Mosquitto MQTT Cheat Sheet

Installation LINUX

```
apt update
apt-cache search mosquitto
apt install mosquitto
```

And Mosquitto was successfully installed and is listening on the default port `1883`:

```
netstat -tlnp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      464/sshd: /usr/sbin
tcp        0      0 127.0.0.1:1883          0.0.0.0:*
LISTEN      1692/mosquitto
tcp6       0      0 :::22                   :::*
LISTEN      464/sshd: /usr/sbin
tcp6       0      0 ::1:1883                :::*
LISTEN      1692/mosquitto
```

The service can be stopped and restarted with `systemctl`:

```
service mosquitto status
service mosquitto stop
service mosquitto start
service mosquitto restart
```

Configuration

To configure Mosquitto put ***.conf** files into the `/etc/mosquitto/conf.d/` directory. Any files placed in this directory that have a .conf ending will be loaded as config files by

the broker. My broker was missing the default config template after installation ~ so I added it here at the end of the article:

```
nano /etc/mosquitto/conf.d/custom.conf
```

Changing the Port

E.g. change the listener port from `1883` to `1885` and restart the service:

```
# Port to use for the default listener
listener 1885
# After copying in the default config this was suddenly set to false
- I set it back to true for now
allow_anonymous false
```

You now have to add the port to every sub & pub:

```
mosquitto_sub -h localhost -t /hello/world -p 1885
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -p 1885
```

Changing the User Login

In the following command replace the `user` with your username:

```
mosquitto_passwd -c passwordfile user
```

The password tool will ask you for your password and add the user to Mosquitto:

```
mosquitto_passwd -c passwordfile admin

Password: instar
Reenter password: instar
```

Adding additional users

```
mosquitto_passwd -b passwordfile user password
```

Deleting Users

```
mosquitto_passwd -D passwordfile user
```

Activating User Auth

Copy the `passwordfile` to where you want to store it:

```
cp passwordfile /etc/mosquitto/passwordfile
```

Deactivate the anonymous login and add the `passwordfile` path to your Mosquitto config:

```
allow_anonymous false
password_file /etc/mosquitto/passwordfile
```

Restart the service and try using it with your new user:

```
mosquitto_sub -h localhost -t /hello/world -p 1885 -u admin -P instar

mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -p 1885
```

Access Control List

Mosquitto comes with a default ACL file `cat /etc/mosquitto/aclfile.example`:

```
# This affects access control for clients with no username.
topic read $SYS/#

# This only affects clients with username "roger".
user roger
topic foo/bar

# This affects all clients.
pattern write $SYS/broker/connection/%c/state
```

Copy that file `cp /etc/mosquitto/aclfile.example /etc/mosquitto/acl.file` and add the `admin` user there:

```
user admin
topic readwrite cameras/#
topic read $SYS/broker/#
topic write hello/world
```

Now add the `acl.file` file location to your mosquitto config and reload the service `service mosquitto restart`:

```
# If an auth_plugin is used as well as acl_file, the auth_plugin
check will be
# made first.
acl_file /etc/mosquitto/acl.file
```

You can now **read & write** all topics starting with `cameras/`, **only read** broker status topics `$SYS/broker/` and only write to `hello/world`.

Publish & Subscribe

Pub Flags

> mosquitto_pub -h host -t topic –m message

- **-r** : Sets retain flag
- **-n** : Sends Null message useful for clearing retain message.
- **-p** : Set Port number Default is 1883
- **-u** : Provide a username
- **-P** : Provide a password
- **-i** : Provide client name
- **-I** : Provide a client id prefix- Used when testing client restrictions using prefix security.
- **-q** : QoS Specify the quality of service to use for the message, from 0, 1 and 2. Defaults to 0

Sub Flags

> mosquitto_sub -h host -t topic

- **-p** : Set Port number Default is 1883
- **-u** : Provide a username
- **-P** : Provide a password
- **-i** : Provide client name
- **-I** : Provide a client id prefix- Used when testing client restrictions using prefix security

Debugging

```
mosquitto_sub -h localhost -t /hello/world -d
```

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -d

Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, '/hello/world', ... (8
bytes))
Client (null) sending DISCONNECT
```

Publish Parameter	Description
d0 / d1	?
q0 - q2	QoS 0-2
r0 / r1	Retained
m0 / m1	Message Payload

```
mosquitto_sub -h localhost -t /hello/world -d

Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: /hello/world, QoS: 0,
Options: 0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
Client (null) received PUBLISH (d0, q0, r0, m0, '/hello/world', ...
```

```
(8 bytes))
welcome!
```

Retaining

If you send an update and then subscribe to the topic afterwards you will not receive the update:

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!'
```

```
mosquitto_sub -h localhost -t /#

<no messages>
```

Unlesss the update was send with the `retained` flag:

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -r
```

Re-run the subscription and you will see the retained message:

```
mosquitto_sub -h localhost -t /#

Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: /#, QoS: 0, Options:
0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
Client (null) received PUBLISH (d0, q0, r1, m0, '/hello/world', ...
(8 bytes))
welcome!
```

Clearing Retained Topics

To clear (null) a retained topic use the `-n` flag

```
mosquitto_pub -h localhost -t /hello/world -d -n
```

```
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, '/hello/world', ... (0
bytes))
Client (null) sending DISCONNECT
```

Your subscriber will now set the topic from `r1` to `r0`:

```
mosquitto_sub -h localhost -t /# -d

Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: /#, QoS: 0, Options:
0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
Client (null) received PUBLISH (d0, q0, r1, m0, '/hello/world', ...
(8 bytes))
welcome!
Client (null) received PUBLISH (d0, q0, r0, m0, '/hello/world', ...
(0 bytes))
```

Quality-of-Service

QoS 0

Fire&Forget

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -d -q 0

Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, '/hello/world', ... (8
bytes))
Client (null) sending DISCONNECT
```

QoS 1

Every Subscriber will Receive the Update

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -d -q 1
```

```
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, '/hello/world', ... (8
bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

QoS 2

Every Subscriber will Receive the Update exactly one time!

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -d -q 2

Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q2, r0, m1, '/hello/world', ... (8
bytes))
Client (null) received PUBREC (Mid: 1)
Client (null) sending PUBREL (m1)
Client (null) received PUBCOMP (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

The receiver always receives the update with `q0` - you need to run your subscription with the `q1` or `q2` flag (**why is that?**). Se below - I published the update 3 times with different QoS values:

q0

```
mosquitto_sub -h localhost -t /# -d

Client (null) received PUBLISH (d0, q0, r0, m0, '/hello/world', ...
(8 bytes))
welcome!
Client (null) received PUBLISH (d0, q0, r0, m0, '/hello/world', ...
(8 bytes))
welcome!
Client (null) received PUBLISH (d0, q0, r0, m0, '/hello/world', ...
(8 bytes))
welcome!
```

If I redo the `q2` publish but make sure that the receiver is also set to `q2`:

q2

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -q 2
```

```
mosquitto_sub -h localhost -t /# -d -q 2

Client (null) received PUBLISH (d0, q2, r0, m1, '/hello/world', ...
(8 bytes))
Client (null) sending PUBREC (m1, rc0)
Client (null) received PUBREL (Mid: 1)
Client (null) sending PUBCOMP (m1)
welcome!
```

Client Name

```
mosquitto_pub -h localhost -t /hello/world -m 'welcome!' -i
'MosquittoPub' -d

Client MosquittoPub sending CONNECT
Client MosquittoPub received CONNACK (0)
Client MosquittoPub sending PUBLISH (d0, q0, r0, m1, '/hello/world',
... (8 bytes))
Client MosquittoPub sending DISCONNEC
```

```
mosquitto_sub -h localhost -t /# -d -i 'MosquittoSub'

Client MosquittoSub sending CONNECT
Client MosquittoSub received CONNACK (0)
Client MosquittoSub sending SUBSCRIBE (Mid: 1, Topic: /#, QoS: 0,
Options: 0x00)
Client MosquittoSub received SUBACK
Subscribed (mid: 1): 0
Client MosquittoSub received PUBLISH (d0, q0, r0, m0, '/hello/world',
... (8 bytes))
welcome!
```

Wildcards

Receiving the the measurements of different sensors with a single subscription:

- sensors/altimeter1/altitude
- sensors/altimeter2/altitude
- sensors/altimeter3/altitude
- sensors/altimeter4/altitude

Plus

```
mosquitto_sub -h localhost -t sensors/+/altitude -v

sensors/altimeter1/altitude 200
sensors/altimeter2/altitude 201
sensors/altimeter3/altitude 199
sensors/altimeter4/altitude 198
```

Hash

```
mosquitto_sub -h localhost -t sensors/# -v

sensors/altimeter1/operation active
sensors/altimeter1/status green
sensors/altimeter1/altitude 200
sensors/altimeter2/operation active
sensors/altimeter2/status green
sensors/altimeter2/altitude 201
sensors/altimeter3/operation active
sensors/altimeter3/status green
sensors/altimeter3/altitude 199
sensors/altimeter4/operation active
sensors/altimeter4/status green
sensors/altimeter4/altitude 198
```

Default mosquitto.conf

This file contains the default values for the Mosquitto (v.2.0.11) configuration. Copy it to `/etc/mosquitto/conf.d/custom.conf` and modify the values you need to change:

```
# Config file for mosquitto
#
# See mosquitto.conf(5) for more information.
```

```
#
# Default values are shown, uncomment to change.
#
# Use the # character to indicate a comment, but only if it is the
# very first character on the line.

# =================================================================
# General configuration
# =================================================================

# Use per listener security settings.
#
# It is recommended this option be set before any other options.
#
# If this option is set to true, then all authentication and access cont
# options are controlled on a per listener basis. The following options
# affected:
#
# password_file acl_file psk_file auth_plugin auth_opt_* allow_anonymous
# auto_id_prefix allow_zero_length_clientid
#
# Note that if set to true, then a durable client (i.e. with clean sessi
set
# to false) that has disconnected will use the ACL settings defined for
# listener that it was most recently connected to.
#
# The default behaviour is for this to be set to false, which maintains
# setting behaviour from previous versions of mosquitto.
#per_listener_settings false


# This option controls whether a client is allowed to connect with a zer
# length client id or not. This option only affects clients using MQTT
v3.1.1
# and later. If set to false, clients connecting with a zero length clie
id
# are disconnected. If set to true, clients will be allocated a client i
# the broker. This means it is only useful for clients with clean sessio
set
# to true.
#allow_zero_length_clientid true

# If allow_zero_length_clientid is true, this option allows you to set a
prefix
# to automatically generated client ids to aid visibility in logs.
# Defaults to 'auto-'
```

```
#auto_id_prefix auto-

# This option affects the scenario when a client subscribes to a topic t
has
# retained messages. It is possible that the client that published the
retained
# message to the topic had access at the time they published, but that
access
# has been subsequently removed. If check_retain_source is set to true,
# default, the source of a retained message will be checked for access
rights
# before it is republished. When set to false, no check will be made and
# retained message will always be published. This affects all listeners
#check_retain_source true

# QoS 1 and 2 messages will be allowed inflight per client until this li
# is exceeded.  Defaults to 0. (No maximum)
# See also max_inflight_messages
#max_inflight_bytes 0

# The maximum number of QoS 1 and 2 messages currently inflight per
# client.
# This includes messages that are partway through handshakes and
# those that are being retried. Defaults to 20. Set to 0 for no
# maximum. Setting to 1 will guarantee in-order delivery of QoS 1
# and 2 messages.
#max_inflight_messages 20

# For MQTT v5 clients, it is possible to have the server send a "server
# keepalive" value that will override the keepalive value set by the cli
# This is intended to be used as a mechanism to say that the server will
# disconnect the client earlier than it anticipated, and that the client
should
# use the new keepalive value. The max_keepalive option allows you to
specify
# that clients may only connect with keepalive less than or equal to thi
# value, otherwise they will be sent a server keepalive telling them to
# max_keepalive. This only applies to MQTT v5 clients. The maximum value
# allowable is 65535. Do not set below 10.
#max_keepalive 65535

# For MQTT v5 clients, it is possible to have the server send a "maximum
packet
# size" value that will instruct the client it will not accept MQTT pack
# with size greater than max_packet_size bytes. This applies to the full
MQTT
```

```
# packet, not just the payload. Setting this option to a positive value
# set the maximum packet size to that number of bytes. If a client sends
# packet which is larger than this value, it will be disconnected. This
applies
# to all clients regardless of the protocol version they are using, but
v3.1.1
# and earlier clients will of course not have received the maximum packe
size
# information. Defaults to no limit. Setting below 20 bytes is forbidden
# because it is likely to interfere with ordinary client operation, even
with
# very small payloads.
#max_packet_size 0

# QoS 1 and 2 messages above those currently in-flight will be queued pe
# client until this limit is exceeded.  Defaults to 0. (No maximum)
# See also max_queued_messages.
# If both max_queued_messages and max_queued_bytes are specified, packet
will
# be queued until the first limit is reached.
#max_queued_bytes 0

# Set the maximum QoS supported. Clients publishing at a QoS higher than
# specified here will be disconnected.
#max_qos 2

# The maximum number of QoS 1 and 2 messages to hold in a queue per clie
# above those that are currently in-flight.  Defaults to 1000. Set
# to 0 for no maximum (not recommended).
# See also queue_qos0_messages.
# See also max_queued_bytes.
#max_queued_messages 1000
#
# This option sets the maximum number of heap memory bytes that the brok
will
# allocate, and hence sets a hard limit on memory use by the broker.  Me
# requests that exceed this value will be denied. The effect will vary
# depending on what has been denied. If an incoming message is being
processed,
# then the message will be dropped and the publishing client will be
# disconnected. If an outgoing message is being sent, then the individua
# message will be dropped and the receiving client will be disconnected
# Defaults to no limit.
#memory_limit 0

# This option sets the maximum publish payload size that the broker wil
```

```
allow.
# Received messages that exceed this size will not be accepted by the
broker.
# The default value is 0, which means that all valid MQTT messages are
# accepted. MQTT imposes a maximum payload size of 268435455 bytes.
#message_size_limit 0

# This option allows persistent clients (those with clean session set to
false)
# to be removed if they do not reconnect within a certain time frame.
#
# This is a non-standard option in MQTT V3.1 but allowed in MQTT v3.1.1
#
# Badly designed clients may set clean session to false whilst using a
randomly
# generated client id. This leads to persistent clients that will never
# reconnect. This option allows these clients to be removed.
#
# The expiration period should be an integer followed by one of h d w m
for
# hour, day, week, month and year respectively. For example
#
# persistent_client_expiration 2m
# persistent_client_expiration 14d
# persistent_client_expiration 1y
#
# The default if not set is to never expire persistent clients.
#persistent_client_expiration

# Write process id to a file. Default is a blank string which means
# a pid file shouldn't be written.
# This should be set to /var/run/mosquitto/mosquitto.pid if mosquitto is
# being run automatically on boot with an init script and
# start-stop-daemon or similar.
#pid_file

# Set to true to queue messages with QoS 0 when a persistent client is
# disconnected. These messages are included in the limit imposed by
# max_queued_messages and max_queued_bytes
# Defaults to false.
# This is a non-standard option for the MQTT v3.1 spec but is allowed in
# v3.1.1.
#queue_qos0_messages false

# Set to false to disable retained message support. If a client publishes
# message with the retain bit set, it will be disconnected if this is set
```

```
# false.
#retain_available true

# Disable Nagle's algorithm on client sockets. This has the effect of
reducing
# latency of individual messages at the potential cost of increasing the
number
# of packets being sent.
#set_tcp_nodelay false

# Time in seconds between updates of the $SYS tree.
# Set to 0 to disable the publishing of the $SYS tree.
#sys_interval 10

# The MQTT specification requires that the QoS of a message delivered to
# subscriber is never upgraded to match the QoS of the subscription.
Enabling
# this option changes this behaviour. If upgrade_outgoing_qos is set tru
# messages sent to a subscriber will always match the QoS of its
subscription.
# This is a non-standard option explicitly disallowed by the spec.
#upgrade_outgoing_qos false

# When run as root, drop privileges to this user and its primary
# group.
# Set to root to stay as root, but this is not recommended.
# If set to "mosquitto", or left unset, and the "mosquitto" user does no
exist
# then it will drop privileges to the "nobody" user instead.
# If run as a non-root user, this setting has no effect.
# Note that on Windows this has no effect and so mosquitto should be sta
by
# the user you wish it to run as.
#user mosquitto

# ================================================================
# Listeners
# ================================================================

# Listen on a port/ip address combination. By using this variable
# multiple times, mosquitto can listen on more than one port. If
# this variable is used and neither bind_address nor port given,
# then the default listener will not be started.
# The port number to listen on must be given. Optionally, an ip
# address or host name may be supplied as a second argument. In
# this case, mosquitto will attempt to bind the listener to that
```

```
# address and so restrict access to the associated network and
# interface. By default, mosquitto will listen on all interfaces.
# Note that for a websockets listener it is not possible to bind to a ho
# name.
#
# On systems that support Unix Domain Sockets, it is also possible
# to create a # Unix socket rather than opening a TCP socket. In
# this case, the port number should be set to 0 and a unix socket
# path must be provided, e.g.
# listener 0 /tmp/mosquitto.sock
#
# listener port-number [ip address/host name/unix socket path]
#listener

# By default, a listener will attempt to listen on all supported IP prot
# versions. If you do not have an IPv4 or IPv6 interface you may wish to
# disable support for either of those protocol versions. In particular,
# that due to the limitations of the websockets library, it will only ev
# attempt to open IPv6 sockets if IPv6 support is compiled in, and so wi
fail
# if IPv6 is not available.
#
# Set to `ipv4` to force the listener to only use IPv4, or set to `ipv6`
# force the listener to only use IPv6. If you want support for both IPv4
# IPv6, then do not use the socket_domain option.
#
#socket_domain

# Bind the listener to a specific interface. This is similar to
# the [ip address/host name] part of the listener definition, but is use
# when an interface has multiple addresses or the address may change. If
used
# with the [ip address/host name] part of the listener definition, then
# bind_interface option will take priority.
# Not available on Windows.
#
# Example: bind_interface eth0
#bind_interface

# When a listener is using the websockets protocol, it is possible to se
# http data as well. Set http_dir to a directory which contains the file
you
# wish to serve. If this option is not specified, then no normal http
# connections will be possible.
#http_dir
```

```
# The maximum number of client connections to allow. This is
# a per listener setting.
# Default is -1, which means unlimited connections.
# Note that other process limits mean that unlimited connections
# are not really possible. Typically the default maximum number of
# connections possible is around 1024.
#max_connections -1

# The listener can be restricted to operating within a topic hierarchy u
# the mount_point option. This is achieved be prefixing the mount_point
string
# to all topics for any clients connected to this listener. This prefix
only
# happens internally to the broker; the client will not see the prefix.
#mount_point

# Choose the protocol to use when listening.
# This can be either mqtt or websockets.
# Certificate based TLS may be used with websockets, except that only th
# cafile, certfile, keyfile, ciphers, and ciphers_tls13 options are
supported.
#protocol mqtt

# Set use_username_as_clientid to true to replace the clientid that a cl
# connected with with its username. This allows authentication to be tie
# the clientid, which means that it is possible to prevent one client
# disconnecting another by using the same clientid.
# If a client connects with no username it will be disconnected as not
# authorised when this option is set to true.
# Do not use in conjunction with clientid_prefixes.
# See also use_identity_as_username.
#use_username_as_clientid

# Change the websockets headers size. This is a global option, it is not
# possible to set per listener. This option sets the size of the buffer
in
# the libwebsockets library when reading HTTP headers. If you are passin
large
# header data such as cookies then you may need to increase this value.
left
# unset, or set to 0, then the default of 1024 bytes will be used.
#websockets_headers_size

# ================================================================
# Certificate based SSL/TLS support
# ================================================================
```

```
# The following options can be used to enable certificate based SSL/TLS
support
# for this listener. Note that the recommended port for MQTT over TLS is
8883,
# but this must be set manually.
#
# See also the mosquitto-tls man page and the "Pre-shared-key based SSL/
# support" section. Only one of certificate or PSK encryption support ca
# enabled for any listener.

# Both of certfile and keyfile must be defined to enable certificate bas
# TLS encryption.

# Path to the PEM encoded server certificate.
#certfile

# Path to the PEM encoded keyfile.
#keyfile

# If you wish to control which encryption ciphers are used, use the ciph
# option. The list of available ciphers can be optained using the "opens
# ciphers" command and should be provided in the same format as the outp
of
# that command. This applies to TLS 1.2 and earlier versions only. Use
# ciphers_tls1.3 for TLS v1.3.
#ciphers

# Choose which TLS v1.3 ciphersuites are used for this listener.
# Defaults to
"TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA
#ciphers_tls1.3

# If you have require_certificate set to true, you can create a certific
# revocation list file to revoke access to particular client certificate
If
# you have done this, use crlfile to point to the PEM encoded revocation
file.
#crlfile

# To allow the use of ephemeral DH key exchange, which provides forward
# security, the listener must load DH parameters. This can be specified
# the dhparamfile option. The dhparamfile can be generated with the comm
# e.g. "openssl dhparam -out dhparam.pem 2048"
#dhparamfile

# By default an TLS enabled listener will operate in a similar fashion t
```

```
# https enabled web server, in that the server has a certificate signed
CA
# and the client will verify that it is a trusted certificate. The overa
aim
# is encryption of the network traffic. By setting require_certificate t
true,
# the client must provide a valid certificate in order for the network
# connection to proceed. This allows access to the broker to be control
# outside of the mechanisms provided by MQTT.
#require_certificate false

# cafile and capath define methods of accessing the PEM encoded
# Certificate Authority certificates that will be considered trusted whe
# checking incoming client certificates.
# cafile defines the path to a file containing the CA certificates.
# capath defines a directory that will be searched for files
# containing the CA certificates. For capath to work correctly, the
# certificate files must have ".crt" as the file ending and you must run
# "openssl rehash <path to capath>" each time you add/remove a certifica
#cafile
#capath


# If require_certificate is true, you may set use_identity_as_username t
true
# to use the CN value from the client certificate as a username. If this
# true, the password_file option will not be used for this listener.
#use_identity_as_username false

# ----------------------------------------------------------------
# Pre-shared-key based SSL/TLS support
# ----------------------------------------------------------------
# The following options can be used to enable PSK based SSL/TLS support
# this listener. Note that the recommended port for MQTT over TLS is 888
but
# this must be set manually.
#
# See also the mosquitto-tls man page and the "Certificate based SSL/TLS
# support" section. Only one of certificate or PSK encryption support ca
# enabled for any listener.

# The psk_hint option enables pre-shared-key support for this listener a
also
# acts as an identifier for this listener. The hint is sent to clients a
may
# be used locally to aid authentication. The hint is a free form string
```

```
# doesn't have much meaning in itself, so feel free to be creative.
# If this option is provided, see psk_file to define the pre-shared keys
be
# used or create a security plugin to handle them.
#psk_hint

# When using PSK, the encryption ciphers used will be chosen from the li
of
# available PSK ciphers. If you want to control which ciphers are availa
# use the "ciphers" option.  The list of available ciphers can be optain
# using the "openssl ciphers" command and should be provided in the same
format
# as the output of that command.
#ciphers

# Set use_identity_as_username to have the psk identity sent by the clie
used
# as its username. Authentication will be carried out using the PSK rath
than
# the MQTT username/password and so password_file will not be used for t
# listener.
#use_identity_as_username false


# =================================================================
# Persistence
# =================================================================

# If persistence is enabled, save the in-memory database to disk
# every autosave_interval seconds. If set to 0, the persistence
# database will only be written when mosquitto exits. See also
# autosave_on_changes.
# Note that writing of the persistence database can be forced by
# sending mosquitto a SIGUSR1 signal.
#autosave_interval 1800

# If true, mosquitto will count the number of subscription changes, reta
# messages received and queued messages and if the total exceeds
# autosave_interval then the in-memory database will be saved to disk.
# If false, mosquitto will save the in-memory database to disk by treati
# autosave_interval as a time in seconds.
#autosave_on_changes false

# Save persistent message data to disk (true/false).
# This saves information about all messages, including
# subscriptions, currently in-flight messages and retained
```

```
# messages.
# retained_persistence is a synonym for this option.
#persistence false

# The filename to use for the persistent database, not including
# the path.
#persistence_file mosquitto.db

# Location for persistent database.
# Default is an empty string (current directory).
# Set to e.g. /var/lib/mosquitto if running as a proper service on Linux
# similar.
#persistence_location


# =================================================================
# Logging
# =================================================================

# Places to log to. Use multiple log_dest lines for multiple
# logging destinations.
# Possible destinations are: stdout stderr syslog topic file dlt
#
# stdout and stderr log to the console on the named output.
#
# syslog uses the userspace syslog facility which usually ends up
# in /var/log/messages or similar.
#
# topic logs to the broker topic '$SYS/broker/log/<severity>',
# where severity is one of D, E, W, N, I, M which are debug, error,
# warning, notice, information and message. Message type severity is use
# the subscribe/unsubscribe log_types and publishes log messages to
# $SYS/broker/log/M/susbcribe or $SYS/broker/log/M/unsubscribe.
#
# The file destination requires an additional parameter which is the fil
be
# logged to, e.g. "log_dest file /var/log/mosquitto.log". The file will
# closed and reopened when the broker receives a HUP signal. Only a sing
file
# destination may be configured.
#
# The dlt destination is for the automotive `Diagnostic Log and Trace` f
# This requires that Mosquitto has been compiled with DLT support.
#
# Note that if the broker is running as a Windows service it will defaul
# "log_dest none" and neither stdout nor stderr logging is available.
```

```
# Use "log_dest none" if you wish to disable logging.
#log_dest stderr

# Types of messages to log. Use multiple log_type lines for logging
# multiple types of messages.
# Possible types are: debug, error, warning, notice, information,
# none, subscribe, unsubscribe, websockets, all.
# Note that debug type messages are for decoding the incoming/outgoing
# network packets. They are not logged in "topics".
#log_type error
#log_type warning
#log_type notice
#log_type information


# If set to true, client connection and disconnection messages will be
included
# in the log.
#connection_messages true

# If using syslog logging (not on Windows), messages will be logged to t
# "daemon" facility by default. Use the log_facility option to choose wh
of
# local0 to local7 to log to instead. The option value should be an inte
# value, e.g. "log_facility 5" to use local5.
#log_facility

# If set to true, add a timestamp value to each log message.
#log_timestamp true

# Set the format of the log timestamp. If left unset, this is the numbe
# seconds since the Unix epoch.
# This is a free text string which will be passed to the strftime functi
To
# get an ISO 8601 datetime, for example:
# log_timestamp_format %Y-%m-%dT%H:%M:%S
#log_timestamp_format

# Change the websockets logging level. This is a global option, it is no
# possible to set per listener. This is an integer that is interpreted b
# libwebsockets as a bit mask for its lws_log_levels enum. See the
# libwebsockets documentation for more details. "log_type websockets" mu
also
# be enabled.
#websockets_log_level 0
```

```
# =================================================================
# Security
# =================================================================

# If set, only clients that have a matching prefix on their
# clientid will be allowed to connect to the broker. By default,
# all clients may connect.
# For example, setting "secure-" here would mean a client "secure-
# client" could connect but another with clientid "mqtt" couldn't.
#clientid_prefixes

# Boolean value that determines whether clients that connect
# without providing a username are allowed to connect. If set to
# false then a password file should be created (see the
# password_file option) to control authenticated client access.
#
# Defaults to false, unless there are no listeners defined in the
configuration
# file, in which case it is set to true, but connections are only allowe
from
# the local machine.
#allow_anonymous false

# -----------------------------------------------------------------
# Default authentication and topic access control
# -----------------------------------------------------------------

# Control access to the broker using a password file. This file can be
# generated using the mosquitto_passwd utility. If TLS support is not
compiled
# into mosquitto (it is recommended that TLS support should be included)
then
# plain text passwords are used, in which case the file should be a text
file
# with lines in the format:
# username:password
# The password (and colon) may be omitted if desired, although this
# offers very little in the way of security.
#
# See the TLS client require_certificate and use_identity_as_username
options
# for alternative authentication options. If an auth_plugin is used as w
as
# password_file, the auth_plugin check will be made first.
#password_file
```

```
# Access may also be controlled using a pre-shared-key file. This requi
# TLS-PSK support and a listener configured to use it. The file should b
text
# lines in the format:
# identity:key
# The key should be in hexadecimal format without a leading "0x".
# If an auth_plugin is used as well, the auth_plugin check will be made
first.
#psk_file

# Control access to topics on the broker using an access control list
# file. If this parameter is defined then only the topics listed will
# have access.
# If the first character of a line of the ACL file is a # it is treated
# comment.
# Topic access is added with lines of the format:
#
# topic [read|write|readwrite|deny] <topic>
#
# The access type is controlled using "read", "write", "readwrite" or
"deny".
# This parameter is optional (unless <topic> contains a space character)
if
# not given then the access is read/write.  <topic> can contain the + or
# wildcards as in subscriptions.
#
# The "deny" option can used to explicity deny access to a topic that wo
# otherwise be granted by a broader read/write/readwrite statement. Any
"deny"
# topics are handled before topics that grant read/write access.
#
# The first set of topics are applied to anonymous clients, assuming
# allow_anonymous is true. User specific topic ACLs are added after a
# user line as follows:
#
# user <username>
#
# The username referred to here is the same as in password_file. It is
# not the clientid.
#
#
# If is also possible to define ACLs based on pattern substitution with
the
# topic. The patterns available for substition are:
#
```

```
# %c to match the client id of the client
# %u to match the username of the client
#
# The substitution pattern must be the only text for that level of
hierarchy.
#
# The form is the same as for the topic keyword, but using pattern as th
# keyword.
# Pattern ACLs apply to all users even if the "user" keyword has previou
# been given.
#
# If using bridges with usernames and ACLs, connection messages can be
allowed
# with the following pattern:
# pattern write $SYS/broker/connection/%c/state
#
# pattern [read|write|readwrite] <topic>
#
# Example:
#
# pattern write sensor/%u/data
#
# If an auth_plugin is used as well as acl_file, the auth_plugin check w
be
# made first.
#acl_file

# ----------------------------------------------------------------
# External authentication and topic access plugin options
# ----------------------------------------------------------------

# External authentication and access control can be supported with the
# auth_plugin option. This is a path to a loadable plugin. See also the
# auth_opt_* options described below.
#
# The auth_plugin option can be specified multiple times to load multipl
# plugins. The plugins will be processed in the order that they are
specified
# here. If the auth_plugin option is specified alongside either of
# password_file or acl_file then the plugin checks will be made first.
#
#auth_plugin

# If the auth_plugin option above is used, define options to pass to the
# plugin here as described by the plugin instructions. All options named
# using the format auth_opt_* will be passed to the plugin, for example
```

```
#
# auth_opt_db_host
# auth_opt_db_port
# auth_opt_db_username
# auth_opt_db_password


# =================================================================
# Bridges
# =================================================================

# A bridge is a way of connecting multiple MQTT brokers together.
# Create a new bridge using the "connection" option as described below.
# options for the bridges using the remaining parameters. You must spec
the
# address and at least one topic to subscribe to.
#
# Each connection must have a unique name.
#
# The address line may have multiple host address and ports specified. S
# below in the round_robin description for more details on bridge behav
if
# multiple addresses are used. Note that if you use an IPv6 address, the
you
# are required to specify a port.
#
# The direction that the topic will be shared can be chosen by
# specifying out, in or both, where the default value is out.
# The QoS level of the bridged communication can be specified with the
# topic option. The default QoS level is 0, to change the QoS the topic
# direction must also be given.
#
# The local and remote prefix options allow a topic to be remapped when
is
# bridged to/from the remote broker. This provides the ability to place
topic
# tree in an appropriate location.
#
# For more details see the mosquitto.conf man page.
#
# Multiple topics can be specified per connection, but be careful
# not to create any loops.
#
# If you are using bridges with cleansession set to false (the default),
then
# you may get unexpected behaviour from incoming topics if you change wh
```

```
# topics you are subscribing to. This is because the remote broker keeps
# subscription for the old topic. If you have this problem, connect your
bridge
# with cleansession set to true, then reconnect with cleansession set to
false
# as normal.
#connection <name>
#address <host>[:<port>] [<host>[:<port>]]
#topic <topic> [[[out | in | both] qos-level] local-prefix remote-prefix

# If you need to have the bridge connect over a particular network
interface,
# use bridge_bind_address to tell the bridge which local IP address the
socket
# should bind to, e.g. `bridge_bind_address 192.168.1.10`
#bridge_bind_address

# If a bridge has topics that have "out" direction, the default behaviour
to
# send an unsubscribe request to the remote broker on that topic. This m
# that changing a topic direction from "in" to "out" will not keep receiv
# incoming messages. Sending these unsubscribe requests is not always
# desirable, setting bridge_attempt_unsubscribe to false will disable
sending
# the unsubscribe request.
#bridge_attempt_unsubscribe true

# Set the version of the MQTT protocol to use with for this bridge. Can
one
# of mqttv50, mqttv311 or mqttv31. Defaults to mqttv311.
#bridge_protocol_version mqttv311

# Set the clean session variable for this bridge.
# When set to true, when the bridge disconnects for any reason, all
# messages and subscriptions will be cleaned up on the remote
# broker. Note that with cleansession set to true, there may be a
# significant amount of retained messages sent when the bridge
# reconnects after losing its connection.
# When set to false, the subscriptions and messages are kept on the
# remote broker, and delivered when the bridge reconnects.
#cleansession false

# Set the amount of time a bridge using the lazy start type must be idle
before
# it will be stopped. Defaults to 60 seconds.
#idle_timeout 60
```

```
# Set the keepalive interval for this bridge connection, in
# seconds.
#keepalive_interval 60

# Set the clientid to use on the local broker. If not defined, this defa
to
# 'local.<clientid>'. If you are bridging a broker to itself, it is
important
# that local_clientid and clientid do not match.
#local_clientid

# If set to true, publish notification messages to the local and remote
brokers
# giving information about the state of the bridge connection. Retained
# messages are published to the topic $SYS/broker/connection/<clientid>
/state
# unless the notification_topic option is used.
# If the message is 1 then the connection is active, or 0 if the connect
has
# failed.
# This uses the last will and testament feature.
#notifications true

# Choose the topic on which notification messages for this bridge are
# published. If not set, messages are published on the topic
# $SYS/broker/connection/<clientid>/state
#notification_topic

# Set the client id to use on the remote end of this bridge connection.
not
# defined, this defaults to 'name.hostname' where name is the connectior
name
# and hostname is the hostname of this computer.
# This replaces the old "clientid" option to avoid confusion. "clientid'
# remains valid for the time being.
#remote_clientid

# Set the password to use when connecting to a broker that requires
# authentication. This option is only used if remote_username is also se
# This replaces the old "password" option to avoid confusion. "password'
# remains valid for the time being.
#remote_password

# Set the username to use when connecting to a broker that requires
# authentication.
```

```
# This replaces the old "username" option to avoid confusion. "username"
# remains valid for the time being.
#remote_username

# Set the amount of time a bridge using the automatic start type will wa
# until attempting to reconnect.
# This option can be configured to use a constant delay time in seconds,
to
# use a backoff mechanism based on "Decorrelated Jitter", which adds a
degree
# of randomness to when the restart occurs.
#
# Set a constant timeout of 20 seconds:
# restart_timeout 20
#
# Set backoff with a base (start value) of 10 seconds and a cap (upper
limit) of
# 60 seconds:
# restart_timeout 10 30
#
# Defaults to jitter with a base of 5 and cap of 30
#restart_timeout 5 30

# If the bridge has more than one address given in the address/addresses
# configuration, the round_robin option defines the behaviour of the bri
on
# a failure of the bridge connection. If round_robin is false, the defau
# value, then the first address is treated as the main bridge connection
# the connection fails, the other secondary addresses will be attempted
# turn. Whilst connected to a secondary bridge, the bridge will periodic
# attempt to reconnect to the main bridge until successful.
# If round_robin is true, then all addresses are treated as equals. If a
# connection fails, the next address will be tried and if successful wil
# remain connected until it fails
#round_robin false

# Set the start type of the bridge. This controls how the bridge starts
# can be one of three types: automatic, lazy and once. Note that RSMB
provides
# a fourth start type "manual" which isn't currently supported by mosqui
#
# "automatic" is the default start type and means that the bridge connec
# will be started automatically when the broker starts and also restarte
# after a short delay (30 seconds) if the connection fails.
#
# Bridges using the "lazy" start type will be started automatically when
```

```
# number of queued messages exceeds the number set with the "threshold"
# parameter. It will be stopped automatically after the time set by the
# "idle_timeout" parameter. Use this start type if you wish the connecti
to
# only be active when it is needed.
#
# A bridge using the "once" start type will be started automatically whe
the
# broker starts but will not be restarted if the connection fails.
#start_type automatic

# Set the number of messages that need to be queued for a bridge with la
# start type to be restarted. Defaults to 10 messages.
# Must be less than max_queued_messages.
#threshold 10

# If try_private is set to true, the bridge will attempt to indicate to
# remote broker that it is a bridge not an ordinary client. If successfu
this
# means that loop detection will be more effective and that retained
messages
# will be propagated correctly. Not all brokers support this feature so
may
# be necessary to set try_private to false if your bridge does not conne
# properly.
#try_private true

# Some MQTT brokers do not allow retained messages. MQTT v5 gives a
mechanism
# for brokers to tell clients that they do not support retained messages
but
# this is not possible for MQTT v3.1.1 or v3.1. If you need to bridge to
# v3.1.1 or v3.1 broker that does not support retained messages, set the
# bridge_outgoing_retain option to false. This will remove the retain bi
# all outgoing messages to that bridge, regardless of any other setting
#bridge_outgoing_retain true

# If you wish to restrict the size of messages sent to a remote bridge,
the
# bridge_max_packet_size option. This sets the maximum number of bytes f
# the total message, including headers and payload.
# Note that MQTT v5 brokers may provide their own maximum-packet-size
property.
# In this case, the smaller of the two limits will be used.
# Set to 0 for "unlimited".
#bridge_max_packet_size 0
```

```
# ----------------------------------------------------------------
# Certificate based SSL/TLS support
# ----------------------------------------------------------------
# Either bridge_cafile or bridge_capath must be defined to enable TLS
support
# for this bridge.
# bridge_cafile defines the path to a file containing the
# Certificate Authority certificates that have signed the remote broker
# certificate.
# bridge_capath defines a directory that will be searched for files
containing
# the CA certificates. For bridge_capath to work correctly, the certific
# files must have ".crt" as the file ending and you must run "openssl re
# <path to capath>" each time you add/remove a certificate.
#bridge_cafile
#bridge_capath


# If the remote broker has more than one protocol available on its port,
e.g.
# MQTT and WebSockets, then use bridge_alpn to configure which protocol
# requested. Note that WebSockets support for bridges is not yet availab
#bridge_alpn

# When using certificate based encryption, bridge_insecure disables
# verification of the server hostname in the server certificate. This ca
# useful when testing initial server configurations, but makes it possib
for
# a malicious third party to impersonate your server through DNS spoofin
for
# example. Use this option in testing only. If you need to resort to usi
this
# option in a production environment, your setup is at fault and there i
# point using encryption.
#bridge_insecure false

# Path to the PEM encoded client certificate, if required by the remote
broker.
#bridge_certfile

# Path to the PEM encoded client private key, if required by the remote
broker.
#bridge_keyfile
```

```
# ----------------------------------------------------------------
# PSK based SSL/TLS support
# ----------------------------------------------------------------
# Pre-shared-key encryption provides an alternative to certificate based
# encryption. A bridge can be configured to use PSK with the bridge_iden
# and bridge_psk options. These are the client PSK identity, and pre-sha
key
# in hexadecimal format with no "0x". Only one of certificate and PSK ba
# encryption can be used on one
# bridge at once.
#bridge_identity
#bridge_psk


# ================================================================
# External config files
# ================================================================

# External configuration files may be included by using the
# include_dir option. This defines a directory that will be searched
# for config files. All files that end in '.conf' will be loaded as
# a configuration file. It is best to have this as the last option
# in the main file. This option will only be processed from the main
# configuration file. The directory specified must not contain the
# main configuration file.
# Files within include_dir will be loaded sorted in case-sensitive
# alphabetical order, with capital letters ordered first. If this optior
# given multiple times, all of the files from the first instance will be
# processed before the next instance. See the man page for examples.
#include_dir
```

Tags: MQTT

Edit this page