**Message Queuing Telemetry Transport (MQTT) Library**

# **Introduction**

MQTT (Message Queuing Telemetry and Transport) is a publish/subscribe messaging transport protocol. It is widely used in many situations, such as communications in Machine to Machine (M2M) or Internet of Things (IoT) applications, especially in bandwidth and/or memory constrained environments. This protocol runs over TCP/IP, or can over other network protocols that provide ordered, lossless, bi-directional connections. In this implementation of MQTT is implemented over TCP.

This MQTT implementation supports all the required features and message types of this protocol along with some optional features.

Supported message types:

* CONNECT
* CONNACK
* PUBLISH (QoS level = 0)
* PINGREQ
* PINGRESP
* SUBSCRIBE (QoS level = 0, beta)
* DISCONNECT

This document describes the structure and the APIs implemented as part of the MQTT library. The public APIs are used by user application for performing all the MQTT specific tasks such as establishing a MQTT connection, publishing data, sending ping requests, etc. The private APIs are used internally by the MQTT library for maintaining its modularity.

# **Public APIs**

## INITIALISE

### MQTT\_ClientInitialise

* + 1. Description

void MQTT\_ClientInitialise(void)

MQTT\_ClientInitialise API initializes all the parameters in the send and receive data buffer used by the MQTT library. The size of these buffers is dictated by the application requirements. The buffers are used by MQTT library to handler packet exchange with MQTT broker.

* + 1. Parameters

None.

* + 1. Return Values

None.

## CONNECT

### MQTT\_CreateConnectPacket

* + 1. Description

bool MQTT\_CreateConnectPacket(mqttConnectPacket \*newConnectPacket)

MQTT\_CreateConnectPacket API creates a CONNECT packet structure, which follows MQTT standard.

* + 1. Parameters

A pointer that points to a MQTT CONNECT packet structure mqttConnectPacket.

* + 1. Return Values

A bool type value indicating whether a connect data packet structure is initialized successfully. A return value of ‘true’ implies that CONNECT packet has been successfully created.

## PUBLISH

### MQTT\_CreatePublishPacket

* + 1. Description

bool MQTT\_CreatePublishPacket(mqttPublishPacket \*newPublishPacket)

MQTT\_CreatePublishPacket API creates a MQTT publish data packet structure, which follows MQTT standard.

* + 1. Parameters

A pointer that points to a MQTT PUBLISH packet structure mqttPublishPacket.

* + 1. Return Values

A bool value indicating whether a publish data packet structure is created successfully. A return value of ‘true’ means that the PUBLISH packet has been created correctly as per the parameters passed by the user application.

## SUBSCRIBE

### MQTT\_CreateSubscribePacket

* + 1. Description

bool MQTT\_CreateSubscribePacket(mqttSubscribePacket \*newSubscribePacket)

MQTT\_CreateSubscribePacket API creates a MQTT subscribe data packet structure, which follows MQTT standard.

* + 1. Parameters

A pointer that points to a MQTT SUBSCRIBE packet structure mqttSubscribePacket.

* + 1. Return Values

A bool type value indicating whether a subscribe data packet structure is created successfully. A return value of ‘true’ means that the SUBSCRIBE packet has been created correctly as per the parameters passed by the user application.

## DISCONNECT

### MQTT\_Disconnect

* + 1. Description

void MQTT\_Disconnect(void)

This API sends a MQTT DISCONNECT packet. This disconnects the client cleanly from the MQTT server.

* + 1. Parameters

None.

* + 1. Return Values

None.

## GET CLIENT CONNECTION INFO

### MQTT\_GetClientConnectionInfo

* + 1. Description

mqttTxRxInformation\* MQTT\_GetClientConnectionInfo(void)

This API returns a pointer to the current MQTT connection’s information, which is essentially a structure with both transmit and receive buffer information.

* + 1. Parameters

None.

* + 1. Return Values

A pointer points to current MQTT connection’s buffer information, which is a structure containing MQTT transmit and receive buffer information.

## GET RECEIVED DATA

### MQTT\_GetReceivedData

* + 1. Description

void MQTT\_GetReceivedData(*uint8\_t* \*pData, *uint8\_t* len)

MQTT\_GetReceivedData API is responsible for receiving packets from the MQTT server and copying the received packets in the reception specific exchange buffer.

* + 1. Parameters

uint8\_t \*pData: the received data buffer pointer.

uint8\_t len: the received data length.

* + 1. Return Values

None.

## GET CONNECTION STATE

### MQTT\_GetConnectionState

* + 1. Description

mqttCurrentState MQTT\_GetConnectionState(void)

MQTT\_GetConnectionState API returns the current MQTT connection state. Possible values include DISCONNECTED, CONNECTING, WAITFORCONNACK, CONNECTED.

* + 1. Parameters

None.

* + 1. Return Values

mqttCurrentState: An enum indicating the current MQTT connection state. Possible valid values: DISCONNECTED, CONNECTING, WAITFORCONNACK, CONNECTED.

## RECEPTION HANDLER

### MQTT\_ReceptionHandler

* + 1. Description

mqttCurrentState MQTT\_ReceptionHandler(mqttTxRxInformation \*mqttConnectionPtr)

MQTT\_ReceptionHandler API handles the received MQTT packet based on the MQTT state and then sets the state to a proper value based on the data received.

* + 1. Parameters

A pointer that points to the current MQTT returns a pointer to the current MQTT connection’s information, which is essentially a structure relevant buffer information.

* + 1. Return Values

mqttCurrentState: An enum indicating the current MQTT connection state. Possible valid values: DISCONNECTED, CONNECTING, WAITFORCONNACK, CONNECTED.

## TRANSMISSION HANDLER

### MQTT\_TransmissionHandler

* + 1. Description

mqttCurrentState MQTT\_TransmissionHandler(mqttTxRxInformation \*mqttConnectionPtr)

MQTT\_TransmissionHandler API sends out an MQTT packet based on the settings of MQTT packet transmission flags and the current MQTT state, then set the current MQTT state to a proper state.

* + 1. Parameters

A pointer that points to the current MQTT returns a pointer to the current MQTT connection’s information, which is essentially a structure relevant buffer information.

* + 1. Return Values

mqttCurrentState: An enum indicating the current MQTT connection state. Possible valid values: DISCONNECTED, CONNECTING, WAITFORCONNACK, CONNECTED.

## SET PUBLISH RECEPTION HANDLER TABLE

### MQTT\_SetPublishReceptionHandlerTable

* + 1. Description

void MQTT\_SetPublishReceptionHandlerTable(publishReceptionHandler\_t \*appPublishReceptionInfo)

MQTT\_SetPublishReceptionHandlerTable is called by the user application to inform the MQTT core of the call back table defined to handle the PUBLISH messages received from the MQTT server.

* + 1. Parameters

A publishReceptionHandler\_t table information defined in the user application, which involves a call back function pointer of a corresponding MQTT topic.

* + 1. Return Values

None.

## GET PUBLISH RECEPTION HANDLER TABLE

### MQTT\_GetPublishReceptionHandlerTable

* + 1. Description

publishReceptionHandler\_t \*MQTT\_GetPublishReceptionHandlerTable();

MQTT\_GetPublishReceptionHandlerTable API returns a publishReceptionHandler\_t table information defined in the user application, which involves a call back function pointer of a corresponding MQTT topic.

* + 1. Parameters

None.

* + 1. Return Values

A publishReceptionHandler\_t table information defined in the user application, which involves a call back function pointer of a corresponding MQTT topic.

## GET CONNECTION AGE

### MQTT\_getConnectionAge

* + 1. Description

int32\_t MQTT\_getConnectionAge(void);

MQTT\_getConnectionAge API calculates the time elapsed since MQTT connection setup.

* + 1. Parameters

None.

* + 1. Return Values

The value indicating the time elapsed since MQTT connection setup.

# Private APIs

## 1. SEND CONNECT

### mqttSendConnect

Description

static bool mqttSendConnect(mqttTxRxInformation \*mqttConnectionPtr);

mqttSendConnect API sends the MQTT CONNECT packet using the underlying TCP layer.

Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

Return Values

Boolean value indicating whether the packet has been successfully sent. The value ‘true’ implies that the packet has been sent successfully to the server.

## SEND PUBLISH

### mqttSendPublish

* + 1. Description

static bool mqttSendPublish(mqttTxRxInformation \*mqttConnectionPtr);

mqttSendPublish API sends the MQTT PUBLISH packet using the underlying TCP layer.

* + 1. Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

* + 1. Return Values

Boolean value indicating whether the packet has been successfully sent. The value ‘true’ implies that the packet has been sent successfully to the server.

## SEND PINGREQ

### mqttSendPingreq

* + 1. Description

static bool mqttSendPingreq(mqttTxRxInformation \*mqttConnectionPtr);

mqttSendPingreq API sends the MQTT PINGREQ packet using the underlying TCP layer.

* + 1. Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

* + 1. Return Values

Boolean value indicating whether the packet has been successfully sent. The value ‘true’ implies that the packet has been sent to the server

## SEND DISCONNECT

### mqttSendDisconnect

* + 1. Description

static bool mqttSendDisconnect(mqttTxRxInformation \*mqttConnectionPtr);

mqttSendDisconnect API sends the MQTT DISCONNECT packet using the underlying TCP layer.

* + 1. Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

* + 1. Return Values

Boolean value indicating whether the packet has been successfully sent. The value ‘true’ implies that the packet has been sent to the server

## PROCESS CONNACK

### mqttProcessConnack

* + 1. Description

static mqttCurrentState mqttProcessConnack(mqttTxRxInformation \*mqttConnectionPtr);

Processes the CONNACK packet received from the broker.

* + 1. Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

* + 1. Return Values

Current state of the MQTT (CONNECTED if correct CONNACK packet is received and DISCONNECTED if certain parameters in the CONNACK packet indicate that the server has not granted a connection).

## PROCESS PINGRESP

### mqttProcessPingresp

* + 1. Description

static mqttCurrentState mqttProcessPingresp(mqttTxRxInformation \*mqttConnectionPtr);

Processes the PINGRESP packet received from the broker.

* + 1. Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

* + 1. Return Values

None.

## PROCESS SUBACK

### mqttProcessSuback

* + 1. Description

static mqttCurrentState mqttProcessSuback(mqttTxRxInformation \*mqttConnectionPtr);

Processes the PINGRESP packet received from the broker.

* + 1. Parameters

Pointer to the MQTT connection structure \*mqttConnectionPtr.

* + 1. Return Values

Current state of the MQTT (CONNECTED if correct SUBACK packet is received correctly and DISCONNECTED if certain parameters in the SUBACK packet indicate that the server has acknowledged the SUBSCRIBE message completely).

## CHECK CONNACK TIMEOUT STATE

### checkConnackTimeoutState

* + 1. Description

static absolutetime\_t checkConnackTimeoutState ();

checkConnackTimeoutState is a call back function that will be called when a timeout (30s) has occurred after sending the CONNECT packet, since a CONNACK packet is expected from the broker within 30s.

* + 1. Parameters

None.

* + 1. Return Values

Number of ticks until the connackTimer expires. In the current implementation it is 0, indicating that the timer function will be executed only once.

## CHECK PINGREQ TIMEOUT STATE

### checkPingreqTimeoutState

* + 1. Description

static absolutetime\_t checkPingreqTimeoutState ();

checkPingreqTimeoutState is a call back function that will be called when a “keep-alive-timeout” defined in user application is near after a MQTT connection has been set up to make sure the connection keeps alive. In the current implementation it is 1 second before “keep-alive-timeout”.

* + 1. Parameters

None.

* + 1. Return Values

Number of ticks until the pingreqTimer expires.

## CHECK PINGRESP TIMEOUT STATE

### checkPingrespTimeoutState

* + 1. Description

static absolutetime\_t checkPingrespTimeoutState ();

checkPingrespTimeoutState is a call back function that will be called when a timeout (30s) has occurred after sending a PINGREQ packet. In the current MQTT client implementation, the client waits for 30s after transmission of PINGREQ packet to receive a PINGRESP packet.

* + 1. Parameters

None.

* + 1. Return Values

Number of ticks until the pingrespTimer expires.

## ENCODING REMAINING LENGTH

### mqttEncodeLength

* + 1. Description

static uint8\_t mqttEncodeLength(uint16\_t length, uint8\_t \*output);

The function encodes the text fields in MQTT packets as UTF-8 strings

* + 1. Parameters

uint16\_t length: the number of bytes remaining within the current packet, including data in the variable header and the payload.

uint8\_t \*output: a pointer points to the encoded bytes.

* + 1. Return Values

The number of bytes encoded.

## DECODING REMAINING LENGTH

### mqttDecodeLength

* + 1. Description

static absolutetime\_t mqttEncodeLength(uint8\_t \*encodedData);

The function decodes UTF-8 encoded string to text fields as per the requirement of MQTT standard.

* + 1. Parameters

uint8\_t \*encodedData: a pointer points to the encoded value of remaining length of a MQTT control packet.

* + 1. Return Values

The decoded value of the remaining length of an MQTT control packet header.

# **Dependent APIs**

## EXCHANGE BUFFER INIT

### ExchangeBufferInit

* + 1. Description

void MQTT\_ExchangeBufferInit(exchangeBuffer \*buffer);

Sets the current location pointer to the beginning of the buffer and sets the length to zero.

* + 1. Parameters

Pointer to Exchange Buffer structure.

* + 1. Return Values

None.

## EXCHANGE BUFFER WRITE

### ExchangeBufferWrite

* + 1. Description

uint16\_t MQTT\_ExchangeBufferWrite(exchangeBuffer \*buffer, uint8\_t \*data, uint16\_t length);

Sets the current location pointer to the beginning of the buffer and sets the length to zero.

* + 1. Parameters

Copies a data buffer to the Exchange Buffer.

* + 1. Return Values

Length.

## EXCHANGE BUFFER READ

### ExchangeBufferRead

* + 1. Description

uint16\_t MQTT\_ExchangeBufferRead(exchangeBuffer \*buffer, uint8\_t \*data, uint16\_t length);

Copies the Exchange buffer to the data buffer. Exchange buffer is reset in the process.

* + 1. Parameters

Copies a data buffer to the Exchange Buffer.

* + 1. Return Values

Number of bytes copied.

## EXCHANGE BUFFER PEEK

### ExchangeBufferPeek

* + 1. Description

uint16\_t MQTT\_ExchangeBufferPeek(exchangeBuffer \*buffer, uint8\_t \*data, uint16\_t length);

Copies data out of the Exchange buffer without modifying the data length or current pointer.

* + 1. Parameters

Pointer to Exchange Buffer structure, Pointer to Data Buffer, dataLength.

* + 1. Return Values

Number of bytes copied.

# References

* <http://mqtt.org/documentation>