

UC20-SL2000-OLAC

Quick Start Guide for MQTT Service

Abstract:

This document explains how to install and use the MQTT client service for the UC20-SL2000-OLAC PLC. The MQTT service reads a configuration provided by an IEC61131-3 project. The user may configure multiple publish intervals with one or more topics and one or more variables per topic. The user may select publish-on-change with or without send-only-changed and raw or JSON message format. The user may also configure subscribe topics and associated variables and configure secure MQTT via TLS. Usage requires to add the MQTT service software unit to your u-create studio installation and then download it to your u-control PLC.

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	UC20-SL2000-EC	2674520000	HW 01.xx.xx / FW 1.20.2 (or higher)
2	UC20-SL2000-EC-CAN	2674620000	HW 01.xx.xx / FW 1.20.2 (or higher)

Software reference

No.	Software name	Article No.	Software version
1	u-create Studio	2660130000	1.20 (or higher)
2	MQTT.fx	-	1.7.1 (or higher)

File reference

No.	Name	Description	Version
1	QSG0011-UC20-SL2000MQTT Service.zip	u-create studio software unit	1.0.1

Contact

Weidmüller Interface GmbH & Co. KG
Klingenbergstraße 26
32758 Detmold, Germany
www.weidmueller.com

For any further support please contact your local sales representative:
<https://www.weidmueller.com/countries>

Content

1	Warning and Disclaimer.....	4
2	Adding the software unit to u-create studio	5
3	Using the MQTT Service in your project	7
3.1	Preparing variables to be used by the Service	7
3.2	Configuration via Expert Entries.....	9
3.3	Broker configuration.....	10
3.4	TLS configuration	11
3.5	Connection status.....	12
3.6	Publish configuration	13
3.7	Subscribe configuration	14
4	Limitations	15
5	Installation of MQTT service on the PLC.....	16
5.1	Installation of MQTT service via “Add / remove software unit”	16
5.2	Installation of MQTT service via “Create target”	17
6	Communication testing	19
6.1	Error handling.....	23

1 Warning and Disclaimer

Warning

Controls may fail in unsafe operating conditions, causing uncontrolled operation of the controlled devices. Such hazardous events can result in death and / or serious injury and / or property damage. Therefore, there must be safety equipment provided / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer

This Application Note / Quick Start Guide / Example Program does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Each user is responsible for the correct operation of his control system. By using this Application Note / Quick Start Guide / Example Program prepared by Weidmüller, you accept that Weidmüller cannot be held liable for any damage to property and / or personal injury that may occur because of the use.

Note

The given descriptions and examples do not represent any customer-specific solutions, they are simply intended to help for typical tasks. The user is responsible for the proper operation of the described products. Application notes / Quick Start Guides / Example Programs are not binding and do not claim to be complete in terms of configuration as well as any contingencies. By using this Application Note / Quick Start Guide / Example Program, you acknowledge that we cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this application note / quick start guide / example at any time without notice. In case of discrepancies between the proposals Application Notes / Quick Start Guides / Program Examples and other Weidmüller publications, like manuals, such contents have always more priority to the examples. We assume no liability for the information contained in this document. Our liability, for whatever legal reason, for damages caused using the examples, instructions, programs, project planning and performance data, etc. described in this Application Note / Quick Start Guide / Example is excluded.

Security notes

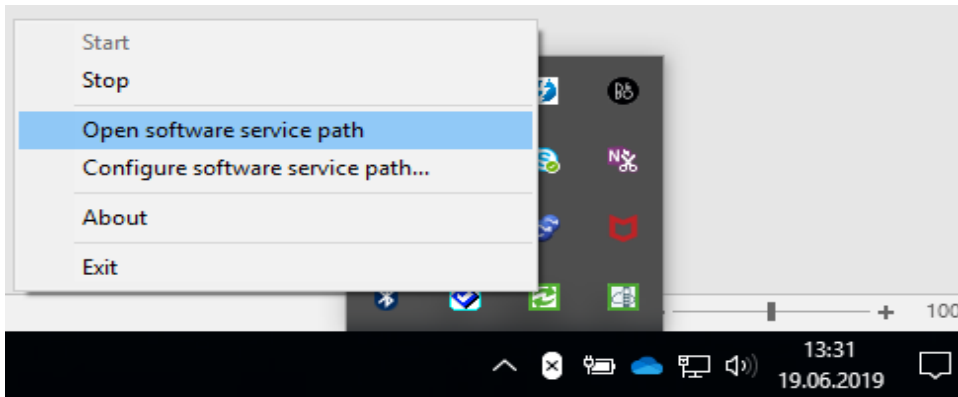
In order to protect equipment, systems, machines and networks against cyber threats, it is necessary to implement (and maintain) a complete state-of-the-art industrial security concept. The customer is responsible for preventing unauthorized access to his equipment, systems, machines and networks. Systems, machines and components should only be connected to the corporate network or the Internet if necessary and appropriate safeguards (such as firewalls and network segmentation) have been taken.

2 Adding the software unit to u-create studio

- To install the MQTT service on the PLC, a software unit must be added to the installation folder on the engineering PC.

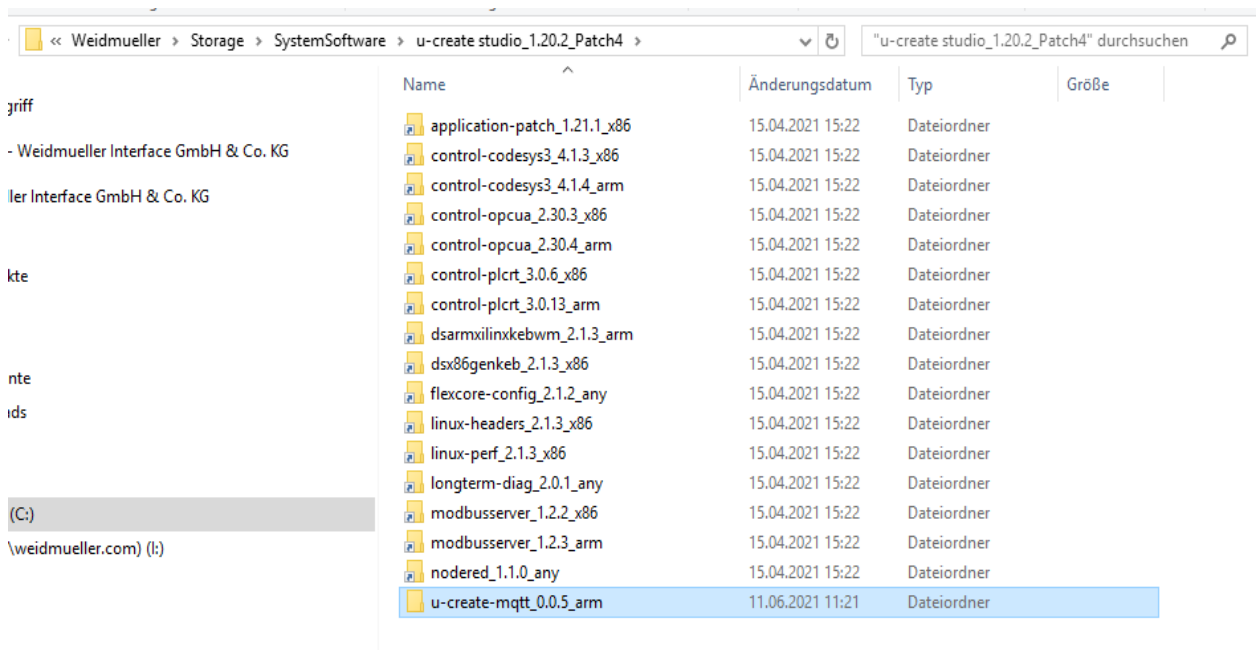
This addition is achieved by the following steps.

- 1) Open the software service path.



- 2) Navigate to the used u-create studio installation.

.../Weidmueller/Storage/SystemSoftware/u-create studio_x.xx :



- 3) Unzip the software unit file. (u-create-mqtt_<x.x.x>_arm.zip).
- 4) Add the folder with the software unit.

Note: Adding a new folder to this path requires admin rights.

The inner structure of the folder should look like this:

Name ^	Änderungsdatum	Typ	Größe
 debianrepository	04.06.2021 16:37	Dateiordner	
 licenses	04.06.2021 16:37	Dateiordner	
 meta.xml	04.06.2021 16:37	XML-Dokument	3 KB

The actual download of the software unit to your PLC has its own description in Chapter 5 and its subchapters.

3 Using the MQTT Service in your project

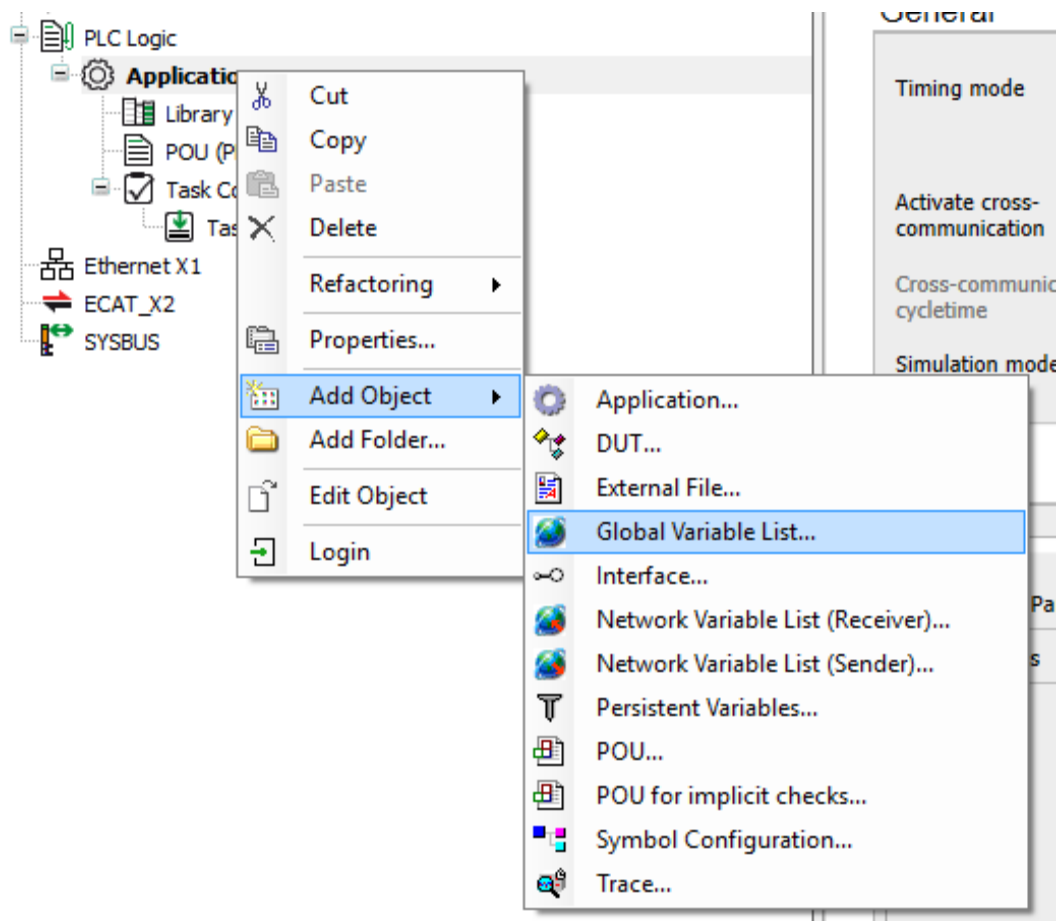
The MQTT Service is a Linux application running on the PLC in parallel and independent of the IEC 61131 runtime. It connects to the variable server on the PLC and opens variables you specify via expert entries in your u-create studio project. The MQTT service then publishes the content of these variables as a JSON object or receives a JSON object and writes its members to said variables. As an alternative to JSON, you may select “raw” format: the MQTT service will then publish one variable per topic as a string or copy a string-ified value from a subscribed MQTT message into one variable per topic.

The following section describes how to setup the communication between the MQTT service and the IEC 61131-3 runtime via the variable server on the PLC.

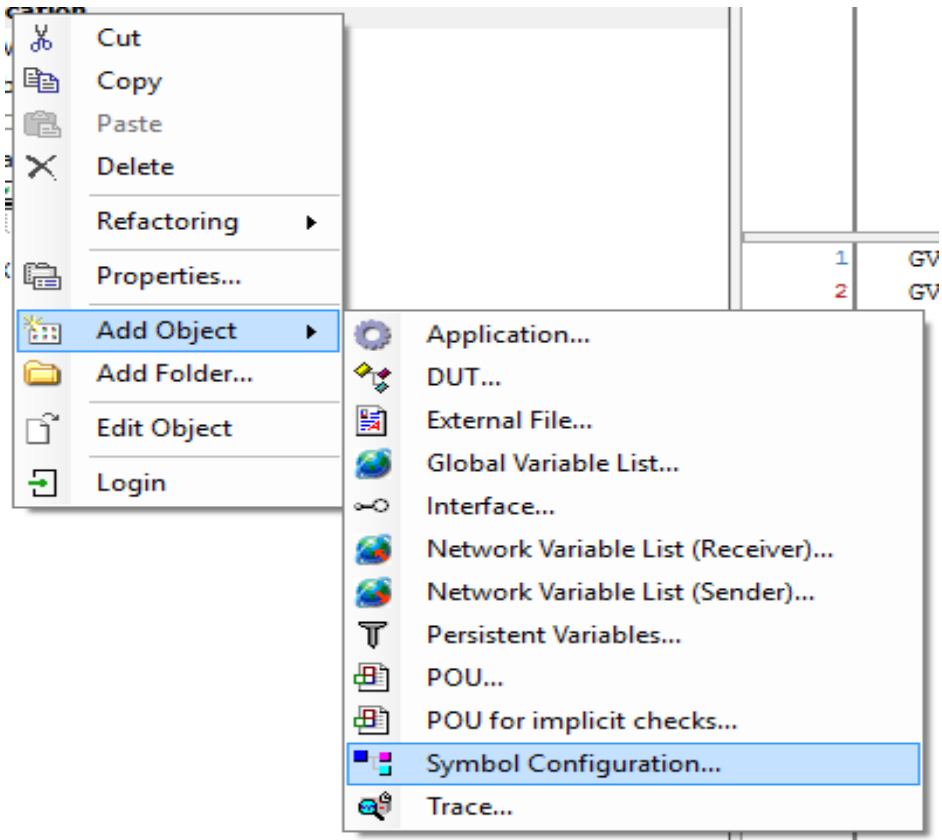
3.1 Preparing variables to be used by the Service

The communication is achieved by using Global Variable Lists.

- 1) Create a new GVL (Global Variable List).



- 2) After some variables have been added, these variables need to be included into the Symbol Configuration.



- 3) Once a Symbol Configuration has been created, the Global Variable List can be added.

Symbols	Access Rights	Maximal	Attribute	Type
Constants				
<input checked="" type="checkbox"/> GVL_MQTT_publish				
<input checked="" type="checkbox"/> iMqttTopicOne				INT
<input checked="" type="checkbox"/> xMqttTobisTwo				BOOL
IoConfig_Globals				



The variables are only visible in the symbol configuration if they are used inside the program and after you have executed the build process at least once.

Variables that are not part of the Symbol Configuration cannot be published or subscribed into.

If new variables are added to the GVL, the Symbol Configuration needs to be updated.

3.2 Configuration via Expert Entries

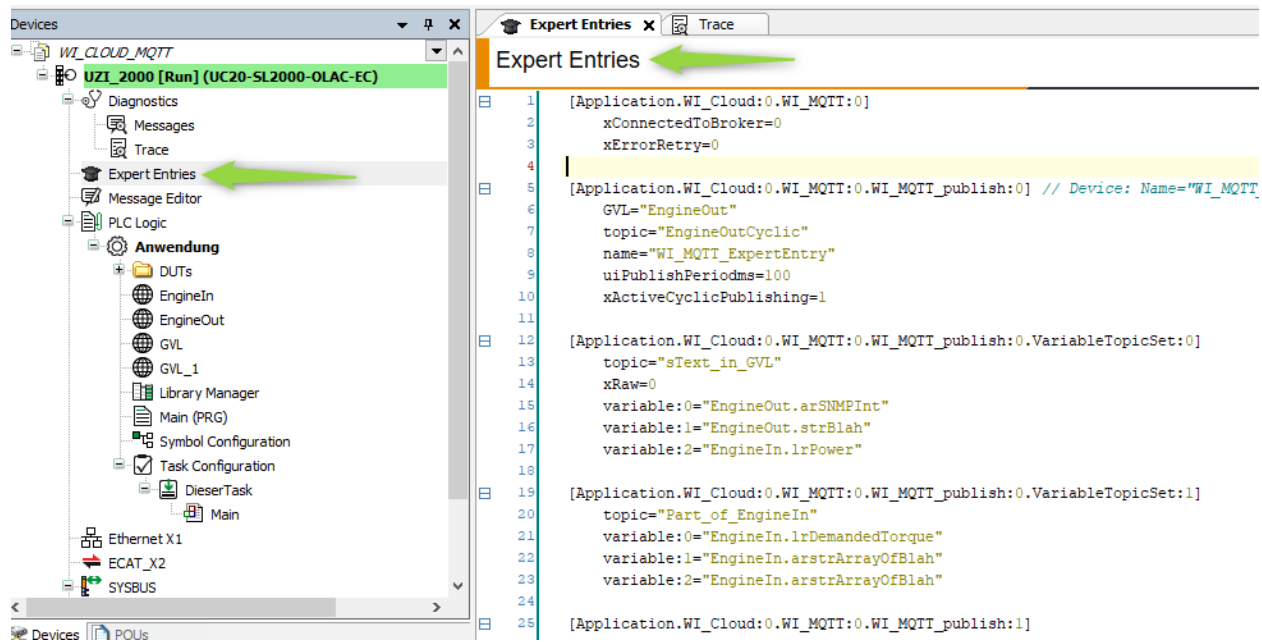
The MQTT service requires the following sets of Information from you:

1. Broker configuration.
2. Client ID and User credentials (optional).
3. Security configuration (optional).
4. Connection status.
5. Publish configuration.
6. Subscribe configuration.



It is not mandatory to provide both a subscribe and a publish configuration. But at least one of (publish configuration / subscribe configuration) must be present in your expert entries or the MQTT service will have no meaningful work to do and stop itself.

u-create studio offers a feature called “Expert Entries” for the above configuration items:



An expert entry consists of a path in the variable server in square brackets and one or more variable-value pairs:

```

[<path in variable server>]
  variable1="string value"
  variable2=123
  
```

The following chapters will discuss the configuration items in detail.

3.3 Broker configuration

- Please add the path **[Application.WI_Cloud:0.WI_MQTT:0]** to your expert entries and add the variables and values as depicted in the example below.

The following section describes their purpose in more detail:

- 1) **ApplicationName**: In order to find the variables, you want to publish out of / subscribe into, the MQTT service needs to know the name of your application. You find this name in u-create studio's device view as a main item in the PLC Logic. The default value for a new IEC project is "Application".
 - 2) **strClientID**: It is the client ID the MQTT service identifies itself towards a broker during e.g. TLS negotiation.
 - 3) **BrokerURL**: It is the IP address or URL of your broker.
 - 4) **BrokerPort**: It is the TCP port number the MQTT service uses for communication with the broker.
 - 5) **keepAliveTime**: The MQTT service as client sends an "I'm still here" – message to the broker every keepAliveTime seconds to verify the connection to the broker.
 - 6) **xCleanSession**: This flag tells the broker to clean all messages and subscriptions on disconnect if true, or to keep them, if false.
 - 7) **xUseUserAuthentication**: This flag is optional. Set to 1 if you want to use a username / password authentication with the broker. If you do not use username/password authentication, you may omit strUsername and strPassword.
 - 8) **strUsername** is the username for optional authentication with the broker.
 - 9) **strPassword** is the password for optional authentication with the broker.
 - 10) **xActivateConnection**
 - 11) **xConnectedToBroker**
 - 12) **xErrorRetry**
- } Please do refer to chapter 3.5 regarding these three flags.
- 13) **uiTraceLevel** This is optional. It tells the MQTT service how much information it shall post in the PLC's Trace log. Zero lets the MQTT service be quiet after it has digested its configuration. A value of one or two lets the MQTT service provide some more information.
 - 14) **uiSubscribeQueueMemLimitKB** This is optional. It tells the MQTT service the limit for how much memory the subscribe message queue may use. If the total memory for incoming and not yet processed subscribe messages exceeds this limit, the service will drop subscribe messages until the queue memory size has shrunk below that limit, again. The default value is 1024, i.e. a subscribe message queue memory size limit of one Megabyte.

<Example follows on next page>

The uiSubscribeQueueMemLimitKB example:

```
[Application.WI_Cloud:0.WI_MQTT:0]
  ApplicationName="Application"
  strClientID="somedevicename"
  BrokerURL="bcondeviothub.azure-devices.net"
  BrokerPort=8883
  keepAliveTime=60
  xCleanSession=1
  xUseUserAuthentication=1
  strUsername="yourusername"
  strPassword="yourpassword"
  xActivateConnection=1
  xConnectedToBroker=0
  xErrorRetry=0
  uiTraceLevel=1
  uiSubscribeQueueMemLimitKB=128
```

3.4 TLS configuration

The MQTT service supports secure MQTT. You may choose between Certificate-based TLS or Pre-shared key TLS (PSK-TLS).

To select the Certificate-based configuration:

- 1) Add the flag xUseCertificateBasedTLS.
- 2) Set it to 1.
- 3) You also need to provide a combination of the following variables.
 - strCAfilename is a path to a file containing the PEM encoded trusted CA certificate files.
 - strCAfilepath is a path to a directory containing the PEM encoded trusted CA certificate files.
 - strCertfileName is a path to a file containing the PEM encoded certificate file for this client. If empty, strKeyfileName must also be empty and no client certificate will be used.
 - strKeyfileName is a path to a file containing the PEM encoded private key for this client. If empty, strCertfileName must also be empty and no client certificate will be used.



either strCAfilename or strCAfilepath **must not be** empty. If they are, the MQTT service will enter the error state. Set CA file path to the directory where the CA file is in.

Certificate-based TLS configuration example:

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_TLS:0]
  xUseCertificateBasedTLS=1
  strCAfilename="/home/admin/certificate/DigiCertBaltimoreRoot.pem"
  strCAfilepath="/home/admin/certificate/"
  strCertfileName="/home/admin/certificate/cert.pem"
  strKeyfileName="/home/admin/certificate/key.pem"
```



Please understand that the CA certificate, your certificate and your private key **have to be** in the folders you have specified on the PLC.
Use e.g. WinSCP to copy the needed files into the proper folders on the PLC.

A PSK-TLS configuration, on the other hand, consists of the following items:

- xUsePSKBasedTLS: this flag needs to be set to 1 to select PSK-based TLS.
- strPSK: a pre-shared key in hex format **without** leading "0x".
- strIdentity: The identity string may be used as the username depending on the server settings.

PSK-TLS configuration example:

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_TLS:0]
  xUsePSKBasedTLS=1
  strPSK="CAACAFFE0123456789"
  strIdentity="myEncryptionUser"
```

3.5 Connection status

An IEC application may observe the connection status of the MQTT service.

The MQTT service sets the variable xConnectedToBroker to "1" after it has established a broker connection and to "0" otherwise. Access to this system variable within the IEC context is via **"SYS.CAT.Application.WI_Cloud:0.WI_MQTT:0.xConnectedToBroker"**.

An IEC application may want to switch the connection to the broker on and off. For this purpose, the configuration includes the variable xActivateConnection. "1" tells the MQTT service to connect; a "0" tells the service to disconnect. Access to this system variable within the IEC context is via **"SYS.CAT.Application.WI_Cloud:0.WI_MQTT:0.xActivateConnection"**.

In some situations, the MQTT service may enter the "error" state, and communicate this error via the "Trace view" inside u-create studio if uiTraceLevel is greater than zero. A transition of xErrorRetry from 0 to 1 will release the MQTT service from the error state and the MQTT service will attempt a restart.

3.6 Publish configuration

The MQTT service accepts multiple publish configurations that configure an association between a **Publish Cycle Time** and one or more **variable-topic-sets**. Each variable topic set associates a MQTT topic with one or more variables and, if so desired, with a “raw” property.

You select the value format in the MQTT message with this property: xRaw not given or xRaw = “0” selects JSON and xRaw = “1” selects the string-ified value of one variable. With **xRaw = 1**, a variable topic set holds **one and only one variable**.

Detailed description:

The path pattern **[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_publish:<n>]**, (n is a number between 0 and infinity) adds publish configurations to the MQTT configuration. Within each publish configuration, the MQTT service expects a variable uiPublishPeriodms. The MQTT service will compose a message for each variable topic set in this publish configuration every uiPublishPeriodms milliseconds.

Example:

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_publish:0]
  uiPublishPeriodms=1000
```

As mentioned above, a publish configuration **must** include at least one variable topic set.

The MQTT service searches for the following path pattern:

[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_publish:<i>.VariableTopicSet:<k>]

where <i> is the publish configuration’s number and <k> is a number between 0 and infinity.

A variable topic set must contain the following items:

- A MQTT topic in the parameter “topic”
- Variable names. Use the variable name pattern “**variable:<n>**” with <n> between 0 and infinity.

A variable topic set may contain the variable “**xRaw**”.

Set xRaw to 1 if you want to put the string-ified value of one (and only one) variable into the MQTT publish message payload.

Variable topic set example:

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_publish:0.VariableTopicSet:0]
  topic="MQTTout"
  xRaw=0
  variable:0="MQTT_OUT.q_iVoltageProTop1"
  variable:1="MQTT_OUT.q_iVoltageProTop5"
```

This example lets the MQTT service publish into the topic “**MQTTout**” and add the values of the variables “**q_iVoltageProTop1**” and “**q_iVoltageProTop5**” from the global variable list MQTT_OUT into the message payload.

It also tells the service to use JSON format for the message payload.

If you want to use the options “send only on value change” and “send only the changed variables”, you need to add this:

```
xSendOnValueChange=1
xSendOnlyChanged=1
```

3.7 Subscribe configuration

A subscribe configuration associates a topic with one or more variables. After the service has established a broker connection, it will subscribe to the topic(s) at the broker. The service will then match the topics of incoming messages with the configured subscribe topics. If it finds a match, it will transfer the message payload’s content into the variables associated with the matching topic.

The MQTT service accepts multiple subscribe configurations by the path pattern

[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_subscribe:<k>] with <k> between 0 and infinity.

A subscribe configuration must contain the following items:

- a topic in the parameter “**topic**”.
- IEC variable names in variables named “variable:<k>” with k between 0 and infinity.
- It may contain a raw flag “**xRaw**”.

With xRaw = 1, the service will accept **one and only one variable** in the subscribe configuration. It will expect a string-ified value of matching type in the message payload and transfer it into the given variable. With xRaw not given or equal to 0, the MQTT service will interpret incoming message payloads as JSON and transfer the JSON blob’s content into the variables provided by name in the subscribe configuration.

Subscribe configuration example:

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_subscribe:0]
  topic="MQTTIN"
  xRaw=0
  variable:0="MQTT_IN.I_diTemperature"
  variable:1="MQTT_IN.I_iVoltage"
```

This subscribe configuration subscribes to the topic “**MQTTIN**” and tells the service to expect a JSON message as a payload. It associates the variables “**I_diTemperature**” and “**I_iVoltage**” from the Global Variable List MQTT_IN with the topic.

4 Limitations

The configuration description in chapter 3.6 and chapter 3.7 mentions that the MQTT service can process many publish and subscribe configurations and many variables each. The service itself does not limit the numbers you can use for publishing. It does limit the memory used by unprocessed subscribe messages to a configured size by dropping incoming subscribe messages until the memory size of the subscribe message queue has reduced below the limit.

Regarding variable topic sets, please also note that the service waits the publish cycle time and then processes the configured variables into a MQTT message. Thus, the time needed to process the variables adds to the cycle time you have configured for that variable topic set. Only after the service has sent the message, it will again start waiting out the publish cycle time.

Nevertheless, each item consumes system resources and thus CPU speed and RAM size of the PLC put a lower limit to publish cycle time and an upper limit to the number of publish configurations, subscribe configurations and variables for the MQTT service.

5 Installation of MQTT service on the PLC

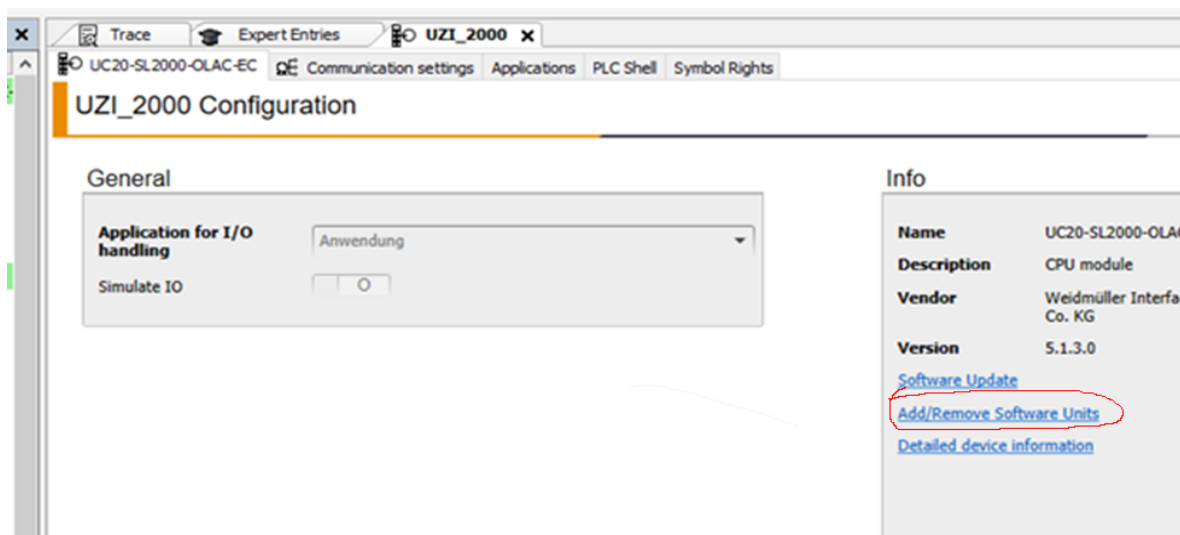
This chapter explains how to get the MQTT service onto the controller.

You can either log in into the PLC and use the “**Add / Remove Software Units**” button or you may create a new target for the PLC that includes the MQTT service.

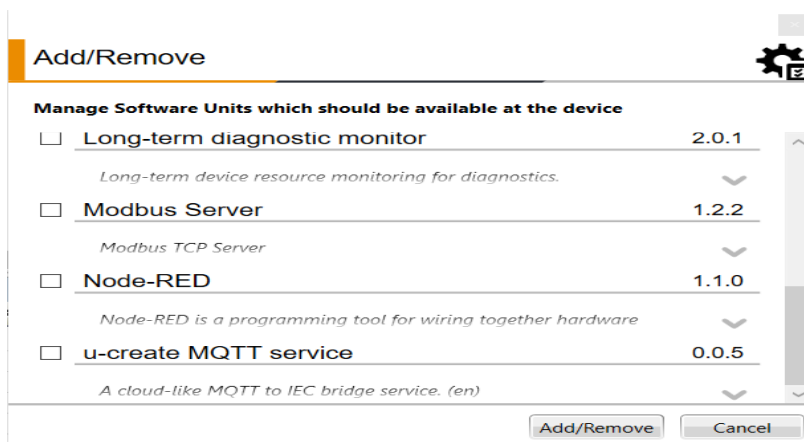
5.1 Installation of MQTT service via “Add / remove software unit”

If you do this the first time since installing U-create Studio, some extra steps may be required. Please refer to AN0068 “create Software Units for U-create Studio”, chapter “Install your Software Unit on the PLC”, subchapter “Add / remove software unit”.

- 1) Log in into the PLC with u-create studio. Then open the PLC device by double-clicking it in the “Devices” view.



- 2) Click on “Add / remove software units”. U-create Studio will list all the software units currently installed on your controller.



- 3) Select the u-create MQTT service and click “Add/Remove”. u-create studio will install the software unit on your PLC.

If an error message is shown, open **Project->Project Settings -> Software Service** and enter the IP address that your PC uses for the connection to the controller. (The PC interface’s IP address, not the PLC’s IP address.)

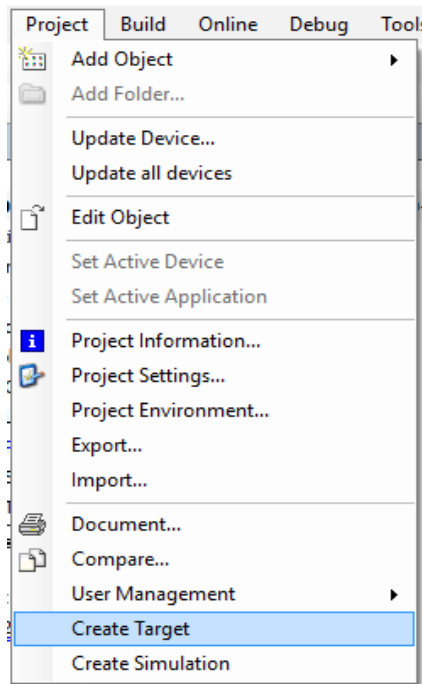
5.2 Installation of MQTT service via “Create target”

Please also read the u-control manual available on WI’s website. In chapter “Commissioning using u-create studio”, subchapter “Installing the firmware on the controller” is a step-by-step description on how to create a target SD card for the PLC.

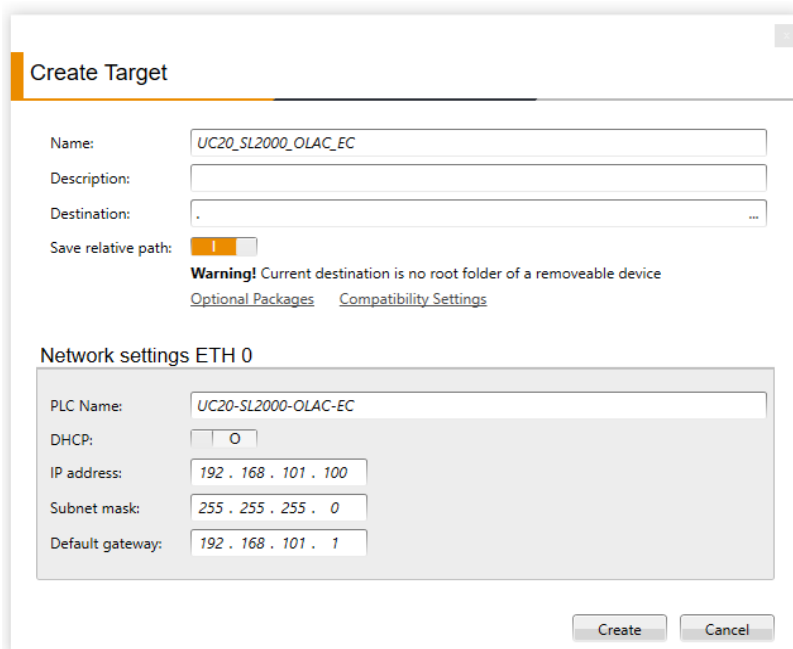
Warning: A new target will download a new Linux installation to the controller. Any previous changes will be overwritten.

To include the MQTT service in the target SD card follow these steps:

- 1) Start u-create studio.
- 2) Navigate to **Project-> Create Target**.



- 3) Click on Optional Packages.



Create Target

Name:

Description:

Destination:

Save relative path: ☒

Warning! Current destination is no root folder of a removeable device

[Optional Packages](#) [Compatibility Settings](#)

Network settings ETH 0

PLC Name:

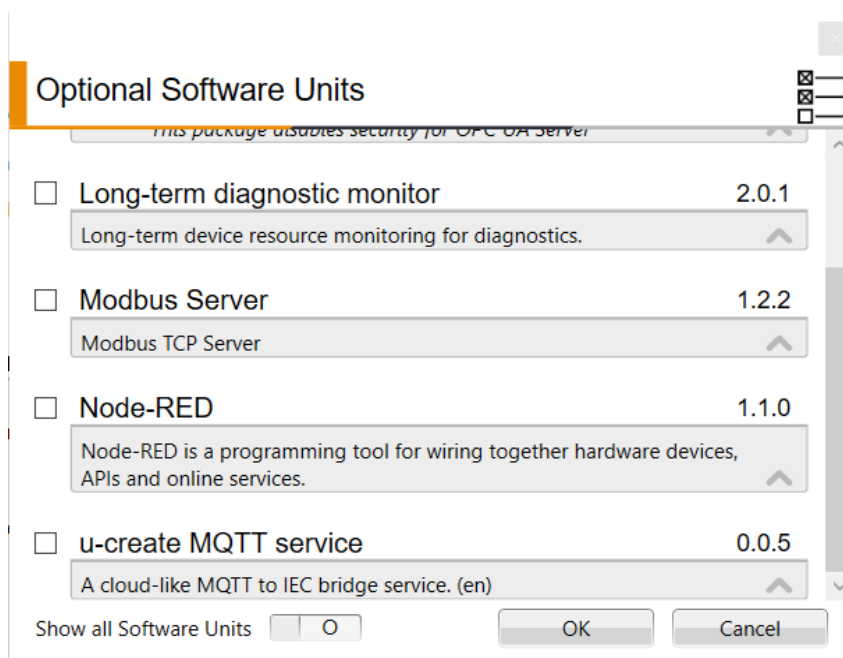
DHCP: ☐ DHCP ☒ Static

IP address:

Subnet mask:

Default gateway:

- 4) Select u-create MQTT service.



Optional Software Units

this package disables security for OPC UA Server

☐ Long-term diagnostic monitor 2.0.1
Long-term device resource monitoring for diagnostics.

☐ Modbus Server 1.2.2
Modbus TCP Server

☐ Node-RED 1.1.0
Node-RED is a programming tool for wiring together hardware devices, APIs and online services.

☐ u-create MQTT service 0.0.5
A cloud-like MQTT to IEC bridge service. (en)

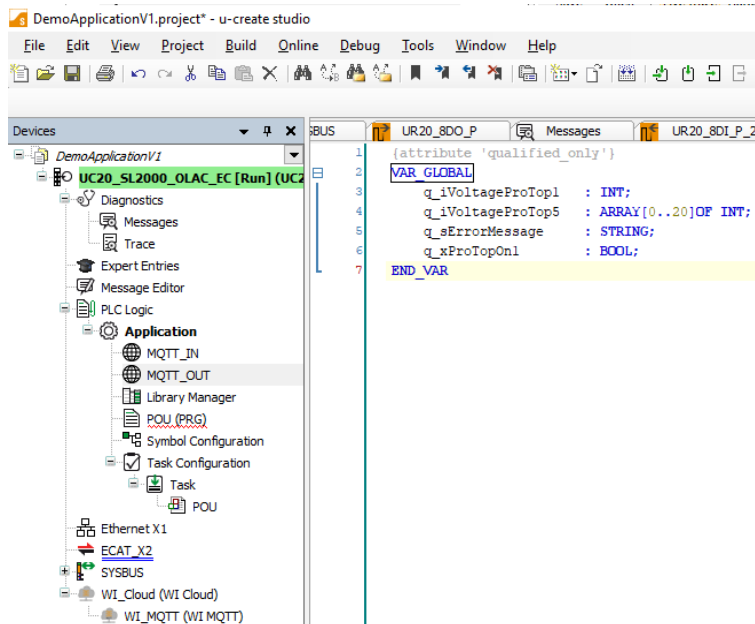
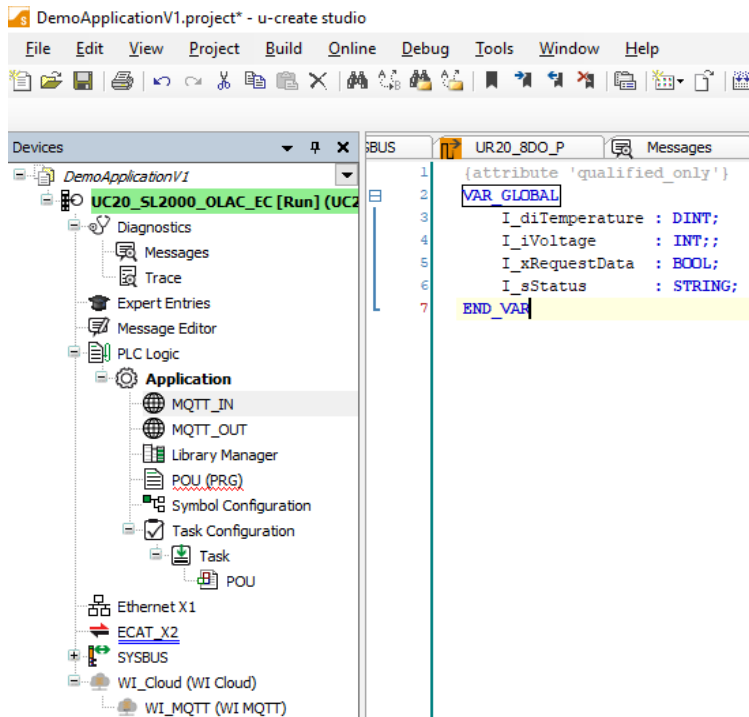
Show all Software Units ☐ Show ☒ Hide

- 5) Create the new target on the SD-card.
- 6) Insert the SD-card into the controller.

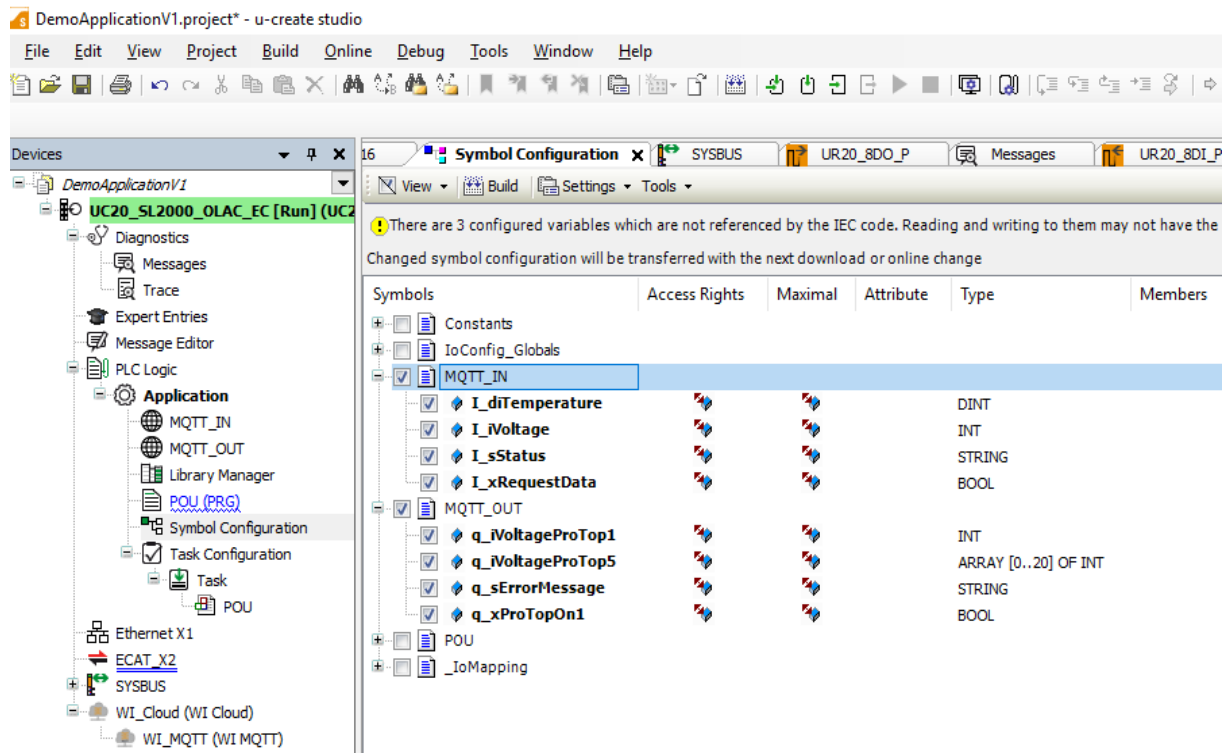
6 Communication testing

For testing the communication, it is necessary to use a MQTT broker. In this case we use the free online broker “test.mosquitto.org”, but it is also possible to use your own local MQTT broker.

- 1) Open a new u-create studio default project.
- 2) Create one or more GVL's with variables you want to subscribe to or publish from.



- 3) Create a Symbol Configuration and activate the variables you want to publish or subscribe.



- 4) Open the Expert Entries and add the configuration as described in chapters 3.3 to 3.7.

Example configuration you might want to copy to your Expert Entries as a starting point.

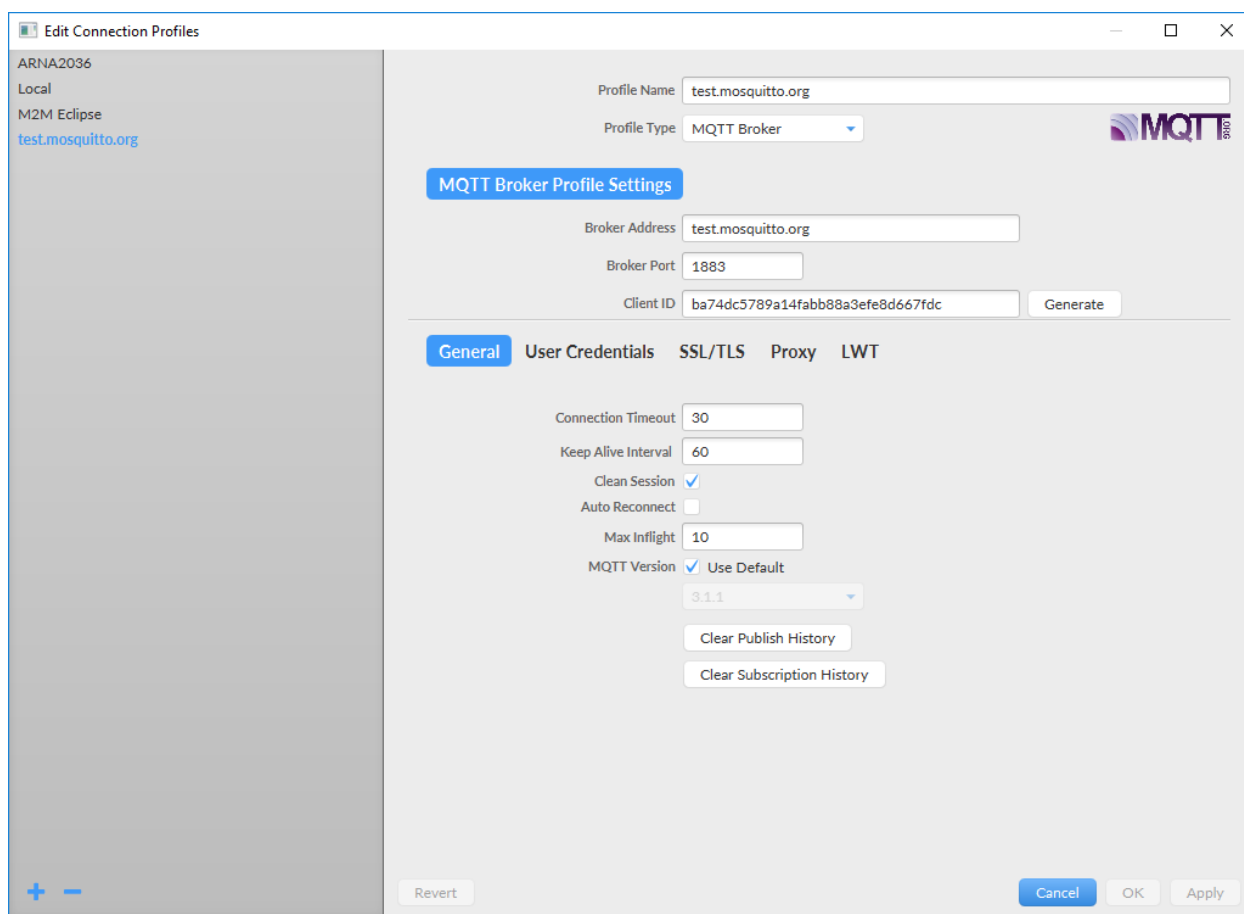
```
[Application.WI_Cloud:0.WI_MQTT:0]
  ApplicationName="Application"
  strClientID="somedevicename"
  xCleanSession=1
  BrokerURL="test.mosquitto.org"
  BrokerPort=1883
  keepAliveTime=10
  xUseUserAuthentication=0
  strUsername=""
  strPassword=""
  xActivateConnection=1
  xConnectedToBroker=0
  xErrorRetry=0
  uiTraceLevel=2
```

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_publish:0]
  uiPublishPeriodms=1000
```

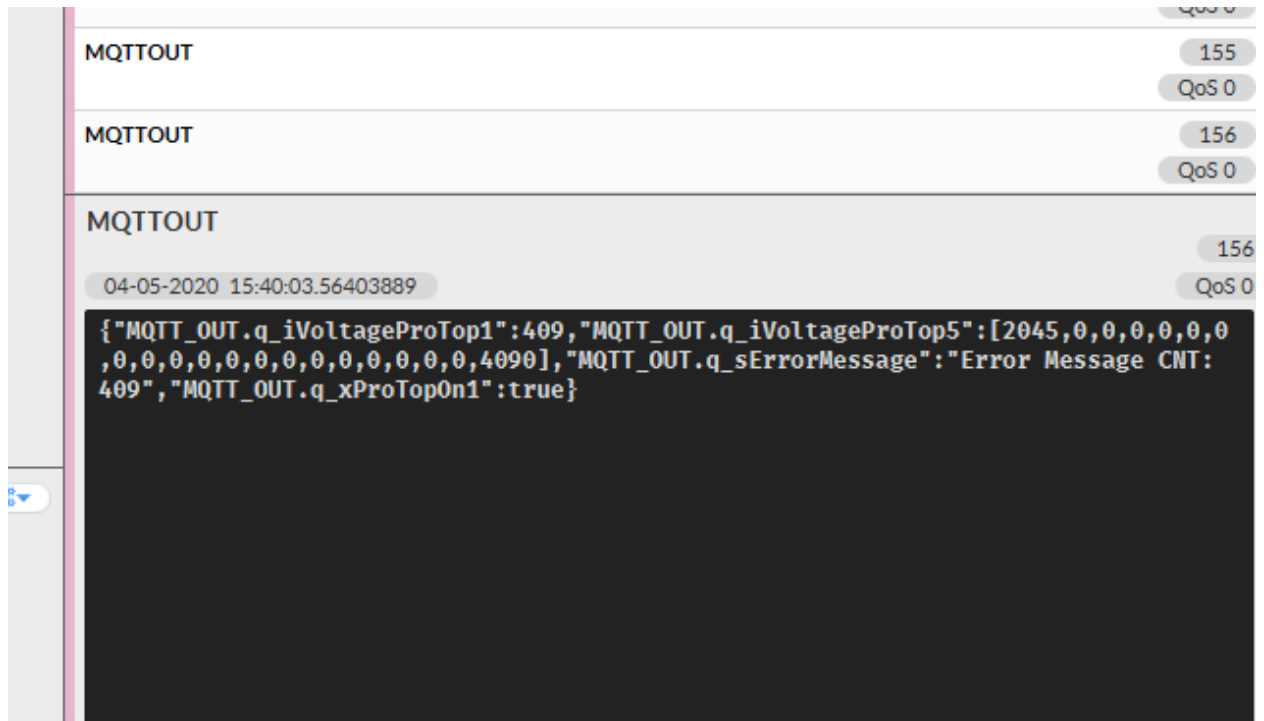
```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_publish:0.VariableTopicSet:0]
  topic="MQTTOUT"
  xRaw=0
  variable:0="MQTT_OUT.q_iVoltageProTop1"
  variable:1="MQTT_OUT.q_iVoltageProTop5"
  variable:2="MQTT_OUT.q_sErrorMessage"
  variable:3="MQTT_OUT.q_xProTopOn1"
```

```
[Application.WI_Cloud:0.WI_MQTT:0.WI_MQTT_subscribe:0]
  topic="WeidmuellerTestMQTTIN"
  xRaw=0
  variable:0="MQTT_IN.I_diTemperature"
  variable:1="MQTT_IN.I_iVoltage"
  variable:2="MQTT_IN.I_xRequestData"
  variable:3="MQTT_IN.I_sStatus"
```

- 5) Generate, Download the program and login to the device.
- 6) Open the tool MQTT.fx on your PC and connect to the MQTT broker.



- 7) Open the “Subscribe” tab, enter the subscribe topic “MQTTOUT” and now you should get messages from the controller.







- 8) To send data to the controller change to the tab “Publish” inside **MQTT.fx** and write a message in JSON format and push the “Publish” button.

Example:

```
{ "MQTT_IN.I_diTemperature":4258,"MQTT_IN.I_iVoltage":24,  
  "MQTT_IN.I_xRequestData" : true, "MQTT_IN.I_sStatus" : "Running the Test" }
```

If everything works fine you should see the values inside u-create studio.

Expression	Type	Value	Prepare
 I_diTemperature	DINT	4258	
 I_iVoltage	INT	24	
 I_xRequestData	BOOL	TRUE	
 I_sStatus	STRING	'Running the Test'	

6.1 Error handling

The MQTT service handles typical MQTT communication problems by retrieving or by transitioning into its error state. If you feel that the MQTT service exhibits unexpected behavior and you want to contact WI about it, please configure uiTraceLog to 4 and run the MQTT service through the error situation, again. The MQTT service provides its version and additional state information in the "Trace view" in u-create studio.

```
WI_MQTT 2021-06-08 16:19:00.000| Service has started. (rev. no.: 806)
WI_MQTT 2021-06-08 16:19:00.000| Service was built against OpenSSL version 3.0.0"-alpha17"
WI_MQTT 2021-06-08 16:19:00.000| Service has found OpenSSL version 3.0.0-alpha17
WI_MQTT 2021-06-08 16:19:00.000| This is an extended test prototype version and not for use in productive environments.
WI_MQTT 2021-06-08 16:19:00.000| Dissemination of this software is prohibited without express permission by Weidmüller
WI_MQTT 2021-06-08 16:19:00.000| Interface GmbH & Co. KG.
WI_MQTT 2021-06-08 16:19:00.000| external event: StartStateMachine
WI_MQTT 2021-06-08 16:19:00.000| state action: Entry initialize
WI_MQTT 2021-06-08 16:19:00.000| connecting to variable server..
```

The information you find here provides important help to WI developers. Please send an eMail to application.automation@weidmueller.com with the following information:

- description of the unexpected behavior
- description of the steps that lead to the unexpected behavior
- run the MQTT service with uiTraceLevel set to 4
- search the PLC's Trace output in u-create Studio for "WI_MQTT", copy all provided information and add it to your message
- add a copy of your expert entries to the message