

ADVANTECH



MQTT Broker




© 2024 Advantech Czech s.r.o. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photography, recording, or any information storage and retrieval system without written consent. Information in this manual is subject to change without notice, and it does not represent a commitment on the part of Advantech.

Advantech Czech s.r.o. shall not be liable for incidental or consequential damages resulting from the furnishing, performance, or use of this manual.

All brand names used in this manual are the registered trademarks of their respective owners. The use of trademarks or other designations in this publication is for reference purposes only and does not constitute an endorsement by the trademark holder.

Used symbols

 *Danger* – Information regarding user safety or potential damage to the router.

 *Attention* – Problems that can arise in specific situations.

 *Information* – Useful tips or information of special interest.

 *Example* – Example of function, command or script.

Contents

1. Changelog	1
1.1 MQTT Broker Changelog	1
2. Basic Information	2
2.1 What is MQTT and MQTT Broker	2
2.2 What does MQTT Broker do?	2
3. Router App Description	3
3.1 Web Interface	3
3.2 Configuration	3
3.3 Security	4
3.3.1 Encryption	5
3.3.2 Authentication	6
3.4 Command Line Tools	7
4. Related Documents	9

List of Figures

1 Router Apps	3
2 Global	3
3 MQTT Broker Configuration used to test Example 1	7
4 MQTT Broker Configuration used to test Example 2	8

List of Tables

1 Configuration items description	4
---	---

1. Changelog

1.1 MQTT Broker Changelog

v2.0.8 (2021-04-28)

- First release of Mosquitto MQTT Broker

v2.0.10 (2021-05-28)

- Updated Mosquitto to 2.0.10
- Added TLS support to GUI

v2.0.12 (2021-09-30)

- Updated Mosquitto to 2.0.12

v2.1.0 (2022-11-03)

- Reworked license information

v2.1.1 (2023-02-28)

- Linked statically with OpenSSL 1.1.1t

v2.0.18 (2024-01-05)

- Updated Mosquitto to 2.0.18
- Added mosquitto_pub and mosquitto_sub command-line tools
- Linked statically with OpenSSL 3.0.12
- Added description and summary files
- Recompiled with ModulesSDK 2.1.0

v2.0.18-1 (2024-01-31)

- Fixed misbehaviour of recent browsers ignoring attribute autocomplete="off" in password input fields

2. Basic Information



Router app *MQTT Broker* is not contained in the standard router firmware. Uploading of this router app is described in the Configuration manual (see Chapter [Related Documents](#)).

2.1 What is MQTT and MQTT Broker

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for efficient communication between devices in IoT (Internet of Things) and mobile environments. It enables devices to publish messages to topics and subscribe to receive messages on those topics, facilitating real-time, reliable, and scalable data exchange with minimal bandwidth and power consumption requirements.

MQTT Broker is a server or middleware that acts as a central hub for facilitating communication between devices and applications in an MQTT-based Internet of Things (IoT) architecture.

2.2 What does MQTT Broker do?

1. **Message Routing:** It routes messages between publishers (devices or applications that send data) and subscribers (devices or applications that receive data). When a device publishes a message to a specific topic, the MQTT Broker ensures that the message is delivered to all subscribers interested in that topic.
2. **Protocol Translation:** It handles the MQTT protocol, allowing devices and applications using MQTT to communicate with each other regardless of the underlying network protocols they use. This enables interoperability among various IoT devices and applications.
3. **QoS (Quality of Service) Management:** MQTT supports different levels of QoS for message delivery—QoS 0 (At most once), QoS 1 (At least once), and QoS 2 (Exactly once). The MQTT Broker manages the delivery of messages based on the QoS level specified by publishers and subscribers, ensuring reliable and efficient message delivery according to the desired level of reliability.
4. **Session Management:** It manages client sessions, ensuring that clients remain connected and handling client reconnects seamlessly. This is essential for maintaining continuous communication between devices and applications, especially in unreliable network conditions.
5. **Security:** MQTT Brokers often provide security features such as authentication, access control, and encryption to protect the data exchanged between devices and applications. This ensures that only authorized clients can publish or subscribe to specific topics and that data remains confidential and secure during transmission.

Overall, an MQTT Broker plays a crucial role in enabling scalable, efficient, and secure communication in IoT ecosystems by facilitating the exchange of data between connected devices and applications.

3. Router App Description

3.1 Web Interface

After Router App installation, the module's GUI can be invoked by clicking the router app name on the Router Apps page of router's web interface.

Left part of this GUI contains menu with Configuration section containing item *Global* and Administration section with *Return* item. All the configuration of the router app is found under the *Global* menu item and the Return item switches back from the router apps web page to the routers web configuration pages. The main menu of router app GUI is shown on Figure below.



Figure 1: Router Apps

3.2 Configuration

By clicking the *Global* menu item the router apps configuration is displayed.

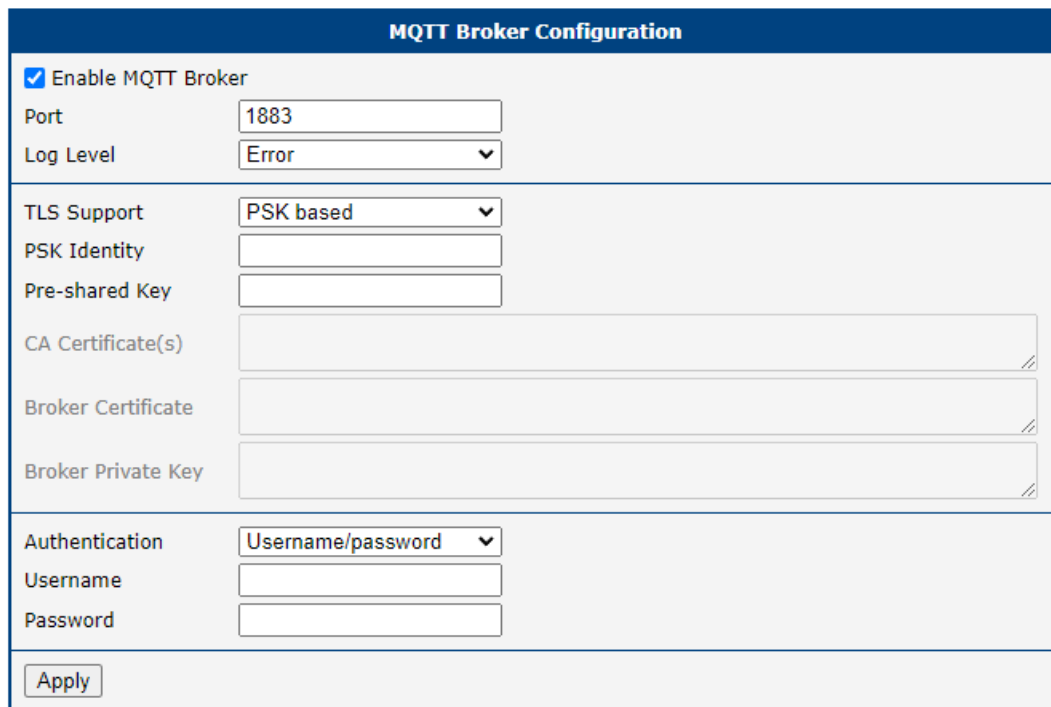
A screenshot of a web configuration page titled "MQTT Broker Configuration". At the top, there is a checked checkbox labeled "Enable MQTT Broker". Below this are several configuration fields: "Port" with a text input containing "1883", "Log Level" with a dropdown menu set to "Error", "TLS Support" with a dropdown menu set to "PSK based", "PSK Identity" with a text input, "Pre-shared Key" with a text input, "CA Certificate(s)", "Broker Certificate", and "Broker Private Key", each with a large text area and a small icon in the bottom right corner. At the bottom, there is an "Authentication" dropdown menu set to "Username/password", followed by "Username" and "Password" text input fields. An "Apply" button is located at the very bottom left of the form.

Figure 2: Global

Item	Description
Enable MQTT Broker	Enables MQTT Broker functionality.
Port	Enter the port where MQTT Broker will listen
Log level	Select level of log information
TLS Support	Options for TLS Support are: <ul style="list-style-type: none"> • None • PSK based • Certificate based
PSK Identity	When selected PSK based option in the TLS Support above, fill the PSK Identity
Pre-shared Key	When selected PSK based option in the TLS Support above, fill the Pre-shared Key as hexa string
CA Certificate(s)	When selected Certificate based option in the TLS Support above, fill the CA Certificate(s) in PEM format
Broker Certificate	When selected Certificate based option in the TLS Support above, fill the Broker Certificate in PEM format
Broker Private Key	When selected Certificate based option in the TLS Support above, fill the Broker Private Key in PEM format
Authentication	Options for Authentication are: <ul style="list-style-type: none"> • None • Username/Password • Certificate CN
Username	Fill in the Username. The Username field is used not only when Authentication is set to Username/Password but also for the Certificate CN (Common Name). In such a case, the client authenticates itself not with a password but with a certificate, where the Common Name (CN) must match the entered information here.
Password	When selected Username/Password option in the Authentication field above, fill the Password

Table 1: Configuration items description

3.3 Security

The MQTT protocol supports encryption of transmitted data and client authentication.

3.3.1 Encryption

Encryption of transmission is not performed at the MQTT protocol level, but rather at the TCP/IP level using TLS. Without encryption enabled, all transmitted data is visible in plain text, including passwords! For encryption, you have two options: *PSK based* and *Certificate based*.

PSK based means encryption using Pre-Shared Key. It involves symmetric encryption using a shared secret, which you should securely exchange between both ends (broker and client). A hexadecimal string is used as the key. You can generate it using a tool like OpenSSL:

```
openssl rand -hex 32
```

This generates a 32-byte/256-bit long key, which you fill in the *Pre-shared key* field in the RouterApp settings and also enter it into the client configuration*. In the *PSK Identity* and client configuration*, you must also enter any chosen string to identify the client to use this shared key. This is because different clients may use different keys. However, in the RouterApp GUI, only one pair of *Client id/PSK* can be set.

Certificate based means encryption using asymmetric keys. You can use commercial certificates (whether paid or free, such as from Let's Encrypt) or generate your own self-signed certificates. If you choose self-signed certificates, you can again use OpenSSL. Here is an example procedure:

1. Generate a key pair for the Certificate Authority:

```
openssl genpkey -algorithm RSA -out ca.key
```

2. Use the key from the first step to create the CA certificate:

```
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
```

3. Generate a key pair for the Broker:

```
openssl genpkey -algorithm RSA -out broker.key
```

4. Create a certificate request for the Broker:

```
openssl req -new -out broker.csr -key broker.key -subj "/CN=router.example.com"
```

(replace *router.example.com* with a real host name of the router, see note about CN bellow)

5. Sign the request using CA certificates from step 2:

```
openssl x509 -req -in broker.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out broker.crt -days 360
```

6. Copy the contents of files ca.crt, broker.crt, and broker.key into the *CA Certificate(s)*, *Broker Certificate*, and *Broker Private Key* fields in the RouterApp settings.
7. Use the file ca.crt in the client* configuration.



You can use different ciphers and parameters during generation. However, it's important that the broker's certificate is not password protected; otherwise, it cannot be used. The CA certificate can be password protected.



The CN (Common Name) of the broker's certificate should be the domain name of the router where the broker is running. If this is not met, clients may refuse connections unless you suppress the match control (such as with the `--insecure` switch in tools like `mosquitto_pub` and `mosquitto_sub`).

3.3.2 Authentication

The client can authenticate using either a username/password or a certificate.

For the *Username/Password* option, simply enter the username and password into the *Username* and *Password* fields in the RouterApp settings, and the same values into the client* settings. Although the broker supports different combinations of username/password for different clients, only one combination can be set for all clients together in the RouterApp GUI.

The *Certificate CN* option can be used only if TLS certificate encryption is enabled. The client certificate must then be signed by the CA certificate used for encryption. Here is the procedure using OpenSSL:

1. Generate the client's private key:

```
openssl genpkey -algorithm RSA -out client.key
```

2. Generate a certificate request with the key from the previous step:

```
openssl req -new -out client.csr -key client.key -subj "/CN=common_name"
```

(replace *common_name* with any name you like)

3. Sign the request using the CA certificate (see section 3.3.1):

```
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out client.  
crt -days 360
```

4. Enter the Common Name filled in step 2 into the *Username* field in the RouterApp settings.
5. Use the files *ca.crt*, *client.crt*, and *client.key* for client* settings.

*Client configuration is not covered in this guide; consult your MQTT client documentation.

3.4 Command Line Tools

The MQTT Broker Router App is built on the Open Source project Mosquitto. After installing the Router App, you can also use two command-line tools:

mosquitto_pub - used to publish a message with a specified topic

Example: Publishing the message "54" with the topic "SENSOR/ROOM8/TEMPERATURE" to a broker running on a router with IPv6 address fc00:202::254. The broker has encryption enabled using a PSK key with the identity "test". Authentication is done with the username "house" and password "12345":

```
mosquitto_pub --host fc00:202::254 --port 1883 --psk
b7e87d9b2074d5889a7969985d9fc3f04e4e5b8cb57956a812b2c52b6411abd8 --psk-identity test -u
house -P 12345 -t SENSOR/ROOM8/TEMPERATURE -m 54
```

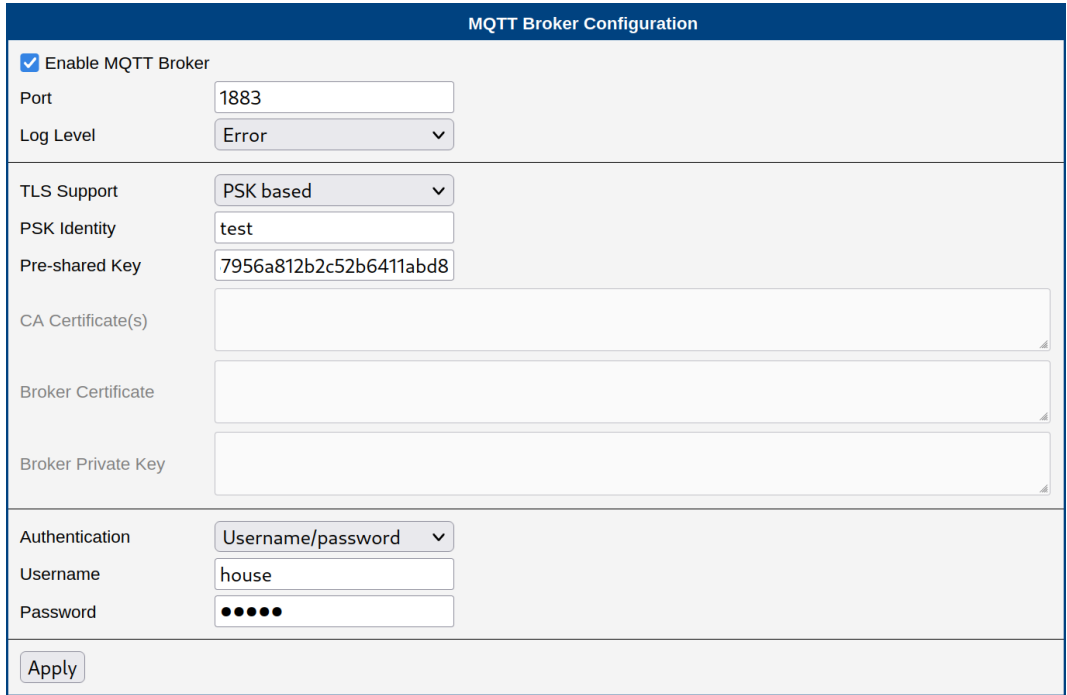
MQTT Broker Configuration	
<input checked="" type="checkbox"/> Enable MQTT Broker	
Port	1883
Log Level	Error
TLS Support	Certificate based
PSK Identity	
Pre-shared Key	
CA Certificate(s)	-----BEGIN CERTIFICATE----- MIIDYTCCAkmGAWIBAgIUDDLXV3S1tvLhDYJip/Ru5g5u0wDQYJKoZIhvcNAQEL
Broker Certificate	-----BEGIN CERTIFICATE----- MIIDBzCCAe8CFDi0jNeP0IimNeVdFNAPRnykuHFMA0GCSqGSIb3DQEBCwUAMEAx
Broker Private Key	-----BEGIN PRIVATE KEY----- MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQQDITCHSZASqSv2A
Authentication	Certificate CN
Username	Data collector
Password	
<input type="button" value="Apply"/>	

Figure 3: MQTT Broker Configuration used to test Example 1

mosquitto_sub - subscribes to the specified topic and prints messages

Example: Reading exactly one message with any topic starting with "factory/sensor" from a broker running on a router with hostname router. The broker has encryption enabled using a certificate (parameter --cafile). The client authenticates using a certificate (parameters --cert and --key):

```
mosquitto_sub -C 1 --host router --port 1883 --cafile ca.crt --cert client.crt --key client.key -t factory/sensor/#
```



The screenshot shows the 'MQTT Broker Configuration' web interface. It features a blue header with the title 'MQTT Broker Configuration'. Below the header, there are several sections of configuration options:

- Enable MQTT Broker:** A checked checkbox.
- Port:** A text input field containing '1883'.
- Log Level:** A dropdown menu set to 'Error'.
- TLS Support:** A dropdown menu set to 'PSK based'.
- PSK Identity:** A text input field containing 'test'.
- Pre-shared Key:** A text input field containing '7956a812b2c52b6411abd8'.
- CA Certificate(s):** A large empty text area.
- Broker Certificate:** A large empty text area.
- Broker Private Key:** A large empty text area.
- Authentication:** A dropdown menu set to 'Username/password'.
- Username:** A text input field containing 'house'.
- Password:** A text input field with five black dots representing a masked password.

At the bottom left of the form is an 'Apply' button.

Figure 4: MQTT Broker Configuration used to test Example 2

Users can use these tools for debugging MQTT issues and scripting.

4. Related Documents

- [1] MQTT Specifications Version 5.0: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>
- [2] MQTT Specifications Version 3.1.1: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
- [3] MQTT Broker Mosquitto: <https://mosquitto.org/>

You can obtain product-related documents on *Engineering Portal* at icr.advantech.cz address.

To get your router's *Quick Start Guide*, *User Manual*, *Configuration Manual*, or *Firmware* go to the *Router Models* page, find the required model, and switch to the *Manuals* or *Firmware* tab, respectively.

The *Router Apps* installation packages and manuals are available on the *Router Apps* page.

For the *Development Documents*, go to the *DevZone* page.