MQTT

Contents

MQTT Driver

MQTT MQ Telemetry Transport (MQTT) (version 1.7, last revised January 23 2019).

The MQTT driver enables communication between the Studio system and remote devices using the MQ Telemetry Transport (MQTT) protocol, according to the specifications discussed in this document.

This document assumes that you have read the "Development Environment" section in the main Studio documentation.

This document also assumes that you are familiar with the Microsoft Windows 7/8/10 environment. If you are not familiar with Windows, then we suggest using the **Help and Support** feature (available from the Windows **Start** menu) as you work through this document.

Driver specifications

This section identifies all of the software and hardware components required to implement communication between the MQTT driver in Studio and remote devices using the MQ Telemetry Transport (MQTT) protocol.

Driver files

The MQTT driver package comprises the following files, which are automatically installed in the `Drv` folder of the Studio application directory:

- `MQTT.DLL`: Compiled driver.
- `MQTT.INI`: Internal driver file. *You must not modify this file.*
- `MQTT.MSG`: Internal driver file defining error messages for the possible error codes. (These error codes are described in detail in the Troubleshooting section.) *You must not modify this file.*
- `MQTT.PDF`: This document, which provides complete information about using the driver.

> **Note:** You must use a compatible PDF reader to view the `MQTT.PDF` file. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's website.

You can use the MQTT driver on the following operating systems:

- Windows Desktop
- Windows Server
- Windows Embedded
- Linux platforms

Device specifications

To establish communication, your target device must meet the following specifications:

- Manufacturer: Multiple
- Compatible Equipment: Multiple
- Programmer Software: Not applicable

Network specifications

To establish communication, your device network must meet the following specifications:

- Device Communication Port: 1883 or 8883 for authentication using SSL
- Physical Protocol: Ethernet and others
- Logic Protocol: MQ Telemetry Transport (MQTT)
- Device Runtime Software: Not applicable
- Specific PC Board: Not applicable
- Cable Wiring Scheme: Not applicable

Additional specifications

This driver also requires that the library file `paho-mqtt3a.dll` (used by MQTT v1.3 or lower) or the `pahomqtt3as.dll` (used by MQTT v1.4 or higher) be properly installed in the Drv sub-folder of the Studio program folder, at `Drv\API\`. If you downloaded and installed this driver separately, some time after you installed Studio, the library file should be automatically installed in the correct location. If it is not, or if it is not correctly named, this driver will return error 1 (Error loading api).

The MQTT driver v1.4 and higher also requires the library files ssleay32.dll and libeay32.dll and they should be properly installed in the Bin sub-folder of the Studio program folder. If you downloaded and installed this driver v1.4 or higher separately, sometime after you installed Studio, the library file should be automatically installed in the correct location. The driver used on IoTView depends on the files libcrypto.so.1.0.0 and libssl.so.1.0.0 found in the Bin subfolder of the linux distribution subfolder under Redist/IoTView.

MQTT Driver supports authentication and secure connections, to use these features the user must first use the Web Studio to configure the authentication information or SSL/TLS certificates for more information please refer to the sections "Driver communication using Authentication", "Driver communication using Certificates and Secure Connections (SSL/TLS)", and "Driver communication using Certificates and Secure Connections (SSL/TLS) - Self Signed Certificates".

To use the MQTT driver version 1.4 or higher on Windows Embedded platform using Remote Management, the latest version of the Microsoft Visual C++ 2008 Redistributable has to be installed separately on the device.

Conformance testing

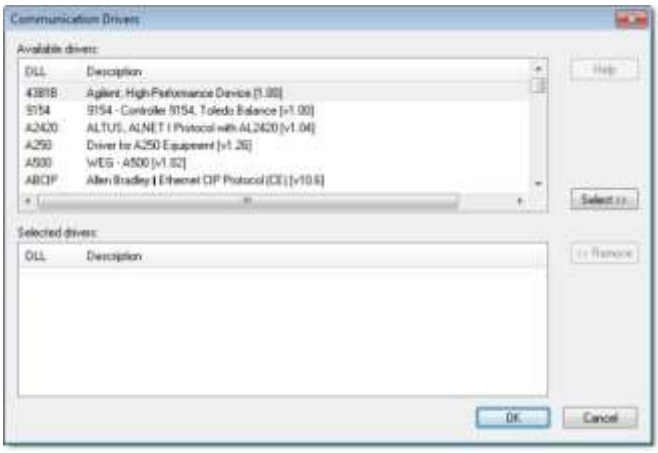
The following hardware/software was used for conformance testing:

- Driver Version: 1.7
- Studio Version: 8.1 SP2
- Operating System (development): Windows 7/8
- Operating System (target): Windows Desktop/Server/Embedded Standard, Linux platforms
- Equipment: Not applicable
- Communication Settings:
- Not applicable
- Cable: Not applicable

Adding a communication driver to your project

This section explains how to add a communication driver to your project.

1. On the **Insert** tab of the ribbon, in the **Communication** group, click **Add/Remove Driver**.
   The *Communication Drivers* dialog is displayed.



Communication Drivers dialog

2. In the *Available drivers* list, click the communication driver that you want to add.
3. Click **Select**.
   The driver is added to the *Selected drivers* list.
4. Click **OK**.
   The *Communication Drivers* dialog is closed and the selected driver is inserted in the **Drivers** folder in the Project Explorer.

Configuring the driver's communication settings

This section explains how to configure the communication settings for the driver.

You must add the communication driver to your project before you can configure its settings. For more information, see Adding a communication driver to your project on page 7.

The general procedure for configuring a driver's communication settings is the same for all drivers. However, the specific settings are different for each driver, depending on the options and protocols used by the target device.

To configure the communication settings:

1. In the **Comm** tab of the Project Explorer, expand the **Drivers** folder.
   The folder contains the drivers that are currently enabled. If you do not see the driver that you want to configure, then you need to add it.
2. Right-click the driver that you want to configure, and then click **Settings** on the shortcut menu.
   The *Communication Settings* dialog is displayed.

Communication Settings: MQTT dialog

3. Configure the remaining, driver-specific settings as needed.

Driver-specific communication settings

Setting	Default Value	Valid Values	Description
Keep Alive (seconds)	10	Unsigned Integer	The "keep alive" interval, measured in seconds, defines the maximum time that should pass without communication between the client and the server The client will ensure that at least one message travels across the network within each keep alive period. In the absence of a data-related message during the time period, the client sends a very small MQTT "ping" message, which the server will acknowledge. The keep alive interval enables the client to detect when the server is no longer available without having to wait for the long TCP/IP timeout. Set to 0 if you do not want any keep alive processing.

Setting	Default Value	Valid Values	Description
Client ID	Computer name	ASCII	The Client Identifier identifies the Client to the Server. Each Client connecting to the Server has a unique Client ID. The Client ID MUST be a unique (not empty) string.

4. Click **OK**.

The settings are saved and the *Communication Settings* dialog is closed.

About driver worksheets

Like the other parts of your project, communication with remote devices is controlled by worksheets. This section explains how to add worksheets to your project and then configure them to associate project tags with device registers.

Each selected driver includes a Main Driver Sheet (MDS) and one or more Standard Driver Sheets (SDS). The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. In contrast, Standard Driver Sheets can be inserted to define tag/ register associations that are triggered by specific project behaviors.

The configuration of these worksheets is described in detail in the "Communication" chapter of the *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only MQTT driver-specific parameters and procedures are discussed here.

Adding and configuring a Standard Driver Sheet

By default, a communication driver does not include any Standard Driver Sheets. This section explains how to add a Standard Driver Sheet to your project and then configure it.

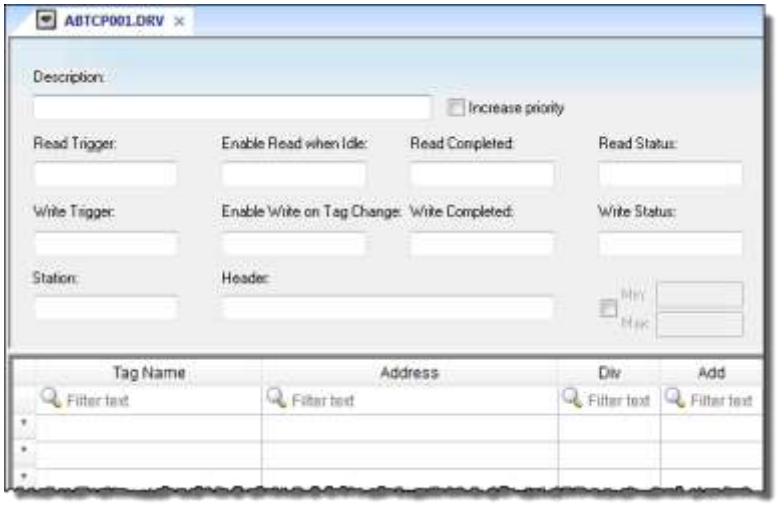
The MQTT driver must be added to the project before you can configure any of its worksheets. For more information, see Adding a communication driver to your project on page 7.

Standard Driver Sheets can be inserted to define additional tag/register associations that are triggered by specific project behaviors.

> **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the “Communication” chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the MQTT driver.

1. Do one of the following.
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **Driver Sheet** and then select **MQTT** from the list.
   - In the **Comm** tab of the Project Explorer, right-click the **MQTT** folder and click **Insert** on the shortcut menu.

   A new MQTT driver worksheet is inserted into the **MQTT** folder, and then it is automatically opened for configuring.



Standard Driver Sheet

> **Note:** Worksheets are numbered in order of creation, so the first worksheet is `MQTT001.drv`.

2. Configure the Station and Header fields as described below.

Station

Specify the cloud URL to connect with the third-party MQTT broker and the custom directories, according to the following syntax:

<IP Address or URL>[:Port Number][:QOS][:R]

Where:

IP Address or URL

IP address of the third-party MQTT broker (Server). (This field is mandatoty.). For secure connection, please use the prefix `ssl://`. For example: `ssl://iot.eclipse.org:8883:0:0`

Port Number

TCP port number used to connect with the third-party MQTT broker (Server). If omitted, the default Port number (1883) is used by the driver. For most brokers using SSL the port used is 8883.

QOS

Quality of Service number (0, 1, or 2). If omitted, the default QOS (1) is used by the driver. The following table describes the difference between the possible QOS options:

Value	Quality	Description
0	At most once	The message is delivered at most once, or it may not be delivered at all. Its delivery across the network is not acknowledged. The message is not stored. The message could be lost if the client is disconnected, or if the server fails. QoS0 is the fastest mode of transfer. It is sometimes called "fire and forget". The MQTT protocol does not require servers to forward publications at QoS0 to a client. If the client is disconnected at the time the server receives the publication, the publication might be discarded, depending on the server implementation.
1	At least once	The message is always delivered by the publisher at least once. It might be delivered multiple times if there is a failure before an acknowledgment is received by the sender. The message must be stored locally at the sender, until the sender receives confirmation that the message has been published by the receiver. The message is stored in case the message must be sent again.

2	Exactly once	The message is always delivered by the publisher exactly once. The message must be stored locally at the sender, until the sender receives confirmation that the message has been published by the receiver. The message is stored in case the message must be sent again. QoS2 is the safest, but slowest mode of transfer. A more sophisticated handshaking and acknowledgement sequence is used than for QoS1 to ensure no duplication of messages occurs.

R

Indicates that the server (broker) must retain the message sent by the MQTT driver (publisher). In this case, even if a third-party subscriber connects to the broker after the message was received by the broker, the value that had been sent by the publisher will be available on the broker to be forwarded to the new subscriber. This parameter is optional. When it is omitted, the messages sent by the MQTT driver (publisher) will not be retained by the broker. This parameter is ignored when configuring the MQTT driver sheet as a subscriber (receive messages from the broker).

You can also specify a tag in curly brackets to change the station during the runtime (e.g. `{Station}`), The tag value must follow the aforementioned syntax.

Examples of **Station**:

```
192.168.125.31 test.MQTT1.com:1883:1:R
192.168.125.31:1883
192.168.125.31:1883:1
192.168.125.31:1883:1:R
ssl://iot.eclipse.org:8883:1:R
```

Header

Each message is sent by the publisher to the broker with a topic (alphanumeric identifier). The broker forwards the messages to the clients that subscribed to the respective topic. The Header can be used as a prefix that will be concatenated to the text configured in the address column of the driver sheet to define the complete topic name associated with each tag (row of the driver sheet). The MQTT driver does not support the wildcard characters (# and +) for the topic name.

3. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Address

Text that will be concatenated to the Header to form the complete Topic used to publish a message to the broker or to subscribe to a topic. The MQTT driver does not support the wildcard characters (# and +) for the topic name.

The address has a length limit of 1024 characters.

Examples of **Header** and **Address** combinations in Standard Driver Sheets:

Header	Address	Actual Topic
`Building/`	`Room`	Building/Room
`Building/Room/`	`1`	Building/Room/1

Building/Room/1/	Temperature	Building/Room/1/Temperature
Building/Room/2/	Number	Building/Room/2/Number
Building/Street/	Localization	Building/Street/Localization

Note: The tag created on the Studio must be always STRING.

Additionally the suffix ***|HEX*** can be used at the end of the topic name to read and write the topic values in hexadecimal format.

If the Topic name itself ends in |HEX then an additional suffix can be used at the end of the topic name **|STR** which will be the string representation of this topic without hexadecimal format.

I/O Address	Actual Topic	Value
Building/:Room	Building/Room	A
Building/:Room|HEX	Building/Room	41
Building/Room/1::/Temperature	Building/Room/1:/Temperature	abc
Building/Room/1::/Temperature|HEX	Building/Room/1:/Temperature	616263
Building/Room/1::/Temperature|HEX|STR	Building/Room/1:/Temperature|HEX	def
Building/Room/1::/Temperature|HEX|HEX	Building/Room/1:/Temperature|HEX	646566

Note: Each Standard Driver Sheet can have up to 4096 rows. However, the **Read Trigger**, **Enable Read When Idle**, and **Write Trigger** commands attempt to communicate the entire block of addresses that is configured in the sheet, so if the block of addresses is larger than the maximum block size that is supported by the driver protocol, then you will receive a communication error (e.g., "invalid block size") during run time. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet.

4. Save and close the worksheet.

Configuring the Main Driver Sheet

When you add the MQTT driver to your project, the Main Driver Sheet is automatically included in the **MQTT** folder in the Project Explorer. This section describes how to configure the worksheet.

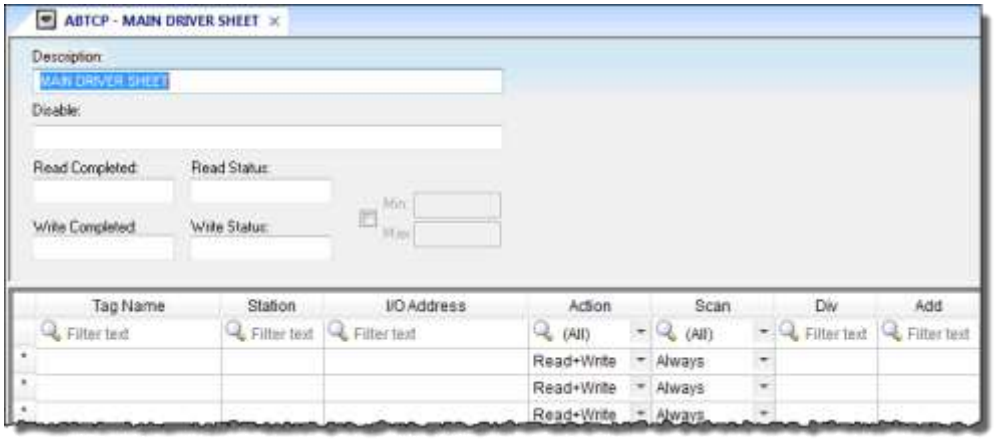
The MQTT driver must be added to the project before you can configure any of its worksheets. For more information, see Adding a communication driver to your project on page 7.

The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. The worksheet is continuously processed during project runtime.

> **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the "Communication" chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the MQTT driver.

1. Do one of the following.
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **Main Driver Sheet** and then select **MQTT** from the list.
   - In the **Comm** tab of the Project Explorer, expand the **MQTT** folder and then double-click **MAIN DRIVER SHEET**.

   The Main Driver Sheet is displayed.



Main Driver Sheet

2. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Station

Supports the same station syntax used for the Standard Driver Sheets. Please see station for standard driver sheet.

I/O Address

Each message is sent by the publisher to the broker with a topic (alphanumeric identifier). The broker forwards the messages to the clients that subscribed to the respective topic. The topic of each message is configured in the I/O Address field. The MQTT driver does not support the wildcard characters (# and +) for the topic name.

The address has a length limit of 1023 characters.

Examples of **I/O Address** in Main Driver Sheet:

I/O Address	Actual Topic
`Building/:Room`	Building/Room
`Building/Room/:1`	Building/Room/1
`Building/Room/1:/Temperature`	Building/Room/1/Temperature
`Building/Room:/2/Number`	Building/Room/2/Number
`Building:/Street:Localization:Home`	Building/Street:Localization:Home

If the topic name has colon character (:) on its name, the first (leftmost) occurrence of this character in the topic name must be duplicated (::) because the driver will always delete the first occurrence of this character on the topic name configured in the I/O address field. This limitation is only specific to Main Driver Sheet and does not happen on the Standard Driver sheet.

Additionally the suffix ***|HEX*** can be used at the end of the topic name to read and write the topic values in hexadecimal format.

If the Topic name itself ends in |HEX then an additional suffix can be used at the end of the topic name **|STR** which will be the string representation of this topic without hexadecimal format.

I/O Address	Actual Topic	Value
`Building/:Room`	Building/Room	A
`Building/:Room|HEX`	Building/Room	41
`Building/Room/1::/Temperature`	Building/Room/1:/Temperature	abc
`Building/Room/1::/Temperature|HEX`	Building/Room/1:/Temperature	616263
`Building/Room/1::/Temperature|HEX|STR`	Building/Room/1:/Temperature|HEX	def
`Building/Room/1::/Temperature|HEX|HEX`	Building/Room/1:/Temperature|HEX	646566

Note: The tag created on the Studio must be always STRING.

Note: The Main Driver Sheet can have up to 32767 rows. If you need to configure more than 32767 communication addresses, then either configure additional Standard Driver Sheets or create additional instances of the driver.

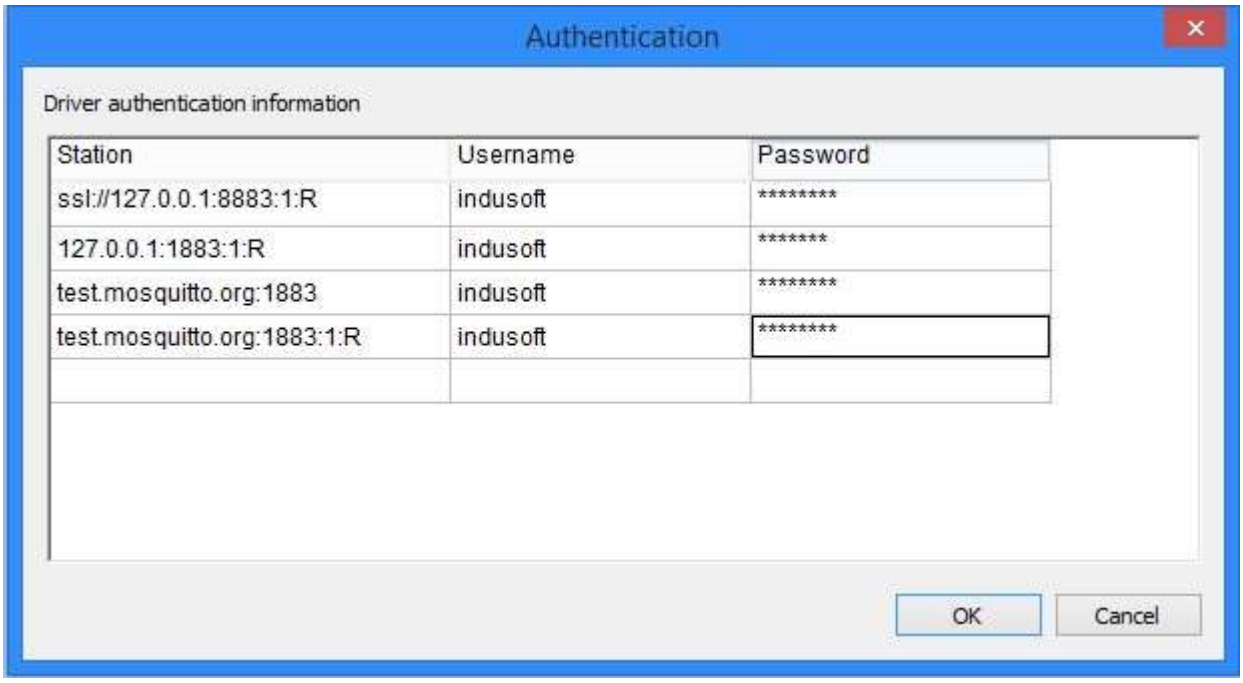
3. Save and close the worksheet.

Additional notes

Additional notes about the MQTT driver.

Driver communication using Authentication

The Authentication feature enables users to specify sensitive information in a secure way. This is used when a broker is configured with a user and password. For every station used which requires the user and password to be specified for successful communication with the broker, has to be set in the **Authentication** tab.



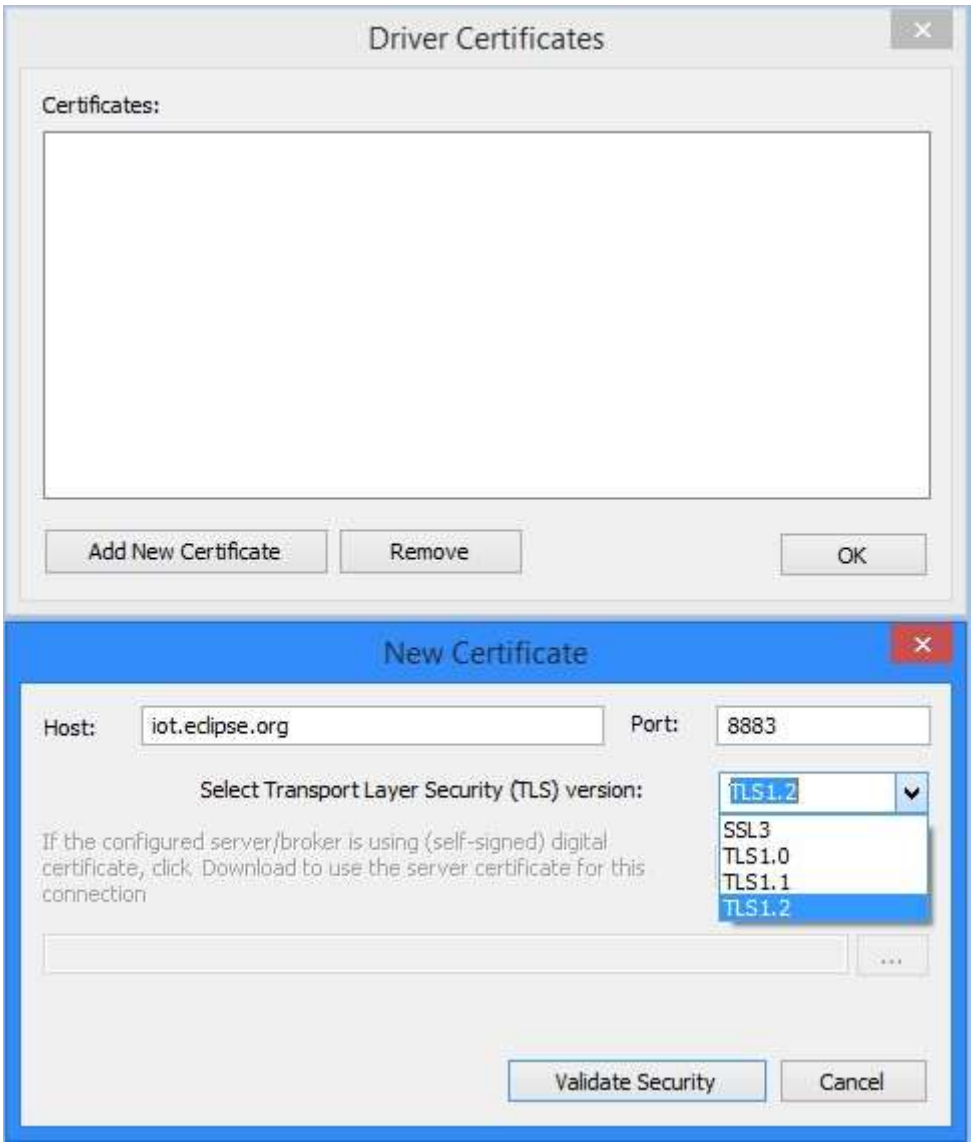
To configure the authentication information click **Authentication** and specify the user and password for a specific station used on the main driver sheet or standard driver sheet. It is not necessary to add information about stations that are unsecured brokers or brokers that use SSL authentication instead of user and password.

> **Note:** The station is treated as an exact string and hence even though the user could communicate with the same broker with the station format configured as the *<IP Address or URL>[:Port Number][:QOS][:R]* or as *<IP Address or URL>[:Port Number]* etc, the user will still have to specify them as 2 separate stations on the **Authentication** tab. As seen in the image above even though the two stations are the same broker, they are treated differently if they both exist on the driver sheets. In this case assuming they have the same user and password, it will still have to be specified again, based on how many different ways the same station is written on the main driver sheet or the standard driver sheets.

> **Note:** To use the Authentication feature of the MQTT driver v1.4 or higher the version of Wonderware Webstudio used should be v8.0 + SP2 or higher.

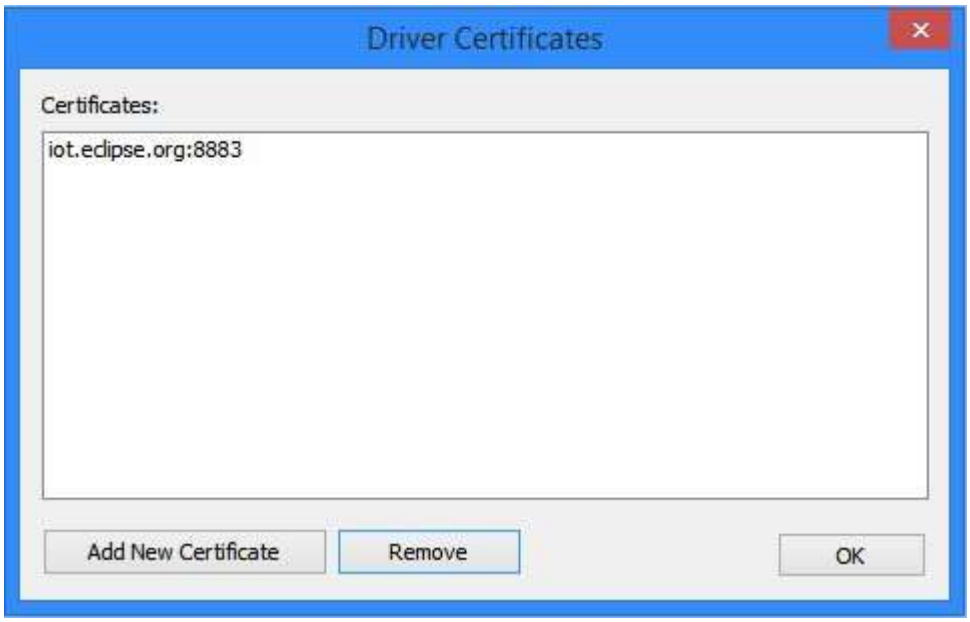
Driver communication using Certificates and Secure Connections (SSL/TLS)

The Certificates feature for the driver enables users to configure certificates for stations to establish a secure connection with the broker. On clicking **Certificates** it will open the *Driver Certificates* window.



Click **Add New Certificate**. On the new window, specify the **Host** which can be the IP or the host URL. Specify the **Port** for communication. By default the SSL port is 8883 for most mosquitto brokers. Then specify the TLS/ SSL verssion, by default the version is set to TLS1.2 which is the most secure version.

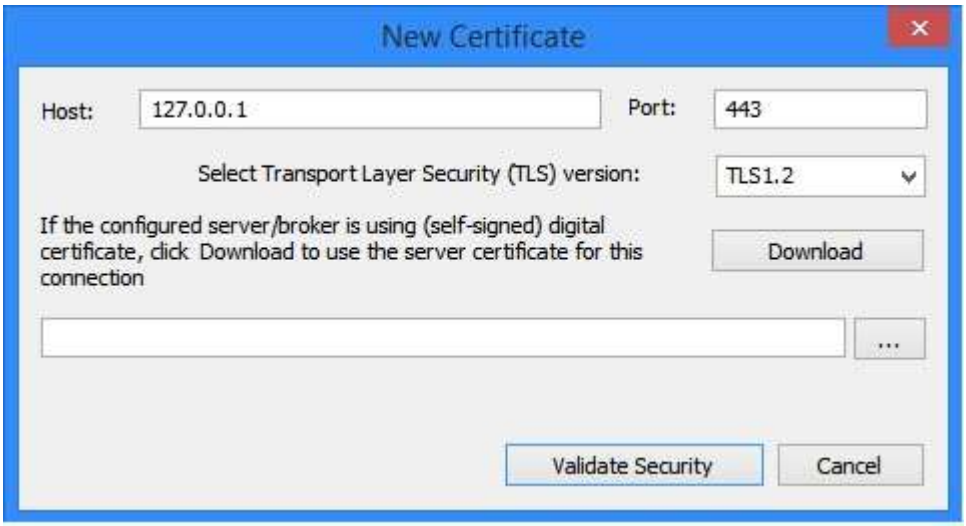
After that click **Validate Security**. This will install the broker's certificate and it should be seen on the *Driver Certificates* window.



Note that when using Certificates and Secure connections the user must denote the station with a prefix of `ssl\\` for example `ssl\\iot.eclipse.org` etc.

Driver communication using Certificates and Secure Connections (SSL/TLS) - Self Signed Certificates

If the broker issues Self signed certificates the **Download** and browse become enabled and the user can see the active message describing that the broker is using self - signed certificates.



Now the user has two options:

1. Download the certificate from the broker by clicking **Download**

Then the user will see a warning message on the *Certificate Download* window explaining that the certificate is being generated from an untrusted source and the user can then click **View Certificate** to ensure its authenticity.

2. Manually specify the file containing the certificate by browsing to it.

ie: browse to the location of the certificate if it is previously saved, instead of making the broker generate it again. The certificate file can be of the type `.pem` or `.crt` extension.

After that the user can click **Validate Security** on the *New Certificate* window and the certificate will be saved and can be seen in *Driver Certificates* window.

Note: Self signed certificates are not secure.

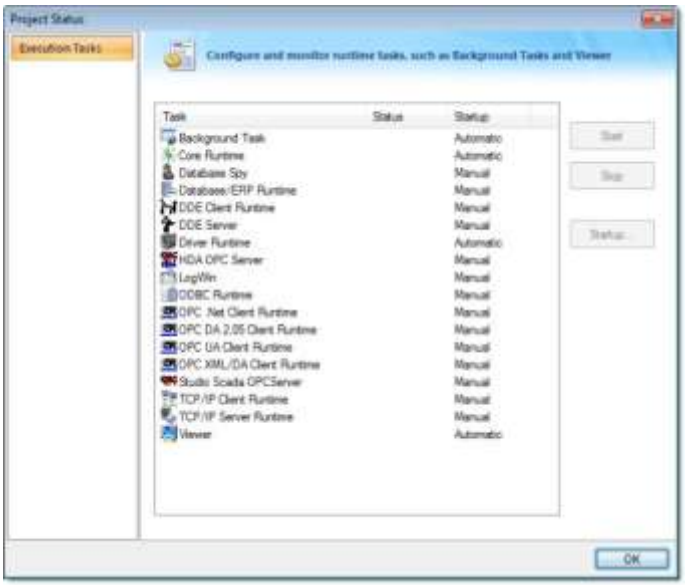
Note: To use the Certificates and Secure Connections using (SSL/TLS) features of MQTT driver v1.4 or higher the version of Wonderware Webstudio used should be v8.0 + SP2 or higher.

Checking the Driver Runtime task

This section describes how to check the status of the Driver Runtime task in the list of execution tasks.

The Driver Runtime task handles communication with remote devices and the processing of the driver worksheets. By default, the task is configured to start up automatically when the project is run, but you can check it for yourself.

1. On the **Home** tab of the ribbon, in either the **Local Management** or the **Remote Management** group (depending on where you project server will be running), click **Tasks**.
   The *Project Status* dialog is displayed.



Project Status dialog

2. Verify that the **Driver Runtime** task is set to **Automatic**.
   - If the setting is correct, then proceed to the next step.
   - If the **Driver Runtime** task is set to **Manual**, select the task and then click **Startup** to change the task to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.

Troubleshooting

This section lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.

Checking status codes

If the MQTT driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** and **Write Status** fields of the driver sheets will receive a status code. Use this status code and the following tables to identify what kind of failure occurred and how it might be resolved. **Status codes for the driver**

Error	Description	Possible Causes	Procedure To Solve
0	OK	Communication without problems.	None required
1	Error. Loading API	Missing paho-mqtt3a.dll (for driver version below 1.3) or paho-mqtt3as.dll (for driver version 1.4 or higher) on Drv/API.	Insert the paho-mqtt3a.dll (for driver version below 1.3) or paho-mqtt3as.dll (for driver version 1.4 or higher) file in the Drv/API sub-folder of Studio. Check if name is correct.
2	Error. Failed to publish to the topic	Invalid address.	Specify a valid address and check the supported address formats described in this document.
3	Error. Failed to subscribe to the topic(s)	Invalid address.	Specify a valid address and check the supported address formats described in this document.
4	Timeout connecting with the Broker	Connection lost or server/broker offline.	Check the station and network connectivity and try again. If using SSL/TLS connection please increase the timeout.

5	Error. Failed to connect with the Broker	Invalid client instance.	Check the IP and TCP Port on station and network connectivity and try again.If using SSL/TLS connection please increase the timeout.
6	Error. Create client instance	Invalid IP or Port on station.	Check the IP and Port on station and try again.
7	Invalid Station	The specified station in the driver worksheet is invalid or out of range.	Check the supported station formats and parameters described in this document, and then correct the station.
8	Invalid Address	Invalid characters. '+' and '#' are not accepted in address names.	Remove invalid characters from the specified address.
9	Invalid Header	Invalid characters. '+' and '#'' are not accepted in header names.	Remove invalid characters from the specified header.
10	Mising SSL/TLS Certificate authentication	User did not added this host on the Security Manager dialog	Right click on the driver, go to Settings, click on Certificates, and add the host you are trying to communicate. (For more information search for certificate manager on the product help).
100	Illegal operation	Tried to write to read-only addresses (1x and 3x).	Writing operations are possible only in Coil Status and Holding Registers areas.
1005	Timeout	The request timed out.	Check the station, tagnames and network connectivity and try again.

Common status codes

Status Code	Description	Possible Causes	Procedure To Solve
0	OK	Communicating without error.	None required.
-15	Timeout waiting for message to start	• Disconnected cables. • PLC is turned off, in stop mode, or in error mode. • Wrong station number. • Wrong parity (for serial communication). • Wrong RTS/CTS configuration (for serial communication).	• Check cable wiring. • Check the PLC mode — it must be RUN. • Check the station number. • Increase the timeout in the driver's advanced settings. • Check the RTS/CTS configuration (for serial communication).
-33	Invalid driver configuration file	The driver configuration file (*drivername*.INI) is missing or corrupt.	Reinstall the driver.
-34	Invalid address	The specified address is invalid or out of range.	Check the supported range of addresses described in this document, and then correct the address.
-35	Driver API not initialized	The driver library was not initialized by the driver.	Contact technical support.
-36	Invalid data type	The specified data type is invalid or out of range.	Check the supported data types described in this document, and then correct the data type.
-37	Invalid header	The specified header in the driver worksheet is invalid or out of range.	Check the supported range of headers described in this document, and then correct the header.
-38	Invalid station	The specified station in the driver worksheet is invalid or out of range.	Check the supported station formats and parameters described in this document, and then correct the station.

-39	Invalid block size	Worksheet is configured with a range of addresses greater than the maximum block size.	Check the maximum block size number of registers described in this document, and then configure your driver worksheet to stay within that limit. Keep in mind that you can create additional worksheets. **Note:** If you receive this error from a Main Driver Sheet or Tag Integration configuration, please contact Technical Support.
-40	Invalid bit write	Writing to a bit using the attempted action is not supported.	• Writing to a bit using Write Trigger is not supported in some drivers. Modify the driver worksheet to use Write On Tag Change. • The bit is read-only.
-42	Invalid bit number	The bit number specified in the address is invalid. The limit for the bit number depends on the registry type.	Check the addresses to see if there are bit numbers configured outside the valid range for the registry.
-43	Invalid byte number	The byte number specified in the address is invalid. The limit for the byte number depends on the registry type.	Check the addresses to see if there are byte numbers configured outside the valid range for the registry.
-44	Invalid byte write	Writing to a byte using the attempted action is not supported.	The byte is read-only or inacessible.
-45	Invalid string size	The string is more than 1024 characters.	Modify the addresses that have string data type to be less than 1024 characters.

Monitoring device communications

You can monitor communication status by establishing an event log in Studio's *Output* window (LogWin module). To establish a log for Field Read Commands, Field Write Commands and Serial Communication, right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the LogWin module to establish an event log on a remote unit that runs Windows Embedded. The log is saved on the unit in the `celog.txt` file, which can be downloaded later.

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software. Quite often, communication is interrupted by a hardware or cable problem or by a device configuration error. If you can successfully communicate using the programming software, then recheck the driver's communication settings in Studio.

Contacting Technical Support

If you must contact Technical Support, please have the following information ready:

- **Operating System** and **Project Information**: To find this information, click **Support** in the **Help** tab of the ribbon.
- **Driver Version** and **Communication Log**: Displays in the *Output* window (LogWin module) when the driver is enabled and the project is running is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Revision history

This section provides a log of all changes made to the driver.

Revision history

Driver Version	Revision Date	Description of Changes	Author
1.0	2015-10-29	First driver revision.	Leandro Gioria
1.1	2015-11-25	Updated Status codes, Standard Driver Sheet and Main Driver Sheet.	Leandro Gioria
1.2	2016-01-28	New driver installer created	Leandro Gioria
1.3	2016-09-15	Fixed issue where value and timestamp of the tag were not correct when it's quality is uncertain	Anushree Phanse
1.4	2017-04-17	Added support for authentication and SSL/TLS connections	Paulo Balbino / Anushree Phanse/Michael Hayden
1.5	2017-11-28	Added support for reading and writing to string values with hexadecimal format with new suffixes.	Anushree Phanse
1.6	2018-03-16	Fixed issue when using read and write triggers on standard driver sheet	Anushree Phanse
1.7	2019-01-23	Fixed communication issue with a broker that uses self signed certificates. Fixed crash on communicating with broker that uses signed certificates.	Anushree Phanse