**mqttbench**

**(Getting Started)**

**IBM Corporation**

(rev 1.56)

June 17, 2019

**Revision History**

| Revision | Description of Version | Date Completed |
| --- | --- | --- |
| 0.5 | Initial version | June 15, 2013 |
| 0.6 | Updates | Feb 26, 2014 |
| 0.7 | Updates | Mar 6, 2014 |
| 0.8 | Updated Environment Variables and Command Line Options | Apr 30, 204 |
| 0.9 | Added Kernel tuning parameters, NIC tuning parameters, User and process limits, OS services stopped and added example of –pfxi option | May 1, 2014 |
| 1.0 | Updated with Client List option (-cl), environment settings for TCP retry (RetryDelay, RetryBackoff), and support for MQTT v3.1.1 | Aug 12, 2014 |
| 1.1 | Updated for PreShared Key Based Cipher Support (-psk <filename>) | Oct 28, 2014 |
| 1.2 | Updated for new commands, IMAServer additional format, link to MessageSight Performance report which includes mqttbench examples. | Mar 4, 2015 |
| 1.3 | Updated for new options: Exit on First Disconnect (-efd), No Reconnection (-noreconn), Jitter statistics (-jitter), and IMAServerSets Reconnection is now the default. | Nov 10, 2015 |
| 1.4 | Updated for the new client list file format (-cl), enhanced docs for miscellaneous other environment variables | Jan 14, 2016 |
| 1.5 | Updates to provide further clarification with either documentation and/or screenshots. | Jan 24, 2016 |
| 1.6 | Internal design change (submitter threads), additional support for new client list format, Wait Ready (-wr) modification, Round Robin sending of predefined message files, setting the number of submitter threads, modification of stats, latency testing for producers only from TCP/IP connection to receiving MQTT CONNACK, Round Robin Send/Subscribe is now default, and new features: -crange <x-y>, -sseq | Feb 25, 2016 |
| 1.7 | Updates for multiple –crange <x1-x2,y1-y2,z1-z2> with support for comments in the client list. Updated support for the “c” command on the command line. | Mar 9, 2016 |
| 1.8 | Additional updates for –crange and “c” command. Connection information explained. | Mar 10, 2016 |
| 1.9 | Updates for –snap and additional information pertaining to statistics provided. | April 5, 2016 |
| 1.10 | Updates for new client list which supports Destination IP Addr & Port, TLS, and WebSockets columns. No longer supporting 4 Column version (ClientID, Topic, Username and Password). Addition of uptime command and TCP Sock Create Statistics. | April 20, 2016 |
| 1.11 | Clarification on –snap parameter with latency. | June 16, 2016 |
| 1.12 | Clean up unused parameters; Addition of MBAutoLog environment variable. | July 12, 2016 |
| 1.13 | Update for new –rl option, which performs Reset Latency Stats | Aug 10, 2016 |
| 1.14 | Fix the headers for the snap counts and rates. | Aug 24, 2016 |
| 1.15 | Updates for client certificates, “resetlat” command and –M option. | Oct 6, 2016 |
| 1.16 | Updates for nesw Environment Variables for trimming memory, SSL Buffer Pool. Correlates to the mqttbench version of 3.19 | Dec 1, 2016 |
| 1.17 | Update to Client List format with Linger Info column. | Jan 24, 2017 |
| 1.18 | Update to Client List format with Client Certificates, Client Private Key and Message Directory. Message Directory will be supported in future release. | Mar 10, 2017 |
| 1.19 | Update to Client List to ensure there are proper column separators. | May 2, 2017 |
| 1.20 | Version 4.4 - Message Directory support; Version 4.4.1 – Support **–as** command line option | Aug 21, 2017 |
| 1.21 | Version 4.5 – Support for timestamping specific error messages and logging, Support for –burst mode. | Sept 1, 2017 |
| 1.22 | Updates | Sept 25, 2017 |
| 1.23 | Updates | Oct 10, 2017 |
| 1.24 | Version 4.5.4 – Properly handles MQTT UNSUBSCRIBE and disconnects. | Oct 31, 2017 |
| 1.25 | Version 4.5.5 – Handling logging of Environment Variables IMAServer, IMAPort and SIPList. Additional bug fixes for when shutting down. Addition of new environment variables for Latency Histograms (ConnHistSize and MsgHistSize). Provide warning message if latency is greater than histograms. | Nov 30, 2017 |
| 1.26 | Correction on ClkSrc Environment Variable. | Dec 11, 2017 |
| 1.27 | Removal of rate controller thread, removal of –rrs and additional info on –sseq. Fix for linux select( ) call with file descriptor > 1024 (select limitation). | Jan 24, 2018 |
| 1.28 | Cleanup and further explanation of the –sseq option. | Jan 28, 2018 |
| 1.29 | Update version dates | Feb 7, 2018 |
| 1.30 | Version 4.5.8 - Bug fix for –burst mode ; Performance enhancements | Feb 27, 2018 |
| 1.31 | Version 4.5.9 – Bug fix for locks | Mar 20, 2018 |
| 1.32 | Version 4.6.0 – Fixes | Apr 11, 2018 |
| 1.33 | Version 4.6.1 – Bug fix for no Source IP. Updated jitter to support multiple QoS’ when using a client list. | Apr 19, 2018 |
| 1.34 | Version 4.6.2 – Bug fix for reconnecting and cleaning up of the message ids. Update limitations pertaining to restarting mqttbench while using QoS 2. | May 2, 2018 |
| 1.35 | Version 4.6.3 – Cleanup of some code. | June 1, 2018 |
| 1.36 | Version 4.6.4 – Bug fixes for jitter function. | June 21, 2018 |
| 1.37 | Version 4.6.5 – Fix for shutting down mqttbench. | July 5 2018 |
| 1.38 | Correct page numbering on document | July 10, 2018 |
| 1.39 | Version 5.0 – Support for MQTT V5 Protocol | Sept 24, 2018 |
| 1.40 | Version 5.0.6 – Bug fixes for Messages Inflight | Oct 1, 2018 |
| 1.41 | Version 5.0.7 – Pickup fix for openssl shutdown. | Oct 2, 2018 |
| 1.42 | Version 5.0.7 – Help page restructure and SSL connection fix | Oct 11, 2018 |
| 1.43 | Version 5.0.8 – Fix for Latency calculations & final statistics | Oct 12, 2018 |
| 1.44 | Version 5.0.9 – Remove restriction for latency with only publishers. | Oct 18, 2018 |
| 1.45 | Version 5.1.0 – Beam fix. | Oct 26, 2018 |
| 1.46 | Version 5.1.1 – Update c command – display associated IOP with each connection. | Nov 2, 2018 |
| 1.47 | Version 5.1.2 – Beam fix. Fix for reconnect segfault. | Dec 6, 2018 |
| 1.48 | Version 5.1.3 – Fix for exit segfault and reset timer fix. | Dec 14, 2018 |
| 1.49 | Version 5.1.4 – Fix for QoS2 during reconnects | Jan 4, 2018 |
| 1.50 | Version 5.1.5 – Fix for TLS Connections and SNI. | Jan 11, 2019 |
| 1.51 | Version 5.1.6 – Update copyright for 2019 | Feb 6, 2019 |
| 1.52 | Version 5.2.0 – Use OpenSSL v1.1.1 and libicucu 60.2 | Feb 14, 2019 |
| 1.53 | Version 5.2.1 – Additional logging information | Mar 25, 2019 |
| 1.54 | Version 5.2.2 – Ability to re-resolve DNS on every reconnect. | June 6, 2019 |
| 1.55 | Updated the new message file format | June 11, 2019 |
| 1.56 | Version 5.2.3 – Pick up openssl v1.1.1c | June 17, 2019 |

**TABLE OF CONTENTS**

[Introduction 7](#_Toc11661142)

[Features 7](#_Toc11661143)

[Limitations 7](#_Toc11661144)

[Performance Report 8](#_Toc11661145)

[Version Information 8](#_Toc11661146)

[Tested environments and system requirements 9](#_Toc11661147)

[Installing mqttbench 9](#_Toc11661148)

[Client machine tuning 11](#_Toc11661149)

[Kernel tuning parameters 11](#_Toc11661150)

[NIC tuning parameters 12](#_Toc11661151)

[User and process limits 12](#_Toc11661152)

[OS services stopped 12](#_Toc11661153)

[Environment Variables 13](#_Toc11661154)

[Setting environment variables for mqttbench 15](#_Toc11661155)

[mqttbench command line options 18](#_Toc11661156)

[Client List Details 21](#_Toc11661157)

[Client List Format 21](#_Toc11661158)

[Client List Examples 22](#_Toc11661159)

[Available Commands on Interactive Command Line 25](#_Toc11661160)

[Examples of commonly used commands on the interactive command line: 27](#_Toc11661161)

[Command Line Statistics 32](#_Toc11661162)

[Connection Statistics: 33](#_Toc11661163)

[Totals: 33](#_Toc11661164)

[Mapping of MQTT Message types to QoS Levels 34](#_Toc11661165)

[Message/QoS Rates 34](#_Toc11661166)

[Message Counts 35](#_Toc11661167)

[Statistics via –snap parameter 36](#_Toc11661168)

[CSV file format for –snap parameter 38](#_Toc11661169)

[Connection Information (connects): 38](#_Toc11661170)

[Message Rates (msgrates) 38](#_Toc11661171)

[Message Counts (msgcounts) 39](#_Toc11661172)

[Connection Latency (connlatency) 40](#_Toc11661173)

[Round Trip Message Latency (msglatency) 40](#_Toc11661174)

[Reconnect Round Trip Message Latency 41](#_Toc11661175)

[Instructions on running MQTT performance tests 42](#_Toc11661176)

[Appendix A: MQTT V5 Protocol Support 43](#_Toc11661177)

[Command Line Parameters no longer supported 43](#_Toc11661178)

[Environment Variables no longer supported 43](#_Toc11661179)

[Updates 43](#_Toc11661180)

[Message File Format Examples 44](#_Toc11661181)

[Assigned Client ID 45](#_Toc11661182)

[Topic Aliases 45](#_Toc11661183)

[Message Order Checking 45](#_Toc11661184)

[MQTT Messaging Enhancements 46](#_Toc11661185)

[CONNECT Packet Options 46](#_Toc11661186)

[CONNACK Processing 46](#_Toc11661187)

[DISCONNECT from the Server 46](#_Toc11661188)

[SUBSCRIBE Packet 46](#_Toc11661189)

# Introduction

**mqttbench** is a multi-threaded C program used for performing MQTT v3.1/v3.1.1/v5.0 throughput and latency benchmark tests. It can simulate 1 million MQTT v3.1/v3.1.1/v5.0 devices. **mqttbench** is a benchmark application, and as such does little to no processing of the messages which it sends or receives.

The following is a list of features and limitations of **mqttbench**:

## Features

* Supports multiple QoS levels: 0, 1, and 2
* Supports pub/sub messaging
* Supports secure communications over TLS/SSL security (OpenSSL v1.1.1)
* Configurable number of destination IPv4 address and port numbers
* Supports the clean session(v3.1/v3.1.1), clean start (v5.0) and retain flags.
* Configurable size window for messages inflight
* Configurable buffer sizes
* Configurable user names and passwords.
* Configurable number of submitter, I/O processing, and I/O listener threads
* Configurable reconnect delay in the event of a disconnect.
* Provide message rate variance information when performing throughput testing.
* Subscribe/transmit messages to multiple servers in a single process.
* Support a client list that contains client configuration details for publish and subscribe. The client list is in a Json format (v5).
* Can measure connection, subscription, and publish/receive RTT latency with or without predefined messages (v5).
* Supports TLS Connections using Client Certificate and Private Key.
* Multiple Message Directories to provide various size messages per client.
* Support of SSL Buffer Pool handled internally.
* Ability to cause a burst of messages for a period of time.
* Ability to perform a trace on a specified client (v5).
* Supports message order checking (v5).
* Support of user properties
* Send a predefined message to the topic that is defined in the message.

## Limitations

* No processing of message data.
* Does not currently support request/response interactions
* Does not currently support the PINGREQ/PINGRES message and KEEPALIVE is configurable, but defaults to 64K seconds.
* Client state is persisted only in memory, not to disk. For QoS 2 messaging when stopping and restarting an instance of mqttbench, this may result in message ids being reused and potentially being disconnected by the server.

## Performance Report

mqttbench is used to conduct the performance tests for the MessageSight Appliance Performance report which can be found at the following URL:

<http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24035590>

## Version Information

|  |  |
| --- | --- |
| **Version** | **Date** |
|  |  |
| 4.6.0 | 04/11/18 |
| 4.6.1 | 04/19/18 |
| 4.6.2 | 05/02/18 |
| 4.6.3 | 06/01/18 |
| 4.6.4 | 06/21/18 |
| 4.6.5 | 07/05/18 |
| 5.0 | 08/12/18 |
| 5.0.2 | 09/04/18 |
| 5.0.3 | 09/10/18 |
| 5.0.4 | 09/11/18 |
| 5.0.5 | 09/20/18 |
| 5.0.6 | 10/01/18 |
| 5.0.7 | 10/10/18 |
| 5.0.8 | 10/12/18 |
| 5.0.9 | 10/18/18 |
| 5.1.0 | 10/26/18 |
| 5.1.1 | 11/02/18 |
| 5.1.2 | 12/06/18 |
| 5.1.3 | 12/14/18 |
| 5.1.4 | 01/04/19 |
| 5.1.5 | 01/11/19 |
| 5.1.6 | 02/06/19 |
| 5.2.0 | 02/14/19 |
| 5.2.1 | 03/25/19 |
| 5.2.2 | 06/06/19 |
| 5.2.3 | 06/17/19 |

# Tested environments and system requirements

* CentOS/RHEL 7.x x86\_64
* net.ipv4.ip\_local\_port\_range should be set to 1024 - 4999 in kernel params. This allows 60000 connections per IP address. In order to achieve 1M outgoing TCP connections you will need 17 IP addresses on the machine running mqttbench. The IP addresses of the destination MQTT broker(s), must be reachable from all source IP addresses used by mqttbench.
* Stable TSC clock for latency measurements. The clock used in latency measurements is the TSC. The frequency of the clock is read at startup of so clock frequency cannot change. CPU frequency scaling features should be disabled at the OS and uEFI level to prevent clock frequency changes.
* By default mqttbench will create 1 I/O listener thread and N number of I/O processor threads (based on the number of CPU available and online on the system running mqttbench). The submitter threads use busy wait rate control for a high degree of accuracy, but this also requires a dedicated CPU for submitter threads. Make sure there are enough CPUs for the number of I/O listener, I/O processor, and submitter threads.
* In order to utilize the mqttbench command line history, the **libedit** RPM must be installed.
* Library dependencies
  + from the MessageSight/MessageGateway Build:
    - libismutil.so
    - libicuuc.so
    - libicudata.so
    - libicuil18n.so
    - OpenSSL v1.1.1: libssl.so, libcrypto.so
  + from the OS:
    - libedit.so (optional)

# Installing mqttbench

1. Create the directory INSTALL\_HOME (e.g. mkdir /opt/ibm/ims)
2. Create the directory <INSTALL\_HOME>/bin (e.g. mkdir /opt/ibm/ims/bin)
3. Copy the mqttbench binary to <INSTALL\_HOME>/bin
4. Create the directory <INSTALL\_HOME>/lib64 (e.g. mkdir /opt/ibm/ims/lib64)
5. Copy the libismutil.so library to <INSTALL\_HOME>/lib64.
6. Copy the libicu\*.so libraries to <INSTALL\_HOME>/lib64
7. Set the LD\_LIBRARY\_PATH and CLASSPATH environment variables

export LD\_LIBRARY\_PATH=<INSTALL\_HOME>/lib64:$LD\_LIBRARY\_PATH

export PATH=<INSTALL\_HOME>/bin:$PATH

# Client machine tuning

## Kernel tuning parameters

The following kernel parameters were set on the client machines for these benchmarks.

|  |  |
| --- | --- |
| **Name** | **Value** |
| fs.nr\_open | 2000000 |
| net.core.wmem\_max | 65536 |
| net.core.rmem\_max | 8388608 |
| net.core.wmem\_default | 16384 |
| net.core.rmem\_default | 16384 |
| net.core.netdev\_max\_backlog | 2097152 |
| net.ipv4.ip\_local\_port\_range | 1024 4999 |
| net.ipv4.tcp\_wmem | 2048 16384 65536 |
| net.ipv4.tcp\_rmem | 4096 16384 8388608 |
| net.ipv4.tcp\_mem | 65536 8388608 16777216 |
| net.ipv4.tcp\_tw\_reuse | 1 |
| net.ipv4.tcp\_timestamps | 1 |
| net.ipv4.tcp\_window\_scaling | 1 |
| net.ipv4.tcp\_sack | 1 |
| net.ipv4.tcp\_synack\_retries | 10 |
| net.ipv4.tcp\_keepalive\_intvl | 15 |
| net.ipv4.tcp\_keepalive\_probes | 5 |
| net.ipv4.tcp\_fin\_timeout | 15 |
| kernel.threads-max | 2000000 |
| kernel.pid\_max | 2000000 |
| vm.nr\_hugepages | 500 |
| vm.max\_map\_count | 4000000 |

**Notes:**

1. The **sysctl** command can be used with the **–w** option to set these options.

Format: **sysctl -w [setting name]=[value]**

1. An alternative is to put the settings in the file: **/etc/sysctl.conf**

Format: **[setting name] = [value]**

## NIC tuning parameters

The following NIC tuning parameters were set on the client machines, which contained Mellanox ConnectX-3 NICs, for the performance benchmarks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | | **Value** | **Ethtool option** |
| RX ring buffers | | 8192 | -G |
| TX ring buffers | | 8192 | -G |
| txqueuelen | | 40000 | N/A |
| MTU | | 1500 | N/A |
| Device Active MTU | | 1024 | N/A |
| Interrupt coalescence | adaptive-rx | off | -C |
| rx-usecs | 10 | -C |
| rx-frames | 0 | -C |
| tx-usecs | 0 | -C |
| tx-frames | 0 | -C |

**Note:** Use the **ethtool** utility to set all the above (except for txqueuelen, MTU and Device Active MTU). Use **ifconfig** command to set MTU and txqueuelen.

## User and process limits

The following user and process limits were set on the client machines for the performance benchmarks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | | **Value** | **ulimit**  **option** |
| nofile | | 2000000 | -n |
| stack | soft | 4096 | -s |
| hard | 32768 |  |
| nproc | | unlimited | -u |
| core | | unlimited | -c |
| memlock | | unlimited | -l |
| rtprio | | 100 | -r |

**Note:** Use the **ulimit** command to apply the settings above.

## OS services stopped

Before running any benchmarks the machine used to run mqttbench should be properly tuned. This means stopping any unnecessary processes and OS services. Run OS at runlevel 3. All firewall related services should be disabled and iptables/netfilter kernel modules unloaded.

# Environment Variables

| **Variable** | **P/C** | **Default** | **Value** | **Description** |
| --- | --- | --- | --- | --- |
|  | | | | |
| **General** | | | | |
| Affinity\_<thread> | C,P | No affinity | hex mask | Allows you to set thread level CPU affinity for various mqttbench threads. This is advanced configuration, do not attempt unless you know what you are doing.  E.g. **export Affinity\_iop1=0x1** will pin iop1 to CPU 0 |
| BatchingDelay | P | 0 | >= 0 | Provide the ability to perform batching. The value specified is in microseconds. Setting the value > 0 will increase the chances of multiple messages being sent in a network write.  **Note:** Setting this value too high will have an effect on throughput. |
| DelayCount | C,P | 0 | >= 0 | Number of connections to initiate between delays |
| DelayTime | C,P | 0 | >= 0 | Time (microseconds) between connections which is based on the value of DelayCount. |
| IOProcThreads | C,P | 3 | 1 to 32 | Number of IO Processor Threads to be used. This should be less than the number of cores available on the system, for example NRCORE – 3 |
| IOListenerThreads | C,P | 1 | > 0 | Number of IO Listener Threads to be used. The default should be sufficient. |
| MBAutoLog | C,P | 0 | 0 or 1 | Will copy the current log and latency file and append a numerical suffix every 24 hrs if set to 1. |
| MemTrimInterval | C,P | Disabled | > 0 | Number of seconds between calls to Linux malloc\_trim( ) which is to release free memory from the top of the heap. |
| MqttbenchLogPath | C,P | mqttbench\_trace.log | !(NULL) | Fully qualified path for mqttbench trace log file (default: mqttbench\_trace.log in the current directory). |
| MQTTKeepAlive | C,P | 0xFFFF | 0 to 0xFFFF | MQTT Keep Alive Timer which is specified in the MQTT CONNECT message. |
| PipeCommands | C,P | 0 | >= 0 | Enable commands to be passed to mqttbench via a pipe when running for a specified test duration (-d <secs>). mqttbench will open and receive commands from the named pipe: **/tmp/mqttbench-np** |
| RetryDelay | C,P | 0 | > 0 | The initial time (usecs) to delay TCP reconnect per client. Dependent on RetryBackoff being set. |
| RetryBackoff | C,P | 0 | > 0 | The Retry Back-off factor used to increase the delay time between each TCP reconnect. Dependent on RetryDelay being set. It is a decaying interval, i.e. the delay between reconnect attempt grows by a factor derived from RetryBackoff |
| RetryMaxAttempts | C,P | 0 | > 0 | The maximum number of retry attempts prior to shutting down the connection. Used in conjunction with RetryDelay and RetryBackoff. |
| SIPList | C,P | NULL | (IPv4 ipadr) | List of Source IPv4 addresses used to connect to a Destination IP Address specified in a client list.    SIPList=”sourceIP1 sourceIP2...”  For example:  SIPList=”169.55.138.111 169.55.138.112…” |
| SourcePortLo | C,P | 5000 | 1025 to  65535 | Specifies the lowest source port used with each IP address when connecting with the IMA Server. mqttbench explicitly binds the source port and src interface for each connection.  **Note:** This provides a way to utilize multiple instances of mqttbench on a single machine. |
| UseNagle | C,P | 0 | 0 or 1 | Enable/Disable Nagle's Algorithm. To disable (0) means setting the socket option with: TCP\_NODELAY |
| UseEphemeralPorts | C,P | 0 (Off) | 0 or 1 | Specifies whether the clients should explicitly bind to a source port or allow the OS ephemeral port selection algorithm to choose the source port. If UseEphemeralPorts is enabled then update the net.ipv4.ip\_local\_port\_range kernel parameter to 5000-65000 in order to increase the ephemeral port range |
|  | | | | |
| **Buffer Related** | | | | |
| BuffersPerReq | C,P | 320 | > 0 | # of TX buffers to get from a TX buffer pool per request. This is a tuning parameter which normally doesn’t need to be changed. |
| MaxIOPTXBuffers | C,P | 32K | > 0 | Maximum # of TX buffers per IOP thread. The size of each TX buffer is derived from the size of the largest message that will be sent (i.e. the –s <min>-<max> command line parameter |
| RecvBufferSize | C,P | 16K | > 0 | Size of the Receive Buffer. |
| RecvSockBuffer | C,P | 32K | > 0 | Size of Receive Socket Buffer. |
| SendBufferSize | C,P | 16K | > 0 | Size of the Send Buffer. |
| SendSockBuffer | C,P | 16K | > 0 | Size of Send Socket Buffer. |
|  | | | | |
| **SSL Related** | | | | |
| SSLCipher | C,P | RSA | **openssl ciphers** | The SSL Cipher to be used. Run **openssl ciphers** for a list of possible ciphers for this environment variable. |
| SSLClientMeth | C,P | TLSv12 | SSLv2  SSLv3  TLSv1  TLSv11  TLSv12 | The SSL Method the clients will use for making connections to the MessageSight/MessageGateway Appliance. This is only applicable when UseSecureConn env var is set (i.e. 1 or 2). |
| TLSBufferPoolSize | C,P | 50 | > 0 | The number of buffers to use for various size buffers for using internal managed SSL Buffer Pools. |
| UseTLSBfrPool | C,P | 0 | 0 or 1 | Use the internal SSL Buffer Pool. 0=Disabled, 1=Enabled |
|  | | | | |
| **Latency Related** | | | | |
| ClockSrc | C,P | 0 (TSC) | 0=TSC  1=GTOD | Defines which Clock Source to use:  0 = TSC – Time Stamp Counter (Intel Based Chips)  1 = GTOD – Get Time Of Day clock |
| ConnHistSize | C,P | 100000 | > 0 | Number of entries in the Connection Latency histograms (default: 100000). The incremental difference between entries is dependent on the Connection Time Unit command line Option (-cu). |
| MsgHistSize | C,P | 100000 | > 0 | Number of entries in the Message Latency histograms (default: 100000). The incremental difference between entries is dependent on the Time Unit Option (-u). |
| SampleRate | C,P | 1000 | >= 0 | Rate at which to sample messages when performing latency testing. Only applicable when –T option is specified. |
| DisableLatencyWarn | C,P | 0 | 0 or 1 | Disable the 30 min timer which checks and provides warning messages if the latency is greater than the specific latency type (Message or Connection) Histogram. |
|  |  |  |  |  |
| **Graphite Related** | | | | |
| GraphiteIP | C,P | NULL | (IPv4 ipadr) | IPv4 address of the Graphite server to send metrics to. |
| GraphitePort | C,P | 2003 | 1 to 65534 | Port number that the Graphite server is listening to. |
| GraphiteMetricRoot | C,P | loadtests | (string) | Root metric path |
|  |  |  |  |  |

## Setting environment variables for mqttbench

* OPTIONAL: A space separated list of source IPv4 addresses on the local system on which mqttbench is running. This list is used by mqttbench to bind to when connecting to the MQTT broker(s). By default mqttbench will discover all local IPv4 addresses and attempts to identify the list of IPv4 addresses from which the MQTT broker(s) are reachable.

**export SIPList=**< list of Source/Client IP Addresses >

**Note:** The list is a space separated list of Source IP addresses used to connect to a Destination IP (e.g. "169.45.138.100 169.45.138.101").

* OPTIONAL: By default mqttbench will bind source ports starting from **SourcePortLo (5000).** By enabling **UseEphemeralPorts** it instructs mqttbench to allow the operating system to select the source port from the ephemeral port range.

**export UseEphemeralPorts=1**

* OPTIONAL: Set the starting source port number for clients to use when creating connections. This is ignored if the **UseEphermeralPorts**  is enabled.

**export SourcePortLo=**<starting port #>

* OPTIONAL: Set the sizes of the send/receive buffers and send/receive socket buffers:

**export RecvSockBuffer=**< receive socket buffer >

**export SendSockBuffer=**< send socket buffer >

**export RecvBufferSize=**< size of buffer to read when reading from network >

**export SendBufferSize=**< size of buffer used to send data to network (single

buffer per client transport) >

The defaults are optimal in most cases.

* OPTIONAL: Set the IOP maximum number of buffers for transmitting (TX) and number of buffers to obtain per request:

**export MaxIOPTXBuffers=**< # of buffers >

**export BuffersPerRequest=**< # of buffers >

* For secure connection tests:

**export SSLClientMeth=**< SSL client method >

**export SSLCipher=**< cipher to use >

**Note:**  Need to create your own certificates.

* OPTIONAL: Set the number of I/O Processor Threads responsible for sending and receiving data. By default mqttbench determines the optimal number of IOP threads to create based on the CPU count:

**export IOProcThreads=**< # of I/O processor threads >

* OPTIONAL: Set the CPU affinity mask for each I/O Processor Thread to a CPU (Affinity\_iopX - where X is the I/O Processor thread number [0-N]=CPU mask):

**export Affinity\_iop**[x]=[cpu mask]

For example: set iop0, iop1 and iop2 threads on CPU 0, CPU 1 and CPU 3 respectively:

**export Affinity\_iop0=0x1**  
**export Affinity\_iop1=0x2**  
**export Affinity\_iop2=0x8**

* OPTIONAL: Set the CPU affinity mask of the Submitter thread (Affinity\_sub[x] where x is the submitter thread number):

**export Affinity\_sub**[x]=[cpu mask]

For example: set sub0 & sub1 threads on CPU 8 & CPU 9 respectively

**export Affinity\_sub0=0x200**

**export Affinity\_sub1=0x400**

* Set connection delay for large connections, by default there is no delay between client connections.

**export DelayCount=**< # of delays >

**export DelayTime=**< # of usecs >

For example, for a 1M TLS connection test a **DelayTime** of 150 microseconds might be a good place to start.

* OPTIONAL: Set the size of the Histograms used for storing message latency data and/or connection latency. Each histogram entry is 4 bytes in size.

**export MsgHistSize=**< size > Message Latency Histogram size

**export ConnHistSize=**≤size> Connection Latency Histogram size

For example: If size=10000, then, requires 40,000 bytes to create histogram.

* OPTIONAL: Set the maximum number of retry attempts prior to disconnection the connection.

**export RetryMaxAttempts=**< number >

* OPTIONAL: Specifying the time to keep the connection alive on the server:

**export MQTTKeepAlive=**< seconds >

* OPTIONAL: Specify the clock source to be user for Round Trip latency:

**export ClockSrc=**<source>

* OPTIONAL: Set the time (RetryDelay) and multiply factor (RetryBackoff) for TCP reconnect per client. TCP reconnects will be delayed longer over time and asynchronous to allow other connections to establish a connection.

**export RetryDelay=**< usecs >

**export RetryBackoff=**< factor >

For example: If RetryDelay=2 and RetryBackoff factor=1.01, then the subsequent delays will be:

|  |  |
| --- | --- |
| Attempt | Delay Time  (usecs) |
| 1 | 2.0 |
| 2 | 2.02 |
| 3 | 2.04 |
| 4 | 2.061 |

**Note:** These environment variables should be set if stopping and starting tests relatively quickly (< 1 minute). Without setting these variables with the TCP connection in **TIME\_WAIT** state, and if there is a failure on the initial attempt, mqttbench will attempt to reconnect a particular connection 10,000 times and display an error message before attempting to do any other connections.

* OPTIONAL: Enable Nagle's Algorithm which is to NOT to set the socket option TCP\_NODELAY

**export UseNagle=**1 default is 0 (Disabled)

* OPTIONAL: Enable log file auto-renaming every 24 hrs for both the –csv and –lcsv parameters

**export MBAutoLog=**1. default is 0 (Disabled)

This renames the current –csv and –lcsv file with a numerical suffix every 24 hrs starting with ‘0’. The intention is to use this when performing extended runs with **-snap** parameter.

After the 1st 24 hrs if using the default files:

mqttbench.csv -> mqttbench\_0.csv

latency.csv -> latency\_0.csv

* OPTIONAL: Enable recapturing memory from the heap every x seconds, which is used to try and improve memory usage. This is used if trying to determine if there is a memory leak.

**export MemTrimInterval=** 360 default is 0 (Disabled)

This results in the Linux malloc\_trim() being called every 360 secs (6 mins).

# mqttbench command line options

mqttbench -cl <clientlist> [-crange <x1-x2,y1-y2,z1-z2>]

[-d <seconds> | -c <numMsgs>] [-r msgs/sec>]

[-s <min>-<max> | -M <directory>] [-mim <maxInflight>]

[-st <numThrds>] [-i <ID>] [-tl <tracelevel>] [--clientTrace <regex>]

[-nodnscache ] [-noreconn] [-t <mode>] [-T <mask>] [-cu <units>]

[-ru <units>] [-u <units>] [-lcsv <filename>] [-rl <seconds>]

[--snap <opts> <seconds>] [-csv <filename>] [--burst <int,dur,rate>]

[-rr <numMsgs>] [-b <seconds>] [-l <seconds>] [-as <loops>] [-sseq]

[-nw] [-quit <type>] [-V <mask] [-dci] [-env] [-psk <filename>]

[-jitter <msecs> <count>] [-v] [--help]

In the table below (C) indicates a parameter applicable to consumers/subscribers and (P) indicates a parameter applicable to producers/publishers.

| **Option** | **C/P** | **Default** | **Val** | **Description** |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Common (Consumer and Producer)** | | | | |
| -cl <filename> | C,P | Not enabled | N/A | The name of a client information file. The file (json format) is a list of MQTT clients to be loaded by mqttbench. Each line in the file represents a single MQTT client. The number of clients from the client to start is determined by either   1. The number of entries in the client list. 2. The –crange <x1-x2,y1-y2,z1-z2> command line param.   See the [Client List Details](#_Client_List_Details) section for more in-depth explanation. |
| --clientTrace  <regex> | C,P | Not enabled | N/A | Match all client IDs with the provided regular expression <regex>, and enable low level trace on these clients to assist in debugging. Put quotes around the regex to prevent evaluation by the bash interpreter. |
| -crange <x1-x2> | C,P | Not enabled | [x1]  [x2] | Select a range of clients by specifying the line numbers in a client list (-cl [filename]) to be used for this particular run.  **Note:** The file line numbers is **NOT** 0 based (i.e. line numbers should start at 1).  Multiple ranges can be specified by separating by a ‘,’  -crange <x1-x2,y1-y2,z1-z2,…etc>  For example:  -crange <1-2,5-7,11-15>  **Note:** Multiple ranges can **NOT** overlap (e.g. … 5-11,6-9…..) |
| -csv <filename> | C,P | See description | [fn] | The <filename> of the csv file to write statistical data to. Default file name is: mqttbench\_stats.csv |
| -d <secs> | C,P | 0 | -1  0  >0 | Test duration. A value 0 means run forever. Duration (-d) and message count (-c) are mutually exclusive. -d takes precedence over -c.  Run mode when value:  0 – Provides an interactive command line console with mqttbench.  -1 – mqttbench will close after completing the task at hand |
| -dci | C,P | Not enabled | N/A | Print to the file: mb\_ClientInfo.txt all the topics per client and whether subscription or publication topics. |
| -env | C,P | Not enabled | N/A | Display the current environment variable values, and command line parameters. |
| --help | C,P | Not Applicable | N/A | Provide a help screen with the command line parameters and a subset of environment variables available. |
| -i <ID> | C,P | 0 | > 0 | The ID for the instance of mqttbench. This is used for message sequence checking. |
| -jitter <msecs>  <count> | C,P | Not enabled | [msecs]  [count] | The number of milliseconds (msecs) between each sample of throughput. The count is the number of previous entries to keep prior to wrapping. |
| -nodnscache | C,P | Not enabled | N/A | Will perform a resolve the DNS every time on a reconnect. |
| -noreconn | C,P | 0 | N/A | When an error occurs the connection will be disconnected and will not attempt to reconnect. Default is to attempt a reconnect. |
| -p <secs> | C,P | Not enabled | > 0 | Send a PINGREQ every x seconds to server.    **Notes: 1.** Should be used with testing over internet.  **2. Currently not supported.** |
| --snap <opts>  <secs> | C,P | Not enabled | [opts]  [> 0] | Print statistical information to a file at an interval specified by <secs>. The statistical information can be a combination of connections, message rates and message counts. The latency information can be a combination of message latency and connection latency. The supported <opts> with description are:  connects – Connection States  msgrates – Message rates for publisher & subscriber  msgcounts – Message counts (MQTT messages with QoS levels.  msglatency – Message Latency (RTT)  connlatency – Connection Latency  The default for interval <secs> is 5 seconds.  **Note:** The snap options for msglatency and connlatency require the corresponding –T option to be specified:  0x1 – Message Latency  0x20 – Connection Latency |
| -sseq | C,P | Not enabled | N/A | Perform sequential sending/subscription on topics. By default all sends and subscribes are processed client breadth first (client1->topic1, client2->topic1, client3->topic1, client1->topic2…etc). If this parameter is specified then sends and subscribes are processed client depth first (client1->topic1, client1->topic2, client1->topic3, client2->topic1….etc). |
| -st <numThrds> | C,P | 1 | > 0 | Specify how many submitter threads to be used in this instance. The clients will be divided up based on the [numThrds] specified. |
| -tl <tracelevel> | C,P | 5 | 0 to 9 | Trace Level setting which is logged to the mqttbench trace log file. Supported values:  0 = No Tracing  1 = Trace Events which normally occur only once. Events which happen at init, start and stop.  5 = Frequent actions which do not commonly occur at a per-message rate.  7 = Trace events which commonly occur on a per-message basis.  8 = Call/Return from internal functions.  9 = Detailed trace used to diagnose a problem. |
| -v | C,P | Not Applicable | N/A | This provides the current version of mqttbench. |
|  | | | | |
| **Consumer Specific** | | | | |
| -as <loops> | C | Not enabled | > 0 | Loop in message processing to simulate application message processing of a message. Current implementation is a sched\_yield() for each loop. |
| -l <secs> | C | 0 | > 0 | The time to keep a connection alive before unsubscribing and/or disconnecting. The action to take after the linger time expires is determined by the value of -quit parameter. |
| -quit <type> | C | 0 | 0  1 | Indicates how the application should terminate the clients:  0 - Perform MQTT UNSUBSCRIBE & MQTT DISCONNECT and close  connection.  1 - Perform MQTT DISCONNECT and close connection. |
| -V <mask> | C | 0 | 0x1  0x2 | Enable message checking. Mask values:  0x1: Enable message length check.  0x2: Enable message sequence number checking. |
|  | | | | |
| **Producer Specific** | | | | |
| --burst  <int,dur,rate> | P | Not enabled | [> 0],  [> 0], [> 0] | Perform a periodic message rate change for a specified duration and repeating interval.  <int,dur,rate>    where:  int - The interval (seconds) between rate changes.  dur - The duration (seconds) for the rate change  rate – The message rate (messages/sec) during  duration. |
| -c <numMsgs> | P | Not enabled | > 0 | Publish a global count <numMsgs> of messages, and then stop publishing. This can be used with or without rate control. |
| -M <directory> | P | Not enabled | [dir] | Use predefined messages. Specifies a <directory> containing \*.json files. Each json file contains the payload of one message. The files are sorted alphabetically. |
| -mim <maxInflight> | P | 65535 | 0 to  65535 | Maximum Inflight Messages per connection (Only for QoS 1 & 2) |
| -r <msgs/sec> | P | 1000 | > 0 | Global message rate (msgs/sec) across all producer threads. If the test is configured for 3 producers and -r 3000 is provided, then each producer will send 1000 msgs/sec. |
| -s <min>-<max> | P | 32-32 | > 32 | The range of messages size to be used in this test. The units for min and max are bytes/message. When <min> == <max>, there is only one message size created. The range of messages is used to generate a binary progression of message size from <min> to <max>. These messages will be sent in round robin order per destination. For example, 32-128 will generate/send 3 message sizes: 32, 64, and 128. |
| -t <mode> | P | 0 | -2  -1  0 | Modes:  -2 - no rate control with random client send distribution.  -1 - no rate control, which sends as fast as possible  0 - each individual producer thread controls rate  (default) |
|  | | | | |
| **Latency Related** | | | | |
| -b <secs> | C,P | Not enabled | > 0 | Begin collecting statistics after <secs> seconds. |
| -cu <units> | C,P | 1e-3 | > 0 | Units used for connection latency measurements. Default is milliseconds. |
| -lcsv <filename> | C,P | See description | [fn] | The <filename> of the csv file to write latency results to. Default file name is: mqttbench\_latstats.csv |
| -rl <numSecs> | C,P | Not enabled | > 60 | Reset the latency information (histograms and variables) every <numSecs> seconds. The minimum value is 60 seconds. |
| -rr <numMsgs> | C,P | Not enabled | > 0 | Option to reset the statistics after <numMsgs> messages received by consumers. |
| -ru <units> | C,P | 1e-3 | > 0 | Units used for Reconnect Message latency measurements. Default is milliseconds. |
| -T <mask> | C,P | Not enabled | See  ====> | Enable latency measurements:  0x1: enable RTT latency measurements.  0x2: enable latency measurement of the send call (prod  only)  0x4: enable latency measurement of the tcp/ssl/ws  connection  0x8: enable latency measurement of the mqtt connection.  0x10: enable latency measurement of the subscription.  0x20: enable latency measurement of the tcp – mqtt  connection (includes WebSocket if –ws is specified).  0x40: enable latency measurement of the tcp – subscription  (includes WebSocket if –ws is specified).  0x80: enable the printing of latency histograms  **Note**s:   1. Printing the histograms can be specified in combination with any other value (e.g. 0x90 would print the histograms for the subscription latency). 2. If RTT latency is specified then tcp/ssl/ws/mqtt connection and subscription are disabled. |
| -u <units> | C,P | 1e-6 | > 0 | Units used for message latency measurements. Default is microseconds. |
|  | | | | |

# Client List Details

Helper Python scripts, for migrating legacy client list files to the new JSON format, are found in the **clientlists** subdirectory of the mqttbench package. Also included are sample Python scripts which provides examples of how to create client list files in the new JSON format.

The command to migrate old client list files (default output file name is mqttbenchv5-cl.json):

**python migrateOldClientList.py --oldcl legacyList.cl**

The command to create a sample using the new client list file format:

**python newClientList\_Ex1.py > newClientList.json**

The full list of client configuration properties supported by mqttbench can be found in the **mqttbenchObjs.py** helper script, located in the **clientlists** subdirectory.

## Client List Format

The mqttbench client list file is formatted as JSON.

The format for each topic that is published (TX) in the list is as follows:

“publishTopics”: [{“qos”: <0,1 or 2>,

“retain”: <true|false>,

“topicStr”: “<topic string>”}]

The format for each topic that is subscribed (RX) in the list is as follows:

“subscriptions”: [{“qos”: <0,1 or 2>,

“topicStr”: “<topic string>”}]

Topics also support a special variable **${COUNT}** to create additional topics which differ by a counting field:

**${COUNT:**<startNum>**-**<endNum>**}**

**For example**:

“topicStr”: topica${COUNT:5-9}

which results in the following topics:

topica5, topica6, topica7, topica8, topica9

LingerInfo has been changed to “lingerTimeSecs” in the json format:

“lingerTimeSecs”: <x>,

where: **x** is the amount of time to stay connected in seconds

The Client Certificate and Client Private Key are fully qualified paths with file names that will be used for the secured connection.

## Client List Examples

**Producer (MQTT V5 Protocol):**

{“\_id”: “d:client\_d:1”,

“cleanStart”: true,

“clientCertPath”: “/etc/pki/tls/certs/abc/cert-client-d-dt0.pem”,

“clientKeyPath”: “/etc/pki/tls/certs/abc/key-client-d-dt0.pem”,

“dst”: “test\_server\_1.sl.com”,

“dstPort”: 17901,

“lingerTimeSecs”: 0,

“publishTopics”: [{“qos”: 1,

“retain”: false,

“topicStr”: “iot-2/evt/evt1/topic2”},

{“qos”: 2,

“retain”: false,

“topicStr”: “iot-2/topic3”}],

“reconnectDelayUSecs”: 0,

“sessionExpiryInterval”: 4294967295,

“useTLS”: true,

“username”: “fred”,

“password”: “pswd\_test”,

“version”: “TCP5”},

Client d:client\_d:1 will make a secured (TLS) connection to the destination IP ‘test\_server\_1.sl.com’ on port ‘17901’ with a client certificate ‘cert-client-d-dt0.pem’ and client key ‘key-client-d-dt0.pem’, cleanStart ‘true’, and Session Expiry Interval of 4294967295 seconds. The username and password credentials are ‘fred’ and ‘pswd\_test’ respectively. The lingerTime and reconnectDelay are both set to 0.

Published messages on 2 topics:

Topic 1: iot-2/evt/evt1/topic2 with QoS 1 and retain off

Topic 2: iot-2/topic3 with QoS 2 and retain off

**Subscriber (MQTT V5 Protocol):**

{“\_id”: “A:client\_c:2”,

“cleanStart”: true,

“dst”: “test\_server\_1.sl.com”,

“dstPort”: 18484,

“lingerTimeSecs”: 15,

“subscriptions”: [{“qos”: 0,

“retain”: false,

“topicStr”: “/iot-2/evt/evt1/topic1”},

{“qos”: 1,

“retain”: false,

“topicStr”: “/iot-2/evt/evt3/topic3”}],

“reconnectDelayUSecs”: 4200,

“useTLS”: true,

“username”: “user2”,

“password”: “pass2”,

“version”: “TCP5”},

Client ID A:client\_c:2 will make a secured (TLS) connection to the destination IP ‘test\_server\_1.sl.com’ on port ‘18484’, and cleanStart ‘true’. The username and password credentials are ‘user2’ and ‘pass2’ respectively. The lingerTime is set to 15 secs and reconnectDelay is set to 4200 usecs.

Subscribes to 2 topics:

Topic 1: iot-2/evt/evt1/topic1 with QoS 0 and retain off

Topic 2: iot-2/evt/evt3/topic3 with QoS 1 and retain off

**Subscriber (MQTT V3.1.1 Protocol):**

{“\_id”: “A:client\_c:3”,

“cleanSession”: true,

“dst”: “test\_server\_1.sl.com”,

“dstPort”: 8484,

“lingerTimeSecs”: 0,

“subscriptions”: [{“qos”: 1,

“retain”: false,

“topicStr”: “/iot-2/evt/evt1/topic1”}],

“reconnectDelayUSecs”: 2000,

“useTLS”: false,

“username”: “user3”,

“password”: “pass3”,

“version”: “TCP4”},

Client ID A:client\_c:3 will make a non-secured connection to the destination IP ‘test\_server\_1.sl.com’ on port ‘8484’, and cleanSession ‘true’. The username and password credentials are ‘user3’ and ‘pass3’ respectively. The lingerTime is set to 0 secs and reconnectDelay is set to 2000 usecs.

Subscribes to 1 topic: iot-2/evt/evt1/topic1 with QoS 1 and retain off

**Publisher & Subscriber (V5 Protocol):**

{"\_id": "d:test12345\_qwerty\_1”,

"cleanStart": true,

"dst": "messaging.testtrial.com",

"dstPort": 17003,

"lingerTimeSecs": 60,

"publishTopics": [{"qos": 0,

"retain": false,

"topicStr": "iot-2/type/test/car1”}],

"reconnectDelayUSecs": 0,

"subscriptions": [{"qos": 0,

"topicStr": "iot-2/type/test/+/topic1"},

{"qos": 0,

"topicStr": "iot-2/type/LoadTest/fmt/json”}],

"useTLS": true,

"username": "use-pietin-auth",

"password": "Appl3P1e”,

"version": "TCP5"},

Client ID d:test12345\_qwerty\_1 will make a secured (TLS) connection to the destination ‘messaging.testtrial.com’ on port ‘17003’, and cleanStart ‘true’. The username and password credentials are ‘use-pietin-auth’ and ‘Appl3P1e’ respectively. The lingerTime is set to 60 secs and reconnectDelay is set to 0 usecs.

Publishes to 1 topic: iot-2/type/test/carl with QoS 0 and retain off

Subscribes to 2 topics with QoS 0:

Topic 1: iot-2/type/+/topic1

Topic 2: iot-2/type/LoadTest/fmt/json

# Available Commands on Interactive Command Line

The following is a list of commands available on the command line if mqttbench is invoked with the duration command line option (-d) is set to zero: **-d 0**

help(h)

close

quit(q)

stat | [Enter Key]

al(l)

env

freemem

nt

rate=[new rate]

resetlat

uptime

ver

c \*

c [clientID]

c [clientID] [dump]

c [clientID [topic]]

c [clientID [topic]] [dump]

badclient [count]

clienttrace [regex] <enable|disable>

findsid [sid]

findtopic [topic]

lse

sidchk [mincount]

tl [new level]

lstate

pstate [state]

tstate [state]

m [state]

jitter [options]

where options are:

[msecs] [count]

reset

report

settings

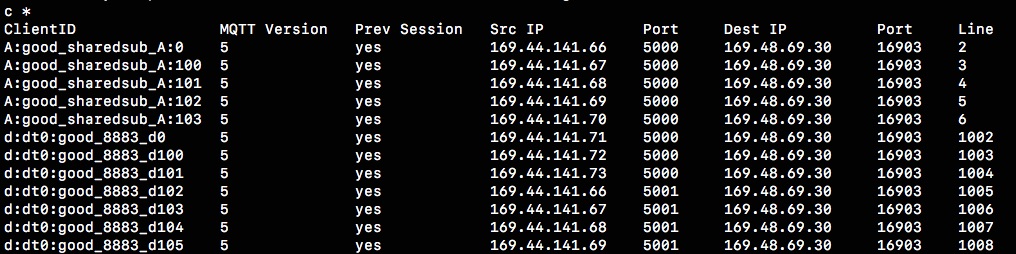
slowacker [#acks]

The table provides more detail about the commands listed above.

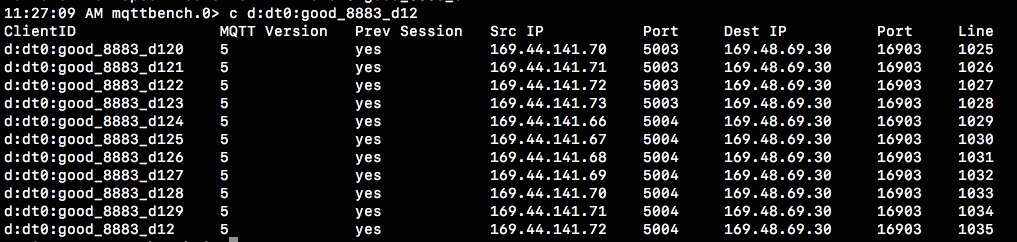
| **Command** | **Description** |
| --- | --- |
|  | |
| close | Close the connections without unsubscribing (consumers) |
| quit(q) | Quit the process which includes unsubscribing all topics. |
| stat **OR [Enter Key]** | Provide the current statistics of the connections (consumers/producers) as well as the IOPs |
| al(l) | Get the latency |
| env | Provide the current environment settings that were used as invocation. |
| freemem | Provide the current amount of free memory. |
| nt | Provide the total number of topics. |
| rate=[new rate] | Provide the publishing rate (messages/seconds), which is an aggregate. |
| resetlat | Reset the latency statistics. |
| uptime | Provide the current runtime. |
| ver | Provide the version string for mqttbench |
| c \*  c [clientID]    c [clientID] [dump]  c [clientID [topic]]  c [clientID [topic]]  [dump] | Provide client connection information (Source IP Address & Port, Destination IP Address & Port) for all connections. |
| Provide client connection information for all connections that match the substring specified in clientID. |
| Provide client state information (Transport State, Client State, MQTT Message counts, and number of messages/topic) for all connections that match the substring specified in clientID. |
| Provide client connection information for all connections that match the substrings specified in the clientID **AND** the topic. |
| Provide client state information for all connections that match the substrings specified in the clientID **AND** the topic. |
| badclient [count] | Provide clients which have at least count of non-zero MQTT return code occurrences. |
| clientTrace [regex]  <enable|disable> | Enable **OR** Disable client tracing. |
| findsid [sid] | Provide the client ID of the publisher that owns the message stream identified by stream ID [sid] specified. |
| Findtopic [topic] | Provide the client ID of the published or subscribers that publish/subscribe to the topic specified. |
| lse | Provide the last x number of Socket Erros per IOP (max of 10) |
| sidchk [mincount] | Provide the message streams which have at least a mincount of redelivered or out-of-order messages. |
| tl [new level] | Set a new trace level (valid range: 0 – 9) |
| lstate | Provide the supported Transport and Client Protocol (MQTT) states. |
| pstate [state] | Provide all the clientIDs in specified protocol state. |
| tstate [state] | Provide all the clientIDs in specified transport state. |
| m [state] | **Provide the current connections in the state specified. Currently only supporting CONNACK** |
| jitter [options] | Provides the average, min and maximum rate (receive or send). |
| jitter [msecs] [count} | The interval in msecs and the number of historical entries to use for the jitter information. |
| jitter [reset] | Reset/clear the current set of data. |
| jitter [report] | Provide the current throughput jitter |
| jitter [settings] | Provide the current settings for the interval (msecs) and number of historical entries (count). |
| slowacker [#acks] | Provide the connections that are slow ack’g based on supplied #acks |
|  | |

# Examples of commonly used commands on the interactive command line:

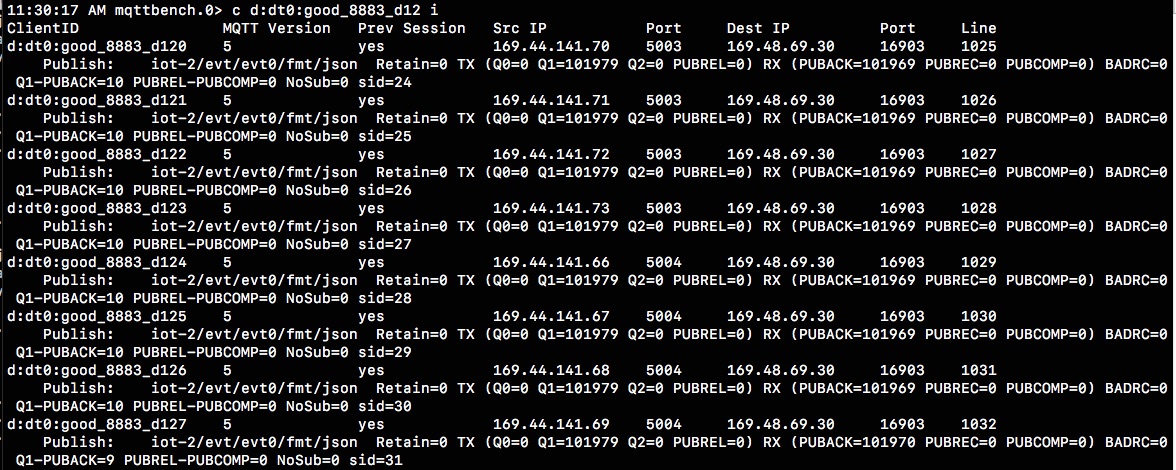
*c \**



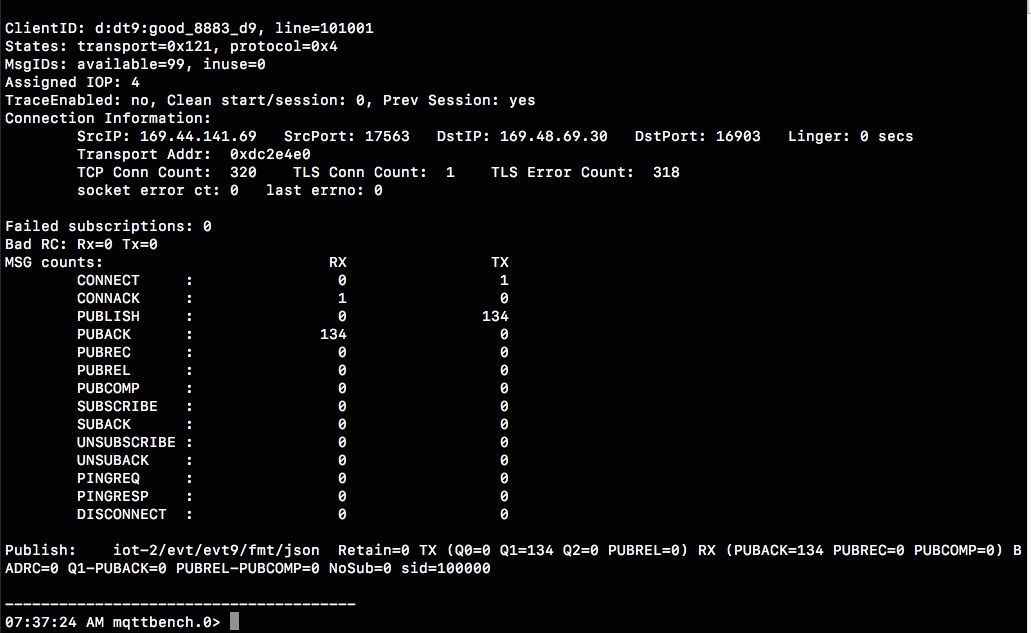
*c [clientid]*

**

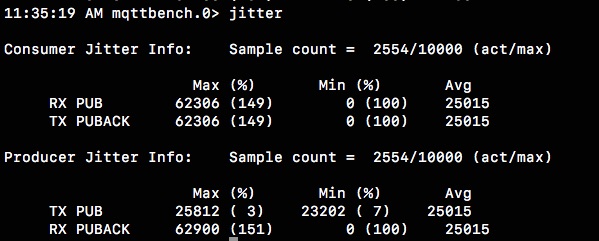
*c [clientid [topic]]*

**

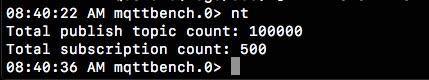
*c [clientid] [dump]*



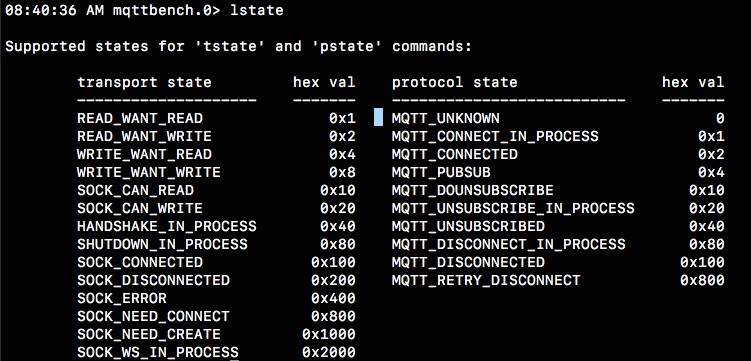
*jitter results*



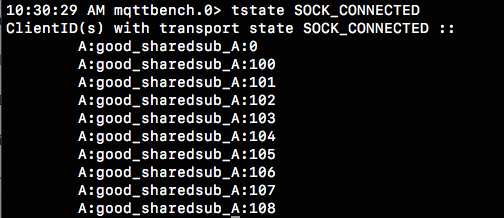
*nt*



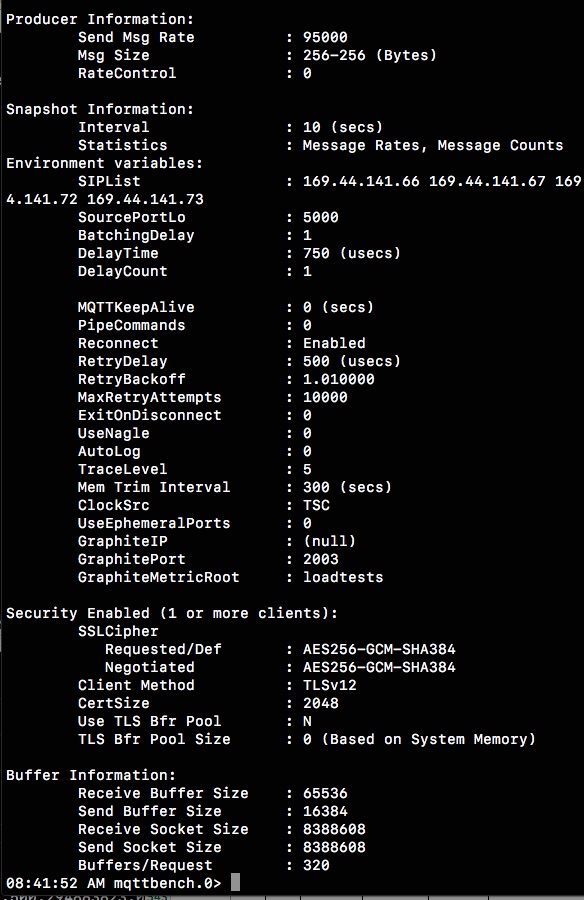
*lstate*



*tstate*



*env*



*help*

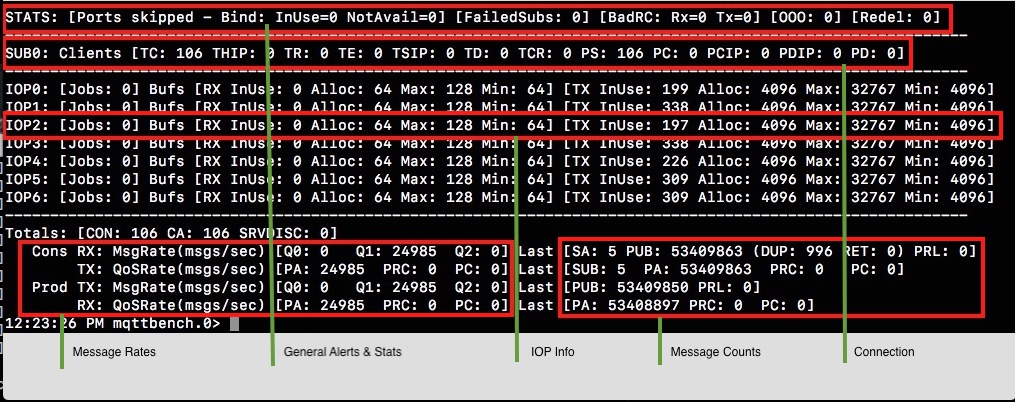
**

# Command Line Statistics

Command line statistics are provided when entering “stat” or simply hitting the [ENTER] key on the mqttbench command line. The mqttbench interactive command line is available only when running with Test Duration (-d) set to: 0. The console will display the following statistics in discrete sections highlighted in the red boxes below:

Top to bottom as they appear in the screenshot below

* General Alerts and Statistics,
* Connection states. One line per submitter thread
* IOP Jobs & Buffer Use (mostly for internal use).
* Message Rates for producers and consumers (aggregation of all IOP threads).
* Message Counts for producers and consumers (aggregation of all IOP threads).



## Connection Statistics:

Connection statistics fall into two categories: states and accumulative:

* **States :** These statistics represent the current states of the connections. The state of a connection will progress from TD (“disconnected”) all the way to PS (“mqtt pubsub ready”) and back (when a disconnect occurs). If a reconnect occurs the states will change till the connection is re-established.
* **Accumulative :** These statistics are cumulative and represent the running total number of the event type

|  |  |  |
| --- | --- | --- |
| **Label** | **Description** | **Type** |
| TC | TCP/IP Connection Complete | States |
| THIP | TCP/IP Handshake In Process | States |
| TR | Total # of TCP/IP Connection Retries | Accumulative |
| TE | TCP/IP Sock Errors | States |
| TSIP | TCP/IP Shutdown In Process | States |
| TD | TCP/IP Disconnected | States |
| TCR | TCP/IP Sock Need Create | States |
| PS | MQTT PUBSUB Ready | States |
| PC | MQTT CONNECT’d | States |
| PCIP | MQTT CONNECT In Process | States |
| PDIP | MQTT DISCONNECT In Process | States |
| PD | MQTT DISCONNECT’d | States |

## Totals:

The totals represents is the accumulation of MQTT CONNECT, MQTT CONNACK and Server Disconnects.

|  |  |  |
| --- | --- | --- |
| **Label** | **Description** | **Type** |
| CON | Total # of MQTT CONNECT Messages Sent | Accumulative |
| CA | Total # of MQTT CONNACK Messages Received | Accumulative |
| SRVDISC | Total # of Server Disconnects | Accumulative |

## Mapping of MQTT Message types to QoS Levels

|  |  |
| --- | --- |
| **Message Type** | **QoS Level** |
| PUBLISH | 1, 2 or 3 |
| PUBACK | 1 |
| PUBREC | 2 |
| PUBREL | 2 |
| PUBCOMP | 2 |

## Message/QoS Rates

All message rates are in units of messages/second.

Publisher (TX):

|  |  |  |
| --- | --- | --- |
| **Rate Type** | **Label** | **Description** |
| Message | Q0 | Sent MQTT PUBLISH - QoS 0 |
| Q1 | Sent MQTT PUBLISH - QoS 1 |
| Q2 | Sent MQTT PUBLISH - QoS 2 |
| QoS Message Type | PA | Received MQTT PUBACK |
| PRC | Received MQTT PUBREC |
| PC | Received MQTT PUBCOMP |

Consumer/Subscriber (RX):

|  |  |  |
| --- | --- | --- |
| **Rate Type** | **Label** | **Description** |
| Message | Q0 | Received MQTT PUBLISH - QoS 0 |
| Q1 | Received MQTT PUBLISH - QoS 1 |
| Q2 | Received MQTT PUBLISH - QoS 2 |
| QoS Message  Type | PA | Sent MQTT PUBACK |
| PRC | Sent MQTT PUBREC |
| PC | Sent MQTT PUBCOMP |

## Message Counts

Publisher (TX):

|  |  |
| --- | --- |
| **Label** | **Description** |
| PUB | # of MQTT PUBLISH messages sent |
| PRL | # of MQTT PUBREL messages sent |
| PA | # of MQTT PUBACK messages received |
| PRC | # of MQTT PURREC messages received |
| PC | # of MQTT PUBCOMP messages received |

Consumer/Subscriber (RX)

|  |  |
| --- | --- |
| **Label** | **Description** |
| SA | # of MQTT SUBACK messages received |
| PUB | # of MQTT PUBLISH messages received |
| PRL | # of MQTT PUBREL messages received |
| SUB | # of MQTT SUBSCRIBE messages sent |
| PA | # of MQTT PUBACK messages sent |
| PRC | # of MQTT PUBREC messages sent |
| PC | # of MQTT PUBCOMP messages sent |

**IMPORTANT:**

1. When making subscriptions, multiple subscriptions will be batched into a MQTT SUBSCRIBE message. Hence, the number of SUBSCRIBE (SUB) messages will not reflect the number of topics subscribed to.
2. The number of SUBACK (SA) messages should match the number of SUBSCRIBE (SUB) messages, NOT the number of subscriptions.

# Statistics via –snap parameter

--snap <comma separated list of options> [interval]

The -**–snap** command line parameter can be used to report various statistics (connections, message rates and/or message counts) and latency (connection and/or message) into csv files. The interval is in units of seconds. If the interval is not specified then the default is 5 seconds.

The Statistical data and Latency data are written to separate csv files:

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Parameter** | **Default** |
| Statistical | **-csv** <filename> | mqttbench\_stats.csv |
| Latency | **-lcsv** <filename> | mqttbench\_latstats.csv |

The following options are supported:

|  |  |
| --- | --- |
| **Options** | **Description** |
| connects | Connection Status |
| msgrates | Message Rates (msgs/sec) |
| msgcounts | MQTT Message Counts |
| msglatency | Message Latency (RTT) |
| connlatency | Connection Latency |

Specifying connection (**connlatency**) and/or message **(msglatency**) latency allows the data to be recorded, but latency must be enabled to capture the data. This is accomplished using the following parameters:

**-T [mask]** - specifies the type of latency to capture

**-cu [units]** - specifies the units to measure connection latency

**-ru[units]**  - specifies the units to measure reconnect message latency

**-u [units]**  - specifies the units to measure message latency

**Note:** If the latency data is 0’s, then the units for the particular latency requested will probably need to be adjusted.

The format of the data captured is always specified in the following orders:

**Statistical**:

|  |  |  |
| --- | --- | --- |
| Connection Info | Message Rates | Message Counts |

**Latency:**

|  |  |
| --- | --- |
| Connection | Message |

Examples:

Collect message rates and message counts statistics at a 10 second interval. Stats are written to **mqttbench\_stats.csv** in the working directory.

**-snap msgrates,msgcounts 10**

|  |  |
| --- | --- |
| Message Rates | Message Counts |

Collect connection and message rates statistics at a 5 second interval. Stats are written to **tmp.csv** in the working directory.

**-snap msgrates,connects –csv tmp.csv**

|  |  |
| --- | --- |
| Connection Info | Message Rates |

Collect connection and message counts statistics at a 5 second interval. Stats are written to **mqttbench\_stats.csv** in the working directory.

**-snap connects,msgcounts**

|  |  |
| --- | --- |
| Connection Info | Message Counts |

Collect TCP-MQTT Connection latency at 10 second interval and using milliseconds units. Stats are written to **latency.csv** in the working directory.

**-snap connlatency 10 -T 0x20 -cu 1.0e-3 -csv latency.csv**

|  |
| --- |
| Connection |

Collect TCP Connection and message latency at 15 second intervals, using milliseconds units for connection and microseconds for latency. Stats are written to **tmpLatData.csv** in the working directory.

**-snap connlatency 15 -lcsv tmpLatData.csv -T 0x5 -cu 1.0e-3 -u 1.0e-6**

|  |  |
| --- | --- |
| Connection | Message |

1. Collect message rates and counts with message latency at 15 second intervals and using microseconds. Message Rates & Counts stats are written to **stat.csv** and Latency data is written to **testLatency.csv** in the working directory.

**-snap msgrates,msgcounts,msglatency 15 -csv stat.csv -lcsv testLatency.csv -T 0x1 -u 1.0e-6**

stat.csv:

|  |  |
| --- | --- |
| Message Rates | Message Counts |

testLatency.csv

|  |
| --- |
| Message |

# CSV file format for –snap parameter

The tables below describe each column in the csv file output generated by the –snap command line parameter.

## Connection Information (connects):

|  |  |  |
| --- | --- | --- |
| **Header** | **Description** | **Type** |
| TC | TCP/IP Connection Complete | States |
| THIP | TCP/IP Handshake In Process | States |
| TR | TCP/IP Connection Retries | Accumulative |
| TE | TCP/IP Sock Errors | States |
| TSIP | TCP/IP Shutdown In Process | States |
| TD | TCP/IP Disconnected | States |
| TCR | TCP/IP Sock Need Create | States |
| CON | Total # of MQTT CONNECT Messages Sent | Accumulative |
| CA | Total # of MQTT CONNACK Messages Received | Accumulative |
| PS | MQTT PUBSUB Ready | States |
| PC | MQTT CONNECTed | States |
| PCIP | MQTT CONNECT In Process | States |
| PDIP | MQTT DISCONNECT In Process | States |
| PD | MQTT DISCONNECTed | States |
| DISC | Total # of TCP/IP Disconnects | Accumulative |
| SRVDISC | Total # of Server Disconnects | Accumulative |

## Message Rates (msgrates)

Publisher (Tx):

|  |  |
| --- | --- |
| **Header** | **Description** |
| TxQ0 | Publish Message Rate for QoS 0 |
| TxQ1 | Publish Message Rate for QoS 1 |
| TxQ2 | Publish Message Rate for QoS 2 |
| RxRPA | Receive Message Rate for MQTT PUBACK messages |
| RxRPRC | Receive Message Rate for MQTT PUBREC messages |
| RxRPC | Receive Message Rate for MQTT PUBCOMP messages |

Subscriber (Rx):

|  |  |
| --- | --- |
| **Header** | **Description** |
| RxQ0 | Receive Message Rate for QoS 0 |
| RxQ1 | Receive Message Rate for QoS 1 |
| RxQ2 | Receive Message Rate for QoS 2 |
| TxRPA | Send Message Rate for MQTT PUBACK messages |
| TxRPRC | Send Message Rate for MQTT PUBREC messages |
| TxRPC | Send Message Rate for MQTT PUBCOMP messages |

## Message Counts (msgcounts)

Publisher (Tx):

|  |  |
| --- | --- |
| **Header** | **Description** |
| TxPUB | # of MQTT PUBLISH messages sent |
| TxPRL | # of MQTT PUBREL messages sent |
| RxPA | # of MQTT PUBACK messages received |
| RxPRC | # of MQTT PURREC messages received |
| RxPC | # of MQTT PUBCOMP messages received |

Subscriber (Rx)

|  |  |
| --- | --- |
| **Header** | **Description** |
| RxSA | # of MQTT SUBACK messages received |
| RxPUB | # of MQTT PUBLISH messages received |
| RxPRL | # of MQTT PUBREL messages received |
| TxSUB | # of MQTT SUBSCRIBE messages sent |
| TxPA | # of MQTT PUBACK messages sent |
| TxPRC | # of MQTT PUBREC messages sent |
| TxPC | # of MQTT PUBCOMP messages sent |
| BADSubs | # of bad MQTT SUBSCRIBE |
| BADRxRC | # of bad Return Codes on received messages |
| BADTxRC | # of bad Return Codes on messages sent |
| OOO | # of Out Of Order messages received |
| REDEL | # of Redelivered messages |

## Connection Latency (connlatency)

|  |  |
| --- | --- |
| **Header** | **Description** |
| conn\_Units | Units used for measuring latency |
| conn\_#Samples | # of samples received |
| conn\_Hist#Samples | # of samples stored in the histogram |
| conn\_Min(x) | Minimum latency in x units |
| conn\_Max(x) | Maximum latency in x units |
| conn\_Avg(x) | Average latency in x units |
| conn\_StdDev | Standard Deviation |
| conn\_50th | 50% |
| conn\_75th | 75% |
| conn\_90th | 90% |
| conn\_99th | 99% |
| conn\_1Sec | # of latency objects that are greater than 1 second |
| conn\_5Sec | # of latency objects that are greater than 5 seconds |

## Round Trip Message Latency (msglatency)

|  |  |
| --- | --- |
| **Header** | **Description** |
| msg\_Units | Units used for measuring latency |
| msg\_#Samples | # of samples received |
| msg\_Hist#Samples | # of samples stored in the histogram |
| msg\_Min(x) | Minimum latency in x units |
| msg\_Max(x) | Maximum latency in x units |
| msg\_Avg(x) | Average latency in x units |
| msg\_StdDev | Standard Deviation |
| msg\_50th | 50% |
| msg\_75th | 75% |
| msg\_90th | 90% |
| msg\_99th | 99% |
| msg\_1Sec | # of latency objects that are greater than 1 second |
| msg\_5Sec | # of latency objects that are greater than 5 seconds |
| msg\_MsgSizes | Range of message sizes in bytes |

## Reconnect Round Trip Message Latency

This is the message latency due to a reconnect occurring prior to receiving message(s).

|  |  |
| --- | --- |
| **Header** | **Description** |
| rmsg\_Units | Units used for measuring latency |
| rmsg\_#Samples | # of samples received |
| rmsg\_Hist#Samples | # of samples stored in the histogram |
| rmsg\_Min(x) | Minimum latency in x units |
| rmsg\_Max(x) | Maximum latency in x units |
| rmsg\_Avg(x) | Average latency in x units |
| rmsg\_StdDev | Standard Deviation |
| rmsg\_50th | 50% |
| rmsg\_75th | 75% |
| rmsg\_90th | 90% |
| rmsg\_99th | 99% |
| rmsg\_1Sec | # of latency objects that are greater than 1 second |
| rmsg\_5Sec | # of latency objects that are greater than 5 seconds |
| rmsg\_MsgSizes | Range of message sizes in bytes |

# Instructions on running MQTT performance tests

It is important before running any test, make sure “iptables” is turned off... iptables can severely limit the packet rate that kernel can handle.

To check iptables status, type in the following command: **lsmod | grep ip\_tables**

If either **iptables** or **ip6tables** appears from running the previous command then need to turn it off by running the following commands as ROOT:

**service network restart**

**service iptables stop**

**service ip6tables stop**

**rmmod ipt\_REJECT**

# Appendix A: MQTT V5 Protocol Support

## Command Line Parameters no longer supported

* Parameters used to create patterns of topics and subscriptions. In v2 the client list file is a **required** parameter. The user of mqttbench can generate all the messaging patterns possible with the parameters below, using a client list file, so these parameters are redundant and have been removed.
  + -rx <rxparams>, -tx <txparams>
  + -np
  + -rrs
  + -pfxi
  + -sm
* -ws, the websocket flag can be specified in the client list file
* -proto, pluggable protocol testing support has been removed

## Environment Variables no longer supported

The following environment variables are used for setting destination IP, Port, TLS, but can all be specified in the client list file and so have been removed:

* + IMAServerSets
  + IMAServer
  + IMAPort
  + UseSecureConn

## Updates

* The client list format is now JSON.
* Message File Format:
  + Previously message files were ASCII text files, but now are JSON documents.
  + Support for the message file containing the topic on which to publish the message file. If multiple topics defined then messages will be published in round robin fashion.
  + Support for User Properties which provides the ability to obtain latency when using message files.
  + Support for the following MQTT v3.1.1. properties:
    - QoS
    - Retain Flag
  + Support for the new MQTT v5 properties;:
    - Payload Format
    - Content Type
    - Message Expiry Interval
    - Response Topic
    - Correlation Data

### Message File Format Examples

There are 3 formats of message files which are illustrated below:

Example 1 – only payload size defined:

{

"payloadFormat": 0,

"payloadSizeBytes": 1024

}

**Note:** Payload will be filled with 1024 bytes of blanks.

Example 2 - payload in text format with topic

{

"contentType": "text/plain",

"payload": "All work no play makes Jack a dull boy.",

"payloadFormat": 1,

"topic": {

"noLocal": true,

"qos": 1,

"retain": true,

"retainHandling": 1,

"topicStr": "my/pubtopic/tree",

"userProperties": []

}

}

Example 3 - payload in json format without topic:

{

"contentType": "application/json",

“expiryIntervalSecs”: 3600,

"payload": {

"sensor1": "OK",

"sensor2": 40

},

"payloadFormat": 1,

"userProperties": [

{

"name": "a-name",

"value": "a-value"

},

{

"name": "b-name",

"value": "b-value"

}

]

}

Example 4 – json document defines payloadfile to be used for payload:

{

"payloadFile”: “/opt/e2e-test/messages/tiny/msg32byte.dat”

}

msg32byte.dat payload contents can be ASCII text or JSON:

{

“phrase”: “Hello World!”

}

**OR**

“The quick brown fox jumps over the lazy dog”

## Assigned Client ID

The client ID maybe be sent to the server as a zero length string, which results in the server creating a unique ID. The client list file allows the ID field to be an empty string for MQTT v5 (or later) clients. Clients created by mqttbench with a zero length client ID will always reconnect with a zero length client ID string (i.e. will require the server to generate a new client ID on every reconnect).

## Topic Aliases

Topic aliases can be used in place of the full topic string to reduce the size of MQTT PUBLISH packet overhead. The client list supports a **topicAliasMax** field for a client object in order to set the maximum number of topic aliases allowed for each client. If the server sends a topic alias greater than **topicAliasMax,** it is a protocol error and mqttbench will send a MQTT DISCONNECT message and close the connection.

## Message Order Checking

Message order checking is enabled by using the **-V** command line parameter.

**-V 0x2 -ssid**<starting sid>

where: starting sid is a non-negative integer.

**Note:**  mqttbench will assign a unique stream id (**sid**) for each client.

## MQTT Messaging Enhancements

### CONNECT Packet Options

* **Max Packet Size** - The server MUST not send mqttbench any packets larger than that specified by the **MaxPacketSize** field in the MQTT CONNECT packet.
* **Topic Aliases** - Specify whether a Topic Alias will be used in place of the Topic.

### CONNACK Processing

* **Assigned Client ID** (field 0x12) - The server may assign a Client ID for the corresponding client that issued the CONNECT message. If this occurs, mqttbench will set the client ID accordingly for that MQTT Client.
* **Server Keep Alive**  (field 0x13) - The server may set a Mandatory KeepAlive time, which requires the client MUST use this value for its’ KeepAlive Time.
* **Maximum Packet Size** (field 0x27) - The sever may set a Maximum Packet Size it is willing to accept, which requires published messages by mqttbench to be no larger than the value specified. If the end user invokes mqttbench with a message size greater than the Server’s Max Packet size the CONNACK Reason Code will indicate if the packets have exceeded the size the sever specified.
* **Receive Maximum – aka flow control** (field 0x21) – The server uses this to limit the number of QoS 1 & 2 messages the client can send. mqttbench currently has the ability to control the number of inflight messages to the server via the **-mim** command line option.
* **Reason Codes**  - If there is a problem performing the MQTT CONNECT the server provides additional information by setting a value in this field. If the value is >= 0x80, then the server will close the connection. mqttbench logs the reason code and handles the disconnect.

### DISCONNECT from the Server

If a reason string is returned in the reason string property or user property of the MQTT DISCONNECT message, mqttbench will log the reason string to the trace log file.

### SUBSCRIBE Packet

The SUBSCRIBE Payload must contain at least 1 Topic and Subscription Options pair. The **Subscription Option**s is defined as:

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit(s)** | **Flag** | **Val** | **Description** |
| 0 & 1 | Maximum QoS | x | Maximum QoS (0, 1 or2) at which the server can send messages to the client. |
|  |  |  |  |
| 2 | No Local | =1 | Messages MUST NOT be forwarded to a connection with ClientID specified. |
|  |  |  |  |
| 3 | Retain As Published | =0  =1 | Messages are forwarded with retain = 0  Messages are forwarded with the retain value that they were published with. |
|  |  |  |  |
| 4 & 5 | Retain Handling | =0  =1  =2 | Send retained messages at the time of the subscribe  Send retained messages at subscribe only if the subscription doesn’t currently exist  Do not send retained messages at time of subscribe. |
|  |  |  |  |
|  |  |  |  |