



AVEVA™ Communication Drivers Pack – Standards - MQTT Driver

User Guide

© 2015-2023 AVEVA Group Limited or its subsidiaries. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of AVEVA Group Limited. No liability is assumed with respect to the use of the information contained herein.

Although precaution has been taken in the preparation of this documentation, AVEVA assumes no responsibility for errors or omissions. The information in this documentation is subject to change without notice and does not represent a commitment on the part of AVEVA. The software described in this documentation is furnished under a license agreement. This software may be used or copied only in accordance with the terms of such license agreement. AVEVA, the AVEVA logo and logotype, OSIsoft, the OSIsoft logo and logotype, Archedra, Avantis, Citect, DYNsIM, eDNA, EYESIM, InBatch, InduSoft, InStep, IntelaTrac, InTouch, Managed PI, OASyS, OSIsoft Advanced Services, OSIsoft Cloud Services, OSIsoft Connected Services, OSIsoft EDS, PIPEPHASE, PI ACE, PI Advanced Computing Engine, PI AF SDK, PI API, PI Asset Framework, PI Audit Viewer, PI Builder, PI Cloud Connect, PI Connectors, PI Data Archive, PI DataLink, PI DataLink Server, PI Developers Club, PI Integrator for Business Analytics, PI Interfaces, PI JDBC Driver, PI Manual Logger, PI Notifications, PI ODBC Driver, PI OLEDB Enterprise, PI OLEDB Provider, PI OPC DA Server, PI OPC HDA Server, PI ProcessBook, PI SDK, PI Server, PI Square, PI System, PI System Access, PI Vision, PI Visualization Suite, PI Web API, PI WebParts, PI Web Services, PRISM, PRO/II, PROVISION, ROMeo, RLINK, RtReports, SIM4ME, SimCentral, SimSci, Skelta, SmartGlance, Spiral Software, WindowMaker, WindowViewer, and Wonderware are trademarks of AVEVA and/or its subsidiaries. All other brands may be trademarks of their respective owners.

U.S. GOVERNMENT RIGHTS

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the license agreement with AVEVA Group Limited or its subsidiaries and as provided in DFARS 227.7202, DFARS 252.227-7013, FAR 12-212, FAR 52.227-19, or their successors, as applicable.

AVEVA Third Party Software Notices and Licenses: <https://www.aveva.com/en/legal/third-party-software-license/>

Publication date: Wednesday, November 8, 2023

Publication ID: 868880

Contact information

AVEVA Group Limited
High Cross
Madingley Road
Cambridge
CB3 0HB. UK

<https://sw.aveva.com/>

For information on how to contact sales and customer training, see <https://sw.aveva.com/contact>.

For information on how to contact technical support, see <https://sw.aveva.com/support>.

To access the AVEVA Knowledge and Support center, visit <https://softwaresupport.aveva.com>.

Contents

Chapter 1 Introduction to the MQTT Communication Driver.	5
About the MQTT Communication Driver.	5
Supported Client Protocols.	5
Licensing for MQTT Communication Driver.	6
Chapter 2 Configuring the MQTT Communication Driver.	7
MQTT Subscriber.	7
Adding and Configuring an MQTT Broker Connection.	7
Browsing Tag Items.	10
Exporting and Importing Tag References.	14
Configuring a Group Connection as an MQTT Subscriber.	15
Configuring MQTT Device Items.	16
MQTT Publisher.	18
Adding and Configuring the Dedicated MQTT Publisher.	19
Adding and Configuring the MQTT Publisher Under the MQTT Broker.	26
Importing and Exporting the MQTT Publisher Configuration.	31
Chapter 3 Diagnostic System Items.	39
Standard System Items.	39
Global System Items.	40
Device-Specific System Items.	40
Device Group-Specific System Items.	41
MQTT Communication Driver-Specific Diagnostics.	42
Global Diagnostic Items.	43
Subscriber Diagnostic Items.	47
Publisher Diagnostic Items.	49
Chapter 4 MQTT Item Naming.	51
Publisher.	51
Subscriber.	53
Chapter 5 MQTT Communication Driver Reference.	57
Runtime Poking Behavior of MQTT Communication Driver.	57

MQTT Topic Names	57
Using JSON Strings	58

Chapter 1

Introduction to the MQTT Communication Driver

- [About the MQTT Communication Driver](#)
- [Supported Client Protocols](#)
- [Licensing for MQTT Communication Driver](#)

About the MQTT Communication Driver

Message Queuing Telemetry Transport (MQTT) is a publish/subscribe messaging protocol for use over TCP/IP. MQTT is designed to ensure that devices can communicate with each other while minimizing power and bandwidth requirements. It is a simple messaging protocol that is well-suited for use with devices that rely on slow or unreliable networks.

The MQTT protocol is an application layer specification and has been published as standard ISO/IEC 20922. MQTT uses a Publish-Subscribe mechanism which requires a mediating broker. In this release, as a subscriber, the MQTT Communication Driver will subscribe MQTT payload of Sparkplug, JSON, and/or plain text format. As a publisher, the MQTT Communication Driver will publish data in Sparkplug and JSON format.

Supported Client Protocols

MQTT Communication Driver leverages the MQTT protocol to publish and/or subscriber messages that are in JSON and/or Sparkplug format. Chapter 3 [MQTT Communication Driver Reference](#) describes the format of JSON payload that is supported by the MQTT Communication Driver.

MQTT is an OASIS standard. The specification is available online at the following link:

<https://mqtt.org/mqtt-specification/>

Sparkplug is another specification managed by the Eclipse Foundation. Sparkplug leverages the communication capabilities of MQTT to optimize device namespace and life-cycle for the SCADA/IIoT solution sector. The specification is available online at the following link:

<https://projects.eclipse.org/projects/iot.sparkplug>

Licensing for MQTT Communication Driver

The MQTT Communication Driver supports the activation-based licensing to acquire the license both locally and remotely. For more information on activation-based licensing, see [Centralized \(Activation-Based\) Licensing](#) in the Communication Drivers Pack Help.

Chapter 2

Configuring the MQTT Communication Driver

- [MQTT Subscriber](#)
- [MQTT Publisher](#)

Note: Refer to the "Configuring Global Parameters" section in the Communication Drivers Core guide for more information on configuring the global parameters in the MQTT Communication Driver. Note that the default value of the **Buffered Data (Maximum Queued Updates)** parameter is 100 for the MQTT Communication Driver.

MQTT Subscriber

- [Adding and Configuring an MQTT Broker Connection](#)
- [Browsing Tag Items](#)
- [Exporting and Importing Tag References](#)
- [Configuring a Group Connection as an MQTT Subscriber](#)
- [Configuring MQTT Device Items](#)

Adding and Configuring an MQTT Broker Connection

To add an MQTT broker connection

Right-click **Configuration** in the hierarchy and select **Add MQTT_BROKER Connection** from the shortcut menu.

A new connection is created in the hierarchy tree, named "New_MQTT_BROKER_000" by default. Rename it, if desired. Multiple MQTT Broker connections can be added to one IOT - MQTT instance.

To configure the connection to the MQTT Broker

1. In the **Broker Address** field, enter the IP address of the MQTT Broker or host name. The number of characters cannot exceed 255 and this field cannot be blank.

The MQTT Communication Driver also supports IPv6 network connectivity with the MQTT Broker. The MQTT Broker can be configured with the link-local IPv6 address of the machine on which the MQTT Broker runs.

2. The default **Port Number** 1883 provided is the network port parameter used by default by MQTT brokers for unsecure connections. If you enable encrypted connection in **Step 2: (Optional) Encrypt connection with TLS** below, the port number automatically switches to 8883, which is the default network port parameter for

secure connections to MQTT brokers. You should only edit this port number if the MQTT broker uses a non-standard port.

3. The **Subscriber Client ID** is used to uniquely identify the subscriber connection to the broker. The specified string must be unique across all client IDs connected to the same broker. If it is blank, the MQTT Communication Driver will automatically generate a unique string.
4. The **Persist Session** ensures that the MQTT broker saves all the information that is relevant for the client on the broker. When persistence is enabled, the broker will store any QoS1 or QoS2 messages that have not been retrieved. In the traditional MQTT standard, the setting is referred to as the “Clean” Session. The MQTT Sparkplug standard refers to it as “Persist Session”. You enable Persistence when you want the broker to store relevant information.
 - a. Select the **Persist Session** check box if you are a subscriber client in the MQTT network and need the broker to store QoS1 and QoS2 messages that have not been retrieved.
 - b. Clear the **Persist Session** check box if you are a subscriber client in the MQTT network and do not need to get messages that you missed while you were offline.

IMPORTANT: Persistence only applies to messages that have been sent with Quality of Service 1 or 2.

5. If you select the **Enable** checkbox inside the **Subscribe to Sparkplug Infrastructure** section, the subscribed data can originate from both Sparkplug and non-Sparkplug infrastructure, and the Sparkplug STATE message will be subscribed. If you do not select the **Enable** checkbox, the subscribed data must be in JSON and/or plain text format and the Sparkplug STATE message will not be subscribed. When the **Enable** checkbox is selected, the following fields will be enabled.
 - a. When the **Primary Application** check box is checked, the broker hierarchy acts as a Primary Host and sends an Online message to the broker on the 'STATE/<Scada Host Id>' topic on the broker. This topic is used to direct Sparkplug publishers to publish the NBIRTH/DBIRTH messages to the broker.
 - b. The **SCADA Host ID** is the name by which this application is known in the network so that all Edge Gateway (EON) nodes can monitor its connectivity to the broker as described above. Refer to the Sparkplug specification for information on the role of the SCADA Host ID.

IMPORTANT: In an MQTT Sparkplug-based network there should only be a single node designated as the Primary Application.




- a. Specify the **Listener Client ID** to uniquely identify a background service to connect to the MQTT broker. The background service is used to monitor birth and death messages sent/received by the MQTT broker. You can change the Listener Client ID to fit your naming nomenclature, but you must ensure that it is unique across all MQTT connections to the same broker.
6. Click the **Validate Address and Port** button to verify that the MQTT Broker can be accessed. The status of the test is displayed in a dialog. The initial status is "Connecting to host..."
 - If the connection to the MQTT Broker is successful, the final status is "Connection to host successful."
 - If the MQTT Broker cannot be accessed, the final status is "Unable to connect to host." Check that the network address and port number are correct.

To enable a TLS-based secure connection (optional)

A digital certificate is required to establish a secure connection with an MQTT broker. The digital certificate, also called a public key certificate, confirms the identity of the broker and is also used to encrypt communications with the broker. Trusted digital certificates are issued by the official, trusted agencies known as certification

authorities (CA), and guarantee the identity of the broker. In contrast, self-signed digital certificates are issued by private parties and do not guarantee the identity of the broker.

When you validate the security setting of the broker connection, three results are possible:

- A green security icon  is displayed with the text **Connection to the broker is secure and trusted.**
The green security icon indicates that connection to the broker is encrypted and the broker's certificate is issued by a trusted certification authority (CA).
- A yellow security icon  is displayed with the text, **Connection to the broker is secure and untrusted.**
The yellow security icon indicates that connection to the broker is encrypted, but the broker's certificate is self-signed and is therefore untrusted. See *Verifying a Self-Signed Certificate from an Untrusted MQTT Data Source* for additional information.
- A red security icon  is displayed with the text, **Connection to the broker is insecure and untrusted.**
The red security icon indicates that the identity of the broker cannot be checked and the connection is unencrypted. Since the broker's identity cannot be verified, it is considered unknown and untrusted.

Note: Enabling a secure connection is separate from connecting to a broker. Once security has been successfully enabled, it is possible to see a green security icon without being connected to the broker. However, you must be connected to the broker to be able to validate security.

To enable a TLS-based secured connection:

1. To enable security, under **Step 2: (Optional) Encrypt connection with TLS**, select the **Enable** check box.
The **Port Number** in Step1 automatically changes to 8883. Edit the port number if necessary.
2. To set up encryption and privacy to the MQTT Broker, select the highest version of the TLS from the **Select Transport Layer Security (TLS) version**. The options available are **tlsv1**, **tlsv1.1**, **tlsv1.2**, and **tlsv1.3**.

Note: **tlsv1** and **tlsv1.1** have been deprecated and are provided only for compatibility purposes.

3. If using a self-signed certificate, it is recommended to first verify the certificate. To verify the digital certificate:
 - a. Click the **Download** button.
 - b. Verify the self-signed certificate of the broker. Do not connect to a broker if you do not trust its self-signed certificate.
 - c. Click **Validate Security** to confirm that the certificate is trusted when the security icon changes to green.
 - d. If the broker prescribes a self-signed CA certificate, the validation may still fail. In this case, consult broker configuration instruction.
 - i. Manually download a CA certificate from the broker.

Note: This indicates that you are trusting the CA certificate.

- ii. Save the CA certificate into one of the file locations on the machine.
- iii. Specify the location of the saved CA certificate in the **CA File** field.

- iv. Click **Validation Security** to confirm again if the certificate is trusted.
4. Click **Validate Security**, to check that an encrypted connection over TLS can be established with the MQTT Broker.
 - If the security validation is successful, the green, yellow, or red security icon is displayed, along with the corresponding description of the connection.
 - If the security validation is not successful, a dialog will be prompted advising the reason and possible corrective action.

Note: You can reset the value of the CA certificate to the system default by clearing the **CA File** field and clicking on **Validation Security**.

To enter Connection and/or User Identity (optional)

Note: Before configuring this section, you should enable the secure connection to the broker by selecting the **Enable** checkbox under **Step 2: (Optional) Encrypt connection with TLS** section.

1. Select the **Client Authentication** check box if the broker provides a certificate which must be used for connection.
 - a. In the **Client Certificate File** field, browse or specify the location of the applicable client certificate.
 - b. In the **Client Key File** field, browse or specify the location of the respective client key file.
 - c. In the **Client Key Password (optional)** field, enter the password of the client key file if applicable. Clear this field if the Client Key file is not password-protected.
2. Under **User Identity**, select the **Enable** check box to turn on user authentication for subscribing/publishing to MQTT messages.

Note: If you enable this option, it is **highly recommended** that you enable MQTT Connection Security to protect the username and password.

MQTT Communication Driver uses the user name and password settings that you enter here to connect to the configured MQTT broker.

3. Click **Validate Identity**, to verify that the MQTT Broker can be accessed on the configured MQTT channel.

Browsing Tag Items

Note: This feature is only available for MQTT Sparkplug-based subscription model. You can only view the **MQTT Browser** tab from a GR (Application Server) node. You cannot view the **MQTT Browser** tab from a remote node.

The user who launches the OCMC must be part of the oiAdministrators Windows Local Users and Groups to access the MQTT Browser tab. Otherwise, an error message is displayed saying the user or the user group is not part of oiAdministrators group. If the user is not a part of the oiAdministrators group, you must add the user to the oiAdministrators group manually and re-login your computer. The user who installs the Communication Drivers Pack will be added to the oiAdministrators group automatically.

If you make any changes to the configuration of the System Management Server (SMS), you must restart the **AVEVA Communications Backend Service** in the **Services** console of your machine. In the Advanced Configuration of the System Management Server (SMS), if you change the **HTTP Port** or the **HTTPS Port** then you must run the following command as an administrator in the command prompt:

If you change the HTTP Port field

```
netsh http add urlacl url=http://localhost:<http port configured in SMS>/oi/ user="NT Authority\Network Service"
```

If you change the HTTPS Port field

```
netsh http add urlacl url=https://localhost:<https port configured in SMS>/oi/ user="NT Authority\Network Service"
```

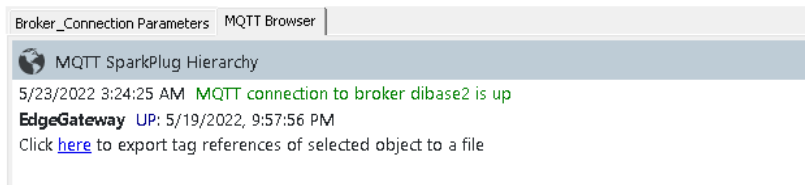
If you do not run the above commands after changing the ports and try to access the MQTT browser functionality, you will get an HTTP 503 error saying the service is unavailable.

The MQTT Browser window displays the item reference, hierarchy, and the corresponding node details. The Sparkplug hierarchy is displayed, once you enter the correct broker IP address and click **Save** in the **Broker Parameters** tab. The MQTT Browser screen is divided into three sections.

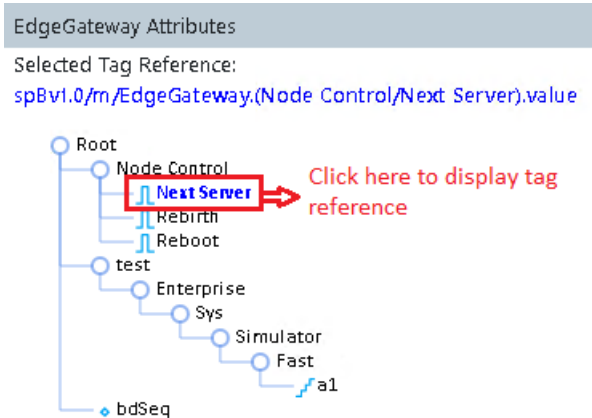
Copying Tag References to the System Clipboard

To facilitate tag reference creations in Sparkplug format, you can use the MQTT Browser to copy selected tag references to the Windows Clipboard.

- Copy all tag references of a Sparkplug Edge Gateway or Device



- Copy a tag reference of an attribute in a Sparkplug Gateway or Device



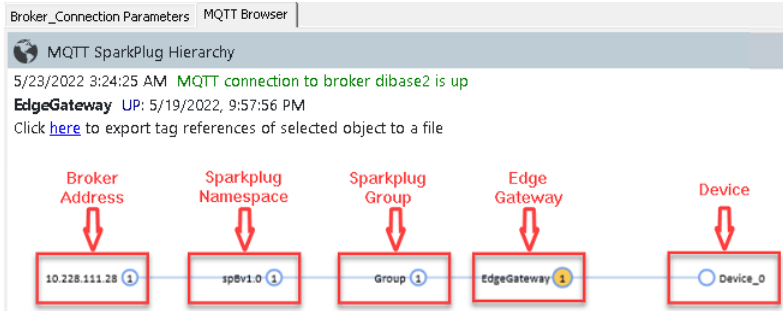
To export the tag references to a CSV file, click anywhere on the line **Click here to export tag references of selected object to a file**. For the procedure on exporting and importing tag references, see [Exporting and Importing Tag References](#). For more information on Device Items, see [Configuring MQTT Device Items](#). You can also drag a required node from the hierarchy and drop it to the Excel file. All the tags below the selected node in the hierarchy are exported to the Excel file.

Hierarchy/GeoLocation Browsing

Hierarchy



Displays the hierarchy from a broker that contains payloads from the AVEVA Edge devices that are Sparkplug encoded. The hierarchy includes:

- Broker address
- Sparkplug Namespace
- Sparkplug Group
- Edge Gateway
- Device Name

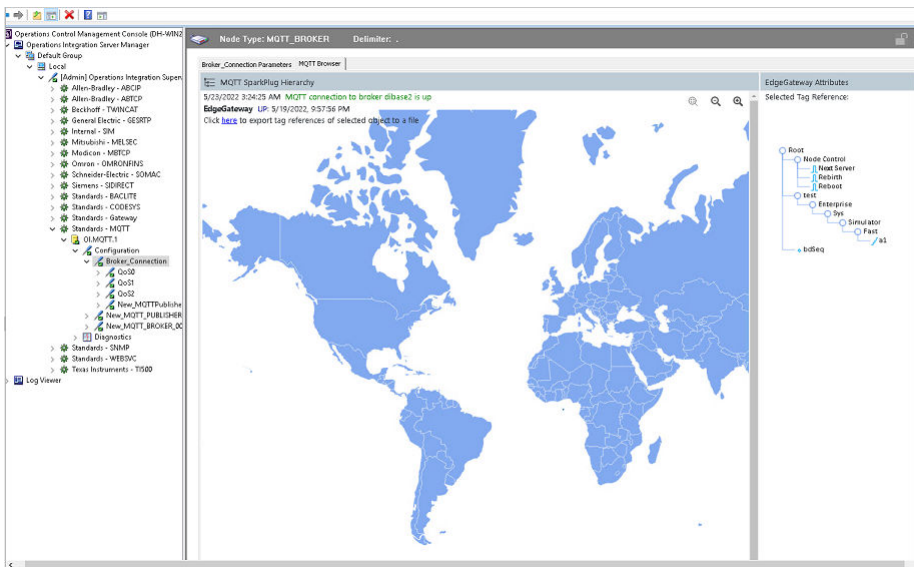


It displays a hierarchical browser that shows the topic hierarchy in the MQTT broker. If a node has children, then you can double-click the node to collapse or expand. Selecting a node displays the details of the node in the right pane. If any of the nodes in the hierarchy is not active or not available, the respective node will be greyed out. If you hover the cursor on a node, you can see the tooltip which briefs about that node.

GeoLocation

Click  to display the GeoLocation, and click  icon to display back the Hierarchy. The GeoLocation option displays the geographical location of an object, such as Edge Gateway and Device. For assets to be displayed in the GeoLocation Browser, the MQTT (sparkplug-based) payload must contain the GeoLocation latitude and longitude in the following format:

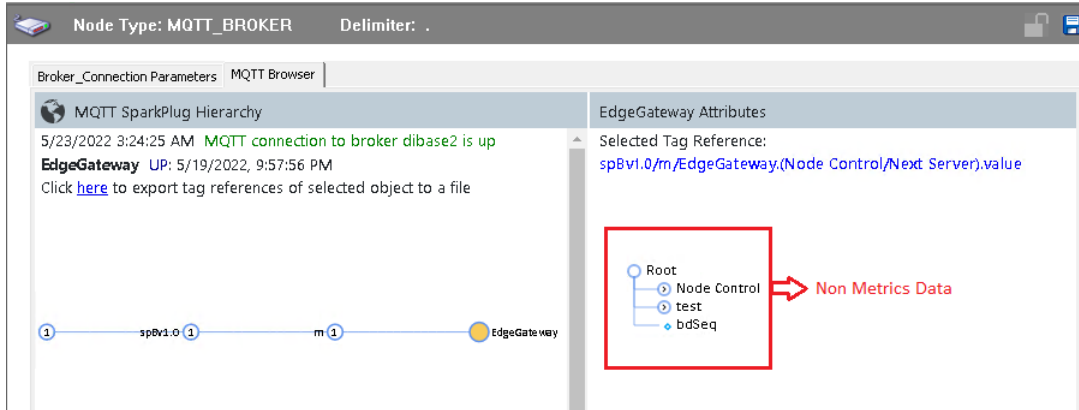
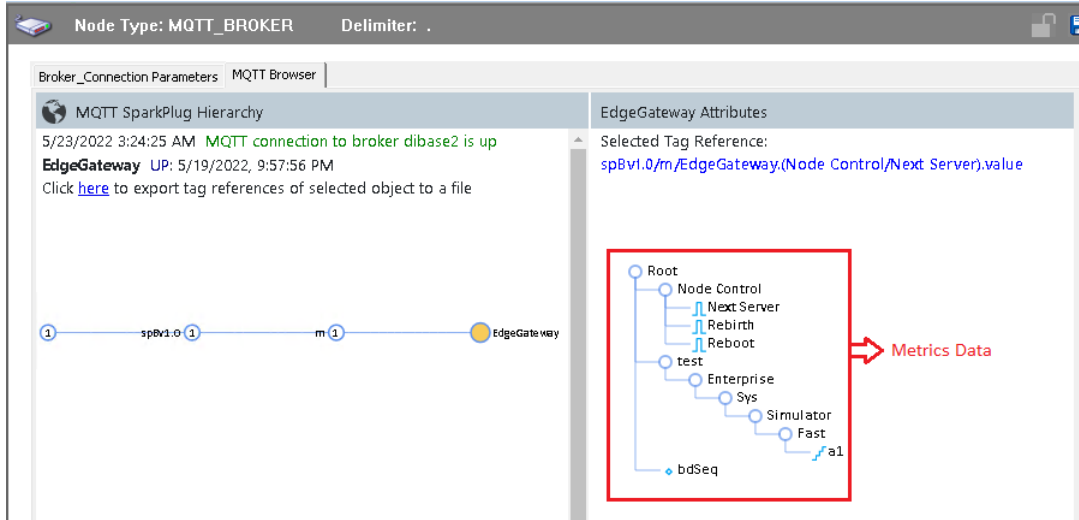
- Position/Latitude
- Position/Longitude



EdgeGateway Attributes

When you click on **EdgeGateway** in the Sparkplug hierarchy on the left pane, the hierarchy of the **Edge Gateway Attributes** is displayed on the right pane. The same hierarchy is displayed in the MQTT publisher as well.

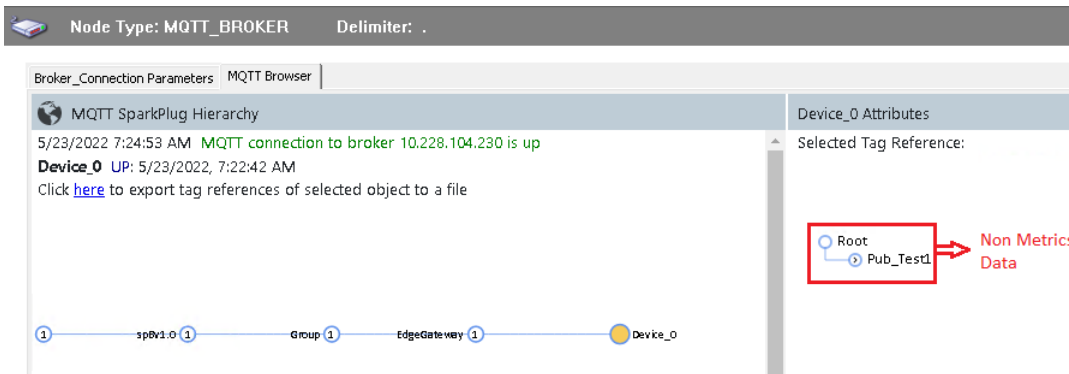
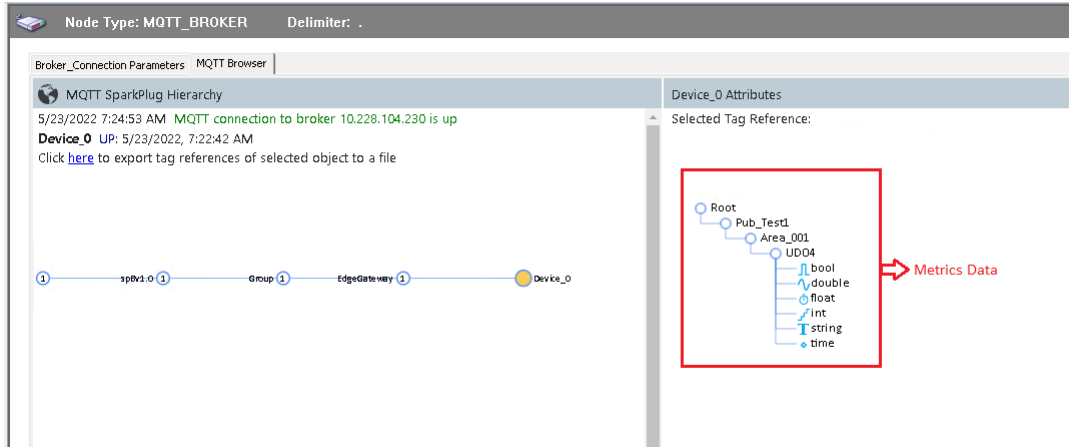
To view the detailed metrics of a node, expand the hierarchy by clicking the arrow button on the left of the node. Click the same button to minimize the metrics.



Device Attributes

When you click on **Device** in the Sparkplug hierarchy on the left pane, the hierarchy of the **Device Attributes** is displayed on the right pane.

To view the detailed metrics of a node, expand the hierarchy by clicking the arrow button on the left of the node. Click the same button to minimize the metrics.



Deleting an Edge Gateway or Device node

- To delete an edge gateway or a device node, right-click the required node, and select **Remove Node**.



This cleans up publishing Edge Gateway node or Device nodes cached in the subscriber that are no longer in existence or used.

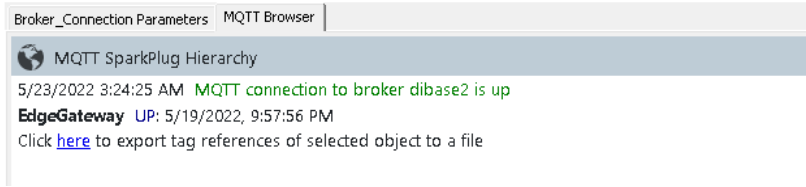
Note: Any deleted node will re-appear in the screen if the publisher starts to publish any deleted nodes.

Exporting and Importing Tag References

The Export option enables you to export tag references of the required device or node to a CSV file. It allows you to copy tag references and edit the tag names in bulk or individually. You can import this CSV file using the **Import** option in the **Device Items** tab of the required **Broker Group**.

To Export tag reference of the selected object to a CSV file:

- In the OCMC, click the required **MQTT Broker** node and then click the **MQTT Browser** tab.
- In the **MQTT Browser** window, click anywhere on the line **Click here to export tag references of selected object to a file**.



3. If required, open the CSV file and make the tag name changes.
4. **Save** and **Close** the CSV file.

To Import the tag references from a CSV file:

1. In the OCMC, click the required **MQTT Group** node and then click the **Device Items** tab.
2. In the **Device Items** window, right-click anywhere in the table, and select **Import**.

The **Open** dialog box appears. It defaults to the .csv file extension within the current-system-configured default directory.

3. Browse and select the specific CSV file you want to import, then click **OK** for confirmation.
4. Once the import is complete, you can see the tags and the respective tag references. The left column is the sample alias name and the right column is the item reference.
5. Click the **Save** icon to save the changes.

Note: When you select another part of the Communication Driver tree hierarchy, you are prompted to save the modifications to the configuration set.

During the imported file processing:

- New item references will be added based on unique names.
- If there are duplicate names, you can replace the existing entry with the new entry or ignore the new entry.

When the MQTT Communication Driver is running and a client requests item information, the imported configured items will show up under the controller hierarchy node.

Other Options to Export the Tag References:

- You can drag a required node from the hierarchy and drop it to the Excel file. All the tags below the selected node in the hierarchy are exported to the Excel file.
- You can also use the system copy buffer to export the tag references. Right-click on the required hierarchy to copy the tag references under the selected hierarchy. You can then go to the IO Mapping table in the IDE and paste the copied tag references by pressing Ctrl+V.

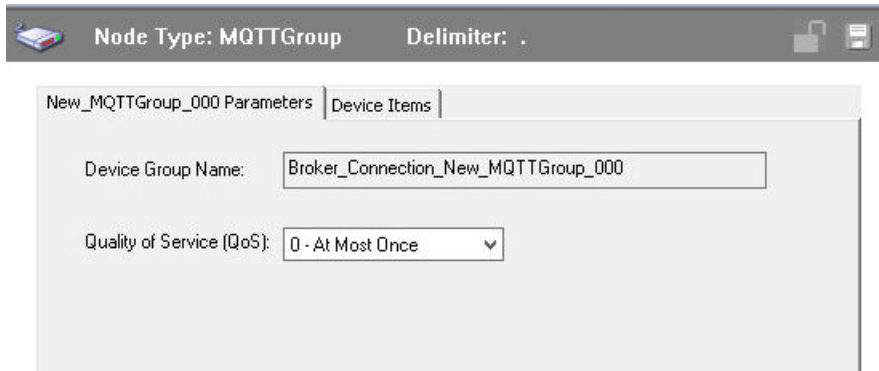
Configuring a Group Connection as an MQTT Subscriber

To add a group connection to your MQTT Broker data source hierarchy

1. Select the MQTT Broker connection, right-click it, and then click **Add MQTTGroup Connection** on the shortcut menu.

A new MQTT group connection is created in the hierarchy tree and is named **New_MQTTGroup_000** by default. Rename it, if desired.

The **New_MQTTGroup_000 Parameters** configuration view (right pane) is displayed.



2. The **Device Group Name** field is automatically filled in and cannot be edited. The Device Group Name is used for accessing MQTT data from a DDE/SuiteLink client.
3. Select a level of **Quality of Service (QoS)** from the drop-down list. The QoS level determines message delivery parameters, with 0 as the lowest level of service, and 2 as the highest level.
 - **0 - At Most Once:** The message will be delivered no more than one time, meaning that it may not be delivered. There is no backup of the message, and if the connection to the client is lost, the message will not be delivered. No delivery acknowledgment is provided.
 - **1 - At Least Once:** The message will be delivered at least one time, but it may be delivered more than once if the sender does not receive an acknowledgment of successful transmission. The sender stores the message until a receipt acknowledgment is received.
 - **2 - Exactly Once:** The message is delivered one time only. The sender stores the message until it receives confirmation of receipt. This is the safest and slowest message transfer mode.

Note: If the MQTT publisher has a different QoS than the configured QoS, the configured QoS is used.

Configuring MQTT Device Items

Device items provide alternative names for specifying MQTT items. To add device items to your group, select the new group object and click the **Device Items** tab.

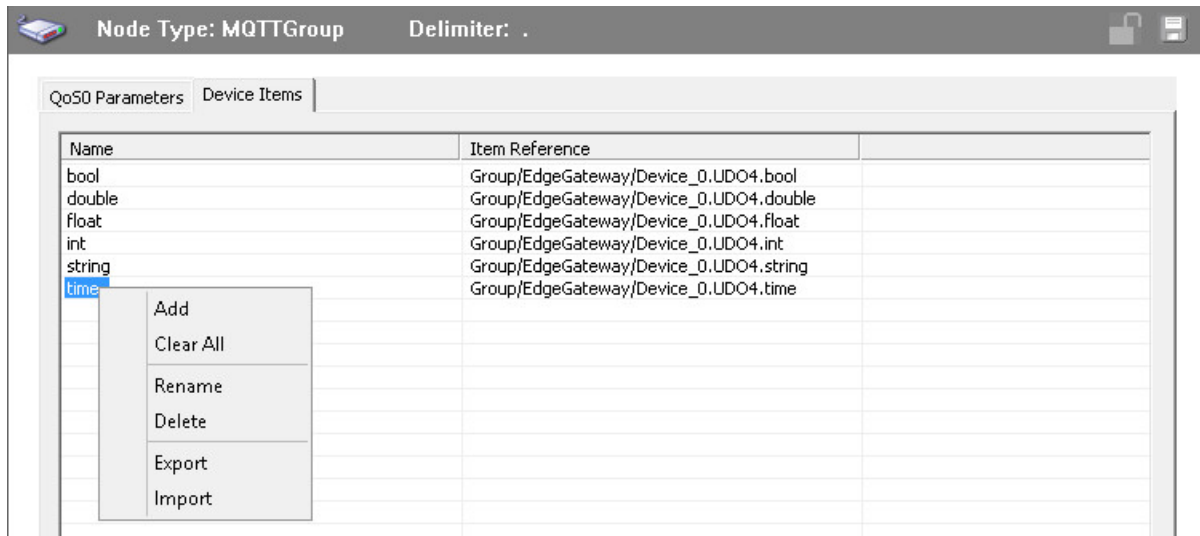
Note: The MQTT topic name and item name are case-sensitive.

The **Device Items** configuration view is used to add, clear all, rename, delete, import, and export device items.

The **Device Items** configuration view has the following two columns:

- **Name:** This column defines the alias names to actual data source items.
- **Item Reference:** The actual data source item names defined in this column.

Note: When you create or add a new device item, a unique name needs to be entered for it.



To create or add device items

1. Right-click anywhere in the **Device Items** configuration view, and select the **Add** command from the shortcut menu.
 - A device item is created, and it is numerically named by default. For example, Item_0, Item_1, and so on.
2. To change the default name, double-click it and enter the new name.
 - Enter a unique name for the new device item.

To add item references

Item references for each of the device items that have been created can be added as follows:

1. In the **Item Reference** column, double-click on the area in the same horizontal line as the selected device item.
2. In the frame that appears, type in the actual data source item name or copy the tag reference from the **Currently Selected Tag Reference** field under the **MQTT Browser** window and paste it. For more information see [Browsing Tag Items](#).

Note: To copy the tag references in bulk, use exporting and importing tag references options. For more information see [Exporting and Importing Tag References](#).

3. Click anywhere in the configuration view or press the **Enter** key to apply the change.

To rename a device item from the list

1. Right-click on the device item to be renamed, and select the **Rename** command from the shortcut menu.
2. Enter the new device item name.
3. Click anywhere in the configuration view or press the **Enter** key to apply the change.

To delete a device item from the list

- Right-click on the device item to be deleted, and select the **Delete** command from the shortcut menu. The device item and its corresponding data source item name are deleted from the configuration view.

Note: When you select another part of the MQTT tree hierarchy, you are prompted to save the modifications to the configuration set.

To clear all device items

- Right-click anywhere in the **Device Items** configuration view, and select the **Clear All** command from the shortcut menu.

All the device items listed in the configuration view, including their corresponding data source item names, are deleted.

To export device items

When you want to archive a list of device items, use the **Export** feature in the **Device Items** configuration view.

1. Right-click anywhere in the **Device Items** configuration view, and select the **Export** command from the shortcut menu.
2. Select the folder into which the list is to be saved.
3. Name the list to be exported.
4. Click the **Save** button.

The entire list is saved as a .csv file.

To import device items

The **Import** feature in the **Device Items** configuration view is used to import an archived list of device items into the configuration view.

1. Right-click anywhere in the **Device Items** configuration view, and select the **Import** command from the shortcut menu.
2. Select the archived list (.csv file) to be imported, and click **Open**.

The entire list is imported into the Device Items configuration view.

Note: Duplicate items with the same Item References are ignored during import. Duplicate items with different Item References cause a dialog box to be displayed, in which you must make a selection.

Resolving Item Names from Clients

MQTT Communication Driver resolves item names from its clients at runtime in the following order:

System items (those prefixed with \$\$SYS\$\$) > Device items (those defined in the **Device Items** configuration view) > All other items (validated directly from the PLC device)

MQTT Publisher

The MQTT Communication Driver Publisher enables you to publish data from the Application Server to the MQTT broker. You can publish payloads encoded as either Sparkplug or JSON using the MQTT publisher.

Note: Tags in the Application Server with data types ElapsedTime and CustomStruct are not supported in the MQTT Communication Driver Publisher.

Important:

The user who launches the OCMC must be part of the oiAdministrators Windows Local Users and Groups to access the MQTT Publisher. Otherwise, an error message is displayed saying the user or the user group is not

part of oiAdministrators group. If the user is not a part of the oiAdministrators group, you must add the user to the oiAdministrators group manually and re-login your computer. The user who installs the Communication Drivers Pack will be added to the oiAdministrators group automatically.

If you make any changes to the configuration of the System Management Server (SMS), you must restart the **AVEVA Communications Backend Service** in the **Services** console of your machine. In the Advanced Configuration of the System Management Server (SMS), if you change the **HTTP Port** or the **HTTPS Port** then you must run the following command as an administrator in the command prompt:

If you change the HTTP Port field

```
netsh http add urlacl url=http://localhost:<https port configured in SMS>/oi/ user="NT Authority\Network Service"
```

If you change the HTTPS Port field

```
netsh http add urlacl url=https://localhost:<https port configured in SMS>/oi/ user="NT Authority\Network Service"
```

If you do not run the above commands after changing the ports and try to access the MQTT Publisher functionality, you will get an HTTP 503 error saying the service is unavailable.

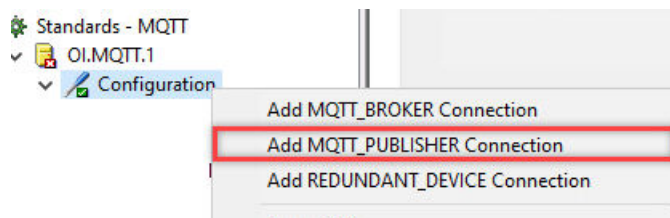
The MQTT Publisher provides an easy-to-configure topology to publisher data. Logical devices can be created which bind appropriate data points from the data source. The logical devices, simply called Devices in this document, are organized underneath an Edge Gateway (EON) which defines the publishing parameters and publishes the data defined in each Device.

Adding and Configuring the Dedicated MQTT Publisher

Adding a Dedicated MQTT Publisher Hierarchy

This is the preferred way to create a publisher so that you can independently start/stop the publisher whenever any configuration change in the publisher is required.

To add the publisher hierarchy, expand the MQTT Communication Driver node, right-click **Configuration**, and then select **Add MQTT_PUBLISHER Connection**.



Configuring the MQTT Broker

The configuration of the MQTT Broker for the MQTT Publisher is similar to the Broker of the MQTT Subscriber. Refer to the [Adding and Configuring an MQTT Broker Connection](#) section for more information on the MQTT Broker configuration.

Node Type: MQTT_PUBLISHER Delimiter: .

New_MQTT_PUBLISHER_000 Parameters | Publisher

Step 1: Configure MQTT broker connection


Broker Address:

Port Number:

Persist Session: Validate Address and Port

Step 2: (Optional) Encrypt connection with TLS

Enable

 Security Unknown: Connection to the MQTT broker not established.

Select Transport Layer Security (TLS) version:

CA File: Download

If the configured broker is using an untrusted (self-signed) digital certificate, click Download to use the broker certificate for this connection

Validate Security

Step 3: (Optional) Enter Connection/User Identity

Connection Identity

Enable

Client Certificate File: ...

Client Key File: ...

Client Key Password: Show

User Identity

Enable

User Name:

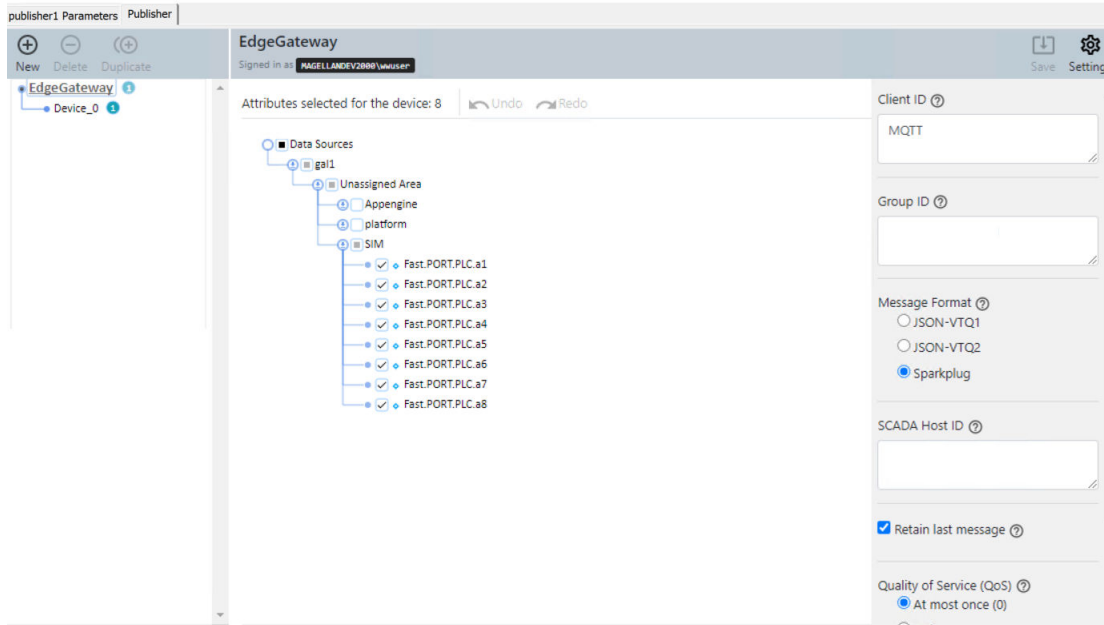
Password: Show

Validate Identity

Configuring the MQTT Publisher

The MQTT Publisher window is divided into three main parts. The left pane displays the list of devices, the middle pane displays the details of each device, and the right pane (Settings pane) displays the configuration options for the device. You can toggle the visibility of the Settings pane by clicking on the **Settings** icon.

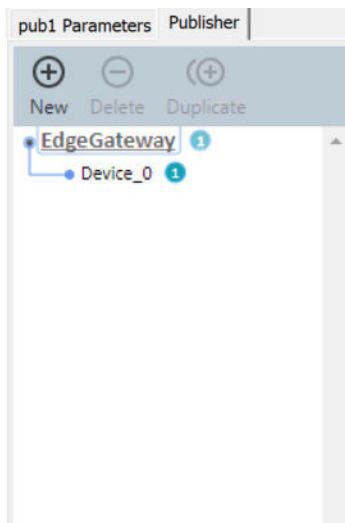
The following figure shows the MQTT Publisher screen:



Devices Pane (Left Pane)

The left pane displays a list of added devices under a gateway. At the top of the hierarchy, you can see the Gateway name, under which you can see the name of the devices.

Note: The Gateway also acts as a device. Hence, when you click the Gateway name, you can view the corresponding attributes in the middle pane. It is recommended to create your own device and not to use the Gateway as a device.



To add a device

1. Click the **New** icon present at the top of the hierarchy on the grey ribbon.
The newly added device appears at the end of the hierarchy, indicated as New in the blue color.

To delete a device

1. Select a device that you want to delete.

2. Click the **Delete** icon present at the top of the hierarchy on the grey ribbon or right-click and select **Delete**.
A message asking for confirmation to delete appears.
3. Click **OK**.
The selected device is deleted.

Note: Deleting a device is irreversible.

To add a duplicate device

1. Select a device of which you want to create a duplicate. The duplicated device will contain the same device settings and attributes bound in the duplicating device.
2. Click the **Duplicate** icon present at the top of the hierarchy on the grey ribbon or right-click and select **Duplicate**.
The duplicate device of the selected device is created.

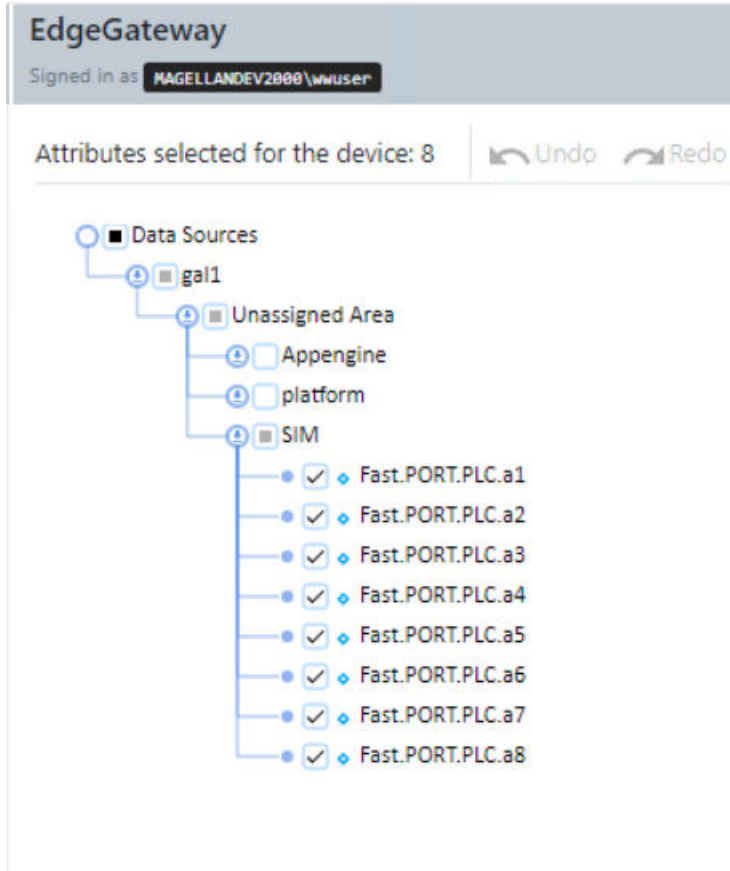
To rename a device

1. Select a device that you want to rename.
2. Right-click and select **Rename**.
3. Type the required name.

Note: Only alphanumeric characters of maximum length 40 are allowed.

Attribute Details Pane (Middle Pane)

The middle pane displays the attribute details of each device in a tree view. It displays the main attributes of the Galaxies which are deployed in the Application Server. Use this pane to select the attributes whose data you want to publish.



Just-In-Time Browsing

To optimize the amount of data downloaded from the data source, the hierarchy tree in the middle pane employs Just-In-Time browsing when displaying information in different hierarchies. If you see the download icon



in the hierarchy, clicking it will download all its immediate child nodes and attributes, when selection of attributes will only be possible on that hierarchy.

To achieve optimal performance in the display, please click only on hierarchies where you are interested to select attributes for the device.

To select the attributes

- To select a single attribute, select the corresponding check box of the required attribute.
- To select multiple attributes together, left-click and drag the mouse over the attributes that you want to select.
- To select all the attributes under a node, select the corresponding parent node of the attributes that you want to select.

To undo and redo the selections, select the **Undo** and **Redo** buttons on the menu bar respectively. The menu bar also displays the number of selected attributes out of the total number of attributes available. You can also view the number of selected attributes inside the blue circle present next to the respective device.

The grey ribbon present on the top of the middle pane displays the name of the device that is selected. To save your changes, select the **Save** icon present at the right end of the grey ribbon.

To unselect the attributes

- To unselect a single attribute, clear the check box of the required attribute.
- To unselect multiple attributes together, left-click and drag the mouse over the attributes while holding the Shift key.
- To unselect all the attributes under a node, clear the corresponding parent node of the attributes that you want to unselect.

To toggle the selection of attributes

To toggle the selection of attributes, left-click and drag the mouse over the attributes while holding the Ctrl key.

Settings Pane (Right Pane)

Use the settings pane to configure the Edge Gateway and Device connection. To view the settings pane, click the **Settings** icon present on the right end of the grey ribbon. To hide the Settings pane, click the **Settings** icon again.

If you click the Edge Gateway node the configuration options for the gateway are displayed. If you click on any Device node, the configuration settings for the device are displayed.

The screenshot shows a settings pane with the following sections:

- Client ID**: A text input field containing "MQTT".
- Group ID**: An empty text input field.
- Message Format**: Three radio button options: "JSON-VTQ1", "JSON-VTQ2", and "Sparkplug" (which is selected).
- SCADA Host ID**: An empty text input field.
- Retain last message**: A checked checkbox.
- Quality of Service (QoS)**: Three radio button options: "At most once (0)" (selected), "At least once (1)", and "Exactly once (2)".
- Enable Store and Forward**: A checked checkbox.
- Storage Limit Options**: Two radio button options: "Wrap-around Data" (selected) and "Stop Collecting".

Click the icon to read the description of the parameter.

To configure the settings of the MQTT Publisher

1. In the **Client ID** field, enter a unique string of up to 100 characters to identify the MQTT connection. If this field is left blank, a random string is assigned while saving the settings. The client ID is used to uniquely identify a connection of the publisher to the broker. Each group must have a unique Client ID.
2. In the **Group** field, enter the name of the group with up to 100 characters. The newline, space, +, or # character is not allowed. All publishing items will be sent under this group name. The group name is used as a prefix in the MQTT topic name. For example, if the group name is Group1 and the Edge Gateway name is EG1, the published topic name will be Group1/EG1.
3. In the **Message Format** field, select either Sparkplug or JSON format that is used to publish data. For the format of the JSON data, refer to Chapter 3 [MQTT Item Naming](#).
4. In the **SCADA Host ID** field, enter the unique ID that identifies the Primary Host/Application in the network. This permits the MQTT Publisher to respond to the STATE Online/Offline message published by the Primary Host to the MQTT Broker.

The value should be identical to the SCADA Host ID that was used during the MQTT Broker configuration (see [Adding and Configuring an MQTT Broker Connection](#) on page 7). The value can be blank if you do not want the publisher to respond to the Sparkplug STATE Online/Offline message.

5. To keep the last published message at the broker, select the **Retain last message** check box.
6. Under the **Quality of Service (QoS)** field, select a level of Quality of Service (QoS). The QoS level determines message delivery parameters, with 0 as the lowest level of service, and 2 as the highest level.
 - **At Most Once (0):** The message will be delivered no more than one time, meaning that it may not be delivered. There is no backup of the message, and if the connection to the client is lost, the message will not be delivered. No delivery acknowledgment is provided.
 - **At Least Once (1):** The message will be delivered at least one time, but it may be delivered more than once if the sender does not receive an acknowledgment of successful transmission. The sender stores the message until a receipt acknowledgment is received.
 - **Exactly Once (2):** The message is delivered one time only. The sender stores the message until it receives confirmation of receipt. This is the safest, and slowest, message transfer mode.
7. To store and forward the data when the connection with the broker is lost, select the **Enable Store and Forward** checkbox. Once the connectivity is restored, the published data is forwarded to the broker.

Each MQTT group has a maximum Store and Forward cache size of 4 GB. Once this limit is reached, depending on the following selection, the data is discarded or the collection of data is stopped.

- **Wrap-around Data**

If this option is selected, the oldest data stored in the cache is discarded to allow storage space for new data.

- **Stop Collecting**

If this option is selected, the system ceases to collect and store any new data. Once the connection is established again, the stored files are forwarded, which reduces the cache size, allowing storage space for new data.

Note: MQTT publisher configuration is shared across all configuration sets archived from the original set. The changes in publisher configuration in any archive set are also reflected in all configuration sets archived from the

original set. For more information on archiving the configuration sets, refer to the Archiving Configuration Sets section of the AVEVA Communication Drivers Pack help.

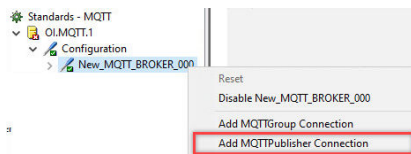
Note: If the data source (that is, Galaxy in System Platform) contains a lot of objects and attributes, you may observe large memory consumption and latency when you interact with Publisher Configuration in the OCMC. Keeping the total number of attributes that you want to view in the Publisher to less than 5000 will improve the browsing experience in the Publisher Configuration in the OCMC.

Adding and Configuring the MQTT Publisher Under the MQTT Broker

Adding Publisher Hierarchy Under the MQTT_BROKER

MQTT publisher created in this way will share the same broker connection with the subscriber. Any change either on the subscriber or on the publisher that requires a restart will also force a restart on both the publisher and subscriber. You can only create 1 publisher per MQTT_BROKER hierarchy.

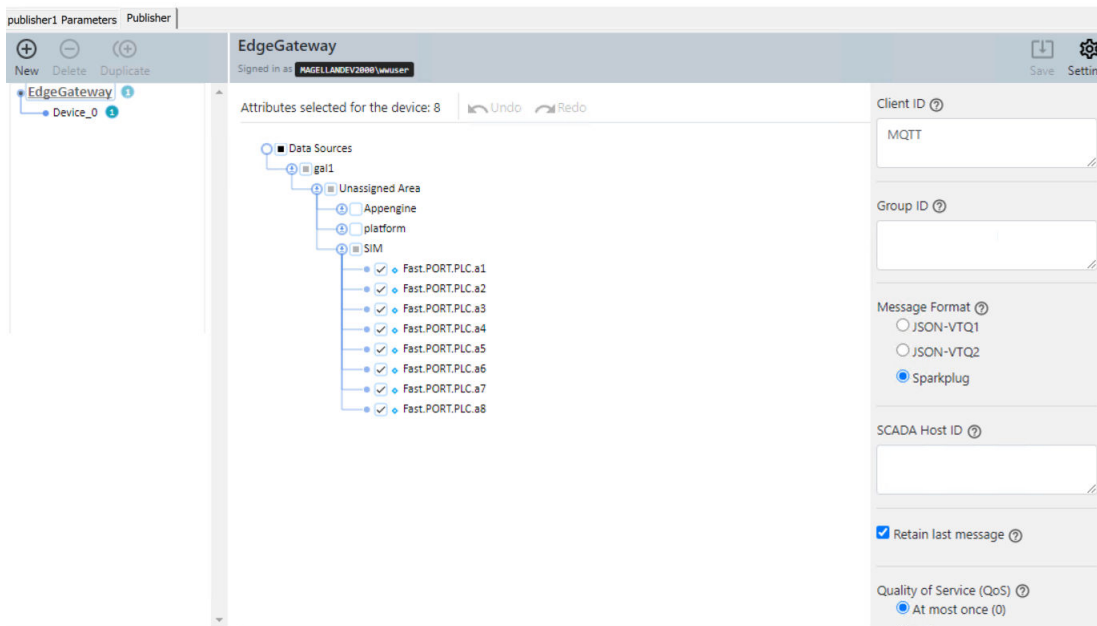
To add the publisher hierarchy, expand the MQTT Communication Driver node, right-click the Broker connection node, and then select **Add MQTTPublisher Connection**.



Configuring the MQTT Publisher

The MQTT Publisher window is divided into three main parts. The left pane displays the list of devices, the middle pane displays the details of each device, and the right pane (Settings pane) displays the configuration options for the device. You can toggle the visibility of the Settings pane by clicking on the **Settings** icon.

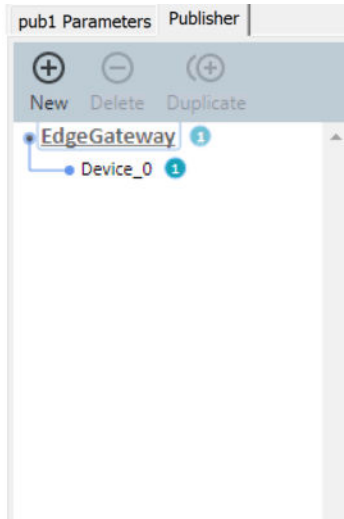
The following figure shows the MQTT Publisher screen:



Devices Pane (Left Pane)

The left pane displays a list of added devices under a gateway. At the top of the hierarchy, you can see the Gateway name, under which you can see the name of the devices.

Note: The Gateway also acts as a device. Hence, when you click the Gateway name, you can view the corresponding attributes in the middle pane. It is recommended to create your own device and not to use the Gateway as a device.



To add a device

1. Click the **New** icon present at the top of the hierarchy on the grey ribbon.

The newly added device appears at the end of the hierarchy, indicated as New in the blue color.

To delete a device

1. Select a device that you want to delete.
2. Click the **Delete** icon present at the top of the hierarchy on the grey ribbon or right-click and select **Delete**.

A message asking for confirmation to delete appears.

3. Click **OK**.

The selected device is deleted.

Note: Deleting a device is irreversible.

To add a duplicate device

1. Select a device of which you want to create a duplicate. The duplicated device will contain the same device settings and attributes bound in the duplicating device.
2. Click the **Duplicate** icon present at the top of the hierarchy on the grey ribbon or right-click and select **Duplicate**.

The duplicate device of the selected device is created.

To rename a device

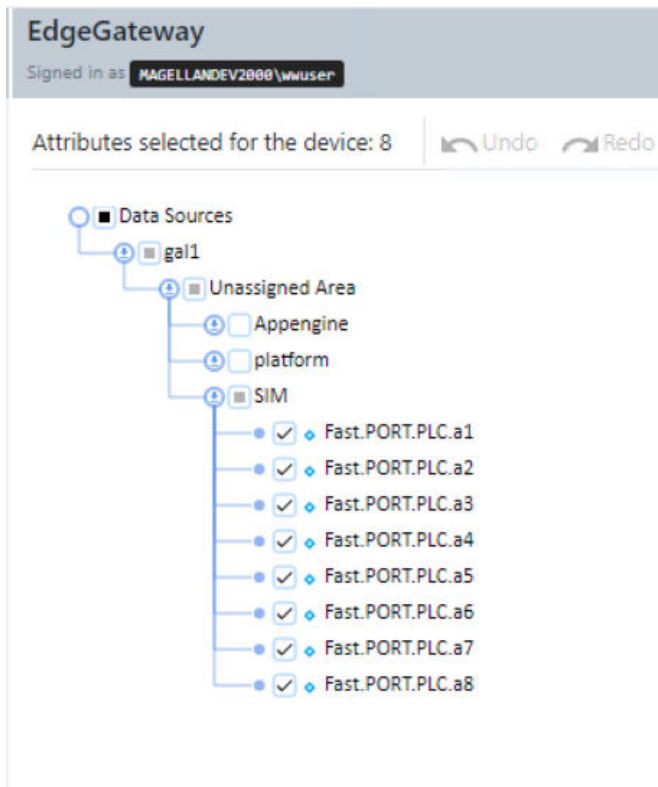
1. Select a device that you want to rename.

2. Right-click and select **Rename**.
3. Type the required name.

Note: Only alphanumeric characters of maximum length 40 are allowed.

Attribute Details Pane (Middle Pane)

The middle pane displays the attribute details of each device in a tree view. It displays the main attributes of the Galaxies which are deployed in the Application Server. Use this pane to select the attributes whose data you want to publish.



Just-In-Time Browsing

To optimize the amount of data downloaded from the data source, the hierarchy tree in the middle pane employs Just-In-Time browsing when displaying information in different hierarchies. If you see the download icon



in the hierarchy, clicking it will download all its immediate child nodes and attributes, when selection of attributes will only be possible on that hierarchy.

To achieve optimal performance in the display, please click only on hierarchies where you are interested to select attributes for the device.

To select the attributes

- To select a single attribute, select the corresponding check box of the required attribute.
- To select multiple attributes together, left-click and drag the mouse over the attributes that you want to select.

- To select all the attributes under a node, select the corresponding parent node of the attributes that you want to select.

To undo and redo the selections, select the **Undo** and **Redo** buttons on the menu bar respectively. The menu bar also displays the number of selected attributes out of the total number of attributes available. You can also view the number of selected attributes inside the blue circle present next to the respective device.

The grey ribbon present on the top of the middle pane displays the name of the device that is selected. To save your changes, select the **Save** icon present at the right end of the grey ribbon.

To unselect the attributes

- To unselect a single attribute, clear the check box of the required attribute.
- To unselect multiple attributes together, left-click and drag the mouse over the attributes while holding the Shift key.
- To unselect all the attributes under a node, clear the corresponding parent node of the attributes that you want to unselect.

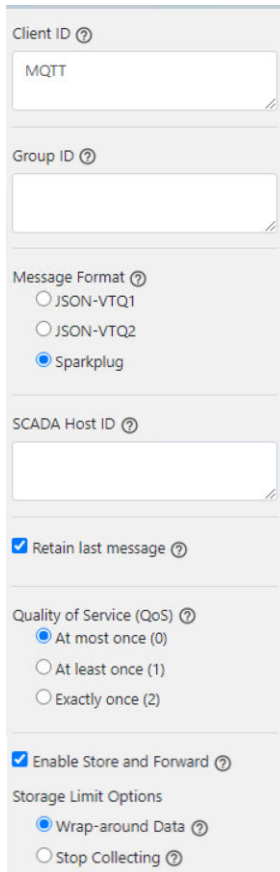
To toggle the selection of attributes


To toggle the selection of attributes, left-click and drag the mouse over the attributes while holding the Ctrl key.

Settings Pane (Right Pane)

Use the settings pane to configure the Edge Gateway and Device connection. To view the settings pane, click the **Settings** icon present on the right end of the grey ribbon. To hide the Settings pane, click the **Settings** icon again.

If you click the Edge Gateway node the configuration options for the gateway are displayed. If you click on any Device node, the configuration settings for the device are displayed.



Click the  icon to read the description of the parameter.

To configure the settings of the MQTT Publisher

1. In the **Client ID** field, enter a unique string of up to 100 characters to identify the MQTT connection. If this field is left blank, a random string is assigned while saving the settings. The client ID is used to uniquely identify a connection of the publisher to the broker. Each group must have a unique Client ID.
2. In the **Group** field, enter the name of the group with up to 100 characters. The newline, space, +, or # character is not allowed. All publishing items will be sent under this group name. The group name is used as a prefix in the MQTT topic name. For example, if the group name is Group1 and the Edge Gateway name is EG1, the published topic name will be Group1/EG1.
3. In the **Message Format** field, select either Sparkplug or JSON format that is used to publish data. For the format of the JSON data, refer to Chapter 3 [MQTT Item Naming](#).
4. In the **SCADA Host ID** field, enter the unique ID that identifies the Primary Host/Application in the network. This permits the MQTT Publisher to respond to the STATE Online/Offline message published by the Primary Host to the MQTT Broker.

The value should be identical to the SCADA Host ID that was used during the MQTT Broker configuration (see [Adding and Configuring an MQTT Broker Connection](#) on page 7). The value can be blank if you do not want the publisher to respond to the Sparkplug STATE Online/Offline message.

5. To keep the last published message at the broker, select the **Retain last message** check box.

6. Under the **Quality of Service (QoS)** field, select a level of Quality of Service (QoS). The QoS level determines message delivery parameters, with 0 as the lowest level of service, and 2 as the highest level.
- **At Most Once (0):** The message will be delivered no more than one time, meaning that it may not be delivered. There is no backup of the message, and if the connection to the client is lost, the message will not be delivered. No delivery acknowledgment is provided.
 - **At Least Once (1):** The message will be delivered at least one time, but it may be delivered more than once if the sender does not receive an acknowledgment of successful transmission. The sender stores the message until a receipt acknowledgment is received.
 - **Exactly Once (2):** The message is delivered one time only. The sender stores the message until it receives confirmation of receipt. This is the safest, and slowest, message transfer mode.
7. To store and forward the data when the connection with the broker is lost, select the **Enable Store and Forward** checkbox. Once the connectivity is restored, the published data is forwarded to the broker.

Each MQTT group has a maximum Store and Forward cache size of 4 GB. Once this limit is reached, depending on the following selection, the data is discarded or the collection of data is stopped.

- **Wrap-around Data**

If this option is selected, the oldest data stored in the cache is discarded to allow storage space for new data.

- **Stop Collecting**

If this option is selected, the system ceases to collect and store any new data. Once the connection is established again, the stored files are forwarded, which reduces the cache size, allowing storage space for new data.

Note: MQTT publisher configuration is shared across all configuration sets archived from the original set. The changes in publisher configuration in any archive set are also reflected in all configuration sets archived from the original set. For more information on archiving the configuration sets, refer to the Archiving Configuration Sets section of the AVEVA Communication Drivers Pack help.

Note: If the data source (that is, Galaxy in System Platform) contains a lot of objects and attributes, you may observe large memory consumption and latency when you interact with Publisher Configuration in the OCMC. Keeping the total number of attributes that you want to view in the Publisher to less than 5000 will improve the browsing experience in the Publisher Configuration in the OCMC.

Importing and Exporting the MQTT Publisher Configuration

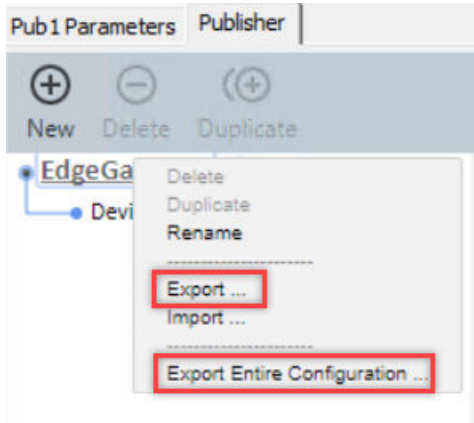
You can import and export an Edge Configuration or a Device Configuration or the whole Publisher configuration to a text file. You can edit the exported file offline. Use the notepad or an externally available JSON editor to view or edit the configuration. You can add, delete, or modify any PCS attributes and edit Edge Gateway or Device connection parameters.

The export/import feature allows you to:

- Edit the configuration offline
- Save the configuration outside
- Create a similar configuration on another server

To export the MQTT publisher configuration

1. Right-click on the required Edge Gateway or a Device which you want to export.
2. Click **Export** to export a particular hierarchy or click **Export Entire Configuration** to export the whole publisher configuration.



3. In the **File Download** Window, provide the location where you want to save the exported file.
4. Click **Save**.

The configuration file will be saved with the file extension .oicfg. The different types of exported files are explained below in detail.

Edge Gateway configuration export file

For edge gateway, the file name of the exported file will be <edge gateway name>.edge.oicfg.

JSON file example for the edge gateway configuration:

```

{
  "EdgeGateway": {
    "name": "EdgeGateway",
    "retain": true,
    "qos": 0,
    "snfwrap": true,
    "snfenable": false,
    "clientid": "test",
    "format": "SP",
    "scadahost": " ",
    "groupname": "val13",
    "updaterate": 1000,
    "attributes": [
      {
        "name": "gal1:SIM.Fast.I1",
        "metrics": {
          "attributepath": "gal1:SIM.Fast.I1",
          "hierpath": "gal1:Area_001.SIM.Fast.I1",
          "datatype": -1,
          "isarray": false,
          "readonly": false
        }
      }
    ]
  }
}
    
```

Settings parameters

Attribute parameters

For the description of different fields in the example, refer to the "Field Description" section.

Device configuration export file

For device, the file name of the exported file will be <device name>.dev.oicfg.

JSON file example for the device configuration:

```

"Device": [
  {
    "name": "Device_0",
    "retain": true,
    "qos": 0,
    "attributes": [
      {
        "name": "gal1:SIM.Fast.PORT.PLC.f1",
        "metrics": {
          "attributepath": "gal1:SIM.Fast.PORT.PLC.f1",
          "hierpath": "gal1:Area_001.SIM.Fast.PORT.PLC.f1",
          "datatype": -1,
          "isarray": false,
          "readOnly": false
        }
      }
    ]
  }
]
    
```

For the description of different fields in the example, refer to the "Field Description" section.

Entire Publisher configuration export file

This will include both Edge Gateway and Device configuration information. For the entire configuration, the file name will be <selected node name>.all.oicfg.

JSON file example for entire publisher configuration:

```

"EdgeGateway": {
  "name": "EdgeGateway",
  "retain": true,
  "qos": 0,
  "snfwrap": true,
  "snfenable": false,
  "clientid": "test",
  "format": "SP",
  "scadahost": " ",
  "groupname": "val13",
  "updaterate": 1000,
  "attributes": [
    {
      "name": "gal1:SIM.Fast.I1",
      "metrics": {
        "attributepath": "gal1:SIM.Fast.I1",
        "hierpath": "gal1:Area_001.SIM.Fast.I1",
        "datatype": -1,
        "isarray": false,
        "readOnly": false
      }
    }
  ]
}

"Device": [
  {
    "name": "Device_0",
    "retain": true,
    "qos": 0,
    "attributes": [
      {
        "name": "gal1:SIM.Fast.PORT.PLC.f1",
        "metrics": {
          "attributepath": "gal1:SIM.Fast.PORT.PLC.f1",
          "hierpath": "gal1:Area_001.SIM.Fast.PORT.PLC.f1",
          "datatype": -1,
          "isarray": false,
          "readOnly": false
        }
      }
    ]
  }
]

```

Edge Gateway Configuration

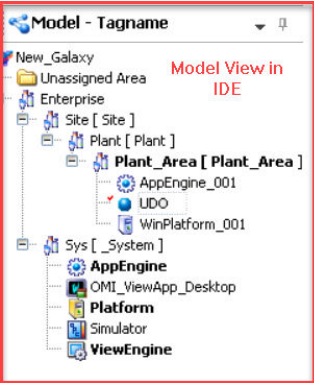
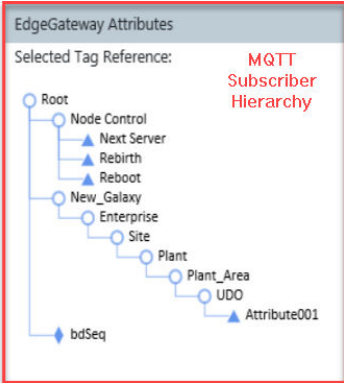
Device Configuration

For the description of different fields in the example, refer to the "Field Description" section.

Field Description

The following table describes different fields that you can see in a JSON file:

Field name	Description
attributepath	Depicts the path or the hierarchy of the attribute in the data source. In this case, the data source is Galaxy. The format of the attribute path is <GalaxyName>.ObjectName.AttributeName. Refer to "Reference Strings" in the Application Server User Guide for more information.

<p>hierpath</p>	<p>Depicts the hierarchical view of the attribute in the OI MQTT subscriber hierarchy. Its format is similar to the the Model View of the hierarchical path in the IDE in System Platform.</p> <p>For example, assume the galaxy name in the IDE is "newGalaxy" and the attribute name is "Attribute001." If the hierarchy of the model view is Enterprise > Site > Plant, then the hierarchy path is represented as newGalaxy:Enterprise.Site.Plant.Attribute001.</p> <div style="display: flex; justify-content: space-around;">   </div> <p>If this field is not specified, the value of attributepath will be used to depict the hierarchical representation of the OI MQTT subscriber hierarchy.</p>
<p>datatype</p>	<p>Specifies the data type. Refer to the Data Types section for more information.</p>
<p>isarray</p>	<p>Specifies whether the attribute is an array attribute or not. Set this field to false as array is not supported.</p>
<p>readOnly</p>	<p>Specifies whether the attribute is a read-only attribute or not. Set this field to true is the attribute is read-only. Otherwise, set it to false.</p>

Data Types

The following table describes the different values for the datatype fields in the JSON file, and the corresponding datatypes:

Value	Data Type	Description
3	Boolean	A simple type representing Boolean values of true or false.

4	Char	An integral type representing unsigned 16-bit integers with values between 0 and 65535.
5	SByte	An integral type representing signed 8-bit integers with values between -128 and 127.
6	Byte	An integral type representing unsigned 8-bit integers with values between 0 and 255.
7	Int16	An integral type representing signed 16-bit integers with values between -32768 and 32767.
8	UInt16	An integral type representing unsigned 16-bit integers with values between 0 and 65535.
9	Int32	An integral type representing signed 32-bit integers with values between -2147483648 and 2147483647.
10	UInt32	An integral type representing unsigned 32-bit integers with values between 0 and 4294967295.
11	Int64	An integral type representing signed 64-bit integers with values between -9223372036854775808 and 9223372036854775807.
12	UInt64	An integral type representing unsigned 64-bit integers with values between 0 and 18446744073709551615.
13	Single	A floating point type representing values ranging from approximately 1.5×10^{-45} to 3.4×10^{38} with a precision of 7 digits.
14	Double	A floating point type representing values ranging from approximately 5.0×10^{-324} to 1.7×10^{308} with a precision of 15-16 digits.

16	DateTime	A type representing a date and time value. The OI MQTT Publisher will publish the value in unit of number of milliseconds since January 1, 1970 UTC. The OI MQTT Subscriber will convert it to VT_DATE Windows data type so that System Platform applications can consume it as a Window data type.
18	String	Represents Unicode character strings.

Importing the MQTT Publisher Configuration

You can edit the exported file and import it back to the MQTT Publisher. This feature helps you to:

- Edit the configuration offline
- Save the configuration outside
- Import to create similar configuration on another server

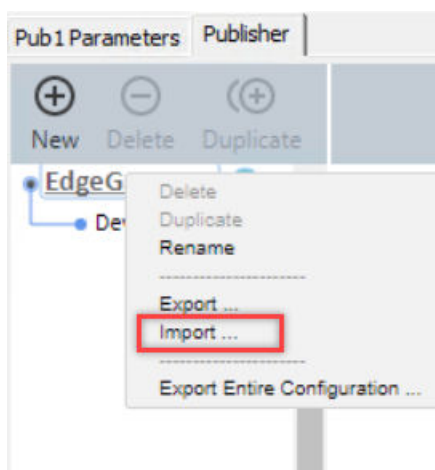
The configuration parameters in an imported file and exported file are similar. Refer to the "Field Description" section above for more information on the parameters.

To import the MQTT publisher configuration

1. On the MQTT Publisher hierarchy, right-click on the required node where you want to import the configuration.
 - If you right-click on the Edge Gateway node, you can either import an Edge Gateway configuration file, or the entire publisher configuration file.
 - If you right-click on a Device node, you can import only a Device configuration file.

Note: You cannot import a device configuration file on an edge gateway node and vice versa. Else, an error message is displayed.

2. Click **Import**.



3. Select the file that you want to import.

- To import the Edge Gateway configuration, the file name should be <edge gateway name>.edge.oicfg.
- To import the Device configuration, the file name should be <device name>.dev.oicfg.
- To import the entire publisher configuration, the file name should be <selected node name>.all.oicfg.

Note: If you import the configuration of an edge gateway or device that already exists, the configuration in the imported file will replace the existing configuration. If you import the entire publisher configuration, the newly imported configuration replaces the entire existing configuration.

Chapter 3

Diagnostic System Items

- [Standard System Items](#)
- [MQTT Communication Driver-Specific Diagnostics](#)

Standard System Items

System items provide you with easy access to the MQTT Communication Driver's status and diagnostics information. They are treated just like ordinary items with respect to the client. However, in most cases these items are not directly acquired via the communications layer. System item values are usually generated through internal calculations, measurements, and the tracking of the OI Engine.

System items, like ordinary items, are defined by the following properties:

- **Group** (client group/OPC group): The arbitrary collection of items, not correlated.
- **Hierarchical location** (link name/OPC path, the hierarchical node section of the fully qualified OPC item ID): The device the item is attached to.
- **Device group** (OPC access path/topic, or a Scan Group on a hierarchical branch): A collection of items on the same physical location with the same protocol update rate.

In the ArchestrA context, the device group plays the most important role of identifying the scope of any item. The device group defines the hierarchical location implicitly when using globally unique device-group names, which is required for DDE/SuiteLink compatibility.

All system items follow the same naming convention:

- All system items start with \$SYS\$.
- The OI Engine scans and parses the name for system items. Parsing of the name is case-insensitive.

All system items can be accessed through subscriptions to a Device Group. However, while some system items return data for that Device Group, others are server-wide.

For DDE/SuiteLink clients, you can access \$SYS\$Status always comes from the leaf level of the MQTT Communication Driver hierarchy branch, which is the destination data source.

For OPC clients, \$SYS\$Status can be accessed at all hierarchy levels. \$SYS\$Status at the root level of the whole hierarchy tree is always good, as it represents the quality status of the local computer itself. Hence, for practical application, OPC clients should reference \$SYS\$Status at any hierarchy levels other than the root. In the case of an ArchestrA data source, \$SYS\$Status is always good, even at the ArchestrA Group level.

Global System Items

The following global system items refer to information regarding the data source(s) the MQTT Communication Driver is connected to.

System Item Name	Type/Access Rights	Description	Values
\$SYS\$Licensed	Boolean/ Read	MQTT connections require a license.	The value is always true.
\$SYS\$ReadOnly	Boolean/Read	Binary status indication of the read-only state of a MQTT Communication Driver. If TRUE, the Read/Write access of all items are overridden to read only and cannot be written. If an item is written, a line is logged in the OCMC Logger and the write request is rejected. If FALSE, the Communication Driver items are read/write or read only according to their individual configurations.	Range: 0, 1 1: Read only 0: Not read only

Device-Specific System Items

The following system items refer to specific information regarding the data source(s) MQTT Communication Driver is connected to.

System Item Name	Type/Access Rights	Description	Values
\$SYS\$Status	Boolean/ Read	Binary status indication of the connection state to the device (hierarchy level) the item is attached to. The device group (OPC access path/topic) does not affect the value. The status can be good even if individual items have errors.	RANGE: 0, 1 1: MQTT connection to the data source is intact. 0: Error communicating with the data source.

System Item Name	Type/Access Rights	Description	Values
\$SYS\$ErrorCode	Longint/ Read	Detailed error code of the communications state to the data source. The device group (OPC access path/topic) does not affect the value.	>= 0: Good status (0 is the default state – connected). >0: is some state like: connecting, initializing, etc. <0: Error status (value indicates the error).
\$SYS\$ErrorText	String/ Read	Detailed error string of the communications state of the data source. The device group (OPC access path/topic) does not affect the value.	Descriptive text for the communications state corresponding to the error code.
\$SYS\$StoreSettings	Integer/ Read Write	Not used.	

Device Group-Specific System Items

The following system items refer to specific information regarding device groups that have been configured in the MQTT Communication Driver.

System Item Name	Type/ Access Rights	Description	Values
\$SYS\$updateInterval		Not used.	
\$SYS\$maxInterval		Not used.	
\$SYS\$WriteComplete	Integer/ ReadWrite	Accesses the state of pending write activities on the corresponding device group. On device group creation (adding items to an OPC group), the value of this system item is initially 1 If the value is not zero, the client can poke 1 or -1 to it (poke a 1 to clear errors, or a -1 to test a client reaction on write errors). If the value of this item is zero, it cannot be poked.	RANGE: -1, 0, 1 1: Write complete (no writes are pending – initial state). 0: Writes are pending. -1: Writes completed with errors.

System Item Name	Type/ Access Rights	Description	Values
\$SYS\$ReadComplete	Integer/ ReadWrite	<p>Accesses the state of initial reads on all items in the corresponding device group.</p> <p>Poking a 0 resets the internal read states of all items in this device group. This resets this item to 0. If all items are read again after this poke, this item changes back to 1 or -1.</p>	<p>RANGE: -1, 0, 1</p> <p>1: Read complete (all values have been read).</p> <p>0: Not all values have been read.</p> <p>-1: All values have been read but some have a non-good quality.</p>
\$SYS\$ItemCount	DWord/ Read	<p>Accesses the number of items in the corresponding device group. This item is read-only.</p>	<p>RANGE: 0... 2147483647</p> <p>>=0: Number of active items.</p>
\$SYS\$ActiveItemCount	DWord/ Read	<p>Accesses the number of active items in the corresponding device group. This item is read-only.</p>	<p>RANGE: 0... 2147483647</p> <p>>=0: Number of active items.</p>
\$SYS\$ErrorCount	DWord/ Read	<p>Accesses the number of all items (active and inactive) that have errors (non-good OPC quality) in the corresponding topic.</p> <p>If the communications status of a device group is bad, all items have errors. This item is read-only.</p>	<p>RANGE: 0... 2147483647</p> <p>>=0: Number of all items (active and inactive) with errors.</p>
\$SYS\$PollNow	Boolean/ ReadWrite	<p>Poking a 1 to this item forces all items in the corresponding device group to be read immediately.</p> <p>This is useful if you want to force to get the newest values from the device, regardless of its update interval.</p> <p>This also works on device groups with a zero update interval (manual protocol triggering).</p>	<p>RANGE: 0, 1,</p>

MQTT Communication Driver-Specific Diagnostics

The MQTT Communication Driver supports driver-specific diagnostic system items that provide essential information on the status and performance of the MQTT Communication Driver.

The diagnostics items are categorized into 3 main groups:

- [Global Diagnostic Items](#) Diagnostic items in this group reflects the different operation metrics across each connection to the MQTT Broker.
- [Subscriber Diagnostic Items](#): Diagnostic items in this group reflect operational metrics of the subscriber of the MQTT driver.
- [Publisher Diagnostic Items](#): Diagnostic items in this group reflect operational metrics of the publisher of the MQTT driver.

Global Diagnostic Items

Global diagnostic items provide diagnostic information related to the entire MQTT connection.

The syntax to access a global diagnostic item is:

`/SYSDIAG/<Item Name>`

For example, the syntax to access the IP address and port number details of the MQTT broker is:

`/SYSDIAG/BrokerAddress`

Refer to the following table for the entire of list of global diagnostic items available for the MQTT Communication Driver.

Global Diagnostic Items

<ItemName>	Data Type	Read/Write	Subscriber/Publisher Availability	Description
BrokerAddress	String	R	Both	Displays the IP address and port number of the MQTT broker or host name that was configured while connecting to the broker. Example: <i>127.0.0.1:1883</i> , where 127.0.0.1 is the IP address and 1883 is the port number.
ClientId	String	R	Both	Displays the unique identifier of the MQTT subscriber/publisher client connection to the broker. Each connection to the broker is assigned a unique ID during the broker or publisher configuration.

PersistSession	Bool	R	Both	Displays 'True' or 'False' based on whether the Persist Session checkbox was enabled during the MQTT broker configuration. When enabled, the broker stores any QoS1 or QoS2 messages that have not been retrieved.
PayloadType	String	R	Both	Displays the format of the message payload. For the subscriber, the value can be either JSON or JSON_Sparkplug. For the publisher, the value can be JSON-VTQ1, JSON-VTQ2, or Sparkplug.
ServerCertificateExpiryDate	VT_DATE (DateTime)	R	Both	Displays the TLS Certificate expiry date of the MQTT broker. If TLS is not used, then value is initialized to 01/01/1970 00:00:00 AM and Quality = 0x20.
ServerCertificateSubject	String	R	Both	Displays the TLS Certificate subject of the MQTT broker. If TLS is not used, then the value is initialized to an empty string.
ClientCertificateExpiryDate	VT_DATE (DateTime)	R	Both	Displays TLS Certificate expiry date of the MQTT client. If TLS is not used, then value is initialized to 01/01/1970 00:00:00 AM and Quality = 0x20.
ClientCertificateSubject	String	R	Both	Displays TLS Certificate subject of the MQTT client. If TLS is not used, then the value is initialized to empty string.
IsAnonymous	Bool	R	Both	Displays 'True' if user credentials are not provided while configuring the broker. Empty if the user credentials are provided.
UserName	String	R	Both	Displays the user credentials provided while configuring the broker. (Empty if IsAnonymous=True)
StartDateTime	VT_DATE (DateTime)	R	Both	Displays the Date and Time when the MQTT Communication Driver was started.

UpTime	Int64 (DateTime)	R	Both	Displays the elapsed time in seconds since the MQTT Communication Driver was started.
ConnectionEvents	Int32	R	Both	Displays the number of times the MQTT publisher/subscriber connects to the MQTT broker.
LastConnectTime	VT_DATE (DateTime)	R	Both	Displays the time when the last connection of the MQTT subscriber/publisher to the broker was successful.
LastDisconnectTime	VT_DATE (DateTime)	R	Both	Displays the time when the MQTT subscriber/publisher last disconnected from the broker.
SubscribedItemCount	Int32	R	Subscriber only	Displays the total number of subscribed items under the broker.
PublishedItemCount	Int32	R	Publisher only	Displays the total number of published items under the broker.
LastPublishTime	VT_DATE (DateTime)	R	Both	Displays the time when the last message was published in the broker. Default value is 01/01/1970 00:00:00 AM and Quality = 0x20
TotalReadErrors	Int32	R	Subscriber only	Displays the total number of failed read operations in all connections since the MQTT Communication Driver was started or last reseted. (An error is counted if one or more item reads failed, or the entire read is not executed).
TotalWriteErrors	Int32	R	Both	Displays the total number of failed write operations in all connections since the MQTT Communication Driver was started or last reseted. (An error is counted if one or more item writes failed, or the entire write is not executed)

TotalPublishErrors	Int32	R	Publisher only	Displays the total number of failed publish operations in all connections to the broker since the MQTT Communication Driver was started or last reseted.
ResetTotals	Bool	RW	Both	Displays 'True' or 'False' based on whether there was a reset of the values of the following system items: TotalReadItemErrors, TotalWriteItemErrors, ConnectionEvents, TotalDataMessagesPublished, CumulativeBytesSent and CumulativeBytesReceived. Default value: False
LastPayloadSizeIncoming	Int32	R	Mainly for Subscriber	Displays the length of incoming buffer in the response from the MQTT broker.
LastPayloadSizeOutgoing	Int32	R	Both	Displays the length of outgoing buffer in the request from the MQTT Communication Driver.
CumulativeBytesReceived	Int64	R	Mainly for Subscriber	Displays the cumulative count of total bytes received from MQTT broker since the driver was started or last reseted (aggregate of LastPayloadSizeIncoming).
CumulativeBytesSent	Int64	R	Both	Displays the cumulative count of total bytes sent to MQTT broker since the driver was started or last reseted (aggregate of LastPayloadSizeOutgoing).
StoreAndForwardActive	Bool	R	Publisher only	Displays 'True' or 'False' based on whether the Enable Store and Forward checkbox was enabled during publisher configuration. If the value is 'True', there is an outage in the connection to the

				<p>MQTT broker. Publishing data is being stored locally and will be sent once the connection is up.</p> <p>If this value is 'False', the MQTT Communication Driver is publishing data directly to the MQTT broker.</p>
TotalPublishingDevices	Int32	R	Publisher only	Displays the total number of publishing devices connected to the broker.
TotalDataMessagesPublished	Int32	R	Publisher only	Displays the total number of data messages published since MQTT Communication Driver the driver was started or last reseted.
Run	Bool	RW	Both	<p>Displays 'True' (non-zero) if the system items calculation and reporting is enabled. True is the default value.</p> <p>Displays 'False' if the system items calculation and reporting is disabled.</p> <p>The quality of system items is set to 0x14 (LAST KNOWN VALUE) if the system item reporting is not enabled.</p>

Subscriber Diagnostic Items

The subscriber diagnostic items provide diagnostic information specific to the MQTT Subscriber.

Subscriber-specific Diagnostic Items

Item Syntax	DataType	Read/Write	Description
\$SYS\$SP\$isPrimaryApp	Boolean	R	Displays "True" if the MQTT subscriber is configured to be the Primary Application in the Sparkplug infrastructure.
\$SYS\$SP\$PrimaryAppStatus	String	R	Takes on a value of "Online" or "Offline" based on the online/offline status of the Sparkplug Infrastructure.

\$SYS\$SP\$SCADA_Host_ID	String	R	This is the SCADA Host ID identifying the Sparkplug infrastructure that the Subscriber/Publisher is part of. This value must be identical to that configured in the Primary Application. It can be blank to indicate that the Subscriber/Publisher does not participate the Sparkplug infrastructure.
--------------------------	--------	---	---

Subscriber Query Items

Query items extract diagnostic data related to the MQTT Subscriber based on the QoS and/or payload type. The query item syntax for the MQTT Subscriber is:

```
/$SYS$DIAG/<ItemName>?<QoS#>&<Payload Type>
```

where,

- **ItemName** is the diagnostic system item. The item name can be one of the following:

<ItemName>	Data Type	Read/Write	Description
SubscribedItemCount	Int32	R	Displays the total number of subscribed items under the broker.
TotalReadItemErrors	Int32	R	Displays the total number of items failed to read in all connections since the MQTT Communication Driver was started or last reseted. (An error is counted if one or more item reads failed, or the entire read is not executed).
TotalWriteItemErrors	Int32	R	Displays the total number of items failed to write in all connections since the MQTT Communication Driver was started or last reseted. (An error is counted if one or more item writes failed, or the entire write is not executed)

Note: These items can also be accessed at the global level.

- **QoS#** is the Quality of Service value. The QoS value can be one of the following:
 - QoS0
 - QoS1
 - QoS2

- **Payload type** is the format of the message payload. The payload type can be one of the following:
 - JSON
 - SP (Sparkplug)

For example, the syntax to access the total number of subscribed items with QoS1 under the broker, in the JSON payload format, is:

```
/$SYS$DIAG/SubscribedItemCount?QoS1&JSON
```

Publisher Diagnostic Items

The publisher diagnostic items provide diagnostic information specific to the MQTT Publisher.

Publisher Query Items

Query items extract diagnostic data related to the MQTT Publisher based on the edge gateway/device and payload type. The query item syntax for the MQTT Publisher is

- `/SYSDIAG/<Item Name>?EG&<Payload Type>` (at the edge gateway level), or
- `/SYSDIAG/<Item Name>?D=<Device Name>&<Payload Type>` (at the device level)

A default device group `<Publisher Node Name>_DIAG$` has been added in the Publisher to access the diagnostics data of **SuiteLink and PCS clients**. The query item syntax for SuiteLink and PCS clients is:

- `<Publisher node name>_DIAG$/<Item Name>?EG&<Payload Type>` (at the edge gateway level), or
- `<Publisher node name>_DIAG$/<Item Name>D=<Device Name>&<Payload Type>` (at the device level)

where,

- **<Item Name>** is the diagnostic system item. The item name can be one of the following:

<ItemName>	Data Type	Read/Write	Description
PublishedItemCount*	Int32	R	Displays the total number of published items under the MQTT broker.
QoS	String	R	Displays the Quality of Service configured in Publisher Device. [Example = QoS0, QoS1, QoS2]
RetainLastPublishMessage	Bool	R	Displays 'True' or 'False' based on whether the Retain last message checkbox was selected during the configuration of the MQTT publisher. Value 'True' means that the last published message is kept in the broker.

PublisherBirthMessageTime	VT_DATE (DateTime)	R	Displays the time when publishing starts. Default value is 01/01/1970 00:00:00 AM and Quality = 0x20.
PublisherDeathMessageTime	VT_DATE (DateTime)	R	Displays the time when publishing stops. Default value is 01/01/1970 00:00:00 AM & Quality = 0x20.

* These items can also be accessed at the global level

- **EG** is the Edge Gateway
- **D=<DeviceName>** is the name of the specific device under the edge gateway

Note: D=0 represents the edge gateway

- **Payload type** is the format of the message payload. The payload type can be one of the following:
 - JSON
 - SP (Sparkplug)

For example, to access the Quality of Service of a publisher device (name: Device_1), the syntax is:

`/SYSDIAG/QoS?D=Device_1`

For a PCS Client to access the Quality of Service of a publisher device (name: Device_1), the syntax is:

`pub1_$DIAG$/QoS?D=Device_1` (**pub1** is the name of the publisher node)

Note: Diagnostic items for Publisher (Node Type = MQTTPublisher) under the MQTT Broker is not supported. The node type MQTT_Publisher which is directly under root Configuration node supports diagnostic items.

Chapter 4

MQTT Item Naming

The MQTT Communication Driver can publish JSON-VTQ1, JSON-VTQ2, and Sparkplug messages.

The MQTT Communication Driver can subscribe to JSON-VTQ1, JSON-VTQ2, Sparkplug, and plain text MQTT messages.

Publisher

The publisher in the MQTT Communication Driver publishes its data in:

- JSON-VTQ1
- JSON-VTQ2
- Sparkplug

JSON-VTQ1

In this format, a single item is published under its own MQTT topic.

The name of the MQTT topic is:

Group/ItemName

The format of the JSON string in each topic is:

```
{“d”: value, “dt”: datatype, “ts”: timestamp, “q”: quality }
```

where,

- **Group:** optional prefix as defined in the publisher **Group ID** field:
 - If Group is blank, the / character is omitted and the ItemName is used as the topic name.
 - If Group is not blank, the / character is added automatically to the topic name between the Group Name and the Item Name.
 - Must not contain the slash / character.
- **ItemName:** attribute name defined in the publisher.
- **value:** data value of item (in JSON convention).
- **datatype:** ordinal value of Microsoft’s VARENUM Variant data type. For example: 2 is VT_I2, 3 is VT_I4, 8 is VT_BSTR, 11 is VT_BOOL
- **timestamp:** timestamp of the data point in UTC expressed in ISO 8601 format. For example: 2021-06-17T22:46:07Z = June 17 2021, 10:46:07 pm UTC. The ISO 8601 timestamp string ends with Z.

- **quality:** OPC DA quality of the data point. For example: 192 = good; 0 = bad, 4 = Configuration error, 24 = Communication failure.

JSON-VTQ2

In this format, multiple items are published under a single topic (up to 1000 items per topic).

Each item in the JSON payload corresponds to a selected attribute from the data source in the System Platform. Multiple values of the same item can be published in the same MQTT payload message if they are changed within a second or when the publisher is disconnected from the broker with store-and-forward mode enabled.

For items in the Edge Gateway (EON), the topic is: **Group/EON**

For items in the Device, the topic is: **Group/EON/Device**

The format of the JSON string in each topic is an array of JSON string is:

```
{
  "timestamp": timestamp_sent, "values": [
    quality { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
    quality { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
    ...
    quality { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
    quality { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
  ]
}
```

where,

- **Group:** optional prefix as defined in the publisher in MQTT Publisher configuration. If this value is not blank, the / character is added automatically to the publishing MQTT topic between the Group name and EON name.
- **EON:** Edge Gateway (EON) name published by the MQTT publisher.
- **Device:** Device name published by the MQTT publisher.
- **timestamp_sent:** timestamp when the message was sent from the MQTT publisher to the MQTT broker. The value is expressed in the number of milliseconds since Jan 1, 1970, UTC (Unix format).
- **itemName:** attribute name defined in the publisher of Gateway Communication Driver.
- **value:** data value of item (in JSON convention).
- **dataType:** ordinal value of Microsoft's VARENUM Variant data type. For example: 2 is VT_I2, 3 is VT_I4, 8 is VT_BSTR, 11 is VT_BOOL
- **timestamp_ds:** timestamp of data value at the data source expressed in the number of milliseconds since Jan 1, 1970, UTC (Unix format).
- **quality:** OPC DA quality of the data point. For example: 192 = good; 0 = bad, 4 = Configuration error, 24 = Communication failure.

Sparkplug

The format of the MQTT item reference published in Sparkplug is:

- For items in an Edge Gateway (EON), the item reference syntax is:
`spvB1.0/Group/EON.(Metric).value`
- For items in the Device, the item reference syntax is:
`spvB1.0/Group/EON/Device.(Metric).value`

where,

- **Group:** Group name defined in the MQTT publisher
- **EON:** Edge Gateway (EON) name published by the MQTT publisher
- **Device:** Device name published by the MQTT publisher
- **Metric:** Item name of the attribute published by the MQTT publisher

Subscriber

The MQTT Subscriber can subscribe to data in:

- Sparkplug
- JSON-VTQ1
- JSON-VTQ2
- JSON-Generic

Sparkplug

A Sparkplug MQTT item reference is of the following format.

- For items in an Edge Gateway (EON), the item reference syntax is:
`spvB1.0/Group/EON.(Metric).value`
- For items in the Device, the item reference syntax is:
`spvB1.0/Group/EON/Device.(Metric).value`

where,

- **Group:** Group name defined in the MQTT publisher
- **EON:** Edge Gateway (EON) name published by the MQTT publisher
- **Device:** Device name published by the MQTT publisher
- **Metric:** Item name of the attribute published by the MQTT publisher

JSON-VTQ1 format

This JSON format is used in the:

- MQTT Publisher in Gateway Communication Driver, and
- MQTT Publisher in the MQTT Communication Driver

to publish MQTT data. In this format, a single item is published under its own MQTT topic.

The name of the MQTT topic is:

Group/ItemName

The format of the JSON string in each topic is:

```
{“d”: value, “dt”: datatype, “ts”: timestamp, “q”: quality }
```

where,

- **Group:** optional prefix as defined in the publisher of the Communication Driver. If this value is not blank, the / character is added automatically to the topic name between the Group name and Item name.
- **ItemName:** attribute name defined in the publisher of the Communication Driver.
- **value:** data value of item (in JSON convention).
- **datatype:** ordinal value of Microsoft’s VARENUM Variant data type. For example: 2 is VT_I2, 3 is VT_I4, 8 is VT_BSTR, 11 is VT_BOOL
- **timestamp:** timestamp of the data point in UTC expressed in ISO 8601 format. For example: 2021-06-17T22:46:07Z = June 17 2021, 10:46:07 pm UTC. The ISO 8601 timestamp string ends with Z.
- **quality:** OPC DA quality of the data point. For example: 192 = good; 0 = bad, 4 = Configuration error, 24 = Communication failure.

As the subscriber in the MQTT Communication Driver recognizes this format, it can interoperate and consume the data published from the MQTT Publisher in the Gateway Communication Driver or MQTT Communication Driver.

JSON-VTQ2

This format is used by the Publisher in the MQTT Communication Driver to publish MQTT data. If you select this option, multiple items are published under a single topic (up to 1000 items per topic).

To extract the values out to System Platform, OPC, SuiteLink or PCS clients transparently without adding any parsing logic, you need to use the MQTT Communication Driver subscriber.

For items in the Edge Gateway (EON), the topic is: **Group/EON**

For items in the Device, the topic is: **Group/EON/Device**

The format of the JSON string in each topic is an array of JSON string is:

```
{
  "timestamp": timestamp_sent, "values": [
    { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
quality },
    { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
quality },
    ...
    { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
quality },
    { "id": itemName, "v": value, "dt": dataType, "t": timestamp_ds, "q":
quality }
  ]
}
```

where,

- **Group:** optional prefix as defined in the publisher in MQTT Publisher configuration. If this value is not blank, the / character is added automatically to the publishing MQTT topic between the Group name and EON name.
- **EON:** Edge Gateway (EON) name published by the MQTT publisher.
- **Device:** Device name published by the MQTT publisher.

- **timestamp_sent**: timestamp when the message was sent from the MQTT publisher to the MQTT broker. The value is expressed in the number of milliseconds since Jan 1, 1970, UTC (Unix format).
- **itemName**: attribute name defined in the publisher of Gateway Communication Driver.
- **value**: data value of item (in JSON convention).
- **dataType**: ordinal value of Microsoft’s VARENUM Variant data type. For example: 2 is VT_I2, 3 is VT_I4, 8 is VT_BSTR, 11 is VT_BOOL
- **timestamp_ds**: timestamp of data value at the data source expressed in the number of milliseconds since Jan 1, 1970, UTC (Unix format).
- **quality**: OPC DA quality of the data point. For example: 192 = good; 0 = bad, 4 = Configuration error, 24 = Communication failure.

Since the MQTT subscriber recognizes this format, the resulting VTQ (Value, Time, Quality) is readily available to any clients connecting to the MQTT subscriber with the following syntax:

- For items in the Edge Gateway (EON), the item reference syntax is: **Group/EON.ItemName**
- For items in the Device, the item reference syntax is: **Group/EON/Device.ItemName**
- If Group is blank, the item reference syntax for EON items is: **EON.ItemName**
- If Group is blank, the item reference syntax for Device items is: **EON/Device.ItemName**

Note: You have to use MQTT Communication Driver to subscribe to this format. The subscriber in Gateway Communication Driver does not recognize this format.

JSON (Generic Format)

MQTT subscriber supports embedded JSON strings, as well as strings at multiple nested levels (objects and arrays). Refer to the following examples of possible JSON string forms and the referenced values.

Example 1: If TagX receives a JSON string of form { “Value1”: 1, “Value2”:2}

Values in the JSON string are referenced as follows:

Tag Reference	Value
TagX	{ “Value1”: 1, “Value2”:2}
TagX.Value1	1
TagX.Value2	2

Example 2: If TagA receives a JSON string of form { “Value1”: 1, “Value2”:{ “value3”: 2, “value4”: 3 }, “Value5”: [5, 6] }

Values in the JSON string are referenced as follows:

Tag Reference	Value
TagA	{ “Value1”: 1, “Value2”:{ “value3”: 2, “value4”: 3 }, “Value5”: [5, 6]}
TagA.Value1	1
TagA.Value2	{ “value3”: 2, “value4”: 3 }

TagA.Value2.value3	2
TagA.Value2.value4	3
TagA.Value5	[5,6]
TagA.Value5[0]	5
TagA.Value5[1]	6

Example 3: If TagA receives a JSON string of form [“stringValue1”, “stringValue2”]

Values in the JSON string are referenced as follows:

Tag Reference	Value
TagS	[“stringValue1”, “stringValue2”]
TagS[0]	stringValue1
TagS[1]	stringValue2

MQTT Communication Driver Reference

[Runtime Poking Behavior of MQTT Communication Driver](#)

[MQTT Topic Names](#)

Runtime Poking Behavior of MQTT Communication Driver

Poking to Sparkplug Template structure items is only allowed for items with the following syntax :

```
spvB1.0/GroupName/EdgeGatewayName.(TemplateName).value.template.(MemberName).value
```

or

```
spvB1.0/GroupName/EdgeGatewayName/DeviceName.(TemplateName).value.template.(MemberName).value
```

where,

GroupName: Name of Sparkplug group

EdgeGatewayName: Name of the Edge Gateway

DeviceName: Name of the Device

TemplateName: Name of the Sparkplug template instance

MemberName: Name of the member in the Sparkplug template instance

Note: Poking to items associated with Sparkplug Data Set, Template Parameters and nested Templates is not supported.

MQTT Topic Names

Note: This section is not applicable to MQTT Sparkplug-based subscription model.

Items subscribed to an MQTT broker are known as topics. MQTT data sources use topic names for sending and receiving messages. A topic name can be divided into multiple topic levels. Each level is separated by a forward slash (/). Wild cards are not supported for MQTT topic names in this release.

The following rules apply to topic names:

- Must be at least once character in length
- Names are case sensitive

- Space character is valid and can be included
- Names can include a leading or trailing forward slash, but the forward slash counts as an identifier (/a, a/ and a are all different topic names)
- '/' is a valid topic name
- Topic names cannot include the null character (U+0000)
- Topic names are UTF-8 encoded strings. The maximum encoded length is 65535 bytes

Note: In general, MQTT supports the usage of "+" as a single level wild card and "#" as a multi-level wild card. However, the AVEVA MQTT Communication Driver does not currently support usage of wild cards.

Syntax Example

site/area1/mixer4/valve/input

Using JSON Strings

Data messages should be in string format. The string can be formatted as a JSON key value pair, which is automatically detected. JSON messages are parsed by the driver to allow the extraction of each key value pair as attributes of an object.

For example, the field device from a pump station transmits a message that includes location, pump running status, oil pressure, and maintenance data. The MQTT topic for this message is Field/FS785/Status. The payload for this message uses a JSON formatted message, such as:

```
{"lat":32.95646, "lon":-96.82275, "Pump_running":1, "Oil_Press":67.23, "Maintenance":"Last Maintenance Dec 14-2015"}
```

An application can subscribe to any of the following topics:

- Field/FS785/Status
- Field/FS785/Status.lat
- Field/FS785/Status.lon
- Field/FS785/Status.Pump_Running
- Field/FS785/Status.Oil_Press
- Field/FS785/Status.Maintenance