Rockwell Automation Application Content

Common Libraries

Reference Manual

MQTT Communication

raC_Opr_MQTT_x v1.x

March, 2023

Important User Information

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

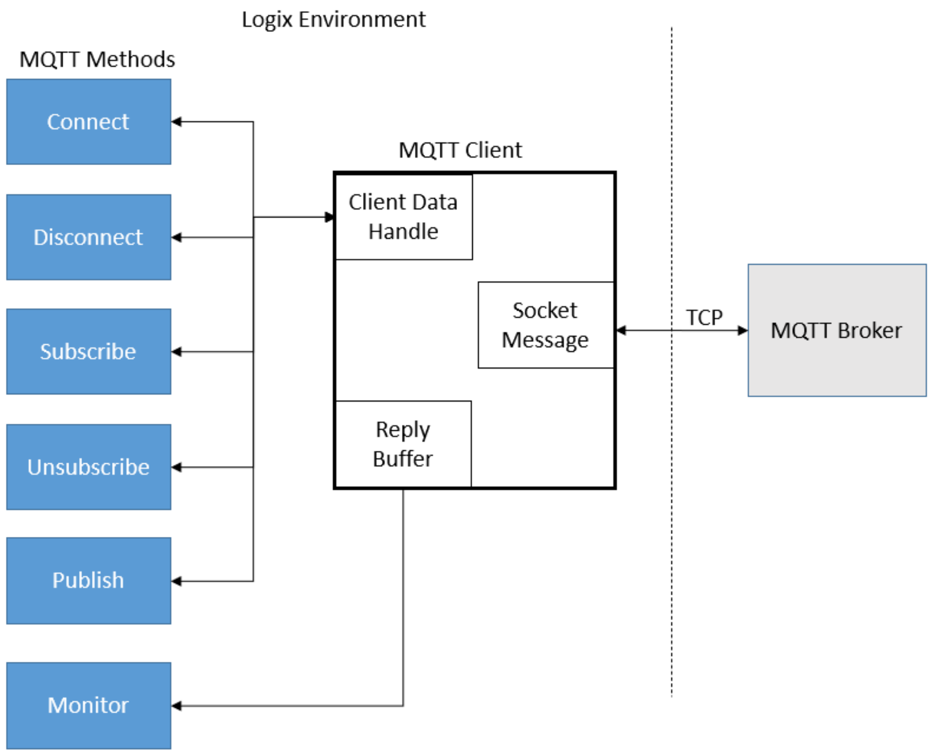
WARNING	Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
ATTENTION	Identifies information about practices or circumstances or death, property damage, or economic loss. Attentions avoid a hazard and recognize the consequence.
SHOCK HAZARD	Labels may be on or inside the equipment, that dangerous voltage may be present.
BURN HAZARD	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

Table of Contents

1 Overview

The MQTT Comms Instructions allows a Socket Capable Logix Controller to connect to MQTT broker or server. Topics can be subscribed, unsubscribed, and published. Updates to subscribed to topics can be monitored for updates. This code supports MQTT version 3.1.1

- MQTT Client instruction contains Message Instructions for Socket Communication with MQTT Broker.
- The Client can be commanded to Connect, Disconnect, Subscribe, Unsubscribe and Publish using respective method instructions.
- The Monitor instruction monitors for specific topic data in the Reply Buffer.
- Client Data Handle tag acts as common interface between MQTT Client instruction and Method instructions



Note: It is recommended to connect MQTT broker and client in a local network.

1.1 <u>Prerequisites</u>

- To use this code, the controller must have a Sockets capable Ethernet interface. CompactLogix 5370 and 5380 and ControlLogix 5580 controllers have this built in. Other ControlLogix Controllers can be made sockets capable with a 1756-EN2T or similar card. For more information, see Knowledge base Article 470690. The controller must have firmware 24 or higher.

- Studio 5000 - Logix Designer
  - V27.0 →

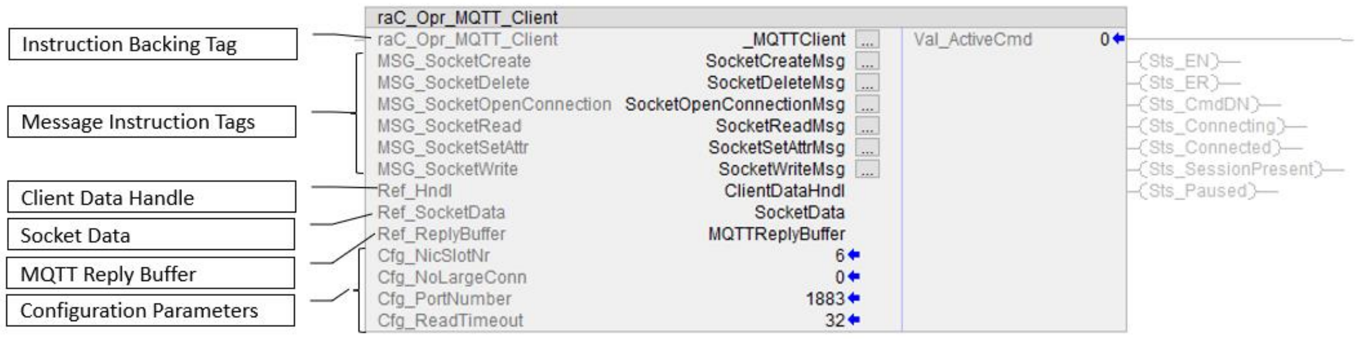
- Studio 5000 - Application Code Manager
  - V4.0 →

2 MQTT Client

2.1 Functional Description

raC_Opr_MQTT_Client instruction manages communication with MQTT Broker. The instructions support Connect, Disconnect, Subscribe, Unsubscribe and Publish commands. These commands can be executed using the respective method instructions.

2.2 Instruction



raC_Opr_MQTT_Client	
raC_Opr_MQTT_Client	_MQTTClient
MSG_SocketCreate	SocketCreateMsg
MSG_SocketDelete	SocketDeleteMsg
MSG_SocketOpenConnection	SocketOpenConnectionMsg
MSG_SocketRead	SocketReadMsg
MSG_SocketSetAttr	SocketSetAttrMsg
MSG_SocketWrite	SocketWriteMsg
Ref_Hndl	ClientDataHndl
Ref_SocketData	SocketData
Ref_ReplyBuffer	MQTTReplyBuffer
Cfg_NicSlotNr	6
Cfg_NoLargeConn	0
Cfg_PortNumber	1883
Cfg_ReadTimeout	32
Val_ActiveCmd	0

2.3 Footprint

Estimated memory required to store code depends on the MQTT Payload size.

Characteristic	Description	Payload 1024 bytes	Payload 4096 bytes	Payload 16384 bytes
Definition	Estimated memory required to store the object definition, including all dependents.	78 kB	82 kB	94 kB
Instance	Estimated memory required per object instantiated. This includes the object instance and all datatypes required to verify the project. In the case of user configurable arrays, an application relevant array length will be used for estimation.	29 kB	65 kB	209 kB
Execution L7x	Estimated execution time / scan footprint evaluated in 1756-L7x PAC	350 us		

2.4 InOut Data

InOut	Function / Description	Datatype
MSG_SocketCreate	Message Object for Create Socket	MESSAGE
MSG_SocketDelete	Message Object for Delete Socket	MESSAGE
MSG_SocketOpenConn	Message Object for Open Socket	MESSAGE
MSG_SocketRead	Message Object for Read Socket	MESSAGE
MSG_SocketSetAttr	Message Object for Set Attribute	MESSAGE
MSG_SocketWrite	Message Object for Write Socket	MESSAGE
Ref_Hndl	MQTT Client Handle Data	raC_UDT_Opr_MQTTHandle
Ref_SocketData	Socket Data	raC_UDT_Opr_MQTTRawData

InOut	Function / Description	Datatype
Ref_ReplyBuffer	MQTT Reply Buffer	raC_UDT_Opr_MQTTBuffer

2.5 Input Data

Input	Function / Description	Datatype
Cfg_NicSlotNr	Logix slot number of the sockets capable network interface. For 1769 slot number is 0. For 5069 and 1756-L8x the slot number is -1. When using a 1756-ENxT(R), EWEB this is the slot number of the ethernet card.	DINT
Cfg_NoLargeConn	Set when socket only supports 462 bytes write. When using 1769-LxxE controllers or a EWEB module.	BOOL
Cfg_PortNumber	TCP Port number. Example - 1883 or 8883	DINT
Cfg_ReadTimeOut	Time Out in msecs during Reading from MQTT Broker. Valid range is 0 to 1000.	DINT
Cfg_KeepAliveTime	Keep Alive Time in Secs for Broker and Client Connection. Valid range is 0 to 1000.	DINT
Cfg_PauseTime	Sequence Pause and Restart Time in Secs in case of Message Error. Valid range is 1 to 60.	DINT

2.6 Output Data

Output	Function / Description	Datatype
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_CmdDN	Command Done	BOOL
Sts_Connecting	Client Connecting to Broker	BOOL
Sts_Connected	Client Connected to Broker	BOOL
Sts_SessionPresent	Client was Connected before	BOOL
Sts_Paused	Client Sequence Paused. In the event of an error in messaging, the Client instruction sequence is paused until pause time. Once done, the client execution sequence will resume	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT
Val_ActiveCmd	Active Command. 3 = Publish, 8 = Subscribe, 10 = Unsubscribe	DINT

2.7 Error Code

Error Code is common across all MQTT Instructions. The same can be viewed in Sts_ERR parameter.

ERR value	Description
0	no error
1000	Client already connected
1001	No Response Received from Broker
1002	Illegal value in Cfg_IPAddress
1003	Illegal value in Client ID
1004	Illegal value in Cfg_WillTopic
1005	Illegal value in Cfg_WillMessage
1006	Illegal value in Cfg_WillQOS
1007	Illegal value in Cfg_UserName
1008	Illegal value in Cfg_Password

ERR value	Description
1009	Cfg_KeepAliveTime is greater than 65536 or less than 0
1010	Unsupported command in Cmd_CPT
1011	Command Timed out
1012	Payload too big to send
1014	Received more data than payload can contain
1015	Unable to create socket. Refer Socket Create Message Error Code.
1016	Unable to connect to server. Refer Socket Open Connection Message Error Code.
1017	Failed to write. Refer Socket Write Message Error Code.
1018	Failed to read. Refer Socket Read Message Error Code.
1019	Error in stream, disconnecting
1020	Unable to set socket buffer size. Refer Socket Set Attribute Message Error Code.
1021	Error from broker received
1022	Attempted Cmd_CPT when client is disconnected
1023	Illegal value in Set_Payload
1024	Illegal value in Set_Topic
1025	Illegal value in Set_QOS
1030	No CONNACK received
1031	Connection refused, bad protocol version
1032	Connection refused; ID refused
1033	Connection refused; server unavailable
1034	Connection refused, malformed username or password

2.8 Client Data Handle

raC_UDT_Opr_MQTTHandle

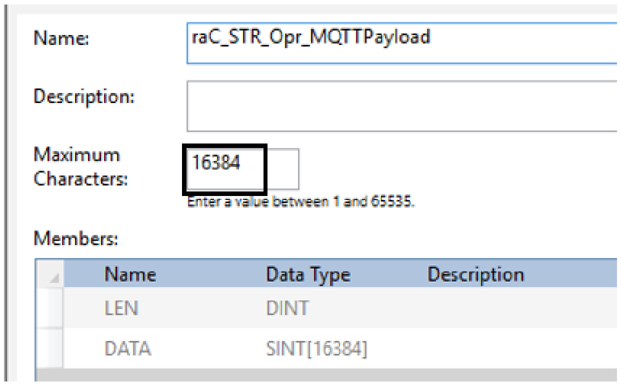
Member	Function / Description	Datatype
Cfg_IPAddress	IP Address of MQTT Server	DINT[4]
Cfg_HostName	Host Name of MQTT Server	STRING
Cfg_ClientId	Client Identifier	STRING
Cfg_WillTopic	Will Topic	STRING
Cfg_WillMessage	Will Message	STRING
Cfg_WillQos	Will Quality of Service	SINT
Cfg_WillRetain	Retain Will Message	BOOL
Cfg_CleanSession	1 = Discard previous session	BOOL
Cfg_UserName	MQTT Server Username	STRING
Cfg_Password	MQTT Server Password	STRING
Cmd_Connect	Connect Command	BOOL
Cmd_Disconnect	Disconnect Command	BOOL
Cmd_Reinit	Set to reset, hold high to block the MQTT code	BOOL
Cmd_CPT	MQTT Control Packet Type 3 = Publish, 8 = Subscribe, 10 = Unsubscribe	DINT
Set_Topic	Topic	STRING
Set_QOS	Requested Quality of service. Valid value is 0,1,2.	SINT
Set_Retain	Retain flag for publish	BOOL
Set_Payload	Payload	raC_STR_MQTTPayload

Member	Function / Description	Datatype
Sts_Connecting	Client Connecting to Broker	BOOL
Sts_Connected	Connection to Broker Open	BOOL
Sts_SessionPresent	This Client was connected before	BOOL
Sts_CmdDN	Last Command Successful	BOOL
Sts_ER	Last Command Failed. Refer Sts_ERR for Error Code.	BOOL
Sts_ERR	Error Code	DINT
Sts_Paused	Client Sequence Paused	BOOL
Val_ActiveCmd	Active Command. 3=Publish, 8 = Subscribe, 10 = Unsubscribe	DINT

2.9 Payload String

raC_STR_Opr_MQTTPayload

The size of Payload String can be customized based on maximum number of bytes that MQTT payload should be able to handle. Setting this to a smaller value will reduce the memory footprint of the application. It can be up to 65535 bytes. The code adapts automatically based on size of string. Incoming messages that are larger than the maximum payload will be received but cut off at the maximum payload length.



Name: raC_STR_Opr_MQTTPayload

Description:

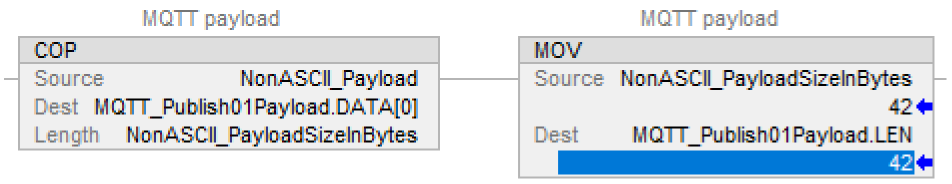
Maximum Characters: 16384

Enter a value between 1 and 65535.

Members:

Name	Data Type	Description
LEN	DINT	
DATA	SINT[16384]	

The Payload String allows the use of ASCII instructions on it, to create JSON or similar syntax. Non-ascii payloads can be published by copying the source to the DATA part of the payload and setting the correct source length in bytes to the LEN part.



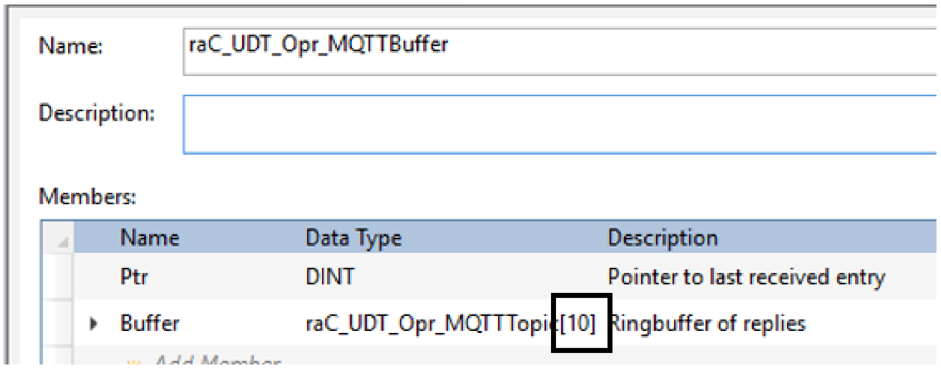
```
MQTT payload
COP
Source              NonASCII_Payload
Dest   MQTT_Publish01Payload.DATA[0]
Length   NonASCII_PayloadSizeInBytes

MQTT payload
MOV
Source  NonASCII_PayloadSizeInBytes
                                 42
Dest       MQTT_Publish01Payload.LEN
                                 42
```

2.10 Reply Buffer

raC_UDT_Opr_MQTTBuffer

The replies for subscriptions from MQTT Broker are stored in the reply buffer. The required topic payload can be extracted using Monitor Instruction. By default, the reply buffer array size is 10, which means last 10 replies will be stored in the buffer. If required, the buffer array size can be increased to handle large number of replies.



Name: raC_UDT_Opr_MQTTBuffer

Description:

Members:

Name	Data Type	Description
Ptr	DINT	Pointer to last received entry
Buffer	raC_UDT_Opr_MQTTTopic[10]	Ringbuffer of replies

2.11 Configuration based on Hardware

Below explanation helps to understand what configuration changes are needed in MQTT Client instance depending on the Logix Controller type.

When instantiating the MQTT Client instruction using ACM or Studio5000 Logix Designer Import Plugin, all the required configurations are pre-done. It is highly recommended to instantiate the instructions from Library to avoid the below steps.

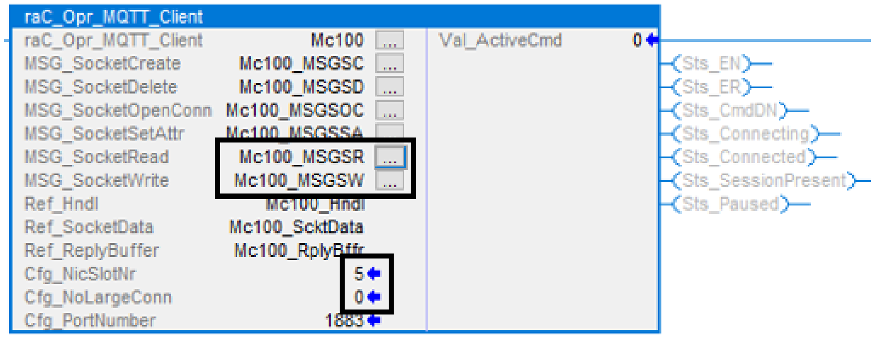
Hardware			MQTT Client Cfg		Socket Read and Write MSG Setting			
Controller	**Backplane CommCard**	**CommCard SlotNumber**	**NICSlotNr**	**NoLargeConn**	**Path**	**Connected**	**Cached**	**Large Connection**
1756-L8x	Not Used	Not Applicable	-1	0	THIS	0	0	0
1756-L8x	1756-Enxx	6	6	0	1,6	1	1	1
1756-L8x	1756-EWEB	6	6	1	1,6	0	0	0
1756-L7x	1756-ENxx	6	6	0	1,6	1	1	1
1756-L7x	1756-EWEB	6	6	1	1,6	0	0	0
5069	Not Applicable	NA	-1	0	THIS	0	0	0
1769	Not Applicable	NA	0	1	1,0	0	0	0

Number 6 in CommCard Slot Number is an example. It shows how NICSlotNumber parameter and Message Path is set as per comm card slot number.

For Example, consider controller used is 1756-L85E and 1756-EN2TR. The EN2TR is in 5th Slot in chassis. Referring above table the following are the values to be set.

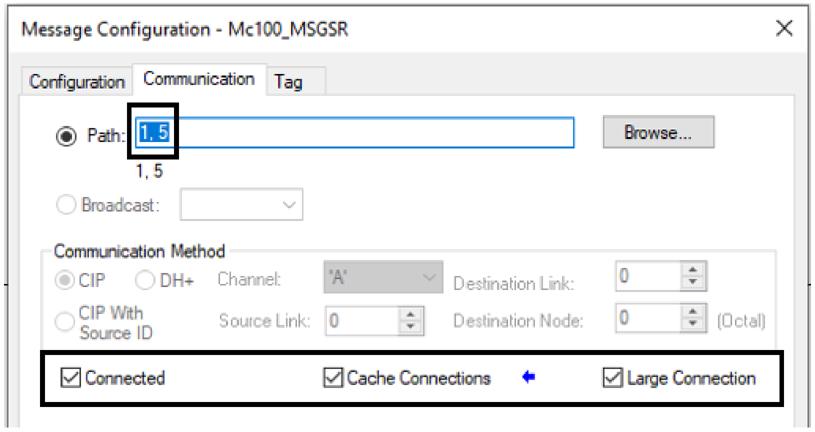
Hardware			MQTT Client Cfg		Socket Read and Write MSG Setting			
Controller	Backplane CommCard	CommCard SlotNumber	NICSlotNr	NoLargeConn	Path	Connected	Cached	Large Connection
1756-L8x	1756-Enxx	5	5	0	1,5	1	1	1

MQTT Client instruction configuration parameters are set as follows
Cfg_NICSlotNr = 5
Cfg_NoLargeConn = 0


raC_Opr_MQTT_Client

raC_Opr_MQTT_Client	Mc100	Val_ActiveCmd 0
MSG_SocketCreate	Mc100_MSGSC	
MSG_SocketDelete	Mc100_MSGSD	
MSG_SocketOpenConn	Mc100_MSGSOC	
MSG_SocketSetAttr	Mc100_MSGSSA	
MSG_SocketRead	Mc100_MSGSR	
MSG_SocketWrite	Mc100_MSGSW	
Ref_Hndl	Mc100_Hndl	
Ref_SocketData	Mc100_ScktData	
Ref_ReplyBuffer	Mc100_RplyBffr	
Cfg_NicSlotNr	5	
Cfg_NoLargeConn	0	
Cfg_PortNumber	1883	

Open the MSG_SocketRead properties


Message Configuration - Mc100_MSGSR

Configuration | Communication | Tag

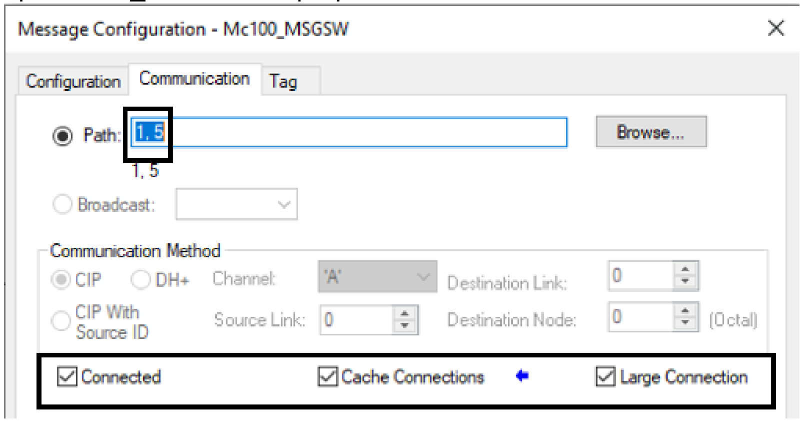
Path: 1, 5
1, 5

Broadcast:

Communication Method
CIP DH+ Channel: 'A' Destination Link: 0
CIP With Source ID Source Link: 0 Destination Node: 0 (Octal)

☑ Connected ☑ Cache Connections ☑ Large Connection

Open MSG_SocketWrite properties


Message Configuration - Mc100_MSGSW

Configuration | Communication | Tag

Path: 1, 5
1, 5

Broadcast:

Communication Method
CIP DH+ Channel: 'A' Destination Link: 0
CIP With Source ID Source Link: 0 Destination Node: 0 (Octal)

☑ Connected ☑ Cache Connections ☑ Large Connection

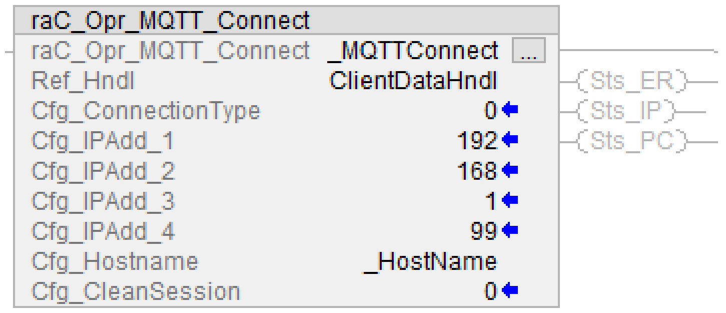
3 MQTT Connect

3.1 Functional Description

raC_Opr_MQTT_Connect instruction is used to connect Client to MQTT Broker. The instruction has option to use IP address or Host Name of the MQTT Broker. The instruction hosts basic client configuration parameters like IP Address, Host Name and Clean session. Further advanced parameters like ClientId, WillTopic, WillMessage, WillQoS, WillRetain, Username, and Password can be set in the Client Data Handle tag.

The instruction execution is edge driven. Once true, the rung condition need not be maintained.

3.2 Instruction



3.3 InOut Data

InOut	Function / Description	Datatype
Ref_Handle	MQTT Client Data Handle	raC_UDT_Opr_MQTTHandle

3.4 Input Data

Input	Function / Description	Datatype
Cfg_ConnectionType	Select Connection Type 0 = IP Address, 1 = HostName	BOOL
Cfg_IPAdd_1	IP Address of MQTT Server. Valid range is 0 to 256.	DINT
Cfg_IPAdd_2	IP Address of MQTT Server. Valid range is 0 to 256.	DINT
Cfg_IPAdd_3	IP Address of MQTT Server. Valid range is 0 to 256.	DINT
Cfg_IPAdd_4	IP Address of MQTT Server. Valid range is 0 to 256.	DINT
Cfg_Hostname	Host Name of MQTT Server	STRING
Cfg_CleanSession	1 = Discard previous Session	BOOL

3.5 Output Data

Output	Function / Description	Datatype
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_IP	Instruction is In Process	BOOL
Sts_PC	Instruction Process Complete	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT

3.6 Error Codes

The Error code description is same as for MQTT_Client instruction. Refer chapter 2.7.

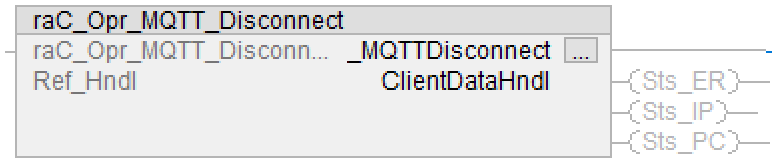
4 MQTT Disconnect

4.1 Functional Description

raC_Opr_MQTT_Disconnect instruction is used to disconnect the current session of Client and MQTT Broker.

The instruction execution is edge driven. Once true, the rung condition need not be maintained.

4.2 Instruction



4.3 InOut Data

InOut	Function / Description	Datatype
Ref_Handle	MQTT Client Data Handle	raC_UDT_Opr_MQTTHandle

4.4 Output Data

Output	Function / Description	Datatype
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_IP	Instruction is In Process	BOOL
Sts_PC	Instruction Process Complete	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT

4.5 Error Codes

The Error code description is same as for MQTT_Client instruction. Refer chapter 2.7.

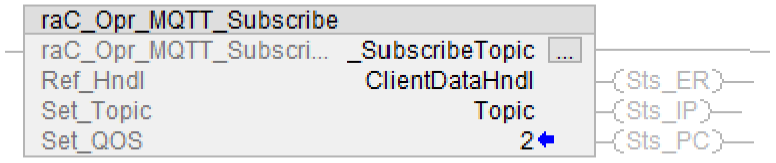
5 MQTT Subscribe

5.1 Functional Description

raC_Opr_MQTT_Subscribe instruction is used to send subscribe request to MQTT Broker. The instruction has parameters for Subscribe topic and requested Quality of Service.

The instruction execution is edge driven. Once true, the rung condition need not be maintained.

5.2 Instruction



5.3 InOut Data

InOut	Function / Description	Datatype
Ref_Handle	MQTT Client Data Handle	raC_UDT_Opr_MQTTHandle

5.4 Input Data

Input	Function / Description	Datatype
Set_Topic	Topic to be subscribed	STRING
Set_QoS	Requested Quality of Service. Valid value are 0,1,2.	DINT

5.5 Output Data

Output	Function / Description	Datatype
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_IP	Instruction is In Process	BOOL
Sts_PC	Instruction Process Complete	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT

5.6 Error Codes

The Error code description is same as for MQTT_Client instruction. Refer chapter 2.7.

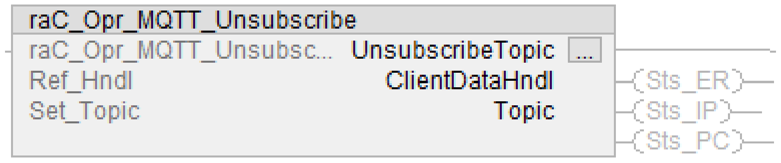
6 MQTT Unsubscribe

6.1 Functional Description

raC_Opr_MQTT_Unsubscribe instruction is used to send unsubscribe request to MQTT Broker. The instruction has parameters for topic to be unsubscribed.

The instruction execution is edge driven. Once true, the rung condition need not be maintained.

6.2 Instruction



6.3 InOut Data

InOut	Function / Description	Datatype
Ref_Handle	MQTT Client Data Handle	raC_UDT_Opr_MQTTHandle

6.4 Input Data

Input	Function / Description	Datatype
Set_Topic	Topic to be subscribed	STRING

6.5 Output Data

Output	Function / Description	Datatype
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_IP	Instruction is In Process	BOOL
Sts_PC	Instruction Process Complete	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT

6.6 Error Codes

The Error code description is same as for MQTT_Client instruction. Refer chapter 2.7.

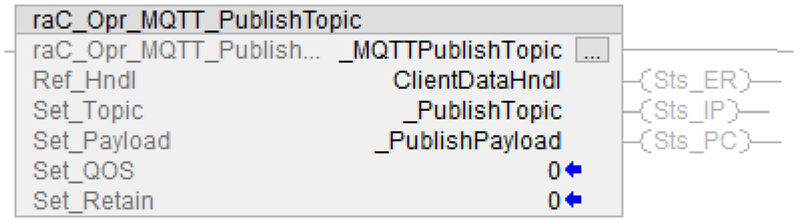
7 MQTT Publish

7.1 Functional Description

raC_Opr_MQTT_Publish instruction is used to publish topic data to MQTT Broker. The instruction has parameters for Topic, Payload, QoS and Retain.

The instruction execution is edge driven. Once true the rung condition need not be maintained.

7.2 Instruction



raC_Opr_MQTT_PublishTopic

raC_Opr_MQTT_Publish...	_MQTTPublishTopic
Ref_Hndl	ClientDataHndl
Set_Topic	_PublishTopic
Set_Payload	_PublishPayload
Set_QOS	0
Set_Retain	0

7.3 InOut Data

InOut	Function / Description	Datatype
Ref_Handle	MQTT Client Data Handle	raC_UDT_Opr_MQTTHandle

7.4 Input Data

Input	Function / Description	Datatype
Set_Topic	Topic to be subscribed	STRING
Set_Payload	MQTT Payload	raC_STR_MQTTPayload
Set_QOS	Requested Quality of Service. Valid values are 0,1,2	SINT
Set_Retain	Retain Publish Packet in MQTT Server	BOOL

7.5 Output Data

Output	Function / Description	Datatype
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_IP	Instruction is In Process	BOOL
Sts_PC	Instruction Process Complete	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT

7.6 Error Codes

The Error code description is same as for MQTT_Client instruction. Refer chapter 2.7.

8 MQTT Monitor

8.1 Functional Description

raC_Opr_MQTT_Monitor instruction is used to monitor a topic in the Reply Buffer of MQTT Client. Once the relevant topic data is received in reply buffer, the instruction will update the same in output parameters and increment the Out_Updates value.

The instruction has continuous and once mode of operation.

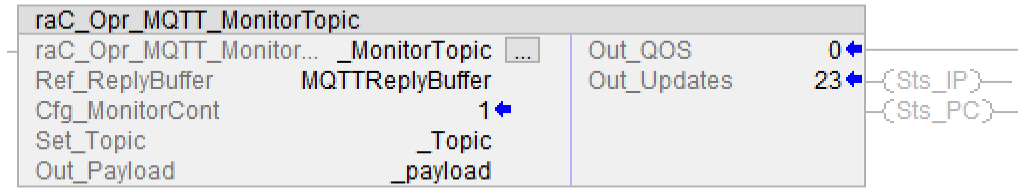
Once Mode:
Cfg_MonitorCont = 0. When rung condition is True the instruction monitor function will execute once. Once the topic is found the Sts_PC output is latched.

Continuous Mode:
Cfg_MonitorCont = 1. The instruction continuously monitors the topic and updates output until rung condition is True. Sts_PC output is not applicable in this mode.

WildCard:
'#' character can be used as wildcard in monitoring topic. For example, a topic 'Sensor/#' will look for all topics with text 'Sensor/'. Like 'Sensor/temperature', 'Sensor/level' etc. Entering just '#' in topic will enable monitor instruction to read every topic data in Reply buffer.

The instruction execution is level driven. The rung condition needs to be maintained true to execute.

8.2 Instruction


raC_Opr_MQTT_MonitorTopic
raC_Opr_MQTT_Monitor... _MonitorTopic
Ref_ReplyBuffer MQTTReplyBuffer
Cfg_MonitorCont 1
Set_Topic _Topic
Out_Payload _payload
Out_QOS 0
Out_Updates 23
Sts_IP
Sts_PC

8.3 InOut Data

InOut	Function / Description	Datatype
Ref_ReplyBuffer	Reply Buffer	raC_UDT_Opr_MQTTBuffer

8.4 Input Data

Input	Function / Description	Datatype
Set_Topic	Topic to be monitored. Can use '#' as wildcard.	STRING
Cfg_MonitorCont	Monitor Continuous	BOOL

8.5 Output Data

Output	Function / Description	Datatype
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR for Additional Error Information	BOOL
Sts_IP	Instruction is In Process	BOOL
Sts_PC	Instruction Process Complete	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT
Out_Payload	MQTT Payload	raC_STR_MQTTPayload
Out_QOS	Quality of Service	DINT
Out_Updates	Update Count. Increments when new update is found	DINT

8.6 Error Codes

The Error code description is same as for MQTT_Client instruction. Refer chapter 2.7.

9 Application Code Manager

9.1 Definition Object raC_Opr_MQTT

This object contains the AOI definition of raC_Opr_MQTT_Client, raC_Opr_MQTT_Connect, raC_Opr_MQTT_Disconnect, raC_Opr_MQTT_Publish, raC_Opr_MQTT_Subscribe, raC_Opr_MQTT_Unsubscribe, raC_Opr_MQTT_Monitor and used as linked library to implement object. This gives flexibility to choose to instantiate only definition and create custom implement code. User may also create their own implement library and link with this definition library object.

Attachments

Name	Description	File Name	Extraction path
V1_{LibraryName}	Reference Manual	RM-{LibraryName}.pdf	{ProjectName}\Documentation

9.2 Implementation Object: raC_LD_MQTT_Client

This library object contains implement code of raC_Opr_MQTT_Client instruction.

Implement Language: Ladder Diagram

Parameter Name	Default Value	Description
ObjectName	raC_LD_MQTT_Client	Object Name
RoutineName	{ObjectName}	Insert Routine Name. Routine will be created, and Object implement rung inserted. A JSR will be inserted in Main Routine. If routine name already exists, then object will be inserted into existing routine. By Default, is Set to Object Name
TagName	{ObjectName}	Enter the MQTT Client Instance Tag Name. By Default, is Set to _Object Name
ControllerType	1756-L8x	Select the controller used from the drop down list
BackplaneCommCard	False	Set when using Comm card on backplane. Example EN2T(R), EN3T(R), EWEB etc..
CommCardType	1756-ENxx	Select the comm card type from drop down list
CommCardSlotNumber	0	Enter the slot number of CommCard
TCPPortNumber	1883	TCP port number used by the broker. Standard is 1883 Non-TLS and 8883 TLS.
ClientId	{ObjectName}	Enter the ClientId
Username		Enter Username of MQTT Broker. Leave blank if not used
Password		Enter Password of MQTTBroker. Leave blank if not used.
WillTopic		Enter the WillTopic. Leave blank if not used
WillMessage		Enter WillMessage for Client. Leave blank if not used
WillQoS	0	Enter WillQos. Leave default if not used
WillRetain	False	Select if WillRetain is enabled

Linked Library

Link Name	Catalog Number	Revision	Solution	Category
raC_Opr_MQTT	raC_Opr_MQTT	1.0	(RA-LIB) Common	CommMQTT

9.3 Implementation Object: raC_LD_MQTT_Methods

This library object contains implement rung of MQTT Method instructions (Connect, Disconnect, Subscribe, Unsubscribe, Publish and Monitor).

Implement Language: Ladder Diagram

Parameter Name	Default Value	Description
ObjectName	raC_LD_MQTT_Methods	Object Name
RoutineName	{ObjectName}	Insert Routine Name. Routine will be created, and Object implement rung inserted. A JSR will be inserted in Main Routine. If routine name already exists, then object will be inserted into existing routine. By Default, is Set to Object Name
IncludeConnect	False	Select True to include Connect Instruction Implementation
IncludeDisconnect	False	Select True to include Connect Instruction Implementation
IncludeSubscribe	False	Select True to include Disconnect Instruction Implementation
IncludeUnsubscribe	False	Select True to include Unsubscribe Instruction Implementation
IncludePublish	False	Select True to include Publish Instruction Implementation
IncludeMonitor	False	Select True to include Monitor Instruction Implementation
Connect_TagName	{ObjectName}_Connect	Enter Connect instance TagName
Disconnect_TagName	{ObjectName}_Disconnect	Enter Disconnect instance TagName
Subscribe_TagName	{ObjectName}_Subscribe	Enter Subscribe instance TagName
Unsubscribe_TagName	{ObjectName}_Unsubscribe	Enter Unsubscribe instance TagName
Publish_TagName	{ObjectName}_Publish	Enter Publish instance TagName
Monitor_TagName	{ObjectName}_Monitor	Enter Monitor instance TagName

Linked Library

Link Name	Catalog Number	Revision	Solution	Category
raC_Opr_MQTT	raC_Opr_MQTT	1.0	(RA-LIB) Common	CommMQTT
MQTTClient	raC_LD_MQTT_Client	1.0	(RA-LIB) Common	CommMQTT

10 Application Example

The following are the application input.

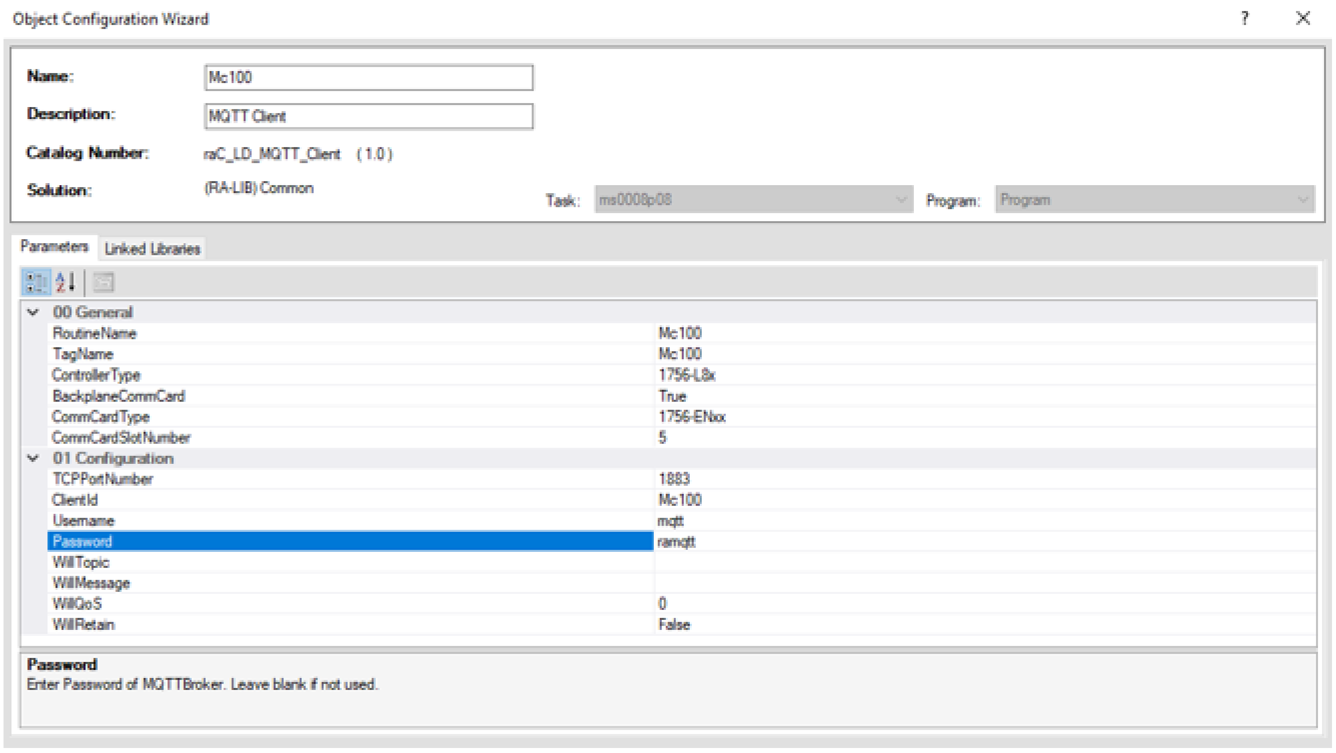
- Controller - 1756-L85E
- Comm Card - 1756-EN2TR
- Comm Card Slot – 5
- MQTT Broker IP Address – 192.168.1.99
- MQTT Broker Username – mqtt
- MQTT Broker Password – ramqtt
- ClientID – Mc100

Machine Libraries can be instantiated using Application Code Manager or in Studio5000 Logix Designer using Studio5000 Import Plugin.

10.1 Instantiate using Application Code Manager

1. Add raC_LD_MQTT_Client library and enter the parameters

    - Name: Mc100
    - RoutineName: Mc100
    - TagName: Mc100
    - ControllerType: 1756-L8x
    - BackplaneCommCard: True
    - CommCardType: 1756-ENxx
    - CommCardSlotNumber: 5
    - TCPPortMumber: 1883
    - ClientId: Mc100
    - Username: mqtt
    - Password: ramqtt
    - Leave WillTopic, WillMessage, WillQos and WillRetain to default values.


Object Configuration Wizard

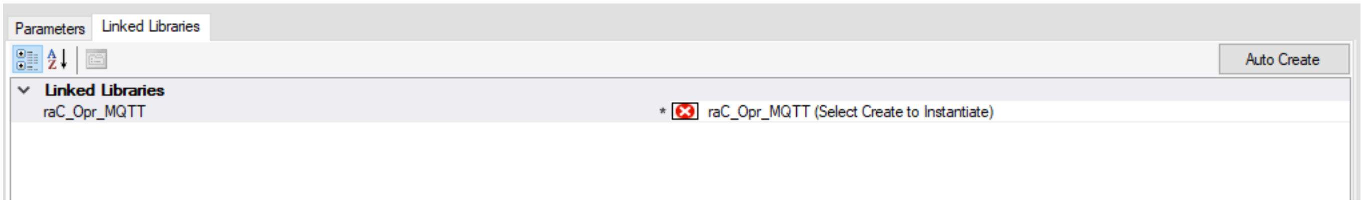
Name:	Mc100
Description:	MQTT Client
Catalog Number:	raC_LD_MQTT_Client (1.0)
Solution:	(RA-LIB) Common

Task: ms0008p08 Program: Program

00 General	
RoutineName	Mc100
TagName	Mc100
ControllerType	1756-L8x
BackplaneCommCard	True
CommCardType	1756-ENxx
CommCardSlotNumber	5
01 Configuration	
TCPPortNumber	1883
ClientId	Mc100
Username	mqtt
Password	ramqtt
WillTopic	
WillMessage	
WillQoS	0
WillRetain	False

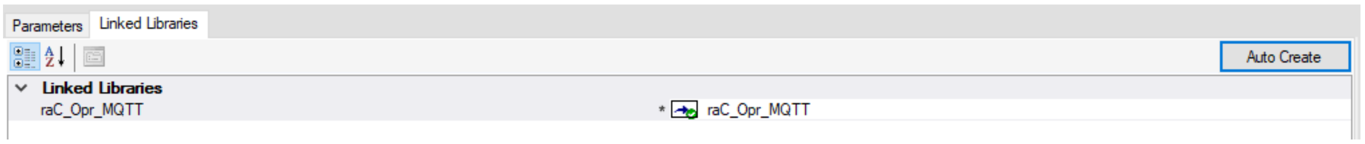
Password
Enter Password of MQTTBroker. Leave blank if not used.

2. Click Next to open Linked Libraries tab



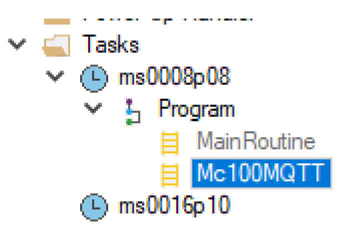
Linked Libraries	
raC_Opr_MQTT	* raC_Opr_MQTT (Select Create to Instantiate)

3. Click 'Auto Create' to create the linked libraries

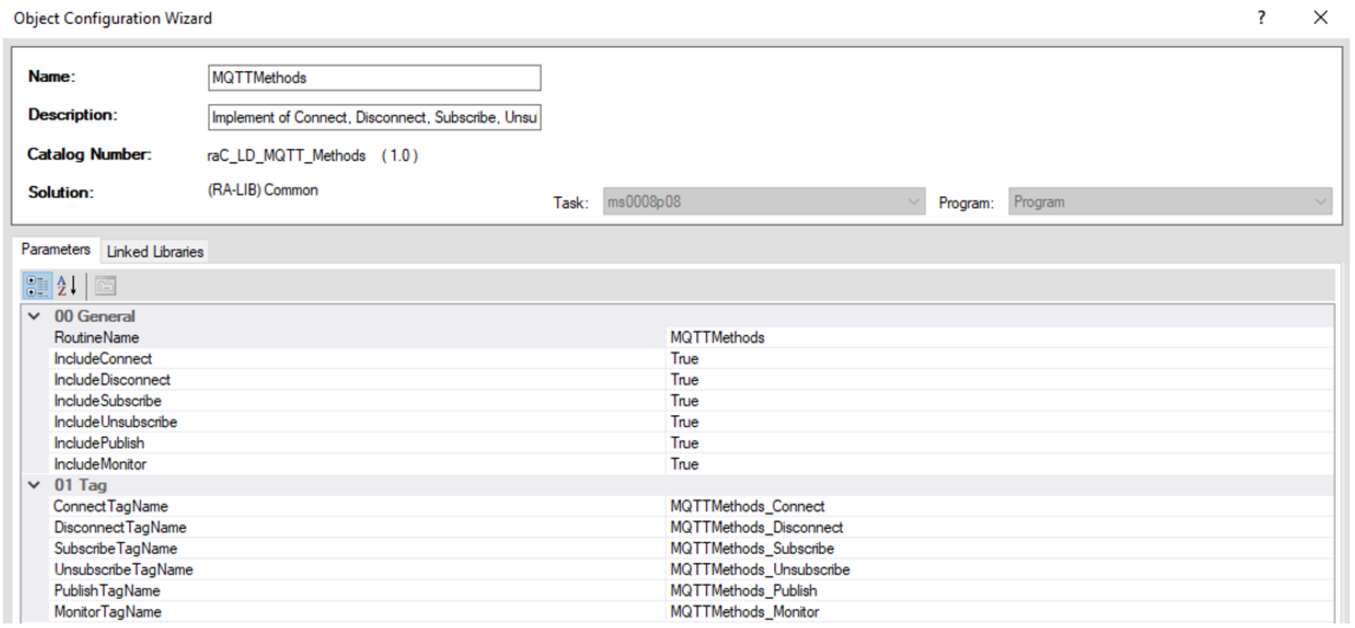


Linked Libraries	
raC_Opr_MQTT	* raC_Opr_MQTT

4. Click Finish to complete the instantiation



5. Add raC_LD_MQTT_Methods library in the program. Select True as all methods are needed for the application.



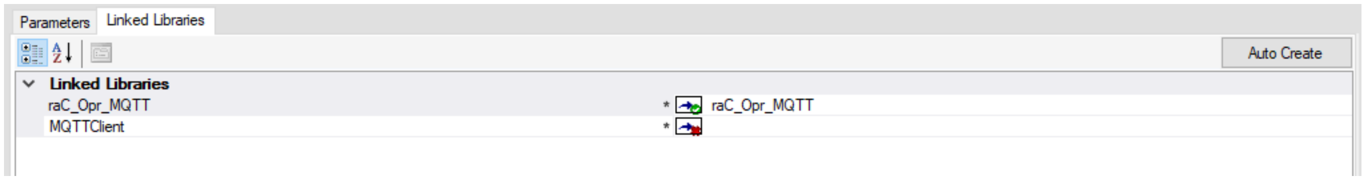
Object Configuration Wizard

Name:	MQTTMethods
Description:	Implement of Connect, Disconnect, Subscribe, Unsu
Catalog Number:	raC_LD_MQTT_Methods (1.0)
Solution:	(RA-LIB) Common
Task:	ms0008p08
Program:	Program

Parameters | Linked Libraries

00 General	
RoutineName	MQTTMethods
IncludeConnect	True
IncludeDisconnect	True
IncludeSubscribe	True
IncludeUnsubscribe	True
IncludePublish	True
IncludeMonitor	True
01 Tag	
ConnectTagName	MQTTMethods_Connect
DisconnectTagName	MQTTMethods_Disconnect
SubscribeTagName	MQTTMethods_Subscribe
UnsubscribeTagName	MQTTMethods_Unsubscribe
PublishTagName	MQTTMethods_Publish
MonitorTagName	MQTTMethods_Monitor

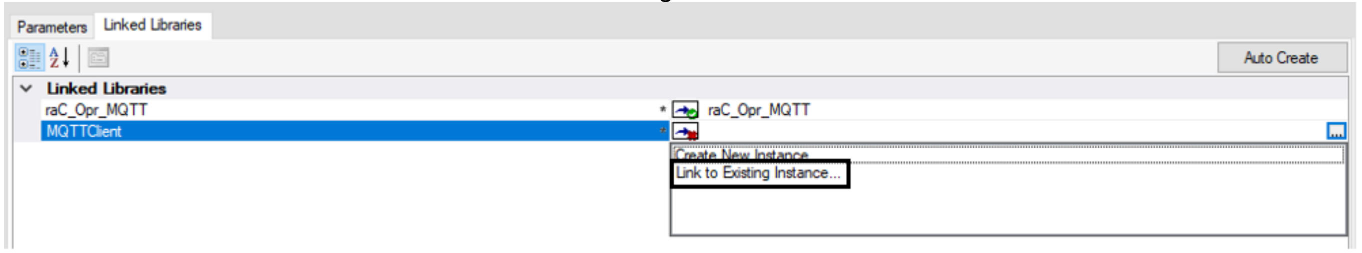
6. Click Next to move to Linked Libraries tab



Parameters | Linked Libraries | Auto Create

Linked Libraries	
raC_Opr_MQTT	* raC_Opr_MQTT
MQTTClient	*

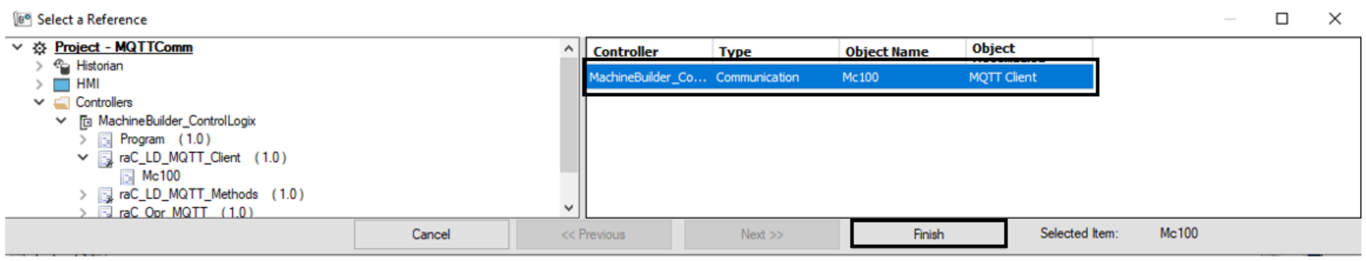
7. raC_Opr_MQTT library is already added in the project when instantiating MQTT Client library. Click on the browse button of MQTTClient and select ‘Link to Existing Instance’



Parameters | Linked Libraries | Auto Create

Linked Libraries	
raC_Opr_MQTT	* raC_Opr_MQTT
MQTTClient	*

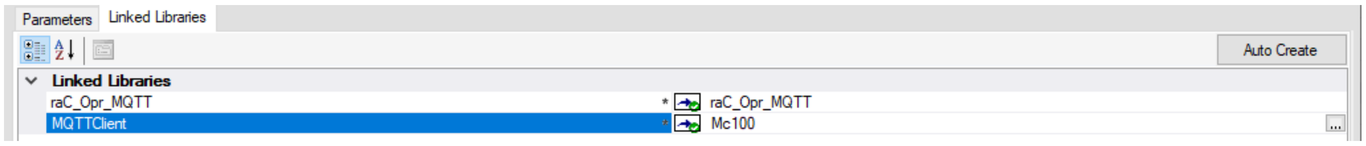
8. Select the Mc100 instance of MQTT Client library previously added in the project and click Finish



Select a Reference

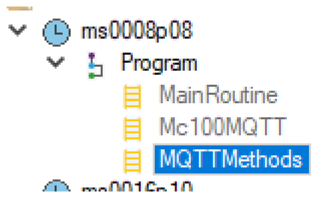
Controller	Type	Object Name	Object
MachineBuilder_Co...	Communication	Mc100	MQTT Client

Selected Item: Mc100

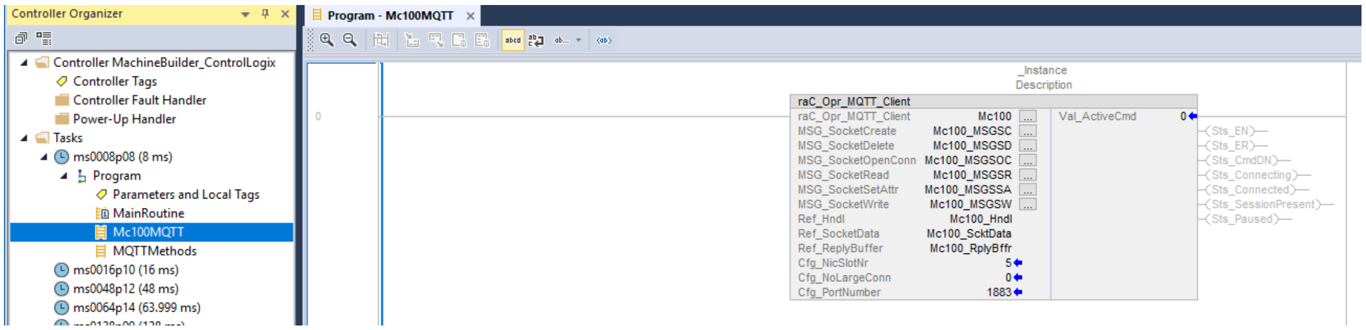


Linked Libraries	
raC_Opr_MQTT	raC_Opr_MQTT
MQTTClient	Mc100

9. Click Finish to complete the instantiation of MQTT Methods library

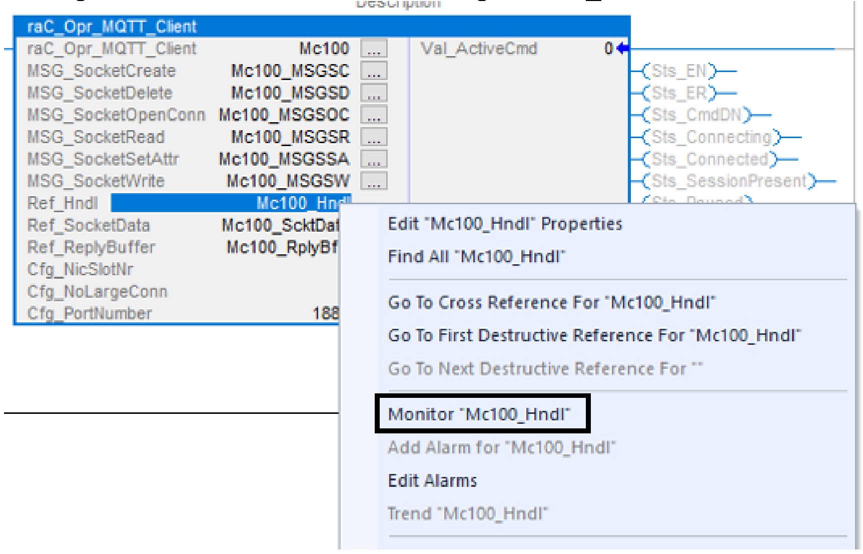


10. Generate the ACD and open the code in Studio5000 Logix Designer. Open Mc100MQTT routine. Note the configuration parameter of MQTT Client instruction is already preconfigured.

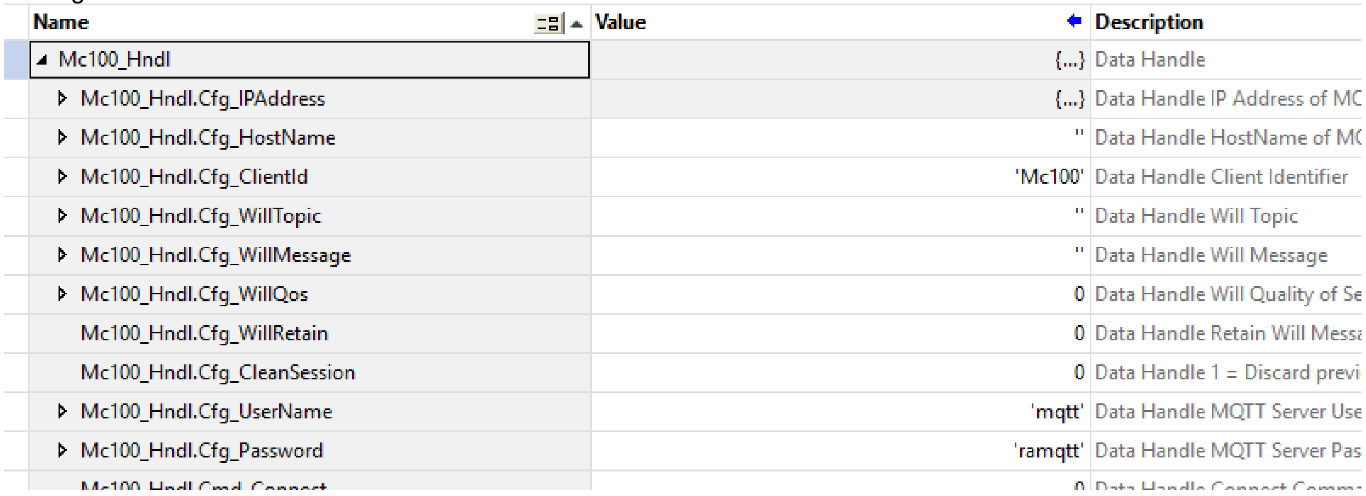


raC_Opr_MQTT_Client		
raC_Opr_MQTT_Client	Mc100	Val_ActiveCmd 0
MSG_SocketCreate	Mc100_MSGSC	Sts_EN
MSG_SocketDelete	Mc100_MSGSD	Sts_ER
MSG_SocketOpenConn	Mc100_MSGSOC	Sts_CmdDN
MSG_SocketRead	Mc100_MSGSR	Sts_Connecting
MSG_SocketSetAttr	Mc100_MSGSSA	Sts_Connected
MSG_SocketWrite	Mc100_MSGSW	Sts_SessionPresent
Ref_Hndl	Mc100_Hndl	Sts_Paused
Ref_SocketData	Mc100_ScktData	
Ref_ReplyBuffer	Mc100_RplyBffr	
Cfg_NicSlotNr	5	
Cfg_NoLargeConn	0	
Cfg_PortNumber	1883	

11. Right-Click on Client Data Handle tag "Mc100_Hndl" and select monitor

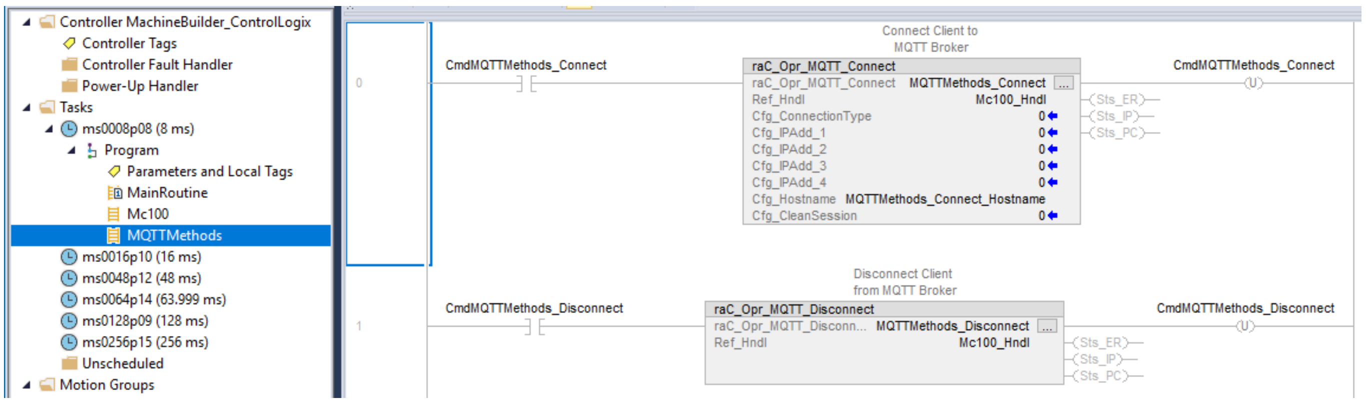

Edit "Mc100_Hndl" Properties
Find All "Mc100_Hndl"
Go To Cross Reference For "Mc100_Hndl"
Go To First Destructive Reference For "Mc100_Hndl"
Go To Next Destructive Reference For ""
Monitor "Mc100_Hndl"
Add Alarm for "Mc100_Hndl"
Edit Alarms
Trend "Mc100_Hndl"

12. Note the configuration parameters like ClientId, Username and Password are as entered in Application Code Manager.



Name	Value	Description
Mc100_Hndl	{...}	Data Handle
Mc100_Hndl.Cfg_IPAddress	{...}	Data Handle IP Address of MC
Mc100_Hndl.Cfg_HostName	''	Data Handle HostName of MC
Mc100_Hndl.Cfg_ClientId	'Mc100'	Data Handle Client Identifier
Mc100_Hndl.Cfg_WillTopic	''	Data Handle Will Topic
Mc100_Hndl.Cfg_WillMessage	''	Data Handle Will Message
Mc100_Hndl.Cfg_WillQos	0	Data Handle Will Quality of Se
Mc100_Hndl.Cfg_WillRetain	0	Data Handle Retain Will Messa
Mc100_Hndl.Cfg_CleanSession	0	Data Handle 1 = Discard previ
Mc100_Hndl.Cfg_UserName	'mqtt'	Data Handle MQTT Server Use
Mc100_Hndl.Cfg_Password	'ramqtt'	Data Handle MQTT Server Pas
Mc100_Hndl.Cmd_Connect	0	Data Handle Connect Comm

13. Open MQTTMethods routine and note the instructions for Connect, Disconnect, Subscribe, Unsubscribe, Publish and Monitor.

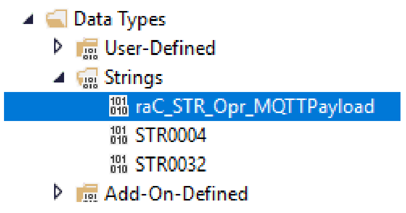


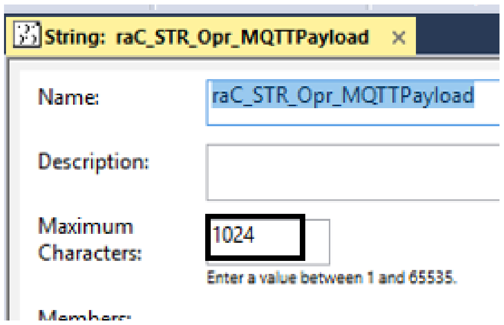
```
0   CmdMQTTMethods_Connect        Connect Client to MQTT Broker                              CmdMQTTMethods_Connect
    ] [                           raC_Opr_MQTT_Connect                                       (U)
                                  raC_Opr_MQTT_Connect  MQTTMethods_Connect  ...
                                  Ref_Hndl                       Mc100_Hndl      (Sts_ER)
                                  Cfg_ConnectionType                      0      (Sts_IP)
                                  Cfg_IPAdd_1                             0      (Sts_PC)
                                  Cfg_IPAdd_2                             0
                                  Cfg_IPAdd_3                             0
                                  Cfg_IPAdd_4                             0
                                  Cfg_Hostname  MQTTMethods_Connect_Hostname
                                  Cfg_CleanSession                        0

1   CmdMQTTMethods_Disconnect     Disconnect Client from MQTT Broker                         CmdMQTTMethods_Disconnect
    ] [                           raC_Opr_MQTT_Disconnect                                    (U)
                                  raC_Opr_MQTT_Disconn...  MQTTMethods_Disconnect  ...
                                  Ref_Hndl                       Mc100_Hndl      (Sts_ER)
                                                                                 (Sts_IP)
                                                                                 (Sts_PC)
```

14. Verify if the number of characters in raC_STR_Opr_MQTTPayload string data type is set as per payload size required for the application. Open the properties and note the default size of 16384 bytes. For this application the payload size of 1024 bytes is enough. Modify the Maximum Characters to 1024.

Optimizing the payload size will help in optimizing memory requirement.




String: raC_STR_Opr_MQTTPayload

Name: raC_STR_Opr_MQTTPayload

Description:

Maximum Characters: 1024

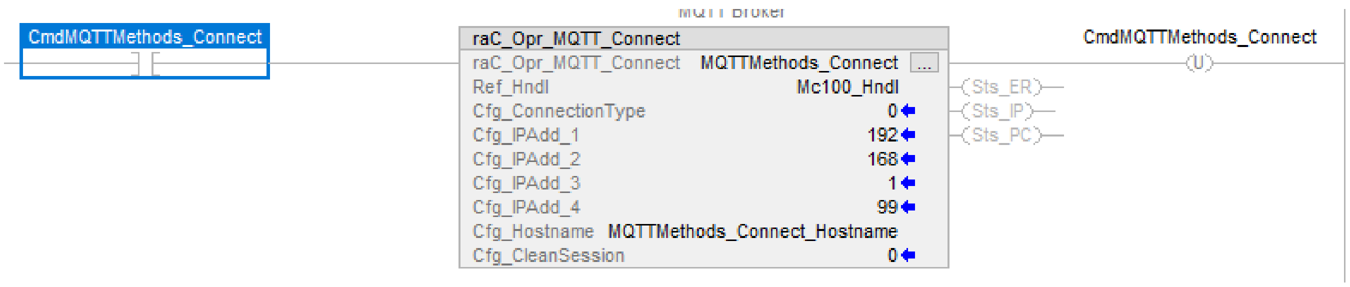
Enter a value between 1 and 65535.

Members:

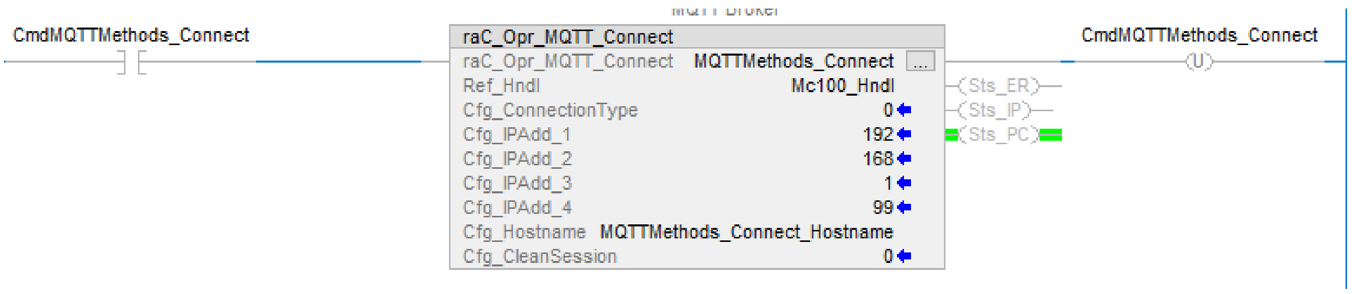
10.2 Testing the Application

10.2.1 Connect Client to MQTT Broker

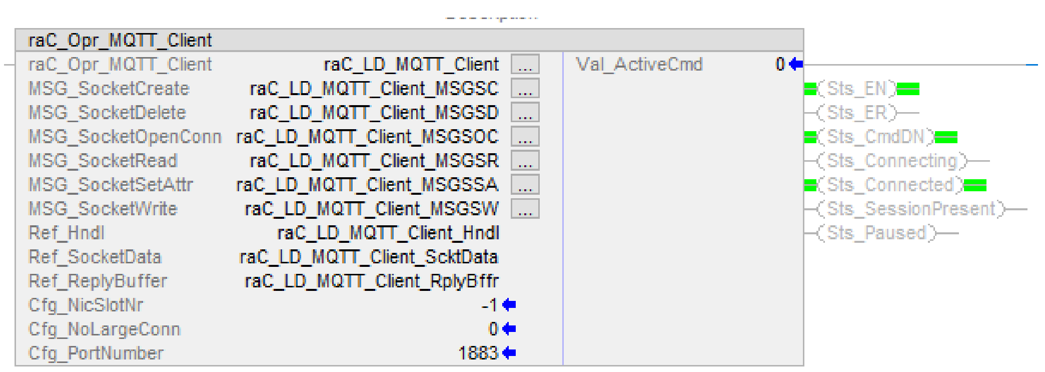
1. Client can be connected to MQTT Broker using IP Address or HostName. For this example, IP Address will be used. Enter the IP Address of MQTT Broker as 192.168.1.99. Toggle the CmdMQTTMethods_Connect tag.


```
CmdMQTTMethods_Connect                MQTT Broker
                          raC_Opr_MQTT_Connect
                          raC_Opr_MQTT_Connect  MQTTMethods_Connect        CmdMQTTMethods_Connect
                          Ref_Hndl                      Mc100_Hndl  (Sts_ER)      (U)
                          Cfg_ConnectionType                    0   (Sts_IP)
                          Cfg_IPAdd_1                         192   (Sts_PC)
                          Cfg_IPAdd_2                         168
                          Cfg_IPAdd_3                           1
                          Cfg_IPAdd_4                          99
                          Cfg_Hostname  MQTTMethods_Connect_Hostname
                          Cfg_CleanSession                      0
```

2. Wait for Sts_PC output


```
CmdMQTTMethods_Connect                MQTT Broker
                          raC_Opr_MQTT_Connect
                          raC_Opr_MQTT_Connect  MQTTMethods_Connect        CmdMQTTMethods_Connect
                          Ref_Hndl                      Mc100_Hndl  (Sts_ER)      (U)
                          Cfg_ConnectionType                    0   (Sts_IP)
                          Cfg_IPAdd_1                         192   (Sts_PC)
                          Cfg_IPAdd_2                         168
                          Cfg_IPAdd_3                           1
                          Cfg_IPAdd_4                          99
                          Cfg_Hostname  MQTTMethods_Connect_Hostname
                          Cfg_CleanSession                      0
```

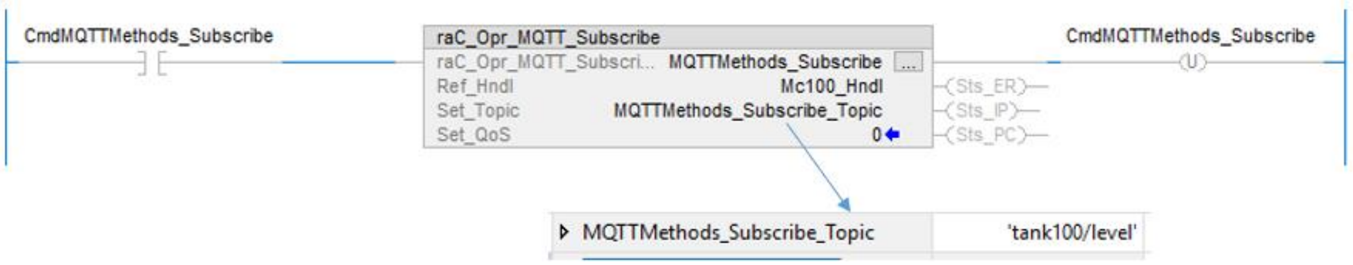
3. The Sts_Connected can be referred from Mc100_Hndl tag or Client Instruction. Sts_CmdDN indicates that last command was executed successfully. If Sts_ER is set, then refer to Sts_ERR for error code. The help text of the instruction contains description of error codes.



raC_Opr_MQTT_Client	
raC_Opr_MQTT_Client	raC_LD_MQTT_Client
MSG_SocketCreate	raC_LD_MQTT_Client_MSGSC
MSG_SocketDelete	raC_LD_MQTT_Client_MSGSD
MSG_SocketOpenConn	raC_LD_MQTT_Client_MSGSOC
MSG_SocketRead	raC_LD_MQTT_Client_MSGSR
MSG_SocketSetAttr	raC_LD_MQTT_Client_MSGSSA
MSG_SocketWrite	raC_LD_MQTT_Client_MSGSW
Ref_Hndl	raC_LD_MQTT_Client_Hndl
Ref_SocketData	raC_LD_MQTT_Client_ScktData
Ref_ReplyBuffer	raC_LD_MQTT_Client_RplyBffr
Cfg_NicSlotNr	-1
Cfg_NoLargeConn	0
Cfg_PortNumber	1883
Val_ActiveCmd	0

10.2.2 Subscribe and Monitor Topic

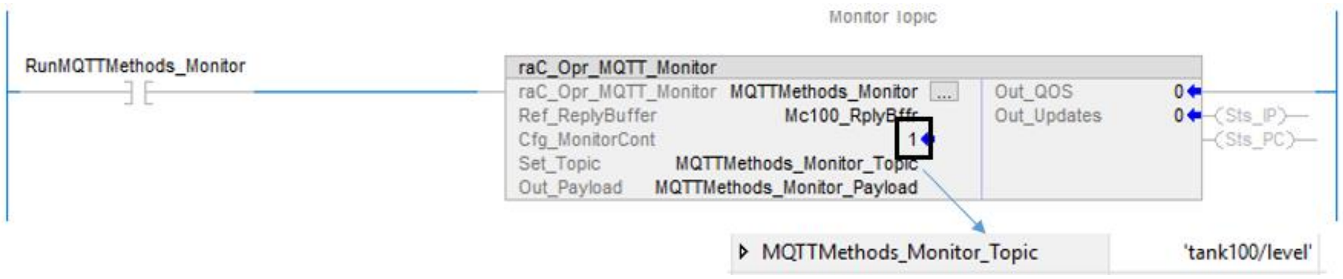
1. A topic can be subscribed and monitored for payload using Subscribe and Monitor Methods. For this example subscribe and monitor the topic ‘tank100/level’. Enter the same in topic of Subscribe.



raC_Opr_MQTT_Subscribe	
raC_Opr_MQTT_Subscri...	MQTTMethods_Subscribe
Ref_Hndl	Mc100_Hndl
Set_Topic	MQTTMethods_Subscribe_Topic
Set_QoS	0

MQTTMethods_Subscribe_Topic	'tank100/level'

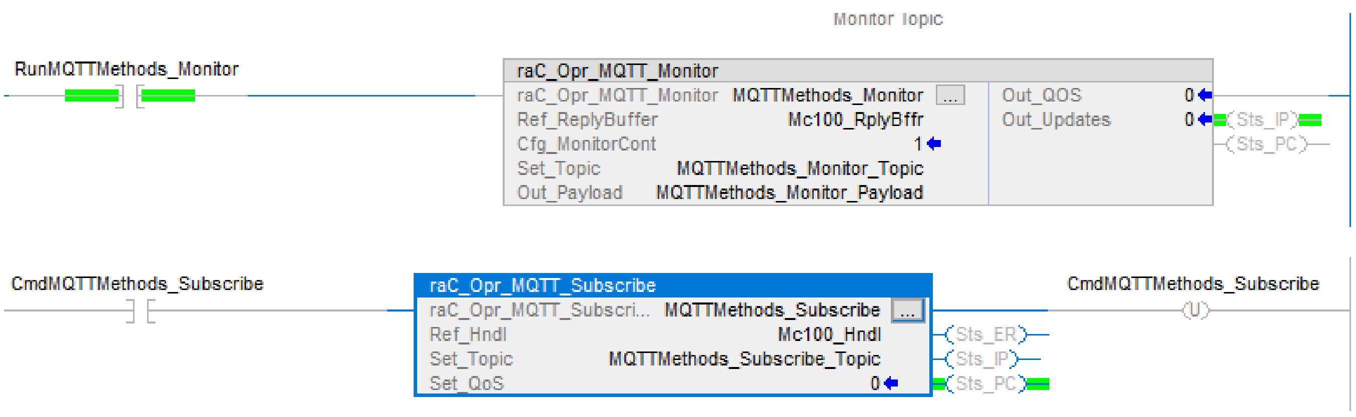
2. Enter the topic as ‘tank100/level’ for monitor instruction and set Cfg_MonitorCont as 1 so the monitor instruction is continuously monitoring and updating Out_Payload when new payload data is received by Client.



raC_Opr_MQTT_Monitor	
raC_Opr_MQTT_Monitor	MQTTMethods_Monitor
Ref_ReplyBuffer	Mc100_RplyBffr
Cfg_MonitorCont	1
Set_Topic	MQTTMethods_Monitor_Topic
Out_Payload	MQTTMethods_Monitor_Payload
Out_QOS	0
Out_Updates	0

MQTTMethods_Monitor_Topic	'tank100/level'

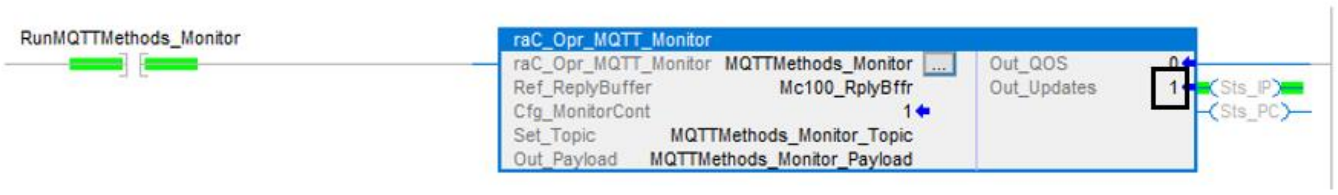
3. Since the Client is already connected to Broker, execute the Subscribe and Monitor instructions. Note, the Sts_IP in Monitor indicates that instruction is monitoring the reply buffer. Sts_PC in subscribe instruction indicates that topic has been subscribed.



```
Monitor Topic
RunMQTTMethods_Monitor
raC_Opr_MQTT_Monitor
raC_Opr_MQTT_Monitor   MQTTMethods_Monitor      Out_QOS       0
Ref_ReplyBuffer        Mc100_RplyBffr           Out_Updates   0
Cfg_MonitorCont        1                                        (Sts_IP)
Set_Topic              MQTTMethods_Monitor_Topic                (Sts_PC)
Out_Payload            MQTTMethods_Monitor_Payload

CmdMQTTMethods_Subscribe                                        CmdMQTTMethods_Subscribe
raC_Opr_MQTT_Subscribe                                          (U)
raC_Opr_MQTT_Subscri...  MQTTMethods_Subscribe
Ref_Hndl                 Mc100_Hndl                             (Sts_ER)
Set_Topic                MQTTMethods_Subscribe_Topic            (Sts_IP)
Set_QoS                  0                                      (Sts_PC)
```

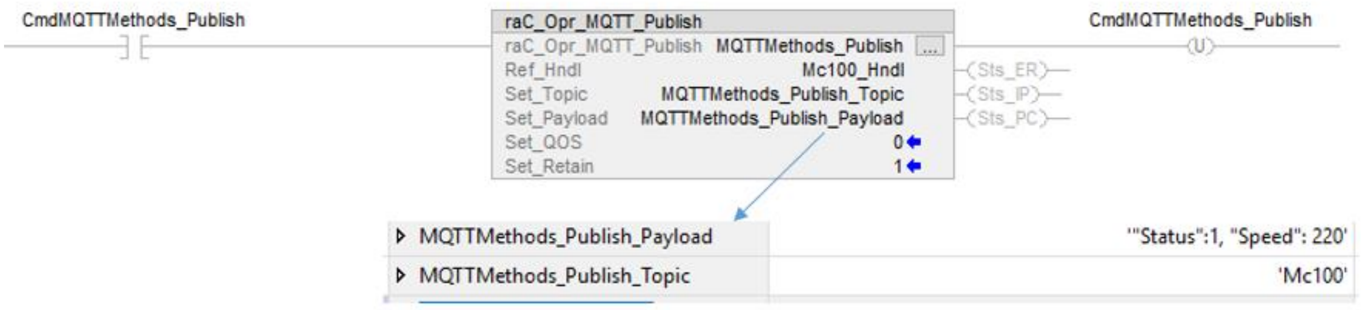
4. The monitor instruction monitors the Reply Buffer for the relevant topic. Once found, the Out_Updates is incremented to indicate a new data found. The Out_Payload and Out_QoS gives the payload data and QoS of the topic.



```
RunMQTTMethods_Monitor
raC_Opr_MQTT_Monitor
raC_Opr_MQTT_Monitor   MQTTMethods_Monitor      Out_QOS       0
Ref_ReplyBuffer        Mc100_RplyBffr           Out_Updates   1   (Sts_IP)
Cfg_MonitorCont        1                                          (Sts_PC)
Set_Topic              MQTTMethods_Monitor_Topic
Out_Payload            MQTTMethods_Monitor_Payload
```

10.2.3 Publish

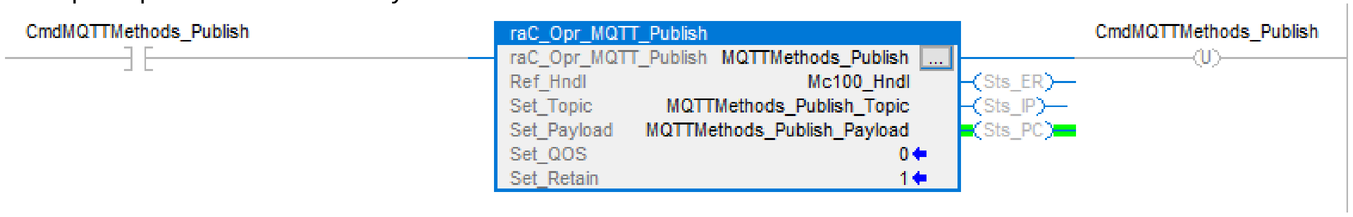
1. To publish a topic enter the topic and payload data in Publish instruction and execute it. For this example the topic will be ‘Mc100’, payload will contain Status, and Speed. Enable Retain so the last topic data is retained by MQTT Broker.



```
CmdMQTTMethods_Publish                                          CmdMQTTMethods_Publish
raC_Opr_MQTT_Publish                                            (U)
raC_Opr_MQTT_Publish   MQTTMethods_Publish
Ref_Hndl               Mc100_Hndl                               (Sts_ER)
Set_Topic              MQTTMethods_Publish_Topic                (Sts_IP)
Set_Payload            MQTTMethods_Publish_Payload              (Sts_PC)
Set_QOS                0
Set_Retain             1
```

MQTTMethods_Publish_Payload	'"Status":1, "Speed": 220'
MQTTMethods_Publish_Topic	'Mc100'

2. Ensure Sts_Connected is available in Mc100_Hndl tag. Execute the Publish instruction. Sts_PC indicates that the topic is published successfully.



11 Appendix

General

This document provides a programmer with details on this instruction for a Logix-based controller, its Application Code Manager library content, and visualization content, if applicable. This document assumes that the programmer is already familiar with how the Logix-based controller stores and processes data.

Novice programmers should read all the details about an instruction before using the instruction. Experienced programmers can refer to the instruction information to verify details.

IMPORTANT This object includes a Logix Designer Asset for use with Version 30 or later of Studio 5000 Logix Designer.

Common Information for All Instructions

Rockwell Automation Application Content may contain many common attributes or objects. Refer to the following reference materials for more information:

- Foundations of Modular Programming, **IA-RM001C-EN-P**

Conventions and Related Terms

Data - Set and Clear

This manual uses set and clear to define the status of bits (Booleans) and values (non-Booleans):

This Term:	Means:
Set	The bit is set to 1 (ON) A value is set to any non-zero number
Clear	The bit is cleared to 0 (OFF) All the bits in a value are cleared to 0

Signal Processing - Edge and Level

This manual uses Edge and Level to describe how bit (BOOL) Commands, Settings, Configurations and Inputs to this instruction are sent by other logic and processed by this instruction.

Send/Receive Method:	Description:
Edge	• Action is triggered by "rising edge" transition of input (0-1) • Separate inputs are provided for complementary functions (such as "enable" and "disable") • Sending logic SETS the bit (writes a 1) to initiate the action; this instruction CLEARS the bit (to 0) immediately, then acts on the request if possible • LD: use conditioned OTL (Latch) to send • ST: use conditional assignment [if (condition) then bit:=1;] to send • FBD: OREF writes a 1 or 0 every scan, should use Level, not Edge Edge triggering allows multiple senders per Command, Setting, Configuration or Input (many-to-one relationship)
Level	• Action ("enable") is triggered by input being at a level (in a state, usually 1) • Opposite action ("disable") is triggered by input being in opposite state (0) • Sending logic SETS the bit (writes a 1) or CLEARS the bit (writes a 0); this instruction does not change the bit • LD: use OTE (Energize) to send • ST: use unconditional assignment [bit:= expression_resulting_in_1_or_0;] or "if-then-else" logic [if (condition) then bit:= 1; else bit:= 0;] • FBD: use OREF to the input bit Level triggering allows only one sender can drive each Level

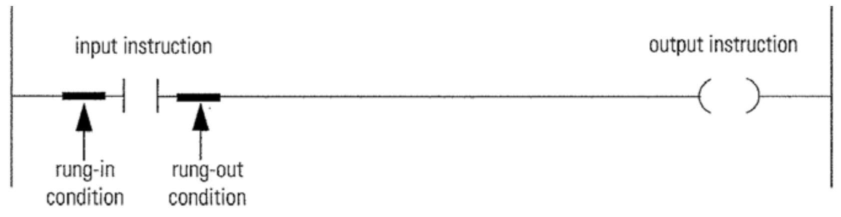
Instruction Execution - Edge and Continuous

This manual uses Edge and Continuous to describe how an instruction is designed to be executed.

Method:	Description:
Edge	• Instruction Action is triggered by "rising edge" transition of the rung-in-condition
Continuous	• Instruction Action is triggered by input being at a level (in a state, usually 1) • Opposite action is triggered by input being in opposite state (0) • Instructions designed for continuous execution should typically be used on rungs without input conditions present allowing the instruction to be continuously scanned

Relay Ladder Rung Condition

The controller evaluates ladder instructions based on the rung condition preceding the instruction (rung-in condition). Based on the rung-in condition and the instruction, the controller sets the rung condition following the instruction (rung-out condition), which in turn, affects any subsequent instruction.



If the rung-in condition to an input instruction is true, the controller evaluates the instruction and sets the rung-out condition based on the results of the instruction. If the instruction evaluates to true, the rung-out condition is true; if the instruction evaluates to false, the rung-out condition is false.

IMPORTANT The rung-in condition is reflected in the EnableIn parameter and determines how the system performs each Add-On Instruction. If the EnableIn signal is TRUE, the system performs the instruction’s main logic routine. Conversely, if the EnableIn signal is FALSE, the system performs the instruction’s EnableInFalse routine.

The instruction’s main logic routine sets/clears the EnableOut parameter, which then determines the rung-out condition. The EnableInFalse routine cannot set the EnableOut parameter. If the rung-in condition is FALSE, then the EnableOut parameter and the rung-out condition will also be FALSE.

Pre-scan

On transition into RUN, the controller performs a pre-scan before the first scan. Pre-scan is a special scan of all routines in the controller. The controller scans all main routines and subroutines during pre-scan but ignores jumps that could skip the execution of instructions. The controller performs all FOR loops and subroutine calls. If a subroutine is called more than once, it is performed each time it is called. The controller uses pre-scan of relay ladder instructions to reset non-retentive I/O and internal values.

During pre-scan, input values are not current, and outputs are not written. The following conditions generate pre-scan:

- Transition from Program to Run mode.
- Automatically enter Run mode from a power-up condition.

Pre-scan does not occur for a program when:

- Program becomes scheduled while the controller is running.
- Program is unscheduled when the controller enters Run mode.

IMPORTANT The Pre-scan process performs the Process Add-On Instruction's logic routine as FALSE and then performs its Pre-scan routine as TRUE.