

Configuration



- [Server name](#)
 - [Authentication/Authorization of MQTT Users](#)
 - [Operator mode](#)
 - [Local mode](#)
 - [Special permissions](#)
-

To enable MQTT support in the server, add an `mqtt` configuration block in the server's configuration file like the following:

```
mqtt {  
    # Specify a host and port to listen for websocket connections  
    #  
    # listen: "host:port"  
    # It can also be configured with individual parameters,  
    # namely host and port.  
    #  
    # host: "hostname"  
    port: 1883  
  
    # TLS configuration.  
    #  
    tls {  
        cert_file: "/path/to/cert.pem"  
        key_file: "/path/to/key.pem"  
  
        # Root CA file  
        #  
        # ca_file: "/path/to/ca.pem"  
  
        # If true, require and verify client certificates.  
        #
```

```

# verify: true

# TLS handshake timeout in fractional seconds.
#
# timeout: 2.0

# If true, require and verify client certificates and map certificate
# values for authentication purposes.
#
# verify_and_map: true
}

# If no user name is provided when an MQTT client connects, will default
# this user name in the authentication phase. If specified, this will
# override, for MQTT clients, any `no_auth_user` value defined in the
# main configuration file.
# Note that this is not compatible with running the server in operator mode.
#
# no_auth_user: "my_username_for_apps_not_providing_credentials"

# See below to know what is the normal way of limiting MQTT clients
# to specific users.
# If there are no users specified in the configuration, this simple authorization
# block allows you to override the values that would be configured in the
# equivalent block in the main section.
#
# authorization {
#     # If this is specified, the client has to provide the same username
#     # and password to be able to connect.
#     # username: "my_user_name"
#     # password: "my_password"
#
#     # If this is specified, the password field in the CONNECT packet has to
#     # match this token.
#     # token: "my_token"
#
#     # This overrides the main's authorization timeout. For consistency
#     # with the main's authorization configuration block, this is expressed
#     # as a number of seconds.

```

```

#      # timeout: 2.0
#}

# This is the amount of time after which a QoS 1 message sent to
# a client is redelivered as a DUPLICATE if the server has not
# received the PUBACK packet on the original Packet Identifier.
# The value has to be positive.
# Zero will cause the server to use the default value (30 seconds).
# Note that changes to this option is applied only to new MQTT subscriptions.
#
# Expressed as a time duration, with "s", "m", "h" indicating seconds,
# minutes and hours respectively. For instance "10s" for 10 seconds,
# "1m" for 1 minute, etc...
#
# ack_wait: "1m"

# This is the amount of QoS 1 messages the server can send to
# a subscription without receiving any PUBACK for those messages.
# The valid range is [0..65535].
#
# The total of subscriptions' max_ack_pending on a given session cannot
# exceed 65535. Attempting to create a subscription that would bring
# the total above the limit would result in the server returning 0x80
# in the SUBACK for this subscription.
# Due to how the NATS Server handles the MQTT "#" wildcard, each
# subscription ending with "#" will use 2 times the max_ack_pending value.
# Note that changes to this option is applied only to new subscriptions.
#
# max_ack_pending: 100
}

```

Server name

MQTT requires a server name to be set. Server names must be unique in a cluster or super-cluster topology. The server name is set in the top-level section of the server configuration. Here is an example:

```
server_name: "my_mqtt_server"
mqtt {
  port: 1883
  ...
}
```

Authentication/Authorization of MQTT Users

Operator mode

In operator mode, all users need to provide a JWT in order to connect. In the standard authentication procedure of this mode, NATS clients are required to sign a `nonce` sent by the server using their private key (see [JWTs and Privacy](#)). MQTT clients cannot do that, therefore, the JWT is used for authentication, removing the need of the seed. It means that you need to pass the JWT token as the MQTT password and use any username (except empty, since MQTT protocol requires a username to be set if a password is set). The JWT has to have the `Bearer` boolean set to true, which can be done with nsc:

```
nsc edit user --name U --account A --bearer
```

Local mode

A new field when configuring users allows you to restrict which type of connections are allowed for a specific user.

Consider this configuration:

```
authorization {
  users [
    {user: foo password: foopwd, permission: {...}}
    {user: bar password: barpwd, permission: {...}}
  ]
}
```

If an MQTT client were to connect and use the username `foo` and password `foopwd`, it would be accepted. Now suppose that you would want an MQTT client to only be accepted if it connected using the username `bar` and password `barpwd`, then you would use the option `allowed_connection_types` to restrict which type of connections can bind to this user.

```
authorization {
  users [
    {user: foo password: foopwd, permission: {...}}
    {user: bar password: barpwd, permission: {...}, allowed_connection_types: ["MQTT"]}
  ]
}
```

The option `allowed_connection_types` (also can be named `connection_types` or `clients`) as you can see is a list, and you can allow several types of clients. Suppose you want the user `bar` to accept both standard NATS clients and MQTT clients, you would configure the user like this:

```
authorization {
  users [
    {user: foo password: foopwd, permission: {...}}
    {user: bar password: barpwd, permission: {...}, allowed_connection_types: ["STANDARD",
"MQTT"]}
  ]
}
```

The absence of `allowed_connection_types` means that all types of connections are allowed (the default behavior).

The possible values are currently:

- `STANDARD`
- `WEBSOCKET`
- `LEAFNODE`
- `MQTT`

Special permissions

When an MQTT client creates a QoS 1 subscription, this translates to the creation of a JetStream durable subscription. To receive messages for this durable, the NATS Server creates a subscription with a subject such as `$MQTT.sub.` and sets it as the JetStream durable's delivery subject.

Therefore, if you have set some permissions for the MQTT user, you need to allow subscribe permissions on `$MQTT.sub.>`.

Here is an example of a basic configuration that sets some permissions to a user named "mqtt". As you can see, the subscribe permission `$MQTT.sub.>` is added to allow this client to create QoS 1 subscriptions.

```
listen: 127.0.0.1:4222
jetstream: enabled
authorization {
    mqtt_perms = {
        publish = ["baz"]
        subscribe = ["foo", "bar", "$MQTT.sub.>"]
    }
    users = [
        {user: mqtt, password: pass, permissions: $mqtt_perms, allowed_connection_types:
["MQTT"]}
    ]
}
mqtt {
    listen: 127.0.0.1:1883
}
```

📄 #2

📄 cocobook-sync 📄 2026-10-28 11:14:22 UTC

📄 cocobook-sync 📄 2026-10-28 11:23:52 UTC