Documentation for the OEMPUT program

Documentation for the OEMPUT program
Colin Paice
April 2017
Document Number OEMPUT Version: 1.2

Property of IBM

Take Note!
Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

This edition works with all supported versions of "IBM MQ for z/OS" and "WebSphere MQ for z/OS"
Sending your comments to IBM You can send your comments electronically to idrcf@uk.ibm.com. When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you. You may continue to use the information that you supply.

Document generated Friday 16th June, 2017 at 13:38

Fourth edition, April 2017
This edition name applies to *Documentation for the OEMPUT program Version 1.2* and to all subsequent versions and modifications until otherwise indicated.
Notices
Trademarks and service marks
The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

1. IBM
2. WebSphere MQ
3. DB2
4. z/OS

OEMPUT

OEMPUT is an MQ application program which may prove useful when evaluating the costs and throughput of MQ on z/OS.

At the simplest level OEMPUT can do the following

1. Put messages to a queue
2. Get messages from a queue
3. Put messages to a queue and get messages from the same or a different(a reply-to-queue) queue

Message properties can be specified by parameters passed to the program. These include

1. Message persistence - Persistent or Non Persistent
2. Message size
3. Message attributes, such as the value to be used in the MQMD.ApplIdentityData
4. The number of messages to process before a commit is issued

At the end of a run message rate, elapsed time and CPU usage information is printed.
The program reports Workload Manager Information about the selected jobs.

Changes

In 1.2 of the program

1. -jms options puts a header on the front of the message suitable for JMS programs
2. -IMStransaction you can specify the transaction name for the IMS bridge data

Program syntax

Below is the a typical use of the OEMPUT program.
OEMPUT -mMQPA -qrequestQ -rreplyQ -n1000

If SYSIN dataset is specified then the parameters are read from the file, before the parameters in the JCL PARM statement. This allows you to have common definitions in a file, and override them at run time. Data following a * in the SYSIN file is treated as a comment up to the end of the input line.

Note that there is no space between an option and its associated value, for example
OEMPUT -mMQM1 -qREQ_Q1 -n1000 -s4096

Option descriptions

The options can be specified in upper case or lower case, so -APPLID, -applid, -Applid are all the same.

If a parameter is specified multiple times, the last one is used.

-applid	Specify the name of the applid to be put into the MQMD.ApplIdentityData.
-browse	Use Browse instead of destructive get.
-c	Commit every *msgs_in_loop* messages. Default is no syncpoint. See option -l below.
-cb	Backout the request
-ccsid<nnn>	Set *nnn* as the MQMD.CodedCharSetId to use for the MQPUT of the request message.

-clear	Drain (clear) the reply before starting test. Be careful using this with multiple jobs getting messages from this queue, in case required replies are deleted.
-cgcpc	Commit processing. After initial load of messages do • Get Msgs_in_loop msgs • COMMIT • put Msgs_in_loop msgs • COMMIT This emulates a typical client program. Compare this with option -cgpc below.
-cgpc	Commit processing. After initial put of messages do • get Msgs_in_loop messages • put Msgs_in_loop messages • COMMIT This is typical of a server application. Compare this with option -cgcpc above. -cgpc will usually give a higher throughput as there are less commits.
-convert	Issue an MQGET with convert.
-coop	Use cooperative Browse
-cp<nnn>	Same as option -ccsid. Set *nnn* as the MQMD.CodedCharSetId to use for the MQPUT of the request message.
-crlf	Each line in the input message file is used in sequence as message data. If end-of-file is reached, data is read from the top of the file again. This option is only applicable if used with option -file.
-crtmh	Issue MQCRTMH at the start of the program and MQDLTMH at the end of the program. See -mp to specify or display a message property.
-crtmh1	Same as -crtmh. Issue MQCRTMH at the start of the program and MQDLTMH at the end of the program. See -mp to specify or display a message property so issued once(1) per job
-crtmhn	Issue MQCRTMH before every MQPUT or MQGET and MQDLTMH after the call. So issued many times in the job. See -mp to specify or display a message property.
-d<description>	The <description> is written into the summary file. This is used to write the output from multiple runs into a summary file.
-dump	Causes the qmgr dump to take an SDUMP after the last close request, just before exiting.
-expiry<Value>	Set the expiry time to Value milliseconds.
-file<msg_file>	Read message data from the indicated file and put into the message body. If *msg_file* is a ddname it must be preceded by DD:, for example: -fileDD:MSGIN
-fmt<format>	Sets an MQMD.Format to *format*, padded to the right with blanks (or sets format to eight blanks if just -fmt is specified.) The default format is MQFMT_STRING (= 'MQSTR ')
-group	Process all the messages in the same group.

-gm	Use same MQMD.MsgId for all MQPUTs in the loop, and MQGET replies by this MsgId
-gc	Use same MQMD.MsgId for all MQPUTs in loop. The get uses MQGET.CorrelId = MQPUT.MsgId for the replies. This can be used if the reply message from a server has put the message id into the Correlid field.
	-IMS<value> for messages going to the IMS bridge and reading the IMS Bridge data from a file, replace the transaction with value.
-j<jobname>	Specify additional address spaces for which CPU usage is to be recorded. Multiple -j options may be specified. A wildcard '*' may be added to the end of jobname to match multiple address spaces.
-jms	This creates a JMS header with the following format <jms> <Dst>queue://requestQueue</Dst> <Rto>queue://ReplytoQueue</Rto> <Tms>timeStamp</Tms> <Cid>TimeStamp</Cid> <Dlv>persistence</Dlv> </jms>
-l<msgs_in_loop>	Number of messages per MQPUT and MQGET batch. E.g. -l10 means MQPUT 10 msgs, then MQGET 10 replies, then repeat. Default is 1 msg per batch.
-lr<nnn>	Where nnn is the expected Length Received. If nnn is not specified the put length is used. If the received length is not what was expected the received message is printed and the program returns with return code 1.
-m<qmgr>	Queue manager name. This option is mandatory.
-mpx=y	For each MQPUT sets message property x to value y, where x and y are strings, for example -mpcolour=red. For MQGETs it inquires on each message properties. This sets option -crtmh.
-n<no_of_msgs>	Total number of request messages to put. A value of -1 will result in an endless put/get loop. Default is to put 1 message.
-nl	Change new line (x'15') to blank. This is useful when processing USS files.
-np	use non persistent messages. See option -p below
-o<model_queue>	Use named model queue for MQOPEN of request queue.
-openclose	open and close the queue for every message processed. This can be used to simulate the behaviour of a short lived CICS transaction.
-p	Use persistent messages. The default is non-persistent.
-ph	Prints the headers for the summary file to the summary file.
-plnnn	Preload the queue with the specified number of messages, followed by a commit, then put and get. This is to allow you to keep the server busy with work.

-pm< print_size, print_freq>	Print MQMD and message buffer. *print_ size* is no. of bytes to print. *print_freq* is how often to print. If just -pm is specified, this will print all message data and every message.
-putapplname	Use this value in the put application.
-putwait<value>	wait for <value> hundredths of a second between puts. So -putwait10 will wait for 0.1 of a second between puts.
-put1	Use put1 instead of open put put ... close
-q<requestQ>	Request queue name, i.e. the queue to which messages are put (MQPUT). This option is mandatory. At least one of -q and -r must be specified. If the *requestQ* starts with a / then the data following the / is used as a topic name.
-r<replyQ>	Reply-to-queue from which replies will be retrieved (MQGET). If the -r option is omitted, MQGETs will not be issued. If -r is used without a *replyQ* name, the reply to queue is the same as the request queue, so the puts and gets use the same queue. At least one of -q and -r must be specified. If you want to specify a reply queue, but do not want to get messages from the queue then specify a wait time of 0 as in -w0.
-retain	sets mqpmo.Options |= MQPMO_RETAIN.
-remqm<value>	sets the remote queue manager name to value.
-rfh2	Create an rfh2, or treat the input as rfh2.
-report<value>	
rrs	Use RRS as the coordinator Value is a hexadecimal value which is converted to internal format and put into the report field.
-s<msg_size>	Size of message (excluding the MQMD header) to put to the request queue. Omit -s (or use -s0) in combination with option -file to have message size determined by the data read from the input file. The default message size is 1024 bytes. You can specify nnn | -nnnK | -nnnM.
-sc<file>	Set MQMD.CorrelId from value in the *file* on MQPUT calls. Default MQCI_NONE. Prefix *file* with DD: for a ddname.
-sf<value>	Write a summary of the transaction to the Summary File(SF). If dd:ddname is not specified it defaults to stderr.
-sm file	Set MQMD.MsgId from value in the *file* on MQPUT calls. Prefix *file* with DD: for a ddname. The default MQMD.MsgId value is MQMI_NONE
-sr<reply size>	Size of reply buffer. The default is to get a 100MB buffer. You might need to change this if you are using products like AMS which intercept MQI requests. You can specify nnn | -nnnK | -nnnM.
-ss.. selection string	Specify a selectionString (which allows you to select a message by message properties see -mp) Use underscore _instead of blanks in the string. For example -ss_COLOUR_= _'RED'
-sub*topic/string*	Issue MQSUB using sd.ObjectName of *topic*, and sd.ObjectString of *string*

-t	Time each MQPUT and MQGET call and report statistics
-tm<n>	Time to run the test for in minutes.
-ts<tn>	Time to run the test for in seconds.
-tydatagram	Sets msgType = MQMT_REPORT
-tyrequest	Sets msgType = MQMT_REQUEST .
-tyreply	Sets msgType = MQMT_REPLY
-tyreport	-tyreport Sets msgType = MQMT_REPORT
-ty<value>	sets the value of msgType to be 1. 1 MQMT_REQUEST 2. 2 MQMT_REPLY 3. 4 MQMT_DATAGRAM 4. 8 MQMT_REPORT
-v	For use when debugging. Pauses before each MQPUT and after each MQGET, using WTOR.
-w<wait_time>	Set the MQGET MQMD.WaitInterval to *wait_ time* seconds. A value of 0 (or <0) means don't get a reply message at all (but still set the MQMD.ReplyToQ from -r<replyQ>). Default wait time is 60 seconds
-x	Set MQMD.Format based on message data contents (XML, RFH and RFH2 data currently) supported.

JCL required

1. Parameters will be read from **SYSIN** if specified. This is a fixed block file with record length of 80 bytes.
2. The results are put to the file **SYSPRINT**. This is typically SYSOUT=*
3. If **SUMMARY** is specified then a one line summary is written to this file. The option **-ph** can be used to display the column headings in the file.
4. If **-fileDD:ddname** option is used, then the ddname is require which points to the data set. This can be a sequential file, a member of a PDS, or an HFS file.

CPUIO option

This option gives a measure of your system in terms of CPU constraint and I/O response time.

The program loops for a short time (approx 1 millisecond) and measures the CPU time used, and elapsed time taken. It then loops for a longer period, (about 1 second on the 2064 used at Hursley). If you are constrained for CPU the elapsed time is likely to be much longer than the CPU used.

After the CPU has been measured, 100 records are written to a data set, to produced a measure of the I/O response time. The records written are 4KB long, which is typically the size of a record written by DB2 and MQ in a lightly loaded system.

The first I/O is always reported separately as this is often a long time

The I/O statistics include average, maximum and minimum response times, and which record had the maximum write response time.

Below is some output from a test system at IBM Hursley

```
CPU loop short et =       983 uS CPU Time       975 uS  99 %busy
CPU loop long  et =    979380 uS CPU Time    962603 uS  98 %busy
CPU Time                       26 uS per write-and-flush
I/O  average elapsed time        1001 uS per write-and-flush
I/O  min Elapsed time             641 uS
I/O  maximum elapsed time          1044 uS, record number 3
I/O  first record took          28743 uS
```

The %busy is calculated *from cpu used /elapsed time* and a value in the high 90's shows an unconstrained system

Examples of using OEMPUT

Example 1:
An application runs on a remote system, AIX1. AIX1 is connected to the local queue manager QM01 on the z/OS system, by an MQ channel pair.

In order to test the correct operation of the system, OEMPUT might be run on ZOS1 using the following JCL:

```
//DBTEST JOB CLASS=A,MSGCLASS=H
// EXEC PGM=OEMPUT
// PARM=('-mQM01 -qAIXQ  -rREPLY -fileDD:MSGIN -n1 -pm -w120 ')
//STEPLIB  DD DISP=SHR,DSN=MQM.SCSQLOAD
// DD DISP=SHR,DSN=PAICE.xxx.LOAD
//SYSPRINT DD SYSOUT=*
//MSGIN    DD DISP=SHR,DSN=WMQ.MSG.DATA
```

1. OEMPUT will MQPUT one request message to the AIX queue (-q option), which is a remote queue pointing to the appropriate queue on AIX1,
2. wait 120 (-w option) seconds for a reply message to appear on queue REPLY(-r option).
3. The message data for the request is read from the input file specified by ddname MSGIN(-file option).
4. The MQMD and message data of the request message and of the reply message received will be printed in SYSOUT(-pm option).

Once it has been established that the system functions correctly, the transaction throughput rate might be measured using OEMPUT as follows:

```
//DBTEST JOB CLASS=A,MSGCLASS=H
// EXEC PGM=OEMPUT
// PARM=('-mQM01 -qAIXQ  -rREPLY -fileDD:MSGIN -ts30 -w120 ')
//STEPLIB  DD DISP=SHR,DSN=MQM.SCSQLOAD
// DD DISP=SHR,DSN=PAICE.xxxx.LOAD
//SYSPRINT DD SYSOUT=*
//MSGIN    DD DISP=SHR,DSN=WMQ.MSG.DATA
```

Here OEMPUT will run for 30 seconds (-ts option), without printing any message data (the -pm option is not specified). The message rate achieved will be reported in SYSOUT.

Example output

Workload Manager data

Data is reported from some of the control blocks relating to Workload Manager. These give an indication of any delays. If there are no samples available then no output is displayed

Note. You should use standard z/OS facilies to give a true picture.

The information is extracted from the WLM control blocks at the start of the job,

Example output

```
  Workload manager data
Starting loop at 2015-08-11 12:15:12.197100
 Workload manager data
                 Samples %idle %unknown(MQ?) %using CPU %doing I/O %Wait for CPU
   MQPACHIN.00A6      20   100             0          0          0             0
   MQPAMSTR.00A5       5     0             0         60          0             0
             CSA paging 40
```

Where

Samples	is the number of samples, the PAICEPUT job was short lived so there was only one WLM sample available. The Queue manager MQ03MSTR had been running for longer, and so there were many more samples. WLM resets these statistics periodically and so the report may be incomplete.
%idle	is the percentage of the number of samples that the job was idle.
%unknown(MQ?)	is the percentage of the number of samples when the job is in an unknown wait - this is often because the job is in an MQGET with wait.
%using CPU	is the percentage of the number of samples that the job was using CPU
%doing I/O	is the percentage of the number of samples when the job was doing I/O
%Wait for CPU	is the percentage of the number of samples when the job would run if it could, but was waiting to be dispatched due to higher priority work.

Any other reasons for delay are printed at the end of the line. See the WLM manuals for more information. For specific fields http://publibz.boulder.ibm.com/epubs/pdf/iea2v260.pdf. See z/OS Diagnosis reference GA22-7588-06 and search for *Hyperspace Delay.*

Interpretation of the CPU figures

The CPU used for the application is a true measure of the amount of CPU used. The CPU used figures for other address spaces is valid only if the OEMPUT application is the **only application using the subsystems**. The CPU used is obtained by examining z/OS control blocks, and extracting the CPU time used before and after the measurement. This is thus the total time used, not the time used by this application, so if there are other concurrent users, the figures for the subsystems will include CPU costs for those users as well.

The % CPU figures on the right hand side of the output is the total CPU used by that address space divided by the duration. The percentages are **not** the percentage of where the CPU time is spent out of all the address spaces.

The figures in the ENCLAVE column are when work is done for the address space by another address space in the enclave, for example when running a DB2 stored procedure, the CPU used is recorded against the Enclave.

Example

Using the input parameters

parm: -MMQPA -TM1 -RREPLY -QREPLY -S2K -CGCPC -P

the output produced is given below

```
Total Transactions  : 607350
Elapsed Time        :   60.004 seconds
Application CPU Time:   29.453 seconds  (49.1%)
Transaction Rate    : 10121.894 trans/sec
-------------------------------------------------------
Round trip per msg  :       98 microseconds
Avg App CPU per msg :       48 microseconds
-------------------------------------------------------
 Jobname.ASID  TCB(uS)  SRB(uS) Tot(uS) (%)
                 /tran    /tran    /tran
------------- -------- -------- -------- ----
MQPAMSTR.00A5 00000000 00000003 00000004 4.2
MQPACHIN.00A6 00000000 00000000 00000000 0.0
Total       CPUmicrosecs/tran          4
Grand Total CPUmicrosecs/msg          52
-------------------------------------------------------
Ending loop at 2015-08-11 12:56:57.347588
OEMPUT Normal Exit: End of program
Exiting at 2015-08-11 12:56:57.357426
```

The data in bold font is explained below.

1. The Transaction rate is how many messages were processed per second.
2. The Round trip per msg is the average time it took to process a message. Note that Transaction rate * Round trip per msg = 1.
3. The Avg App CPU per msg is the amount of CPU used by the OEMPUT program when putting and getting messages
4. The figures for VCP0BRK6 show the average CPU used by this address space, TCB time 475 micro seconds, SRB 2 microseconds, total 478 microseconds. The small difference is due to rounding,
5. The 77.9% in the VCP0BRK6 line is the percentage amount of CPU time used during the time period. It is possible for the CPU time used/elapsed time to be greater than 100% if there are more than one flow, or instance of a flow, within an execution group, and there is more than one CPU in the z/OS image.
6. The VCP0BRK* is the sum of the figures for the VCP0BRK jobs

-End of document-