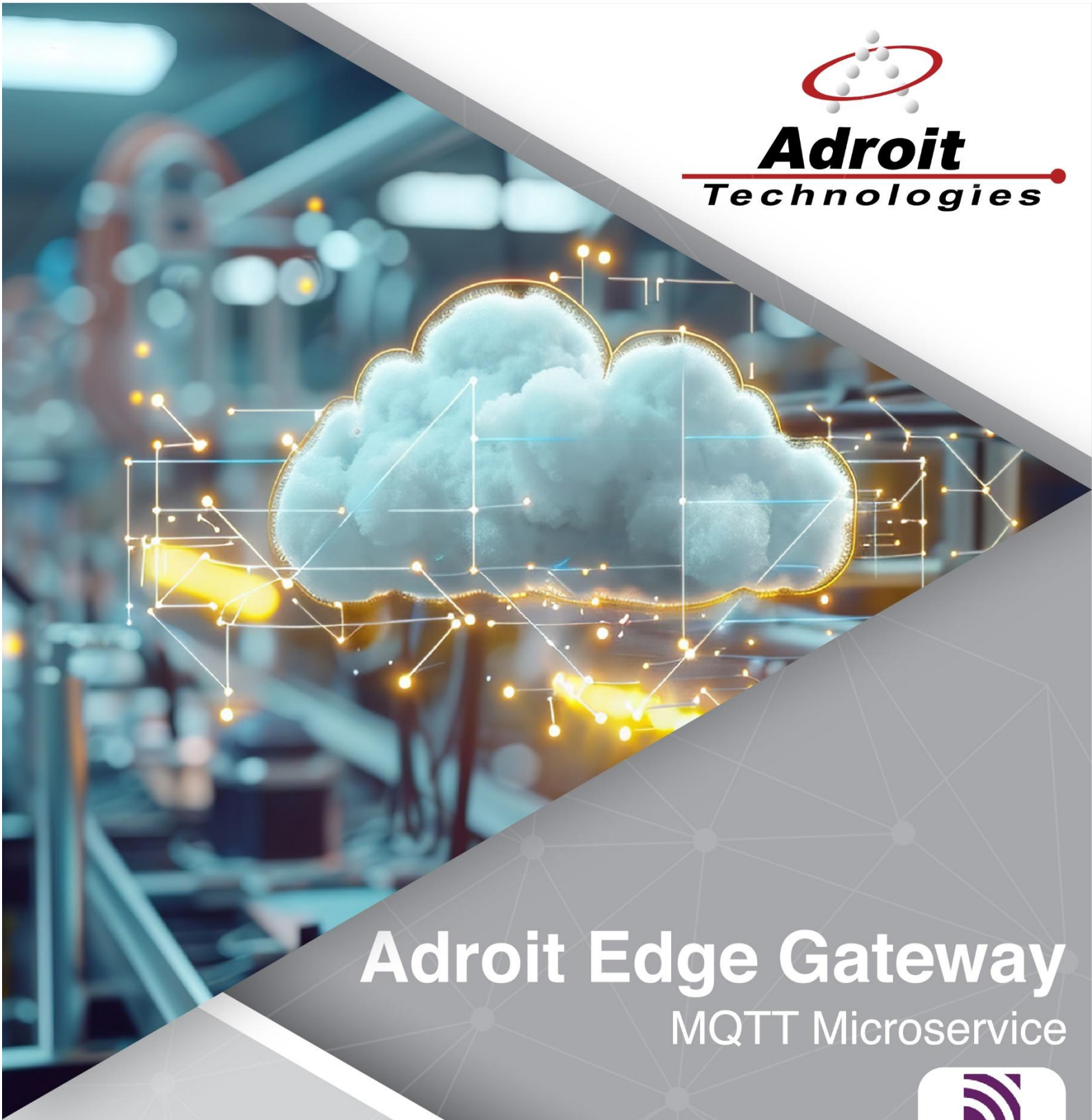
Adroit Technologies



Adroit Edge Gateway

MQTT Microservice

MQTT

Adroit
Edge Gateway

COPYRIGHT

Advanced Worx 112 (Pty) Ltd t/a Adroit Technologies
20 Waterford Office Park
189 Witkoppen Road (c/o Waterford Drive)
Fourways
South Africa

www.adroitscada.com

Version	Date	Author	Comment	Approval
1.0	25.02.2025	D. Jordaan	Initial Draft	
2.0	20.05.2025	D. Jordaan	RC28 version	
3.0	19.08.2025	D. Wibberley	Final	

Contents

1. Introduction

The Adroit Edge Gateway MQTT Microservice allows for the publishing of a user definable payload consisting of static text and one or more Dataelement - values, status and/or timestamp information.

The MQTT Broker can be anywhere, local on premises or in the Cloud that is reachable on the network/Internet.

This Document demonstrates the Microservice set up using **Mosquitto** as an on-premise MQTT broker.

The Dataelements markers referenced in the topic payload will be replaced by the real time values provided by the applicable Datasource(s).
Publishing can either be interval (periodic) or triggered.
Interval publishing is done on a cyclic basis (at a maximum of one minute resolution) on the PC clock epoch period counting from 00:00:00.

For example:
a periodic Publisher of 5 minutes, will wait for the next epoch before it attempts to publish. So if the Microservice started at 06:52:32 , the next epoch period will be 06:55:00. (See publish efficiency counter in the detailed tile view + debug level set to 4 – debug and in the datalog.)

Each publish configuration, inserts the last received Dataelement value to the payload variable to the topic. The topic is a fixed static string that is published to any broker, in CSV, XML, or JSON or any user defined text.

Note: The term Publisher and Topic terms are used interchangeably in this document.

1.1. Web Links of additional software used in this document

Mosquitto.org - Broker
Download | Eclipse Mosquitto
download the applicable binary 32/64bit - current version as of this document 2.0.21

MQTT 3rd Party browser
MQTT Explorer | An all-round MQTT client that provides a structured topic overview
Acknowledgement and thank you Thomas Nordquist
Adroit MQTT driver and Adroit Edge MQTT Microservice.

2. Configuration and Usage

Configuration of all Adroit Edge Publishers is done within the **Config Editor**, under:
Configuration → Welcome → Edge Gateway Configuration.

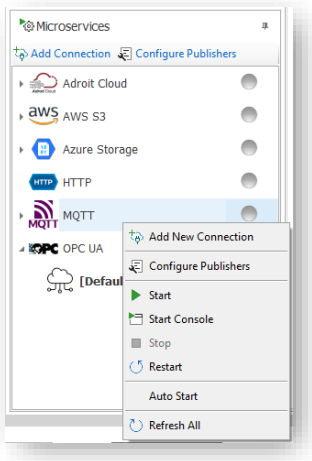
Note: Configuration is achieved by an appropriate config.xml file that is monitored for changes, some changes affecting, for example, the whole connection may result in ALL Publishers being recreated and restarted, some changes affect only the Publisher in question. Diagnostic changes to the config.xml will not result in any interruption of connections or Publishers or the interval epochs

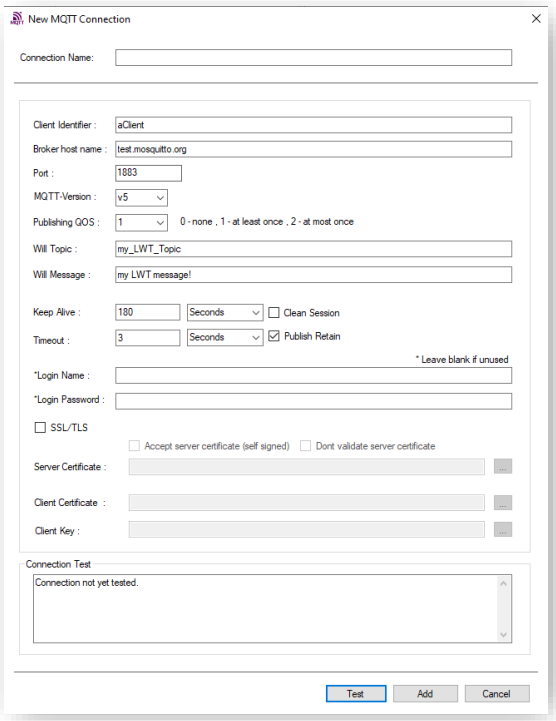
2.1. MQTT Connection

2.1.1. Adding a New Connection

To create a new **MQTT** connection:

1. Right-click the **MQTT** Microservice in the treeview.
2. Select the option to create a new MQTT connection.





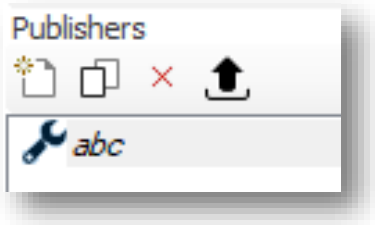
In the MQTT Connection dialog, configure the following settings:

- **Connection Name** – this is the unique name of this connection within the Microservice, the Microservice can host multiple connections to one or more brokers.
- **Client Identifier** – All connections to a broker are considered client connections.
    - The Microservice connections are push only.
    - To read data back from a broker, the Adroit MQTT driver should be used.
- **Broker host name** – Usually this will be a host name that can be used as part of the identification process used within SSL/TLS. It can also be a IP address.
- **Port** – 1883 for no SSL/TLS connection, 8883 for SLL/TLS connections. But is configurable and broker dependent.
- **MQTT-version** – V3 and V5 is supported.
- **Publishing QOS** – Quality of service agreement between client and broker used when publishing.
    - 0 – none - don't care if the message is received or consumed no acknowledgement.
    - 1 - At least once, delivery guarantee with acknowledgement.
    - 2 – Exactly once, with acknowledgement.
- **Will Topic** - Last Will and Testament topic that will be registered with this connection that potential clients can subscribe to, to be alerted if this client (the Adroit Edge Microservice) disconnects.
- **Will Message** – Payload content text that will be sent to other clients that subscribe to this topic.
- **Keep Alive** – Instructs the client to keep the connection alive by sending PINGREQ messages, useful if the connection is not very active.
- **Timeout** – timeout in seconds to wait for an acknowledgement for a transaction, including connection.
- **Clean Session** – When a client connects to a broker with clean session set to true, also persistence/retained messages are discarded.
- **Publish Retain** – Part of the MQTT integrity system, instructs the broker when the client publishes a message to persist the data as last known message, later clients will get this update when first connecting.
- **Login Name** – A unique user name. (usually not used when using SSL/TLS)
- **Login Password** – A unique password paired to the username.
- **SSL\TLS** – Enable SSL\TLS handshaking
- **Accept server certificate (self-signed)** – If checked then the user must supply the Server certificate that the server(broker) with provide during authentication (CA certificate) for comparison. Only use if the server cert is trustworthy.
- **Don't validate server certificate** – Force the client to ignore the validity of the presented certificate.
- **Server Certificate** – The CA signed broker cert (only used if using a self-signed Broker CA cert. (PEM format).
- **Client Certificate** – This client certificate signed by a trusted authority, or the same CA used to sign the server certificate. (PEM format).
- **Client Key** – The clients public key certificate (PEM format).
- **Connection Test (button and feedback list box)** – this Test button is live, reloads the various configuration parameters on each test click. Will establish a connection and send "Test" to a "Test" topic on the broker, (*future versions will allow configuration of the topic and topic content and where the publish is required.*) This is also only for indication purposes to assist in correcting configuration errors. See Sections 3 and 4 for various test scenarios.

2.2. MQTT Publishers

After creating a connection, multiple Publishers can be added by selecting the connection in the tree view and clicking **Configure Publishers** from the toolbar or by double-clicking the connection label.

2.2.1. Adding Publishers



To add a Publisher:

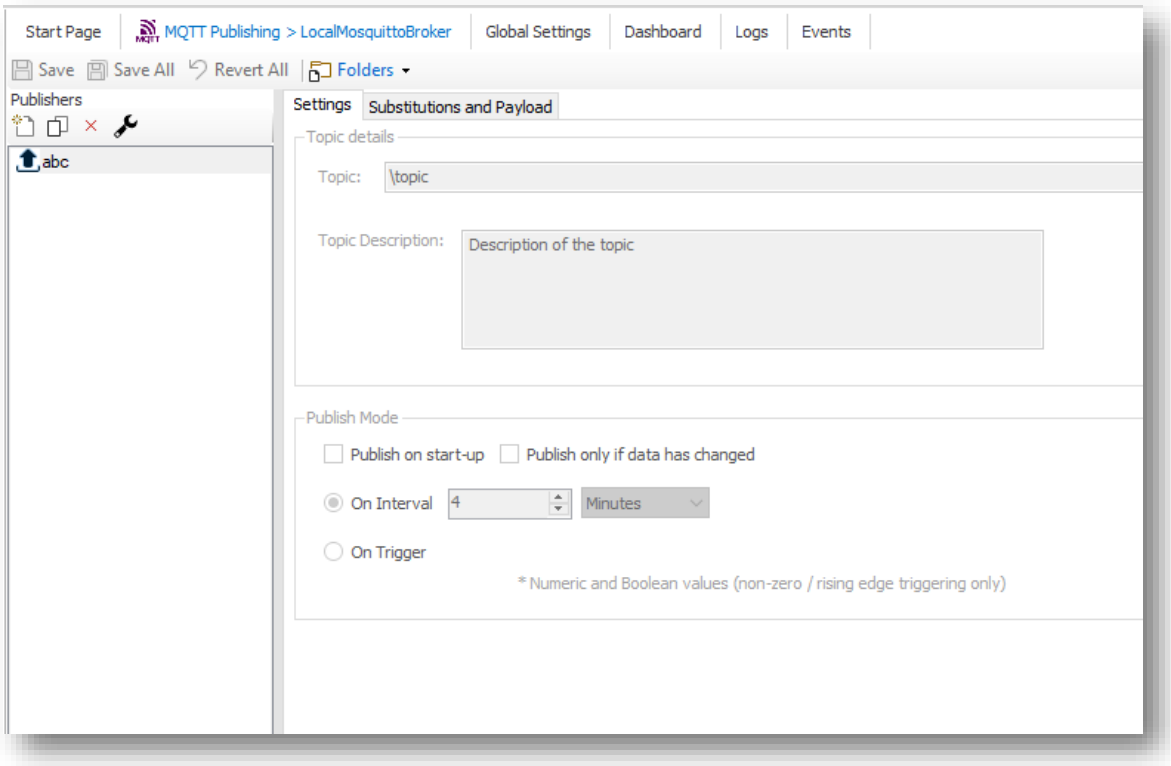
1. Click the **Add New Publisher** from the toolbar.
2. Provide a name and description for the Publisher.

Each Publisher starts in **Maintenance Mode** (not actively publishing) by default. To switch to **Publish Mode**, use the context menu or toolbar buttons options. Additional options allow you to:

- Edit the Publisher's description.
- Delete the Publisher.
- Copy the Publisher, including all settings, substitutions, and body configurations. (after a successful copy one can do a find and replace on the Tag Substitution list and regenerate the payload by clicking "Generate" in the payload view.)

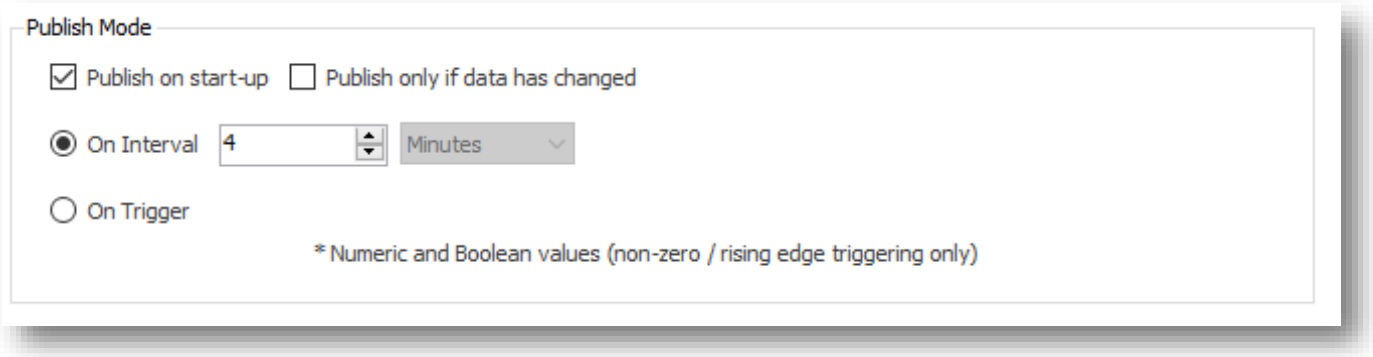
Switching modes and all configuration changes are "offline" in other words these changes only take effect when the configuration file is updated. The MQTT Microservice monitors the file for changes and will affect these changes when found. Manually editing the config.xml file is not recommended, rather use the UI tool provided.

2.2.2. Publisher Settings



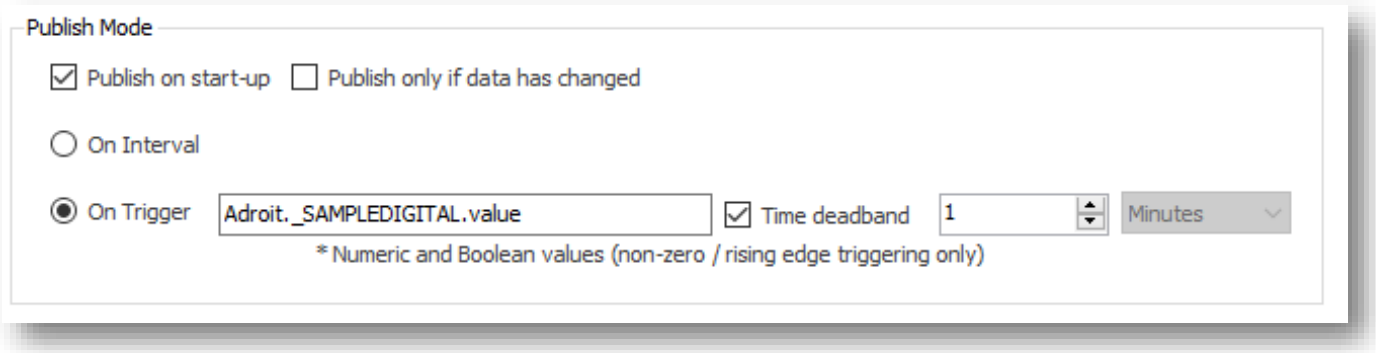
The **Settings** tab includes:

- **Topic Details**- This is the Topic on the broker that will be published to.
  - It cannot have wild cards
  - Does not need to be unique, topic Publishers are serialized on the connection.
  - No spaces are allowed but standard MQTT domain label conventions are supported
- **Topic description –** An Adroit Edge UI side prompt for greater detail or info for reference of the configurator.
- **Publish Mode Settings**:
  - **On Interval**: Publishes messages at a regular interval (defined in minutes).
    - Interval (in minutes): Specifies the publishing epoch interval.



  - **On Trigger:** Publishes messages triggered by a Dataelement value change, provided that the trigger Dataelement has changed and holds a non-zero value (Digitals – rising edge only).

- Time Deadband: If required defines the waiting period (in minutes) before accepting the next trigger after the last published time.
- The Trigger tag can be dragged and dropped from any appropriate Dataelement in any Datasource.



For both publishing modes above you can also add:

- **Publish on start-up:** This publishes on startup of the Microservice and when the Publisher is set to publish mode at runtime, the interval update epoch is also recalculated at this time.
- **Publish only if data has changed**: Publishes only if any substitution Dataelement values have changed, even if triggered.

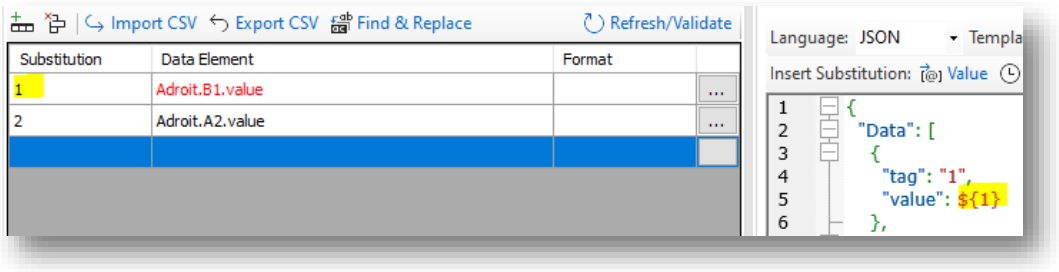
2.2.3. Substitutions and Payload

In the Body tab we can configure the substitution Dataelements for the MQTT Publisher as well as the payload content which will be sent.
We are also provided with an option to validate, preview the substituted payload message before it's been sent.

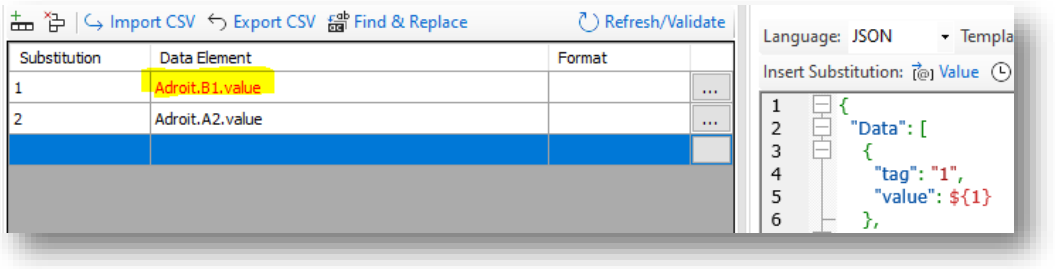
2.2.3.1. Substitutions

The Substitutions grid allows users to define multiple substitutions with corresponding Dataelements and formatting.
To add Dataelements to the Substitutions grid:

1. Navigate to the Dataelements tree view under the relevant Adroit Datasource.
2. Select the desired Dataelements (you can multiselect) and drag and drop them into the Substitutions grid.
3. Alternatively, you can directly input the full names of the Dataelements into the Substitutions grid.
4. The Substitution grid element colour will be black if there is at least one occurrence of the substitution variable in the payload, red otherwise.



5. The Dataelement grid element colour will be red if the Datasource it is derived from is either offline\unavailable or even unconfigured, or the Dataelement does not exist in the Datasource, black otherwise. E.g. B1 below does not exist in Adroit.



Additional Substitutions functionalities include:

- **CSV Import/Export**: Handles substitution lists and their configurations using CSV files. Import or export them seamlessly within the Substitution Grid.
- **Find and Replace**: Locate and modify text within substitution Dataelement names directly in the Substitution Grid. **NOTE:** This only applies to the Dataelement column.

2.2.3.2. Payload

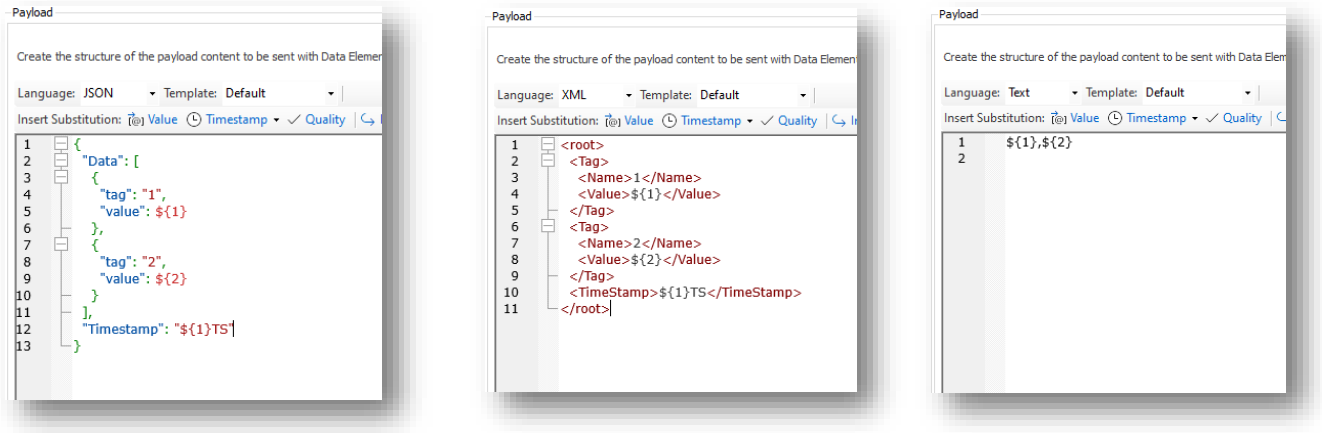
The Payload configuration provides a markup editor for creating the JSON body. Substitutions from the grid can be inserted into the JSON by selecting a substitution and using the Payload toolbar buttons to insert a substitution of specific Dataelement's data i.e. its value, quality and timestamp:

- **Value**: Inserts a ${substitution name}, which will substitute in the Dataelements value.
- **Quality**: Inserts a ${substitution name}ST, which will substitute in the Dataelements quality i.e. bad, good etc.
- **Timestamp**: Inserts a ${substitution name}TS, which will substitute in the Dataelements timestamp, Unix time UTS , Unix time milliseconds UTMS

These Substitutions can be also added manually into the JSON in the markup editor.

Additional Payload features include:

- **Templates**: Create TEXT/CSV,JSON or XML payload using predefined imported templates. (future) or Built in function when Default template ius selected. A default template generates a CSV,JSON or XML structure containing simple key-value pairs for each substitution, such as *substitution name: ${substitution name}*
- **Validation**: Verifies the JSON or XML is not malformed to ensure accuracy and receive in place feedback on any errors.

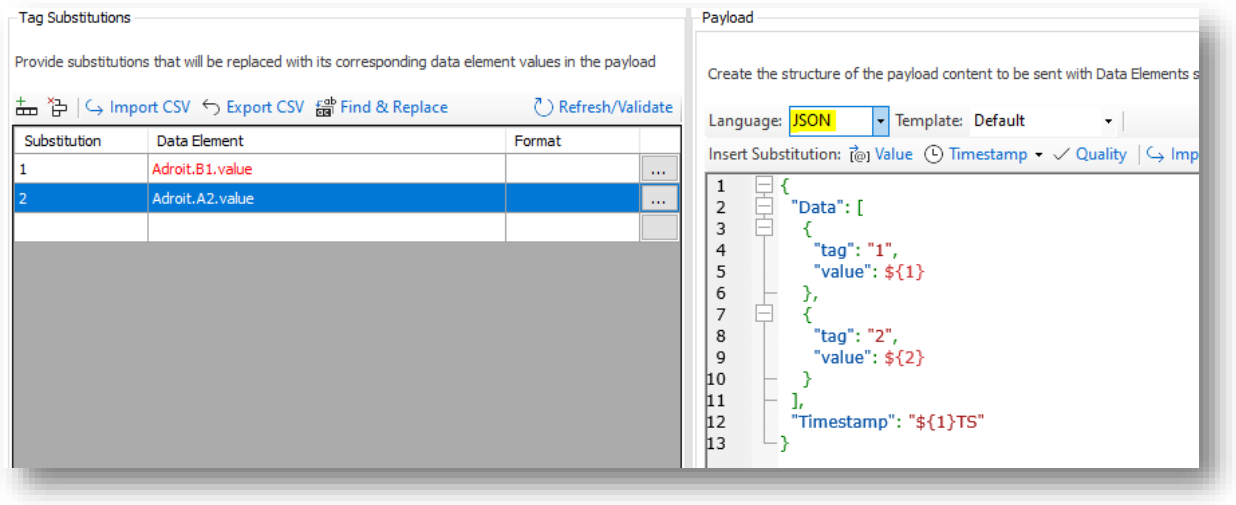


Payload generated for JSON, XML , TEXT(CSV)

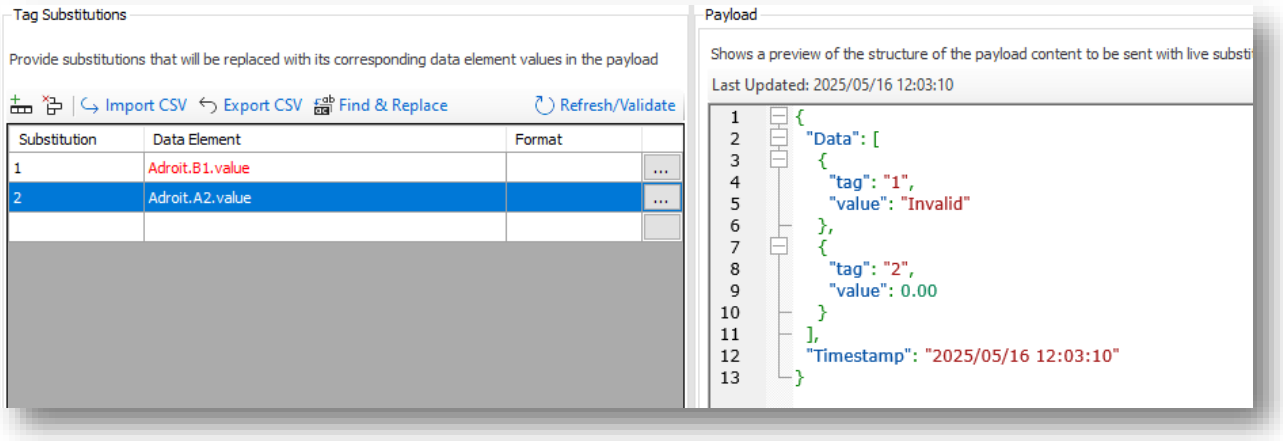
2.2.3.3. Preview and Test

The **Preview** tab allows you to:

- **View** – View the final Payload with live substituted values.
- **Test** - Post the body message to ensure functionality. If the test fails, an error response will be displayed.



Substitution and Payload generated for JSON



Payload previewed for JSON, not the invalid Dataelement showing up as "Invalid". Dataelement from row 2 showing up a 0.0 which is the last known value.

2.3. Work Folders Location

All configuration files related to the MQTT Publisher are stored in:
C:\ProgramData\Adroit Technologies\SmartDataServices\MQTT

Users typically do not need direct access to these folders, as configuration is meant be managed via the Config Editor. Key files include:
Subfolders include:

- Configuration –
  - *config.xml*: Stores all settings, Dataelement substitutions, and body configurations. Configuration changes are "offline" in other words these changes only take effect when this configuration file is updated. The MQTT Microservice monitors this file for changes and will affect these changes when found. Manually editing this file is not recommended , rather use the UI tool provided.
  - *MQTT.log*: Available only if "Log to file" is enabled. This log file must be opened manually or on the Log tab provided

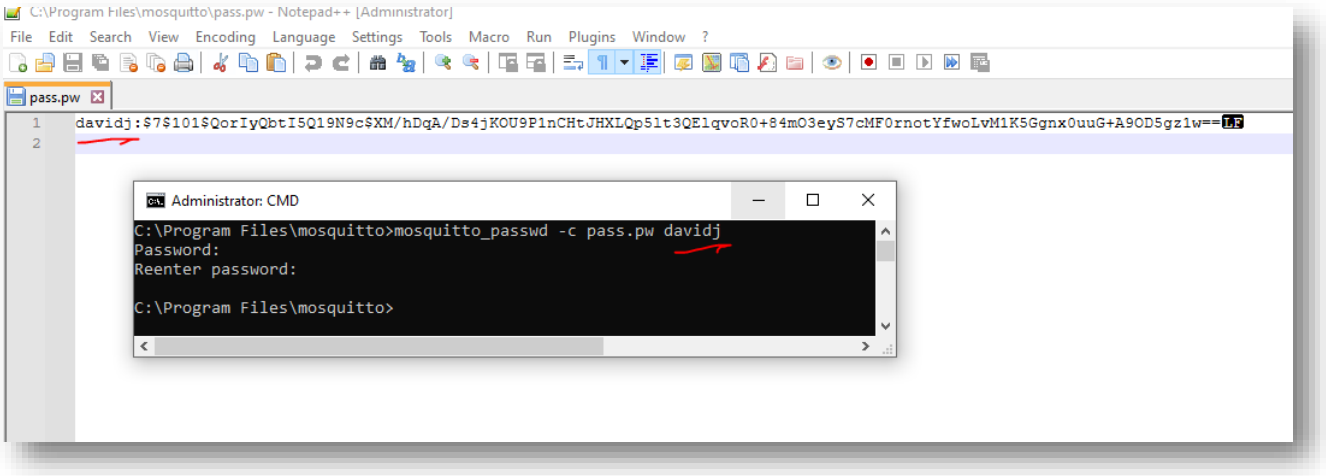
- **CSV -** Contains CSV files for importing and exporting Dataelement substitutions.
- **Templates -** Contains predefined JSON templates for generating specific body messages. Currently not supported, the only options that have a result is
  - *None – Manual or import manually.*
  - *Default – A internal simple key value pair if XML or JSON or a simple comma CSV that can be created when the "Generate" button is pressed.*

- **Backup –** Up to 10 previous saved config files will be kept in this folder , a manual copy and paste will be required to restore to a previous state. This is provided as a simple safe guard.

3. Authentication using Username and Password

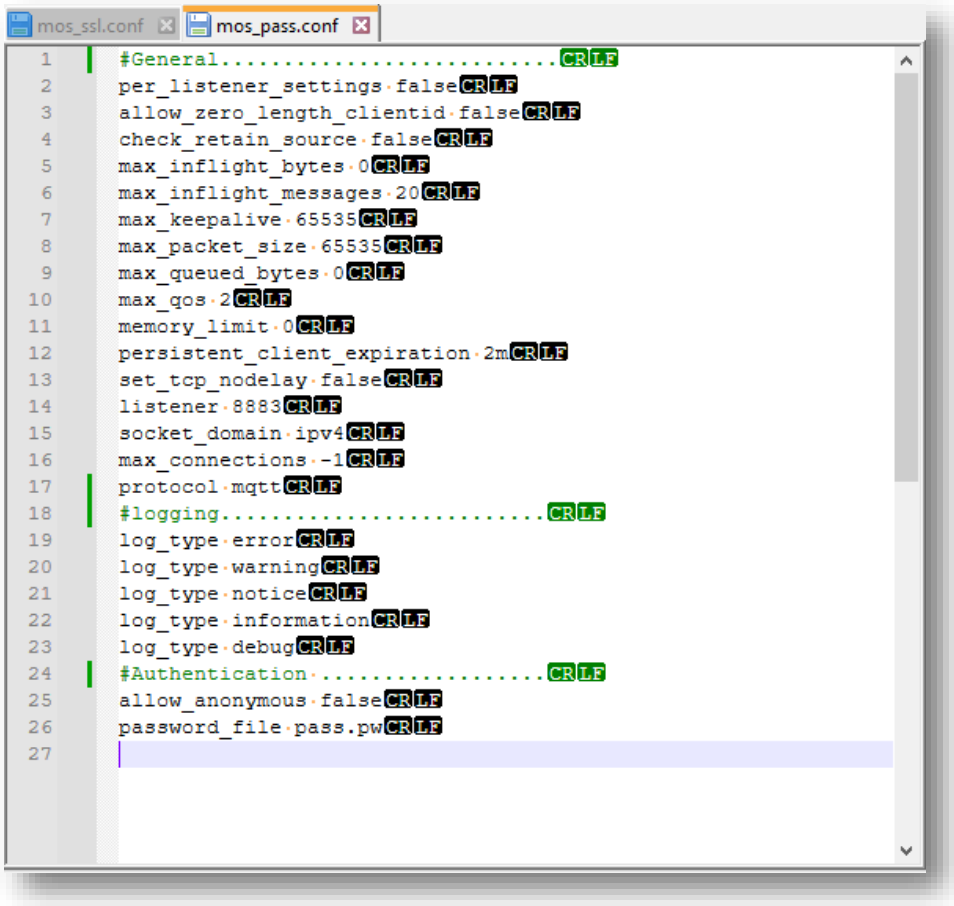
3.1. Example using the Mosquitto broker, using the supplied password file generator.

A plain text file could also be used but with lowered security.



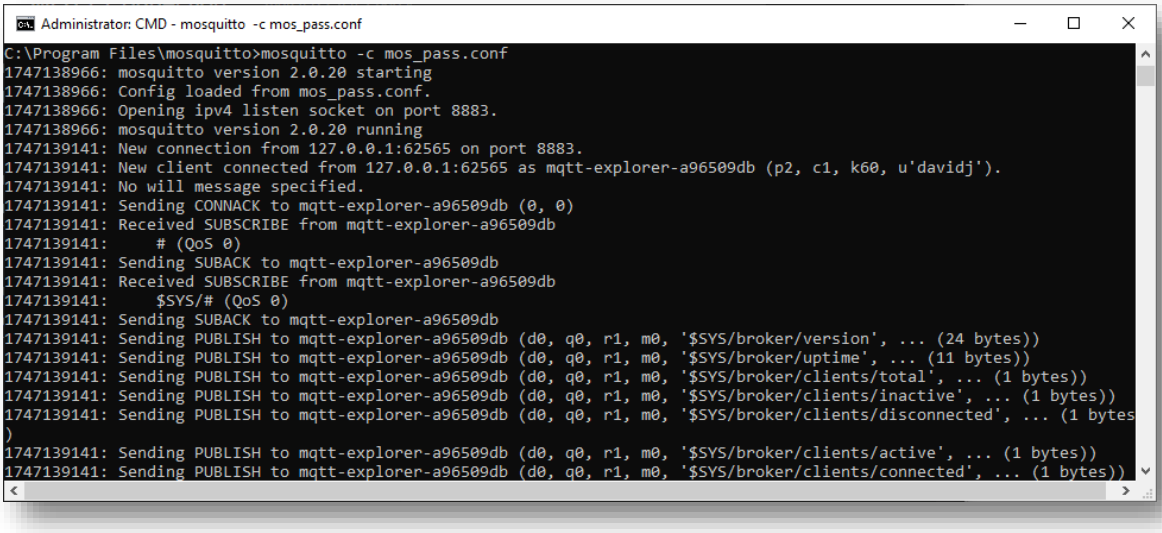
Creating an encrypted password file using mosquito_passwd.exe.

Mosquitto conf file : See Text cut and paste version for the Username and Password example "conf" file for Mosquito



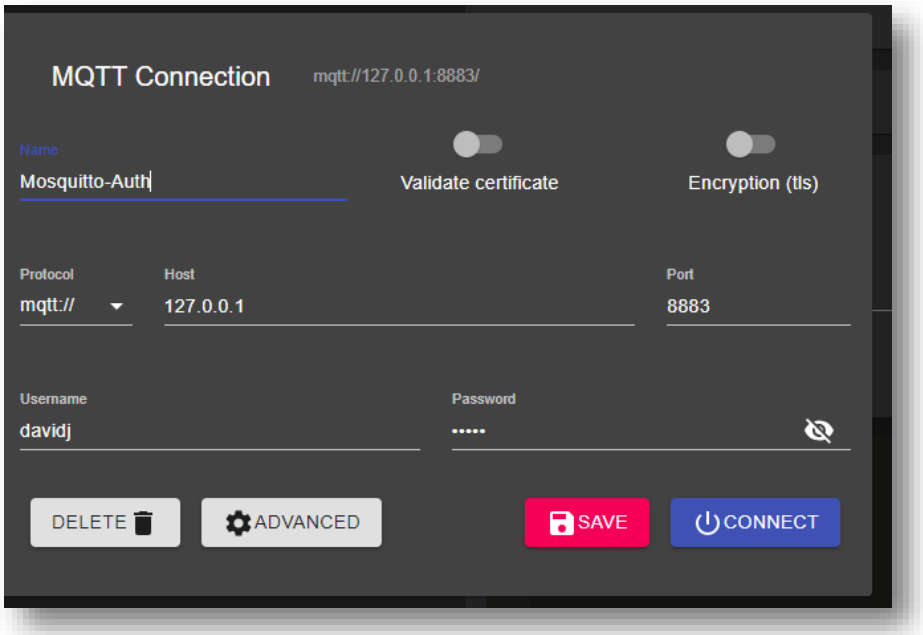
Example of a conf file to use with Mosquitto.exe with Username and Password authentication. See below for a text copy and paste version.

Run : (Administrative command line)
"Mosquitto -c mos_pass.conf"

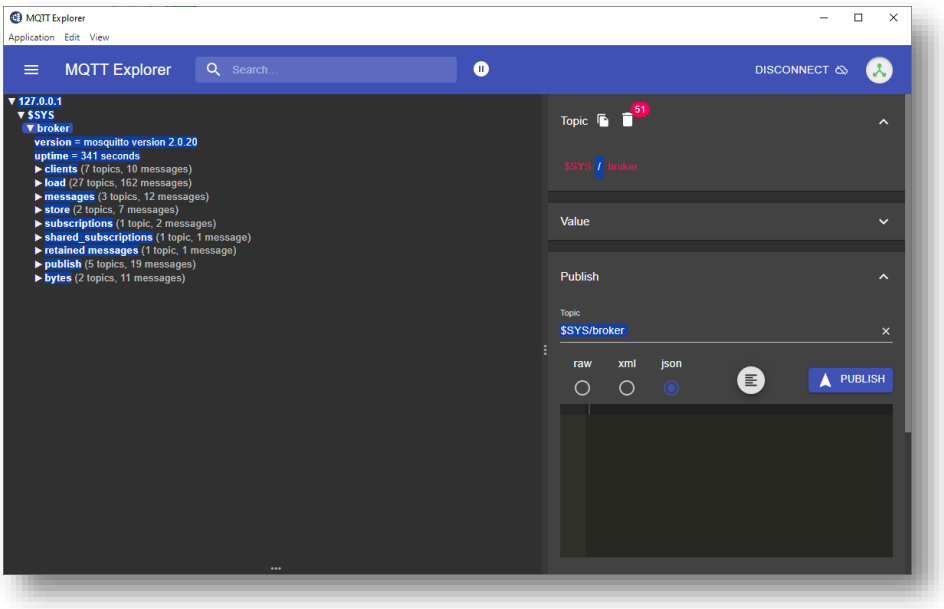


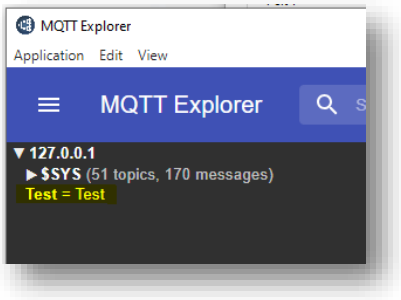
Mosquitto.exe application running in an administrative console showing connectivity using username and password authentication.

Using a 3rd Party broker or the Adroit MQTT driver will give a useful view as a separate client onto the broker, make sure the broker allows multiple connections.

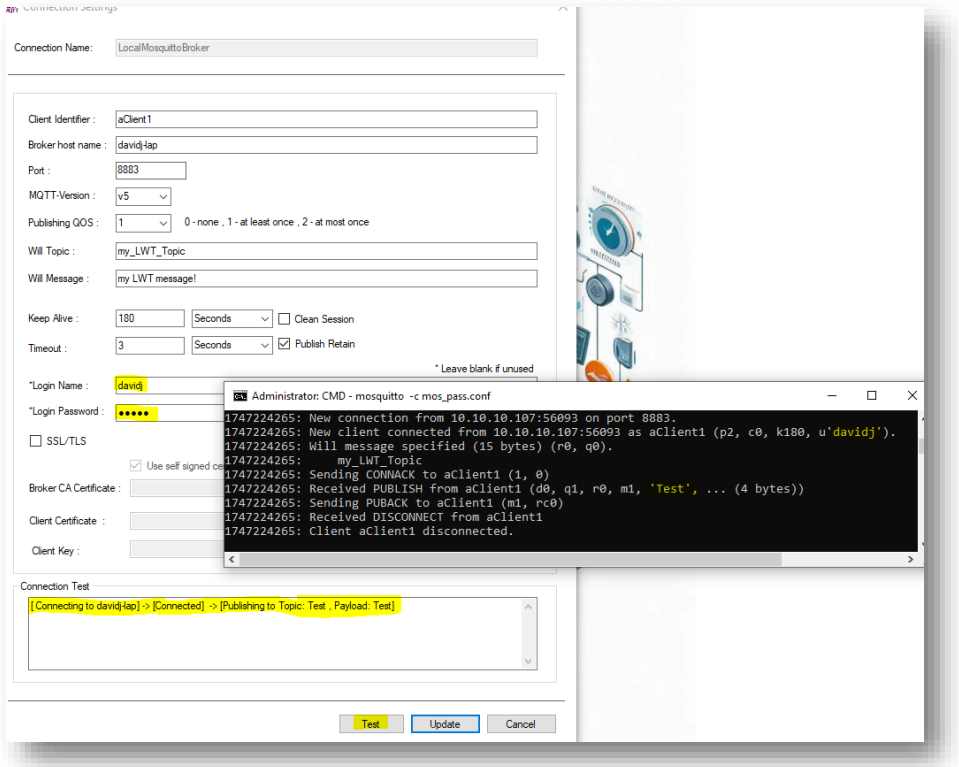


MQTT explorer configuration

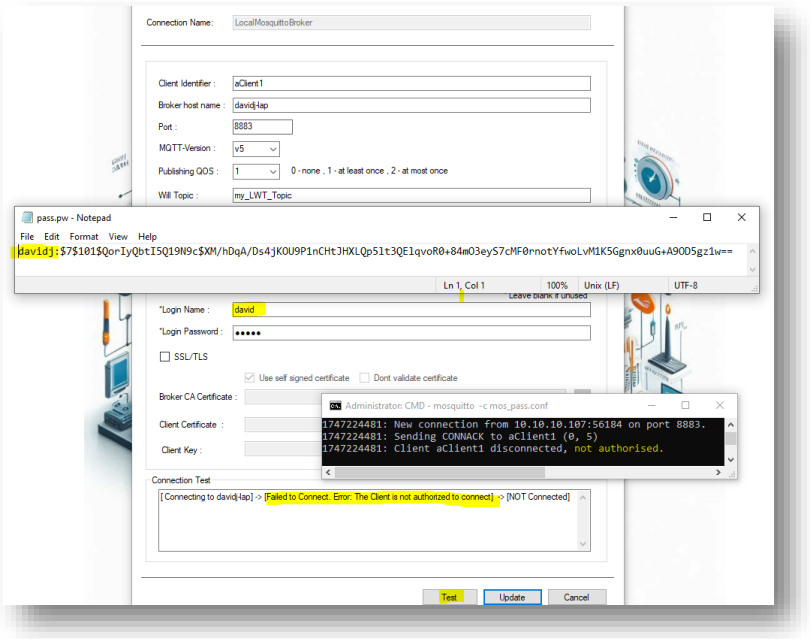




Output of MQTT outputs



Output of Adroit Edge MQTT Microservice successful connection test to the local Mosquitto broker.



Output of Adroit Edge MQTT Microservice failed connection test to the local Mosquitto broker where an incorrect user name was specified.

3.1.1. Text cut and paste version for the Username and Password example "conf" file for Mosquitto.

```
#General.....
per_listener_settings false
allow_zero_length_clientid false
check_retain_source false
max_inflight_bytes 0
max_inflight_messages 20
max_keepalive 65535
max_packet_size 65535
max_queued_bytes 0
max_qos 2
memory_limit 0
persistent_client_expiration 2m
set_tcp_nodelay false
listener 8883
socket_domain ipv4
max_connections -1
protocol MQTT
#logging.....
log_type error
log_type warning
log_type notice
log_type information
log_type debug
#Authentication .....
allow_anonymous false
password_file pass.pw
```

4. Authentication using SSL/TLS

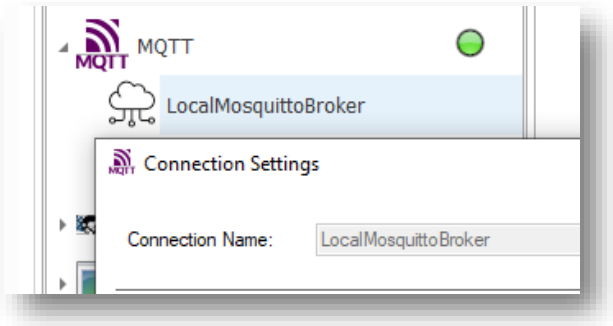
4.1. Example : Using a self-signed certificate with OpenSSL

Tested with Mosquitto, a popular online open source broker.

The Microservice will allow SSL certification via PEM format either as a .crt or a .pem extension, the Microservice will use the server crt, and combine the client key and client crt into another combined PEM formatted file with the prefix "ClientAndKey_<Connectionname>.crt" which is stored in the MQTT/CERTS directory.

The default SSL/TLS port is 8883.

E.g. "ClientAndKey_LocalMosquittoBroker.crt"



Connection name used as a prefix for the concatenated client crt and key certificate

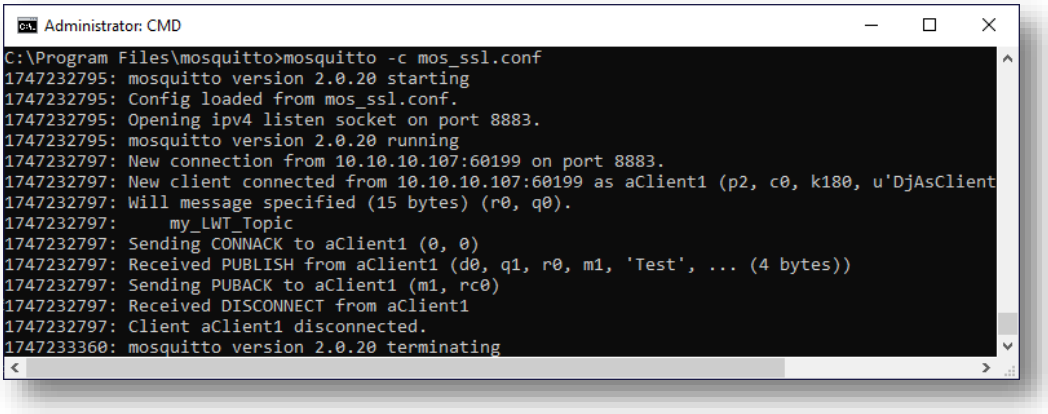
Mosquitto conf file : See Text cut and paste version for the SSL example "conf" file for mosquito

mos_ssl.conf | mos_pass.conf

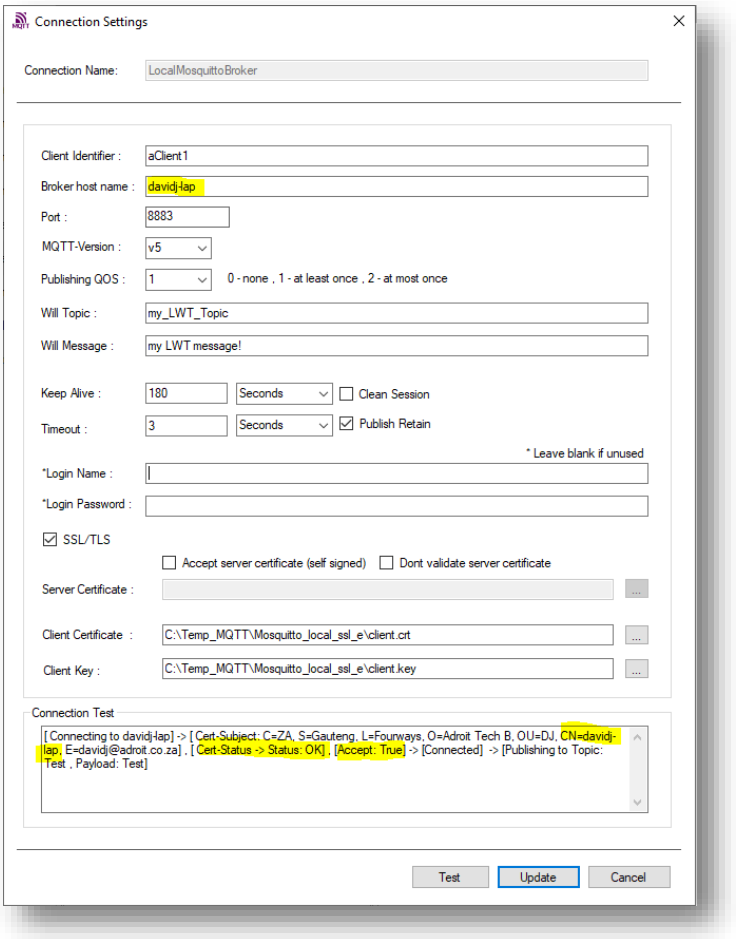
```
#General.....CRLF
per_listener_settings false CRLF
allow_zero_length_clientid false CRLF
check_retain_source false CRLF
max_inflight_bytes 0 CRLF
max_inflight_messages 20 CRLF
max_keepalive 65535 CRLF
max_packet_size 65535 CRLF
max_queued_bytes 0 CRLF
max_qos 2 CRLF
memory_limit 0 CRLF
persistent_client_expiration 2m CRLF
set_tcp_nodelay false CRLF
listener 8883 CRLF
socket_domain ipv4 CRLF
max_connections -1 CRLF
protocol mqtt CRLF
#SSL.....CRLF
certfile c:\temp_MQTT\Mosquitto_local_SSL_e\server.crt CRLF
keyfile c:\temp_MQTT\Mosquitto_local_SSL_e\server.key CRLF
require_certificate true CRLF
cafile c:\temp_MQTT\Mosquitto_local_SSL_e\ca.crt CRLF
use_identity_as_username true CRLF
#logging.....CRLF
log_type error CRLF
log_type warning CRLF
log_type notice CRLF
log_type information CRLF
log_type debug CRLF
```

Example of Mosquitto configuration file for SSL where a CA file is prompted to compare against incoming authentication certificates. And the brokers (server) certificate and key that will be presented to the client.

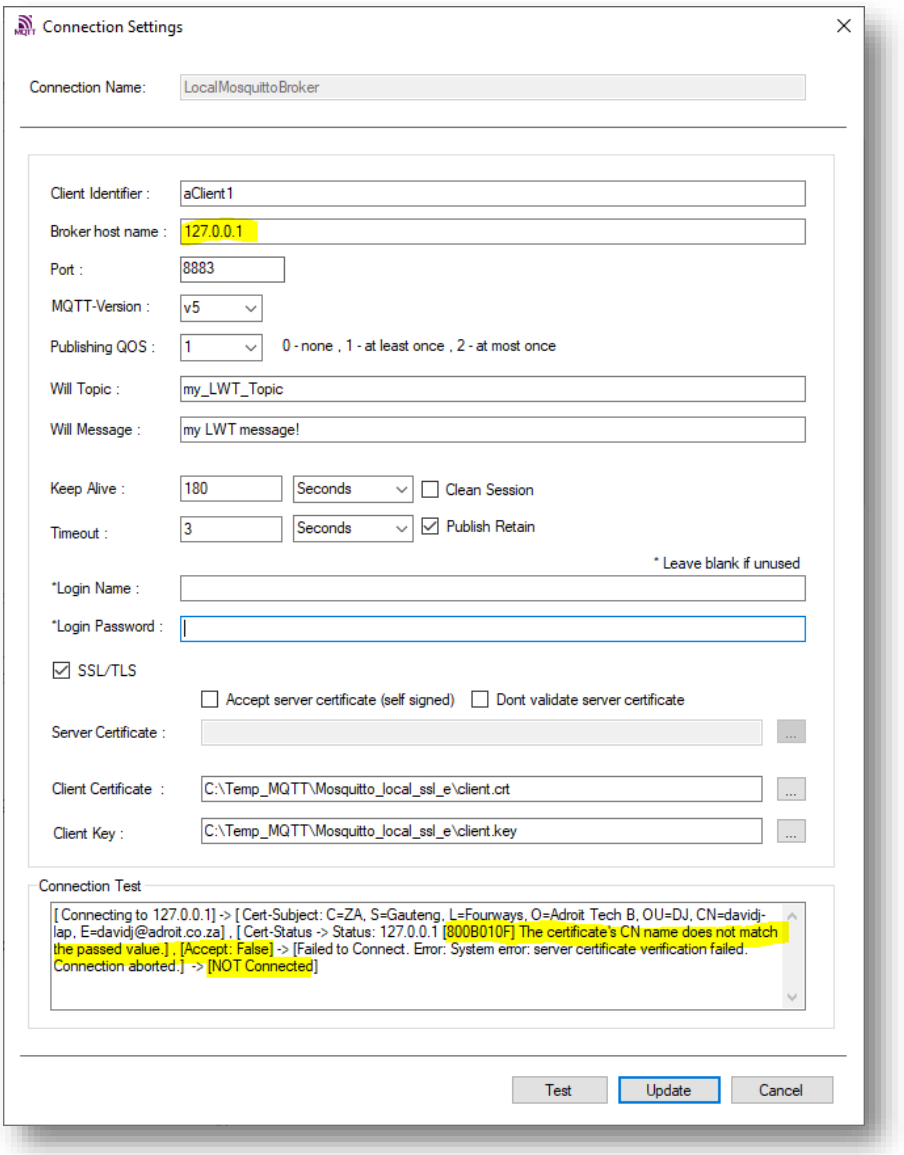
Run : (Administrative command line)
"Mosquitto -c mos_ssl.conf"



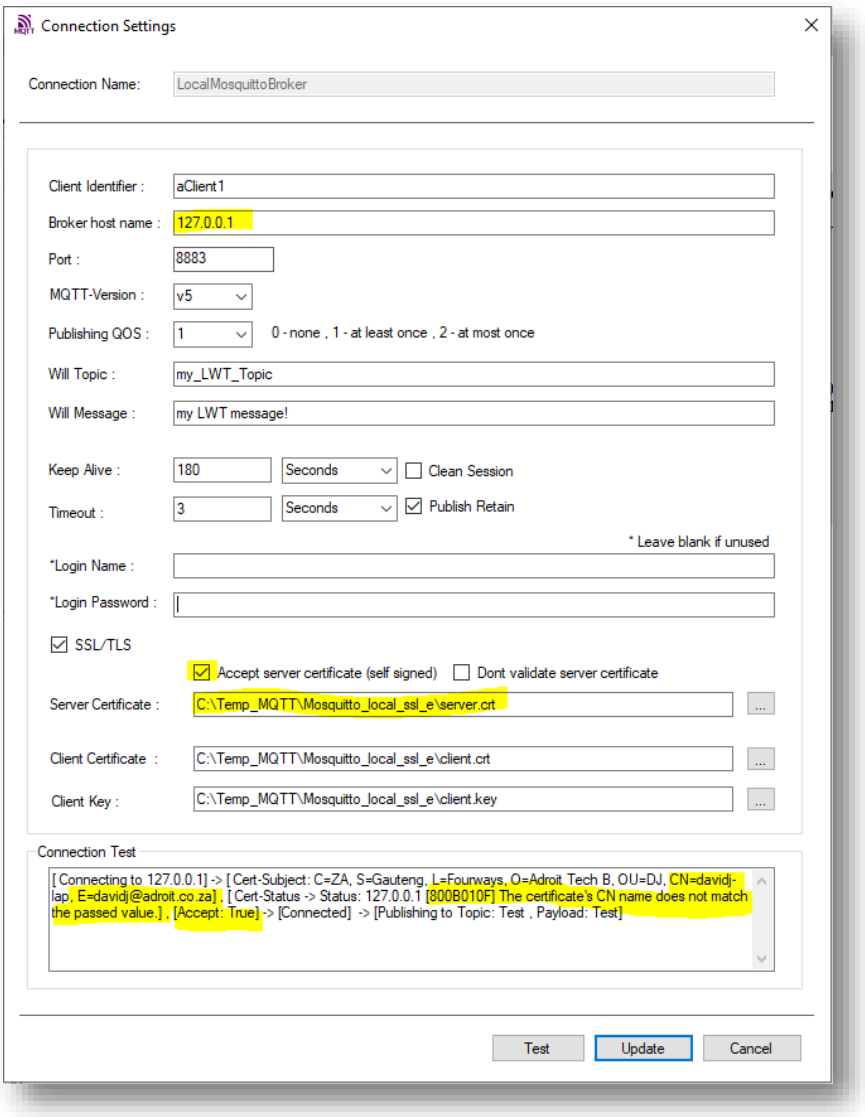
Example of Mosquitto.exe application running in an administrative console showing connectivity using self-signed SSL Authentication.



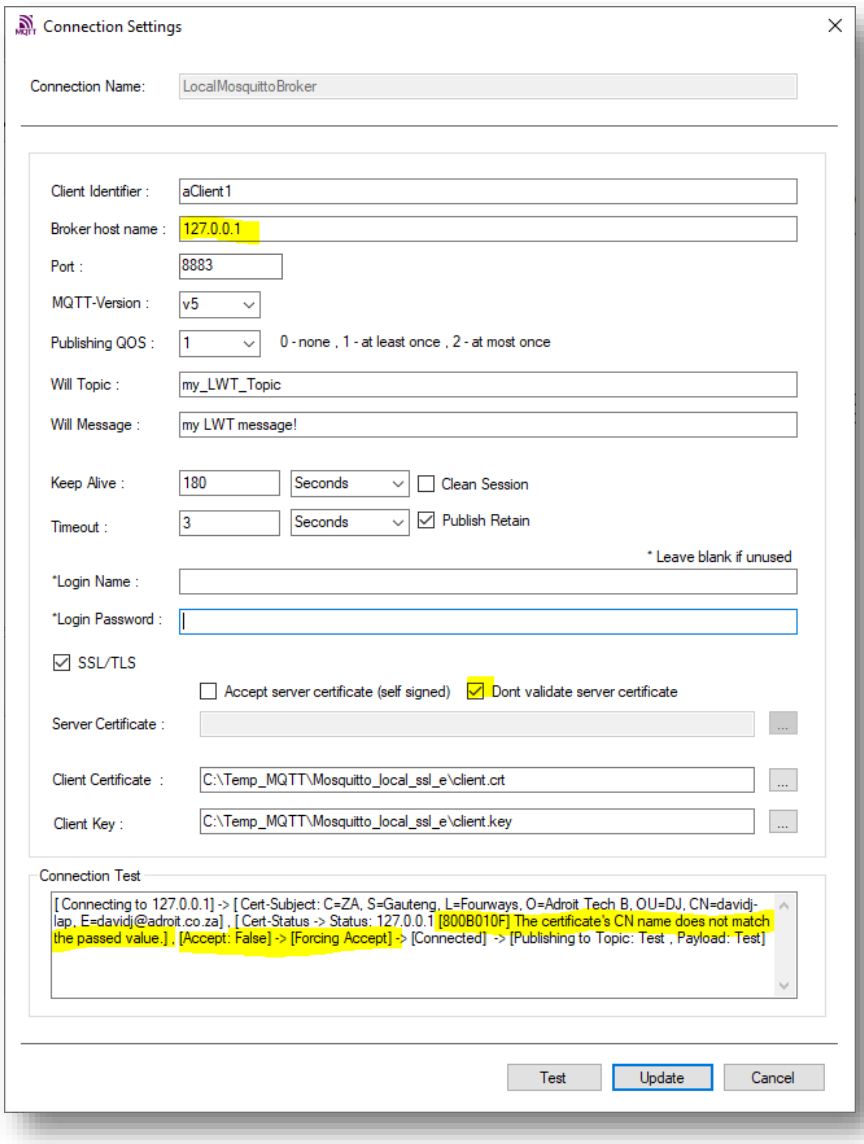
Correctly configured SSL (Broker host name = CN name matching what was used for the signing CA key) – Cert status returns OK.



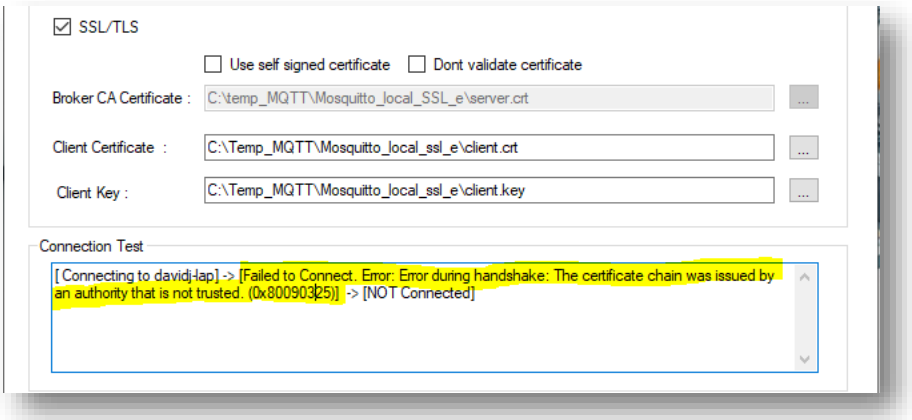
Incorrectly configured SSL (Broker host name ~ CN name mismatched to what was used for the signing key) – where the Client cert presents CN=davidj-lap , we use localhost 127.0.0.1 which causes a mismatch. Error 0x800B010F

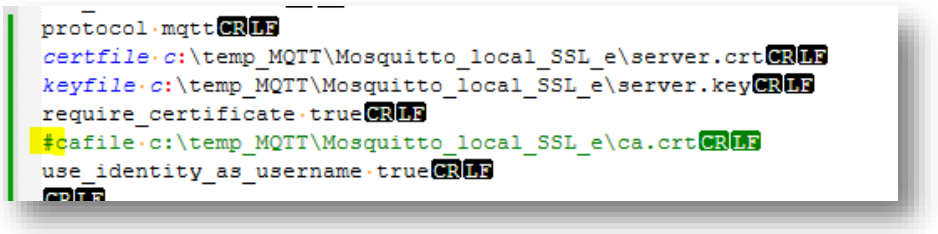


Incorrectly configured SSL (Broker host name ~ CN name mismatched to what was used for the signing CA key) – where the Client cert presents CN=davidj-lap , we use localhost 127.0.0.1 which causes a mismatch. Error 0x800B010F but this time we supply the server CA cert to the Adroit Edge Microservice and instruct the client to use the servers cert via Use self-signed certificate check box, as a comparison during authentication and if the cert provided by the broker matches we accept the connection regardless.



Incorrectly configured SSL (Broker host name ~ CN name mismatched to what was used for the signing CA key) – where the Client cert presents CN=davidj-lap , we use localhost 127.0.0.1 which causes a mismatch.
Error 0x800B010F but this time we instruct the client to not validate the servers cert that is presented during authentication and forced the acceptance of the connection. (Less secure and not recommended)





Error 0x80090325

Removing the CA cert from the Mosquitto setup results in our self-signed certificates being rejected during authorization

If in the case the client certificates are signed by an acceptable CA authority the connection should succeed.

4.1.1. Text cut and paste version for the SSL example "conf" file for Mosquitto

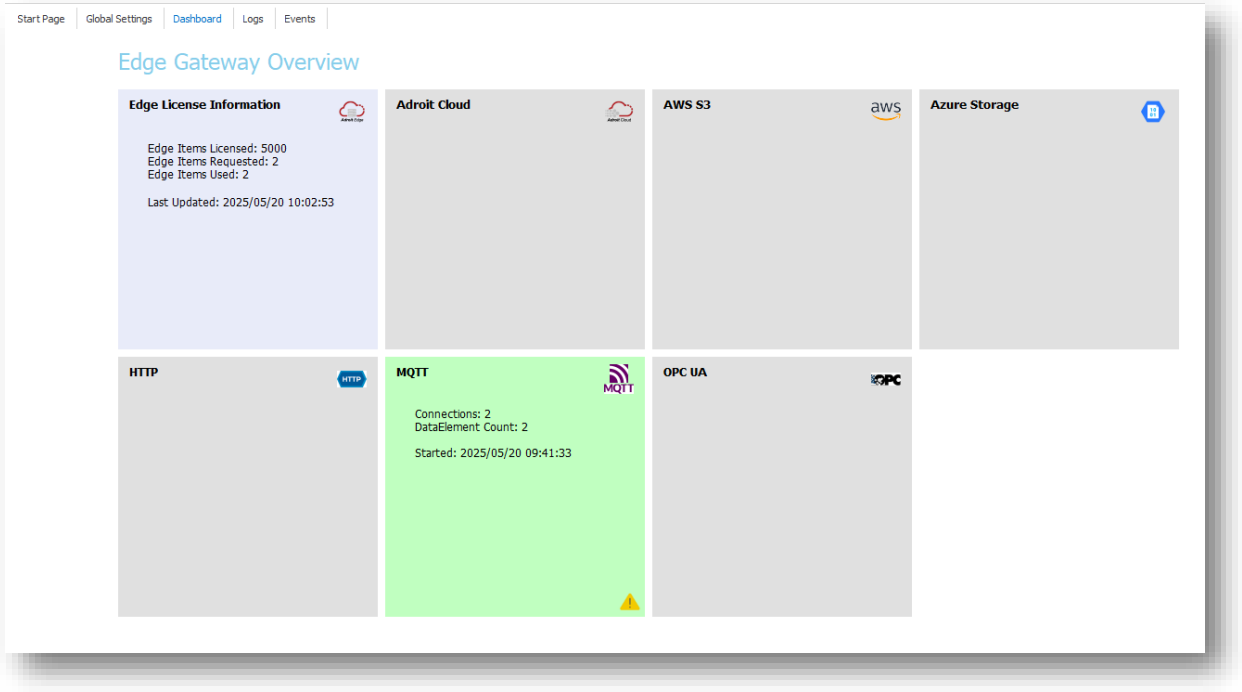
#General.....
per_listener_settings false
allow_zero_length_clientid false
check_retain_source false
max_inflight_bytes 0
max_inflight_messages 20
max_keepalive 65535
max_packet_size 65535
max_queued_bytes 0
max_qos 2
memory_limit 0
persistent_client_expiration 2m
set_tcp_nodelay false
listener 8883
socket_domain ipv4
max_connections -1
protocol mqtt
#SSL.....
certfile c:\temp_MQTT\Mosquitto_local_SSL_e\server.crt
keyfile c:\temp_MQTT\Mosquitto_local_SSL_e\server.key
require_certificate true
cafile c:\temp_MQTT\Mosquitto_local_SSL_e\ca.crt
use_identity_as_username true
#logging.....
log_type error
log_type warning
log_type notice
log_type information
log_type debug

5. Dashboard

This is a real time view of every Microservice running on the system, divided into 2 levels : Overview and Detailed View.

5.1. Overview

Describes a summary of each Microservice including the MQTT Microservice:

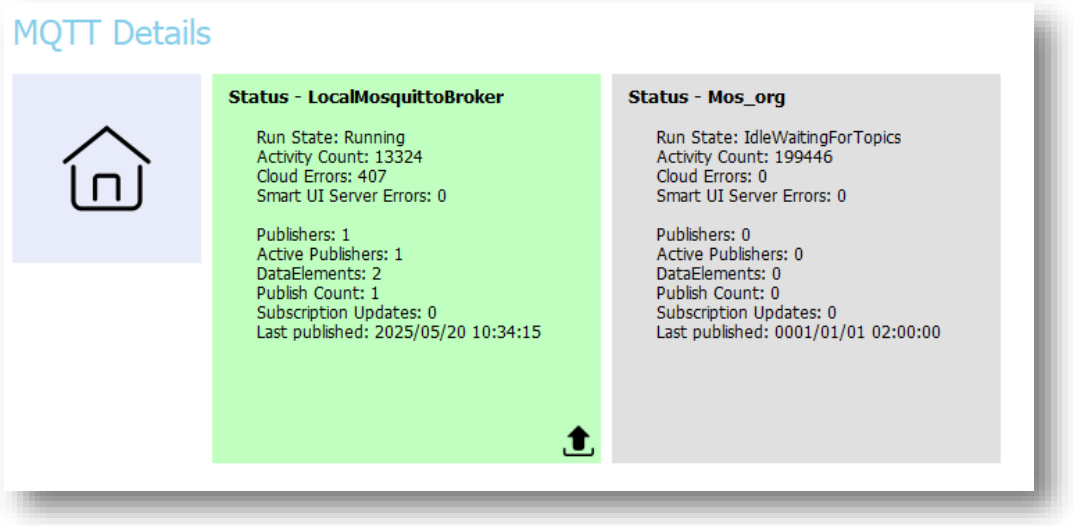


Edge Gateway Overview where a summary of each Microservice running on the system.

Connections - The number of connections that are active in the Microservice.
Start time - The time the Microservice activates.
Stop time - Displayed when the last connection within Microservice stops.
DataElement count - A sum total count of all licensed Dataelements for all underlying connections. (Triggers that do not for part of any payload and not counted)

5.2. Detailed View

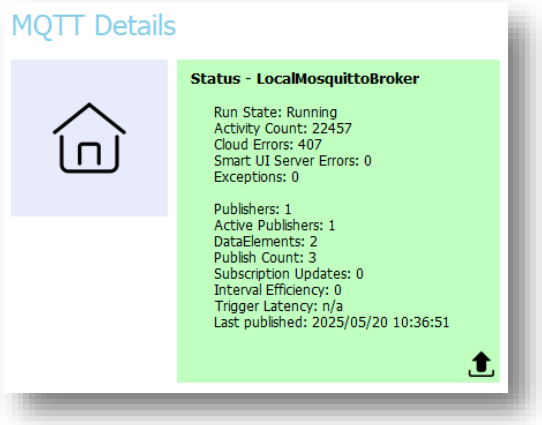
Adds a tile for each MQTT connection and describes more information



MQTT detailed dashboard view , showing 2 configured connections , where the first is publishing , the second is configured but have zero Publishers configured.

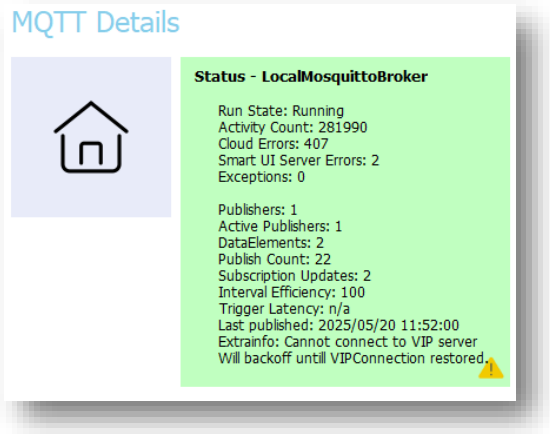
Run state - Connection state , should be "Running"
Activity Count - A increment count indicating connection heartbeat, if it increments its good.
Cloud Errors - Connection error events increment this, should be static under normal running conditions
Smart UI Server Errors - error events increment this, should be static under normal running conditions
Publishers - Count of configured Publishers
Active Publishers - Count of Publishers set to active publish mode
DataElements - Total Sum of all Dataelements that form part of any payload, triggers don't count if these do not form part of any payload.
Publish Count - Total number of publish events to the cloud for this start cycle, online changes of the configuration affecting this connection may zero this count.
Subscription Updates - The total number of subscription updates from the Smart UI Server
Last published - The timestamp for the last publish event.

For MQTT tiles : Extra debug info is obtained if the logging level is set to debug level 4, in the diagnostics dialogue.



Interval Efficiency – This is a 0 to 100, percentage indication of the average percentage of all Publishers achieving the required publish rate, 99 is good, 0 is bad. Publishers in maintenance will affect this average.

Trigger Latency - This is the average trigger delay time in milliseconds. The delay from the time of arrival of the trigger event to the time the Publisher data is sent averaged over all triggered Publishers. Interval Publishers are not part of this calculation. 0 is good.
For MQTT tiles : Some errors can result in the tile displaying addition error information.



Extrainfo – This can show import information regarding the connection , in this case the Smart UI Server was disabled. Note the Error icon

5.3. Tile Colouring

Grey and blank – never started.
Grey tile background means not publishing
Empty – no Publishers
All Publishers in maintenance mode.
Green (overview) - The Microservice is started, a start time and number of connections and Dataelement count should be visible. And if no icons are visible all connections within the Microservice is publishing without error.

5.4. Tile Iconography

Error

An Error condition usually north bound Cloud/connection or south bound VIPServer connectivity issue. Error icons supersede any other icon states and must be resolved to see the others. Alos visible on the overview if any constituent connection is experiencing an error state.

Maintenance

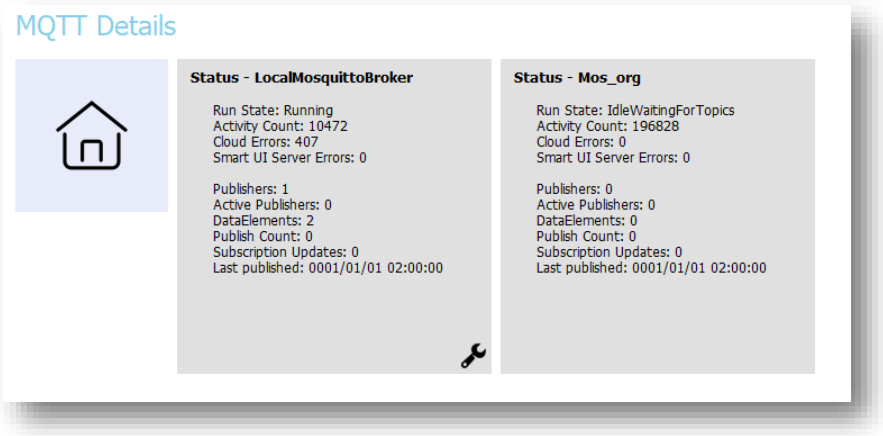
One or more Publishers are in maintenance mode (in one or more connections on the overview tile – removed if all connections are publishing) , this icon supersedes the All-Publishing icon in the detailed view.

All-Publishing

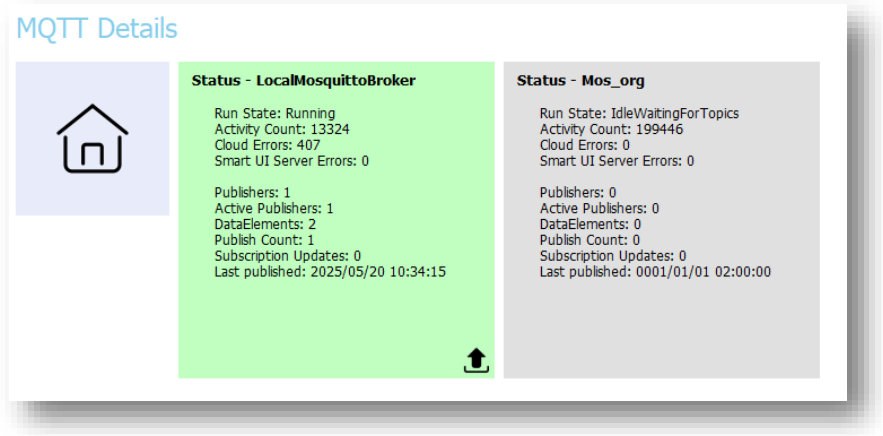
All Publishers on the connection are set to publish, this icon is visible per connection only in the detailed view.

See some examples below:

MQTT detailed dashboard view , both connections inactive (grey) , the first has an Error icon , caused by cloud errors incrementing.



MQTT detailed dashboard view, both connections inactive (grey) , the first has a maintenance icon, indicating one or more Publishers are in maintenance mode, Grey + active Publishers = 0 indicating ALL are in maintenance mode.



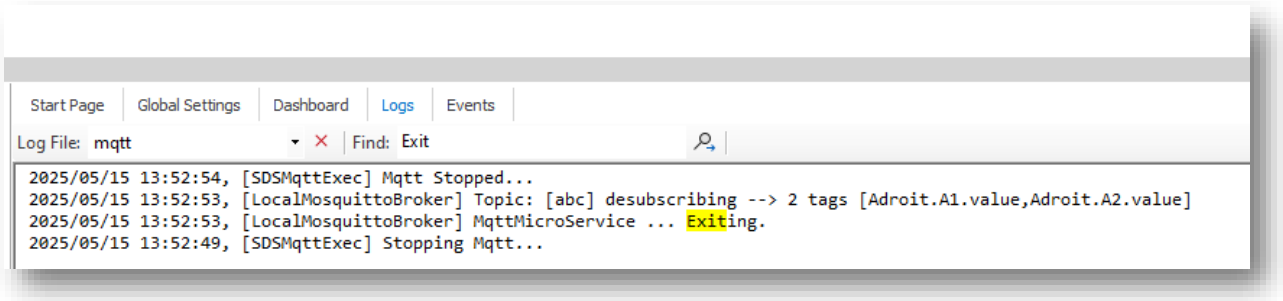
MQTT detailed dashboard view, first connections active (green), the first has a publishing icon meaning it is good.

6. Troubleshooting

The MQTT Microservice provides various tools to help resolve issues:

- **Connection Testing**: The MQTT connection settings includes a live test functionality to verify the connection to the broker.
- **Substitution Validation**: In the MQTT Publishers configuration, the substitution grid highlights invalid Dataelements of a substitution in red for easy identification.
- **Body Validation**: The body configuration features a validation tool to ensure the JSON and XML body is properly formatted, providing detailed feedback on any errors. TEXT and CSV formats do not.
- **Preview and Testing**: The Preview tab allows users to preview the final JSON body with live (last known values) substituted. The MQTT Configurator does not actually send the topic to the broker.
- **Diagnostics**: Access additional tools on the main Edge Gateway tabs for advanced troubleshooting:
  - **Event Log Viewer**: View all Microservice event log messages sourced from the SmartDataServices event log , filters can be applied.
  - **Log File Viewer**: Examine the complete contents of a selected Microservice log file.
  - **Dashboard**: Gain real-time dashboard-like feedback from Microservices, featuring specific Microservices statuses, counters, and indicators, licensing

Diagnostics information logging, including its log level and output destination (e.g., Event log, log file, console), is configured through the Diagnostics dialog for each MQTT connection. The log file itself is a single shared file but each connection will include the connection name.



Logs view , NOTE: the syntax highlighting provided for easy analysis

- **Microservice Console View**: Operate each Microservice as a console application to observe its real-time console messages. How ever console information can also be sent to the log file, and event log. See the diagnostics dialogue.

6.1. Troubleshooting SSL - Error 0x800B010F

The certificate's CN does not match the passed value.

See Section 4, Diagrams referring to "Error 0x800B010F"

Possible resolutions :

1. Supply the correct CN name, used for signing CA key in this case the machine host name : davidj-lap and not the machine IP x.x.x.107 or local host 127.0.0.1
2. Or supply a pre shared CA certificated used to compare during handshaking and set use self-signed certificate
3. Check "Don't validate certificate" which instructs the client to not validate the servers cert that is presented during authentication and forces the acceptance of the connection. (Less secure and not recommended as they true identity of the server connection is in question)

6.2. Troubleshooting SSL - Error 0x80090325

The certificate chain was issued by an authority that is not trusted.

See Section 4, Diagram referring to "Error 0x80090325"

In this case we removed the CA file from the Mosquitto setup causing it to not recognize our signing authority (self-signed)

Possible resolutions :

1. Supply a trusted acceptable CA file to Mosquitto. In our case a certificate signed by our self-signing key.

6.3. Troubleshooting - Failed to connect Error

Error – Failed to connect because the target machine actively refused it.

The certificate chain was issued by an authority that is not trusted.

Possible resolutions :

1. Usually means an incorrect IP address or port was used to a target broker or it the broker is not running. In this case we changed the broker port number

6.4. Troubleshooting - System Error

No such host is known

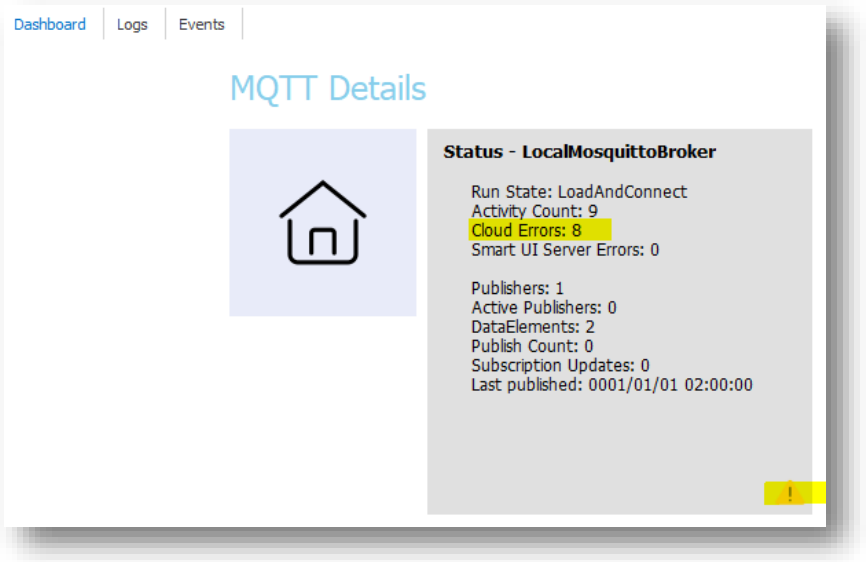
In this case we changed the broker IP to an invalid IP.

Possible resolutions :

1. Usually means the connection could not be completed on the network. In this case we changed the broker IP address.

6.5. Troubleshooting - Dashboard connection startup

Status Run state remains in [LoadAndConnect] cloud errors incrementing



Monitor the Activity count , it displays a count of activity for this broker connection it should never remain static and the tile updates every 5 seconds.
Monitor the Cloud errors , if it is incrementing it means the connection to the configured broker connection is unavailable , the connection state will not progress until this is resolved. (See the event log, log file for possible indications or cause descriptions.)

Possible resolutions:

1. Check connectivity, is the MQTT broker reachable?
    a. Ping the IP address
    b. Try a 3rd Party MQTT browser

In this case the local Mosquitto broker was not running.