Jetter automation

Application-Oriented Manual

MQTT-Client

We automate your success.

This document has been compiled by Jetter AG with due diligence, and based on the known state of the art. Revisions and further development of our products are not automatically mentioned in a reviewed document. Jetter AG shall not be liable for errors in form or content, or for missing updates, as well as for damages or disadvantages resulting from such failure.

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg
Germany

Phone

Switchboard	+49 7141 2550-0
Sales	+49 7141 2550-531
Technical Hotline	+49 7141 2550-444

E-mail

Technical Hotline	hotline@jetter.de
Sales	sales@jetter.de

Translation of the original User Manual

Revision	1.00
Date of issue	3/17/2020

Table of Contents

1 Product description

This document describes the use of a Jetter controller as MQTT client and the STX functions required for this.

2 MQTT

The MQTT protocol is an open message protocol that has established itself as a standard in machine-to-machine communication and the Internet of Things. It is based on the publisher/subscriber architecture and enables clients to provide messages about specific topics via a broker. Messages, such as measured temperature values of a sensor, are published by a client and sent to the broker, which forwards these measured values to all MQTT clients that have subscribed to the corresponding topic.

2.1 Structure of topics

A topic in MQTT is structured hierarchically, whereby the individual levels of the hierarchy are separated by a /:

```
Sensor/COMPUTER_NAME/Temperature/DISK_NAME
```

This topic can be subscribed to by an MQTT client. The placeholders `COMPUTER_NAME` and `DISK_NAME` are replaced accordingly. The placeholders can be uniquely filled so that only measured values from a specific computer or hard disk are received. So-called "wildcards" can also be used.

Wildcard +

The plus can be set as a wildcard at a level in the hierarchy of a topic in order to subscribe to all possible values at this level. The above-mentioned example would allow you to subscribe to the temperature values of all computers and hard disks:

```
Sensor/+/Temperature/+
```

Wildcard #

The hash can be used as a wildcard to subscribe to all values of the remaining hierarchy levels. For this reason, the hash must always be the last character in a subscription. In the example, the hash key could be set instead of "COMPUTER_NAME" to subscribe to all published messages from all computers:

```
Sensor/#
```

2.2 Quality of Service (QoS)

Three different QoS levels can be selected to transmit messages between an MQTT client and an MQTT broker:

Quality of Service	Definition
0	Messages with QoS 0 are sent only once without waiting for the recipient to acknowledge receipt. The sender does not save the message. This can cause messages to be lost on networks with poor connections.
1	Messages with QoS 1 are sent until the recipient has acknowledged receipt. The message is saved by the sender and resent if no acknowledgment is received. This can cause a recipient to receive a message more than once.
2	Messages with QoS 2 are sent exactly once and delivered via a 4-step handshake between client and broker. This is the highest form of transmission security in MQTT networks.

2.3 Retained Messages

In MQTT, messages can be assigned a so-called "retained flag". Messages with this flag set are published by the client as usual to the broker who distributes the message to the other clients. The difference with retained messages is that the broker stores these messages and also sends them to new clients who subscribe to the corresponding topic later. This ensures that all new clients receive the latest status.

2.4 System requirements

2.4.1 Hardware

The MQTT client can be used on the following controllers:

Type	Min. OS	Comment
JC-440EXT	1.8.0.0	-

2.4.2 Software

Operating system

The minimum version of the operating system (min. OS) of your controller can be found in the hardware requirements [▶ 7].

License

To be able to use the MQTT client, a valid license for the operating system function JCF4-C_MQTT must be available on the controller.

Integrated Development Environment

You must have at least JetSym 6.0 to access the MQTT features and the MQTT class.

Programming

The MQTT client is programmed exclusively via the STX user program. Configuration via the Hardware Manager is not required.

3 Programming

3.1 STX Functions

The following STX functions are available in JetSym for programming a Jetter controller as MQTT client:

3.1.1 Function MQTTInitialize()

This function initializes the MQTT functionality. It should be called when the user program is started. When the function is called, the existing connections are terminated.

Declaration

```
Function MQTTInitialize() : int;
End_Function
```

Return values

Value	Description
0	Initialization successful
<0	Error during initialization

3.1.2 Function MQTTCreate()

This function creates a connection between a broker and a client. The connection between client and broker is not yet established. The connection is established using the MQTTConnect() function [▸ 9].

Declaration

```
Function MQTTCreate() : int;
End_Function
```

Return values

Value	Description
>0	Handle for further access to the connection
<0	Error while establishing the connection

3.1.3 Function MQTTDestroy()

This function terminates and clears a previously established connection.

Declaration

```
function MQTTDestroy(handle : int):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function

Return values

Value	Description
0	Connection successfully terminated and cleared
<0	Error while terminating or clearing the connection

3.1.4 Function MQTTSetCredentials()

This feature sets the credentials for connecting to the broker.

Declaration

```
function MQTTSetCredentials(
    handle : int,
    const ref userName,
    const ref passWord,
    const ref clientId
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
userName	User name for the connection
passWord	Password for the connection
clientID	Unique MQTT Client ID

Return values

Value	Description
0	Credentials set successfully
<0	Error when setting the credentials

3.1.5 Function MQTTSetBroker()

This function sets the connection data of the client to an MQTT broker. For this purpose, the IP address and the port number of the broker must be transferred.

ⓘ INFO

The MQTT client only supports TLS encrypted connections. By default, port 8883 is used for this purpose.

Declaration

```
function MQTTSetBroker(
    handle : int,
    brokerAdress,
    port,
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
brokerAdress	IP address of the broker
port	IP port of the broker

Return values

Value	Description
0	Connection information received successfully
<0	Error when retrieving connection information

3.1.6 Function MQTTConnect()

This function establishes a connection to the broker.

Declaration

```
function MQTTConnect(
    handle : int,
    timeout_msec,
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
timeout_msec	Timeout time in milliseconds, e.g. 1000 for 1 second

Return values

Value	Description
0	Successfully connected to broker
-8	Error: Timeout

3.1.7 Function MQTTDisconnect()

This function disconnects the connection to the broker.

Declaration

```
function MQTTDisconnect(
    handle : int,
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function

Return values

Value	Description
0	The connection to the broker was successfully disconnected
<0	Error: Socket Error
-8	Error: Timeout

3.1.8 Function MQTTPublish()

This function publishes messages about a topic and sends them to the broker.

Declaration

```
function MQTTPublish(
    handle : int,
    const ref topic,
    const ref message,
    messageLength : int,
    retain : int,
    qos : int
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
topic	MQTT topic e.g. '/a/hello/c/'
message	MQTT message e.g. 'message2'
messageLength	Length of the message e.g. StrLength(_MqttMessageSend_2)
retain	Retained Messages [▶ 7]
qos	Quality of Service (QoS) [▶ 6]

Return values

Value	Description
0	The message was successfully published
<0	Error: Socket Error

3.1.9 Function MQTTSubscribe()

This function subscribes to a topic from the MQTT broker.

Declaration

```
function MQTTSubscribe(
    handle : int,
    const ref topic,
    qos : int
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
topic	MQTT topic e.g. '/a/hello/c/'
qos	See also: Quality of Service (QoS) [▶ 6]

Return values

Value	Description
0	The topic was successfully subscribed
<0	Error: Socket Error

3.1.10 Function MQTTUnsubscribe()

This function cancels the subscription of a topic.

Declaration

```
function MQTTUnsubscribe(
    handle : int,
    const ref topic,
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
topic	MQTT topic e.g. '/a/hello/c/'

Return values

Value	Description
0	Subscription to the topic successfully cancelled
<0	Error: Socket Error

3.1.11 Function MQTTGetMessage()

This function receives messages sent from the broker to the client.

Declaration

```
function MQTTGetMessage(
    handle : int,
    ref msg : string,
    messageLength : int,
    timeout_msec : int
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
msg	Content of the incoming message
messageLength	Length of the receive buffer in STX. Message may be truncated if the STX buffer is too small
timeout_msec	Must be set to 0. The timeout value of MQTTConnect() is used.

Return values

Value	Description
>0	A positive value indicates that a message or parts of a message have been received.
<0	A negative value indicates an error: ■ -10007: SMQTTE_CONREFUSED ■ -10000: SMQTTE_ERRORBASE ■ -10005: SMQTTE_OVERFLOW ■ -1: SMQTTE_SOCKET

3.1.12 Function MQTTGetTopic()

This function returns the name of the topic for which a message was received (if the return value of the MQTTGetMessage() function is greater than 0).

Declaration

```
function MQTTGetTopic(
    handle : int,
    ref topic : string,
    topicLength : int
    ):int;
end_function
```

Parameter

Parameter	Description
Handle	Return value of the MQTTCreate() function
topic	Name of the topic for which a message was received
topicLength	Length of the receive buffer in STX. Message may be truncated if the STX buffer is too small

Return values

Value	Description
0	Topic name successfully received
<0	An error has occurred

3.2 Class _MqttClient

The _MqttClient class defines further variables and functions which are used to set up and use the MQTT client and combine some STX functions in a meaningful way. The class is integrated as a platform file in JetSym and is automatically available to you after creating a project with a controller of the type JC-440EXT.

3.2.1 Structure _MqttClientParams

The `_MqttClientParams` structure contains all parameters required to establish a connection between client and broker.

Declaration

```
type
    _MqttClientParams : struct
        BrokerAddress : string;
        BrokerPort : int;
        Username : string;
        Password : string;
        ClientId : string;
        LastWillTopic : string;
        LastWillMessage : string;
    end_struct;
end_type;
```

Parameter

Parameter	Value
BrokerAdress	IP address or name of MQTT broker
BrokerPort	Port number of the MQTT broker (usually 1883 or 8883)
Username	User name for the account at the MQTT broker
Password	Password for the account at the MQTT broker
ClientId	Unique client ID used to identify the client at the broker
LastWillTopic	Topic for the LastWillMessage
LastWillMessage	Message sent by the broker to clients when the connection is lost

ⓘ INFO

The MQTT client only supports TLS encrypted connections. By default, port 8883 is used for this purpose.

3.2.2 Variables

The following variables are defined in the _MQTTClient class:

variable	Type	Features
hMqtt	Int	Handle that identifies the client
bConnected	Boolean	Indicates whether a connection is active
bReceived	Boolean	Indicates whether a message has been received from the broker
LastMqttError	Int	Saves the last error message

3.2.3 Function MqttClient()

This function is used to initialize the client variables.

Declaration

```
function _MqttClient._MqttClient();
    hMqtt := 0;
    bConnected := false;
    bReceived := false;
    LastMqttError := 0;
end_function;
```

Parameter

Parameter	Description
hMqtt	Value 0 is overwritten by the MQTTCreate function. Serves to identify the client at the broker.
bConnected	Is `true` if the connection to a broker is active
bReceived	Is `true` if it was detected that a message from the broker has been received
LastMqttError	Is always overwritten by the value of the last detected error message

3.2.4 Function InitMqttClient()

This function creates and initiates the MQTT client.

Declaration

```
function InitMqttClient(ClientParams : _MqttClientParams) :
int;
```

Sample program

In this function definition, the MQTT client is created, the credentials for access to the MQTT broker are defined, and a LastWillMessage is set if one is defined:

```
function _MqttClient.InitMqttClient(ClientParams : _MqttClient-
Params) : int;
    // Creates the MQTT client:
    hMqtt := MQTTCreate();
    if (hMqtt > 0) then
    MQTTSetCredentials(
        hMqtt,
        ClientParams.Username,
        ClientParams.Password,
        ClientParams.ClientId
    );
    MQTTSetBroker(
        hMqtt,
        ClientParams.BrokerAdress
        ClientParams.BrokerPort
    );
        if (StrLength(ClientParams.LastWillTopic) > 0) then
        MQTTLastWIllMessage(
            hMqtt,
            ClientParams.LastWillTopic,
            StrLength(ClientParams.LastWillMessage),
            MQTT_RETAIN_OFF,
            MQTT_QOS
        );
        end_if;
    InitMqttClient := 0;
    else
    InitMqttClient := -1;
    end_if;
end_function;
```

Parameter

Parameter	Description
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.
ClientParams.Username	See Structure _MqttClientParams [▶ 12]
ClientParams.Password	
ClientParams.ClientId	
ClientParams.BrokerAdress	
ClientParams.BrokerPort	
ClientParams.LastWillTopic	
ClientParams.LastWillMessage	
MQTT_RETAIN_OFF	Retained Messages [▶ 7]
MQTT_QOS	Quality of Service (QoS) [▶ 6]

3.2.5 Function Connect()

This function establishes the connection to an MQTT broker.

Declaration

```
function Connect(Timeout : int) : int;
```

Application

In this function definition, the connection to an MQTT broker is established and a timeout for establishing the connection is defined. If the connection has been established without error, the variable bConnected is set to `True` . If an error occurred, an error message is sent:

```
function _MqttClient.Connect(
    Timeout : int
    ) : int;
    if (hMqtt > 0) then
        Connect := MQTTConnect(hMqtt, Timeout);
        if (Connect == 0) then
        bConnected := true;
        else
            trace('connect failed with code'
            + intToStr(Connect) + '$n.');
        end_if;
    else
        Connect := -1;
    end_if;
end_function;
```

Parameter

Parameter	Description
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.
bConnected	Is `true` if the connection to a broker is active
Timeout	After the timeout period has elapsed, the connection process is terminated and an error message is sent.

3.2.6 Function IsConnected()

This function indicates whether a connection to a broker is active.

Declaration

```
function IsConnected() : bool;
```

Application

In this function definition, the value of the bConnected variable is set as the result of the function:

```
function _MqttClient.IsConnected() : bool;
    IsConnected := bConnected;
end_function
```

Parameter

Parameter	Description
bConnected	Is `true` if the connection to a broker is active

3.2.7 Function Subscribe()

This function subscribes to a topic from the broker.

Declaration

```
function Subscribe(Topic : string) : int;
```

Application

In this function definition, a topic is subscribed to by specifying the topic name and the desired QoS level, provided there is a connection to the broker:

```
function _MqttClient.Subscribe(Topic : string) : int;
    if (bConnected) then
        Subscribe := MQTTSubscribe(
            hMqtt,
            Topic,
            MQTT_QOS);
    else
        Subscribe := -1;
    end_if;
end_function;
```

Parameter

Parameter	Description
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.
Topic	MQTT topic e.g. 'a/hello/c
MQTT_QOS	Quality of Service (QoS) [▶ 6]
bConnected	Is `true` if the connection to a broker is active

3.2.8 Function ProcessSubscription()

This function processes the incoming messages at the client.

Declaration

```
function ProcessSubscription(
    ref SubMessage : string,
    ref Topic : string,
    Timeout : int);
```

Application

In this function definition, the MQTTGetMessage function is executed until it is successful or the defined timeout has elapsed. The MQTTGetTopic function is then executed. If an error occurs when the function is executed, a corresponding trace message is sent:

```
function _MqttClient.ProcessSubscriptions(
    ref SubMessage : string,
    ref Topic : string,
    Timeout : int)
var
    Result : int;
    TimeoutTimer : Timer;
end_var;
    if (Timeout > 0) then
        TimerStart(TimeoutTimer, Timeout);
        repeat
            Result := MQTTGetMessage(
                hMqtt,
                SubMessage,
                sizeof(SubMessage),
                0);
            delay(t#2ms);
        until
            ((Result <> SMQTT_TIMEOUT)
        or
            (TimerEnd(TimeoutTimer)))
        end_repeat;
    else
        Result := MQTTGetMessage(
                hMqtt,
                SubMessage,
                sizeof(SubMessage),
                -1);
    end_if;
    if (Result > 0) then
        Result := MQTTGetTopic(
                hMqtt,
```

```
                Topic,
                sizeof(Topic)
                );
        if (Result == 0) then
        else
            Topic := '';
        end_if;
        ProcessSubscriptions := 0;
    else
        case Result of
    SMQTT_TIMEOUT : // Error handling of a timeout
    trace('GetMessage has timed out.$n');
    break;
    SMQTT_SUBACK : // Handling of suback
    trace(
    'Asynchronious subscribe response message received.$n');
    break;
    SMQTT_UNSUBACK : // Handling of unsuback
    trace(
    'Asynchronious unsubscribe response message received.$n');
    break;
    SMQTTE_TIMEOUT : // Handling of an unexpected timeout
    trace('Unexpected timeout!$n');
    LastMqttError := SMQTTE_TIMEOUT;
    break;
    SMQTTE_PONGTIMEOUT : // Handling of pong timeout
    trace(
    'Server did not respond to ping within timeout time.$n');
    LastMqttError := SMQTTE_PONGTIMEOUT;
    break;
    SMQTTE_PROTOCOL_ERROR : // Handling of protocol error
    trace('A protocol error occured.$n');
    LastMqttError := SMQTTE_PROTOCOL_ERROR;
    break;
    SMQTTE_OVERFLOW : // Handling of buffer overflow
    trace(
    'Buffer overflow. Output buffer must be larger!$n');
    LastMqttError := SMQTTE_OVERFLOW;
    break;
    else
    trace(
    'GetMessage failed. Error code:' + intToStr(Result) + '!$n');
    LastMqttError := Result;
    end_case;
    ProcessSubscriptions := -1;
    end_if;
end_function;
```

Parameter

Parameter	Description
SubMessage	Content of the published message
Topic	MQTT topic e.g. 'a/hello/c
Timeout	Timeout for receiving new messages
TimeoutTimer	Timer for the MQTTGetMessage function
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.
SMQTT_TIMEOUT	Timeout of the MQTTGetMessage function

3.2.9 Function GetErrorEvent()

This function returns the last stored error that occurred in the communication between the MQTT client and the MQTT broker.

Declaration

```
function GetErrorEvent() : int;
```

Application

This function definition checks if the value of the variable `LastMqttError` is less than 0. If yes, the value is written as the result of the function:

```
function _MqttClient.GetErrorEvent() : int;
    When (LastMqttError < 0) continue;
    GetErrorEvent := LastMqttError;
end_function;
```

Parameter

Parameter	Description
LastMqttError	Contains the value of the last error message

3.2.10 Function Unsubscribe()

This function cancels the subscription of a topic. No more messages of this topic will be received.

Declaration

```
function Unsubscribe(topic : string) : int;
```

Application

In this function definition, the subscription of a topic is canceled if there is a connection to the broker:

```
function _MqttClient.Unsubscribe(Topic : string) : int;
    if (bConnected) then
        MQTTUnsubscribe(hMqtt, Topic);
    end_if;
end_function;
```

Parameter

Parameter	Description
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.
bConnected	Is `true` if the connection to a broker is active
Topic	MQTT topic e.g. 'a/hello/c

3.2.11 Function Publish()

This function publishes news about a topic to the MQTT broker.

Declaration

```
function Publish(
    Topic : string,
    Message : string,
    RetainValue : int) : int;
```

Application

In this function definition, the MQTTPublish function is executed if there is a connection to the broker. Otherwise, a corresponding error message is written:

```
function _MqttClient.Publish(Topic : string, Message : string,
RetainValue : int) : int;
    if (bConnected) then
        Publish := MQTTPublish(
            hMqtt,
            Topic,
            Message,
            StrLength(Message),
            0,
            MQTT_QOS);
    else
        trace('Publish failed because client is not connected.');
```

```
        Publish := -1;
    end_if;
end_function;
```

Parameter

Parameter	Description
Topic	MQTT topic e.g. 'a/hello/c
Message	Content of the message that is published by the client
RetainValue	Retained Messages [▶ 7]
bConnected	Is `true` if the connection to a broker is active
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.
MQTT_QOS	Quality of Service (QoS) [▶ 6]

3.2.12 Function Close()

This function disconnects the connection between the MQTT client and an MQTT broker.

Declaration

```
function Close() : int;
```

Application

This function definition checks whether there is a connection between the MQTT client and the MQTT broker. If so, it will be disconnected:

```
function _MqttClient.Close()
    if (hMqtt > 0) then
        MQTTDisconnect(hMqtt);
    end_if;
end_function;
```

Parameter

Parameter	Description
hMqtt	Result of the MQTTCreate function. Identifies the client at the broker.

3.3 MQTT error codes

The following error messages may occur when using the _MQTTClient class:

Error code	Value	Description
SMQTT_TIMEOUT	-20000	Timeout of the function `GetMessage`
SMQTT_SUBACK	-20001	Asynchronous input of a subscribe response via `GetMessage`
SMQTT_UNSUBACK	-20002	Asynchronous input of an unsubscribe response via `GetMessage`
SMQTTE_TIMEOUT	-10001	Unexpected timeout
SMQTTE_PONGTIMEOUT	-10002	The client has sent a PING to the broker but has not received a PONG
SMQTTE_PROTOCOL_ERROR	-10003	An error has occurred
SMQTTE_OVERFLOW	-10005	The output buffer is not large enough to perform the planned action

List of tables

Glossary

Index

Jetter automation

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg
www.jetter.de

E-mail info@jetter.de
Phone +49 7141 2550-0

We automate your success.