IBM MQ C Performance Harness

For all updates and feedback, please visit

https://github.com/ibm-messaging/mq-cph/

Table of Contents

Contents

What is the IBM MQ C Performance Harness?

The MQ C Performance Harness (hereafter referred to as 'MQ-CPH'), is a native MQI interface (C/C++), performance test tool, based largely on the function and externals of the "JMSPerfHarness" Performance Harness for Java™ Message Service application <insert link to GITHUB>. It provides a complete set of performance messaging functionality as well as many other features such as throttled operation (a fixed rate and/or number of messages), multiple destinations, live performance reporting. It is one of the many tools used by IBM MQ performance teams for tests ranging from a single client to more than 10,000 clients. Whilst JMSPerfharness can be used to drive workloads on any suitable JMS messaging provider, MQ-CPH is designed to test the native, proprietary IBM MQ interface (MQI).

MQ-CPH will print the current throughput rate on a user-selected periodic basis and also output summary statistics at the end of a test. The included help and documentation provide detailed usage instructions and describe many further features and configuration parameters for investigation.

Using MQ-CPH

As with any tool, this one has many different uses depending on the goals of the user, and can also be thoroughly misdirected to produce useless data. Ensure the performance scenarios you choose to measure bear some relation to the real world. Failure to do so will inevitably lead to incorrect facts, figures, assumptions and decisions. For instance, it is common to see competitive product comparisons being "won conclusively" by using scenarios that mean nothing in real customer environments. It is also worthy of note that performance is usually not the most important factor in any such comparison, it is simply the easiest to create charts from.

Requirements

- IBM/WebSphere MQ (v7+) installation

HOWTO

This section should explain how to get up and running with MQ-CPH. There are many more parameters beyond those discussed here, please use the parameter reference in this doc to see the many additional capabilities.

Building MQ-CPH

Make Pre-requisites

Before running make, make sure the MQ header files are available. For Unix platforms, the makefile will assume they are in $MQ_INSTALLATION_DIR/inc or /opt/mqm/inc (if MQ_INSTALLATION_DIR is not defined). Alternatively you can set the 'INCLUDE' environment variable to a specific directory.

On Windows, the include directory will be something like "C:\Program Files\IBM\WebSphere MQ\Tools\c\include"

Build, using the following commands, depending on platform:

Linux

```
export installdir=~/cph
make
```

(On Linux distributions, g++ is required).

AIX

```
export installdir ~/cph
gmake
```

AIX requires the GNU make tool, available here: http://www-03.ibm.com/systems/power/software/aix/linux/toolbox/alpha.html

Windows

On Microsoft Windows, you can use the included VC++ 2005 project.

Debug Build

To enable tracing (-tr command line option) you must compile a debug build of cph. E.g. on Linux run ‘make Debug’. This should only be used for troubleshooting.

Post Make Tasks

After running make, the cph executable will be built in the local directory (./Release/...). If you specified an 'installdir' then the cph executable and the required property files (in the 'props' directory) will have been copied to that directory, otherwise you need to copy them yourself. The cph executable needs to be run from the directory in which it is 'installed' to access the property files, unless the CPH_INSTALLDIR environment variable is set to indicate their location.

How to set the LD_LIBRARY_PATH

MQ-CPH uses MQ client libraries supplied with MQ. You need to tell MQ-CPH where these library files are by setting the LD_LIBRARY_PATH environment variable.

Therefore, before invoking MQ-CPH, ensure the LD_LIBRARY_PATH environment variable is set appropriately.

On Windows use:

```
set LD_LIBRARY_PATH=C:\Program Files\IBM\WebSphere MQ\bin;C:\Program
Files\IBM\WebSphere MQ\bin64;%LD_LIBRARY_PATH%
```

On UNIX and Linux platforms use:

```
export LD_LIBRARY_PATH=/opt/mqm/lib:/opt/mqm/lib64:$LD_LIBRARY_PATH
```

How to use the built in help

MQ-CPH is a very modular tool, and certain modules need to be selected via the command-line for different modes of operation. At any time, using "-h" will print help on the current context (i.e. the currently loaded modules).

Parameter	Description
-h	At any time, using "-h" will print help on the current context (i.e. the currently active modules). It will not give help on modules that are not active.

-hf	This performs the same as "-h", but prints additional details on parameters and includes other, seldom used, parameters
-hm	This gives the full help for a named module, regardless of whether it is active or not. You do not need to pass the full module name as the tool will search intelligently for the named module. Example: -hm Receiver or -hm Responder

How to choose your test module

The tool's operation is defined by the test module being run and there are many selections of test module. Each of the following modules may provide a few additional options to fine tune behaviour. More details can be found in the previous section.

Parameter	Description
-tc Sender	Sends messages to a named queue destination.
-tc Receiver	Receives messages from a named queue destination. This can be used in conjunction with the Sender module.
-tc PutGet	Sends a message to queue then retrieves the same message (using CorrelationId). This is the default setting.
-tc ReconnectTimer	Special version of PutGet, with reconnect timer logic.
-tc RequesterReconnectTimer	Special version of Requester, with reconnect timer logic.
-tc Requester	Sends a message to a queue then waits for a corresponding reply on a second queue.
-tc Responder	Waits for a message on a queue then replies to it on another queue. This can be used in conjunction with the Requester module.
-tc Publisher	Sends messages to a named topic destination.
-tc Subscriber	subscribes and receives messages from a named topic. This can be used in conjunction with the Requester module.

How to use multiple destinations

The tool will handle multiple destinations (publish-subscribe topics or point-to-point queues) with the right configuration parameters. This allows more complicated scenarios to be constructed across multiple instances of the tool.

The general concept being applied is that an ordered set of destinations are created and then distributed evenly amongst the active threads.

Parameter	Description
-d	Destination prefix. The default is "DEST"
-db	First number in the range.
-+	Last number in the range.
-dn	Number of destinations in the range (or the first destination to use in a fixed range, see example XXX).

The module will infer the set of destinations from the parameters being passed.

Notes:

- These parameters only control the names given to destinations. Specifying "-d TOPIC" does not, in itself, enable publish-subscribe (you could have a queue named TOPIC).
- Each single thread is assigned a single destination.
- It is not considered invalid to specify more destinations than there are threads or to create an uneven balance of destinations amongst threads.

Example invocations

Putting together the lessons from the HOWTO section will give you a basic operational performance tool. The following are some sample invocations of the functionality in this tool, see the command-line reference for the meaning of any unknown parameters:

Point-to-point with IBM MQ

Persistent, transacted point-to-point with local bindings. 10 queue-triplets (each queue has 1 sender and 1 receiver) running on queues (QUEUE1..QUEUE10). The number of triplets can varied arbitrarily. A corresponding test with topics simply requires different test module parameters.

```
export TRIPLETS=10

cph -tc Sender -nt $TRIPLETS -jb QM_RED -jt mqb -d QUEUE -db 1 -pp
-tx

cph -tc Receiver -nt $TRIPLETS -jb QM_RED -jt mqb -d QUEUE -db 1 -
pp -tx
```

Nonpersistent point-to-point with local bindings. Put a million 100-byte messages to a destination QUEUE, then get them back again.

```
export MSGSIZE=100

cph -tc Sender -jb QM_RED -jt mqb -d QUEUE -ms $MSGSIZE -mg
1000000

cph -tc Receiver -jb QM_RED -jt mqb -d QUEUE
```

Publish-subscribe

- Persistent, transacted publish-subscribe with client bindings. 4 publishers and 40 durable subscribers spread evenly across 4 topics (TOPIC1..TOPIC4). Durable subscribers will use the same name (by setting -id) and do not unsubscribe (un=false). This means the subscribing application can be started and stopped without message loss.

```
cph -tc Publisher -nt 4 -jh server1 -jb QM_RED -jt mqc -jp 1415 -d
TOPIC -db 1 -dx 4
cph -tc Subscriber -nt 40 -jh server1 -jb QM_RED -jt mqc -jp 1415 -d
TOPIC -db 1 -dx 4 -du -id SUBS -un false
```

Reconnection testing

You can use the ReconnectTimer module to report on how long it takes for all of your clients to reconnect to a secondary/standby queue manager, after a switch/fail-over scenario in an MIQM or RDQM HA topology.

E.g. If you have two queue managers QM1 (active, on primaryHost), and QM1 (standby, on secondaryHost), with channel definitions defined in a ccdt, then the following command will test how long it takes for 3 threads to re-connect following (e.g.) the issue of endmqm -s QM1 in an MIQM environment.

```
cph -vo 4 -ve 4 -ss 1 -wt 120 -wi 0 -rl 0 -tc ReconnectTimer -to 1 -
co -pp true -tx true -d QUEUE -db 1 -dx 10 -dn 1 -jp 1414 -jb QM1 -jt
mqc -jh primaryQMHost -h2 secondaryQMHost -nt 3
```

Output:

```
Shared library libmqic_r.so loaded ok
Secondary port number: 1414
[ReconnectTimer0] START
[ReconnectTimer0] First session open - entering RUNNING state.
[ReconnectTimer1] START
[ReconnectTimer1] First session open - entering RUNNING state.
[ReconnectTimer2] START
[ReconnectTimer2] First session open - entering RUNNING state.
rate=2461.00,threads=3
rate=2607.00,threads=3
rate=2590.00,threads=3
rate=2594.00,threads=3
rate=2587.00,threads=3

.. .. ← endmqm -s issued here

MQGET Return code caused error or not recognized; mqrc:2009
MQGET Throwing exception; mqcc:2 ;Name: REPLY1
MQGET Return code caused error or not recognized; mqrc:2009
MQGET Throwing exception; mqcc:2 ;Name: REPLY1
[ReconnectTimer0] MQI call failed, attempting to reconnect all
threads to queue manager QM1 on host secondaryHost:
[ReconnectTimer0]: Time to connect to secondary host is 646 ms (min:
646 ms  max: 646 ms)
rate=740.00,threads=3
[ReconnectTimer1]: Time to connect to secondary host is 806 ms (min:
646 ms  max: 806 ms)
[ReconnectTimer2]: Time to connect to secondary host is 806 ms (min:
646 ms  max: 806 ms)
[ReconnectTimer2]: All threads reconnected at 16_04_2020 17:56:20.880
rate=2065.00,threads=3
rate=2133.00,threads=3
rate=2170.00,threads=3
... ...
```

If the QM on the primaryHost is re-started, and the QM on the secondary ended with endmqm -s, then the client will reconnect back to the QM on the primaryHost, and so forth.

Instead of using h2 & (optionally) p2 to specify the secondary host & port, you can use a ccdt to provide the channel definitions of the queue managers:

```
cph -ss 1 -wt 120 -wi 0 -rl 0 -tc ReconnectTimer -to 30 -iq REQUEST -
oq QUEUE -db 1 -dx 10 -dn 1 -jb QM1 -ccdt
file:///mqperf/pharris/AMQCLCHL_QM1_NOSSL.TAB -jt mqc -dq 10 -nt 3
```

Command-line Parameter reference

The system is self-documenting through the command-line. Use -h, -hf and -hm to learn about the functionality.

The following is a snapshot of the parameters of the tool, the latest lists and descriptions are always available using the tools help options:

Config

Centralises parsing, access and reporting of configuration parameters.

Arg	Default	Description
-h	false	Display basic help on current configuration.
-hf	false	Display detailed help on current configuration.
-hm		Display detailed help on a specific module or modules. Specify multiple modules as space-separated tokens. Example: -hm "Requester"
-tr	false	Trace calls. This requires a debug build of cph (make Debug). Trace is written to a file of the form cph_<pid>.txt
-v	false	Show version.

Log

A proxy to output to stdout or stderr. There are currently no extensions to support
external logging, though this could be easily added. There are 5 levels (0-4) of
verbosity defined as (NONE, ERROR, WARNING, INFO, VERBOSE).

Arg	Default	Description
-ve	0	Verbosity below which goes to stderr. The default is such that nothing goes to stderr.
-vo	4	Verbosity below which goes to stdout. The default is such that everything goes to stdout.

ControlThread

Manage the lifecycle of the application and any WorkerThreads. This also controls the aggregation and reporting of performance counters.

Arg	Default	Description
-id		Process identfier. If set, this will be displayed in the statistics reporting. This is of use if you have to merge the output of more than one instance of the tool.
-ls	false	Collect latency stats (reported as avg/min/max micro-seconds) taken to complete one iteration (i.e. execute the "oneIteration" function) of the test in thread 0 (does not include wait time introduced that may be added between iterations by use of the rate parm (rt) on the WorkerThread). Only thread 0 is measured, as a sample. For a Sender test module this would be the latency of n (typically 1) PUT(s), for a Requester module, this would be the latency of a Put and the subsequent Get, combined.
-nt	1	Number of WorkerThreads.
-rl	60	Run length in seconds. Setting this to 0 will disable the timer and run forever.
-sd	normal	Sets what is reported as totalDuration. "normal" = from 1st iteration to last iteration, excluding setup/takedown. "tlf" = Time to Last Fire, from start of main thread till last iteration completes (includes setup time but not takedown)
-sh	true	Use signal handler to trap SIGINT (CTRL-C).
-sp	false	Display per-thread performance data.
-ss	10	Statistics reporting period. Setting this to 0 will disable periodic reporting entirely.
-su	true	Display final summary. This setting is independant of the periodic statistics reporting.
-tc	Dummy	Worker thread implementation to use.
-ts	0	Thread stack size (Kb, Linux only). Setting this to 0 will disable MQ-CPH from setting a per thread stack allocation.
-wi	1000	WorkerThread start interval (ms). This controls the pause between starting multiple threads.
-wt	30	WorkerThread start timeout (s). The maximum number of seconds after starting a worker thread to wait for its status to change to 'running', CPH will fail if any thread takes longer than this time to start. Special case: if set to zero (0), CPH will not wait for each worker thread to start.

WorkerThread

Base module for all varieties of test. This module implements a general acing algorithm for those tests that wish to use it. The performance overhead of this is minimal.

Arg	Default	Description
-mg	0	Fixed number of iterations to run. The default setting of 0 means there is no iteration limit.
-rp	0	Time period (s) to ramp up to the full rate.
-rt	0	Desired rate (operations/sec). If this rate is greater than the maximum achievable, the behaviour is such that it runs as fast as possible. A value of 0 means to always run as fast as possible. Rates of <1 op/sec are not currently possible.
-si	0	Session interval (milliseconds). The number of milliseconds to sleep between closing one session and opening the next. This value is ignored if sn is 1 or mg is 0.
-sn	1	Maximum number of messaging sessions to run. Setting this to 0 means there's no limit to the number of sessions. This value is ignored if mg is 0.
-tc	PutGet	Test definition module. This defines the actual type of WorkerThreads that will be started. Modules include: • PutGet • Sender • Receiver • ReconnectTimer • Requester • RequesterReconnectTimer • Responder • Publisher • Subscriber
-yd	0	Frequency to call Thread.yield(). This may be of use if the WorkerThreads are not being evenly scheduled.

DestinationFactory

This handles destinations for publish-subscribe and point-to-point domains.

These options only control the **names** given to destinations. Specifying
"-d TOPIC" does not enable publish-subscribe ("-tc Publisher -d TOPIC" does that)

Examples:

Parameters Specified	Destinations Used
-d QUEUE	All threads operate on destination named QUEUE
-d MYTOPIC -dn 3	destinations are distributed round-robin in the order MYTOPIC1..MYTOPIC3
-d MYTOPIC -db 6 -dn 3	destinations are distributed round-robin in the order MYTOPIC6..MYTOPIC8
-d MYTOPIC -dx 6 -dn 3	destinations are distributed round-robin in the order MYTOPIC4..MYTOPIC6
-d MYTOPIC -db 4 -dx 6 -dn 5	destinations are distributed round-robin in the order MYTOPIC4..MYTOPIC6 starting with MYTOPIC5

Arg	Default	Description
-d	DEST	Destination prefix. If no other destination parameters are set, then nothing will be appended to this.
-db	0	Multi-destination numeric base.
-dn	0	Multi-destination numeric range.
-dx	0	Multi-destination numeric maximum.
-iq	REQUEST	Put destination prefix. If no other destination parameters are set, then nothing will be appended to this.
-oq	REPLY	Get destination prefix. If no other destination parameters are set, then nothing will be appended to this.

PutGet

Sends a message then receives one from the same queue. Normal usage is with correlation identifier to ensure the same message is received.

Arg	Default	Description
-co	true	Attach a correlId to the message and use it to get the same message back. This allows multiple PutGet clients to work with the same queue concurrently. An effort is made to keep the correlId for each instance unique. This is true by default for historical compatibility.
-cs	false	Use message selectors to get messages off the queue
-gs	UNSPECIFIED	Use generic selector instead of correlId to get messages off REPLY queue
-txp	false	Use transactions for PUTs, (tx must not be set)
-txg	false	Use transactions for GETs, (tx must not be set)

ReconnectTimer

Special version of PutGet, with re-connect timer logic. This can be used to test MIQM or RDQM to see the time it takes to to re-connect to a queue manager. The available queue managers can be specified via a CCDT (in MQOpts), or with jh/h2 & jp/p2 args.

Arg	Default	Description
-co	true	Attach a correlId to the message and use it to get the same message back. This allows multiple PutGet clients to work with the same queue concurrently. An effort is made to keep the correlId for each instance unique. This is true by default for historical compatibility.
-cs	false	Use message selectors to get messages off the queue
-gs	UNSPECIFIED	Use generic selector instead of correlId to get messages off REPLY queue
-h2	secondaryhost	Hostname or IP address of standby QM host machine (primary host specified by jh)

-p2	0	Port of standby QM host machine (default value of 0 will cause p2 to be set to the same value as jp)
-txp	false	Use transactions for PUTs, (tx must not be set)
-txg	false	Use transactions for GETs, (tx must not be set)

Requester

Puts a message to a queue then waits for a reply on another queue.

Arg	Default	Description
-co	true	Get the reply message, using the request message ID as a correlation ID. This allows multiple Requesters to work with the same output queue concurrently. The correlID of the reply is expected to be the same as the automatically-generated message ID of the request (-co must be set on the Responders for this to be enforced). This option is true by default for historical compatibility.
-cs	false	Use message selectors to get messages off the queue
-dq	1	The style of reply-to queue to use. 1 – Value of oq is used only. 2 – Value of oq is prefixed by the QM name (e.g. QM1.REPLY1), Useful in dq scenarios where there are multiple 'client' queue managers.
-gs	UNSPECIFIED	Use generic selector instead of correlId to get messages off REPLY queue
-iq	REQUEST	Queue to place requests on.
-oq	REPLY	Queue to place replies on, set in the ReplyToQ field of the message descriptor. Setting this value to "" implies the use of temporary queues for each reply, correlation-ids are not used in this mode.
-txp	false	Use transactions for PUTs, (tx must not be set)
-txg	false	Use transactions for GETs, (tx must not be set)

RequesterReconnectTimer

Special version of Requester, with reconnect timer logic. This can be used to test MIQM or RDQM to see the time it takes to to re-connect to a queue manager. The available queue managers can be specified via a CCDT (in MQOpts), or with jh/h2 & jp/p2 args.

Reconnect timer is the easier class to use, but this can be useful for distributed queueing re-connect tests where you can start a responder on the remote queue manager, then configure this class with primary and secondary queue managers used to send messages to (optionally more if using a ccdt).

Arg	Default	Description
-co	true	Get the reply message, using the request message ID as a correlation ID. This allows multiple Requesters to work with the same output queue concurrently. The correlID of the reply is expected to be the same as the automatically-generated message ID of the request (-co must be set on the Responders for this to be enforced). This option is true by default for historical compatibility.
-cs	false	Use message selectors to get messages off the queue
-dq	1	The style of reply-to queue to use. 1 – Value of oq is used only. 2 – Value of oq is prefixed by the QM name (e.g. QM1.REPLY1), Useful in dq scenarios where there are multiple 'client' queue managers.
-gs	UNSPECIFIED	Use generic selector instead of correlId to get messages off REPLY queue
-h2	secondaryhost	Hostname or IP address of standby QM host machine (primary host specified by jh)
-iq	REQUEST	Queue to place requests on.
-oq	REPLY	Queue to place replies on, set in the ReplyToQ field of the message descriptor. Setting this value to "" implies the use of temporary queues for each reply, correlation-ids are not used in this mode.
-p2	0	Port of standby QM host machine (default value of 0 will cause p2 to be set to the same value as jp)

-txp	false	Use transactions for PUTs, (tx must not be set)
-txg	false	Use transactions for GETs, (tx must not be set)

Responder

Takes messages off the request queue and places the same message on the reply queue specified in the message descriptor of the message on the request queue (see -oq arg of Requester class above).

Arg	Default	Description
-cb	false	Commit between getting request and putting reply. This option is mainly provided to allow the recovery of the old behaviour of MQ-CPH, whereby, if -tx was specified, an MQCMIT would be done both after calling MQGET on the request and after calling MQPUT or MQPUT1 for the reply on iterations whose sequence-number was a multiple of the commit-count (-cc) option. This option is ignored if -tx is not specified.
-co	true	Use the correlId of the request message (or msgId, if correlId not set) to set the correlId of the reply message. This can be used in conjunction with the same parm on the Requesters, to allow multiple Requesters to work with the same output queue concurrently. This option is true by default.
-cr	true	Copy request message to response. If true, the MessageFactory settings are ignored for replies.
-cs	false	Use message selectors to get messages off the queue
-iq	REQUEST	Queue to place requests on. If no other destination parameters are set, then nothing will be appended to this.
-oq	REPLY	Unused. Replies are always sent to the queue specified in the ReplyToQ field of the message descriptor of the incoming message (see the -oq arg for Requester class above).

MQOpts

Generic options for MQ messaging worker thread modules.

Arg	Default	Description
-an	null	Set ApplName in the MQCNO (displayed as the APPLTAG value in MQSC (DIS CONN(*) ALL). A null value will result in an APPLTAG of 'cph' being set by MQ. You can use this to start different uniform cluster appl groups by starting multiple mq-cph processes each with their own unique APPLTAG used by its threads. **Note:** This option requires that MQ-CPH is built and runs with client libraries of MQ V9.1.2 or later.
-ar	null	Enable client auto-reconnect in the MQCNO. Valid values are: MQCNO_RECONNECT_AS_DEF MQCNO_RECONNECT MQCNO_RECONNECT_DISABLED or MQCNO_RECONNECT_Q_MGR. Any option other than null (the default) or MQCNO_RECONNECT_DISABLED will require a ccdt to be used. Any MQ errors triggered by switching over a queue manager are currently not handled in mq-cph, so may cause threads to stop. Not for use with RecconectTimer or RequesterReconnectTimer. For testing MQ Uniform Cluster set this to MQCNO_RECONNECT and optionally set -an option above.
-cc	1	Commit count (transaction batching). The number of operations completed within a single transaction. This only applies to test modules which only normally perform a single operation (such as Sender or Subscriber). Ignored if tx=false.

-ccdt	Not defined (no ccdt)	MQ Client channel definition table (CCDT) URL. This will be used to set the client channel, if defined. Parms jh, jp, jc and jl will all be ignored if this is specified.
-cv	true	Convert message data. By default, MQGMO.CONVERT is specified in the MQGET options. Specifying false can be useful when reading a message encoded with a different codepage, where the message is non-standard (custom format).
-d		Destination prefix. The first part of the name of an MQ destination object (queue or topic) to put or get messages to, with the second part (a numeric suffix) to be provided by cphDestinationFactory.
-jb	QM	IBM MQ queue manager to connect to.
-jc	SYSTEM.DEF.SVRCONN	IBM MQ Channel to connect to. Ignored unless jt=mqc
-jf	false	Use the fastpath option on the MQCONNX call. Using this option means the application and the local-queue-manager agent are part of the same unit of execution. Using this option should give much higher throughput when using high message rates as it avoids a lot of thread switching. For this reason it is also easier to compare raw publish and subscribe performance. Ignored unless jt=mqb.
-jh	localhost	DNS/IP of provider host machine. Ignored unless jt=mqc
-jl	UNSPECIFIED	Specify which SSL CipherSpec to use. If not using SSL, do not set parameter, the default "UNSPECIFIED" will map to empty string.
-jp	1414	Port to connect to. Ignored unless jt=mqc.

-jt	mqc	IBM MQ transport (mqb, mqc). "mqb" is local-bindings connections, "mqc" is TCP/IP connections.
-jy	false	Use the readahead option on the MQCONNX call. Using this option will enable any getter with the readahead option. Disabling it, defaults to the QM-defined behaviour. Ignored unless jt=mqc.
-mf	null	Fully qualified filespec of a file containing the data to be used in the test message. If this is not specified then data is generated using the -ms, & -mr properties.
-mh	false	Use message handle for message properties.
-mr	false	Randomise data in message (repeats every 57 bytes by default). This does not apply if message is being read from a file (i.e. -mf is set)
-ms	1000	Message size in bytes (if -mf is not specified)
-p1	false	Use Put1 (Open q, put, close q)
-pp	false	Use persistent messages.
-ppn	false	If pp is set to false, then the default persistence setting of the target queue is used to set whether the message is persisted or not. Set to true to force use of non persistent message even if queue property has DEFPSIST set to true. Cannot be set if pp is set to true.
-rf	false	Use RFH2 on message headers. Default value of false means RFH1 header (MQC.MQFMT_RF_HEADER_1) is used. Setting this to true means the read as RFH2 header (MQC.MQFMT_RF_HEADER_2) is used.
-to	5	Polling timeout interval on receiving messages (in seconds). Threads will exit if a timeout occurs. Set this option to -1 to wait indefinitely.

-tx	false	Transactionality. Setting this flag will cause any PUT's and GET's to be inside syncpoint. Note that for classes with a PUT+GET pattern (PutGet & Requester), the txp & txg flags can be used instead, for finer grained control.
-txz	false	Set this to true on z/OS platforms to explicitly request no UOW to be used. Cannot be set if tx is set to true.
-us	UNSPECIFIED	The username to authenticate with when creating a connection to MQ.
-pw	UNSPECIFIED	The password to authenticate with when creating a connection to MQ.
-kr	UNSPECIFIED	The key repository location. This can only be set if a CipherSpec is specified (-jl). If not set, the QM will only look in the default key repository location.
-cz	False	If set to true, the client MsgCompList will be populated with all four compression options (MQCOMPRESS_RLE, MQCOMPRESS_ZLIBFAST, MQCOMPRESS_ZLIBHIGH, MQCOMPRESS_NONE) The specific compression algorithm required can then be selected in the SVRCONN channel definition.

Forwarder

Gets a message off a queue & puts to a different queue.

Arg	Default	Description
-iq	REQUEST	Queue to place requests on.
-oq	REPLY	Queue to place replies on.
-cr	true	Copy input message to output. If true, the MessageFactory settings are ignored for replies.
-cm	true	Copy MD from input message to output.
-cb	false	Commit between getting input and putting output.

Publisher

Send messages to a Topic.

Arg	Default	Description
-tp	true	Use one topic per publisher thread. Set to false to publish to a different topic each iteration.

Subscriber

Subscribe to Topic-domain messages.

Arg	Default	Description
-du	false	Durable subscriptions. Note, if using more than one process, these names will clash. To avoid this, use\n\ the -id parameter to differentiate the processes.
-un	true	Unsubscribe subscribers when closing. Set this false to leave durable subscriptions after the subscription is closed. This is ignored unless du=true.

ReconnectTimer

Special module (based on PutGet above) to time reconnection of threads after an MQ switch/fail-over, in an MIQM or RDQM HA environment. Multiple queue managers (via a ccdt) need to be specified.

Arg	Default	Description
-co	true	Attach a correlId to the message and use it to get the same message back. This allows multiple PutGet clients to work with the same queue concurrently. An effort is made to keep the correlId for each instance unique. This is true by default for historical compatibility.
-cs	false	Use message selectors to get messages off the queue
-gs	UNSPECIFIED	Use generic selector instead of correlId to get messages off REPLY queue
-txp	false	Use transactions for PUTs, (tx must not be set)

-txg	false	Use transactions for GETs, (tx must not be set)

Troubleshooting

Please check the "known issues" section in the release notes.

- **Invalid parameter: Parameter [??] is not known/valid in this configuration**.
  Parameters in this tool belong to specific modules. If that module is not loaded, no knowledge of its parameters exists. If you look at the help for the current context (parameter "-h"), you will see that the corresponding module is not included. Check your "-tc" setting, you are probably not referencing the correct module. A full list of options for these parameters is given in this manual or by parameter "-hf".

- **Invalid parameter: Destination range is negative**.
  You have either set the minimum value ("-db") greater than the maximum ("-dx") or have used a combination of "-dx" and "-dn" which implies a negative starting range. Consult the HOWTO on multiple destinations.

Requesting help

If these few tips do not answer your query, or you have a suggestion for improvements, please ask on the git forum page for this product.
When submitting a problem (particularly a crash report) then please do the following to help us help you:

- Run the tool with "-vo=ALL" to turn on as much debugging output as possible.
- Include the command line invocation you used to run the program.

Acknowledgements

We would like to acknowledge the contribution of Jerry Stevens and various members of the IBM WebSphere MQ Performance Team at IBM Hursley UK.

Feedback

Feedback can be given on the git forum.