and a lower QoS is used. The default is **0 (At most once)**.

Suppress Parsing Warnings: Select Enable to omit payload JSON parsing messages. The default is Disable. If payloads are not JSON formatted, JSON parsing warnings in the event log can be turned off by changing the property to Enable.

Tip: This property should be set to Disable if payloads are expected to be JSON formatted.

Channel Properties – Timing

Property Groups	☑ Communication Timeouts	
General	Connect Timeout (s)	10
Write Optimizations	Reconnect Minimum (s)	10
Advanced	Reconnect Maximum (s)	10
MQTT Server	Keep Alive (s)	60
Timing		

Connect Timeout (s): Specify the number of seconds the client waits for the MQTT server to confirm the connection. The valid range for the **Connect Timeout** is 1 second to 600 seconds.

Tip: The actual connect timeout could be doubled when connecting to a 3.1.0 MQTT server (broker) with the **MQTT Server Version** set to **Auto** since the initial connection attempts to use 3.1.1.

Reconnect Minimum (s): Specify the minimum amount of time the MQTT Client Driver waits before trying to reconnect to the MQTT server. The valid range for the reconnect minimum is 1 second to 43200 seconds.

Reconnect Maximum (s): Specify the maximum amount of time the MQTT Client Driver waits before trying to reconnect to the MQTT server. This value must be equal to or greater than the reconnect minimum time. The valid range for the reconnect maximum is 1 second to 43200 seconds.

Tip: To always have the same amount of time between retries set the minimum and maximum to the same value.

● **Note:** When a reconnection attempt fails, the current value is doubled. This continues until the maximum is reached. For example, if the minimum is 3 and maximum is 10, the first retry attempt has a three-second delay; the next attempt has a six-second delay; and any attempt after that has a ten-second delay.

Keep Alive (s): Specify the amount of time between PINGREQ requests sent by this client to the MQTT server to confirm it is still active. The valid range for **Keep Alive** is 0 or 10 to 65535 seconds. Setting a value of 0 disables the property.

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.

Property Groups	<input type="checkbox"/> Identification	
General	Name	
	Description	
	Channel Assignment	
	Driver	
	Model	
	ID Format	Decimal
	ID	2

Identification

Name: Specify 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: Specify the user-defined information about this device.

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

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

Driver: Selected protocol driver for this device.

Model: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend 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 documentation.*

ID: Specify the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The format is set by the driver by default. Options include Decimal, Octal, and Hexadecimal.

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

Operating Mode

Property Groups	+ Identification	
General	- Operating Mode	
	Data Collection	Enable
	Simulated	No

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: Place the device into or out of 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 data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

Notes:

1. Updates are not applied until clients disconnect and reconnect.
2. The 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.
3. 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.
4. When a device is simulated, updates may not appear faster than one (1) second in the client.

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

Tag Counts

Property Groups	- Identification	
General	- Operating Mode	
	- Tag Counts	
	Static Tags	130

Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

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.

Property Groups	<input checked="" type="checkbox"/> Scan Mode	
General	Scan Mode	Respect Client-Specified Scan Rate ▼
Scan Mode	Initial Updates from Cache	Disable

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

- **Respect Client-Specified Scan Rate:** This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate:** This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
 - **Note:** When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate:** This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only:** This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the OPC client's responsibility to poll for updates, either by writing to the `_DemandPoll` tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help.*
- **Respect Tag-Specified Scan Rate:** This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

Device Properties – Tag Generation

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

● **Note:** Automatic tag database generation's mode of operation is completely configurable.

Property Groups	<input checked="" type="checkbox"/> Tag Generation	
General	Topic	#
Scan Mode	Discovery Duration (s)	60
Tag Generation	On Duplicate Tag	Do Not Overwrite
	Allow Automatically Generated Subgro...	Enable
	Create	Create tags
	Cancel	Cancel tag creation

Topic: Specify the topic to be subscribed to during automatic tag generation. This must be a valid MQTT topic and can have wild cards. The Topic field cannot be empty. By default it is '#'. The topic name is case sensitive.

● Be careful using '#' because all topics published from that MQTT server will generate tags.

● **Tip:** Using wild cards as the first character in a topic does not generate topics that start with '\$'. This standard MQTT behavior is described in the specification. To get data from '\$SYS', set the topic to '\$SYS/#'.

Discovery Duration: Specify the amount of time, in seconds, the automatic tag generation process runs once it starts. Any publishes to the topic during this time is considered for tag generation. The valid range is 10 to 3600 seconds. The default is 60.

On Duplicate Tag: When automatic tag database generation is enabled, the server needs to recognize what to do with the tags that it may have previously added or tags that have been added or modified after 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. The options are:

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

◆ **Notes:**

1. Removing OPC tags affects tags automatically generated by the communications driver as well as any tags added using names that match generated tags. Avoid adding tags to the server using names that may match tags automatically generated by the driver.
2. MQTT Topics are case sensitive, but OPC server tag groups and tag names are case insensitive. It is possible for tags to be overwritten if the same topic is published with different case.

Allow Automatically Generated Subgroups: Indicate whether the server should create 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.

◆ **Notes:** If a tag is assigned the same name as an existing tag as the server is generating tags, 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** forces the driver to reevaluate the device for possible tag changes.

◆ **Tip:** Accessible from the system tags to allow a client application to initiate tag database creation.

◆ **Notes:**

1. Create tags is disabled if a project is edited offline.
2. While creation can be initiated on multiple devices under a channel at the same time, the driver only processes one create request at a time. When one completes, the next begins.

Cancel: Terminates the automatic tag generation in progress. No tags will be created even if publishes were received.

◆ **Note:** **Cancel** is disabled if there is no tag generation in progress.

◆ **Tip:** To cancel the automatic tag generation in progress through the Config API, set a value of true to the "mqtt_client.DEVICE_CANCEL_TAG_GENERATION" property on the device.

Data Types Supported through Automatic Tag Generation

When tags are created through automatic tag generation (ATG), the process must assign a data type to each tag. That process uses a combination of the JSON values received and certain rules to determine which data type would be the most appropriate.

A tag created through ATG is assigned one of the following types:

- Boolean
- Long (32-bit signed integer)
- LLong (64-bit signed integer)
- Double (64-bit floating point)
- String

The rules are:

- A value of true or false is a Boolean.
- A value that is a string is a String.
- A number that contains a decimal point is a Double.
- A number from -2,147,483,648 to 2,147,483,647 is a Long.
- A number from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 is a LLong.
- A number outside of the LLong range is a String.
- A value of null is a String.

During the tag generation process, the parser continues to adjust the data type of the key value pair as necessary to accommodate the values being received. The more data points received for a tag, the more accurate the data type. For example, if a publish comes through with a value of 67 it is considered a Long. If another publish comes through during ATG with a value of 67.3, that tag is considered a Double. If a third publish comes through with a value of 68, the tag is still considered a Double.

 **Tip:** Once the tag generation process is complete, the user can change the data type of any tag.

Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit
Char	Signed 8-bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
QWord	Unsigned 64-bit value bit 0 is the low bit bit 63 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
LongLong	Signed 64-bit value bit 0 is the low bit bit 62 is the high bit bit 63 is the sign bit
Float	32-bit floating point value The driver interprets two consecutive 16-bit registers as a floating point value by making the second register the high word and the first register the low word.
Double	64-bit floating point value
String	Null-terminated character array

Timestamp Behavior

The Timestamp assigned to a tag is either the time the data was received from the MQTT server or the time that the tag was read. Below describes when each of those occurs:

Timestamp when data is received from MQTT server:

- When the payload is received from the MQTT server, each data point parsed from the payload is assigned the current system time as a timestamp. When a tag referencing this data point is read from a client, it is assigned this timestamp.
- If the same payload is published multiple times, the timestamp assigned to these tags may not change because the value and quality did not change (depends on client).

Timestamp when the tag is read:

- On startup, before connected to the MQTT server and subscribed to tag topic; tags should be "Bad (Out of Service)" quality.
- Connected to the MQTT server and subscribed to tag topic, but have not yet received a publish; tags should be "Uncertain" quality.
- On connection loss, tags should be "Bad" quality.

• **See Also:** [Quality Behavior](#)

Quality Behavior

The quality of a tag reflects the success or failure of the subscribe request for that tag. Different scenarios for the quality of the tags are:

Bad (Out of Service)

- Until connection is resolved.
- If connection succeeded and subscribe is in progress.

Uncertain

- If connection succeeded and subscribe succeeded, but no publishes have been received during the current subscription.

Good

- If connection succeeded, subscribe succeeded, and at least one publish has been received during the current subscription.

Bad

- If connection succeeded and subscribe failed.
- If connection failed.

Tag Address Descriptions

The Tag Address is made up of the topic to subscribe to and the path of the value in the JSON payload. The two parts are separated by a '+' sign. For example, if the topic "Device/Home/LivingRoom" publishes payloads like this {"temp":72.3, "Light1":"On", "Light2":"Off", "TV-On": true}, the tag addresses might look like this:

```
Device/Home/LivingRoom+temp
Device/Home/LivingRoom+Light1
Device/Home/LivingRoom+Light2
Device/Home/LivingRoom+TV-On
```

Notes:

1. MQTT topics are case sensitive.
2. Even though '+' is a wild card in MQTT, wild cards are not supported in the topic of the address. The '+' in the tag address is used as the separator between the topic and the path of the value in the payload.
3. The 'topic#Payload' tag always supplies the last text-based payload received from that topic even if it is not valid JSON. This is ideal for trouble shooting issues or doing client side parsing. For example, to see the entire payload for the example above define a tag whose address is: Device/Home/LivingRoom#Payload. Payloads that are binary or Big-Endian formatted are not resolved to a readable string value in the #Payload tag.
4. Only UTF-8 and UTF-16 (Little Endian) formatted JSON payloads are supported.

Tag Addressing Examples

While working with MQTT one of the more difficult things is to define the mapping between content within a payload and OPC tags. This document is to help with that process with the MQTT Client Driver.

Tag Address

Tag address are of the form mqtt_topic+payload_item. The mqtt_topic is the topic in which the desired data value is to be published. The payload_item is a path, in the payload, to the specific item for which data is desired. Below are some examples:

Single-Level JSON

Topic: vendor/device/data

Sample Payload:

```
{
  "s":4,
  "t":"2017-09-29T19:52:19Z",
  "q":192,
  "c":6,
  "tempint":67.1,
  "vbatt":3.28,
  "ai1":8.92,
  "ai2":0.03,
  "temp1":46.4,
  "temp2":68.0
}
```

Tag addresses and their values would be:

```
vendor/device/data+s = 4
vendor/device/data+t = 2017-09-29T19:52:19Z
vendor/device/data+q = 192
vendor/device/data+c = 6
vendor/device/data+tempint = 67.1
vendor/device/data+vbatt = 3.28
vendor/device/data+ai1 = 8.92
vendor/device/data+ai2 = 0.03
vendor/device/data+temp1 = 46.4
vendor/device/data+temp2 = 68.0
```

Multiple-Level JSON

Topic: vendor/device

Sample Payload :

```
{
  "ModuleUnoccupied": {
    "EquipId": "E12",
    "CarrierId": "C12",
    "SubstrateLocId": "S12",
    "LotId": "L12",
    "DesignId": "D12",
    "EventTime": "12322131"
  }
}
```

Tag addresses and their values would be:

```
vendor/device+ModuleUnoccupied/EquipId = E12
vendor/device+ModuleUnoccupied/CarrierId = C12
vendor/device+ModuleUnoccupied/SubstrateLocId = S12
vendor/device+ModuleUnoccupied/LotId = L12
vendor/device+ModuleUnoccupied/DesignId = D12
vendor/device+ModuleUnoccupied/EventTime = 12322131
```

Single-Level JSON Array

Topic: vendor/device

Sample Payload:

```
{
  "FormatId": "DeviceState",
  "ApiVersion": 1,
  "CurrentTime": "2012-06-11T14:26:59.690+02:00",
  "UserSwitch": "State:Run",
  "Leds": [{
    "Name": "IO",
    "State": "Blinking",
    "Color": "Red"
  },
  {
    "Name": "SYS",
    "State": "On",
    "Color": "Green"
  },
  {
    "Name": "USR",
    "State": "On",
    "Color": "Off"
  }
]
```

Tag addresses and their values would be:

```
vendor/device+FormatId = DeviceState
vendor/device+ApiVerion = 1
vendor/device+CurrentTime = 2012-06-11T14:26:59.690+02:00
vendor/device+UserSwitch = State:Run
vendor/device+Leds[0]/Name = IO
vendor/device+Leds[0]/State = Blinking
vendor/device+Leds[0]/Color = Red
vendor/device+Leds[1]/Name = SYS
vendor/device+Leds[1]/State = On
vendor/device+Leds[1]/Color = Green
vendor/device+Leds[2]/Name = USR
vendor/device+Leds[2]/State = On
vendor/device+Leds[2]/Color = Off
```

Multiple-Level JSON Array

Topic: Sample/NestedArrays

Sample Payload:

```
{
```


```
"name": "John",
"age": 30,
"cars": [{
  "name": "Ford",
  "models": ["Fiesta",
    "Focus",
    "Mustang"]
},
{
  "name": "BMW",
  "models": ["320",
    "X3",
    "X5"]
},
{
  "name": "Fiat",
  "models": ["500",
    "Panda"]
}]
}
```

Tag addresses and their values would be:

```
Sample/NestedArrays+name = John
Sample/NestedArrays+age = 30
Sample/NestedArrays+cars[0]/name = Ford
Sample/NestedArrays+cars[0]/models[0] = Fiesta
Sample/NestedArrays+cars[0]/models[1] = Focus
Sample/NestedArrays+cars[0]/models[2] = Mustang
Sample/NestedArrays+cars[1]/name = BMW
Sample/NestedArrays+cars[1]/models[0] = 320
Sample/NestedArrays+cars[1]/models[1] = X3
Sample/NestedArrays+cars[1]/models[2] = X5
Sample/NestedArrays+cars[2]/name = Fiat
Sample/NestedArrays+cars[2]/models[0] = 500
Sample/NestedArrays+cars[2]/models[1] = Panda
```

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the OPC server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

 **Tip:** Messages that originate from a data source (such as third-party software, including databases) are presented through the Event Log. Troubleshooting steps should include researching those messages online and in vendor documentation.

Datatype conversion failed. | Device = '<device>', Address = '<address>', Expected = '<datatype>', Actual = '<datatype>'.

Error Type:

Error

Possible Cause:

The datatype on the tag does not match the type of data from the payload. This can happen for overflow conditions or if the conversion is invalid.

Possible Solution:

Correct the datatype of the tag to match the data coming from the payload.

Unexpected character found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>'.

Error Type:

Warning

Possible Cause:

The payload is not valid JSON. Expected a token, got something else.

Possible Solution:

Reconfigure the payload to be valid JSON.

Invalid Unicode character found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>'.

Error Type:

Warning

Possible Cause:

Found an incomplete or invalid Unicode character escape sequence.

Possible Solution:

Reconfigure the payload to be valid JSON.

Missing matching quote while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>'.

Error Type:

Warning

Possible Cause:

Found the End of doc before the end of a string

Possible Solution:

Reconfigure the payload to be valid JSON.

Invalid value found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.

Error Type:

Warning

Possible Cause:

Found an invalid value.

Possible Solution:

Reconfigure the payload to be valid JSON.

Invalid number found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.

Error Type:

Warning

Possible Cause:

A numerical value in the payload is not correct. It may contain invalid characters.

Possible Solution:

Reconfigure the payload to be valid JSON.

Unexpected token found within an array while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.

Error Type:

Warning

Possible Cause:

An array within the payload is not formatted correctly.

Possible Solution:

Reconfigure the payload to be valid JSON.

Unexpected token found within an object while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.

Error Type:

Warning

Possible Cause:

An object within the payload is not formatted correctly.

Possible Solution:

Reconfigure the payload to be valid JSON.

Unexpected token found while looking for the end of the MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>.

Error Type:

Warning

Possible Cause:

Unexpected token while looking for the end of the document.

Possible Solution:

Reconfigure the payload to be valid JSON.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>'.

Error Type:

Warning

Possible Cause:

1. The driver was unable to connect to the MQTT server over TCP/TLS due to an unspecified reason, but will continue to attempt connection.
2. The channel is configured to use a DNS host name for the MQTT server rather than an IP address. The host name cannot be resolved by the server to an IP address.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Correct any connectivity issues with the MQTT server.
3. Verify that the correct MQTT port was specified.
4. Verify that the MQTT server IP is within the subnet of the IP to which the server is bound.
5. Verify that the MQTT server is registered with the domain.
6. If using SSL/TLS, verify that the server certificate is uploaded into the driver's MQTT Trust Store.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = '<non-localized reason>'.

Error Type:

Warning

Possible Cause:

1. The driver was unable to connect to the MQTT server over TCP/TLS due to the reason provided, but will continue to attempt connection.
2. The channel is configured to use a DNS host name for the MQTT server rather than an IP address. The host name cannot be resolved by the server to an IP address.
3. The MQTT server is configured to use TLS 1.0 or 1.1. The MQTT Client Driver supports only TLS 1.2.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Correct any connectivity issues with the MQTT server.
3. Verify that the correct MQTT port was specified.
4. Verify that the MQTT server IP is within the subnet of the IP to which the server is bound.
5. Verify that the MQTT server is registered with the domain.
6. If using SSL/TLS, verify that the server certificate is uploaded into the driver's MQTT Trust Store.
7. Configure the MQTT server to use TLS 1.2.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Error Code = '<code>'.

Error Type:

Warning

Possible Cause:

1. The driver was unable to connect to the MQTT server over TCP/TLS due to the error code provided, but will continue to attempt connection.
2. The channel is configured to use a DNS host name for the MQTT server rather than an IP address. The host name cannot be resolved by the server to an IP address.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Verify that the correct MQTT port was specified.
3. Verify that the MQTT server IP is within the subnet of the IP to which the server is bound.
4. Verify that the MQTT server is registered with the domain.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = '<reason>', Error Code = '<code>'.

Error Type:

Warning

Possible Cause:

1. The driver was unable to connect to the MQTT server over TCP/TLS due to the reason and error code provided, but will continue to attempt connection.
2. The channel is configured to use a DNS host name for the MQTT server rather than an IP address. The host name cannot be resolved by the server to an IP address.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Verify that the correct MQTT port was specified.
3. Verify that the MQTT server IP is within the subnet of the IP to which the server is bound.
4. Verify that the MQTT server is registered with the domain.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = 'Connection refused; MQTT Version not supported'.

Error Type:

Warning

Possible Cause:

The driver was able to connect to the MQTT server over TCP/TLS but the server refused our MQTT CONNECT request.

Possible Solution:

Verify the MQTT version specified matches the MQTT server's supported versions.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = 'Connection refused; Client ID rejected'.

Error Type:

Warning

Possible Cause:

The driver was able to connect to the MQTT server over TCP/TLS but the server refused our MQTT CONNECT request.

Possible Solution:

Verify a valid Client ID has been specified. Empty Client IDs or Client IDs longer than 23 characters are not supported in MQTT version 3.1.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = 'Connection refused; MQTT server unavailable'.

Error Type:

Warning

Possible Cause:

The driver was able to connect to the MQTT server over TCP/TLS but the server refused our MQTT CONNECT request.

Possible Solution:

Verify that the MQTT server is started and online.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = 'Connection refused; Bad Username or Password'.

Error Type:

Warning

Possible Cause:

The driver was able to connect to the MQTT server over TCP/TLS but the server refused our MQTT CONNECT request.

Possible Solution:

Verify a valid username and password, according to the MQTT server, have been specified.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = 'Connection refused; Not authorized to connect'.

Error Type:

Warning

Possible Cause:

The driver was able to connect to the MQTT server over TCP/TLS but the server refused our MQTT CONNECT request.

Possible Solution:

Verify a valid username and password, according to the MQTT server, have been specified.

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<host-name:port>', Reason = 'Connection refused', Error Code = '<code>'.

Error Type:

Warning

Possible Cause:

The driver was able to connect to the MQTT server over TCP/TLS but the server refused our MQTT CONNECT request.

Possible Solution:

Error code is undocumented. Please refer to latest MQTT specification or MQTT server for error code description.

Connection with MQTT server lost. | Channel = '<channel>', Server = '<host-name:port>'.

Error Type:

Warning

Possible Cause:

The driver lost communications with MQTT server, but will attempt to reconnect at the reconnect intervals configured.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Correct any connectivity issues with the MQTT server.

Unable to subscribe to topic. | Channel = '<channel>', Topic = '<topic>'.

Error Type:

Warning

Possible Cause:

The driver lost communications with MQTT server during a subscribe request, but will attempt to reconnect at the reconnect intervals configured.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Correct any connectivity issues with the MQTT server.

Unable to unsubscribe from topic. | Channel = '<channel>', Topic = '<topic>'.

Error Type:

Warning

Possible Cause:

The driver lost communications with MQTT server during an unsubscribe request, but will attempt to reconnect at the reconnect intervals configured.

Possible Solution:

1. Verify that the MQTT server is started and online.
2. Correct any connectivity issues with the MQTT server.

MQTT server is not able to support the requested QoS. Subscribed with a lower QoS. | Channel = '<channel>', Topic = '<topic>', QoS Requested = <QoS>, QoS Subscribed With = <QoS>'.

Error Type:

Warning

Possible Cause:

The MQTT server may not be configured to support the requested Quality of Service.

Possible Solution:

Verify the MQTT server configuration.

Operation cancelled by user.

Error Type:

Warning

Possible Cause:

Automatic tag generation cancelled by the user.

Possible Solution:

Retry automatic tag generation and allow to run to completion.

Operation cancelled due to a system event.

Error Type:

Warning

Possible Cause:

1. The server runtime was stopped or reinitialized.
2. The channel was deleted during automatic tag generation.
3. The demo timer expired during automatic tag generation.

Possible Solution:

Correct and restart automatic tag generation.

Operation cancelled due to a property change.

Error Type:

Warning

Possible Cause:

A property was changed that required a new connection to the MQTT server.

Possible Solution:

Make any desired changes to connection properties, then restart automatic tag generation.

No publishes received from MQTT server on specified topic.

Error Type:

Warning

Possible Cause:

1. There may have been connection issues to the MQTT server.
2. There may not have been any publishes to the topic used for automatic tag generation.

Possible Solution:

1. Review other event log messages to see if there was a connection issue. Fix and retry automatic tag generation.

2. Check the Topic property to verify the topic is receiving at least one data publish. If needed, use an MQTT client to verify that publishes are occurring to the topic selected for automatic tag generation.

MQTT version automatically selected by server. | Channel = '<channel>', Server = '<hostname:port>', MQTT Version = <version>.

Error Type:

Informational

Note:

This message will only be displayed if the property 'MQTT Server Version' on the channel is set to 'Auto'.

Unable to generate a tag database for '<channel>'. '<device>': <Reason>.

If there is a problem creating a tag database, the message that appears in the event log is a combination of the server error and a reason provided by this driver.

The server message "**Unable to generate a tag database for channel.device.**" is combined with one of the following reasons from the driver:

- [Operation cancelled by user.](#)
- [Operation cancelled due to a system event.](#)
- [Operation cancelled due to a property change.](#)
- [No publishes received from MQTT server on specified topic.](#)

Index

A

Authentication 5

B

Bad 15

Bad Username or Password'. 23

Boolean 14

Broker 5

Byte 14

C

Channel Assignment 9

Channel Properties – Advanced 7

Channel Properties – General 6

Char 14

Client Certificate 8

Client ID 8

Client ID rejected'. 23

Connect Timeout 8

Connection with MQTT server lost. | Channel = '<channel>', Server = '<hostname
port>'. 24

Create 12

D

Data Collection 10

Data Types Description 14

Datatype conversion failed. | Device = '<device>', Address = '<address>', Expected = '<datatype>', Actual =
'<datatype>'. 19

Device Properties – General 9

Diagnostics 6

Discovery Duration 11

Do Not Scan, Demand Poll Only 11

Double 14

Driver 9

Duplicate 12

DWord 14

E

Event Log Messages 19

F

Failed to connect to MQTT server. | Channel = '<channel>', Server = '<hostname

port>', Error Code = '<code>'. 22

port>', Reason = '<non-localized reason>'. 21

port>', Reason = '<reason>', Error Code = '<code>'. 22

port>', Reason = 'Connection refused 22-23

port>', Reason = 'Connection refused', Error Code = '<code>'. 23

port>'. 21

Float 14

G

General 9

Good 15

H

Host 7

I

ID 9

Identification 6, 9

Initial Updates from Cache 11

Inter-Device Delay 7

Invalid number found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 20

Invalid Unicode character found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 19

Invalid value found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 20

J

JSON 5, 16

L

Long 14

LongLong 14

M

Missing matching quote while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 19

Model 9

MQTT 5

MQTT Server 5, 7

MQTT server is not able to support the requested QoS. Subscribed with a lower QoS. | Channel = '<channel>', Topic = '<topic>', QoS Requested = <QoS>, QoS Subscribed With = <QoS>. 24

MQTT server unavailable'. 23

MQTT Server Version 8

MQTT version automatically selected by server. | Channel = '<channel>', Server = '<hostname port>', MQTT Version = <version>. 26

MQTT Version not supported'. 22

N

Name 9

NestedArrays 17

No publishes received from MQTT server on specified topic. 25

Non-Normalized Float Handling 7

Not authorized to connect'. 23

O

Operating Mode 10

Operation cancelled by user. 25

Operation cancelled due to a property change. 25

Operation cancelled due to a system event. 25

Out of Service 15

Overview 5

Overwrite 12

P

Password 8

Payload 16

Port 7

Q

Quality 15

QWord 14

R

Reconnect Maximum 8

Reconnect Minimum 8

Replace with Zero 7

Respect Tag-Specified Scan Rate 11

S

Scan Mode 11

Setup 6

Short 14

Simulated 10

SSL/TLS 7

String 14

Subgroups 12

Subscription QoS 8

Suppress Parsing Warnings 8

T

Tag Address Descriptions 16

Tag Addressing Examples 16

Tag Counts 6, 10

Tag Generation 11

Timestamp Behavior 15

Timing 8

Topic 11

U

Unable to generate a tag database for channel.device. 26

Unable to subscribe to topic. | Channel = '<channel>', Topic = '<topic>'. 24

Unable to unsubscribe from topic. | Channel = '<channel>', Topic = '<topic>'. 24

Uncertain 15

Unexpected character found while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 19

Unexpected token found while looking for the end of the MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 20

Unexpected token found within an array while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 20

Unexpected token found within an object while parsing MQTT payload. | Channel = '<channel>', Topic = '<topic>', Line = <line>, Col = <col>. 20

Unmodified 7

User Name 8

W

Wild card 16

Word 14