Application Note

FESTO

CODESYS MQTT IIOT Library with Festo PLC

This document outlines the usage of the MQTT Client, MQTT Publish and MQTT subscribe function blocks from the MQTT CLIENT_SL library

MQTT

Title	Codesys MQTT IIOT Library with Festo PLC
Version	1.10
Document no.	100693
Original	en
Author	Festo
Last saved	13.09.2024

Copyright Notice

Legal Notice

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with components recommended by Festo SE & Co. KG.

Festo SE & Co. KG does not accept any liability for damages arising from the use of any incorrect or incomplete information contained in this documentation or any information missing therefrom.

Defects resulting from the improper handling of devices and modules are excluded from the warranty.

The data and information specified in this document should not be used for the implementation of safety functions relating to the protection of personnel and machinery.

No liability is accepted for claims for damages arising from a failure or functional defect. In other respects, the regulations with regard to liability from the terms and conditions of delivery, payment and use of software of Festo SE & Co. KG, which can be found at www.festo.com and can be supplied on request, shall apply.

All data contained in this document do not represent guaranteed specifications, particularly with regard to functionality, condition or quality, in the legal sense.

The information in this document serves only as basic information for the implementation of a specific, hypothetical application and is in no way intended as a substitute for the operating instructions of the respective manufacturers and the design and testing of the respective application by the user.

The operating instructions for Festo products can be found at www.festo.com .

Users of this document (application note) must verify that all functions described here also work correctly in the application. By reading this document and adhering to the specifications contained therein, users are also solely responsible for their own application.

Table of contents

1 Components/Software used

Type/Name	Version Software/Firmware
FESTO PLC	3.3.8-968ce3d31.20231019
CODESYS V3.5	SP18 Patch 4
Mosquitto Broker	2.0.18
MQTT Explorer	Version 0.40-beta6

Table 1.1: Components/Software used.

1.1 Overview Connectivity

For the development of this application note, a CPX-E PLC will be used, which will be connected to a laptop where the Mosquitto Broker has been installed and is running. Any other type of broker can also be utilized.

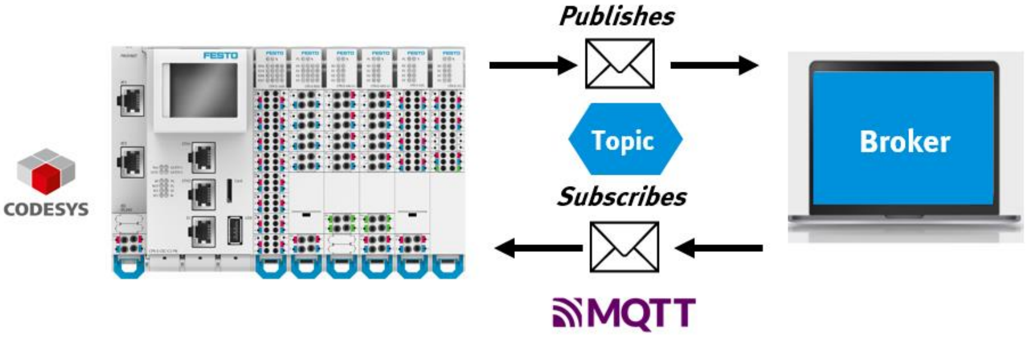
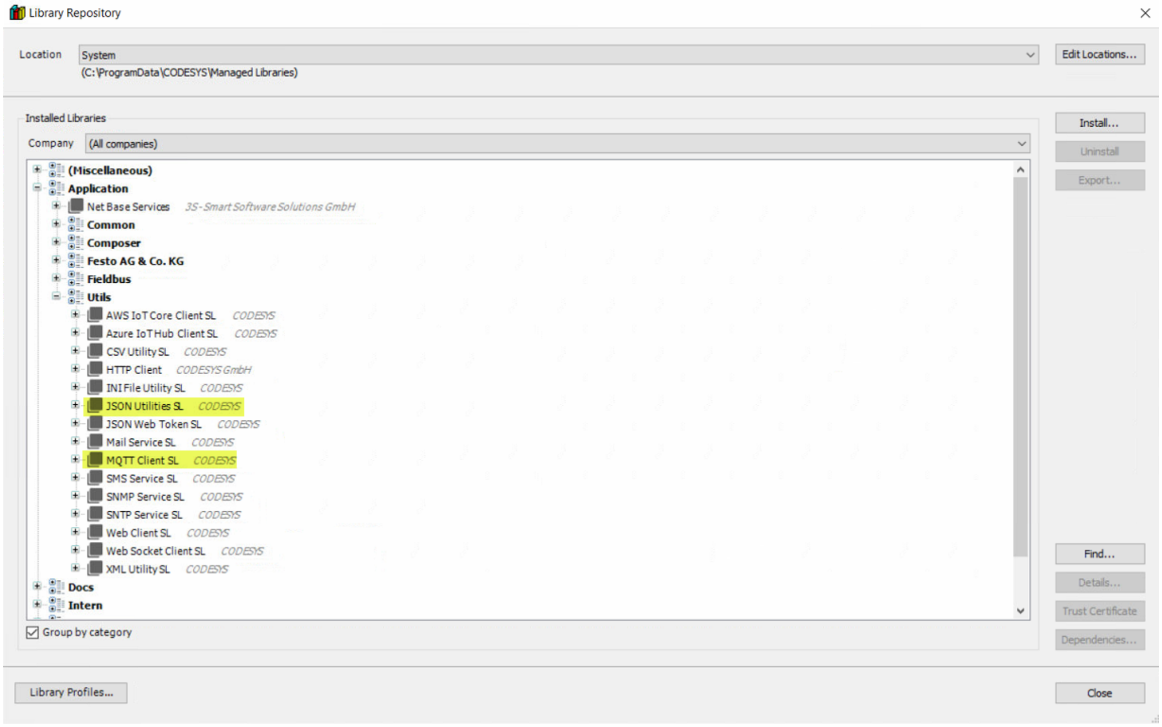


Figure 01: MQTT concept used throughout this application note.

2 Installation of the Codesys IIot Library.

The product IIoT Libraries SL includes numerous libraries to support various communication protocol such as MQTT, HTTPS… and provides tools for encoding and decoding data structures such as JSON. Additionally, components for direct communication with cloud services such as AWS or Azure are also provided. This application note focuses on the MQTT Client SL library, and the function blocks related to JSON will be used in chapter 3.5.



Library Repository

Location: System (C:\ProgramData\CODESYS\Managed Libraries)

Installed Libraries

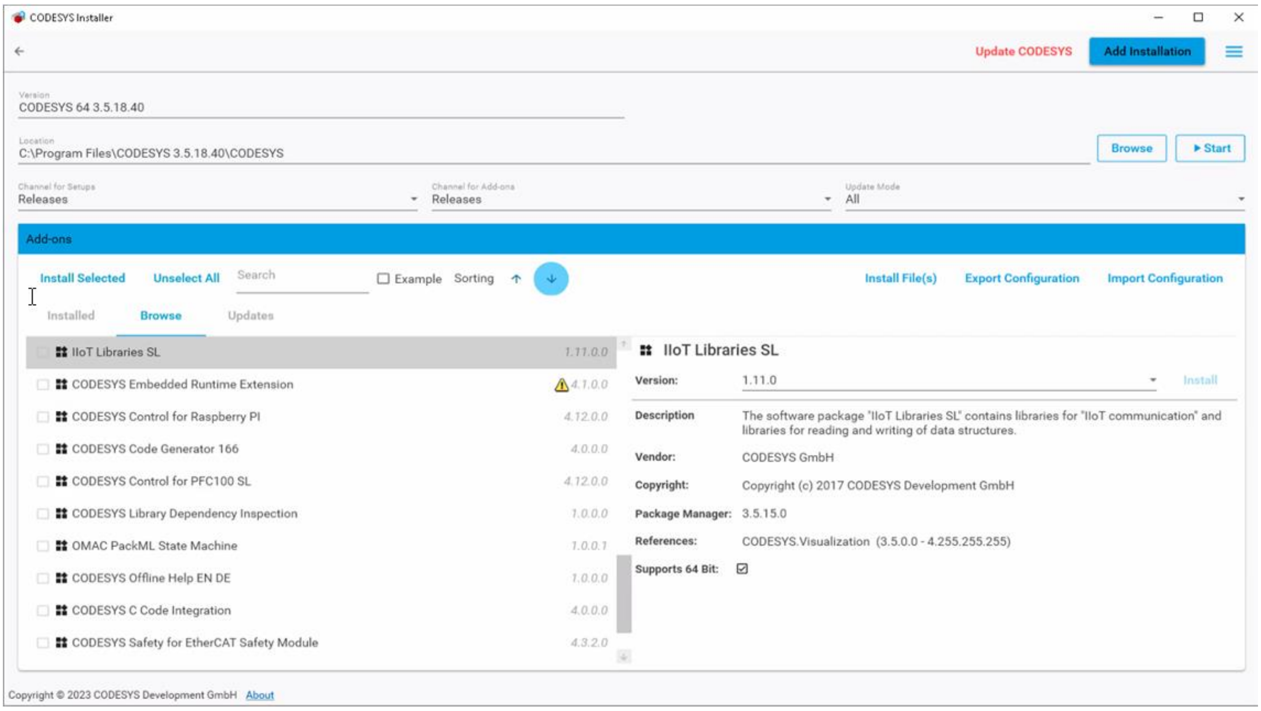
Company: (All companies)

- (Miscellaneous)
- Application
  - Net Base Services 3S-Smart Software Solutions GmbH
  - Common
  - Composer
  - Festo AG & Co. KG
  - Fieldbus
  - Utils
    - AWS IoT Core Client SL CODESYS
    - Azure IoT Hub Client SL CODESYS
    - CSV Utility SL CODESYS
    - HTTP Client CODESYS GmbH
    - INI File Utility SL CODESYS
    - JSON Utilities SL CODESYS
    - JSON Web Token SL CODESYS
    - Mail Service SL CODESYS
    - MQTT Client SL CODESYS
    - SMS Service SL CODESYS
    - SNMP Service SL CODESYS
    - SNTP Service SL CODESYS
    - Web Client SL CODESYS
    - Web Socket Client SL CODESYS
    - XML Utility SL CODESYS
- Docs
- Intern

☑ Group by category

Figure 02: CODESYS IIoT Libraries SL.

With the introduction of the CODESYS Installer, with an existing internet connections can simply install the latest version of the IIoT libraries in the CODESYS development system without having to create an account with CODESYS.



CODESYS Installer

Version: CODESYS 64 3.5.18.40

Location: C:\Program Files\CODESYS 3.5.18.40\CODESYS

Channel for Setups: Releases

Channel for Add-ons: Releases

Update Mode: All

Add-ons

☐	IIoT Libraries SL	1.11.0.0
☐	CODESYS Embedded Runtime Extension	4.1.0.0
☐	CODESYS Control for Raspberry PI	4.12.0.0
☐	CODESYS Code Generator 166	4.0.0.0
☐	CODESYS Control for PFC100 SL	4.12.0.0
☐	CODESYS Library Dependency Inspection	1.0.0.0
☐	OMAC PackML State Machine	1.0.0.1
☐	CODESYS Offline Help EN DE	1.0.0.0
☐	CODESYS C Code Integration	4.0.0.0
☐	CODESYS Safety for EtherCAT Safety Module	4.3.2.0

IIoT Libraries SL

Version: 1.11.0

Description: The software package "IIoT Libraries SL" contains libraries for "IIoT communication" and libraries for reading and writing of data structures.

Vendor: CODESYS GmbH

Copyright: Copyright (c) 2017 CODESYS Development GmbH

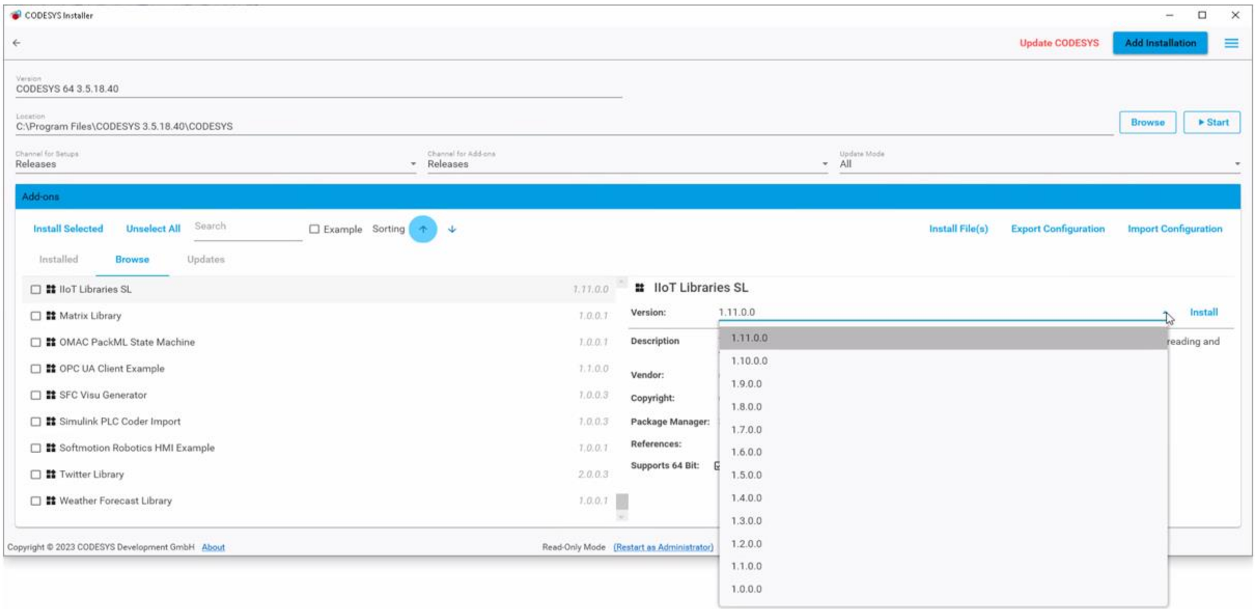
Package Manager: 3.5.15.0

References: CODESYS.Visualization (3.5.0.0 - 4.255.255.255)

Supports 64 Bit: ☑

Figure 03: CODESYS Installer.

Additionally, there is also the option to install any version of the library.



Version: CODESYS 64 3.5.18.40

Location: C:\Program Files\CODESYS 3.5.18.40\CODESYS

Channel for Setups: Releases

Channel for Add-ons: Releases

Update Mode: All

□ IIoT Libraries SL	1.11.0.0
□ Matrix Library	1.0.0.1
□ OMAC PackML State Machine	1.0.0.1
□ OPC UA Client Example	1.1.0.0
□ SFC Visu Generator	1.0.0.3
□ Simulink PLC Coder Import	1.0.0.3
□ Softmotion Robotics HMI Example	1.0.0.1
□ Twitter Library	2.0.0.3
□ Weather Forecast Library	1.0.0.1

IIoT Libraries SL

Version: 1.11.0.0

1.11.0.0
1.10.0.0
1.9.0.0
1.8.0.0
1.7.0.0
1.6.0.0
1.5.0.0
1.4.0.0
1.3.0.0
1.2.0.0
1.1.0.0
1.0.0.0

Description … reading and

Vendor:

Copyright:

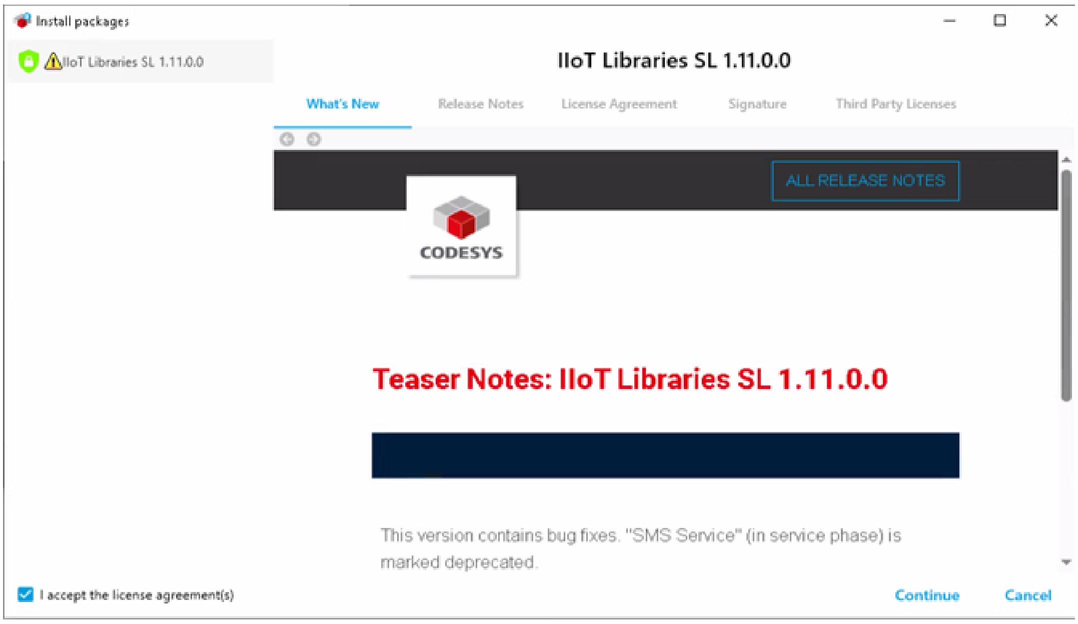
Package Manager:

References:

Supports 64 Bit:

Figure 04: Codesys Installer – versions of the IIoT library.

Please follow the steps indicated by the Codesys installer.



IIoT Libraries SL 1.11.0.0

Teaser Notes: IIoT Libraries SL 1.11.0.0

This version contains bug fixes. "SMS Service" (in service phase) is marked deprecated.

☑ I accept the license agreement(s)

Figure 05: Codesys Installer – installation of the IIoT library.

2.1 Include the MQTT_Client library in the project.

The MQTT library is included in the “Library Manager” of the project in the following manner:

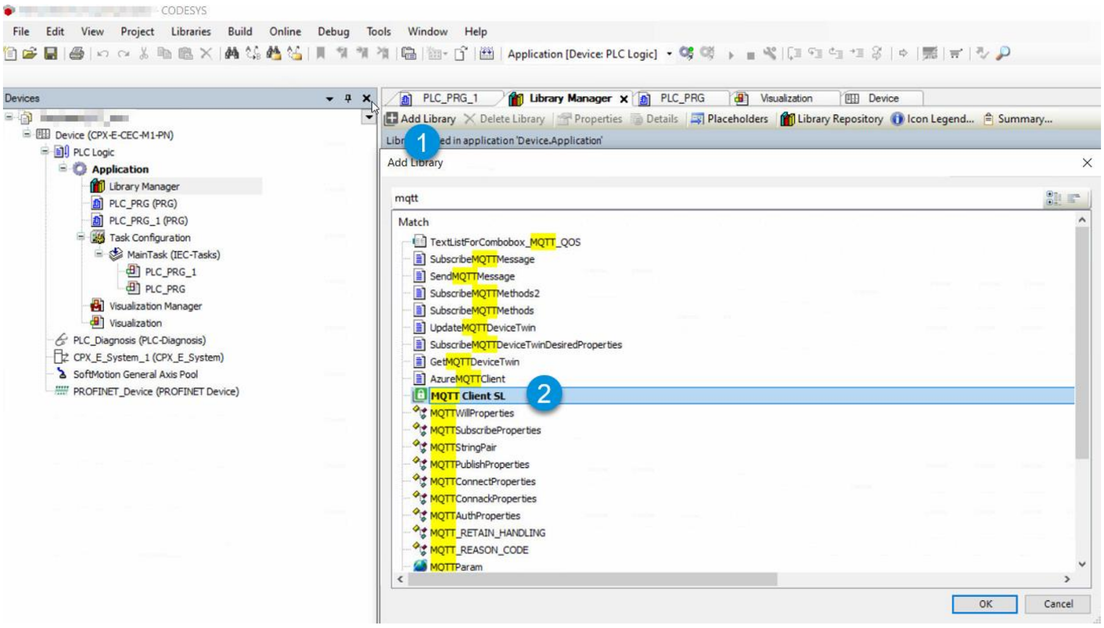
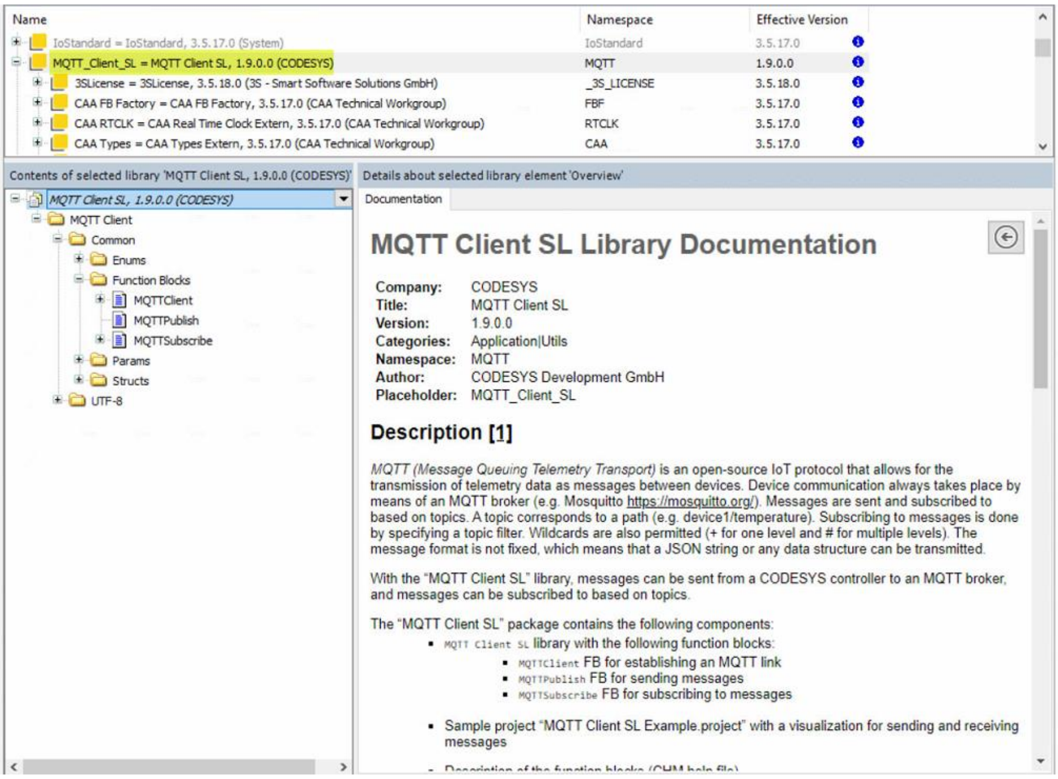


Figure 06: Include MQTT_Client_SL library.

The library has 3 function block that will be detailed in the next chapter.



Name	Namespace	Effective Version
IoStandard = IoStandard, 3.5.17.0 (System)	IoStandard	3.5.17.0
MQTT_Client_SL = MQTT Client SL, 1.9.0.0 (CODESYS)	MQTT	1.9.0.0
3SLicense = 3SLicense, 3.5.18.0 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.18.0
CAA FB Factory = CAA FB Factory, 3.5.17.0 (CAA Technical Workgroup)	FBF	3.5.17.0
CAA RTCLK = CAA Real Time Clock Extern, 3.5.17.0 (CAA Technical Workgroup)	RTCLK	3.5.17.0
CAA Types = CAA Types Extern, 3.5.17.0 (CAA Technical Workgroup)	CAA	3.5.17.0

Contents of selected library 'MQTT Client SL, 1.9.0.0 (CODESYS)'
MQTT Client SL, 1.9.0.0 (CODESYS)
MQTT Client
Common
Enums
Function Blocks
MQTTClient
MQTTPublish
MQTTSubscribe
Params
Structs
UTF-8
Details about selected library element 'Overview'
Documentation

MQTT Client SL Library Documentation

Company:	CODESYS
Title:	MQTT Client SL
Version:	1.9.0.0
Categories:	Application|Utils
Namespace:	MQTT
Author:	CODESYS Development GmbH
Placeholder:	MQTT_Client_SL

Description [1]

MQTT (Message Queuing Telemetry Transport) is an open-source IoT protocol that allows for the transmission of telemetry data as messages between devices. Device communication always takes place by means of an MQTT broker (e.g. Mosquitto https://mosquitto.org/). Messages are sent and subscribed to based on topics. A topic corresponds to a path (e.g. device1/temperature). Subscribing to messages is done by specifying a topic filter. Wildcards are also permitted (+ for one level and # for multiple levels). The message format is not fixed, which means that a JSON string or any data structure can be transmitted.

With the “MQTT Client SL” library, messages can be sent from a CODESYS controller to an MQTT broker, and messages can be subscribed to based on topics.

The “MQTT Client SL” package contains the following components:

- `MQTT Client SL` library with the following function blocks:
  - `MQTTClient` FB for establishing an MQTT link
  - `MQTTPublish` FB for sending messages
  - `MQTTSubscribe` FB for subscribing to messages
- Sample project “MQTT Client SL Example.project” with a visualization for sending and receiving messages
- Description of the function blocks (CHM help file)

Figure 07: MQTT_Client_SL library.

3 Definition of the MQTT FBs

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices. Codesys IIoT MQTT library provides three function blocks: MQTT_Client, MQTT_Publish and MQTT_Subscribe.

3.1 MQTT Client

This Function Block connects to the MQTT broker.

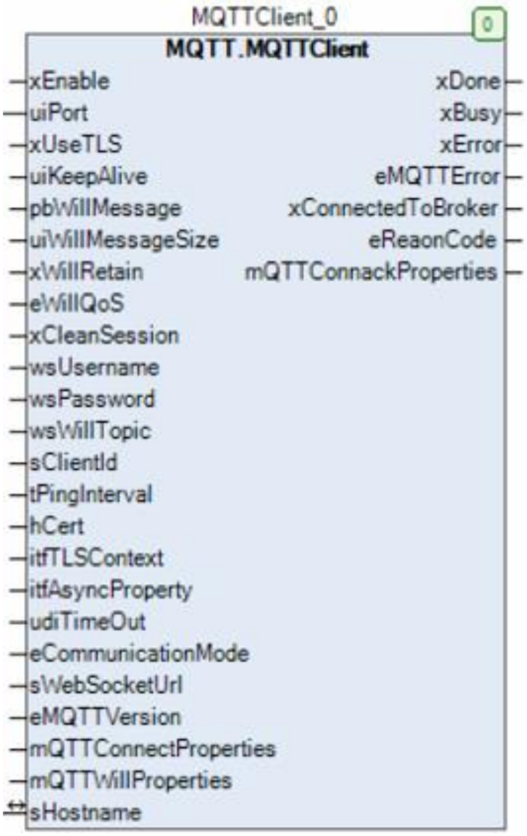


Figure 08: MQTT Client Function block.

3.1.1 Inputs

Tag Name	Data Type	Function Description
xEnable	BOOL	**TRUE –** Activates the defined operation. **FALSE –** Aborts/reset the defined operation.
uiPort	UINT	Port of MQTT Broker Server by default is 1883.
xUseTLS	BOOL	**TRUE –** Crypted Connection. **FALSE –** Encrypted Connection.
uiKeepAlive	UINT	Keep Alive Time in Seconds (optional). Initial = 5.
pbWillMessage	POINTER TO BYTE	Pointer to the 'Last Will' message (optional).
uiWillMessgeSize	UINT	Size of 'Last Will' message (optional).
xWillRetain	BOOL	TRUE: Saves the 'Last Will' message on server. If Client subscribes later, then it receives the last stack message from server.
eWillQoS	MQTT_QOS	QoS Level of 'Last Will' message.
xCleanSession	BOOL	TRUE: Creates a new session, FALSE: Uses already. existing session if available.
wsUsernamme	WSTRING(255)	User Name (optional).
wsPassword	WSTRING(1024)	User Password (optional).
wsWillTopic	STRING(1024)	Will Topic (optional).
sClientId	STRING(255)	Client ID, if empty then new ID is generated, allowed are only these characters *0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ* (optional).
tPingInterval	TIME	Interval time in seconds how often should be pinged, if Time<=0 then no pings
hCert	RTS_IEC_HANDLE	Handle to the client certificate (optional)
itTLSContext	NBS.ITLSContext	
itfAsyncProperty	NBS.IAsyncProperty	Runs the connect process in a own background task. Use this property if the connection setup takes longer than one task cycle (e.g. TLS connections)
udfTimeOut	UDINT	Defines the time (µs) after which the connection setup aborts with xError.
eCommunication-Mode	COMMUNICATION_MODE	Communication mode: TCP or WEB_SOCKET, default: TCP COMMUNICATION_MODE.TCP: Configure the connection via the inputs: sHostname, uiPort and itfTLSContext COMMUNICATION_MODE.WEB_SOCKET: Configure the connection via the inputs: sWebSocketUrl and itfTLSContext Additional WebSocket options can be set via the method SetWebSocketOptions.
sWebSocektUrl	REFERENCE TO STRING (1024)	The url of the websocket server. e.g 'ws://localhost:8080' ws-URI = "ws:" "//" host [":" port] path ["?" query] wss-URI = "wss:" "//" host [":" port] path ["?" query] The input is only relevant for eCommunicationMode =COMMUNICATION_MODE.WEB_SOCKET.
eMQTTVersion	MQTT_VERSION	MQTT protocol version, default 3.1.1 Available v3.1.1.1 and v5.

Tag Name	Data Type	Function Description
mQTTConnectProperties	REFERENCE TO MQTT-ConnectProperties	Connection properties for MQTT v5.
mQTTWillProperties	REFERENCE TO MQTT-WillProperties	Will properties for MQTT v5.

3.1.2 Outputs

Tag Name	Data Type	Function Description
xDone	BOOL	**TRUE**: Ready condition reached.
xBusy	BOOL	**TRUE**: Operation is running.
xError	BOOL	**TRUE**: Error condition reached.
eMQTTError	MQTT_ERROR	MQTT Error Type
xConnectedBroker	BOOL	Inform the use that the connection to the broker has been successfully established.
eReasonCode	MQTT_REASON_CODE	Reason code of the last operation (MQTT V5 only). For more information, refer to the library's help documentation
mqttConnackProperties	MQTTConnackProperties	Connection response properties (MQTT V5 only). For more information, refer to the library's help documentation.

3.1.2.1 MQTT_ERROR

Name	Comment
NO_ERROR	No error
TCP_INIT_ERROR	Unable to initialize the TCP Socket.
TCP_READ_ERROR	Error whcile reading response
TCP_WRITE_ERROR	Error while sending the request.
MAX_RESPONSE_SIZE_EXCEEDED	Size of incoming packet exceeds the maximum packet size. See MQTTParam.g_udiMaxPacketSize
DECODE_REMAINING_LENGTH_MALFORMED	Malformation during Decoding of Remaining Length of Packet.
RESPONSE_PACKET_EMPTY	The Response Packet is empty.
INVALID_PACKET_TYPE	Invalid Packet Type in first byte of Fixed Header.
INVALID_PACKET_BIT_FLAGS	Invalid Packet Bit Flags in first byte of Fixed Header
INVALID_PACKET	Invalid Packet
KEEP_ALIVE_TIME_EXCEEDED	Keep Alive Time is too much.

Name	Comment
WRONG_SESSION_PRESENT_CONNACK	Wrong Session Present in CONNACK packet.
UNACCEPTABLE_PROTOCOL_VERSION	Connection to Broker is Refused, because of Unacceptable Protocol Version.
IDENTIFIER_REJECTED	Connection to Broker is Refused, because of Rejection of Client Identifier.
SERVER_UNAVAILABLE	Connection to Broker is Refused, because of Bad User name or Bad Password.
BAD_USER_NAME_PASSWORD	Connection to Broker is Refused, because of Not Authorized access.
NOT_AUTHORIZED	Connection to Broker is Refused, because of Not Authorized access.
TOPIC_FILTER_EMPTY	Topic Filter is empty “”, MUST be at least one character long.
TOPIC_NAME_NOT_ALLOWED_WILDCARD	Topic Name contains Wildcards, what is not allowed. Only Topic Filter can contain these.
TOPIC_INVALID_LENGTH	Topic Length out of valid range (valid range: 1 - 1024).
TOPIC_IS_EMPTY	Topic name is empty.
SUBSCRIBE_FAILURE	Subscription Failure.
ADD_MQTT_PACKET_COLLECTION_ERROR	Collection Error while trying to add a MQTT packet to the stack.
ADD_SUBSCRIBER_COLLECTION_ERROR	Collection Error while trying to add a subscriber to the stack.
REMOVE_SUBSCRIBER_COLLECTION_ERROR	Collection Error while trying to remove a subscriber from the stack.
ACKNOWLEDGE_TIMEOUT	Client waits for ping response from Server, but Server does not response within a given time interval (2*ping interval).
ALLOCATED_PAYLOAD_SIZE_EXCEEDED	The Size of the received payload is more than give allocated memory.
MAX_NUMBER_OF_PACKETS_EXCEEDED	The maximum size of packets has been exceeded.
CAN_NOT_ADD_ELEMENT_TO_QUEUE	Can not add the element to queue (maybe the maximum size has been exceeded).
QUERYINTERFACE_ERROR	Call of function QUERYINTERFACE failed (Internal error).

Name	Comment
TIME_OUT	Action returned.
INVALID_LICENSE	No valid license found, or demo mode has expired.
CLIENT_NOT_CONNECTED	The MQTT client is not connected to a broker.
RESOLVE_HOSTNAME_FAILED	The hostname cannot be resolved.
MAX_REQUEST_SIZE_EXCEEDED	Size of publish packet exceeds the maximum packet size.
UNSUPPORTED_VERSION	Unsupported MQTT version.
OPERATION_NOT_SUCCESSFUL	The operation was not successful. See eReasonCode for more information (MQTT version 5).
SEND_QUOTA_LIMIT_REACHED	The send quota limit has been reached. Try to publish the message later.
INVALID_REASON_CODE	The reason code is not allowed in this context.
MAX_RECEIVE_BUFFER_SIZE_EXCEEDED	Size of incoming packets exceeds the maximum size of the receive buffer. See MQTTParam.g_udiMaxReceiveBufferSize.
MAX_TOPIC_LEVEL_EXCEEDED	The maximum topic level has been exceeded.
MAX_STRING_LENGTH_EXCEEDED	The maximum length of a STRING has been exceeded (Max length: 1024).
MAX_CORRELATION_DATA_LENGTH_EXCEEDED	The maximum size of Correlation Data has been exceeded. See MQTT-Param.g_udiMaxcorrelationDataSize.
MAX_NUMBER_OF_PUBLISHER_AND_SUBSCRIBER_EXCEEDED	The maximum number of publishers and subscribers has been exceeded. See MQTTParam.g_udiMax-PubliserhsAndSubscribers.

3.1.3 Inputs/Outputs

Tag Name	Data Type	Function Description
xHostname	STRING(80)	Host name of MQTT Broker Server. Can be an IP Address or a URL.

3.2 MQTT Publish

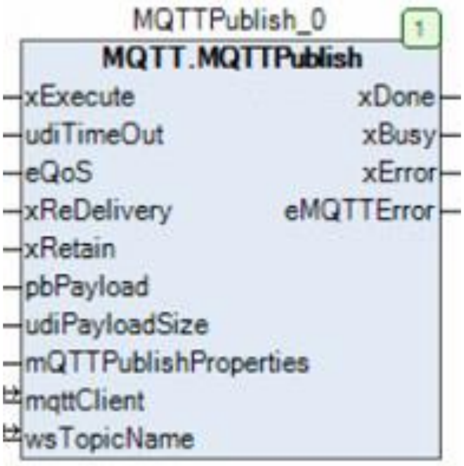


Figure 09: MQTT Publish Function block.

3.2.1 Inputs

Tag Name	Data Type	Function Description
xExecute	BOOL	**Rising Edge –** Start defined operation: Publish a payload message **FALSE –** Resets the defined operation after ready conditioned was reached.
udiTimeOut	UINT	Max. operating time for executing [us], 0: No operation time limit.
eQos	BOOL	Quality Of Service level of message.
xReDelivery	BOOL	**TRUE –** When packet got already send and now should got resend. **FALSE –** First time**.**
xRetain	BOOL	**TRUE –** Stores the message on Server
pbPayload	POINTER TO BYTE	Pointer to the message.
udiPayloadSize	UDINT	Size of the payload.
mQTTPublishProperties	REFERENCE TO MQTT-PublishProperties	Optional publish properties (MQTT V5 only).

3.2.2 Outputs

Tag Name	Data Type	Function Description
xDone	BOOL	Ready condition reached.
xBusy	BOOL	Operation is running.
xError	BOOL	Error condition reached.
eMQTTError	MQTT_ERROR	MQTT Error type.

3.2.3 Inputs/Outputs

Tag Name	Data Type	Function Description
mqttClient	MQTTCLIENT	Instance of the MQTT Client function block.
wsTopicName	WSTRING(1024)	Topic Name of the message.
xDone	BOOL	Ready condition reached.
xBusy	BOOL	Operation is running.
xError	BOOL	Error condition reached.
eMQTTError	MQTT_ERROR	MQTT Error type.

3.3 MQTT Subscribe

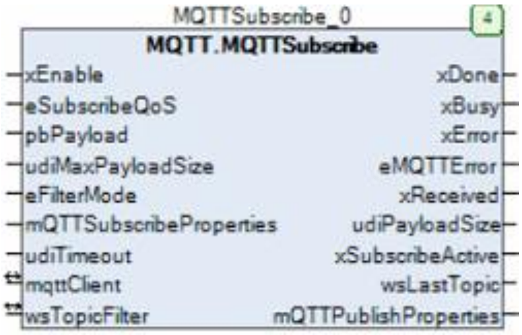


Figure 10: MQTT Subscribe Function block.

3.3.1 Inputs

Tag Name	Data Type	Function Description
xEnable	BOOL	**TRUE –** Activates the defined operation. **FALSE –** Aborts/resets the defined operation.
eSubscribeQoS	MQTT_QOS	Quality of Service Level for Subscribe operation.
pbPayload	POINTER TO BYTE	Pointer to memory for incoming payload.
udiMaxPayloadSize	UDINT	Maximum size of incoming payload.
eFilterMode	FILTER MODE	Filter mode for incoming telegrams. Default: *FILTER_MODE.FILTER_ON*
mQTTSubscribeProperties	REFERENCE TO MQTT Subscribe Properties	Subscribe properties (MQTT version 5 only).
udiTimeout	UDINT	Time in μ_s in which an subscription acknowledgment (SUBACK) must be received.

3.3.2 Outputs

Tag Name	Data Type	Function Description
xReceived	MQTT ERROR	**TRUE –** If a message is received in the new cycle, otherwise False. **FALSE –** as long as there is no message.
udiPayloadSize	UDINT	**TRUE –** Subscriber is listening. **FALSE –** When there is no active subscription.
xSubscribeActive	BOOL	**TRUE –** There is connection to the MQTT Broker. **FALSE –** Otherwise false.
wsLastTopic	WSTRING (1024)	The real topic value from the publish packet that is corresponding to this topic filter.
mQTTPublishProperties	MQTTPublishProperties	Last received publish properties.

3.3.3 Inputs/Outputs

Tag Name	Data Type	Function Description
mqttClient	MQTT CLIENT	Function block MQTT Client
wsTopicFilter	WSTRING(1024)	Topic Filter

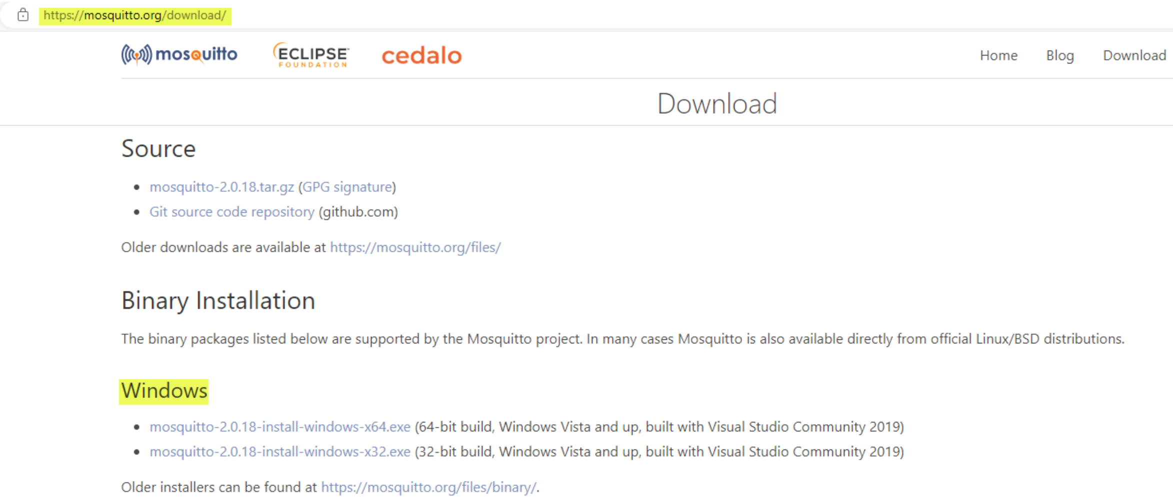
3.4 Example of MQTT Publish and Subscribe.

In this part of the chapter 3, a practical example is shown for publishing and subscribing to MQTT messages.

3.4.1 Configure Mosquitto MQTT Broker

First of all, the MQTT Broker will be configured. It will be used Mosquitto as a MQTT Broker. It is worth noting that any other type of Broker could also be used. The MQTT Broker is running on a laptop with a Windows operating system.

Open a browser and write: https://mosquitto.org/download/ and choose correct package depending on your operating system.



Download

Source

- mosquitto-2.0.18.tar.gz (GPG signature)
- Git source code repository (github.com)

Older downloads are available at https://mosquitto.org/files/

Binary Installation

The binary packages listed below are supported by the Mosquitto project. In many cases Mosquitto is also available directly from official Linux/BSD distributions.

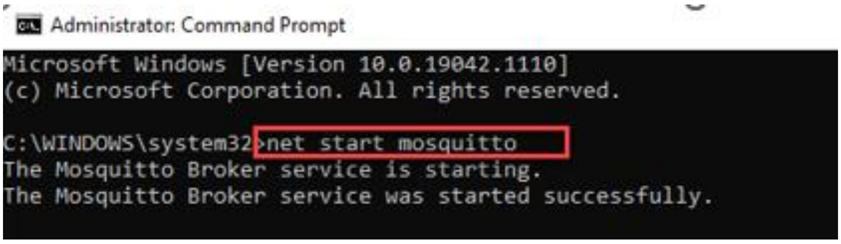
Windows

- mosquitto-2.0.18-install-windows-x64.exe (64-bit build, Windows Vista and up, built with Visual Studio Community 2019)
- mosquitto-2.0.18-install-windows-x32.exe (32-bit build, Windows Vista and up, built with Visual Studio Community 2019)

Older installers can be found at https://mosquitto.org/files/binary/.

Figure 11: Mosquitto support portal web link.

As soon as you installed the Mosquitto Broker package go to command prompt with Admin right and type “ *net start mosquitto*”. In the same way, to stop the mosquito services, type the following command “*net stop mosquitto*”.



```
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>net start mosquitto
The Mosquitto Broker service is starting.
The Mosquitto Broker service was started successfully.
```

Figure 12: command start mosquito services.

3.4.2 Example of publishing a message payload using the MQTT publish Function Block.

The aim of this chapter is to provide a practical example of publishing a message payload. The PLC CPX-E-CEC-xx is the client and it will publish the message. The Broker will received it, and if any other client is subscribed to the topic, the broker will forward it to them.

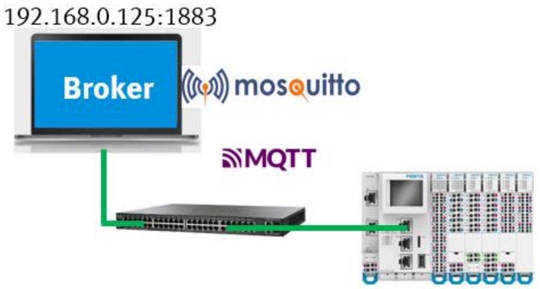
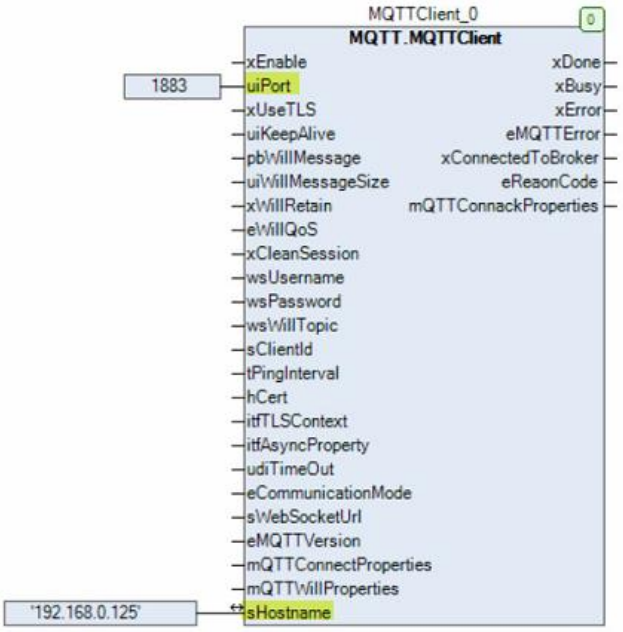


Figure 13: Network environment – Broker and Festo PLC.

3.4.2.1 Configuration of the MQTT Client Function Block.

The MQTT client will be configured to connect with the Broker. The connection from the client to the Broker (MQTTClient) must be active whether messages are published or the client subscribes to any message.

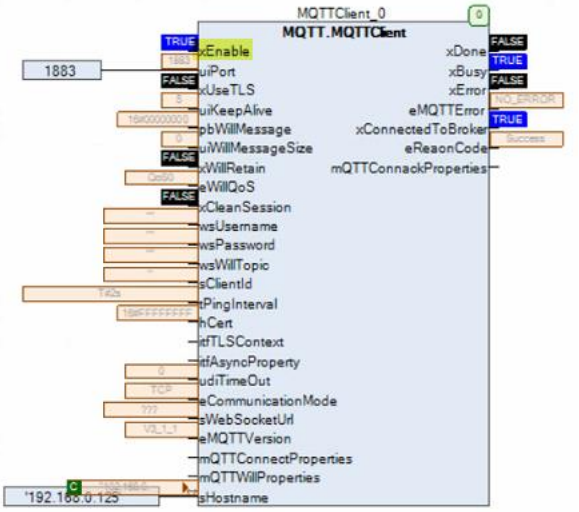
No.	Action
1	Configure the uiPort = 1883 by default MQTT port.
2	Write the IP Address of your Broker.
3	Enable the function clock (xEnable input).



1883 → uiPort

'192.168.0.125' → sHostname

Figure 14: MQTT Client FB configuration.

No.	Action
4	Result: The correct status of the outputs are --› **xBusy := True** **xConnectedToBroker := True**
5	

3.4.2.2 Configuration of the MQTT Publish Function Block.

Once the MQTT Client function block has been configured and connected successfully to the Broker, the next step is to publish a message payload. The function block (MQTT Publish) is configured with the minimum required settings.

No.	Action

6	Configure the following inputs:
	pPayload: STRING;
	TopicName: WSTRING(1024);
	MQTTClient_0: MQTT.MQTTClient;

7	It is a must to parametrize the Payload size. One way to automatically determine the size of the message payload is by using the LEN function. For further information, please refer to the following Codesys link: [LEN (FUN) (helpme-codesys.com)](helpme-codesys.com) Returns the number of characters of a string.

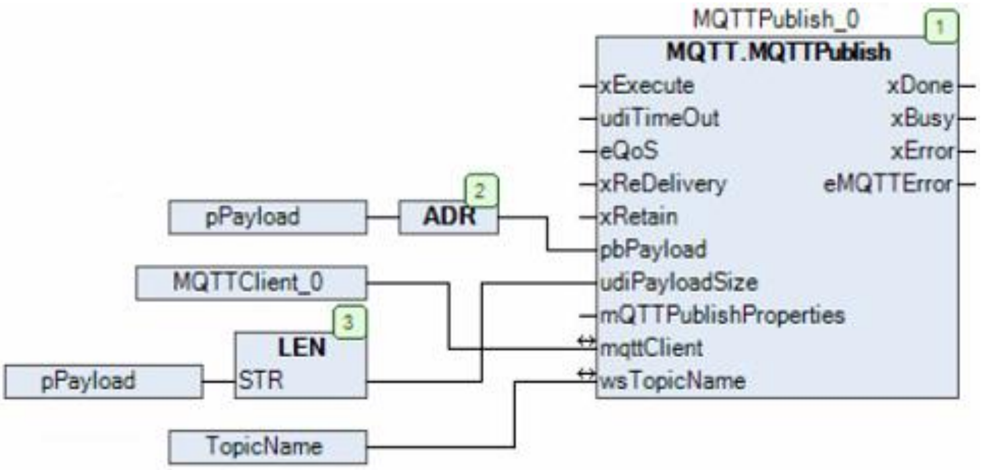
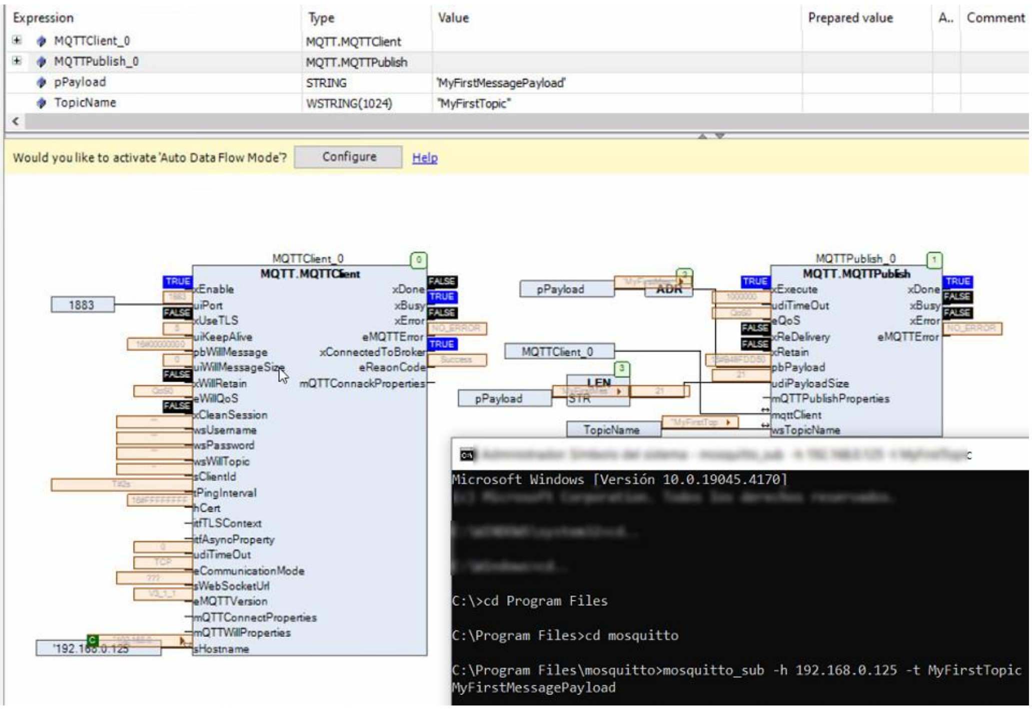
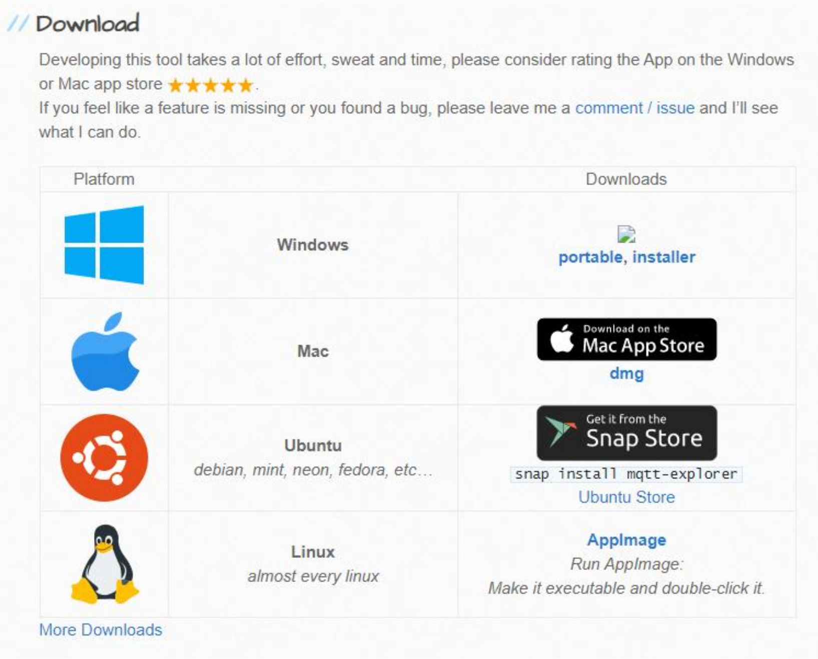
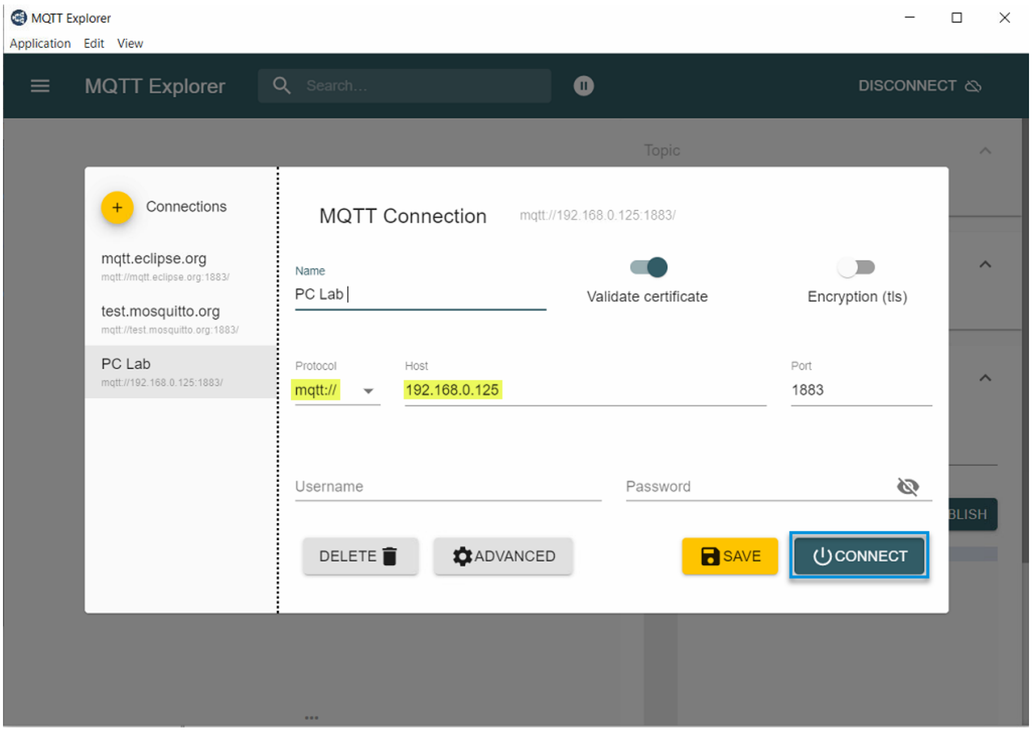
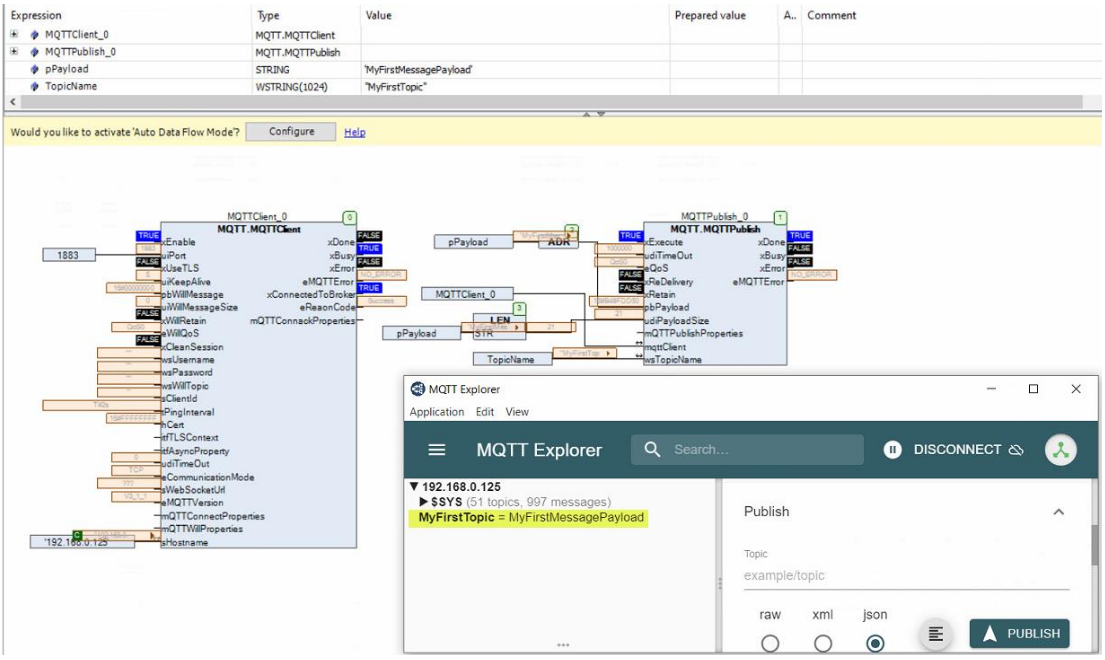


Figure 15: MQTT Publish FB configuration.

No.	**Action**
8	The MQTT Publish block publishes the message and the Broker receives it from the client.
9	The correct status of the outputs are --> **xDone=True** and **eMQTTError = No error!**
10	To verify that it has been sent successfully, we can subscribe to the topic. To perform this task: by using the command prompt or by using additional software such as MQTT explorer.
11	Open command prompt and navigate to the directory where your Mosquitto Broker is installed. After this, type the following command: *mosquitto_sub -h 192.168.0.125 -t MyTopicName*
12	 Expression | Type | Value | Prepared value | A.. | Comment MQTTClient_0 | MQTT.MQTTClient MQTTPublish_0 | MQTT.MQTTPublish pPayload | STRING | 'MyFirstMessagePayload' TopicName | WSTRING(1024) | "MyFirstTopic" Would you like to activate 'Auto Data Flow Mode'? Configure Help Microsoft Windows [Versión 10.0.19045.4170] C:\>cd Program Files C:\Program Files>cd mosquitto C:\Program Files\mosquitto>mosquitto_sub -h 192.168.0.125 -t MyFirstTopic MyFirstMessagePayload MQTTClient_0 MQTT.MQTTClient xEnable uiPort xUseTLS uiKeepAlive pbWillMessage uiWillMessageSize xWillRetain eWillQoS xCleanSession wsUsername wsPassword wsWillTopic sClientId tPingInterval hCert itfTLSContext itfAsyncProperty udiTimeOut eCommunicationMode sWebSocketUrl eMQTTVersion mQTTConnectProperties mQTTWillProperties sHostname xDone xBusy xError eMQTTError xConnectedToBroker eReasonCode mQTTConnackProperties 1883 '192.168.0.125 TRUE FALSE NO_ERROR Success pPayload ADR LEN STR 21 MQTTClient_0 TopicName MQTTPublish_0 MQTT.MQTTPublish xExecute udiTimeOut eQoS xReDelivery xRetain pbPayload udiPayloadSize mQTTPublishProperties mqttClient wsTopicName xDone xBusy xError eMQTTError

No.	Action
13	Open a Web Browser and please download MQTT Explorer. MQTT Explorer is a comprehensive MQTT client that provides a structured overview of your MQTT topics and makes working with devices/services on your broker dead-simple. http://mqtt-explorer.com Please select exe file according to your operating system.
14	 // Download Developing this tool takes a lot of effort, sweat and time, please consider rating the App on the Windows or Mac app store ★★★★★. If you feel like a feature is missing or you found a bug, please leave me a comment / issue and I'll see what I can do. Platform | Downloads Windows | portable, installer Mac | Download on the Mac App Store, dmg Ubuntu, debian, mint, neon, fedora, etc… | Get it from the Snap Store, `snap install mqtt-explorer`, Ubuntu Store Linux, almost every linux | AppImage, Run AppImage: Make it executable and double-click it. More Downloads

15	Write the IP address of the Mosquitto Broker and select **MQTT** as **protocol**. Finally press **connect**.
16	 MQTT Connection mqtt://192.168.0.125:1883/ Connections: mqtt.eclipse.org (mqtt://mqtt.eclipse.org:1883/), test.mosquitto.org (mqtt://test.mosquitto.org:1883/), PC Lab (mqtt://192.168.0.125:1883/) Name: PC Lab Validate certificate Encryption (tls) Protocol: mqtt:// Host: 192.168.0.125 Port: 1883 Username Password DELETE ADVANCED SAVE CONNECT
17	 Expression | Type | Value | Prepared value | A.. | Comment MQTTClient_0 | MQTT.MQTTClient MQTTPublish_0 | MQTT.MQTTPublish pPayload | STRING | 'MyFirstMessagePayload' TopicName | WSTRING(1024) | "MyFirstTopic" Would you like to activate 'Auto Data Flow Mode'? Configure Help MQTT Explorer: ▼ 192.168.0.125 ▶ $SYS (51 topics, 997 messages) MyFirstTopic = MyFirstMessagePayload Publish Topic: example/topic raw ○ xml ○ json ● PUBLISH

3.4.3 Example of subscribing to a topic using the MQTT Subscribe Function Block.

The aim of this chapter is to provide a practical example of subscribing to a topic. The PLC CPX-E-CEC-xx acts as the client and will subscribe to a topic to receive a message payload. The PLC will be subscribed to the topic and the broker will send the message payload as soon as it becomes available. If there are any changes in the message payload, the broker will notify and update the data.

3.4.3.1 Configuration of the MQTT Subscribe Function Block.

To subscribe to a message, a connection with the configuration of the MQTT Client is a must and it is described in chapter 2.4.2.1.

No.	Action

18	Configure the following inputs:
	pPayload2:STRING;
	MQTT_Client_0:MQTT.MQTTClient;
	udiMaxPayloadSize:50; *//Configure the maximum size of the payload. The configured value must be greater than the size of the received payload, otherwise parts of the message will be lost.*
	TopicFilter:WSTRING(1024);

19	***udiMaxPayloadSize*** can be configured with a constant value, but it must be ensured that the received payload does not exceed the configured maximum value.

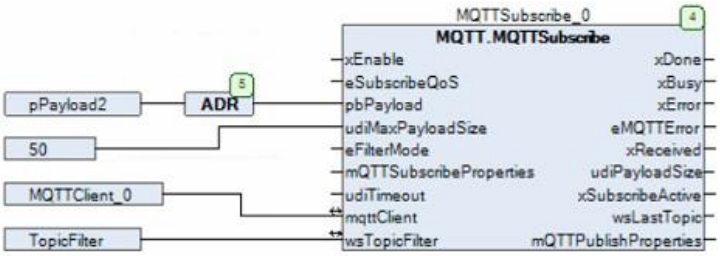
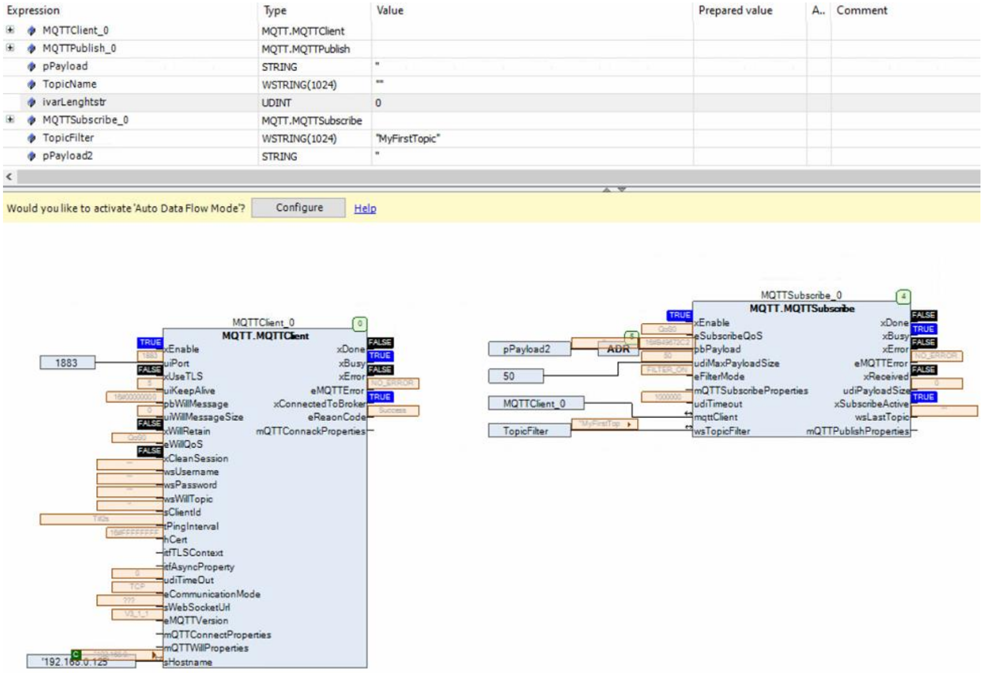


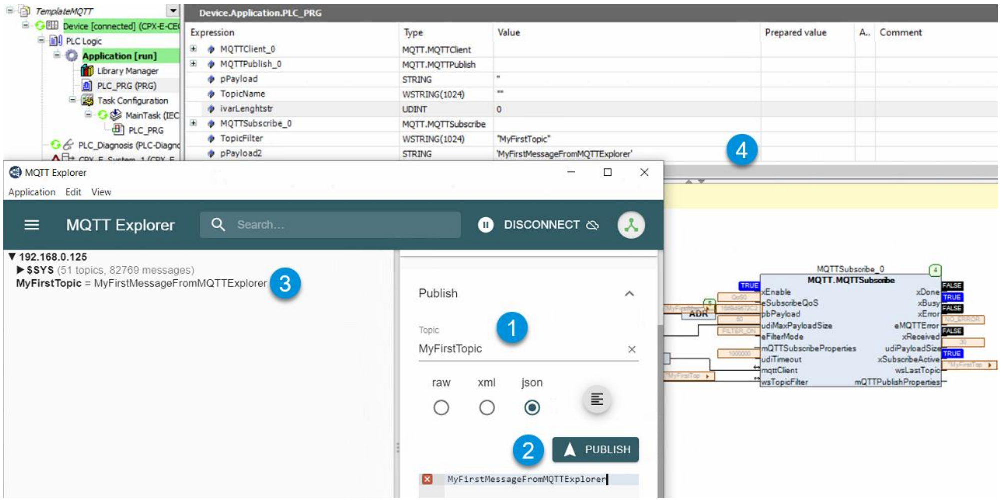
Figure 16: MQTT Subscribe FB configuration.

No.	Action
20	
21	
22	MQTT Explorer is used to publish a message payload. Message Payload = ***MyFirstMessageFromMQTTExplorer***

Expression	Type	Value	Prepared value	A..	Comment
MQTTClient_0	MQTT.MQTTClient				
MQTTPublish_0	MQTT.MQTTPublish				
pPayload	STRING	"			
TopicName	WSTRING(1024)	""			
ivarLenghtstr	UDINT	0			
MQTTSubscribe_0	MQTT.MQTTSubscribe				
TopicFilter	WSTRING(1024)	"MyFirstTopic"			
pPayload2	STRING	"			

Would you like to activate 'Auto Data Flow Mode'? Configure Help

23



TemplateMQTT · Device [connected] (CPX-E-CE · PLC Logic · Application [run] · Library Manager · PLC_PRG (PRG) · Task Configuration · MainTask (IEC · PLC_PRG · PLC_Diagnosis (PLC-Diagno

Device.Application.PLC_PRG

Expression	Type	Value	Prepared value	A..	Comment
MQTTClient_0	MQTT.MQTTClient				
MQTTPublish_0	MQTT.MQTTPublish				
pPayload	STRING	'			
TopicName	WSTRING(1024)	""			
ivarLenghtstr	UDINT	0			
MQTTSubscribe_0	MQTT.MQTTSubscribe				
TopicFilter	WSTRING(1024)	"MyFirstTopic"			
pPayload2	STRING	'MyFirstMessageFromMQTTExplorer'			

MQTT Explorer

192.168.0.125
$SYS (51 topics, 82769 messages)
MyFirstTopic = MyFirstMessageFromMQTTExplorer

Publish

Topic: MyFirstTopic

○ raw ○ xml ● json

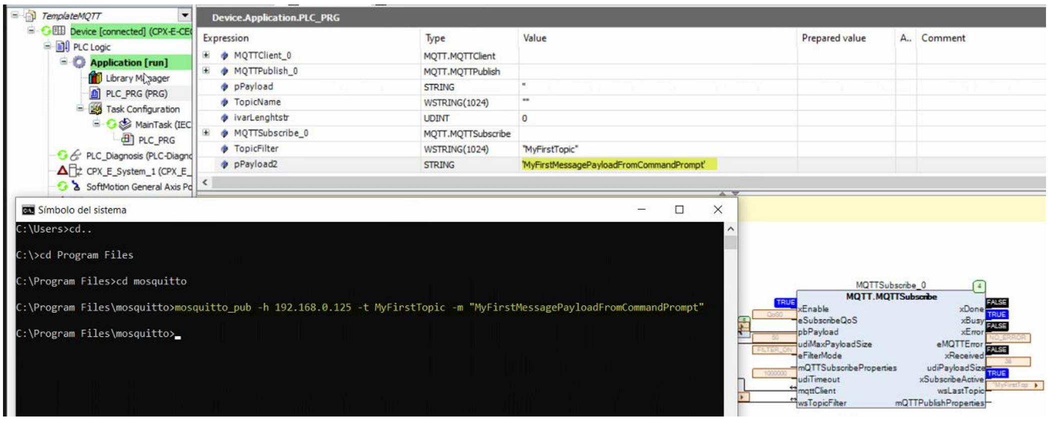
MyFirstMessageFromMQTTExplorer

24

It is also possible to publish a message by entering the following command in the comand prompt console.

mosquitto_pub -h 192.168.0.125 -t MyFirstTopic -m “MyFirstMessagePayloadFromCommandPrompt”

25



Device.Application.PLC_PRG

Expression	Type	Value	Prepared value	A..	Comment
MQTTClient_0	MQTT.MQTTClient				
MQTTPublish_0	MQTT.MQTTPublish				
pPayload	STRING	'			
TopicName	WSTRING(1024)	""			
ivarLenghtstr	UDINT	0			
MQTTSubscribe_0	MQTT.MQTTSubscribe				
TopicFilter	WSTRING(1024)	"MyFirstTopic"			
pPayload2	STRING	'MyFirstMessagePayloadFromCommandPrompt'			

Símbolo del sistema

```
C:\Users>cd..

C:\>cd Program Files

C:\Program Files>cd mosquitto

C:\Program Files\mosquitto>mosquitto_pub -h 192.168.0.125 -t MyFirstTopic -m "MyFirstMessagePayloadFromCommandPrompt"

C:\Program Files\mosquitto>_
```

3.5 Extra knowledge about MQTT features and JSON format.

In this section, some important concepts about the library IIOT and MQTT protocol itself are presented.

3.5.1 MQTT properties Topic Alias

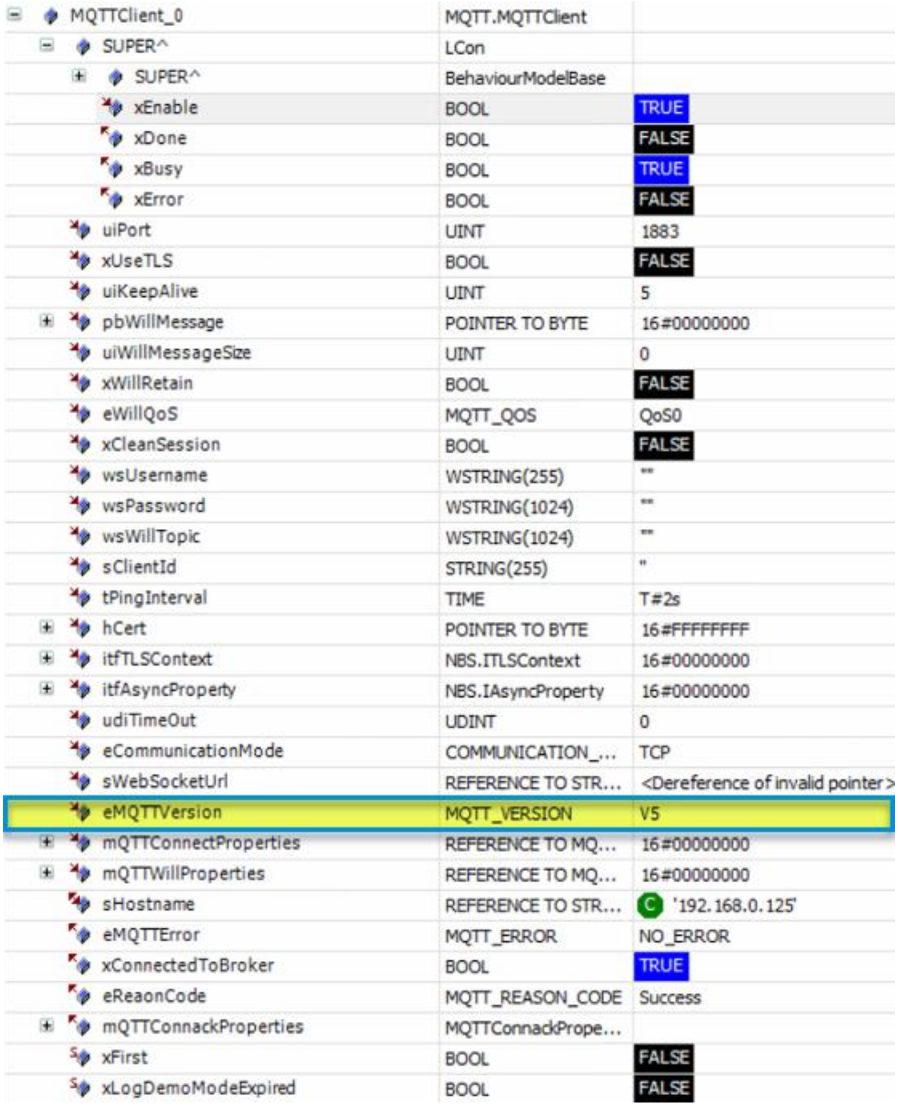
In the previous sections, the **"TopicName"** and **"Payload"** have been used, which are two essential variables in the MQTT protocol when publishing data or subscribing. In this section, a next concept called **"Topic Alias"** is introduced.

Topic Aliases are used to substitute a defined number of topic string with integers. Topic Aliases are a nifty little feature where a client and a broker can negotiate with one another and this feature is available in MQTT specification version 5. The client and the broker do support the feature.

To understand the usefulness of the Topic Alias, an example will be developed.

No.	**Action**
1	If you think about, for example, these complicated and rather long topic string: **TopicName =** *Europe/Germany/Esslingen/Festo/CMMT/AS/ActualTorque* **Payload =** '1.14' (Nm) **TopicName =** *Europe/Germany/Esslingen/Festo/CMMT/AS/ActualPosition* **Payload =** '120.20' (mm)
2	The client (Festo PLC) decides whether to use a Topic Alias and chooses the integer value. It sets a Topic Alias mapping. *Europe/Germany/Esslingen/Festo/CMMT/AS/ActualTorque* = **1 <-- Topic Alias** *Europe/Germany/Esslingen/Festo/CMMT/AS/ActualPosition = **2 <--*** **Topic Alias**
3	Connect to the Broker using MQTT_Client as explained in chapter 2.4.2.1
4	Input eMQTTVersion select **V5**

5



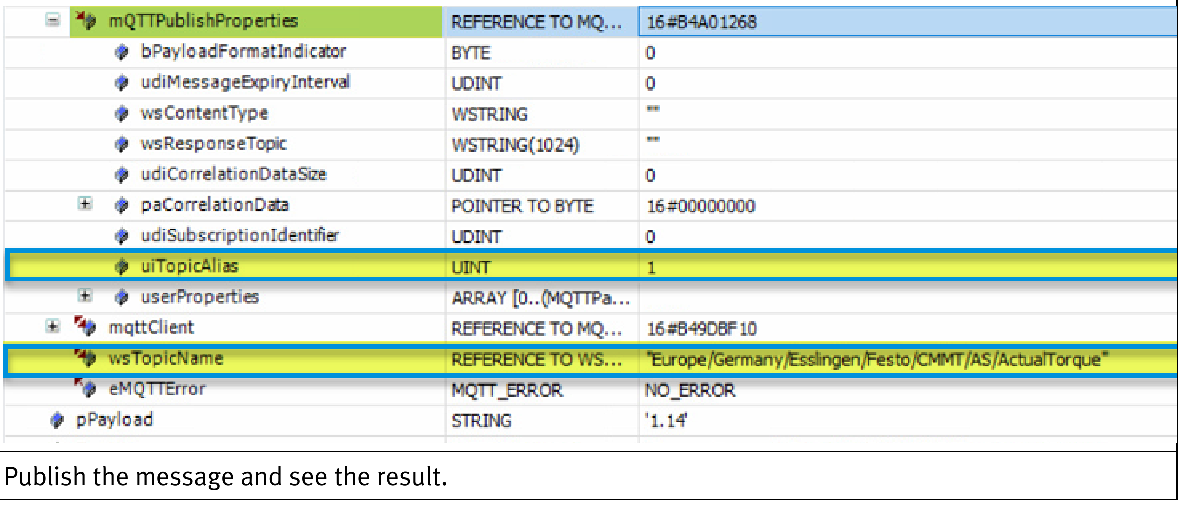
MQTTClient_0	MQTT.MQTTClient	
SUPER^	LCon	
SUPER^	BehaviourModelBase	
xEnable	BOOL	TRUE
xDone	BOOL	FALSE
xBusy	BOOL	TRUE
xError	BOOL	FALSE
uiPort	UINT	1883
xUseTLS	BOOL	FALSE
uiKeepAlive	UINT	5
pbWillMessage	POINTER TO BYTE	16#00000000
uiWillMessageSize	UINT	0
xWillRetain	BOOL	FALSE
eWillQoS	MQTT_QOS	QoS0
xCleanSession	BOOL	FALSE
wsUsername	WSTRING(255)	""
wsPassword	WSTRING(1024)	""
wsWillTopic	WSTRING(1024)	""
sClientId	STRING(255)	''
tPingInterval	TIME	T#2s
hCert	POINTER TO BYTE	16#FFFFFFFF
itfTLSContext	NBS.ITLSContext	16#00000000
itfAsyncProperty	NBS.IAsyncProperty	16#00000000
udiTimeOut	UDINT	0
eCommunicationMode	COMMUNICATION_...	TCP
sWebSocketUrl	REFERENCE TO STR...	<Dereference of invalid pointer>
eMQTTVersion	MQTT_VERSION	V5
mQTTConnectProperties	REFERENCE TO MQ...	16#00000000
mQTTWillProperties	REFERENCE TO MQ...	16#00000000
sHostname	REFERENCE TO STR...	C '192.168.0.125'
eMQTTError	MQTT_ERROR	NO_ERROR
xConnectedToBroker	BOOL	TRUE
eReaonCode	MQTT_REASON_CODE	Success
mQTTConnackProperties	MQTTConnackPrope...	
xFirst	BOOL	FALSE
xLogDemoModeExpired	BOOL	FALSE

6

Go to MQTT_Publish function block and do the relation between the **TopicName** and the **TopicAlias**.

The link has to be made the first time, once the message is published, the message payload can be sent with the desired **Topic Alias** integer.

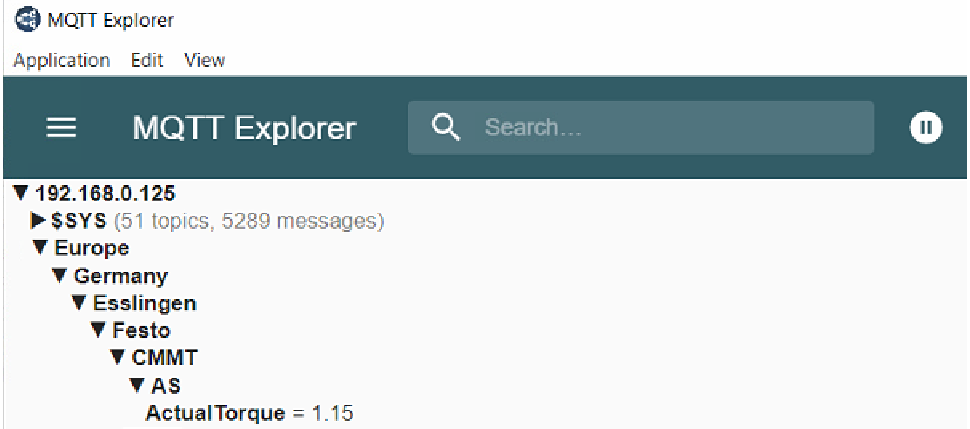
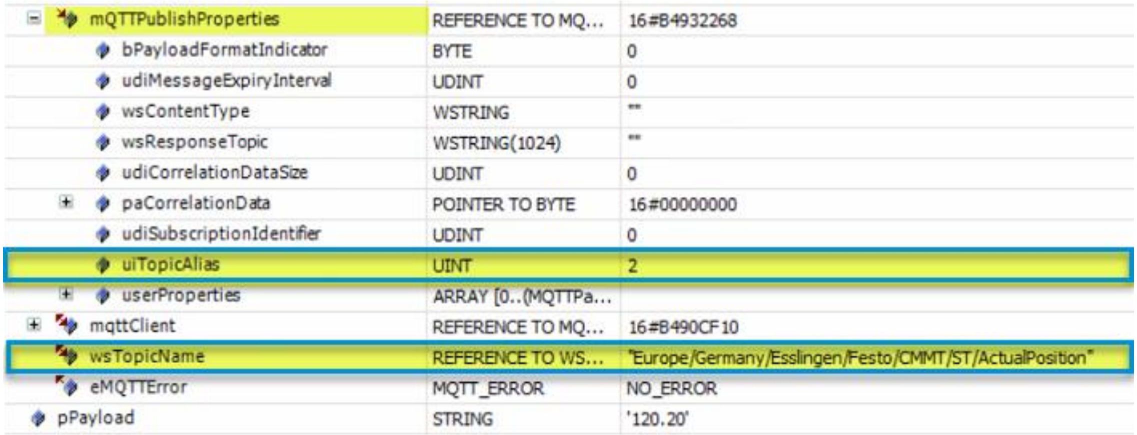
7



mQTTPublishProperties	REFERENCE TO MQ...	16#B4A01268
bPayloadFormatIndicator	BYTE	0
udiMessageExpiryInterval	UDINT	0
wsContentType	WSTRING	""
wsResponseTopic	WSTRING(1024)	""
udiCorrelationDataSize	UDINT	0
paCorrelationData	POINTER TO BYTE	16#00000000
udiSubscriptionIdentifier	UDINT	0
uiTopicAlias	UINT	1
userProperties	ARRAY [0..(MQTTPa...	
mqttClient	REFERENCE TO MQ...	16#B49DBF10
wsTopicName	REFERENCE TO WS...	"Europe/Germany/Esslingen/Festo/CMMT/AS/ActualTorque"
eMQTTError	MQTT_ERROR	NO_ERROR
pPayload	STRING	'1.14'

8

Publish the message and see the result.

<table>
<tr><td>9</td><td>
MQTT Explorer
Application Edit View
MQTT Explorer
Search…
▼ 192.168.0.125
▶ $SYS (51 topics, 5289 messages)
▼ Europe
▼ Germany
▼ Esslingen
▼ Festo
▼ CMMT
▼ AS
ActualTorque = 1.15</td></tr>
<tr><td>10</td><td>
<table>
<tr><td>mQTTPublishProperties</td><td>REFERENCE TO MQ…</td><td>16#B4932268</td></tr>
<tr><td>bPayloadFormatIndicator</td><td>BYTE</td><td>0</td></tr>
<tr><td>udiMessageExpiryInterval</td><td>UDINT</td><td>0</td></tr>
<tr><td>wsContentType</td><td>WSTRING</td><td>""</td></tr>
<tr><td>wsResponseTopic</td><td>WSTRING(1024)</td><td>""</td></tr>
<tr><td>udiCorrelationDataSize</td><td>UDINT</td><td>0</td></tr>
<tr><td>paCorrelationData</td><td>POINTER TO BYTE</td><td>16#00000000</td></tr>
<tr><td>udiSubscriptionIdentifier</td><td>UDINT</td><td>0</td></tr>
<tr><td>uiTopicAlias</td><td>UINT</td><td>2</td></tr>
<tr><td>userProperties</td><td>ARRAY [0..(MQTTPa…</td><td></td></tr>
<tr><td>mqttClient</td><td>REFERENCE TO MQ…</td><td>16#B490CF10</td></tr>
<tr><td>wsTopicName</td><td>REFERENCE TO WS…</td><td>"Europe/Germany/Esslingen/Festo/CMMT/ST/ActualPosition"</td></tr>
<tr><td>eMQTTError</td><td>MQTT_ERROR</td><td>NO_ERROR</td></tr>
<tr><td>pPayload</td><td>STRING</td><td>'120.20'</td></tr>
</table></td></tr>
<tr><td>9</td><td></td></tr>
<tr><td></td><td></td></tr>
</table>

3.5.2 JSON format

JSON stands for Javascript Object Notation. JSON is a lightweight format for storing and transporting data, it is often used when is sent from a server to a web page. JSON is self-describing and easy to understand and usually used as message transfer format in cloud services (e.g. Azure, AWS, Google).

It is important to emphasize that JSON is not a protocol but an encoding format. JSON enables structured data to be serialized into a text format, which is then sent over the wire to the receiving end. Json is typically used together with IoT protocols, and for that reason, an example will be explained using the IIOT libraries of Codesys.

Within the IIoT package of Codesys, The JSON Utilities library will be found. In the following example, two function blocks will be used: JSONFileReader and FindFirstValueByKey.

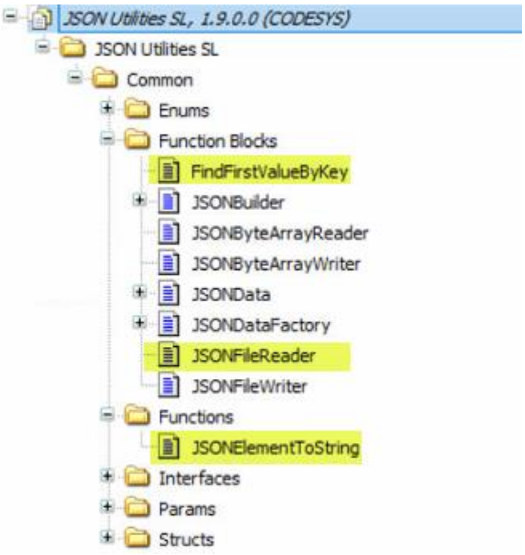


Figure 17: JSON Utilities from IIoT Library package.

3.5.2.1 JSONFileReader Function Block

The purpose of this function block is to enable reading JSON file.

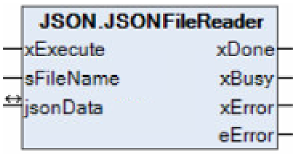


Figure 18: JSON File Reader Function Block.

3.5.2.1.1 Inputs

Tag Name	Data Type	Function Description
xExecute	BOOL	**Rising Edge –** Action starts. **Falling edge –** Reset outputs.
sFilename	STRING(255)	Path to a JSON file.
xIgnorevalueStringLenght	BOOL	If TRUE, the error is ignored and the value is shortened and xValueTrunked will be TRUE.
xAsyncMode	BOOL	If TRUE, reader runs in asynchronous mode.

3.5.2.1.2 Outputs

Tag Name	Data Type	Function Description
xDone	BOOL	**TRUE –** Action successfully completed.
xBusy	BOOL	**TRUE –** Function block active.
xError	BOOL	**TRUE –** Error occurred, function block aborts action. **FALSE –** No error.
eError	ERROR	**Rising Edge –** Action starts. **Falling edge –** Reset outputs.
xValueTruncated	BOOL	Minimum one string value was › GParams.g_diMaxStringSize

3.5.2.1.3 Inputs/Outputs

Tag Name	Data Type	Function Description
jsonData	JSONData	JSON data object.

ERROR

Name	Initial	Comment
NO_ERROR	0	No error.
INDEX_OUT_OF_BOUNDS	1	Array index was out of bounds.
NOT_FOUND	2	The requested element cannot be found.
NULL_POINTER	3	Null pointer.
MAX_STRING_SIZE_EXCEEDED	4	Max size of Strings exceeded.
READ_ERROR	5	Read error.
INVALID_HANDLE	6	Invalid file handle.
FILE_OPEN_ERROR	7	File open error.
INVALID_ENCODING	8	Invalid encoding.
WRITE_ERROR	9	Write error.
UNKNOWN_JSON_TYPE	10	Unknown JSON type
MAX_FILE_SIZE_EXCEEDED	11	Max size of tile exceeded.
INVALID_DECIMAL_PLACE	12	Decimal place is ‹ 0.
MAX_OBJECT_DEPTH_EXCEEDED	13	Maximum object depth g_diMaxDepth exceeded.
INVALID_KEY_PARENT	14	Parent of a key must be an object.
INVALID_VALUE_PARENT	15	Parent of a value must be an array or a key.
INVALID_LICENSE	16	No valid license found or demo mode has expired.
INVALID_STRUCTURE	17	Invalid JSON structure.
NOTSUPPORTED_DATATYPE	18	Datatype is not supported.

3.5.2.1 JSON.FindFirstValuebyKey

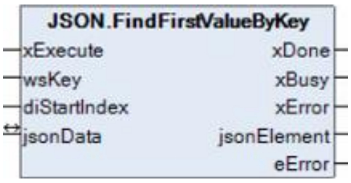


Figure 19: FindFirstValueByKey.

3.5.2.1.1 Inputs

Tag Name	Data Type	Function Description
xExecute	BOOL	**Rising Edge –** Action starts. **Falling edge –** Reset outputs. If a falling edge occurs before the function block has completed its action, the outputs operate in the usual manner and are only reset if either the action is completed or in the event of an error. In this case, the corresponding output values (xDone, xError) are present at the outputs for exactly one cycle.
wsKey	WSTRING	The key to search.
diStartIndex	DINT	Start index.

3.5.2.1.2 Outputs

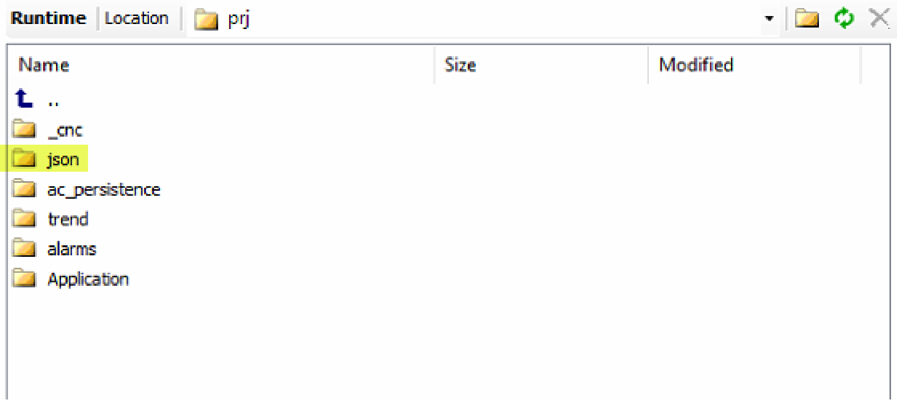
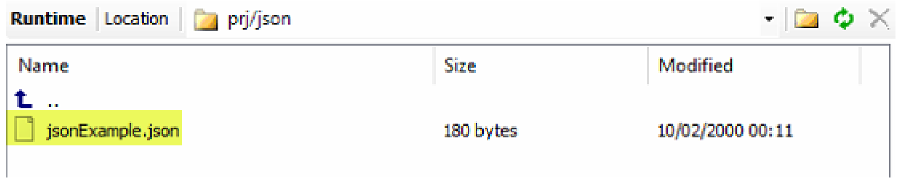
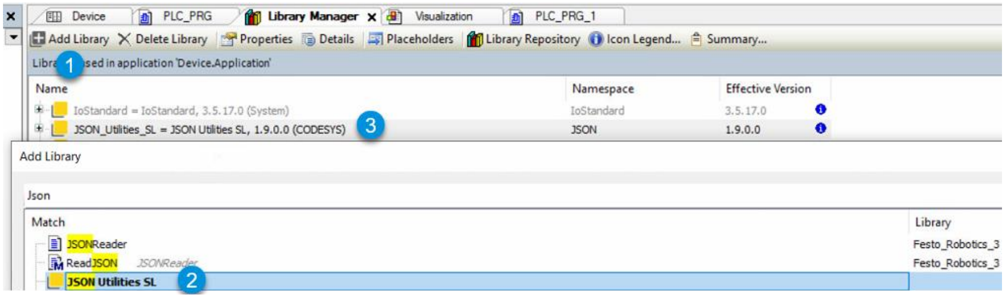
Tag Name	Data Type	Function Description
xDone	BOOL	**TRUE** – Action successfully completed.
xBusy	BOOL	**TRUE** – Function block active
xError	BOOL	**TRUE** – Error occurred, function block aborts action. **FALSE** – No error.
eError	ERROR	

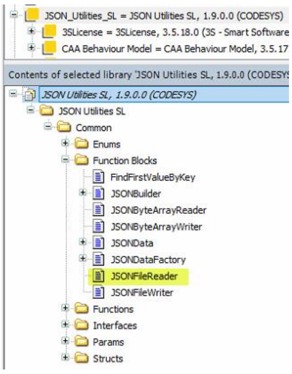
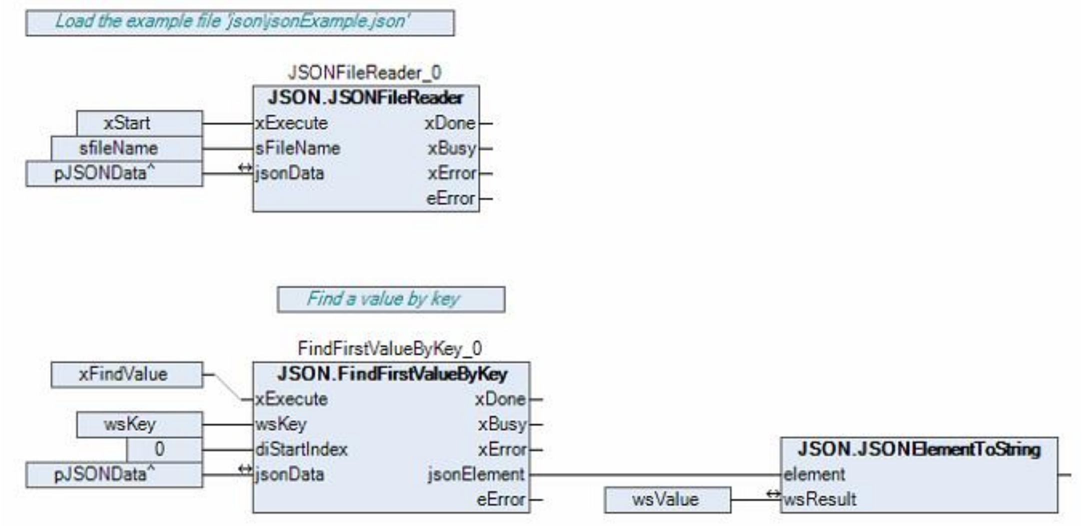
3.5.2.1.3 Inputs/Outputs

Tag Name	Data Type	Function Description
jsonData	JSONData	JSON Data Object.

3.5.3 JSON Example

In the following example, loads a .json file and find the value of the first key in the file.

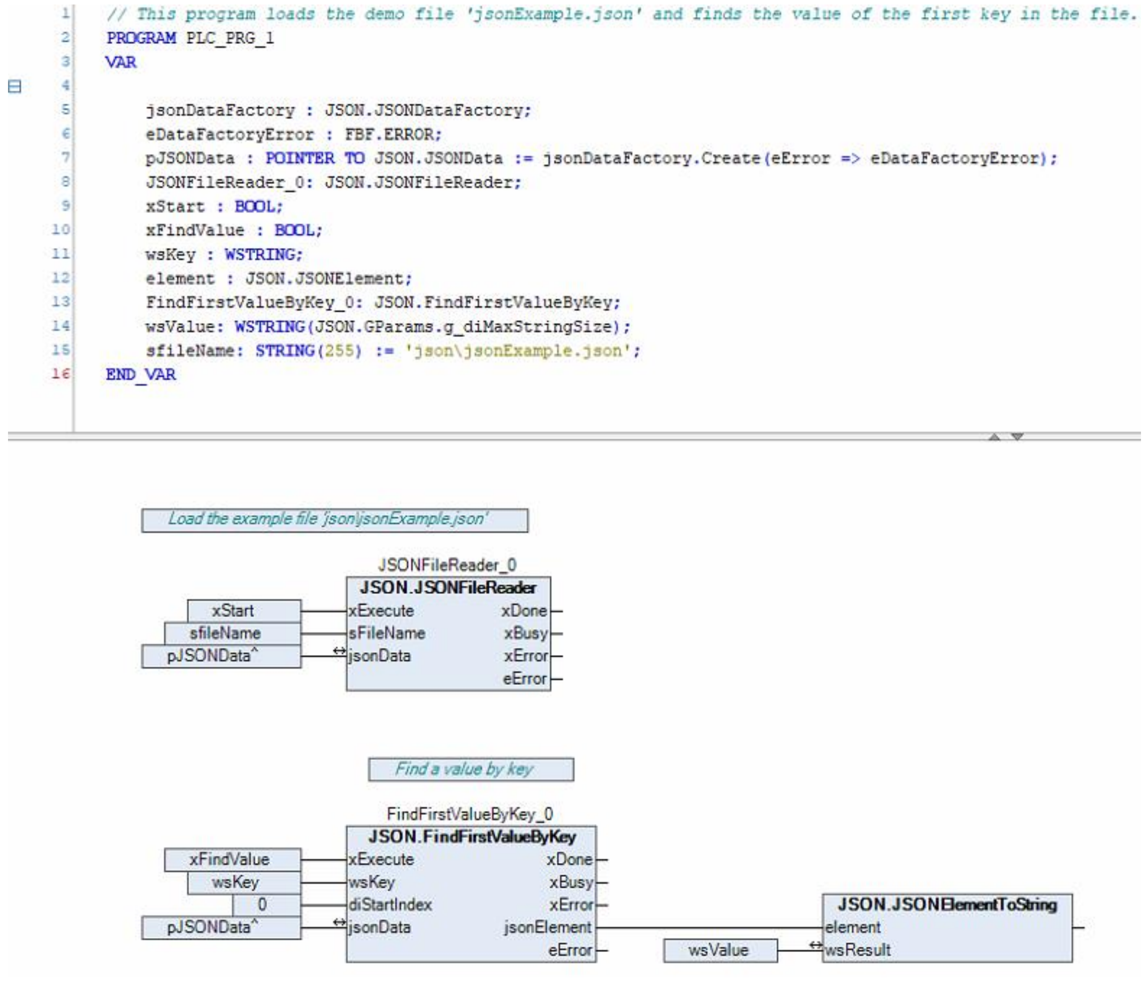
No.	Action
1	Here an example of Json format:
2	{ « location »: »Esslingen », « datetime »: »2024-04-14 12:01:59 », « device »: »CPX-E », “machine”: “OP625pump16”, “temperature”: “34.6”, “pressure”:”6” }
3	The json file needs to be load in the memory of the PLC. Create a folder in the PLC */prj/json/jsonExample.json*
	 Runtime · Location · prj Name · Size · Modified .. _cnc json ac_persistence trend alarms Application
4	 Runtime · Location · prj/json Name · Size · Modified .. jsonExample.json · 180 bytes · 10/02/2000 00:11
3	Add the library: JSON_Utilities_SL
4	 Libra[1]sed in application 'Device.Application' Name · Namespace · Effective Version IoStandard = IoStandard, 3.5.17.0 (System) · IoStandard · 3.5.17.0 JSON_Utilities_SL = JSON Utilities SL, 1.9.0.0 (CODESYS) [3] · JSON · 1.9.0.0 Add Library Json Match · Library JSONReader · Festo_Robotics_3 ReadJSON · JSONReader · Festo_Robotics_3 JSON Utilities SL [2]

5	 JSON_Utilities_SL = JSON Utilities SL, 1.9.0.0 (CODESYS) 3SLicense = 3SLicense, 3.5.18.0 (3S - Smart Software CAA Behaviour Model = CAA Behaviour Model, 3.5.17 Contents of selected library 'JSON Utilities SL, 1.9.0.0 (CODESYS JSON Utilities SL, 1.9.0.0 (CODESYS) JSON Utilities SL Common Enums Function Blocks FindFirstValueByKey JSONBuilder JSONByteArrayReader JSONByteArrayWriter JSONData JSONDataFactory JSONFileReader JSONFileWriter Functions Interfaces Params Structs
6	 Load the example file 'json\jsonExample.json' JSONFileReader_0 JSON.JSONFileReader xStart sfileName pJSONData^ xExecute sFileName jsonData xDone xBusy xError eError Find a value by key FindFirstValueByKey_0 JSON.FindFirstValueByKey xFindValue wsKey 0 pJSONData^ xExecute wsKey diStartIndex jsonData xDone xBusy xError jsonElement eError JSON.JSONElementToString element wsResult wsValue

7	**JSONFileReader Function Block**
	Create input variables as:
	JSONFileReader_0:JSONFileReader;
	jsonDataFactory: JSON.JSONDataFactory
	xStart: BOOL;
	sfileName: STRING(255) := 'json\jsonExample.json';
	pJSONData^ : POINTER TO JSON.JSONData := jsonDataFactory.Create(eError => eDataFactoryError);

8	**JSON.FindFirstValueByKey Function Block**
	Create input variables as:
	xFindValue:BOOL;
	wsKey:WSTRING;
	pJSonData^: POINTER TO JSON.JSONData := jsonDataFactory.Create(eError => eDataFactoryError);

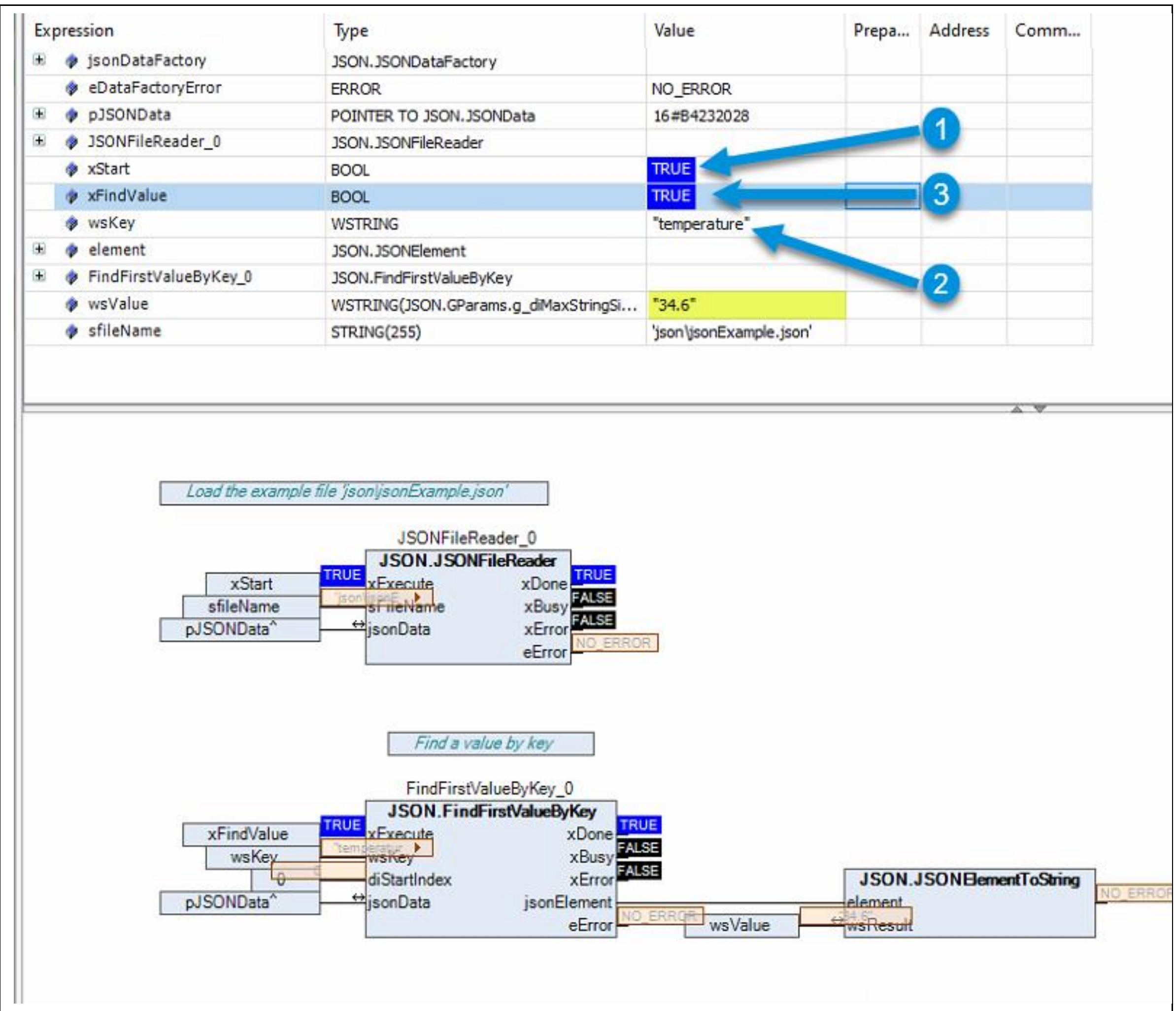
9	**JSON.JSONElementToString** Create input variable as: **wsValue**: WSTRING(JSON.GParams.g_diMaxStringSize);
10	The program has the following appearance:



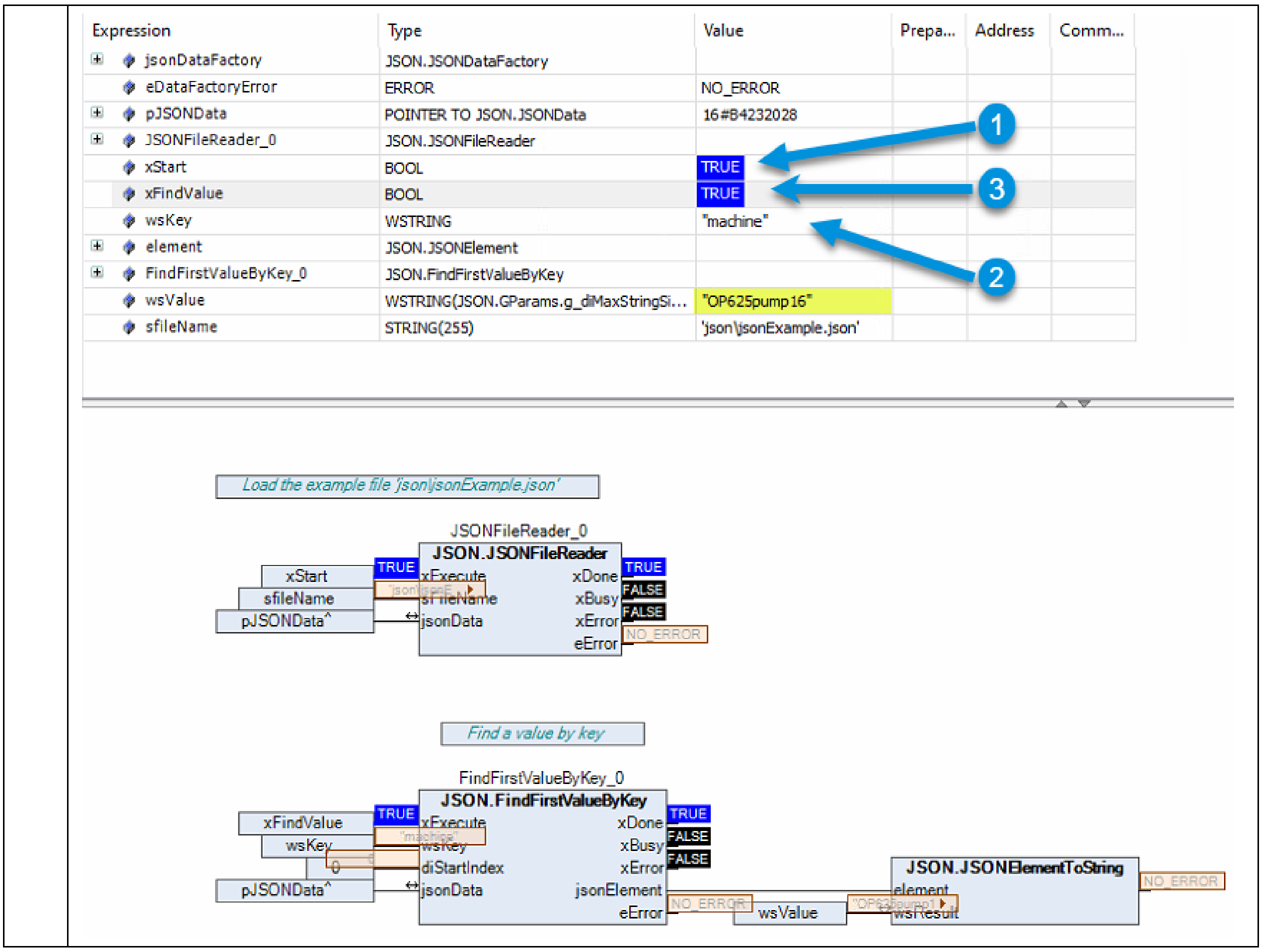
```
// This program loads the demo file 'jsonExample.json' and finds the value of the first key in the file.
PROGRAM PLC_PRG_1
VAR

    jsonDataFactory : JSON.JSONDataFactory;
    eDataFactoryError : FBF.ERROR;
    pJSONData : POINTER TO JSON.JSONData := jsonDataFactory.Create(eError => eDataFactoryError);
    JSONFileReader_0: JSON.JSONFileReader;
    xStart : BOOL;
    xFindValue : BOOL;
    wsKey : WSTRING;
    element : JSON.JSONElement;
    FindFirstValueByKey_0: JSON.FindFirstValueByKey;
    wsValue: WSTRING(JSON.GParams.g_diMaxStringSize);
    sfileName: STRING(255) := 'json\jsonExample.json';
END_VAR
```

11	Result: wsValue = **Temperature** and returned **OP625pump16**

	
12	Result: wsValue = **machine** and returned **36.5**

Expression	Type	Value	Prepa...	Address	Comm...
⊞ jsonDataFactory	JSON.JSONDataFactory				
eDataFactoryError	ERROR	NO_ERROR			
⊞ pJSONData	POINTER TO JSON.JSONData	16#B4232028			
⊞ JSONFileReader_0	JSON.JSONFileReader				
xStart	BOOL	TRUE			
xFindValue	BOOL	TRUE			
wsKey	WSTRING	"temperature"			
⊞ element	JSON.JSONElement				
⊞ FindFirstValueByKey_0	JSON.FindFirstValueByKey				
wsValue	WSTRING(JSON.GParams.g_diMaxStringSi...	"34.6"			
sfileName	STRING(255)	'json\jsonExample.json'			



Expression	Type	Value	Prepa...	Address	Comm...
jsonDataFactory	JSON.JSONDataFactory				
eDataFactoryError	ERROR	NO_ERROR			
pJSONData	POINTER TO JSON.JSONData	16#B4232028			
JSONFileReader_0	JSON.JSONFileReader				
xStart	BOOL	TRUE			
xFindValue	BOOL	TRUE			
wsKey	WSTRING	"machine"			
element	JSON.JSONElement				
FindFirstValueByKey_0	JSON.FindFirstValueByKey				
wsValue	WSTRING(JSON.GParams.g_diMaxStringSi...	"OP625pump16"			
sfileName	STRING(255)	'json\jsonExample.json'			

4 Appendix

MQTT: Message Queuing Telemetry Transport.

IIoT: Industrial Internet of Things.

Topic : Topics are an alphanumeric identifier that is assigned to MQTT messages in order to MQTT messages to classify them according to a context.

Topic Alias: It is a featured introduced in MQTT version 5.0. It allows clients to use a shorter, numeric identifier to represent a topic name, reducing the overhead of repeatedly sending the same big topic name in subsequent messages.

Payload: Refers to the actual data contained within an MQTT message. This is the content or the body of the message that is being transmitted from a publisher to subscriber through the MQTT broker.

QoS (**Quality of Service) :** It is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message.

- QoS 0 – at most once : The minimal QoS level is zero. There is no guarantee of delivery. The receiver does not acknowledge receipt of the message and the message is not stored and retransmitted by the sender.
- QoS 1 – At least once: Level 1 guarantees that a message is delivered at least one time to the receiver. The sender stores the message until it gets a Puback packet form the receiver that acknowledges receipt of the message. It is possible for a message to be sent or delivered multiple times.
- QoS 2 – Exactly once : Qo2 is the highest level of service in MQTT protocol. This quality level guarantees that each message is received only once by the intended recipients. QoS 2 is the safest and slowest quality of service level. The guarantee is provided by at least two request/response flows (a four-part handshake) between the sender and the receiver. The sender and receiver use the packet identifier of the original PUBLISH message to coordinate delivery of the message.

JSON : Java Script Object Notation.