MQTT Tools Reference

Test and configure MQTT connections.

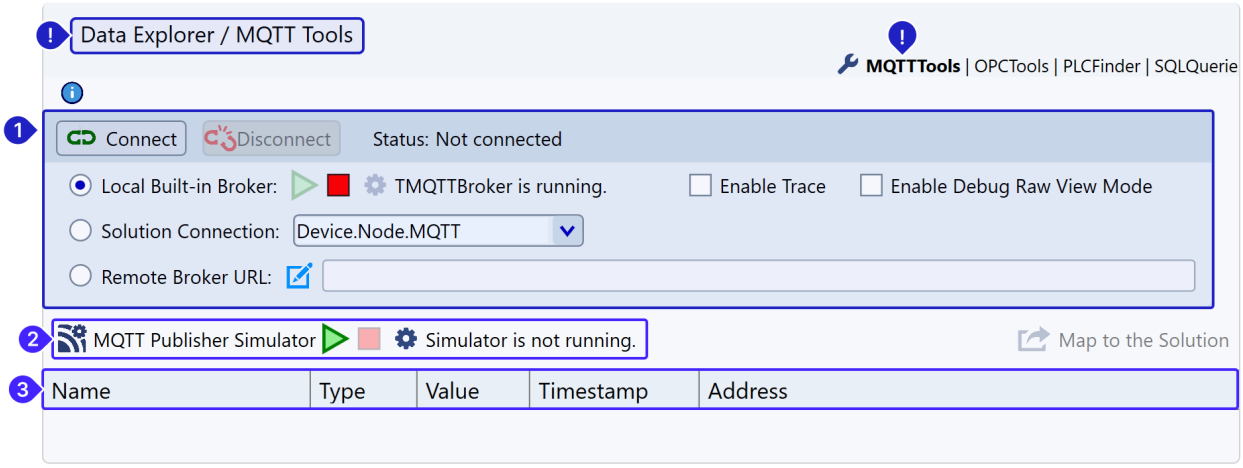
Reference Solution Designer Data Explorer MQTT | OPC | PLC | SQL

MQTT Tools (Reference): shows data from MQTT Broker in its data browsing tree, plain data form. It also correctly parses and format the data from SparkplugB topics.

Connect to a MQTT Broker
MQTT Publisher Simulator
Browsing the Data
Mapping data to Solution
Exporting Data to the Solution

Connect to a MQTT Broker

The User Interface presents the following options to locate the server:



- **Local Built-in Broker**: connects with the internal MQTT broker (the Built-in Broker must be running).
- **Solution Connection**: connects with the MQTT broker used by a device node or a TagProvider.
- **Remote Broker URL**: manually set up the connection settings to connect to an external MQTT broker.

The MQTT Publish Simulator, allows to run a MQTT Client application that publishes data using SparkplugB.

Connect and Disconnect

Select the desired option using the radio buttons and configure the settings based on your selection. Once configured, use the top buttons to connect to the server.

After connecting, if you want to change to another server, use the Disconnect button first.

Refer to Built-in MQTT Broker for information on how to change your default settings for TMQTTBroker and run it as a service.

Local Built-in Broker

This configuration option allows the user to use a built-in MQTT broker that is integrated within the tool.

The button starts the broker, and the button stops it. When you click on the icon next to the command buttons, it opens the **TMQTT Broker Settings** dialog window.

Go to Integrations / MQTT / Built-In MQTT Broker, for details on the execution and configuration of the built-in Broker.

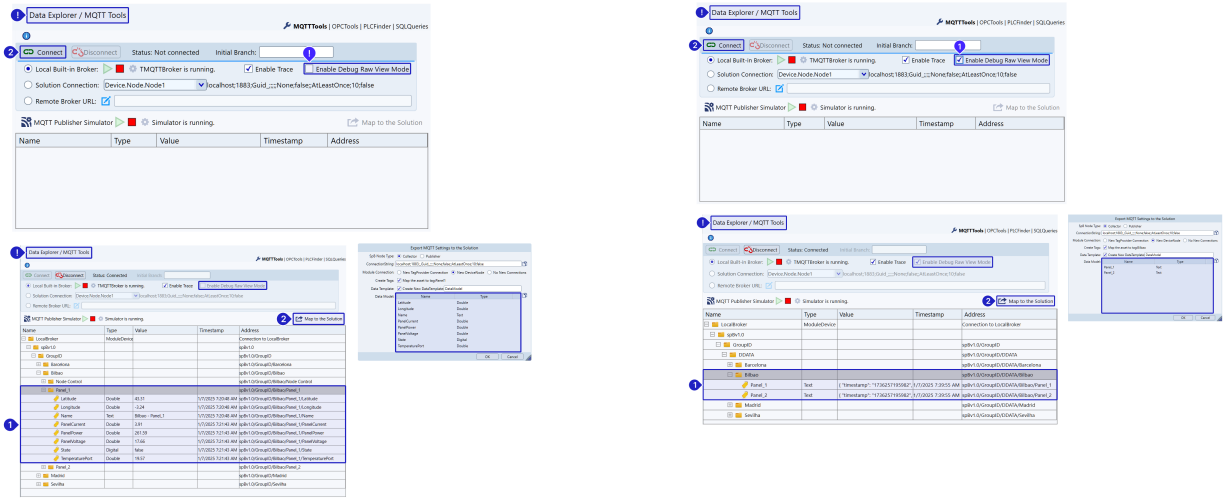
> In most of cases, the default settings are enough. It's not necessary to open the Configuration Dialog, you can just run the Broker directly.
>
> At SolutionCenter, you can setup the Broker to run as Windows Service.

The message "TMQTTBroker is running" confirms that the broker is active.

Additionally, there are two options:

Enable Trace: When enabled, it provides the option to enable tracing of MQTT messages. It logs detailed information about MQTT messages, including their content, source, destination, and timestamps.

Enable Raw View Mode: When selected, it displays the raw data of MQTT messages without any formatting or parsing, providing a clear view of the actual transmitted data. It presents raw data and low-level information for debugging and shows unprocessed data streams, memory contents, register values, and network packets that facilitate in-depth troubleshooting.



Solution Connections

Within the Solution Configuration, there are two location you can link to MQTT Brokers:

- Using the Device Module, Channels, selecting an MQTT protocol.
- Using the TagProvider connections, to dynamic use the data from remote MQTT Brokers.

To speed up the testing of these connections, the Data Explorer tool allows you to easily select the settings and explore the connection and data of the MQTT broker.

Just select the configuration you want to test, and connect to it.

> If the Radio Button for Solution Connections is disabled, it means the solution doesn't have any DeviceChannel or TagProvider using MQTT Brokers.

Remote Broker URL

To connect using the Remote Broker URL option, start by entering the broker's IP address or domain name in the specified field. Click the  icon next to this field to access the **MQTT Broker Connection** dialog. In this dialog:

1. Specify the port number, typically 1883.
2. Assign a unique ClientID to identify the client.
3. If authentication is required, input the username and password.
4. For secure connections, provide the path to the SSL certificate file and its password.
5. Select the network security protocol, such as SSL or TLS 1.2, to ensure encrypted communication.
6. Enable WebSocket connections if traditional MQTT ports are blocked.
7. Choose the Quality of Service (QoS) level based on the desired message delivery guarantee.
8. Configure the KeepAlive interval to maintain an open connection during inactivity.

The following tables has the complete set of fields available when setting up a Broker connection.

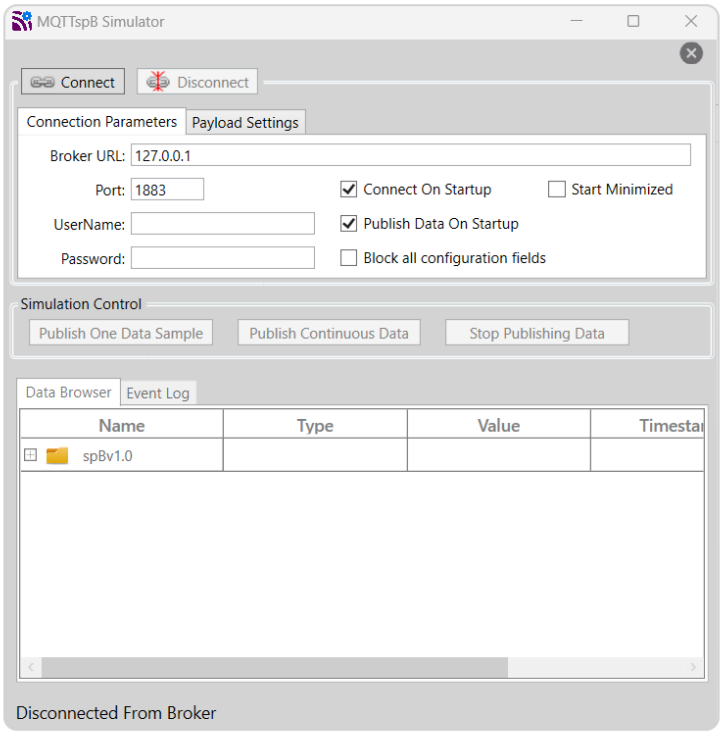
Broker Connection Configuration Settings	
Field	**Description**
Broker URL	Specifies the address of the MQTT broker. This is typically an IP address or a domain name (e.g., localhost).

Port	Indicates the port number used for the connection. The default MQTT port is 1883.
ClientID	A unique identifier for the client connecting to the broker. This ensures that each client can be uniquely identified.
Username	(Optional) username required for authentication with the broker.
Password	(Optional) password required for authentication with the broker.
Certificate File	Path to the SSL certificate file used for secure connections. This is important for encrypted communication.
CertificatePassword	Password for the SSL certificate file, ensuring secure access to the certificate.
Network Security	Specifies the security protocol used (e.g., None, SSL, TLS). Determines the encryption level for the connection. Network security options for MQTT brokers include: **None**: No encryption or protocols are applied. This option is suitable for secure or isolated networks but risky in public settings. **TLS 1.0**: An early TLS version that enhances SSL 3.0 security. However, it is now outdated and vulnerable to attacks like POODLE and BEAST. It is generally discouraged. **TLS 1.1**: An improvement over TLS 1.0, addressing initialization vector issues. However, it remains outdated and susceptible to attacks, with limited support for new standards. **TLS 1.2**: Offers significant security upgrades, supporting modern cryptographic algorithms like AEAD ciphers. It ensures data confidentiality and integrity. TLS 1.2 is the recommended secure protocol for network communications until TLS 1.3 sees wider adoption.
WebSocket	Checkbox option to enable WebSocket connections. This is useful for scenarios where traditional MQTT ports are blocked.
QoS	The Quality of Service (QoS) levels in MQTT define the guarantee of message delivery between the sender and the receiver. The available QoS levels are: **AtMostOnce**: Ensures that each message is delivered at least once to the receiver. This level may result in duplicate messages but guarantees that no message is lost. **AtLeastOnce**: Guarantees that a message is delivered at most once, meaning it might not be delivered at all. This level is suitable for scenarios where occasional message loss is acceptable but duplicate messages are not. **ExactlyOnce**: Ensures that each message is delivered exactly once. This is the highest level of service, preventing both message loss and duplication, but it incurs the highest overhead.
KeepAlive (sec)	Time interval in seconds to keep the connection alive. Ensures that the connection remains open even if no messages are sent.

MQTT Publisher Simulator

This tool allows users to simulate an MQTT Publisher device integrated into the platform.

The  button starts the Simulator, and the  button stops it. Clicking on the green play button starts the execution the simulator. When doing so the **MQTTspB Simulator** window shows, closing that window will cease the execution the simulator. You can minimize if you want to keep the simulation in background.



When you click on the  icon next to the command buttons, it opens the **MQTTspB Simulator Configuration** dialog which allows to custom the Simulator data model and execution behaviors.

Go to Integrations / MQTT / SparkplugB Simulator, for detailed information for customization the Simulator Settings.

> In many cases, the default settings are enough. It's not necessary to open the Configuration Dialog, you can just run the Simulator directly.

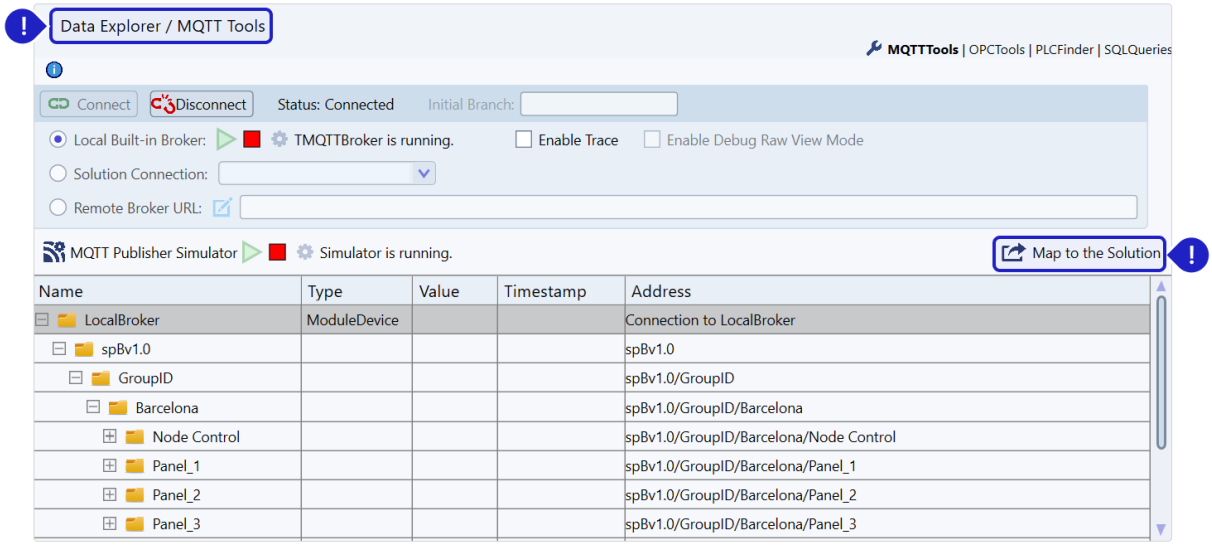
Browsing the Data

The table in the MQTT Tools displays the hierarchical organization and current state of various parameters. It includes columns for each parameter's name, type, value, timestamp, and address, allowing users to browse the asset model and view the data model and its organization. Each row represents a specific data point within the hierarchy, providing details on its current state and location within the MQTT topic structure, facilitating effective monitoring and management of the system. Here is a brief description and explanation of each column in the table:

- **Name**: This column lists the names of various entities and parameters in the simulation. It includes hierarchical names that represent groups, panels, and specific parameters within those panels. Examples include "LocalBroker," "spBv1.0," "Cities," "Barcelona," "Panel_1," and specific data points like "Latitude," "Power," etc.
- **Type**: This column indicates the data type of each parameter listed in the "Name" column. Common types shown include "ModuleDevice," "Double," "Text," and "Digital." These types describe the nature of the data, such as numerical values (Double), text strings (Text), and binary states (Digital).
- **Value**: This column displays the current value of each parameter at the given timestamp. The values can be numerical, textual, or binary. For example, "37.36" for Latitude, "Sevilha - Panel1" for Name, and "true" for State.
- **Timestamp**: This column shows the exact date and time when the value of each parameter was recorded. This is useful for tracking the time-specific states and readings of the various parameters. The format used is "M/D/YYYY H:MM AM/PM."
- **Address**: This column provides the full address or path within the MQTT topic hierarchy for each parameter. This path helps locate the specific data points within the overall structure of the simulation. Examples include "spBv1.0/Cities/Sevilha/Panel_1/Latitude" and "spBv1.0 /Cities/Sevilha/Panel_1/Power."

Mapping data to Solution

After a user has already connected to the MQTT broker and initialized the MQTT publisher simulator.



It will enable Map to the Solution button on the left-corner of the data table, clicking this button will open the Export MQTT Settings to the Solution dialog.

Exporting Data to the Solution

After connecting to an MQTT broker and browsing the topic tree, the toolbar above the data table provides actions to export the discovered data into the solution configuration. There are two separate export paths, plus tree navigation controls:

Toolbar Button	Description
Add TagProvider Link	Creates a dynamic TagProvider connection. Data is discovered and accessed at runtime without creating local tags. Best for monitoring and asset navigation scenarios.
Add to Device Module	Creates a Device Channel, Node, and optionally Tags and a UserType. Best for applications that need local tags with alarms, historian, or scripting.
Expand	Expands all nodes in the data browser tree.
Collapse	Collapses all nodes in the data browser tree.

Add TagProvider Link

Click the **Add TagProvider Link** button to open the **Add TagProvider Linked Folder to the UNS** dialog. This creates a TagProvider connection that makes the broker data available as dynamic tags in the Unified Namespace, without creating individual local tags.

Field	Description
ConnectionString	Auto-populated from the current broker connection parameters (address, port, client ID, security settings).
Linked Folder	The broker branch to link is the node selected in the data browser tree (its parent when the node has no children). The dialog creates a folder named after the connection at the root of the Asset Tree, if it does not exist yet, and inside it a folder of the same name linked to that branch. A linked folder never sits at the root. With no node selected, no Asset Tree folder is created.
UserType	When a SparkplugB DeviceID node is selected in the tree, the dialog offers a **Create New DataTemplate** checkbox. When checked, it creates a UserType with members matching the variables discovered on that device node. The Data Model table shows the inferred variable names and types (e.g., Current/Double, Latitude/Double, Name/Text, State/Digital). This is optional for TagProvider connections.

> ***Tip:*** *The DataTemplate option only appears when a leaf-level device node (DeviceID) is selected in the tree. When a branch node (GroupID or NodeID) is selected, only ConnectionString and Linked Folder are shown.*

Add to Device Module

Click the **Add to Device Module** button to open the **Add Module Device mapping to the Solution** dialog. This creates a full Device Module integration: a Channel with the MQTTspB protocol, a Node with the broker connection, and optionally Tags and a DataTemplate.

Field	Description
SpB Node Type	Defines how the channel operates when connecting to the MQTT Broker. **Collector** (default): acts as a primary MQTT client that subscribes to data and can publish commands. **Publisher**: acts as an edge node that publishes data to the broker.
ConnectionString	Auto-populated from the current broker connection parameters.
Module Connection	Creates a Device Node linked to the MQTT source, placing it within the solution's device structure (Devices Channel + Devices Node).
Create Tags	When checked, creates local tags in the Unified Namespace mapped to the selected MQTT topic. The tag name is derived from the selected node (e.g., "Map the topic to tag:Panel1").
UserType	When checked, creates a new UserType with members matching the variables discovered on the selected device node. The Data Model table shows the inferred names and types. This is required for SparkplugB device mapping, since the MQTTspB protocol does not support simple tags directly in Devices Points — a UserType with the proper structure must be used.

> ***Note:*** *For SparkplugB data, the **Add to Device Module** path with **Create Tags** and **UserType** enabled is the recommended approach when you need local tags with alarms, historian, or scripting. The UserType is mandatory because the MQTTspB protocol requires structured data models for device point mapping.*

In this section...
