IBM MQ

MQI Test Program (mqpgf)

Ver 1.4.2.10
22 April, 2022

Pulsar Integration Inc.
e-mail : support@pulsarintegration.com

Program Version 1.4.2.10

Tested MQ and OS version

  Windows 10 64bit / IBM MQ 9.0
  Windows 10 64bit / IBM MQ 9.1 Client
  Windows 10 64bit / IBM MQ 9.2.0
  Windows 10 64bit / IBM MQ 9.2.1 Client
  Linux RHEL Server release 7.4 (Maipo) / IBM MQ 9.0
  CentOS Linux release 7.7.1908 64bit / WebSphere MQ 9.1
  HP NonStop i J06.21.01 / IBM MQ 8.1, WebSphere MQ 5.3.1
  HP NonStop X L20.10.00 / IBM MQ 8.1

Compiled and operational experienced MQ and OS version

  SunOS 5.10 sparc / WebSphere MQ 7.5
  SunOS 5.10 sparc / IBM MQ 9.0
  HP-UX 11iV2 (11.23) HP rp3410-2 (PA8900) / WebSphere MQ 7.0.1
  HP-UX 11iV2 (11.23) HP rx1620-2 (IA-64, IPF) / WebSphere MQ 7.0.1
  HP-UX 11iV3 (11.31) ia64 / IBM MQ 9.0
  Linux ppc64 / WebSphere MQ 7.5
  AIX 6.1 / WebSphere MQ 8.0
  AIX 5.3 / WebSphere MQ 7.0.1
  Linux RED Hat 5.5 x86 32bit / WebSphere MQ 7.5
  Linux RED Hat 5.8 x86 64bit / WebSphere MQ 7.5
  Windows 7 64bit / IBM MQ 9.0
  HP NonStop i J06.14.01 / WebSphere MQ 5.3.1
  HP NonStop i J06.20.00, J06.21.01 / IBM MQ 8.0, 8.1, WebSphere MQ 5.3.1
  HP NonStop X L16.05.00 / IBM MQ 8.0, 8.1
  HP NonStop X L20.05.00, L20.10.00 / IBM MQ 8.1

*The Windows version of this program is compiled in 32bit so that it can operate on both 64bit and 32bit Windows OS. Linux x86 version programs are available in both 32-bit and 64-bit versions from version 1.4.2.6. If the 32-bit runtime library is installed on a Linux x86 64-bit OS, the 32-bit version of this program can also run.
This program may be able to run on many OS levels other than the tested environments listed above.

Table Of Contents

7

1. Product overview

About this program

This program is created for the purpose of verifying and confirming the functions and usage of WebSphere MQ / IBM MQ and the MQI that is the API provided by WebSphere MQ / IBM MQ. (MQI uses librarys for the C language.) Although detailed function verification is possible, it may be necessary to specify many options and constants to execute one operation. This program can be used in any process from the project design process to the system operation stage. However, you need to understand the details of the MQI features.

This document does not discuss the details of IBM MQ itself. Please refer to the product documentation as needed.
Manuals for all versions of the product can be found at the URL below.

IBM MQ (formerly IBM WebSphere® MQ)
https://www.ibm.com/support/knowledgecenter/SSFKSJ/com.ibm.mq.helphome.doc/product_welcome_wmq.htm

To check the execution results of the mqpcf command, in addition to the programs provided by MQ products, the MQI test program (mqpgf) command is also used. For details on the MQI program (mqpgf) command, refer to the document "MQI Test Program (mqpgf)".

The version naming scheme

This product uses a similar naming scheme as IBM MQ.

This product releases have a four-digit Version, Release, Modification, and Fix (VRMF) level code.
V: Version
R: Revision
M: Modification
F: Fix

The version of mqpgf / mqpcf does not correspond to the version of the IBM MQ product itself.

・Explanation of each level

The meaning of each level is:

Version: Many features have been added and changed, and the source code is not compatible. However, operation compatibility is maintained as much as possible. User's Guide is created separately.

Revision: Most of the source code is maintained, but Many features have been added. User's Guide is created separately.

Modification: Most of the source code has been maintained, but code has been added for minor new features. Version information is added to the description of the new functions in the user's guide.

Fix: One or more product defects have been fixed in the source code.

・How to upgrade

As each level goes up, the additions and modifications of functions applied at lower levels will be applied at the same time. For example, if the Modification level goes up, all previous Fixes have been applied.
mqpgf / mqpcf are each a single module, so applying the fix is a replacement of the module itself.

・Timing of version upgrade

Revison and Modification level upgrades will be performed on an irregular basis, except when requested by users.
Basically, we do not create a specific version for a specific user. Additional functions with general specifications will be considered.

・Creation of modified version

Upon request, it is possible to fix specific V.R.M, but it is not possible to apply only certain fixes. All previous fixes will be applied. For example, if the version in which the defect was found is 1.4.0.1 and the current Fix level is 1.4.0.15, the fix will be applied to the latest Fix level source code and 1.4.0.16 will be released.
However, if the V.R.M requested to apply the correction is at a significantly earlier level, it may be difficult to apply the correction. In that case, we may ask you to use the latest version with the correction applied.

2. The program execution environment

As a prerequisite for using mqpgf / mqpgfc and mqpcf / mqpcfc, a WebSphere MQ7.0.1 or higher (5.3.1 or higher for HP NonStop) MQ server or client must be installed on your machine. The environment must be able to operate IBM MQ.
mqpgf (c) / mqpcf (c) itself does not require any special installation. All you need to do is download the module for your platform, set the appropriate permissions for that module, and make the command visible in your PATH environment variable. However, depending on your environment, the following operations may be required.

* mqpgf and mqpcf are for bind mode, and mqpgfc and mqpcfc are for client mode.

MQ Install Environment

If you are using MQ7.1 or higher, you need to load the environment of your MQ installation depending on your environment. If the MQ execution environment is not loaded in the startup environment such as the login shell, execute the following to set up the MQ environment to be used.

$. <MQ Install Directory>/bin/setmqenv –s

Reference to MQ libraries

If a message indicating that the MQ libraris cannot be referenced (the following is an example for Solaris) is displayed when executing the program in a UNIX environment, set LD_LIBRARY_PATH (LIBPATH for AIX) and export.

$ mqpgf
ld.so.1: mqpgf: fatal: libmqm.so: open failed: No such file or directory Killed

$ export LD_LIBRARY_PATH=<MQ Install Directory>/lib64:$LD_LIBRARY_PATH
or
$ export LIBPATH=<MQ Install Directory>/lib64:$LIBPATH

Execution user

To execute the program, the execution user must have appropriate access rights set in the queue manager. If you do not know the details of the authority, use a user that is a member of the mqm group (MQ administrator), or include the user you are using in the mqm group.

3. Command usage

Display usage

If you execute mqpgf without any arguments, the usage and the parameters that can be specified are displayed.

Ex. 3.1 Display usage
```
-------------------------------------
$ mqpgf
USAGE:
-qm  : queue manager name(e.g. -qm qm1,qm2,...)
-q   : queue name
-m   : input message
-mx  : input message(hexadecimal notation e.g. 09af..)
-f   : input file name
-o   : output file name
-oq  : output queue name(for queue to queue)('*': ReplyToQ, '**': + ReplyToQMgr)
-iq  : input queue name(for send and receive)
-d   : input directory name
-g   : output directory name
-r   : get repeatedly
-b   : force backout
-l   : message length for writing
-n   : message count for writing or reading
-i   : interval(ms) for writing or reading
-sz  : max message size for reading(byte)(default 12KByte)
-ds  : max message display size(byte)(default 128Byte) (all: entire message)
-br  : browse message
-brv : browse message(verbose)
-dp  : dump message
-dpv : dump message(verbose)
-raw : raw mode output
-hex : dump hexadecimal
-s   : stop before MQI call (e.g. -s MQCMIT)
-p   : process name
-nl  : namelist
-pcf : pcf format file name
-ss  : switch to parameters for secondary
-sp  : switch to parameters for primary
-mc  : get the message with the same CorrelId as MsgId sent
-im  : Inherit MQMD
```

```
-as  : segmentation size
-dl  : delimiter for logical messages
-nt  : number of threads
-ni  : number of threads that call MQCONN/MQDISC internally
-tr  : enable api trace
-sf  : the file for synchronization start
-c   : connection loop count
-sd  : skip MQDISC
-wp  : wait time to next processing
-ca  : continue processing after MQCONN(X) fails
-ac  : applend the counter to message automatically
-cr  : The number of connection retry
-y   : Invoke yield function after every MQI call

Platform-specific options :
-gt  : Using Global UOW for NSK
-gti : Using Global UOW for NSK(TMFAPI: per PUT/GET)

MQI functions :
-set : MQSET (e.g. MQIA_INHIBIT_GET:MQQA_GET_ALLOWED,..:..)
-inq : MQINQ (e.g. MQCA_ALTERATION_DATE,MQIA_CLWL_Q_PRIORITY,..)
-smp : MQSETMP (e.g. MQTYPE_STRING:property name:value,..:..)

MQCD fields (use MQCONNX) :
-x   : ConnectionName (e.g. -x "localhost(1414)")
-ch  : ChannelName              -la  : LocalAddress
-cl  : CertificateLabel         -cs  : SSLCipherSpec
-er  : SSLPeerName

MQMD fields :
-ex  : Expiry(par 100ms)        -ec  : Encoding
-cc  : CodedCharSetId           -pr  : Priority
-mi  : MsgId                    -ci  : CorrelId
-rq  : ReplyToQ                 -rm  : ReplyToQMgr
-ui  : UserIdentifier           -at  : AccountingToken
-ap  : ApplIdentityData         -pn  : PutApplName
-pd  : PutDate                  -pt  : PutTime
-ao  : ApplOriginData
MQMD Version 2 fields :
-gi  : GroupId                  -ms  : MsgSeqNumber
-of  : Offset                   -ol  : OriginalLength
```

```
MQRFH2 fields:
-re  : Encoding(<encoding> or MQENC_*)
-rc  : CodedCharSetId(<ccsid> or MQCCSI_*)
-rf  : Format(e.g. -rf MQFMT_STRING)
-fg  : Flags                      -nc  : NameValueCCSID(<ccsid> or MQCCSI_*)
-nd  : NameValueData(e.g. -nd "data1,data2,data3")

MQCSP fields:
-cu  : CSPUserId                  -cp  : CSPPassword

MQOD fields:
-om  : ObjectQMgrName             -au  : AlternateUserId
-or  : ObjectRec(MQOR)(e.g. ObjectName:ObjectQMgrName,..:..,...)
-dq  : DynamicQName(for receive queue, e.g. "DQ*", "*" or 33bytes full name)

MQPMO fields:
-mr  : PutMsgRec(MQPMR)(e.g. <MsgId>:<CorrelId>:<GroupId>:MQFB_xx:<AccountingTok
en>,..)

MQGMO fields:
-wi  : WaitInterval(ms)           -mt  : MsgToken

MQIMPO fields:
-pe  : RequestedEncoding(<encoding> or MQENC_*)
-pc  : RequestedCCSID(<ccsid> or MQCCSI_*)

MQCB fields :
-op : Operation (e.g. -op MQOP_REGISTER -op MQOP_SUSPEND ...)

MQCBD fields:
-cf : CallbackFunction (e.g. EventHandler)

Constants:
 MQMD       : MQMD_*, MQRO_*, MQMT_*, MQEI_*, MQFB_*, MQENC_*, MQCCSI_*, MQFMT_*,
 MQPRI_*,
             MQPER_*, MQMI_*, MQCI_*, MQACT_*, MQACTT_*, MQAT_*
 MQMDV2     : MQGI_*, MQMF_*, MQOL_*
 MQRFH2     : MQRFH_*
 MQCONNX    : MQCNO_*, MQCSP_*, MQCD_*
 MQOPEN     : MQOO_*, MQOT_*, MQOD_*
 MQPUT      : MQPMO_*, MQPMRF_*
 MQGET      : MQGMO_*, MQWI_*, MQMO_*
```

```
MQCLOSE    : MQCO_*
MQSETMP    : MQPD_*
MQINQMP    : MQIMPO_*
MQCRTMH    : MQCMHO_*
MQCB       : MQCBT_*
------------------------------------
```

Display license information, version information and available constants

When -v is specified for mqpgf, license information, version information of this program and linked library is displayed in addition to USAGE display. In addition, specifying "all" after "-v" will also display all available constants that can be specified for this program. "Constant" as used here means "#define" written in cmqc.h. mqpgf set these to the appropriate fields of constructs defined IBM MQ or options. And mqpgf automatically determines "or" or "overwrite". More than 400 constants, MQMD_ *, MQPMO_ *, MQGMO_ *, etc. can be used. If no constant is specified, default values such as MQMD_DEFAULT are used for the structures and fields used in each MQI.

Note: Not all parameters have been tested.

Ex. 3.2 Display license information, version information and available constants.
* "System number" is displayed only for HPE NonStop.

$ mqpgf -v
. . . .
[License information]
System number   999999
Expires   2023.03.31

version 1.4.2.10 2022/04/22
library version 1.0.0.1 2021/03/10
$
$ mqpgf —v all
. . . .
[License information]
System number   999999
Expires   2023.03.31

version 1.4.2.10 2022/04/22
library version 1.0.0.1 2021/03/10

MQMD_VERSION_1
MQMD_VERSION_2
MQMD_CURRENT_VERSION

MQRO_EXCEPTION
MQRO_EXCEPTION_WITH_DATA
. . . .

Using client mode

When using in client mode, use the mqpgfc command.
Except for the option for client connection, the usage is the same as mqpgf for bind mode.
mqpgfc receives the connection destination IP address or host name and connection port number with the -x option, the MQI channel name with the -ch option, and the local address with the -la option. If -x is specified, MQCMO.Version is automatically set to MQCNO_VERSION_2.
The format of the -x parameter is "ipaddr or hostname (port)". For Windows, there is no need to enclose in double or single quotes.
If -x is specified, mqpgfc passes the connection parameters directly to MQCONNX (), so no other connection settings such as the channel definition table are required.
If -x is not specified, it is necessary to specify connection parameters in the channel definition table, MQSERVER environment variable, or mqclient.ini.
If you need to specify source information (source ipaddr / hostname, source port, tcpip process (HP NonStop)), specify LOCLADDR with -la.
The format of the -la parameter is "local ipaddr or hostname (sender port, port) [/ tcp process name]". "/ tcp process name" can be specified only on HP NonStop.
If you do not specify a channel name with -ch, the default is SYSTEM.DEF.SVRCONN.

mqpgfc -qm <qmgr> -q <queue> -br -x <"ipaddr or hostname(port)"> -ch <channel name> -la <"source ipaddr or hostname(source port)/tcpip processname"

e.g.
mqpgfc -qm SampleQM -q SampleQ -br -x "hostname(1414)" -ch PULSAR.MQICHL -la "localhost(1234) "

* If you need to specify a specific TCPIP process on HP NonStop, use -la "localhost(1234)/ztc3". (When specifying $ZTC3)

4. Basic test

4.1 Common parameters

In almost all cases, mqpgf needs to specify the queue manager name (-qm) and
queue name (-q).
In the rest of this document, the description of these parameters will be omitted.

mqpgf **-qm <qmgr> -q <queue>** ….

-qm: Queue manager name
-q: Queue name

* In this program, it may be necessary to specify many parameters depending on
the type of test. On some platforms such as Solaris, the number of characters that
can be specified on one line of the command line may be set smaller by default. If
you can not enter at once, please enter multiple lines separated by '¥'.

4.2 Put messages specified on the command line

Specify the message to be put directly on the command line. By default, MQ MD uses MQMD_DEFAULT.

mqpgf -qm <qmgr> -q <queue> **-m <message>** -n <count> -i <interval> -l <size>

-m: Input message

(sample options)
-n: Number of messages put
-i: Put interval(ms) (Effective when -n is specified)
-l: Put message length (If it is larger than the specified message, the following is filled with 0x00. If it is smaller than the specified message, it is cut to that size.)

Ex. 4.2.1 Put with the number of messages, a put interval and a message length parameter.

$ mqpgf -qm HM8A -q LQ1 -m "test message" **-n 3 -i 5000 -l 20**
[20/01/23 17:48:35.994310789] 1: message length: 20 put message: test messag
e........
[20/01/23 17:48:40.995207421] 2: message length: 20 put message: test messag
e........
[20/01/23 17:48:45.995712958] 3: message length: 20 put message: test messag
e........
Elapsed time = 10.012609 sec
$
$ mqpgf -qm HM8A -q LQ1 -r
[20/01/23 17:49:04.255936507] 1: message length: 20 get message : test messag
e........
[20/01/23 17:49:04.256519249] 2: message length: 20 get message : test messag
e........
[20/01/23 17:49:04.256687651] 3: message length: 20 get message : test messag
e........
no message available : LQ1 CompCd=02 ReasonCd=2033
Elapsed time = 0.010942 sec
* If '-r' option is specified, mqpgf get message repeatedly until to be returned RC = 2033 (MQRC_NO_MSG_AVAILABLE). 0x00 added by "-l 20" is displayed as "."
Because it is an invisible character.

4.3 Specify a message to put in HexaDecimal notation on the command line

Specify a message in hexadecimal notation on the command line.

mqpgf -qm <qmgr> -q <queue> **-mx <input message(hexadecimal notation)>** -n <count> -i <interval> -l <size>

Ex. 4.3.1 Put a message in hexadecimal notation.

$ mqpgf -qm HM8A -q LQ1 **-mx 0123abcdABCD**
[20/01/23 17:52:25.875333411] 1: message length: 6 put message: 0x0123ABCDABCD
Elapsed time = 0.011225 sec
$
$ mqpgf -qm HM8A -q LQ1 -hex
[20/01/23 17:52:35.760981978] 1: message length: 6 get message : 0x0123ABCDABCD
Elapsed time = 0.011612 sec
* "-hex" option displays get messages in hexadecimal notation.

4.4 Write messages to the standard output(in visible characters)

Writes get messages to the standard output. By default, invisible characters are output as ".". MQMD is discarded. If you get from the transmission queue, the transmission queue header and the message body MQMD and MQMDE are also deleted. If you get from the dead letter queue, MQDLH will also be deleted.

mqpgf -qm <qmgr> -q <queue> -r

(sample options)
-r: Read a message repeatedly until a queue become empty.

Ex. 4.4.1 Get messages (binary data) containing invisible characters.

$ mqpgf -qm HM8A -q LQ1 **-mx 01020304057FFF**
[20/01/23 17:54:06.975709566] 1: message length: 7 put message: **0x01020304057F
FF**
Elapsed time = 0.010685 sec
$
$ mqpgf -qm HM8A -q LQ1
[20/01/23 17:54:13.377889722] 1: message length: 7 get message : ....
Elapsed time = 0.008381 sec

4.5 Put a file data

Put a file data in the queue as it is. File data is data excluding MQMD. Unless otherwise specified, MQMD uses the default value (MQMD_DEFAULT).
The following -n and -i options may be available for other tests.

mqpgf -qm <qmgr> -q <queue> **-f <filename>** -n <count> -i <interval> -l <size>

-f: Input file name

(sample options)
-n: Nmber of files to put
-i: Put interval(ms) (Effective when -n is specified)
-l: Put message length (If it is larger than the specified message, the following is filled with 0x00. If it is smaller than the specified message, it is cut to that size.)

Ex. 4.5.1 Example of specifying the number of file and an interval to put.

```
$ ls -l input.txt
-rw-r--r--   1 mq80       mqm                 19 Dec 14 19:13 input.txt
$
$ cat input.txt
input file message
$
$ od -x input.txt
0000000  696e 7075 7420 6669 6c65 206d 6573 7361
0000020  6765 0a00
0000023
$
* Specify 2 times for the number of puts and 3 seconds for the interval between puts
$ mqpgf -qm HM8A -q LQ1 -f input.txt -n 2 -i 3000
[20/01/23 17:56:22.285117540] 1: put from: input.txt
[20/01/23 17:56:25.285613927] 2: put from: input.txt
Elapsed time = 3.010767 sec
$
$ mqpgf -qm HM8A -q LQ1
[20/01/23 17:56:32.492878183] 1: message length: 19 get message : input file mes
sage

Elapsed time = 0.009688 sec
$ mqpgf -qm HM8A -q LQ1 -hex
[20/01/23 17:56:38.486576036] 1: message length: 19 get message : 0x696E707574
2066696C65206D6573736167650A
```

Elapsed time = 0.008977 sec

4.6 Write a message read from a queue to a file

Write a get message to a file. MQMD is discarded. If you get from the trans mission queue, the transmission queue header and the message body MQMD and MQMDE are also deleted. If you get from the dead letter queue, MQDL H will also be deleted.

mqpgf -qm <qmgr> -q <queue> **-o <filename>**

-o: Output file name

If the file specified by -o already exists, you will be asked if you want to ove rwrite it.

file allready exist. overwrite ? y/n : y

Enter (y | Y) to continue processing, or enter (n | N) to suspend processing.

*The following options will be ignored even if specified at the same time. -r (Get Re peatedly) and -n (Message count for writing or reading) and -raw (raw mode output) and -hex (output in hexadecimal notation)

Ex. 4.6.1 Write a get message to a file

$ mqpgf -qm HM8A -q LQ1 -m "line 1
> line 2
> line 3
> "
[20/01/23 18:19:05.753343645] 1: message length: 21 put message: line 1
line 2
line 3

Elapsed time = 0.013499 sec
* In a Unix shell, line breaks can be entered as above.
$
$ mqpgf -qm HM8A -q LQ1 -o /home/mqm/tmp/output.txt
file already exist. overwrite? y/n : y
[20/01/23 18:20:51.591537336] 1: message length: 21 output filename : /home/mqm
/tmp/output.txt
Elapsed time = 1.311320 sec
$ cat /home/mqm/tmp/output.txt
line 1
line 2
line 3
$

* If the file specified in the output already exists, it will be checked if it is OK
to overwrite.

  If you specify MQMD Version 1 when you get a message that uses MQMD Version
  2, MQMDE (extended message descriptor) is created automatically by the queue
  manager. However, if the field extended in Version 2 is not actually used (the
  default value is kept), no MQMDE header is created.
  The following is a test example when MQMDE is generated. mqpgf deletes
  MQMDE and outputs the message "cut MQMDE header".

Ex. 4.6.2 Get message with specifying MQMD version.
----
* Put a message with MQMD V2.

$ mqpgf -qm HM8A -q LQ1 -m test **-gi GID -ms 3 -of 100 -ol 1000 MQMD_VER
SION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP**
[20/01/24 09:43:01.721510414] 1: message length: 4 put message: test
Elapsed time = 0.014961 sec

* When putting, specifiy MQMD Version2(MQMD_VERSION_2), and fields extended
by Version2 are set to other than default. MQMF_SEGMENT is specified to set Offset
(-of) and GroupId (-gi), MQMT_REPORT is specified to set OriginalLength (-ol), and
MQMF_MSG_IN_GROUP is specified to set MsgSeqNumber (-ms).

* Get the message using MQMD V2 with specifying MQMD V1. (MQMD_VERSION_
1 is used by default.)

$ mqpgf -qm HM8A -q LQ1 -o /home/mqm/tmp/output.txt
file already exist. overwrite? y/n：y
Cut MQMDE Header
[20/01/24 09:43:51.817819504] 1: message length: 4 output filename：/home/mqm/
tmp/output.txt
Elapsed time = 1.919485 sec

  Messages using MQMD Version 2 are separated into MQMD V1 and MQMDE on
  the transmission queue.

Ex. 4.6.3 Read a message on a transmission queue.

C:¥Users¥mqm>mqpgfc -qm HM8A -q REQ8B1 -x "remotehost(1414)" -ch PULSA
R.MQICHL -f input.txt -gi GID -ms 3 -of 100 -ol 1000 **MQMD_VERSION_2** MQ
MF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP
[2012/01/30 11:08:13.801] 1: put from: input.txt

Elapsed time = 187 msec

* Because the transmission queue is normally prohibited from being read, set it to be readable before getting it.

C:¥Users¥mqm>mqpcfc que -qm HM8A -q HM8B -x "remotehost(1414)" -ch PUL
SAR.MQICHL GET
1: QUEUE(HM8B) TYPE(QLOCAL) **GET(DISABLED)**

C:¥Users¥mqm>mqpcfc get enable -qm HM8A -q HM8B -x "remotehost(1414)" -ch
  PULSAR.MQICHL
Get Enabled : HM8B

C:¥Users¥mqm>mqpcfc que -qm HM8A -q HM8B -x "remotehost(1414)" -ch PUL
SAR.MQICHL GET
1: QUEUE(HM8B) TYPE(QLOCAL) **GET(ENABLED)**

C:¥Users¥mqm>mqpgfc -qm HM8A -q HM8B -x "remotehost(1414)" -ch PULSAR.
MQICHL -o output.txt
file already exist. overwrite? y/n : y
Cut Xmit Queue Header and message body MQMD
Cut MQMDE Header
[2012/01/30 11:08:54.287] 1: message length: 12 output filename : output.txt
Elapsed time = 2578 msec

C:¥Users¥mqm>type output.txt
test message

  MQDLH is inserted between MQMD V1 and MQMDE, (MQMD V1, MQDLH, MQMDE), when a message using MQMD Version 2 is put in the dead letter queue.

Ex. 4.6.4 Write a messages on dead letter queue to a file.

$ mqpgfc -qm HM8A -q SYSTEM.DEAD.LETTER.QUEUE -x "172.21.10.50(18
591)" -ch PULSAR.MQICHL -brv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[4] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[819] Format[MQDEAD] Priority[0] Persistence[0] MsgId[0
x414D5120484D394C2020202020202087A9405E03630A22] CorrelId[0x0000000000
00000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                                   ] ReplyToQMgr[HM9L
               ] UserIdentifier[mqm   ] AccountingToken[0x043130
303000000000000000000000000000000000000000000000000000000006] ApplIdentityD

ata[   ] PutApplType[6] PutApplName[mqpgf
                         ] PutDate[20120210] PutTime[00550607] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[DLH] Version[1] Reason[2085] DestQName[LLQ
                          ] DestQMgrName[HM8A
        ] Encoding[546] CodedCharSetId[1208] Format[MQHMDE] PutApplType
[13] PutApplName[amqrmppa   ] PutDate[20120210] PutTime[00
560505]

*StrucId[MDE] Version[2] StrucLength[72] Encoding[546] CodedCharSetId[1208]
Format[   ] Flags[0] GroupId[0x474944000000000000000000000000000000000
000000000] MsgSeqNumber[3] Offset[100] MsgFlags[10] OriginalLength[1000]

data length: 13
00000000: 7465 7374 206D 6573 7361 6765 0A   'test message.
 '

Elapsed time = 0.105855 sec

$
$ mqpgfc -qm HM8A -q SYSTEM.DEAD.LETTER.QUEUE -x "172.21.10.50(18
591)" -ch PULSAR.MQICHL -o output.txt
file already exist. overwrite? y/n：y
Cut Dead Letter Header
Cut MQMDE Header
[20/02/10 11:35:27.245865904] 1: message length: 13 output filename：output.txt
Elapsed time = 1.536663 sec
$

4.7 Write a read message to a queue(re-queue)

Re-queue the received message to another queue.
If it is necessary to specify parameters on the writing side (secondary side), use the "-ss" option to switch the parameters to the writing side (secondary side). (cf. "5. All parameters reference" - "Switch subsequent paremetera to secondery (-ss)")

mqpgf -qm <qmgr> -q <queue> **-oq <output queue>** -r

-oq: output queue name

(sample options)
-r: Read a message repeatedly until a queue become empty.

Ex. 4.7.1 Outputs all messages in an input queue to another queue.
----
* Put three messages on the input queue
$ mqpgf -qm HM8E2 -q LQ1 -m "sample message" **-n 3**
[20/02/28 15:38:15.736915] 1: message length: 14 put message: sample message
[20/02/28 15:38:15.742337] 2: message length: 14 put message: sample message
[20/02/28 15:38:15.742439] 3: message length: 14 put message: sample message
Elapsed time = 0.179671 sec

* Get messages of the input queue repeatedly and transfer all to the output queue with specifying MQMD.MsgType and CodedCharSetId.
$ mqpgf -qm HM8E2 -q LQ1 -oq LQ2 -r -ss MQMT_REPLY -cc 930
[20/02/28 15:39:16.385869] 1: message length: 14 get message : sample message
[20/02/28 15:39:16.393922] 1: message length: 14 put message : sample message
[20/02/28 15:39:16.394323] 2: message length: 14 get message : sample message
[20/02/28 15:39:16.394436] 2: message length: 14 put message : sample message
[20/02/28 15:39:16.394974] 3: message length: 14 get message : sample message
[20/02/28 15:39:16.395086] 3: message length: 14 put message : sample message
no message available : LQ1 CompCd=02 ReasonCd=2033
Elapsed time = 0.129444 sec

$ mqpgf -qm HM8E2 -q LQ2 -dp -r
message number: 1
*StrucId[MD] .... **MsgType[2]** .... **CodedCharSetId[930]** ....
....
00000000: 7361 6D70 6C65 206D 6573 7361 6765   'sample message '

message number: 2
*StrucId[MD] .... **MsgType[2]** .... **CodedCharSetId[930]**....
....
00000000: 7361 6D70 6C65 206D 6573 7361 6765   'sample message '

message number: 3
*StrucId[MD] .... **MsgType[2]** .... **CodedCharSetId[930]** ....
....
00000000: 7361 6D70 6C65 206D 6573 7361 6765   'sample message '

no message available : LQ2 CompCd=02 ReasonCd=2033
Elapsed time = 0.063891 sec

4.8 Send messages and receive reply messages

Put messages and receive their responses.
If it is necessary to specify parameters on the reading side (secondary side), use the   "-ss" option to switch the parameters to the reading side (secondary side). (cf. "5. All parameters reference" - "Switch subsequent paremetera to secondery (-ss)")

mqpgf -qm <qmgr> -q <queue> -m <input message> **-iq <input queue>** -n <count > -i <interval>

-iq: input queue name

(sample options)
-n: Number of messages put
-i: Put interval(ms) (Effective when "-n" is specified)

Ex. 4.8.1 After putting messages, receive their response messages.
----
* After putting messages, receive response messages three times at 1 second intervals from the queue specified by "-iq" option.
$ mqpgf -qm HM8E2 -q RemoteQ -m "request reply test" -iq ReplyQ MQMT_REQUEST -n 3 -i 1000 -ss MQGMO_WAIT MQWI_UNLIMITED
[20/02/28 16:05:40.042057] 1: message length: 18 put message: request reply test
[20/02/28 16:05:40.047693] 1: message length: 18 **get message** : request reply test
[20/02/28 16:05:41.048099] 2: message length: 18 put message: request reply test
[20/02/28 16:05:41.048588] 2: message length: 18 **get message** : request reply test
[20/02/28 16:05:42.049564] 3: message length: 18 put message: request reply test
[20/02/28 16:05:42.050027] 3: message length: 18 **get message** : request reply test
Elapsed time = 2.332243 sec

4.9 Put all files in a directory

Put all the data of the files in the directory.
The data in the file is the data excluding MQMD. Unless otherwise specified,
MQMD uses the default value (MQMD_DEFAULT).

mqpgf -qm <qmgr> -q <queue> **-d <directory>** -n <count> -i <interval> -l <size>

-d: Input directory name

(sample options)
-n: Number of puts per file
-i: Put interval(ms) (Effective when "-n" is specified)
-l: Put message length(If the value is larger than the specified message, the follo
wing is padded with 0x00. If it is smaller than the specified message, it is cut t
o that size.)

Ex. 4.9.1 Put files in a directory with specifying the number of times and interval.
----
*PUT files in the directory "input" twice at 3 second intervals.
$ mqpgf -qm HM8E2 -q LQ1 -d input -n 2 -i 3000
[20/02/28 16:30:16.551001] 1: put from: input/test1.txt
[20/02/28 16:30:19.558533] 2: put from: input/test1.txt
[20/02/28 16:30:22.559016] 1: put from: input/test2.txt
[20/02/28 16:30:25.560174] 2: put from: input/test2.txt
[20/02/28 16:30:28.560654] 1: put from: input/test3.txt
[20/02/28 16:30:31.561816] 2: put from: input/test3.txt
Elapsed time = 15.299729 sec
$
$ amqsbcg LQ1 HM8E2 | grep PutDate
  PutDate ：'20120228'　　PutTime ：'07301655'
  PutDate ：'20120228'　　PutTime ：'07301955'
  PutDate ：'20120228'　　PutTime ：'07302255'
  PutDate ：'20120228'　　PutTime ：'07302556'
  PutDate ：'20120228'　　PutTime ：'07302856'
  PutDate ：'20120228'　　PutTime ：'07303156'
* MQMD PutDate/PutTime is GMT(JST-9)

4.10 Get messages from a queue and output to a directory

Get the specified number of messages in the queue or all of them, and save the data with a unique file name for each message. MQMD is discarded. When GET is performed from the transmission queue, the transmission queue header and the message body MQMD and MQMDE are also deleted. In addition, when getting from the dead letter queue, MQDLH is also deleted.

mqpgf -qm <qmgr> -q <queue> **-g <directory>** [-n <count> | -r] -i <interval>

-g: Output directory name

(sample options)
-n: Number of message to read(Cannot be specified with "-r" option)
-r: Get a message on a queue repeatedly(Cannot be specified with "-n" option)
-i: Get interval(ms) (Effective when "-r" or "-n" is specified)

Output file name format:
yyyymmdd_HHMMSS_msec_seqno
yyyymmdd, HHMMSS and msec are obtained from PutDate and PutTime of the get message MQMD.
If there are multiple messages with the same time, seqno is not added for the first time, and is added as 1,2,3 ... for the second and subsequent times.

If the directory specified with "-g" option already exists, you will be asked if you want to output the file to that directory.

directory allready exist. overwrite ? y/n : y

Enter (y | Y) to continue processing, or enter (n | N) to suspend processing.

Ex. 4.10.1 Get all messages on a queue with a specified interval and write them to a directory.
----
*Specify to get at 0.5 second intervals.
$ mqpgf -qm HM8E2 -q LQ1 **-g /home/mqm/output -r -i 500**
directory already exist. overwrite? y/n : y
[20/02/28 16:55:39.711720] 1: write file : /home/mqm/output/20120228_163016_55
[20/02/28 16:55:40.262528] 2: write file : /home/mqm/output/20120228_163019_55
[20/02/28 16:55:40.799099] 3: write file : /home/mqm/output/20120228_163022_55
[20/02/28 16:55:41.336690] 4: write file : /home/mqm/output/20120228_163025_56
[20/02/28 16:55:41.879415] 5: write file : /home/mqm/output/20120228_163028_56
[20/02/28 16:55:42.423150] 6: write file : /home/mqm/output/20120228_163031_56

no message available : LQ1 CompCd=02 ReasonCd=2033
Elapsed time = 5.614104 sec
* When "-r" is specified, get repeatedly until RC = 2033 (MQRC_NO_MSG_AVAI
LABLE).

4.11 Browse and dump messages on a queue(normal mode)

Browse messages on a queue and dump them in hexadecimal. Only MQMD is displayed for each field.

* MQMD_VERSION_2 is used by default as an exception when browsing / dumping. (Other tests using MQGET use MQMD_DEFAULT, so MQMD_VERSION_1 is the default.) If you want to use MQMD_VERSION_1 specifically, you need to specify MQMD_VERSION_1 as an argument.

mqpgf -qm <qmgr> -q <queue> **-br** -r

-br: browse(nomal mode)

(sample options)
-r: Read a message repeatedly until a queue become empty.
MQMD_VERSION_1: An MQMDE header is generated if the fields added in MQMD_VERSION_2 are used.

Ex. 4.11.1 Browse all messages on a queue.

$ mqpgf -qm HM8E2 -q LQ1 **-br -r**
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[819] Format[   ] Priority[0] Persistence[0] MsgId[0x41
4D5120484D3845322020202020205E58AD7820003B0D] CorrelId[0x000000000000
00000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                       ] ReplyToQMgr[HM8E2
              ] UserIdentifier[mqm   ] AccountingToken[0x05343430
33310000000000000000000000000000000000000000000000000006] ApplIdentityData
[   ] PutApplType[13] PutApplName[mqpgf
         ] PutDate[20120228] PutTime[08343590] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 8
00000000: 6D65 7373 6167 6531   'message1   '

message number: 2
....
Elapsed time = 0.062943 sec

Ex. 4.11.2 Example when MQMDE header is generated when getting a message.
----
* Put a message with MQMD V2
$ mqpgf -qm TESTQM -q TQ -m test **-gi GID -ms 3 -of 100 -ol 1000 MQMD_VE RSION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP**
[16/12/22 19:55:43] 1: message length: 4 put message : test
$
* Read the message with specifying MQMD V1
$ mqpgf -qm HM8E2 -q LQ1 -br MQMD_VERSION_1
message number: 1
*StrucId[MD] Version[1] Report[0] MsgType[4] Expiry[-1] Feedback[0] Encoding [273] CodedCharSetId[819] Format[MQHMDE] Priority[0] Persistence[0] MsgId[0 x414D5120484D384532202020202020205E58AD7820003F08] CorrelId[0x0000000000 0000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                           ] ReplyToQMgr[HM8E2
              ] UserIdentifier[mqm   ] AccountingToken[0x053434 30333310000000000000000000000000000000000000000000000006] ApplIdentityD ata[   ] PutApplType[13] PutApplName[mqpgf
          ] PutDate[20120228] PutTime[08422337] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 76
00000000:   4D44 4520 0000 0002 0000 0048 0000 0111 'MDE ....H....'
00000010:   0000 0333 2020 2020 2020 2020 0000 0000 '...3   ....'
00000020:   4749 4400 0000 0000 0000 0000 0000 0000 'GID.............'
00000030:   0000 0000 0000 0000 0000 0003 0000 0064 '...............d'
00000040:   0000 000A 0000 03E8 7465 7374   '........test   '

Elapsed time = 0.061495 sec

* "MQHMDE" (MQFMT_MD_EXTENSION) is set in MQMD.Format, and the beginni ng of the data part is "MDE" (MQMDE_STRUC_ID).

4.12 Browse and dump messages on a queue(verbose mode)

Browse messages on the queue and display the result in hex dump. MQMD (independent message descriptor), MQXQH (transmission queue header), MQMD of message data (embedded message descriptor), MQMDE (extended message descriptor), MQDLH (dead letter header), MQRFH2 (rule and format header 2) and PCF (Programmable Command Format) are displayed for each field.

For the items shown in the table below, only visible characters are output, and for invisible characters, "." Is displayed as a substitute character. These fields can be displayed differently.If '-hex' is specified, the message is displayed in hexadecimal notation, and if '-raw' is specified, it is written to stdio as it is.

Table 4.12.1 Fields affected by display mode

Datatype	Field	Note
MQCA_ALTERATION_DATE	Non-numeric fields	PCF
MQCFT_BYTE_STRING_FILTER	Non-numeric fields	PCF
MQCFT_BYTE_STRING	Non-numeric fields	PCF
MQCFT_STRING_FILTER	Non-numeric fields	PCF
MQCFT_STRING_LIST	Non-numeric fields	PCF
MQCFT_STRING	Non-numeric fields	PCF
MQDLH	Non-numeric fields	
MQRFH2	Items including NameValueData other than numbers	
MQMD	Report	
MQMD	MsgFlags	MQMD Ver 2
MQMDE	MsgFlags	

mqpgf -qm <qmgr> -q <queue> **-brv** -hex

-brv: browse(verbose mode)

(sample options)
-raw: Output in raw mode.
-hex: Output in hexadecimal notation.

Ex. 4.12.1 Dump a message on a transmission queue in detail.
----
* After stopping the channel, put the message specified MQMD_VERSION_2 and MQMF_LAST_MSG_IN_GROUP (0x00000010) to the local definition of the remote queue.
The default value when put by mqpgf is MQMD_VERSION_1. And MQMF_LAST_MSG_IN_GROUP is specified to make the fields extended by MQMD V2 non-default values. If you specify MQMF_LAST_MSG_IN_GROUP as an argument, it will be automatically set in MsgFlags of MQMD V2.

$ mqpgf -qm HM8B -q REP5A1 -m "dump xmitq" MQMD_VERSION_2 MQMF_L AST_MSG_IN_GROUP
[20/03/02 17:47:06.595354] 1: message length: 10 put message: dump xmitq
Elapsed time = 0.119847 sec

* Set the transmission queue to read enabled.
mqpcf get enable -qm HM8B -q HM5A
Get Enabled : HM5A

$ mqpgf -qm HM8B -q HM5A **-brv**
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[819] Format[MQXMIT] Priority[0] Persistence[0] MsgId[0x 414D5120484D384220202020202020205E5CC1B120002A0A] CorrelId[0x414D512048 4D384220202020202020205E5CC1B120002A08] BackoutCount[0] ReplyToQ[
                                        ] ReplyToQMgr[HM8B
                ] UserIdentifier[mqm   ] AccountingToken[0x0534 343033331000000000000000000000000000000000000000000000000006] ApplIdentity Data[   ] PutApplType[7] PutApplName[HM8B
            ] PutDate[20120302] PutTime[08470659] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[XQH] Version[1] RemoteQName[REP1
          ] RemoteQMgrName[HM5A
 ]

*StrucId[MD] Version[1] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[819] Format[MQHMDE] Priority[0] Persistence[0] MsgId[0 x414D5120484D384220202020202020205E5CC1B120002A08] CorrelId[0x000000000 000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                    ] ReplyToQMgr[HM8B
                ] UserIdentifier[mqm   ] AccountingToken[0x05343

4303331000000000000000000000000000000000000000000000000006] ApplIdentity
Data[   ] PutApplType[13] PutApplName[mqpgf
                    ] PutDate[20120302] PutTime[08470659] ApplOriginData[   ]

*StrucId[MDE] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[819] F
ormat[   ] Flags[0] GroupId[0x414D5120484D38422020202020202020205E5CC1
B120002A09] MsgSeqNumber[1] Offset[0] **MsgFlags[24]** OriginalLength[-1]

data length: 10
00000000: 6475 6D70 2078 6D69 7471   'dump xmitq   '

Elapsed time = 0.074383 sec

* If MQMF_LAST_MSG_IN_GROUP (0x00000010) is specified, the queue manager
additionally sets MQMF_MSG_IN_GROUP (0x00000008). As a result, the MsgFlags of
the MQMDE that appears when browsing shows the sum of the two, 0x00000018 = 24.

* Return the transmission queue to the read-only.
$ mqpcf get disable -qm HM8B -q HM5A
Get Disabled : HM5A

Ex. 4.12.2 Dump messages on a dead letter queue.

* DLH is created and put in the encoding of the machine with the dead letter q
ueue. Therefore, the encoding of the machine that has DLQ is also set for the e
ncoding of MQMD. When dump is displayed, mqpgf displays the value of the nu
meric field after encoding conversion.

$ mqpgfc -qm mqm90b **-q SYSTEM.DEAD.LETTER.QUEUE** -x "remotehost(1414)"
 -brv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[4] Expiry[-1] Feedback[0] Encoding
[546] CodedCharSetId[943] Format[MQDEAD] Priority[0] Persistence[1] MsgId[0
x414D51206D716D393061202020202020F6B95358204CC002] CorrelId[0x000000000
000000000000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                              ] ReplyToQMgr[mqm90a
                    ] UserIdentifier[testuser   ] AccountingToken[0x1601051
50000005CB9193C9FEF8154FF9CFF8AE8030000000000000000000000B] ApplIdentit
yData[   ] PutApplType[11] PutApplName[ojects¥mq
pgf¥Debug¥mqpgf.exe] PutDate[20161216] PutTime[12313887] ApplOriginData[
 ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe

r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[DLH] Version[1] Reason[2051] DestQName[TQ1
                        ] DestQMgrName[mqm90b
        ] Encoding[273] CodedCharSetId[943] Format[MQHMDE] PutApplType[1
1] PutApplName[es¥IBM¥MQ¥bin64¥amqrmppa.exe] PutDate[20161216] PutTime
[12362262]

*StrucId[MDE] Version[2] StrucLength[72] Encoding[546] CodedCharSetId[943] F
ormat[   ] Flags[0] GroupId[0x474944000000000000000000000000000000000
00000000] MsgSeqNumber[1] Offset[100] MsgFlags[2] OriginalLength[1000]

data length: 248
00000000: 7465 7374   'test   '

Ex. 4.12.3 Dump JMS message including RFH2 header.

$ mqpgf -qm HM8E2 -q LQ1 -brv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[1208] **Format[MQHRF2]** Priority[4] Persistence[1] MsgId
[0x414D5120484D384532202020202020205E5DB3D420004002] CorrelId[0x00000000
00000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                                ] ReplyToQMgr[HM8E2
            ] UserIdentifier[mqm   ] AccountingToken[0x0534
34303333100000000000000000000000000000000000000000000000000006] ApplIdentity
Data[   ] PutApplType[13] PutApplName[java
            ] PutDate[20120303] PutTime[03132220] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[RFH] Version[2] StrucLength[148] Encoding[273] CodedCharSetId[1208]
Format[MQSTR   ] Flags[0] NameValueCCSID[1208]
NameValueLength[32] NameValueData[<mcd><Msd>jms_text</Msd></mcd>]
NameValueLength[72] NameValueData[<jms><Dst>queue:///LQ1</Dst><Tms>15832
05202175</Tms><Dlv>2</Dlv></jms>]
data length: 11
00000000: 4A4D 5320 4D65 7373 6167 65   'JMS Message   '

Elapsed time = 0.066488 sec

Ex. 4.12.4 Dump a PCF format message.
* The following is an excerpt of an example of browsing the cluster transmission
  queue.

$ mqpgf -qm HM8E2 -q SYSTEM.CLUSTER.TRANSMIT.QUEUE -brv
message number: 1
*StrucId[MD   ] Version[2] Report[0] MsgType[8] Expiry[22535239] Feedback[0] En
coding[273] CodedCharSetId[819] **Format[MQXMIT   ]** Priority[0] Persistence[1] M
sgId[0x414D5120484D384533202020202020205E58AD782000020A] CorrelId[0x544F
2E484D384D3220202020202020202020202020202020] BackoutCount[0] ReplyToQ[
                                                        ] ReplyToQMgr[HM8E2
                          ] UserIdentifier[mqm   ] AccountingToken[0
x0000000000000000000000000000000000000000000000000000000000000000] ApplId
entityData[   ] PutApplType[7] PutApplName[HM8E
2   ] PutDate[20120228] PutTime[06044187] ApplOriginData[
   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[XQH] Version[1] **RemoteQName[SYSTEM.CLUSTER.COMMAND.QUEU
E**   ] RemoteQMgrName[HM8M2
         ]

*StrucId[MD   ] Version[1] Report[0] MsgType[8] Expiry[22535239] Feedback[0] En
coding[273] CodedCharSetId[819] **Format[MQADMIN]** Priority[0] Persistence[1] M
sgId[0x414D5120484D384533202020202020205E58AD7820000204] CorrelId[0x00000
000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                                        ] ReplyToQMgr[HM8E2
                          ] UserIdentifier[mqm   ] AccountingToken[0x0
53434303331000000000000000000000000000000000000000000000000000006] ApplIden
tityData[   ] PutApplType[7] PutApplName[amqrrmf
a   ] PutDate[20120228] PutTime[06044185] ApplOriginData[
 ]

***MQCFH(MQCFT_COMMAND) Type[1] StrucLength[36] Version[1] Command[100
1] MsgSeqNumber[1] Control[1] CompCode[0] Reason[0] ParameterCount[83]**
(MQCFT_STRING) Type[4] StrucLength[68] Parameter[5514] CodedCharSetId[0] S
tringLength[48] String[HM8E2_2012-02-03_16.49.24   ]
(MQCFT_STRING) Type[4] StrucLength[68] Parameter[5502] CodedCharSetId[0] S
tringLength[48] String[HM8M2_2011-11-26_15.28.42   ]
(MQCFT_STRING) Type[4] StrucLength[68] Parameter[5515] CodedCharSetId[0] S
tringLength[48] String[HM8E2_2012-02-03_16.49.24   ]
…

(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1502] Value[50]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1503] Value[0]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1505] Value[10]
...
(MQCFT_INTEGER_LIST) Type[5] StrucLength[24] Parameter[1575] Count[2] Values[0,-1]
(MQCFT_INTEGER_LIST) Type[5] StrucLength[80] Parameter[1576] Count[16] Values[0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]
...
(MQCFT_STRING) Type[4] StrucLength[20] Parameter[3516] CodedCharSetId[0] StringLength[0] String[]
(MQCFT_STRING_LIST) Type[6] StrucLength[72] Parameter[5500] CodedCharSetId[0] Count[1] StringLength[48] String1[PLDCL
        ]

Elapsed time = 0.371101 sec

4.13 Write a message read from a queue to standard output(raw mode)

If you want to redirect a message to a file as is, output it in Raw mode. The "-raw" option is not required when outputting to a file with the "-o" or "-g" options. (Required and effective only when writing to standard output)

mqpgf -qm <qmgr> -q <queue> **-raw** -r

-raw: Output in raw mode

(sample options)
-r: Read a message repeatedly until a queue become empty.

Ex. 4.13.1 Save binary data to a file by redirecting.

$ mqpgf -qm HM8A -q LQ1 -mx 010231324142
[20/03/05 13:43:09.883428] 1: message length: 6 put message: 0x010231324142
Elapsed time = 0.388001 sec

$ mqpgf -qm HM8A -q LQ1 **-raw > redirect.msg**
Elapsed time = 0.148884 sec

$ od -x redirect.msg
0000000   0102 3132 4142
0000006

4.14 Write a message read from a queue to standard output(Hexadecimal notation)

Write get messages to stdout in hexadecimal notation. Ignored if specified with the "-o" or "-g" options. (Effective only when writing to standard output)

mqpgf -qm <qmgr> -q <queue> **-hex** -r

-hex: Output in hexadecimal notation

(sample options)
-r: Read a message repeatedly until a queue become empty.

Ex. 4.14.1 Write binary data to stdout in hexadecimal notation.

$ mqpgf -qm HM5A -q LQ1 -mx 010231324142
[20/03/05 14:06:08.975989] 1: message length: 6 put message: 0x010231324142
Elapsed time = 0.480365 sec
$ mqpgf -qm HM5A -q LQ1 -mx 080938396162
[20/03/05 14:06:19.078512] 1: message length: 6 put message: 0x080938396162
Elapsed time = 0.194654 sec
%mqpgf -qm HM5A -q LQ1 **-hex** -r
[20/03/05 14:06:27.181172] 1: message length: 6 get message : **0x010231324142**
[20/03/05 14:06:27.181743] 2: message length: 6 get message : **0x080938396162**
no message available : LQ1 CompCd=02 ReasonCd=2033
Elapsed time = 0.207737 sec

4.15 Create and put a message in pcf format

By writing the PCF definition in a plain text file as shown below, a binary PCF format can be created and put into the specified queue.
When putting, the created format is also displayed on the standard output. CCSID can be set individually for the following parameter structures. Normally, only the displayable code is displayed, but if you specify a different ccsid for the platform, it will not be displayed normally. In that case, you can display the parameter part of the character string in hexadecimal notation by specifying -hex.

MQCFT_STRING: String parameter
MQCFT_STRING_FILTER: String filter parameter
MQCFT_STRING_LIST: String list parameter

```
#-----------------------
# MQMD
#-----------------------
# Version <1 or 2>
#MD_VERSION=1
MD_VERSION=2

# MsgType <REQUEST or REPLY or DATAGRAM or REPORT>
#MD_MSGTYPE=REQUEST
#MD_MSGTYPE=REPLY
MD_MSGTYPE=DATAGRAM
#MD_MSGTYPE=REPORT

# Format <ADMIN or EVENT or PCF>
#MD_FORMAT=ADMIN
MD_FORMAT=EVENT
#MD_FORMAT=PCF

# ReplyToQ
MD_REPLYTOQ=<Reply To Queue>

# ReplyToQMgr
MD_REPLYTOQMGR=<Reply To Queue Manager>

#-----------------------
# MQCFH
# PCF hedder
#-----------------------
#Type <COMMAND or COMMAND_XR or RESPONSE or XR_MSG or XR_ITEM
or XR_SUMMARY or USER or NONE or EVENT or TRACE_ROUTE or REPOR
T or GROUP or STATISTICS or ACCOUNTING or APP_ACTIVITY>
```

```
#MQCFH_TYPE=COMMAND
#MQCFH_TYPE=COMMAND_XR
#MQCFH_TYPE=RESPONSE
#MQCFH_TYPE=XR_MSG
#MQCFH_TYPE=XR_ITEM
#MQCFH_TYPE=XR_SUMMARY
MQCFH_TYPE=USER
#MQCFH_TYPE=NONE
#MQCFH_TYPE=EVENT
#MQCFH_TYPE=TRACE_ROUTE
#MQCFH_TYPE=REPORT
#MQCFH_TYPE=GROUP
#MQCFH_TYPE=STATISTICS
#MQCFH_TYPE=ACCOUNTING
#MQCFH_TYPE=APP_ACTIVITY

# Version <MQCFH_VERSION_1 or MQCFH_VERSION_2 or MQCFH_VERSION_
3 or MQCFH_CURRENT_VERSION>
MQCFH_VERSION=MQCFH_VERSION_1
MQCFH_VERSION=MQCFH_VERSION_2
MQCFH_VERSION=MQCFH_VERSION_3
MQCFH_VERSION=MQCFH_CURRENT_VERSION

# Command <MGR or PERFM or CHANNEL> or <command number>
# For Event Message the following parameters are available.
#MQCFH_COMMAND=MGR
#MQCFH_COMMAND=PERFM
#MQCFH_COMMAND=CHANNEL
#
# For non-event message, specify the command number directly.
# In the following example '28' is the start channel
# $ grep MQCMD_START_CHANNEL /usr/mqm/inc/cmqcfc.h
# #define MQCMD_START_CHANNEL          28
MQCFH_COMMAND=28

# MsgSeqNumber <1 or 2>
MQCFH_MSGSEQNUM=1
#MQCFH_MSGSEQNUM=2

#Control
MQCFH_CTRL=MQCFC_LAST
#MQCFH_CTRL=MQCFC_NOT_LAST

# CompCode <OK or WARNING or FAILED>
MQCFH_COMPCODE=OK
```

#MQCFH_COMPCODE=WARNING
#MQCFH_COMPCODE=FAILED

# Reason < number >
MQCFH_REASON=0

ParameterCount(for MQCFGR(MQCFT_GROUP))
If you specify MQCFGR, explicitly specify that number in the MQCFH
ParameterCount. You can omit this parameter if you are not using MQCFGR. If
omitted, mqpgf automatically sets the number of subsequent parameters.

#------------------------
MQCFST/MQCFIN etc
PCF data
#------------------------
MQCFGR Structure - PCF Group Parameter
MQCFGR=Parameter,ParameterCount
#
MQCFBS - PCF Byte String Parameter
MQCFBS=Parameter,StringLength,String
#
MQCFBF - PCF Byte String Filter Parameter
MQCFBF=Parameter,Operator,FilterValueLength,FilterValue
#
MQCFST - PCF String Parameter
MQCFST=Parameter,CodedCharSetId,StringLength,String
#
MQCFSF - PCF String Filter Parameter
MQCFSF=Parameter,Operator,CodedCharSetId,FilterValueLength,FilterValue
#
MQCFSL - PCF String List Parameter
MQCFSL=Parameter,CodedCharSetId,Count,StringLength,String1,String2,...
#
MQCFIN - PCF Integer Parameter
MQCFIN=Parameter,Value
#
MQCFIF - PCF Integer Filter Parameter
MQCFIF=Parameter,Operator,FilterValue
#
MQCFIL - PCF Integer List Parameter
MQCFIL=Parameter,Count,Value1,Value2,...
#
MQCFIN64 - 64 bit Integer Parameter
MQCFIN64=Parameter,Value
#

```
# MQCFIL64 - 64 bit Integer List Parameter
# MQCFIL64=Parameter,Count,Value1,Value2,...
#------------------------
#
# < Description of each parameter field >
#
# Parameter: Parameter ID
#
# e.g. For MQCACH_CHANNEL_NAME, specify 3501.
#
# $ grep MQCMD_START_CHANNEL /usr/mqm/inc/cmqcfc.h.h
# #define MQCACH_CHANNEL_NAME          3501
#
# ParameterCount: Number of PCF parameters contained in MQCFGR (MQCFT_
GROUP)
# StringLength: String length
# String: Stirings
# Operator: Specify a numerical value corresponding to MQCFOP_*.
#
# Operator:
# MQCFOP_LESS         1
# MQCFOP_EQUAL        2
# MQCFOP_NOT_GREATER 3
# MQCFOP_GREATER      4
# MQCFOP_NOT_EQUAL    5
# MQCFOP_NOT_LESS     6
#
# FilterValueLength: Filter Value Length
# FilterValue: Fileter Value
# CodedCharSetId: CCSID of Strings
# Count: Number of list
# Value: Integer Value
# Range that can be specified for Value / FilterValue of MQCFIN, MQCFIF, MQ
CFIL
# 0x00000000 - 0xffffffff
# -2147483648 - (+)2147483647
# Range that can be specified for Value of MQCFIN64, MQCFIL64
# The most significant bit can not be set(Negative value can not be specified in
hexadecimal notation).
# 0x00000000 00000000 - 0x7fffffff ffffffff
# -9223372036854775808 - (+)9223372036854775807
```

mqpgf -qm <qmgr> -q <queue> **-pcf <pcf format file>**

-pcf: PCF format definition file

Ex. 4.15.1 Create and put a PCF message in user defined format.

$ cat sample1.def
MD_VERSION=2

MD_MSGTYPE=DATAGRAM

MD_FORMAT=EVENT

MD_REPLYTOQ=PCF.ANSWER
MD_REPLYTOQMGR=TESTQM

MQCFH_TYPE=USER

MQCFH_VERSION=MQCFH_VERSION_3

MQCFH_COMMAND=99

MQCFH_MSGSEQNUM=1

MQCFH_CTRL=MQCFC_LAST

MQCFH_COMPCODE=OK

MQCFH_REASON=0

MQCFBS=1111,10,1234567890
MQCFBF=2222,1,5,0x3141324233
MQCFST=3501,943,17,TESTQM.to.TESTQM2
MQCFSF=4444,2,1208,30,123456789012345678901234567890
MQCFSL=5555,930,5,3,0xf1f1f1,0xf1f2f3,0xf3f3f3,0xf5f6f7,0xf5f5f5
MQCFIN=6666,1234567890
MQCFIF=7777,6,-3
MQCFIL=8888,3,1234,0xffffffff,5678
MQCFIN64=9999,0x7fffffffffffffff
MQCFIL64=1234,5,4294967294,0x00000000fffffffb,-5,0x7ffffffffffffffb,4294967290
$
$ mqpgf -qm HM8E2 -q LQ1 -pcf sample1.def
Command ： 99
Id ： 1111, MQCFT_BYTE_STRING ： 10, 1234567890
Id ： 2222, MQCFT_BYTE_STRING_FILTER ： 1 5 1A2B3
Id ： 3501, MQCFT_STRING ： 943 17 TESTQM.to.TESTQM2

Id ： 4444, MQCFT_STRING_FILTER ： 2 1208 30 12345678901234567890123456 7 890
Id ： 5555, MQCFT_STRING_LIST ： 930 5 3 [...],[...],[...],[...],[...]
Id ： 6666, MQCFT_INTEGER ： 1234567890
Id ： 7777, MQCFT_INTEGER_FILTER ： 6 -3
Id ： 8888, MQCFT_INTEGER_LIST ： 3 [1234],[-1],[5678]
Id ： 9999, MQCFT_INTEGER64 ： 9223372036854775807
Id ： 1234, MQCFT_INTEGER64_LIST ： 5 [4294967294],[4294967291],[-5],[9223372 036854775803],[4294967290]
[20/03/06 09:24:05.389505] 1: put from sample1.def
Elapsed time = 0.109636 sec

* In this example, Japanese EBCDIC (ccsid 930) is specified in MQCFT_STRING _LIST, so the character string is not displayed correctly. When -hex is specified, i t is displayed in hexadecimal notation as shown below.

$ mqpgf -qm HM8E2 -q LQ1 -pcf sample1.def -hex
Command ： 99
Id ： 1111, MQCFT_BYTE_STRING ： 10, 1234567890
Id ： 2222, MQCFT_BYTE_STRING_FILTER ： 1 5 1A2B3
Id ： 3501, MQCFT_STRING ： 943 17 0x54455354514D2E746F2E54455354514D32
Id ： 4444, MQCFT_STRING_FILTER ： 2 1208 30 0x31323334353637383930313233 3343536373839303132333435363738393 0
Id ： 5555, MQCFT_STRING_LIST ： 930 5 3 [0xF1F1F1],[0xF1F2F3],[0xF3F3F3],[0 xF5F6F7],[0xF5F5F5]
Id ： 6666, MQCFT_INTEGER ： 1234567890
Id ： 7777, MQCFT_INTEGER_FILTER ： 6 -3
Id ： 8888, MQCFT_INTEGER_LIST ： 3 [1234],[-1],[5678]
Id ： 9999, MQCFT_INTEGER64 ： 9223372036854775807
Id ： 1234, MQCFT_INTEGER64_LIST ： 5 [4294967294],[4294967291],[-5],[9223372 036854775803],[4294967290]
[20/03/06 09:24:34.851645] 1: put from sample1.def
Elapsed time = 0.071464 sec

* Even when browsing a PCF format, specifying "-hex" allows the specified items to be displayed in hexadecimal notation.

$ mqpgf -qm HM8E2 -q LQ1 -brv -hex
message number: 1
*StrucId[MD] Version[2] Report[0x00000000] MsgType[8] Expiry[-1] Feedback[0]
Encoding[273] CodedCharSetId[819] Format[MQEVENT] Priority[0] Persistence[0]
 MsgId[0x414D5120484D384532202020202020205E61952120002602] CorrelId[0x000
00000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[P
CF.ANSWER   ] ReplyToQMgr[TESTQM
                        ] UserIdentifier[mqm   ] AccountingTo

ken[0x0534343033310000000000000000000000000000000000000000000006]
ApplIdentityData[   ] PutApplType[13] PutApplNam
e[mqpgf   ] PutDate[20120306] PutTime[00240539] ApplOrigi
nData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0x00000000] OriginalLength[-1]

*MQCFH(MQCFT_USER) Type[8] StrucLength[36] Version[3] Command[99] MsgS
eqNumber[1] Control[1] CompCode[0] Reason[0] ParameterCount[10]
(MQCFT_BYTE_STRING) Type[9] StrucLength[28] Parameter[1111] StringLength
[10] String[0x31323334353637383930]
(MQCFT_BYTE_STRING_FILTER) Type[15] StrucLength[28] Parameter[2222] Ope
rator[1] FilterValueLength[5] FilterValue[0x3141324233]
**(MQCFT_STRING) Type[4] StrucLength[40] Parameter[3501] CodedCharSetId[943]
StringLength[17] String[0x54455354514D2E746F2E54455354514D32]**
**(MQCFT_STRING_FILTER) Type[14] StrucLength[56] Parameter[4444] Operator
[2] CodedCharSetId[1208] FilterValueLength[30] FilterValue[0x31323334353637383
930313233343536373838393031323334353637383930]**
**(MQCFT_STRING_LIST) Type[6] StrucLength[40] Parameter[5555] CodedCharSetI
d[930] Count[5] StringLength[3] String1[0xF1F1F1] String2[0xF1F2F3] String3[0x
F3F3F3] String4[0xF5F6F7] String5[0xF5F5F5]**
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[6666] Value[1234567890]
(MQCFT_INTEGER_FILTER) Type[13] StrucLength[20] Parameter[7777] Operator
[6] FilterValue[-3]
(MQCFT_INTEGER_LIST) Type[5] StrucLength[28] Parameter[8888] Count[3] Val
ues[1234,-1,5678]
(MQCFT_INTEGER64) Type[23] StrucLength[24] Parameter[9999] Value[92233720
36854775807]
(MQCFT_INTEGER64_LIST) Type[25] StrucLength[56] Parameter[1234] Count[5]
Values[4294967294,4294967291,-5,9223372036854775803,4294967290]

Elapsed time = 0.075103 sec

* The following is an example using MQCFGR (MQCFT_GROUP: PCF Group Pa
rameter).
$ cat sample1_2.def
MD_VERSION=2
MD_MSGTYPE=DATAGRAM
MD_FORMAT=EVENT

MQCFH_TYPE=EVENT
MQCFH_VERSION=MQCFH_VERSION_3
MQCFH_COMMAND=99

MQCFH_MSGSEQNUM=1

MQCFH_CTRL=MQCFC_LAST

MQCFH_COMPCODE=OK

MQCFH_REASON=2413

# ParameterCount(for MQCFGR(MQCFT_GROUP))
MQCFH_PARAMCOUNT=2

#----------------
Command Event
#----------------
1   2   3   4   5   6
1234567890123456789012345678901234567890123456789012345
7890
MQCFGR=8001,2
MQCFST=3045,819,12,mqm
MQCFIN=1011,3
MQCFGR=8002,1
MQCFIN=99,1
$
$ mqpgf -qm PLQMTS -q LQ1 -pcf sample1_2.def
Command : 99
Id : 8001, MQCFT_GROUP : 2
Id : 3045, MQCFT_STRING : 819 12 mqm
Id : 1011, MQCFT_INTEGER : 3
Id : 8002, MQCFT_GROUP : 1
Id : 99, MQCFT_INTEGER : 1
[20/11/05 19:55:53.088531] 1: put from sample1_2.def
MQCMIT success : CompCd=00 ReasonCd=00
Elapsed time = 0.081116 sec

$ mqpgf -qm PLQMTS -q LQ1 -dpv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[
273] CodedCharSetId[819] Format[MQEVENT] Priority[0] Persistence[0] MsgId[0
x414D5120504C514D54532020202020205FA3D34120002504] CorrelId[0x0000000000
00000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                   ] ReplyToQMgr[PLQMTS
           ] UserIdentifier[mqm   ] AccountingToken[0x05343
4303331000000000000000000000000000000000000000000000000000006] ApplIdentity
Data[   ] PutApplType[13] PutApplName[mqpgf
          ] PutDate[20201105] PutTime[10555308] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*MQCFH(MQCFT_EVENT) Type[7] StrucLength[36] Version[3] Command[99] Msg SeqNumber[1] Control[1] CompCode[0] Reason[2413] ParameterCount[2]
(MQCFT_GROUP) Type[20] StrucLength[16] Parameter[8001] ParameterCount[2]
(MQCFT_STRING) Type[4] StrucLength[32] Parameter[3045] CodedCharSetId[819]
 StringLength[12] String[mqm   ]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1011] Value[3]
(MQCFT_GROUP) Type[20] StrucLength[16] Parameter[8002] ParameterCount[1]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[99] Value[1]

MQCMIT success ： CompCd=00 ReasonCd=00
Elapsed time = 0.076464 sec

Ex. 4.15.2 Send the start channel command to the command server.

* When sending PCF to the command server, specify MD_REPLYTOQMGR and MD_REPLYTOQ for the response of the command server, and also define the queue.

$ cat sample2.def
MD_VERSION=2

MD_MSGTYPE=REQUEST

MD_FORMAT=ADMIN

MD_REPLYTOQ=PCF.ANSWER
MD_REPLYTOQMGR=HM8E2

MQCFH_TYPE=COMMAND

MQCFH_VERSION=MQCFH_VERSION_1

# start channel
MQCFH_COMMAND=28

MQCFH_MSGSEQNUM=1

MQCFH_CTRL=MQCFC_LAST

MQCFH_COMPCODE=OK

MQCFH_REASON=0

MQCFST=3501,943,8,TO.HM8M1

$ mqpcf chs -qm HM8E2 -c TO.HM8M1 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TO.HM8M1) **STATUS(STOPPED)** CHLT
YPE(CLUSSDR) CONNAME(remotehost(1414)) RQMNAME(HM8M1) STOPREQ(N
O) SUBSTATE(OTHER) XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
$
$ mqpgf -qm HM8E2 -q SYSTEM.ADMIN.COMMAND.QUEUE -pcf sample2.def
Command：28
Id：3501, MQCFT_STRING：943 8 TO.HM8M1
[20/03/06 10:21:37.857249] 1: put from sample2.def
Elapsed time = 0.055745 sec
$
$ mqpcf chs -qm HM8E2 -c TO.HM8M1 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TO.HM8M1) **STATUS(RUNNING)** CHLT
YPE(CLUSSDR) CONNAME(remotehost(1414)) RQMNAME(HM8M1) STOPREQ(N
O) SUBSTATE(MQGET) XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)

* The following is the response message of the start channel command returned
by the command server.
$ mqpgf -qm HM8E2 -q PCF.ANSWER -brv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[2] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[819] Format[MQADMIN] Priority[0] Persistence[0] MsgId[0
x414D5120484D384532202020202020205E61952120001E12] CorrelId[0x414D512048
4D384532202020202020205E61952120003002] BackoutCount[0] ReplyToQ[
                                        ] ReplyToQMgr[HM8E2
             ] UserIdentifier[mqm   ] AccountingToken[0x05343
430333310000000000000000000000000000000000000000000000006] ApplIdentity
Data[   ] PutApplType[7] PutApplName[amqpcsea
          ] PutDate[20120306] PutTime[01213802] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*MQCFH(MQCFT_RESPONSE) Type[2] StrucLength[36] Version[1] **Command[28]**
MsgSeqNumber[1] Control[1] CompCode[0] Reason[0] ParameterCount[0]

Elapsed time = 0.061344 sec

4.16 Call MQSET

   Invoke MQSET function for a specified queue. You can specify multiple selector and attribute combinations.
   MQSET can change attributes only for queues. You cannot change the attributes of other objects, such as processes and queue managers.
   Also, model queues cannot be changed, and cluster queues require a local instance.

mqpgf -qm <qmgr> -q <queue> -set: selecter:attribute,..:..,(e.g. MQIA_INHIBIT_GET:MQQA_GET_ALLOWED,..:..)

-inq: selector

   Selectors in the table below are available.

Table 4.16.1 MQSET attribute selector for queue

Selector	Description(Possible values)
MQCA_TRIGGER_DATA	Trigger data (MQ_TRIGGER_DATA_LENGTH).
MQIA_DIST_LISTS	Distribution list support. (MQDL_SUPPORTED/ MQDL_NOT_SUPPORTED)
MQIA_INHIBIT_GET	Whether get operations are allowed. (MQQA_GET_INHIBITED/MQQA_GET_ALLOWED)
MQIA_INHIBIT_PUT	Whether put operations are allowed. (MQQA_PUT_INHIBITED/MQQA_PUT_ALLOWED)
MQIA_TRIGGER_CONTROL	Trigger control. (MQQA_PUT_INHIBITED/MQQA_PUT_ALLOWED)
MQIA_TRIGGER_DEPTH	Trigger depth. (MQTC_ON/MQTC_OFF)
MQIA_TRIGGER_MSG_PRIORITY	Threshold message priority for triggers. (Positive integer)
MQIA_TRIGGER_TYPE	Trigger type. (MQTT_NONE/ MQTT_FIRST/ MQTT_EVERY/ MQTT_DEPTH)

Ex. 4.16.1 Example of specifying a single parameter to MQSET().

$ mqpgf -qm HM8E2 -q LQ1 **-set MQIA_INHIBIT_GET:MQQA_GET_INHIBITED**
[20/03/06 11:05:57.627602] 1: MQSET MQIA_INHIBIT_GET:MQQA_GET_INHIBIT
ED
Elapsed time = 0.308728 sec

$ mqpcf que -qm HM8E2 -q LQ1 GET
1: QUEUE(LQ1) TYPE(QLOCAL) GET(DISABLED)

Ex. 4.16.2 Example of specifying multiple parameters to MQSET().

$ mqpcf que -qm HM8E2 -q LQ1 GET PUT TRIGDATA DISTL
1: QUEUE(LQ1) TYPE(QLOCAL) DISTL(NO) GET(DISABLED) PUT(ENABLED)
TRIGDATA(triger data)

$ mqpgf -qm HM8E2 -q LQ1 **-set MQIA_INHIBIT_GET:MQQA_GET_ALLOWED,**
MQIA_INHIBIT_PUT:MQQA_PUT_ALLOWED,MQCA_TRIGGER_DATA:" ",MQIA_
DIST_LISTS:MQDL_SUPPORTED
[20/03/06 11:15:36.565387] 1: MQSET MQIA_INHIBIT_GET:MQQA_GET_ALLOWE
D,MQIA_INHIBIT_PUT:MQQA_PUT_ALLOWED,MQCA_TRIGGER_DATA: ,MQIA_
DIST_LISTS:MQDL_SUPPORTED
Elapsed time = 0.089051 sec

$ mqpcf que -qm HM8E2 -q LQ1 GET PUT TRIGDATA DISTL
1: QUEUE(LQ1) TYPE(QLOCAL) DISTL(YES) GET(ENABLED) PUT(ENABLED)
TRIGDATA()

4.17 Call MQINQ

Invokde MQINQ() to query the attributes of a specified queue (local, remote, alias), name list, process and queue manager.

mqpgf -qm <qmgr> **-inq: selecter**(e.g. MQCA_CHANNEL_AUTO_DEF_EXIT, MQCA_CLUSTER_WORKLOAD_DATA,..) **MQOT_Q_MGR**
mqpgf -qm <qmgr> **-q <queue> -inq: selecter**(e.g. MQCA_ALTERATION_DATE,MQIA_CLWL_Q_PRIORITY,..)
mqpgf -qm <qmgr> **-nl <namelist> -inq: selecter**(e.g. MQIA_NAMELIST_TYPE,MMQCA_NAMES,..) **MQOT_NAMELIST**
mqpgf -qm <qmgr> **-p <process> -inq: selecter**(e.g. MQCA_APPL_ID, MQCA_ENV_DATA,..) **MQOT_PROCESS**

-q: queue name (for queue attributes)
-nl: namelist name (for name list attributes)
-p: process name (for process attributes)
-inq: selector
MQOT_Q_MGR (for queue manager attributes)
MQOT_NAMELIST (for name list attributes)
MQOT_PROCESS (for process attributes)

Selectors in the table below are available.

Table 4.17.1 MQINQ attribute selectors for queue.

Selector	Description	Note
MQCA_ALTERATION_DATE	Date of most-recent alteration	
MQCA_ALTERATION_TIME	Time of most-recent alteration	
MQCA_BACKOUT_REQ_Q_NAME	Excessive backout requeue name	
MQCA_BASE_Q_NAME	Name of queue that alias resolves to	
MQCA_CF_STRUC_NAME	Coupling-facility structure name	z/OS
MQCA_CLUS_CHL_NAME	Name of the cluster-sender channel that uses this queue as a transmission queue.	
MQCA_CLUSTER_NAME	Cluster name	
MQCA_CLUSTER_NAMELIST	Cluster namelist	
MQCA_CREATION_DATE	Queue creation date	

Table 4.17.1 MQINQ attribute selectors for queue.

Selector	Description	Note
MQCA_CREATION_TIME	Queue creation time	
MQCA_INITIATION_Q_NAME	Initiation queue name	
MQCA_PROCESS_NAME	Name of process definition	
MQCA_Q_DESC	Queue description	
MQCA_Q_NAME	Queue name	
MQCA_REMOTE_Q_MGR_NAME	Name of remote queue manager	
MQCA_REMOTE_Q_NAME	Name of remote queue as known on remote queue manager	
MQCA_STORAGE_CLASS	Name of storage class	z/OS
MQCA_TRIGGER_DATA	Trigger data	
MQCA_XMIT_Q_NAME	Transmission queue name	
MQIA_ACCOUNTING_Q	Controls collection of accounting data for queue	Not z/OS
MQIA_BACKOUT_THRESHOLD	Backout threshold	
MQIA_CLWL_Q_PRIORITY	Priority of queue	
MQIA_CLWL_Q_RANK	Rank of queue	
MQIA_CLWL_USEQ	Use remote queues	
MQIA_CURRENT_Q_DEPTH	Number of messages on queue	
MQIA_DEF_BIND	Default binding	
MQIA_DEF_INPUT_OPEN_OPTION	Default open-for-input option	
MQIA_DEF_PERSISTENCE	Default message persistence	
MQIA_DEF_PRIORITY	Default message priority	
MQIA_DEFINITION_TYPE	Queue definition type	
MQIA_DIST_LISTS	Distribution list support	Not z/OS
MQIA_HARDEN_GET_BACKOUT	Whether to harden backout count	
MQIA_INDEX_TYPE	Type of index maintained for queue	z/OS

Table 4.17.1 MQINQ attribute selectors for queue.

Selector	Description	Note
MQIA_INHIBIT_GET	Whether get operations are allowed	
MQIA_INHIBIT_PUT	Whether put operations are allowed	
MQIA_MAX_MSG_LENGTH	Maximum message length	
MQIA_MAX_Q_DEPTH	Maximum number of messages allowed on queue	
MQIA_MSG_DELIVERY_SEQUENCE	Whether message priority is relevant	
MQIA_NPM_CLASS	Level of reliability for nonpersistent messages	
MQIA_OPEN_INPUT_COUNT	Number of MQOPEN calls that have the queue open for input	
MQIA_OPEN_OUTPUT_COUNT	Number of MQOPEN calls that have the queue open for output	
MQIA_PROPERTY_CONTROL	Property control attribute	
MQIA_Q_DEPTH_HIGH_EVENT	Control attribute for queue depth high events	Not z/OS
MQIA_Q_DEPTH_HIGH_LIMIT	High limit for queue depth	Not z/OS
MQIA_Q_DEPTH_LOW_EVENT	Control attribute for queue depth low events	Not z/OS
MQIA_Q_DEPTH_LOW_LIMIT	Low limit for queue depth	Not z/OS
MQIA_Q_DEPTH_MAX_EVENT	Control attribute for queue depth max events	Not z/OS
MQIA_Q_SERVICE_INTERVAL	Limit for queue service interval	Not z/OS
MQIA_Q_SERVICE_INTERVAL_EVENT	Control attribute for queue service interval events	Not z/OS
MQIA_Q_TYPE	Queue type	
MQIA_QSG_DISP	Queue-sharing group disposition	z/OS

Table 4.17.1 MQINQ attribute selectors for queue.

Selector	Description	Note
MQIA_RETENTION_INTERVAL	Queue retention interval	
MQIA_SCOPE	Queue definition scope	Not z/OS
MQIA_SHAREABILITY	Whether queue can be shared for input	
MQIA_STATISTICS_Q	Controls collection of statistics data for queue	Not z/OS
MQIA_TRIGGER_CONTROL	Trigger control	
MQIA_TRIGGER_DEPTH	Trigger depth	
MQIA_TRIGGER_MSG_PRIORITY	Threshold message priority for triggers	
MQIA_TRIGGER_TYPE	Trigger type	
MQIA_USAGE	Usage	

Table 4.17.2 MQINQ attribute selectors for namelists.

Selector	Descrition	Note
MQCA_ALTERATION_DATE	Date of most-recent alteration	
MQCA_ALTERATION_TIME	Time of most-recent alteration	
MQCA_NAMELIST_DESC	Namelist description	
MQCA_NAMELIST_NAME	Name of namelist object	
MQIA_NAMELIST_TYPE	Namelist type	z/OS
MQCA_NAMES	Names in the namelist	
MQIA_NAME_COUNT	Number of names in the namelist	
MQIA_QSG_DISP	Queue-sharing group disposition	z/OS

Table 4.17.3 MQINQ attribute selectors for process definitions.

Selector	Descrition	Note
MQCA_ALTERATION_DATE	Date of most-recent alteration	

Table 4.17.3 MQINQ attribute selectors for process definitions.

Selector	Description	Note
MQCA_ALTERATION_TIME	Time of most-recent alteration	
MQCA_APPL_ID	Application identifier	
MQCA_ENV_DATA	Environment data	
MQCA_PROCESS_DESC	Description of process definition	
MQCA_PROCESS_NAME	Name of process definition	
MQCA_USER_DATA	User data	
MQIA_APPL_TYPE	Application type	
MQIA_QSG_DISP	Queue-sharing group disposition	z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Description	Note
MQCA_ALTERATION_DATE	Date of most-recent alteration	
MQCA_ALTERATION_TIME	Time of most-recent alteration	
MQCA_CHANNEL_AUTO_DEF_EXIT	Automatic channel definition exit name	
MQCA_CHINIT_SERVICE_PARM	Reserved for use by IBM	
MQCA_CLUSTER_WORKLOAD_DATA	Data passed to cluster workload exit	
MQCA_CLUSTER_WORKLOAD_EXIT	Name of cluster workload exit	
MQCA_COMMAND_INPUT_Q_NAME	System command input queue name	
MQCA_DEAD_LETTER_Q_NAME	Name of dead-letter queue	
MQCA_DEF_XMIT_Q_NAME	Default transmission queue name	
MQCA_DNS_GROUP	Name of the group for the TCP listener that handles inbound transmissions for the queue-sharing group to join. The name applies when using Workload Manager Dynamic Domain	z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Description	Note
	Name Services.	
MQCA_IGQ_USER_ID	Intra-group queuing user identifier	z/OS
MQCA_INSTALLATION_DESC	Description of the associated installation	Not z/OS. Not IBM i
MQCA_INSTALLATION_NAME	Name of the installation associated with the queue manager	Not z/OS. Not IBM i
MQCA_INSTALLATION_PATH	Path where the associated IBM MQ is installed	Not z/OS. Not IBM i
MQCA_LU_GROUP_NAME	Generic LU name for the LU 6.2 listener that handles inbound transmissions for the queue-sharing group to use	z/OS
MQCA_LU_NAME	Name of the LU to use for outbound LU 6.2 transmissions. Set this name to the same LU that the listener uses for inbound transmissions	z/OS
MQCA_LU62_ARM_SUFFIX	Suffix of the SYS1.PARMLIB member APPCPM xx, that nominates the LUADD for this channel initiator	z/OS
MQCA_PARENT	Name of a hierarchically connected queue manager that is nominated as the parent of this queue manager	
MQCA_Q_MGR_DESC	Queue manager description	
MQCA_Q_MGR_IDENTIFIER	Queue-manager identifier (H)	
MQCA_Q_MGR_NAME	Name of local queue manager	

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Description	Note
MQCA_QSG_NAME	Queue-sharing group name	z/OS
MQCA_REPOSITORY_NAME	Name of cluster for which queue manager provides repository services	
MQCA_REPOSITORY_NAMELIST	Name of namelist object containing names of clusters for which queue manager provides repository services	
MQCA_TCP_NAME	Name of the TCP/IP system that you are using	z/OS
MQIA_ACCOUNTING_CONN_OVERWRITE	Override accounting settings	Not z/OS
MQIA_ACCOUNTING_INTERVAL	How often to write intermediate accounting records	Not z/OS
MQIA_ACCOUNTING_MQI	Controls collection of accounting information for MQI data	Not z/OS
MQIA_ACCOUNTING_Q	Controls collection of accounting information for queues	Not z/OS
MQIA_ACTIVE_CHANNELS	Maximum number of channels that can be active at any time	z/OS
MQIA_ADOPTNEWMCA_CHECK	Elements that are checked to determine whether to adopt an MCA. The check is performed when a new inbound channel is detected that has the same name as an MCA that is already active.	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	Amount of time, in seconds, that the new channel waits for the orphaned channel to end	Not z/OS
MQIA_ADOPTNEWMCA_TYPE	Whether to restart an orphaned instance of an MCA of a particular channel type automatically when a new inbound channel request matching the AdoptNewMCACheck	z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Descrition	Note
	parameters is detected	
MQIA_AUTHORITY_EVENT	Control attribute for authority events	Not z/OS
MQIA_BRIDGE_EVENT	Control attribute for IMS bridge events	z/OS
MQIA_CHANNEL_AUTO_DEF	Control attribute for automatic channel definition	Not z/OS
MQIA_CHANNEL_AUTO_DEF_EVENT	Control attribute for automatic channel definition events	Not z/OS
MQIA_CHANNEL_EVENT	Control attribute for channel events	
MQIA_CHINIT_ADAPTERS	Number of adapter subtasks to use for processing IBM MQ calls	z/OS
MQIA_CHINIT_DISPATCHERS	Number of dispatchers to use for the channel initiator	z/OS
MQIA_CHINIT_TRACE_AUTO_START	Whether to start channel initiator trace automatically	z/OS
MQIA_CHINIT_TRACE_TABLE_SIZE	Size of the trace data space (in MB) of the channel initiator	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	Cluster workload length.	
MQIA_CLWL_MRU_CHANNELS	Number of most recently used channels for cluster workload balancing	
MQIA_CLWL_USEQ	Use remote queues	
MQIA_CODED_CHAR_SET_ID	Coded character set identifier	
MQIA_COMMAND_EVENT	Control attribute for command events	
MQIA_COMMAND_LEVEL	Command level supported by queue manager	
MQIA_CONFIGURATION_EVENT	Control attribute for configuration events	Not z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Description	Note
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	Default transmission queue type to be used for cluster-sender channels.	
MQIA_DIST_LISTS	Distribution list support	Not z/OS
MQIA_DNS_WLM	Whether the TCP listener that handles inbound transmissions for the queue-sharing group registers with Workload Manager for Dynamic Domain Name Services	z/OS
MQIA_EXPIRY_INTERVAL	Interval between scans for expired messages	z/OS
MQIA_GROUP_UR	Control attribute for whether GROUP units of recovery are enabled for this queue manager. The GROUP unit of recovery disposition is only available if the queue manager is a member of a queue-sharing group	z/OS
MQIA_IGQ_PUT_AUTHORITY	Intra-group queuing put authority	z/OS
MQIA_INHIBIT_EVENT	Control attribute for inhibit events	Not z/OS
MQIA_INTRA_GROUP_QUEUING	Intra-group queuing support	z/OS
MQIA_LISTENER_TIMER	Time interval (in seconds) between IBM MQ attempts to restart the listener if APPC or TCP/IP failed.	z/OS
MQIA_LOCAL_EVENT	Control attribute for local events	Not z/OS
MQIA_LOGGER_EVENT	Control attribute for inhibit events	Not z/OS
MQIA_LU62_CHANNELS	Maximum number of channels that can be current, or clients that can be connected, using the	z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Description	Note
	LU 6.2 transmission protocol	
MQIA_MSG_MARK_BROWSE_INTERVAL	Time interval (in milliseconds) after which the queue manager can automatically remove a mark from browse messages	
MQIA_MAX_CHANNELS	Maximum number of channels that can be current (including server-connection channels with connected clients)	z/OS
MQIA_MAX_HANDLES	Maximum number of handles	
MQIA_MAX_MSG_LENGTH	Maximum message length	
MQIA_MAX_PRIORITY	Maximum priority	
MQIA_MAX_UNCOMMITTED_MSGS	Maximum number of uncommitted messages within a unit of work	
MQIA_OUTBOUND_PORT_MAX	With MQIA_OUTBOUND_PORT_MIN, defines range of port numbers to use when binding outgoing channels	z/OS
MQIA_OUTBOUND_PORT_MIN	With MQIA_OUTBOUND_PORT_MAX, defines range of port numbers to use when binding outgoing channels	z/OS
MQIA_PERFORMANCE_EVENT	Control attribute for performance events	Not z/OS
MQIA_PLATFORM	Platform on which the queue manager resides	
MQIA_PROT_POLICY_CAPABILITY	Indicates whether security capabilities of IBM MQ Advanced Message Security are available for a queue manager.	
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	The number of attempts to reprocess a failed command	

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Descrition	Note
	message under sync point	
MQIA_PUBSUB_MODE	Whether the publish/subscribe engine and the queued publish/subscribe interface are running. Applications to publish or subscribe using the application programming interface require the publish/subscribe engine. Queues that are monitored by the queued publish/subscribe interface require the queued publish/subscribe interface to be running.	
MQIA_PUBSUB_NP_MSG	Whether to discard (or keep) an undelivered input message	
MQIA_PUBSUB_NP_RESP	Controls the behavior of undelivered response messages	
MQIA_PUBSUB_SYNC_PT	Whether only persistent (or all) messages are processed under sync point	
MQIA_QMGR_CFCONLOS	Specifies the action to be taken when the queue manager loses connectivity to the administration structure or any CF structures with CFCONLOS set to ASQMGR	z/OS
MQIA_RECEIVE_TIMEOUT	Approximately how long a TCP/IP channel waits to receive data, including heartbeats, from its partner, before returning to the inactive state. The value is numeric, qualified by MQIA_RECEIVE_TIMEOUT_TYPE.	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	Minimum time that a TCP/IP channel waits to receive data, including heartbeats, from its	z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Description	Note
	partner, before returning to the inactive state	
MQIA_RECEIVE_TIMEOUT_TYPE	Approximately how long a TCP/IP channel waits to receive data, including heartbeats, from its partner, before returning to the inactive state. MQIA_RECEIVE_TIMEOUT_TYPE is the qualifier applied to MQIA_RECEIVE_TIMEOUT.	z/OS
MQIA_REMOTE_EVENT	Control attribute for remote events	Not z/OS
MQIA_SECURITY_CASE	Case of security profiles	z/OS
MQIA_SSL_EVENT	Control attribute for channel events	
MQIA_SSL_FIPS_REQUIRED	Use only FIPS-certified algorithms for cryptography	
MQIA_SSL_RESET_COUNT	TLS key reset count	
MQIA_START_STOP_EVENT	Control attribute for start stop events	Not z/OS
MQIA_STATISTICS_AUTO_CLUSSDR	Controls collection of statistics monitoring information for cluster sender channels	
MQIA_STATISTICS_CHANNEL	Controls collection of statistics data for channels	
MQIA_STATISTICS_INTERVAL	How often to write statistics monitoring data	Not z/OS
MQIA_STATISTICS_MQI	Controls collection of statistics monitoring information for queue manager	Not z/OS
MQIA_STATISTICS_Q	Controls collection of statistics data for queues	Not z/OS
MQIA_SYNCPOINT	sync point availability	
MQIA_TCP_CHANNELS	Maximum number of channels	z/OS

Table 4.17.4 MQINQ attribute selectors for queue manager.

Selector	Descrition	Note
	that can be current, or clients that can be connected, using the TCP/IP transmission protocol	
MQIA_TCP_KEEP_ALIVE	Whether to use the TCP KEEPALIVE facility to check that the other end of the connection is still available	z/OS
MQIA_TCP_STACK_TYPE	Whether the channel initiator can use only the TCP/IP address space specified in TCPNAME, or can optionally bind to any selected TCP/IP address	z/OS
MQIA_TRACE_ROUTE_RECORDING	Controls recording of trace-route information	z/OS
MQIA_TREE_LIFE_TIME	Lifetime of unused non-administrative topics	
MQIA_TRIGGER_INTERVAL	Trigger interval	

Ex. 4.17.1 Query local queue attributes.

$ mqpgf -qm HM8E2 -q LQ1 **-inq MQCA_ALTERATION_DATE,MQCA_ALTERAT
ION_TIME,MQCA_BACKOUT_REQ_Q_NAME**
[20/03/06 11:29:07.891216] 1: ALTDATE(2012-03-06) ALTTIME(11.28.58) BOQNA
ME(BO4TQ)
Elapsed time = 0.059579 sec

Ex. 4.17.2 Query remote queue attributes.

$ mqpgf -qm HM8E2 -q RQ1 **-inq MQCA_REMOTE_Q_MGR_NAME,MQCA_REM
OTE_Q_NAME,MQCA_XMIT_Q_NAME**
[20/03/06 11:39:09.850786] 1: RQMNAME(HM8M1) RNAME(LQ1) XMITQ()
Elapsed time = 0.060893 sec

Ex. 4.17.3 Query alias queue attributes.

$ mqpgf -qm HM8E2 -q AQ1 **-inq MQCA_BASE_Q_NAME**
[20/03/06 11:43:57.045627] 1: TARGET(LQ1)
Elapsed time = 0.059196 sec

Ex. 4.17.4 Query name list.attributes.

$ mqpgf -qm HM8E2 **-nl NL1 MQOT_NAMELIST -inq MQCA_ALTERATION_DA
TE,MQCA_ALTERATION_TIME,MQCA_NAMELIST_DESC,MQCA_NAMELIST_NA
ME,MQCA_NAMES,MQIA_NAME_COUNT**
[20/03/06 11:59:54.593427] 1: ALTDATE(2012-03-06) ALTTIME(11.58.48) DESCR(s
ample name list) NAMELIST(NL1) NAMES('NAME1','NAME2','NAME3') NAMCO
UNT(3)
Elapsed time = 0.060212 sec

* MQOT_NAMELIST (object type) must be specified

Ex. 4.17.5 Query process attributes.

>mqpgf -qm HM9S **-p SYSTEM.DEFAULT.PROCESS MQOT_PROCESS -inq MQ
CA_ALTERATION_DATE,MQCA_ALTERATION_TIME,MQCA_APPL_ID,MQCA_EN
V_DATA,MQCA_PROCESS_DESC,MQCA_PROCESS_NAME,MQCA_USER_DATA,
MQIA_APPL_TYPE**
[2012/03/09 17:03:39.963] 1: ALTDATE(2011-11-26) ALTTIME(15.29.19) APPLICID
() ENVRDATA() DESCR() PROCESS(SYSTEM.DEFAULT.PROCESS) USERDATA
() APPLTYPE(WINDOWSNT)
Elapsed time = 101 msec

* MQOT_PROCESS (object type) must be specified

Ex. 4.17.6 Query queue manager attributes.

$ mqpgf -qm TESQM **MQOT_Q_MGR -inq MQCA_ALTERATION_DATE,MQCA_A
LTERATION_TIME,MQCA_CHANNEL_AUTO_DEF_EXIT,MQCA_CLUSTER_WOR
KLOAD_DATA**
[16/12/20 19:30:48] 1: ALTDATE(2016-12-13) ALTTIME(14.19.03) CHADEXIT() CL
WLDATA()

4.18 Specifying message properties

Put messages with specifying message properties of arbitrary data type.

mqpgf -qm <qmgr> -q <queue> -m <message> **-smp: Tyep:Property Name:Property Value (e.g. MQTYPE_STRING:property name:value,..:..)**

-smp: Message Properties

MQPMO_VERSION_3: Requires MQPMO_VERSION_3 when creating message properties.

Table 4.18.1 Data types of property values.

Type	Property Value	Sample
MQTYPE_BOOLEAN	TRUE/FAULSE	
MQTYPE_BYTE_STRING	Hexadecimal notation	01ef
MQTYPE_INT8	8bit Integer	-128 - 127
MQTYPE_INT16	16bit Integer	-32,768 - 32,768
MQTYPE_INT32	32bit Integer	-2,147,483,648 - 2,147,483,647
MQTYPE_INT64	64bit Integer	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807
MQTYPE_FLOAT32	32bit Floating point	-3.402823e+38 - 3.402823e+38
MQTYPE_FLOAT64	64bit Floating point	-1.797693e+308 - 1.797693e+308
MQTYPE_STRING	Strings	
MQTYPE_NULL	* No valude can be specified for MQTYPE_NULL	

Ex. 4.18.1 Example of specifying message properties with maximum values for numeric type.

$ mqpgf -qm TESTQM -q TQ -m "test" **-smp "MQTYPE_BOOLEAN:boolean1:TRUE,MQTYPE_BYTE_STRING:byteString:0102feff,MQTYPE_INT8:int8:127,MQTYPE_INT16:int16:32767,MQTYPE_INT32:int32:2147483647,MQTYPE_INT64:int64:922337**

2036854775807,MQTYPE_FLOAT32:float32:3.402823e+38,MQTYPE_FLOAT64:float6
4:1.797693e+308,MQTYPE_STRING:string:data,MQTYPE_NULL:null1:" MQPMO_V
ERSION_3
[16/12/22 20:12:16] 1: message length: 4 put message：test
$
$ mqpgf -qm TESTQM -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VE
RSION_4 -br
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x41
4D51206F6B61716D38306120202020 58586FA220003003] CorrelId[0x0000000000000
0000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                   ] ReplyToQMgr[TESTQM
              ] UserIdentifier[mq80   ] AccountingToken[0x033234320
0000000000000000000000000000000000000000000000000006] ApplIdentityData
[   ] PutApplType[6] PutApplName[mqpgf
          ] PutDate[20161220] PutTime[12443000] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

****Message properties****

  boolean1：TRUE
  byteString：X'0102FEFF'
  int8：127
  int16：32767
  int32：2147483647
  int64：9223372036854775807
  float32：3.402823e+38
  float64：1.797693e+308
  string：'data'
  null1：NULL
MQINQMP faild：CompCd=02 ReasonCd=2471

data length: 4
00000000: 7465 7374   'test   '
* This program queries all message properties with MQPROP_INQUIRE_ALL spe
cified in the argument InqPropOpts when calling MQINQMP. Call MQINQMP ()
repeatedly until reason code: 2471 (MQRC_PROPERTY_NOT_AVAILABLE) is retu
rned.

Ex. 4.18.2 Example of specifying message properties with minimum values for

numeric type.

$ mqpgf -qm TESTQM -q TQ -m "test" **-smp "MQTYPE_BOOLEAN:boolean1:FAL SE,MQTYPE_BYTE_STRING:byteString:0102feff,MQTYPE_INT8:int8:-128,MQTYPE _INT16:int16:-32768,MQTYPE_INT32:int32:-2147483648,MQTYPE_INT64:int64:-922 3372036854775808,MQTYPE_FLOAT32:float32:-3.402823e+38,MQTYPE_FLOAT64:fl oat64:-1.797693e+308,MQTYPE_STRING:string:data,MQTYPE_NULL:null1:"** MQP MO_VERSION_3

[16/12/22 20:14:16] 1: message length: 4 put message : test

$

$ mqpgf -qm TESTQM -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VE RSION_4 -br

message number: 1

*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding [273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x41 4D51206F6B61716D3830612020202058586FA220003203] CorrelId[0x0000000000000 0000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[   ] ReplyToQMgr[TESTQM   ] UserIdentifier[mq80   ] AccountingToken[0x033234320 000000000000000000000000000000000000000000000000006] ApplIdentityData [   ] PutApplType[6] PutApplName[mqpgf   ] PutDate[20161220] PutTime[12490416] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

****Message properties****

  boolean1 : FALSE
  byteString : X'0102FEFF'
  int8 : -128
  int16 : -32768
  int32 : -2147483648
  int64 : -9223372036854775808
  float32 : -3.402823e+38
  float64 : -1.797693e+308
  string : 'data'
  null1 : NULL
MQINQMP faild : CompCd=02 ReasonCd=2471

data length: 4
00000000: 7465 7374   'test   '

4.19 Using Distribution Lists

Put a message into multiple queues with specifying an object record.

mqpgf -qm <qmgr> -or <queue1[:qmgr1]>,<queue2[:qmgr2]>,<queue3[:qmgr3]>,... -mr <msgId>:<correlId>:<groupId>:<feedback>:<accountingtoken>,... -m "distribution lists" MQOD_VERSION_2 MQPMO_VERSION_2 [MQPMRF_MSG_ID MQPMRF_CORREL_ID MQPMRF_GROUP_ID MQPMRF_FEEDBACK MQPMRF_ACCOUNTING_TOKEN MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT MQMD_VERSION_2 MQMF_MSG_IN_GROUP]

-or: Object record.

MQOD_VERSION_2: Requires MQOD_VERSION_2 or more when using distribution list.
MQPMO_VERSION_2: Requires MQPMO_VERSION_2 or more when using distribution list.

(sample options)
-mr: Put message record

The following table shows MQMD fields that can be specified in put message record.

Table 4.19.1 Fields of put message record.

Order	Field	MQPMRF_*	Note
1	MsgId	MQPMRF_MSG_ID	
2	CorrelId	MQPMRF_CORREL_ID	
3	GroupId	MQPMRF_GROUP_ID	MQMD_VERSION_2 and one of MQMF_MSG_IN_GROUP, MQMF_LAST_MSG_IN_GROUP, MQMF_SEGMENT, MQMF_LAST_SEGMENT, MQMF_SEGMENTATION_ALLOWED must be specified.
4	Feedback	MQPMRF_FEEDBACK	
5	AccountingToken	MQPMRF_ACCOUNTING_TOKEN	MQOO_SET_ALL_CONTEXT and MQPMO_SET_ALL_CONTEXT, or MQOO_SET_IDENTITY_CONTEXT and MQPMO_SET_IDENTITY_CONTEXT must be specified.

Ex. 4.19.1 Put a message into multiple local queues with specifying an object record.

$ mqpgf -qm TESTQM **-or INQ1,INQ2,INQ3** -m "distribution lists" MQOD_VERSI
ON_2 MQPMO_VERSION_2
[16/12/22 20:15:16] 1: message length: 18 put message : distribution lists
$
$ mqpgf -qm TESTQM -q INQ1 -br
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[546] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x41
4D512053545343514D202020202020DA9B365620001E02] CorrelId[0x0000000000000
00000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                      ] ReplyToQMgr[TESTQM
          ] UserIdentifier[testuser   ] AccountingToken[0160105150000005CB9
193C9FEF8154FF9CFF8AE8030000000000000000000000000B] ApplIdentityData[
                      ] PutApplType[11] PutApplName[ojects¥mqpgf¥Debug¥m
qpgf.exe] PutDate[20151101] PutTime[23142905] ApplOriginData[   ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 17
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973   'distribution lis'
00000010: 7473   'ts   '

$ mqpcf ques -qm TESTQM -q "INQ*" CURDEPTH
1: QUEUE(INQ1) TYPE(QUEUE) CURDEPTH(1)
2: QUEUE(INQ2) TYPE(QUEUE) CURDEPTH(1)
3: QUEUE(INQ3) TYPE(QUEUE) CURDEPTH(1)

Ex. 4.19.2 Put a message into a cluster queue with specifying a object queue manager.

$ mqpgf -qm QMA **-or CLUS_Q1:QMA,CLUS_Q1:QMB,CLUS_Q2:QMB** -m "distrib
ution lists" MQOD_VERSION_2 MQPMO_VERSION_2
[16/12/22 20:15:53] 1: message length: 18 put message : distribution lists
$
$ mqpcf ques -qm QMA -q CLUS_Q1 CURDEPTH
1: QUEUE(CLUS_Q1) TYPE(QUEUE) CURDEPTH(1)
$
$ mqpcf ques -qm QMB -q CLUS_Q1 CURDEPTH
1: QUEUE(CLUS_Q1) TYPE(QUEUE) CURDEPTH(1)

```
$
$ mqpgf -qm QMB -q CLUS_Q2 -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[546] CodedCharSetId[932] Format[        ] Priority[0] Persistence[0] MsgId[0x41
4D5120434C41202020202020202020E29B365620003905] CorrelId[0x0000000000000
0000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                            ] ReplyToQMgr[QMA
          ] UserIdentifier[testuser    ] AccountingToken[0160105150000005CB91
93C9FEF8154FF9CFF8AE80300000000000000000000B] ApplIdentityData[
                          ] PutApplType[11] PutApplName[ojects¥mqpgf¥Debug¥m
qpgf.exe] PutDate[20151101] PutTime[23453955] ApplOriginData[    ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 17
00000000:  6469 7374 7269 6275 7469 6F6E 206C 6973   'distribution lis'
00000010:  7473                                      'ts              '
-------------------------------------
```

Ex. 4.19.3 Example of specifying a message record only in the first queue of an object
record.
```
-------------------------------------
$ mqpgf -qm QMA -or CLUS_Q1:QMA,CLUS_Q1:QMB,CLUS_Q2:QMB ¥
-mr msgId1:correlId1:groupId1:MQFB_QUIT:account1 ¥
-m "distribution lists" MQOD_VERSION_2 MQPMO_VERSION_2 ¥
MQPMRF_MSG_ID MQPMRF_CORREL_ID MQPMRF_GROUP_ID MQPMRF_FEE
DBACK MQPMRF_ACCOUNTING_TOKEN ¥
MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT ¥
MQMD_VERSION_2 MQMF_MSG_IN_GROUP
[16/12/22 20:20:55] 1: message length: 18 put message : distribution lists
-------------------------------------
```

Ex. 4.19.4 Use distribution list with client mode.
```
-------------------------------------
$ mqpgfc -qm QMA -or CLUS_Q1:QMA,CLUS_Q1:QMB,CLUS_Q2:QMB -mr msgI
d1:correlId1:groupId1:MQFB_QUIT:account1 ¥
 -m "distribution lists" -x 'remotehost(1414)' ¥
 MQOD_VERSION_2 MQPMO_VERSION_2 ¥
MQPMRF_MSG_ID MQPMRF_CORREL_ID MQPMRF_GROUP_ID MQPMRF_FEE
DBACK MQPMRF_ACCOUNTING_TOKEN ¥
 MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT MQMD_VERSION_2
```

MQMF_MSG_IN_GROUP
[16/12/22 20:21:50] 1: message length: 18 put message : distribution lists

$ mqpgfc -qm QMA -q CLUS_Q1 -x 'remotehost(1414)' -br
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[256] Encodin
g[273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x6
D736749643100000000000000000000000000000000] CorrelId[0x636F7272656C4
9643100000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                      ] ReplyToQMgr[QMA
            ] UserIdentifier[   ] AccountingToken[0x6163636F75
6E7431000000000000000000000000000000000000000000000000] ApplIdentityData[
                          ] PutApplType[0] PutApplName[
       ] PutDate[   ] PutTime[   ] ApplOriginData[   ]

GroupId[0x67726F75704964310000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 17
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973   'distribution lis'
00000010: 7473   'ts   '

....

Ex. 4.19.5 Omit some of parameters in a message record.

$ mqpgf -qm QMA -or CLUS_Q1:QMA,CLUS_Q1:QMB,CLUS_Q2:QMB ¥
-mr msgId1:correlId1:groupId1:MQFB_QUIT:account1,¥
MQMI_NONE:MQCI_NONE:MQGI_NONE:MQFB_NONE:MQACT_NONE,¥
0x111111:0x222222:0x333333:MQFB_EXPIRATION:0x444444¥
 -m "distribution lists" MQOD_VERSION_2 MQPMO_VERSION_2 MQPMRF_MSG
_ID MQPMRF_CORREL_ID MQPMRF_GROUP_ID MQPMRF_FEEDBACK MQPM
RF_ACCOUNTING_TOKEN MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CO
NTEXT MQMD_VERSION_2 MQMF_MSG_IN_GROUP
[16/12/22 20:22:25] 1: message length: 18 put message : distribution lists
$

$ mqpgf -qm QMB -q CLUS_Q1 -br
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] **Feedback[0]** Encoding
[546] CodedCharSetId[932] Format[   ] Priority[0] Persistence[0] **MsgId[0x41
4D5120434C41202020202020202020E29B365620004103]** **CorrelId[0x0000000000000
00000000000000000000000000000000000]** BackoutCount[0] ReplyToQ[

] ReplyToQMgr[QMA
               ] UserIdentifier[   ] **AccountingToken[00000000000000000000**
00000000000000000000000000000000000000000000] ApplIdentityData[
                    ] PutApplType[0] PutApplName[   ] Put
Date[   ] PutTime[   ] ApplOriginData[   ] **GroupId[0x414D5120434C4**
12020202020202020E29B365620004102] MsgSeqNumber[1] Offset[0] MsgFlags[8]
  OriginalLength[-1]

data length: 17
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973   'distribution lis'
00000010: 7473   'ts   '

$ mqpgf -qm QMB -q CLUS_Q2 -br
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[258] Encodin
g[546] CodedCharSetId[932] Format[   ] Priority[0] Persistence[0] MsgId[0x1
11111000000000000000000000000000000000000000000] CorrelId[0x22222200000000
0000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                          ] ReplyToQMgr[QMA
               ] UserIdentifier[   ] AccountingToken[0x444444000000000000
00000000000000000000000000000000000000000] ApplIdentityData[
                    ] PutApplType[0] PutApplName[   ] P
utDate[   ] PutTime[   ] ApplOriginData[   ]

GroupId[0x33333300000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 17
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973   'distribution lis'
00000010: 7473   'ts   '

Ex. 4.19.6 Specify only some fields of a message record.

$ mqpgf -qm QMA -or TQ1,TQ2,TQ3 -m "dest" ¥
-mr msgId1:groupId1,msgId2:groupId2 MQOD_VERSION_2 MQPMO_VERSION_2
¥
MQPMRF_MSG_ID MQPMRF_GROUP_ID MQMD_VERSION_2 MQMF_MSG_IN_
GROUP
[16/12/22 20:23:35] 1: message length: 4 put message : dest

$ mqpgf -qm QMA -q TQ1 -br -r
message number: 1

*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[1208] Format[   ] Priority[0] Persistence[0] **MsgId[0x6
D73674964310000000000000000000000000000000000]** CorrelId[0x0000000000000
000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                        ] ReplyToQMgr[TESTQM
              ] UserIdentifier[mqm   ] AccountingToken[0x03373031
0000000000000000000000000000000000000000000000006] ApplIdentityData
[   ] PutApplType[6] PutApplName[mqpgf
         ] PutDate[20151108] PutTime[23022907] ApplOriginData[   ]

GroupId[0x67726F7570496643100000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 4
00000000: 6465 7374   'dest   '
....

Ex. 4.19.7 An example where putting to a part of cluster queues in an object record
fails.

$ mqpcf que -qm QMA -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(0) PUT(ENABLED)
$ mqpcf que -qm QMB -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(0) **PUT(DISABLED)**
$ mqpcf que -qm QMB -q CLUS_Q2 PUT CURDEPHT
1: QUEUE(CLUS_Q2) TYPE(QLOCAL) CURDEPTH(0) PUT(ENABLED)

$ mqpgf -qm QMA -or CLUS_Q1:QMA,CLUS_Q1:QMB,CLUS_Q2:QMB -m "distrib
ution lists" MQOD_VERSION_2 MQPMO_VERSION_2
[16/12/22 20:24:35] 1: message length: 18 put message : distribution lists
$
$ mqpcf que -qm QMA -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(1) PUT(ENABLED)
$ mqpcf que -qm QMB -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) **CURDEPTH(0) PUT(DISABLED)**
$ mqpcf que -qm QMB -q CLUS_Q2 PUT CURDEPHT
1: QUEUE(CLUS_Q2) TYPE(QLOCAL) PUT(ENABLED) CURDEPTH(1)

Ex. 4.19.8 Example in which MQRC_MULTIPLE_REASONS is returned.

$ mqpgf -qm QMA -or CQ1:QMB,CQQ:QMA,CQ2:QMC -m "distribution lists" MQ

OD_VERSION_2 MQPMO_VERSION_2

MQRC_MULTIPLE_REASONS:MQOPEN for CQ1(QMB) returned CompCode=0, Reason=0

MQRC_MULTIPLE_REASONS:MQOPEN for CQQ(QMA) returned CompCode=2, Reason=2085

MQRC_MULTIPLE_REASONS:MQOPEN for CQ2(QMC) returned CompCode=0, Reason=0

[16/12/22 20:25:39] 1: message length: 18 put message : distribution lists

MQRC_MULTIPLE_REASONS:MQPUT for CQ1(QMB) returned CompCode=0, Reason=0

MQRC_MULTIPLE_REASONS:MQPUT for CQQ(QMA) returned CompCode=2, Reason=2137

MQRC_MULTIPLE_REASONS:MQPUT for CQ2(QMC) returned CompCode=0, Reason=0

$ mqrc 2085

    2085   0x00000825   MQRC_UNKNOWN_OBJECT_NAME

$ mqrc 2137

    2137   0x00000859   MQRC_OPEN_FAILED

4.20 Segmentation by queue manager

It is possible to perform segmentation by queue manager against a message put. Segmentation is performed with the smaller value of MAXMSGL property of a queue or queue manager.

mqpgf -qm <qmgr> -q <queue> -f <filename> **MQMF_SEGMENTATION_ALLOWE D MQMD_VERSION_2**

MQMF_SEGMENTATION_ALLOWED: To cause the queue manager to perform segmentation, specify MQMF_SEGMENTATION_ALLOWED in MQMD.MsgFlags. MQMD_VERSION_ 2: It is necessary to use MQMD_VERSION_ 2 for segmentation.

(sample options)
-f: Specify the path of the file containing message data to be PUT.

Ex. 4.20.1 Segmentation by queue manager
----
* Set MAXMSGL of the queue to 100 to make the test easier.
$ echo "alter ql('SampleQ') maxmsgl(100)" | runmqsc SampleQM

* Prepare an arbitrary text file of 128 bytes or more.
$ ls -l largemsg.txt
-rw-r--r--   1 MQM.MANAGER   MQM   315 Apr 25 17:34 largemsg.txt

$ cat largemsg.txt
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz<

$ mqpgf -qm SampleQM -q SampleQ -f largemsg.txt MQMF_SEGMENTATION_ALLOWED MQMD_VERSION_2
[18/05/09 15:12:05] 1: put from largemsg.txt

$ mqpgf: ./mqpgf -qm SampleQM -q SampleQ -dpv -r<
message number: 1
....
GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNumber[1] Offset[0] MsgFlags[3] OriginalLength[96]

data length: 96

00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'

....

00000050:   4849 4A4B 4C4D 4E4F 5051 5253 5455 5657 'HIJKLMNOPQRSTUV
W'

message number: 2

....

GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNum
ber[1] Offset[96] MsgFlags[3] OriginalLength[96]

data length: 96

00000000:   5859 5A61 6263 6465 6667 6869 6A6B 6C6D 'XYZabcdefghijklm'

....

00000050:   6F70 7172 7374 7576 7178 797A 0A31 3233 'opqrstuvqxyz.123'

message number: 3

....

GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNum
ber[1] Offset[192] MsgFlags[3] OriginalLength[96]

data length: 96

00000000:   3435 3637 3839 3041 4243 4445 4647 4849 '4567890ABCDEFGHI'

....

00000050:   4B4C 4D4E 4F50 5152 5354 5556 5758 595A 'KLMNOPQRSTUVWXY
Z'

message number: 4

....

GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNum
ber[1] Offset[288] MsgFlags[7] OriginalLength[27]

data length: 27

00000000:   6162 6364 6566 6768 696A 6B6C 6D6E 6F70 'abcdefghijklmnop'
00000010:   7172 7374 7576 7178 797A 0A   'qrstuvqxyz.   '

no message available : SampleQ CompCd=02 ReasonCd=2033

* Segmentation by queue manager may not be executed with accurate MAXMSG
L value. For example, in the result confirmed with MQ 9.0 for Windows or MQ
8.0 for HPNonStop, it is used by the largest multiple of 8 that does not exceed
MAXMSGL.

$ echo "alter ql('SampleQ') maxmsgl(4194304)" | runmqsc SampleQM

4.21 Application segmentation

This program executes segmentation by itself by specifying the size of the se gment following "-as" parameter. Specifically, it divides the message with the s pecified size, sets MQMF_SEGMENT to MsgFlags of the segment and MQMF_ LAST_SEGMENT for the last segment. If MQPMO_LOGICAL_ORDER is not s pecified in MQPMO.Options, it is necessary to set GroupId, Offset in an applic ation. This program sets Offset if MQPMO_LOGICAL_ORDER is not specified as an argument, but for GroupId, it does not use the value automatically app ended to the first segment for subsequent segments. If GroupId differs for eac h message, MQGMO_COMPLETE_MSG or MQGMO_ALL_SEGMENTS_AVAILA BLE can not be used when calling MQGET(). To enable message reassembly, s pecify GroupId directly in this program argument or specify MQPMO_LOGICA L_ORDER. If mqpgf directly specifies "GroupId" with "-gi", set that GroupId fo r all segments. If an application specifies MQPMO_LOGICAL_ORDER, the que ue manager automatically sets appropriate values for GroupId, MsgSeqNumber and Offset. The application only needs to specify MQMF_SEGMENT for Msg Flags and MQMF_LAST_SEGMENT for the last segment. If MQPMO_LOGICA L_ORDER is also given as an argument, mqpgf only sets MsgFlags.

mqpgf -qm <qmgr> -q <queue> -f <filename> **-as MQMD_VERSI ON_2 MQPMO_LOGICAL_ORDER** MQPMO_SYNCPOINT

-as: Perform message segmentation with the specified size
MQMD_VERSION_ 2: It is necessary to use MQMD_VERSION_ 2 for segmentati on.
MQPMO_LOGICAL_ORDER: Queue manager automatically sets appropriate value s for GroupId, MsgSeqNumber and Offset

(sample options)
-f: Specify the path of the file containing message data to be put.
MQPMO_SYNCPOINT: Process(Commit / Backout) a series of segmented message s with one UOW (Unit Of Work).

Ex. 4.21.1 Application segmentation

$ mqpgf -qm SampleQM -q SampleQ -f largemsg.txt -as 100 MQMD_VERSION_2
MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
[18/05/23 17:55:16] 1: put from: largemsg.txt
[18/05/23 17:55:16] 1: logical message: 1 length: 315 put message: 1234567890AB CDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz.1234567890ABCD EFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz.12
[18/05/23 17:55:16] 1: segment: 1 length: 100 put message: 1234567890ABCDEFG HIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz.1234567890ABCDEFGHIJ

KLMNOPQRSTUVWXYZa
[18/05/23 17:55:16] 1: segment: 2 length: 100 put message: bcdefghijklmnopqrstuv
qxyz.1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxy
z.1234567890A
[18/05/23 17:55:16] 1: segment: 3 length: 100 put message: BCDEFGHIJKLMNOP
QRSTUVWXYZabcdefghijklmnopqrstuvqxyz.1234567890ABCDEFGHIJKLMNOPQR
STUVWXYZabcdefghijkl
[18/05/23 17:55:16] 1: segment: 4 length: 15 put message: mnopqrstuvqxyz.
MQCMIT success : CompCd=00 ReasonCd=00

* Physical messages are shown **in bold.**

$ mqpgf -qm SampleQM -q SampleQ -dpv -r
message number: 1
....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNum
ber[1] Offset[0] MsgFlags[2] OriginalLength[100]

data length: 100
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
....
00000060: 5859 5A61   'XYZa   '

message number: 2
....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNum
ber[1] Offset[100] MsgFlags[2] OriginalLength[100]

data length: 100
00000000: 6263 6465 6667 6869 6A6B 6C6D 6E6F 7071 'bcdefghijklmnopq'
....
00000060: 3839 3041   '890A   '

message number: 3
....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNum
ber[1] Offset[200] MsgFlags[2] OriginalLength[100]

data length: 100
00000000: 4243 4445 4647 4849 4A4B 4C4D 4E4F 5051 'BCDEFGHIJKLMNOP
Q'
....
00000060: 696A 6B6C   'ijkl   '

message number: 4

....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNumber[1] Offset[300] MsgFlags[6] OriginalLength[15]

data length: 15
00000000:   6D6E 6F70 7172 7374 7576 7178 797A 0A   'mnopqrstuvqxyz. '

no message available : SampleQ CompCd=02 ReasonCd=2033

* It is segmented by the specified size (100). (by application)
* GroupId is automatically numbered, and the same value is set for all messages. (by queue manager)
* MsgSeqNumber is also set to "1" for all. (by queue manager)
* Offset is set to the following message, starting with "0" and adding the divided sizes. (by queue manager)
* MsgFlags is set to "2" except for the last segment and "6" is set for the last segment. "2" is MQMF_SEGMENT, "6" is ORed with MQMF_SEGMENT and MQMF_LAST_SEGMENT. Since it was divided by application, MQMF_SEGMENTATION_ALLOWED is not set this time.

MQMF_SEGMENTATION_ALLOWED   0x00000001
MQMF_SEGMENT   0x00000002
MQMF_LAST_SEGMENT   0x00000004

Here, mqpgf specifies only MQMF_LAST_SEGMENT for the last segment. If MQMF_LAST_SEGMENT is specified, the queue manager will automatically turn on (OR) MQMF_SEGMENT and send the message. (by application + queue manager)
* The same value as the segmented message size is set in OriginalLength. (by queue manager)

4.22 Reassembly by queue manager

To have queue manager reassemble segmented messages, call MQGET() with MQGMO_COMPLETE_MSG set in MQGMO.Options.

mqpgf -qm <qmgr> -q <queue> -dpv **MQGMO_COMPLETE_MSG**

MQGMO_COMPLETE_MSG: Requests queue manager to reassemble segmented messages.

(sample options)
-dpv: Get and dump a message (Verbose)

Ex. 4.22.1 Reassembly by queue manager
----
* Put the message segmented for every 100 bytes by the application.

$ mqpgf -qm SampleQM -q SampleQ -f largemsg.txt -as 100 MQMD_VERSION_2 MQPMO_LOGICAL_ORDER
[18/05/23 17:57:48] 1: put from: largemsg.txt
[18/05/23 17:57:48] 1: logical message: 1 length: 315 put message: 1234567890AB CDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz.1234567890ABCD EFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz.12
[18/05/23 17:57:48] 1: segment: 1 length: 100 put message: 1234567890ABCDEFG HIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz.1234567890ABCDEFGHIJ KLMNOPQRSTUVWXYZa
[18/05/23 17:57:48] 1: segment: 2 length: 100 put message: bcdefghijklmnopqrstuv qxyz.1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxy z.1234567890A
[18/05/23 17:57:48] 1: segment: 3 length: 100 put message: BCDEFGHIJKLMNOP QRSTUVWXYZabcdefghijklmnopqrstuvqxyz.1234567890ABCDEFGHIJKLMNOPQR STUVWXYZabcdefghijkl
[18/05/23 17:57:48] 1: segment: 4 length: 15 put message: mnopqrstuvqxyz.

$ mqpcf ques -qm SampleQM -q SampleQ CURDEPTH -t
[18/05/09 17:59:54] 1: QUEUE(SampleQ) TYPE(QUEUE) **CURDEPTH(4)**

* It is segmented into 4 messages(CURDEPTH). **(Boldface part)**
By specifying MQGMO_COMPLETE_MSG, let the queue manager reassemble the segmented message.

$ mqpgf -qm SampleQM -q SampleQ -dpv MQGMO_COMPLETE_MSG
message number: 1
….
GroupId[0x414D512053616D706C65514D202020205B05070120002B04] MsgSeqNum

ber[1] Offset[0] MsgFlags[6] OriginalLength[315]

data length: 315
```
00000000:    3132 3334 3536 3738 3930 4142 4344 4546  '1234567890ABCDEF'
00000010:    4748 494A 4B4C 4D4E 4F50 5152 5354 5556  'GHIJKLMNOPQRSTU
V'
00000020:    5758 595A 6162 6364 6566 6768 696A 6B6C  'WXYZabcdefghijkl'
00000030:    6D6E 6F70 7172 7374 7576 7178 797A 0A31  'mnopqrstuvqxyz.1'
00000040:    3233 3435 3637 3839 3041 4243 4445 4647  '234567890ABCDEFG'
00000050:    4849 4A4B 4C4D 4E4F 5051 5253 5455 5657  'HIJKLMNOPQRSTUV
W'
00000060:    5859 5A61 6263 6465 6667 6869 6A6B 6C6D  'XYZabcdefghijklm'
00000070:    6E6F 7071 7273 7475 7671 7879 7A0A 3132  'nopqrstuvqxyz.12'
00000080:    3334 3536 3738 3930 4142 4344 4546 4748  '34567890ABCDEFGH'
00000090:    494A 4B4C 4D4E 4F50 5152 5354 5556 5758  'IJKLMNOPQRSTUVW
X'
000000A0:    595A 6162 6364 6566 6768 696A 6B6C 6D6E  'YZabcdefghijklmn'
000000B0:    6F70 7172 7374 7576 7178 797A 0A31 3233  'opqrstuvqxyz.123'
000000C0:    3435 3637 3839 3041 4243 4445 4647 4849  '4567890ABCDEFGHI'
000000D0:    4A4B 4C4D 4E4F 5051 5253 5455 5657 5859  'JKLMNOPQRSTUVWX
Y'
000000E0:    5A61 6263 6465 6667 6869 6A6B 6C6D 6E6F  'Zabcdefghijklmno'
000000F0:    7071 7273 7475 7671 7879 7A0A 3132 3334  'pqrstuvqxyz.1234'
00000100:    3536 3738 3930 4142 4344 4546 4748 494A  '567890ABCDEFGHIJ'
00000110:    4B4C 4D4E 4F50 5152 5354 5556 5758 595A  'KLMNOPQRSTUVWXY
Z'
00000120:    6162 6364 6566 6768 696A 6B6C 6D6E 6F70  'abcdefghijklmnop'
00000130:    7172 7374 7576 7178 797A 0A              'qrstuvqxyz.     '
```

* It has been reassembled and read as one physical message.

$ mqpcf ques -qm SampleQM -q SampleQ CURDEPTH -t
[18/05/09 18:04:18] 1: QUEUE(SampleQ) TYPE(QUEUE) **CURDEPTH(0)**

* All four segments have been deleted with one GET.

4.23 Reassembly by application

Specify MQGMO_LOGICAL_ORDER so that segments are always retrieved in sequence even if there are multiple routes or if the segments do not arrive in order due to effects such as application threading. In addition, specify MQGMO_ALL_MSGS_AVAILABLE so that MQGET() processing is not performed until all segments arrive at the receive queue.

mqpgf -qm <qmgr> -q <queue> -dp -r **MQGMO_VERSION_2 MQGMO_LOGICAL_ ORDER MQGMO_ALL_SEGMENTS_AVAILABLE** MQGMO_SYNCPOINT

MQGMO_VERSION_ 2: When specifying MQGMO_LOGICAL_ORDER, it is necessary to use MQGMO_VERSION_ 2 or more.
MQGMO_LOGICAL_ORDER: Always ensure that segments are retrieved in order.
MQGMO_ALL_SEGMENTS_AVAILABLE: Do not perform MQGET() processing until all segments arrive at the receive queue.

(sample options)
-dp: GET and dump a message
-r: Read a message repeatedly until a queue become empty.

Ex. 4.23.1 Reassembly by application
----
* In order to understand the behavior of MQGMO_LOGICAL_ORDER and MQGMO_ALL_SEGMENTS_AVAILABLE, here we will manually create a single segment message in a queue.
We assume a 24 byte logical message segmented into three. Split segment size is 8 bytes. Since it is the same logical message, GroupId and MsgSeqNumber are the same for all segments.

* Put the last (third) segment to the queue first. Specify MQMF_LAST_SEGMENT. Offset is 16.

$ mqpgf -qm SampleQM -q SampleQ -m Segment3 -gi GID -ms 1 -of 16 MQMD_ VERSION_2 MQMF_LAST_SEGMENT
[18/05/23 18:01:58] 1: message length: 8 put message: Segment3

* Next, put the first segment in the queue. Specify MQMF_SEGMENT. Offset is 0.

$ mqpgf -qm SampleQM -q SampleQ -m Segment1 -gi GID -ms 1 -of 0 MQMD_V ERSION_2 MQMF_SEGMENT
[18/05/23 18:02:12] 1: message length: 8 put message: Segment1

* CURDEPTH of the queue indicates 2 messages, of course.

$ mqpcf ques -qm SampleQM -q SampleQ CURDEPTH
1: QUEUE(SampleQ) TYPE(QUEUE) **CURDEPTH(2)**

* Here, try specifying MQGMO_LOGICAL_ORDER MQGMO_ALL_SEGMENTS_A
VAILABLE and getting it. When specifying MQGMO_LOGICAL_ORDER, it is nec
essary to specify MQGMO_VERSION_ 2 or more and MQMD_VERSION_ 2. Whe
n "-dp" is specified in mqpgf, MQMD_VERSION_2 is used by default.

$ mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_LOG
ICAL_ORDER MQGMO_ALL_SEGMENTS_AVAILABLE
no message available : SampleQ CompCd=02 ReasonCd=2033

$ mqrc 2033

    2033 0x000007f1 MQRC_NO_MSG_AVAILABLE

* Because all segments have not arrived at the receive queue, MQGET () returns
 MQRC_NO_MSG_AVAILABLE and no messages are got.

* Put the remaining second segment. Specify MQMF_SEGMENT. Offset is 8.

$ mqpgf -qm SampleQM -q SampleQ -m Segment2 -gi GID -ms 1 -of 8 MQMD_V
ERSION_2 MQMF_SEGMENT
[18/05/23 18:03:35] 1: message length: 8 put message: Segment2

* Just to be sure, we will confirm the messages in the queue at this point by br
owsing in the order of physical order / FIFO (default).

$ mqpgf -qm SampleQM -q SampleQ -br -r
message number: 1
....
GroupId[0x474944000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] **Offset[16]** MsgFlags[6] OriginalLength[8]

data length: 8
00000000:   5365 676D 656E 7433   'Segment3   '

message number: 2
....
GroupId[0x474944000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] **Offset[0]** MsgFlags[2] OriginalLength[8]

data length: 8

```
00000000:  5365 676D 656E 7431                    'Segment1        '
```

message number: 3
….
GroupId[0x4749440000000000000000000000000000000000] MsgSeqNumber[1] **Offset[8]** MsgFlags[2] OriginalLength[8]

data length: 8
```
00000000:  5365 676D 656E 7432                    'Segment2        '
```

no message available : SampleQ CompCd=02 ReasonCd=2033

* The third, first, and second segments are arranged in this order.

* Specify MQGMO_LOGICAL_ORDER, MQGMO_ALL_SEGMENTS_AVAILABLE and try to get repeatedly.

$ mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_LOGICAL_ORDER MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_SYNCPOINT
message number: 1
….
GroupId[0x4749440000000000000000000000000000000000] MsgSeqNumber[1] **Offset[0]** MsgFlags[2] OriginalLength[8]

data length: 8
```
00000000:  5365 676D 656E 7431                    'Segment1        '
```

message number: 2
….
GroupId[0x4749440000000000000000000000000000000000] MsgSeqNumber[1] **Offset[8]** MsgFlags[2] OriginalLength[8]

data length: 8
```
00000000:  5365 676D 656E 7432                    'Segment2        '
```

message number: 3
….
GroupId[0x4749440000000000000000000000000000000000] MsgSeqNumber[1] **Offset[16]** MsgFlags[6] OriginalLength[8]

data length: 8
```
00000000:  5365 676D 656E 7433                    'Segment3        '
```

no message available : SampleQ CompCd=02 ReasonCd=2033
MQCMIT success : CompCd=00 ReasonCd=00

* The segments are getted correctly in logical order according to the value of Off
set.
* mqpgf can not be completed until the actual logical messages are reassembled.
You can confirm up to when these segments can be received in the intended ord
er.

4.24 Grouping logical messages

Following "-dl" you can specify a delimiter to use to create logical messages grouped from the specified message or file. Specifically, it divides it into logical messages excluding delimiters, sets MQMF_MSG_IN_GROUP to MsgFlags of the divided logical message, and MQMF_LAST_MSG_IN_GROUP for the last logical message. MQPMO_LOGICAL_ORDER also works for grouping logical messages. If MQPMO_LOGICAL_ORDER is not specified as an argument and GroupId is also not specified, a separate GroupId is assigned for each logical message and MsgSeqNumber is not incremented. If MQPMO_LOGICAL_ORDER is specified, the queue manager automatically sets appropriate values for GroupId, MsgSeqNumber (and Offset).

mqpgf -qm <qmgr> -q <queue> -f <filename> **-dl <delimiter> MQMD_VERSION_2 MQPMO_LOGICAL_ORDER** MQPMO_SYNCPOINT

-dl: Delimiter for dividing into logical messages specified in character string or hexadecimal notation
MQMD_VERSION_ 2: It is necessary to use MQMD_VERSION_ 2 for message grouping
MQPMO_LOGICAL_ORDER: Let the queue manager set appropriate values for GroupId and MsgSeqNumber.

(sample options)
-f: Specify the path of the file containing message data to be put.
MQPMO_SYNCPOINT: Process(Commit / Backout) a series of grouped messages with one UOW (Unit Of Work).

Ex. 4.24.1 Grouping logical messages
----
* Prepare an arbitrary text file of multiple lines as shown below.
$ cat largemsg2.txt
1234567890
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz

* Specify 0x0a (LF) as the delimiter for this file of multiple lines (LF: 0x0a), divide it into logical messages for each line and put the message. Also specify MQPMO_LOGICAL_ORDER. In Windows, the line breaks are normally 0x0d0a (CRLF), so specify it as "- dl 0x0d0a". In Unix, Linux or HP NonStop OSS environment, line breaks are 0x0a (LF) i.

$ mqpgf -qm SampleQM -q SampleQ -f largemsg2.txt -dl 0x0a MQMD_VERSION
_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
[18/05/23 18:05:31] 1: put from: largemsg2.txt
[18/05/23 18:05:31] 1: logical message: 1 length: 10 put message: 1234567890
[18/05/23 18:05:31] 1: logical message: 2 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
[18/05/23 18:05:31] 1: logical message: 3 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
[18/05/23 18:05:31] 1: logical message: 4 length: 36 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZ
[18/05/23 18:05:31] 1: logical message: 5 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
MQCMIT success : CompCd=00 ReasonCd=00

* **Bold text** indicates physical message.

* I will check the message that was split. In this case the logical message is eq
ual to the physical message.

$ mqpgf -qm SampleQM -q SampleQ -dp -r
message number: 1
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNum
ber[1] Offset[0] **MsgFlags[8]** OriginalLength[-1]

data length: 10
00000000: 3132 3334 3536 3738 3930   '1234567890   '

message number: 2
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNum
ber[2] Offset[0] **MsgFlags[8]** OriginalLength[-1]

data length: 62
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020: 5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghijkl'
00000030: 6D6E 6F70 7172 7374 7576 7178 797A   'mnopqrstuvqxyz '

message number: 3
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNum
ber[3] Offset[0] **MsgFlags[8]** OriginalLength[-1]

data length: 62
```
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghijkl'
00000030:   6D6E 6F70 7172 7374 7576 7178 797A      'mnopqrstuvqxyz  '
```

message number: 4

....

GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[4] Offset[0] **MsgFlags[8]** OriginalLength[-1]

data length: 36
```
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A                               'WXYZ           '
```

message number: 5

....

GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[5] Offset[0] **MsgFlags[24]** OriginalLength[-1]

data length: 62
```
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghijkl'
00000030:   6D6E 6F70 7172 7374 7576 7178 797A      'mnopqrstuvqxyz  '
```

no message available : SampleQ CompCd=02 ReasonCd=2033

* Each line (LF: 0x0a) is divided into logical messages. (by application)
* GroupId is automatically numbered, all the same value. (by queue manager)
* MsgSeqNumber is also incremented in order from "1". (by queue manager)
* MsgFlags is set to "8" except for the last segment and "24" to the last segment.
* "3" is MQMF_MSG_IN_GROUP, "24" is OR value of MQMF_MSG_IN_GROUP and MQMF_LAST_MSG_IN_GROUP.

MQMF_MSG_IN_GROUP   0x00000008
MQMF_LAST_MSG_IN_GROUP   0x00000010

MQMF_MSG_IN_GROUP | MQMF_LAST_MSG_IN_GROUP = 0x00000008 | 0x00000010 = 0x00000018(Hexadecimal) = 24(Decimal)

* mqpgf specify only MQMF_MSG_IN_GROUP for the last segment. If MQMF_L
AST_MSG_IN_GROUP is specified, the queue manager will automatically turn on
 (OR) MQMF_MSG_IN_GROUP and send the message. (by application and queue
 manager)

4.25 Grouping logical messages and Segmentation

It is also possible to combine logical message grouping and logical message s egmentation.

mqpgf -qm <qmgr> -q <queue> -f <filename> **-dl <delimiter> -as **
MQMD_VERSION_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT

-dl: Delimiter for dividing into logical messages specified in character string or h exadecimal notation
-as: Perform message segmentation with the specified size
MQMD_VERSION_ 2: It is necessary to use MQMD_VERSION_ 2 for message gr ouping and segmentation.
MQPMO_LOGICAL_ORDER: Let the queue manager set appropriate values for G roupId, MsgSeqNumber and Offset.

(sample options)
-f: Specify the path of the file containing message data to be put.
MQPMO_SYNCPOINT: Process(Commit / Backout) a series of grouped and/or seg mented messages with one UOW (Unit Of Work).

Ex. 4.25.1 Grouping logical messages and Segmentation
----
* The data in the file is divided into logical messages with delimiter (0x0a: LF), and additionally specifies "- as 40" as an argument so that logical messages exce eding 40 bytes are segmented by the application.

$ mqpgf -qm SampleQM -q SampleQ -f largemsg2.txt -dl 0x0a -as 40 MQMD_VE RSION_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
[18/05/25 15:12:29] 1: put from: largemsg2.txt
[18/05/25 15:12:29] 1: logical message: 1 length: 10 put message: 1234567890
[18/05/25 15:12:29] 1: logical message: 2 length: 62 put message: 1234567890ABC DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
　[18/05/25 15:12:29] 1: segment: 1 length: 40 put message: 1234567890ABCDEFG HIJKLMNOPQRSTUVWXYZabcd
　[18/05/25 15:12:29] 1: segment: 2 length: 22 put message: efghijklmnopqrstuvqxy z
[18/05/25 15:12:29] 1: logical message: 3 length: 62 put message: 1234567890ABC DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
[18/05/25 15:12:29] 1: segment: 1 length: 40 put message: 1234567890ABCDEFG HIJKLMNOPQRSTUVWXYZabcd
[18/05/25 15:12:29] 1: segment: 2 length: 22 put message: efghijklmnopqrstuvqxyz
[18/05/25 15:12:29] 1: logical message: 4 length: 36 put message: 1234567890ABC DEFGHIJKLMNOPQRSTUVWXYZ

[18/05/25 15:12:29] 1: logical message: 5 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvqxyz
**[18/05/25 15:12:29] 1: segment: 1 length: 40 put message: 1234567890ABCDEFG
HIJKLMNOPQRSTUVWXYZabcd**
[18/05/25 15:12:29] 1: segment: 2 length: 22 put message: efghijklmnopqrstuvqxyz
MQCMIT success : CompCd=00 ReasonCd=00

* **Bold** text indicates physical message.

* Confirm the message that was split.

$ mqpgf -qm SampleQM -q SampleQ -dp -r
message number: 1
….
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNum
ber[1] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 10
00000000:   3132 3334 3536 3738 3930   '1234567890   '

message number: 2
….
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNum
ber[2] Offset[0]** MsgFlags[10] OriginalLength[40]

data length: 40
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A 6162 6364   'WXYZabcd   '

message number: 3
….
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNum
ber[2] Offset[40]** MsgFlags[14] OriginalLength[22]

data length: 22
00000000:   6566 6768 696A 6B6C 6D6E 6F70 7172 7374 'efghijklmnopqrst'
00000010:   7576 7178 797A   'uvqxyz   '

message number: 4
….
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNum
ber[3] Offset[0]** MsgFlags[10] OriginalLength[40]

data length: 40
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A 6162 6364   'WXYZabcd   '

message number: 5
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNum
ber[3] Offset[40]** MsgFlags[14] OriginalLength[22]

data length: 22
00000000:   6566 6768 696A 6B6C 6D6E 6F70 7172 7374 'efghijklmnopqrst'
00000010:   7576 7178 797A   'uvqxyz   '

message number: 6
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNum
ber[4] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 36
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A   'WXYZ   '

message number: 7
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNum
ber[5] Offset[0]** MsgFlags[26] OriginalLength[40]

data length: 40
00000000:   3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010:   4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU
V'
00000020:   5758 595A 6162 6364   'WXYZabcd   '

message number: 8
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNum
ber[5] Offset[40]** MsgFlags[30] OriginalLength[22]

data length: 22
00000000:   6566 6768 696A 6B6C 6D6E 6F70 7172 7374 'efghijklmnopqrst'
00000010:   7576 7178 797A   'uvqxyz   '

no message available : SampleQ CompCd=02 ReasonCd=2033
MQCMIT success : CompCd=00 ReasonCd=00

* It can be confirmed that Offset is set to the physical message of 2nd, 3rd and 5th logical messages (MsgSeqNumber is 2, 3, 5) and divided into multiple segments.

4.26 Reading grouped logical messages

   In the example described here, we create grouped logical messages by splitin
g the data of one file, but mqpgf does not reassemble them into one message
or file when reading them. You can only confirm that logical messages can be
received in the intended order.

mqpgf -qm SampleQM -q SampleQ -dp -r **MQGMO_VERSION_2 MQGMO_COMP
LETE_MSG MQGMO_LOGICAL_ORDER MQGMO_ALL_MSGS_AVAILABLE MQ
GMO_ALL_SEGMENTS_AVAILABLE** MQGMO_WAIT MQWI_UNLIMITED MQG
MO_NO_SYNCPOINT

MQGMO_VERSION_2: It is necessary to use MQGMO_VERSION_2 or more.
MQGMO_COMPLETE_MSG: Requests the queue manager to reassemble the seg
mented message
MQGMO_LOGICAL_ORDER: Always ensure that logical messages and segmentsa
re retrieved in order.
MQGMO_ALL_SEGMENTS_AVAILABLE: Do not perform MQGET() processing un
til all group messages and segments arrive at the receive queue.
MQGMO_ALL_SEGMENTS_AVAILABLE: Do not perform MQGET() processing un
til all group messages and segments arrive at the receive queue.

(sample options)
-dp: GET and dump a message
-r: Read a message repeatedly until a queue become empty.
MQGMO_WAIT: Wait for message arrival
MQWI_UNLIMITED: Unlimited time to wait for message arrival
MQPMO_SYNCPOINT: Process(Commit / Backout) a series of grouped and/or seg
mented messages with one UOW (Unit Of Work).

Ex. 4.26.1 Reading grouped logical messages
----
* In order to make the behavior of each option easier to understand, here we cr
eate the physical message one by one manually in the queue.

We assume two types of group messages divided as shown below.
Group 1, Logical message 1, Segment 1
Group 1, Logical message 1, Segment 2
Group 1, Logical message 2
Group 1, Logical message 3, Segment 1
Group 1, Logical message 3, Segment 2

Group 2, Logical message 1
Group 2, Logical message 2, Segment 1

Group 2, Logical message 2, Segment 2

These corresponding command lines are below.

mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment1 -gi GroupId1 -ms 1 -of 0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment2 -gi GroupId1 -ms 1 -of 22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical2 -gi GroupId1 -ms 2 MQMD_VERSION_2 MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment1 -gi GroupId1 -ms 3 -of 0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment2 -gi GroupId1 -ms 3 -of 22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP

mqpgf -qm SampleQM -q SampleQ -m Group2Logical1 -gi GroupId2 -ms 1 MQMD_VERSION_2 MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment1 -gi GroupId2 -ms 2 -of 0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment2 -gi GroupId2 -ms 2 -of 22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP

* First, start mqpgf for reception with the following argument at another terminal / command prompt.

$ mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_COMPLETE_MSG MQGMO_LOGICAL_ORDER MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_WAIT MQWI_UNLIMITED MQGMO_NO_SYNCPOINT

* Wait indefinitely until all the messages of the group arrive in the queue by specifying MQGMO_WAIT and MQWI_UNLIMITED. Processing of each group's message should start when all arrives. It repeats it with multiple groups by specifying "-r" option.

* We will put messages in random order.
$ mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment2 -gi GroupId2 -ms 2 -of 22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP
[18/05/25 16:45:38] 1: message length: 22 put message : Group2Logical2Segment2

$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment2 -gi GroupId1 -ms 3 -of 22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_I

N_GROUP
[18/05/25 16:45:44] 1: message length: 22 put message : Group1Logical3Segment2

$ mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment1 -gi GroupId2 -ms 2 -of 0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
[18/05/25 16:45:49] 1: message length: 22 put message : Group2Logical2Segment1

$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment1 -gi GroupId1 -ms 3 -of 0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
[18/05/25 16:45:55] 1: message length: 22 put message : Group1Logical3Segment1

$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical2 -gi GroupId1 -ms 2 MQMD_VERSION_2 MQMF_MSG_IN_GROUP
[18/05/25 16:46:01] 1: message length: 14 put message : Group1Logical2

$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment2 -gi GroupId1 -ms 1 -of 22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_MSG_IN_GROUP
[18/05/25 16:46:06] 1: message length: 22 put message : Group1Logical1Segment2

$ mqpgf -qm SampleQM -q SampleQ -m Group2Logical1 -gi GroupId2 -ms 1 MQMD_VERSION_2 MQMF_MSG_IN_GROUP
[18/05/25 16:46:11] 1: message length: 14 put message : Group2Logical1

* Since we have written all the messages of group 2 at this point, mqpgf waiting at MQGET() at another terminal will process the message and it will be displayed.

message number: 1
....
GroupId[0x47726F757049643200000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 14
00000000:   4772 6F75 7032 4C6F 6769 6361 6C31   'Group2Logical1 '

message number: 2
....
GroupId[0x47726F757049643200000000000000000000000000000000] MsgSeqNumber[2] Offset[0] MsgFlags[30] OriginalLength[44]

data length: 44
00000000:   4772 6F75 7032 4C6F 6769 6361 6C32 5365 'Group2Logical2Se'

```
00000010:  676D 656E 7431 4772 6F75 7032 4C6F 6769  'gment1Group2Logi'
00000020:  6361 6C32 5365 676D 656E 7432            'cal2Segment2    '
```

* It can be confirmed that it is got in the order of logical messages, and the div
ided segments are one logical message.

* Lastly write the remaining messages of group 1.

$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment1 -gi GroupId1 -
ms 1 -of 0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_MSG_IN_GROUP
[18/05/25 16:46:16] 1: message length: 22 put message : Group1Logical1Segment1

* Since all the messages of group 1 have also been written at this point, mqpgf
waiting at MQGET() at another terminal will process the message and it will be
displayed.

message number: 3

....
GroupId[0x47726F757049643100000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[14] OriginalLength[44]

data length: 44
```
00000000:  4772 6F75 7031 4C6F 6769 6361 6C31 5365  'Group1Logical1Se'
00000010:  676D 656E 7431 4772 6F75 7031 4C6F 6769  'gment1Group1Logi'
00000020:  6361 6C31 5365 676D 656E 7432            'cal1Segment2    '
```

message number: 4

....
GroupId[0x47726F757049643100000000000000000000000000000000] MsgSeqNumbe
r[2] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 14
```
00000000:  4772 6F75 7031 4C6F 6769 6361 6C32       'Group1Logical2  '
```

message number: 5

....
GroupId[0x47726F757049643100000000000000000000000000000000] MsgSeqNumbe
r[3] Offset[0] MsgFlags[30] OriginalLength[44]

data length: 44
```
00000000:  4772 6F75 7031 4C6F 6769 6361 6C33 5365  'Group1Logical3Se'
00000010:  676D 656E 7431 4772 6F75 7031 4C6F 6769  'gment1Group1Logi'
00000020:  6361 6C33 5365 676D 656E 7432            'cal3Segment2    '
```

* Group 1 messages are also gotten in the order of logical messages, and it can

be confirmed that the divided segments are one logical message.
----

5. All parameters reference

5.1 Basic parameters

Queue Manager Name (-qm)

In all cases, queue manager name(-qm) to be connected is required.

mqpgf **-qm <qmgr>**....

Ex. 5.1.1 Connect to multiple queue managers
----
* When connecting from multiple threads to multiple queue managers, you need to create "shared (thread independent) connection" by specifying MQCNO_HANDL E_SHARE_BLOCK or MQCNO_HANDLE_SHARE_NO_BLOCK. In the example be low, "- tr" (simple API trace) is specified and the status of each MQI call is conf irmed together with the connection handle to each queue manager.

```
$ mqpgf -qm SampleQM,RemoteQM,PartialQM -q LQ1 -m test -tr MQCNO_HAN
DLE_SHARE_BLOCK MQPMO_SYNCPOINT
[18/07/26 16:24:38.817780] MQCONNX start qmgr:SampleQM Options:0x00000040
[18/07/26 16:24:38.876475] MQCONNX stop hcon:20971525 qmgr:SampleQM Comp
Cd=00 ReasonCd=00
[18/07/26 16:24:38.876659] MQCONNX start qmgr:RemoteQM Options:0x00000040
[18/07/26 16:24:38.912757] MQCONNX stop hcon:20971527 qmgr:RemoteQM Comp
Cd=00 ReasonCd=00
[18/07/26 16:24:38.912892] MQCONNX start qmgr:PartialQM Options:0x00000040
[18/07/26 16:24:38.958100] MQCONNX stop hcon:20971529 qmgr:PartialQM Comp
Cd=00 ReasonCd=00
[18/07/26 16:24:38.958741] MQOPEN start hcon:20971525 ObjectName:LQ1 Option
s:0x00000010
[18/07/26 16:24:38.959152] MQOPEN stop hcon:20971525 ObjectName:LQ1 CompC
d=00 ReasonCd=00
[18/07/26 16:24:38.959511] 1: message length: 4 put message: test
[18/07/26 16:24:38.959655] MQPUT start hcon:20971525 Options:0x00000000
[18/07/26 16:24:38.966288] MQPUT stop hcon:20971525 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.966422] MQCMIT start hcon:20971525
[18/07/26 16:24:38.967003] MQCMIT stop hcon:20971525 CompCd=00 ReasonCd=0
0
MQCMIT success : CompCd=00 ReasonCd=00
[18/07/26 16:24:38.967198] MQCLOSE start hcon:20971525 Options:0x00000000
```

[18/07/26 16:24:38.976426] MQCLOSE stop hcon:20971525 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.976530] MQOPEN start hcon:20971527 ObjectName:LQ1 Options:0x00000010
[18/07/26 16:24:38.976956] MQOPEN stop hcon:20971527 ObjectName:LQ1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.977041] 1: message length: 4 put message: test
[18/07/26 16:24:38.977168] MQPUT start hcon:20971527 Options:0x00000000
[18/07/26 16:24:38.982164] MQPUT stop hcon:20971527 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.982252] MQCMIT start hcon:20971527
[18/07/26 16:24:38.982710] MQCMIT stop hcon:20971527 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/07/26 16:24:38.982838] MQCLOSE start hcon:20971527 Options:0x00000000
[18/07/26 16:24:38.987732] MQCLOSE stop hcon:20971527 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.987812] MQOPEN start hcon:20971529 ObjectName:LQ1 Options:0x00000010
[18/07/26 16:24:38.988205] MQOPEN stop hcon:20971529 ObjectName:LQ1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.988289] 1: message length: 4 put message: test
[18/07/26 16:24:38.988414] MQPUT start hcon:20971529 Options:0x00000000
[18/07/26 16:24:38.994190] MQPUT stop hcon:20971529 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.994299] MQCMIT start hcon:20971529
[18/07/26 16:24:38.994816] MQCMIT stop hcon:20971529 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/07/26 16:24:38.994947] MQCLOSE start hcon:20971529 Options:0x00000000
[18/07/26 16:24:38.996036] MQCLOSE stop hcon:20971529 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.996115] MQDISC start hcon:20971525
[18/07/26 16:24:38.996454] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.996535] MQDISC start hcon:20971527
[18/07/26 16:24:38.996830] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.996917] MQDISC start hcon:20971529
[18/07/26 16:24:39.021688] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00

Queue Name (-q)

For queue operations, queue name (-q) is requested.

mqpgf -qm <qmgr> **-q <queue>**

Input Message (-m)

The input message to be put.

mqpgf -qm <qmgr> -q <queue> **-m <input message>**

e.g.
mqpgf -qm <qmgr> -q <queue> **-m "input message"**

Input Message(Hexadecimal notation) (-mx)

The input message in hexadecimal notation to be put.

mqpgf -qm <qmgr> -q <queue> **-mx <input message(hexadecimal notation)>**

Input File Name (-f)

The path of a message data file to be put.

mqpgf -qm <qmgr> -q <queue> **-f <input file path>**

e.g.
mqpgf -qm <qmgr> -q <queue> **-f work/inputMessage1.txt**

Output File Name (-o)

The path of a file to which a message data read is written.

mqpgf -qm <qmgr> -q <queue> **-o <output file path>**

e.g.
mqpgf -qm <qmgr> -q <queue> **-o work/outputMessage1.txt**

Output Queue Name (-oq)

Specify a queue to write a message data read.(cf. "4. Basic test" - "Write the read message to a queue(re-queue)")

A single character "*" can also be specified. If this character is specified, reply messages are written to the queue set in MQMD.ReplyToQ. When "**" is specified, MQMD.ReplyToQMgr is also set in MQOD.ObjectQMgrName.

In addition, if MQPMO_PASS_ * is specified, set the handle of an input queue to MQPMO.Context.(cf. "Ex. 6.9.1 Example of re-queuing while inheriting input identification, origin, user context")

If it is necessary to specify parameters on the writing side (secondary side), use the "-ss" option to switch the parameters to the writing side (secondary side). (cf. "5. All parameters reference" - "Switch subsequent paremetera to secondery (-ss)")

mqpgf -qm <qmgr> -q <queue> **-oq <output queue name>**

Input Queue Name (-iq)

Specify a queue to read a reply message after writing a message.(cf. "4. Basic test" - " Send messages and receive reply messages")

The queue name specified here is set MQMD.ReplyToQ of outgoing message.

If it is necessary to specify parameters on the reading side (secondary side), use the "-ss" option to switch the parameters to the reading side (secondary side). (cf. "5. All parameters reference" - "Switch subsequent paremetera to secondery (-ss)")

mqpgf -qm <qmgr> -q <queue> **-iq <input queue name>**

Input Directory Name (-d)

The name of a directory containing files to be put.

mqpgf -qm <qmgr> -q <queue> **-d <directory>**

e.g.
mqpgf -qm <qmgr> -q <queue> **-d "work/input message dir"**

Output Directory Name (-g)

The path of a directory to write messages on a queue.

mqpgf -qm <qmgr> -q <queue> **-g <directory>**

e.g.
mqpgf -qm <qmgr> -q <queue> **-g "work/output message dir"**

Get Repeatedly (-r)

Get all messages on a queue.

mqpgf -qm <qmgr> -q <queue> **-r**

Force Backout (-b)

Invoke MQBACK() after put or get operation.

mqpgf -qm <qmgr> -q <queue> -m <input message> -n <count> **-b MQPMO_SYN
CPOINT**
mqpgf -qm <qmgr> -q <queue> -r **-b MQGMO_SYNCPOINT**

Ex. 5.1.2 Call MQBACK() after put operation.

$ mqpcf que -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QLOCAL) **CURDEPTH(0)**

$ mqpgf -qm TESTQM -q TQ -m "test message" -n 3 **-b MQPMO_SYNCPOINT**
[16/12/22 20:30:59] 1: message length: 12 put message : test message
[16/12/22 20:30:59] 2: message length: 12 put message : test message
[16/12/22 20:30:59] 3: message length: 12 put message : test message
MQBACK success : CompCd=00 ReasonCd=00

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QUEUE) CURDEPTH(0)

Ex. 5.1.3 Call MQBACK() after get operation.

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QUEUE) **CURDEPTH(3)**

$ mqpgf -qm TESTQM -q TQ -r -b MQGMO_SYNCPOINT
[16/12/21 19:56:15] 1: message length: 12 get message : test message
[16/12/21 19:56:15] 2: message length: 12 get message : test message

[16/12/21 19:56:15] 3: message length: 12 get message : test message
no message available : TQ CompCd=02 ReasonCd=2033
MQBACK success : CompCd=00 ReasonCd=00

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QUEUE) **CURDEPTH(3)**

$ mqpgf -qm TESTQM -q TQ -br -r |grep MD
*StrucId[MD   ] .... **BackoutCount[1]** ....
*StrucId[MD   ] .... **BackoutCount[1]** ....
*StrucId[MD   ] .... **BackoutCount[1]** ....

The size of the message to be put (-l)

The size of a message to be put.

mqpgf -qm <qmgr> -q <queue> -m <input message> **-l <size>**
mqpgf -qm <qmgr> -q <queue> -mx <input message(hexadecimal notation)> **-l <si
ze>**
mqpgf -qm <qmgr> -q <queue> -f <filename> **-l <size>**
mqpgf -qm <qmgr> -q <queue> -d <directory> **-l <size>**

Ex. 5.1.4 Examples used with the '-m' option.

$ mqpgf -qm TESTQM -q TQ **-m 1234567890 -l 5**
[16/12/22 20:33:08] 1: message length: 5 put message : 12345

$ mqpgf -qm TESTQM -q TQ -br
....
data length: 5
00000000: 3132 3334 35   '12345   '

$ mqpgf -qm TESTQM -q TQ **-m 1234567890 -l 15**
[16/12/22 20:33:45] 1: message length: 15 put message : 1234567890.....

$ mqpgf -qm TESTQM -q TQ -br
....
data length: 15
00000000: 3132 3334 3536 3738 3930 0000 0000 00   '1234567890..... '

Ex. 5.1.5 Examples used with the '-mx' option.

$ mqpgf -qm TESTQM -q TQ **-mx 31323334353637383930 -l 5**
[16/12/22 20:34:36] 1: message length: 5 put message : 0x3132333435

$ mqpgf -qm TESTQM -q TQ -br
....
data length: 5
00000000:   3132 3334 35   '12345   '

$ mqpgf -qm TESTQM -q TQ **-mx 31323334353637383930 -l 15**
[16/12/22 20:35:52] 1: message length: 15 put message : 0x31323334353637383930
0000000000

$ mqpgf -qm TESTQM -q TQ -br
message number: 1
....
data length: 15
00000000:   3132 3334 3536 3738 3930 0000 0000 00   '1234567890..... '

Ex. 5.1.6 Examples used with the '-f' option.

$ ls -l input.txt
-rw-r--r--   1 guest   staff   6 Jun 15 18:44 input.txt

$ cat input.txt
123456

$ mqpgf -qm TESTQM -q TQ **-f input.txt -l 3**
[16/12/22 20:37:50] 1: put from input.txt

$ mqpgf -qm TESTQM -q TQ -br
....
data length: 3
00000000:   3132 33   '123   '

$ mqpgf -qm TESTQM -q TQ **-f input.txt -l 15**
[16/12/22 20:37:55] 1: put input.txt
$ mqpgf -qm TESTQM -q TQ -br
....
data length: 15
00000000:   3132 3334 3536 0000 0000 0000 0000 00   '123456......... '

Ex. 5.1.7 Examples used with the '-d' option.

$ ls -l input
total 24
-rw-r--r--   1 mq80   mqm   262 Oct 13 21:00 input1.txt
-rw-r--r--   1 mq80   mqm   262 Oct 13 21:00 input2.txt
-rw-r--r--   1 mq80   mqm   262 Oct 13 21:00 input3.txt

$ mqpgf -qm TESTQM -q TQ **-d input -l 3**
put input/input1.txt
put input/input2.txt
put input/input3.txt
% mqpgf -qm TESTQM -q TQ -br -r
message number: 1
....
data length: 3
00000000: 3530 30   '500   '

message number: 2
....
data length: 3
00000000: 3530 30   '500   '

message number: 3
....
data length: 3
00000000: 3530 30   '500   '

no message available : TQ CompCd=02 ReasonCd=2033

$ mqpgf -qm TESTQM -q TQ **-d input -l 500**
put input/input1.txt
put input/input2.txt
put input/input3.txt
$ mqpgf -qm TESTQM -q TQ -r
message length: 500
....
message length: 500
....
message length: 500
....
no message available : TQ CompCd=02 ReasonCd=2033

Message count for writing or reading (-n)

The number of messages to put or get.
Specifies the number of messages to PUT/GET. It can be used with -i (Interval f
or writing or reading) / -m (message) / -mx (message hexadecimal notation) / -f (P
UT file) / -d (PUT files in directory) / -g (Output to directory) option etc..

PUT Time:
mqpgf -qm <qmgr> -q <queue> -m <input message> **-n <count>**
mqpgf -qm <qmgr> -q <queue> -mx <input message(hexadecimal notation)> **-n <c
ount>**
mqpgf -qm <qmgr> -q <queue> -f <filename> **-n <count>**
mqpgf -qm <qmgr> -q <queue> -d <directory> **-n <count>**

GET Time:
mqpgf -qm <qmgr> -q <queue> **-n <count>**
mqpgf -qm <qmgr> -q <queue> -g <directory> **-n <count>**

Ex. 5.1.8 An example used with the '-d' option.

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QUEUE) **CURDEPTH(0)**

$ ls -l input
total 24
-rw-r--r--   1 mq80   mqm   262 Oct 13 21:00 input1.txt
-rw-r--r--   1 mq80   mqm   262 Oct 13 21:00 input2.txt
-rw-r--r--   1 mq80   mqm   262 Oct 13 21:00 input3.txt

$ mqpgf -qm TESTQM -q TQ -d input -n 3
[16/12/22 20:41:24] 1: put from input/test1.txt
[16/12/22 20:41:24] 2: put from input/test1.txt
[16/12/22 20:41:24] 3: put from input/test1.txt
[16/12/22 20:41:24] 1: put from input/test2.txt
[16/12/22 20:41:24] 2: put from input/test2.txt
[16/12/22 20:41:24] 3: put from input/test2.txt
[16/12/22 20:41:24] 1: put from input/test3.txt
[16/12/22 20:41:24] 2: put from input/test3.txt
[16/12/22 20:41:24] 3: put from input/test3.txt

> mqpcf ques -qm TESTQM -q TQ CURDEPTH

1: QUEUE(TQ) TYPE(QUEUE) **CURDEPTH(9)**

Interval for writing or reading (-i)

Specifies the number of messages to PUT or GET. It can be used with -i (Interval for writing or reading) / -m (message) / -mx (message hexadecimal notation) / -f (PUT file) / -d (PUT files in directory) / -g (output to directory) option etc..

PUT Time:
mqpgf -qm <qmgr> -q <queue> -m <input message> -n <count> **-i <interval(ms)>**
mqpgf -qm <qmgr> -q <queue> -m <input message> -d <directory> **-i <interval(ms)>**

GET Time:
mqpgf -qm <qmgr> -q <queue> -n <count> **-i <interval(ms)>**
mqpgf -qm <qmgr> -q <queue> -g <directory> -n <count> **-i <interval(ms)>**

Maximum size of message to read (-sz)

The buffer size to get messages.

Ex. 5.1.9 Example of getting a message with a receive buffer larger than the default size.

$ mqpgf -qm TESTQM -q TQ -m "put 1Mbyte" **-l 1000000**
[16/12/22 20:48:15] 1: message length: 1000000 put message : put 1Mbyte...

$ mqpgf -qm TESTQM -q TQ
MQGET with warning : TQ CompCd=01 **ReasonCd=2080** len=1000000

$ mqrc 2080

2080   0x00000820   **MQRC_TRUNCATED_MSG_FAILED**

$ mqpgf -qm TESTQM -q TQ **-sz 1000000**
[16/12/22 20:48:57] 1: message length: 1000000 get message : put 1Mbyte...

Ex. 5.1.10 Example of getting a message with a receive buffer smaller than the default size.
```
------------------------------------
$ mqpgf -qm TESTQM -q TQ -m "put 1Mbyte" -l 1000000
[16/12/22 20:48:20] 1: message length: 1000000 put message : put 1Mbyt
e.........................................................................................................

$ mqpgf -qm TESTQM -q TQ -sz 10 MQGMO_ACCEPT_TRUNCATED_MSG
MQGET with warning : TQ CompCd=01 ReasonCd=2079 len=1000000
message length: 1000000
get message : put 1Mbyte
$ mqrc 2079

    2079   0x0000081f   MQRC_TRUNCATED_MSG_ACCEPTED

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QUEUE) CURDEPTH(0)
------------------------------------
```

Size of messages written to standard output (-ds)

The size to write messages read to stdout or the size to echo back to stdout when specifying message to put on command line. Both default sizes are 128 bytes.

mqpgf -qm <qmgr> -q <queue> -sz 100000 -ds <display size or all>
mqpgf -qm <qmgr> -q <queue> -m "large message" -l 100000 -ds <display size or all>

Ex. 5.1.11 Adjust display message size when putting or getting large messages.
```
------------------------------------
$ mqpgf -qm TESTQM -q TQ -m "put 1Mbyte" -l 1000000 -ds 256
[16/12/22 21:24:09] 1: message length: 1000000 put message : put 1Mbyt
e...........................................................................................................................
.......................................................................................
$ mqpgf -qm TESTQM -q TQ -sz 1000000 -ds 256
[16/12/22 21:24:15] 1: message length: 1000000 get message : put 1Mbyt
e...........................................................................................................................
......................................................................................
------------------------------------
```

Ex. 5.1.12 Display an entire message read from a queue.

$ mqpgf -qm TESTQM -q TQ **-m "put 256byte"** -l 256 -ds all
  [16/12/22 21:39:25] 1: message length: 256 put message : put 256byt
e...
....
$ mqpgf -qm TESTQM -q TQ -ds all
[16/12/22 21:39:33] 1: message length: 256 get message : put 256byt
e...
....

Browse and dump messages on a queue(normal mode) (-br)

Browse messages on a queue and dump them in hexadecimal.

mqpgf -qm <qmgr> -q <queue> **-br** -r

Browse and dump messages on a queue(verbose mode) (-brv)

Browse messages on a queue and dump them in hexadecimal with verbose m
ode.

mqpgf -qm <qmgr> -q <queue> **-brv** -r

GET and dump messages on a queue(normal mode) (-dp)

This option is the same as "-br", except that it actually gets message, not br
owse mode.

mqpgf -qm <qmgr> -q <queue> **-dp** -r

GET and dump messages on a queue(verbose mode) (-dpv)

This option is the same as "-dpv", except that it actually gets message, not b
rowse mode.

mqpgf -qm <qmgr> -q <queue> **-dpv** -r

Write a message read from a queue to standard output(raw mode) (-raw)

Write a message read from a queue to stdout without changing as it is.

mqpgf -qm <qmgr> -q <queue> **-brv -raw**
mqpgf -qm <qmgr> -q <queue> **-o <output file path> -raw**
mqpgf -qm <qmgr> -q <queue> **-raw > <output file path>** (* Redirect standard output)

Write a message read from a queue to standard output(Hexadecimal notation) (-hex)

Write a message read from a queue to stdout in hexadecimal notation.

mqpgf -qm <qmgr> -q <queue> **-hex** -r

Stop before invoking a specified MQI function (-s)

It stops just before a specified MQI function is called. Input of any key will restart the process.

The MQI functions that can be specified are MQCONNX, MQCONN, MQPUT, MQOPEN, MQGET, MQSET, MQINQ, MQCLOSE, MQDISC, MQBACK, MQCMIT, MQCRTMH, MQDLTMH, MQSETMP, MQINQMP.
MQCONNX is used when making a client connection with the -x option and when specifying MQCNO_* option other than MQCNO_NONE.
MQCMIT is called if you specify his MQPMO_SYNCPOINT or MQGMO_SYNCPOINT.
MQBACK is called when MQPMO_SYNCPOINT or MQGMO_SYNCPOINT and "force backout(-b)" are specified.
When "GET all messages on the queue(-r)" and MQGMO_SYNCPOINT, or "Message count for writing or reading(-n)" and MQPMO_SYNCPOINT are used together, MQCMIT/MQBACK is called once (same UOW).
MQCRTMH/MQDLTMH is used when PUT by "specifying message property(-smp)" and when GET by specifying MQGMO_PROPERTIES_IN_HANDLE.
MQSETMP is used when PUT with "specifying message property(-smp)".
MQINQMP is used when GET with MQGMO_PROPERTIES_IN_HANDLE.

mqpgf -qm <qmgr> -q <queue> **-s <MQI function name>** -r

Ex. 5.1.13 Example of specifying "-s MQGET" with '-r' option.

$ mqpgf -qm TESTQM -q TQ -m "sample message" -n 2
[16/12/22 21:31:19] 1: message length: 14 put message : sample message
[16/12/22 21:31:19] 2: message length: 14 put message : sample message
$ mqpgf -qm TESTQM -q TQ **-s MQGET -r**
stop before calling MQGET().
Hit Any Key!!!
[16/12/22 21:31:26] 1: message length: 14 get message : sample message
stop before calling MQGET().
Hit Any Key!!!
[16/12/22 21:31:28] 2: message length: 14 get message : sample message
stop before calling MQGET().
Hit Any Key!!!
no message available : TQ CompCd=02 ReasonCd=2033

Process Name (-p)

It is specified together with -inq (MQINQ call).

mqpgf -qm <qmgr> **-p <process>** -inq: **selecter**(e.g. MQCA_APPL_ID, MQCA_ENV
_DATA,..) **MQOT_PROCESS**

Name List (-nl)

It is specified together with -inq (MQINQ call).

mqpgf -qm <qmgr> **-nl <namelist>** -inq: **selecter**(e.g. MQIA_NAMELIST_TYPE,M
MQCA_NAMES,..) **MQOT_NAMELIST**

Pcf format file (-pcf)

By writing the following PCF definition in a plain text file, you can create a binary
PCF format and put it to the specified queue. It can be specified with -n (specify the
number of messages to PUT/GET) and -i (Interval for writing or reading). (Refer to "4.
Basic test- Create and put a message in pcf format")

Switch subsequent paremetera to secondery (-ss)

Use this option with either "-iq" or "-oq". Subsequent parameters of this option are used for the queue specified by "-iq" or "-oq".

Switch subsequent paremetera to primary (-sp)

Use this option with either "-iq" or "-oq". Subsequent parameters of this option are used for the queue specified by "-q".

Get a message with the same CorrelId as the MsgId sent(-mc)

Use this option with "-iq". It gets a message with same CorrelID as MsgId sent. When using this option, MQMO_MATCH_CORREL_ID is automatically used. (MQGMO_DEFAULT MatchOptions is "MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID")

Inherit the received MQMD (-im)

Use this option with "-oq". The MQMD of a received message is copied to an outgoing message. However, message context information is not inherited by this option. This option is not necessary to inherit the message context. See "Ex. 6.9.1 Example of re-queuing while inheriting input identification, origin, user context " for detail.

Segmentation size (-as)

Instructs the application (not the queue manager) to perform segmentation at the specified size. It is specified in bytes. For details, see "Ex. 4.21.1 Application segmentation.".

Delimiter for logical messages (-dl)

It creates multiple logical messages from the message specified by "-m" or "-f ". The delimiter is specified as a character string or hexadecimal notation. Del imiters are not included in the created logical message. For details, refer to " Ex. 4.24.1 Grouping Logical Messages".

Number of threads (-nt)

If you specify the number of threads with this parameter, the main thread c reates a connection handle and calls MQI() other than MQDISC() using that h andle for each specified number of child threads.

Ex. 5.1.14 Specifying the number of startup threads
----
* In order to share connections between threads, it is necessary to create "share d (thread independent) connection" by specifying MQCNO_HANDLE_SHARE_ BL OCK or MQCNO_HANDLE_SHARE_NO_BLOCK. If "2219 MQRC_CALL_IN_PRO GRESS" is returned by specifying MQCNO_HANDLE_SHARE_NO_BLOCK, mqpgf will not retry calling MQI. Therefore, only MQCNO_HANDLE_SHARE_BLOCK i s effective when calling MQI in multi-thread.

$ mqpgf -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnn) MQCNO _CLIENT_BINDING -nt 3 -tr MQCNO_HANDLE_SHARE_BLOCK
[2018/07/26 18:19:10.208 tid=0] MQCONNX start qmgr:SampleQM Options:0x0000 0840
[2018/07/26 18:19:10.333 tid=0] MQCONNX stop hcon:33554437 qmgr:SampleQM CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.333 tid=31108] MQOPEN start hcon:33554437 ObjectName:S ampleQ Options:0x00000010
[2018/07/26 18:19:10.333 tid=38164] MQOPEN start hcon:33554437 ObjectName:S ampleQ Options:0x00000010
[2018/07/26 18:19:10.333 tid=38156] MQOPEN start hcon:33554437 ObjectName:S ampleQ Options:0x00000010
[2018/07/26 18:19:10.333 tid=31108] MQOPEN stop hcon:33554437 ObjectName:Sa mpleQ CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.333 tid=31108] 1: message length: 4 put message: test
[2018/07/26 18:19:10.349 tid=38164] MQOPEN stop hcon:33554437 ObjectName:Sa mpleQ CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.349 tid=31108] MQPUT start hcon:33554437 Options:0x00000 000
[2018/07/26 18:19:10.349 tid=38156] MQOPEN stop hcon:33554437 ObjectName:Sa mpleQ CompCd=00 ReasonCd=00

[2018/07/26 18:19:10.349 tid=38164] 1: message length: 4 put message: test
[2018/07/26 18:19:10.365 tid=38164] MQPUT start hcon:33554437 Options:0x00000
000
[2018/07/26 18:19:10.365 tid=38156] 1: message length: 4 put message: test
[2018/07/26 18:19:10.365 tid=38156] MQPUT start hcon:33554437 Options:0x00000
000
[2018/07/26 18:19:10.380 tid=31108] MQPUT stop hcon:33554437 CompCd=00 Rea
sonCd=00
[2018/07/26 18:19:10.380 tid=31108] MQCLOSE start hcon:33554437 Options:0x00
000000
[2018/07/26 18:19:10.396 tid=38164] MQPUT stop hcon:33554437 CompCd=00 Rea
sonCd=00
[2018/07/26 18:19:10.396 tid=38164] MQCLOSE start hcon:33554437 Options:0x00
000000
[2018/07/26 18:19:10.396 tid=38156] MQPUT stop hcon:33554437 CompCd=00 Rea
sonCd=00
[2018/07/26 18:19:10.396 tid=38156] MQCLOSE start hcon:33554437 Options:0x00
000000
[2018/07/26 18:19:10.412 tid=31108] MQCLOSE stop hcon:33554437 CompCd=00 R
easonCd=00
[2018/07/26 18:19:10.427 tid=38164] MQCLOSE stop hcon:33554437 CompCd=00 R
easonCd=00
[2018/07/26 18:19:10.443 tid=38156] MQCLOSE stop hcon:33554437 CompCd=00 R
easonCd=00
[2018/07/26 18:19:10.458 tid=0] MQDISC start hcon:33554437
[2018/07/26 18:19:10.490 tid=0] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00

Number of threads that call MQCONN/MQDISC internally (-ni)

If you specify the number of threads with this parameter, MQCONN () / MQ
CONNX () and MQDISC () are called in the child thread started from the mai
n thread.In this case, there is no need to create a "shared (thread independen
t) connection" by specifying MQCNO_HANDLE_SHARE_BLOCK.

Ex. 5.1.15 Number of threads that call MQCONN/MQDISC internally
----
$ mqpgf -qm SampleQM -q SampleQ -m "thread test" -ni 3 -tr MQPMO_SYNCP
OINT
[18/08/07 17:56:12.161562 tid=44144] MQCONN start qmgr:SampleQM
[18/08/07 17:56:12.163679 tid=47424] MQCONN start qmgr:SampleQM

[18/08/07 17:56:12.163723 tid=50704] MQCONN start qmgr:SampleQM
[18/08/07 17:56:12.284938 tid=44144] MQCONN stop hcon:20971526 qmgr:Sample
QM CompCd=00 ReasonCd=00
[18/08/07 17:56:12.285045 tid=44144] MQOPEN start hcon:20971526 ObjectName:
SampleQ Options:0x00000010
[18/08/07 17:56:12.285394 tid=44144] MQOPEN stop hcon:20971526 ObjectName:S
ampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:12.285453 tid=44144] 1: message length: 11 put message: thread t
est
[18/08/07 17:56:12.285557 tid=44144] MQPUT start hcon:20971526 Options:0x0000
0000
[18/08/07 17:56:12.300966 tid=47424] MQCONN stop hcon:20971528 qmgr:Sample
QM CompCd=00 ReasonCd=00
[18/08/07 17:56:12.301005 tid=47424] MQOPEN start hcon:20971528 ObjectName:
SampleQ Options:0x00000010
[18/08/07 17:56:12.308055 tid=50704] MQCONN stop hcon:20971530 qmgr:Sample
QM CompCd=00 ReasonCd=00
[18/08/07 17:56:12.308091 tid=50704] MQOPEN start hcon:20971530 ObjectName:
SampleQ Options:0x00000010
[18/08/07 17:56:12.308375 tid=44144] MQPUT stop hcon:20971526 CompCd=00 Re
asonCd=00
[18/08/07 17:56:12.308421 tid=44144] MQCMIT start hcon:20971526
[18/08/07 17:56:12.308476 tid=47424] MQOPEN stop hcon:20971528 ObjectName:S
ampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:12.308536 tid=47424] 1: message length: 11 put message: thread t
est
[18/08/07 17:56:12.308581 tid=47424] MQPUT start hcon:20971528 Options:0x0000
0000
[18/08/07 17:56:12.308634 tid=50704] MQOPEN stop hcon:20971530 ObjectName:S
ampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:12.308673 tid=50704] 1: message length: 11 put message: thread t
est
[18/08/07 17:56:12.308707 tid=50704] MQPUT start hcon:20971530 Options:0x0000
0000
[18/08/07 17:56:12.308753 tid=47424] MQPUT stop hcon:20971528 CompCd=00 Re
asonCd=00
[18/08/07 17:56:12.308787 tid=47424] MQCMIT start hcon:20971528
[18/08/07 17:56:12.308830 tid=50704] MQPUT stop hcon:20971530 CompCd=00 Re
asonCd=00
[18/08/07 17:56:12.308870 tid=50704] MQCMIT start hcon:20971530
[18/08/07 17:56:12.309450 tid=44144] MQCMIT stop hcon:20971526 CompCd=00 R
easonCd=00
MQCMIT success：CompCd=00 ReasonCd=00
[18/08/07 17:56:12.309516 tid=44144] MQCLOSE start hcon:20971526 Options:0x0
0000000

[18/08/07 17:56:12.309560 tid=47424] MQCMIT stop hcon:20971528 CompCd=00 R
easonCd=00
MQCMIT success：CompCd=00 ReasonCd=00
[18/08/07 17:56:12.309618 tid=47424] MQCLOSE start hcon:20971528 Options:0x0
0000000
[18/08/07 17:56:12.309660 tid=50704] MQCMIT stop hcon:20971530 CompCd=00 R
easonCd=00
MQCMIT success：CompCd=00 ReasonCd=00
[18/08/07 17:56:12.309716 tid=50704] MQCLOSE start hcon:20971530 Options:0x0
0000000
[18/08/07 17:56:12.309758 tid=44144] MQCLOSE stop hcon:20971526 CompCd=00
ReasonCd=00
[18/08/07 17:56:12.309791 tid=44144] MQDISC start hcon:20971526
[18/08/07 17:56:12.309835 tid=47424] MQCLOSE stop hcon:20971528 CompCd=00
ReasonCd=00
[18/08/07 17:56:12.309866 tid=47424] MQDISC start hcon:20971528
[18/08/07 17:56:12.315516 tid=50704] MQCLOSE stop hcon:20971530 CompCd=00
ReasonCd=00
[18/08/07 17:56:12.315556 tid=50704] MQDISC start hcon:20971530
[18/08/07 17:56:12.316555 tid=44144] MQDISC stop hcon:-1 CompCd=00 ReasonCd
=00
[18/08/07 17:56:12.317559 tid=47424] MQDISC stop hcon:-1 CompCd=00 ReasonCd
=00
[18/08/07 17:56:12.321606 tid=50704] MQDISC stop hcon:-1 CompCd=00 ReasonCd
=00

$ mqpgf -qm SampleQM -q SampleQ -ni 3 -tr MQGMO_SYNCPOINT
[18/08/07 17:56:19.812864 tid=44144] MQCONN start qmgr:SampleQM
[18/08/07 17:56:19.814992 tid=47424] MQCONN start qmgr:SampleQM
[18/08/07 17:56:19.815036 tid=50704] MQCONN start qmgr:SampleQM
[18/08/07 17:56:19.936089 tid=44144] MQCONN stop hcon:20971526 qmgr:Sample
QM CompCd=00 ReasonCd=00
[18/08/07 17:56:19.936178 tid=44144] MQOPEN start hcon:20971526 ObjectName:
SampleQ Options:0x00000001
[18/08/07 17:56:19.936448 tid=44144] MQOPEN stop hcon:20971526 ObjectName:S
ampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:19.936496 tid=44144] MQGET start hcon:20971526 Options:0x0000
0000
[18/08/07 17:56:19.936588 tid=47424] MQCONN stop hcon:20971528 qmgr:Sample
QM CompCd=00 ReasonCd=00
[18/08/07 17:56:19.936624 tid=47424] MQOPEN start hcon:20971528 ObjectName:
SampleQ Options:0x00000001
[18/08/07 17:56:19.936696 tid=50704] MQCONN stop hcon:20971530 qmgr:Sample
QM CompCd=00 ReasonCd=00
[18/08/07 17:56:19.936731 tid=50704] MQOPEN start hcon:20971530 ObjectName:

SampleQ Options:0x00000001
[18/08/07 17:56:19.949521 tid=44144] MQGET stop hcon:20971526 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949560 tid=44144] 1: message length: 4 get message：test
[18/08/07 17:56:19.949636 tid=44144] MQCMIT start hcon:20971526
[18/08/07 17:56:19.949686 tid=47424] MQOPEN stop hcon:20971528 ObjectName:SampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949757 tid=47424] MQGET start hcon:20971528 Options:0x00000000
[18/08/07 17:56:19.949820 tid=50704] MQOPEN stop hcon:20971530 ObjectName:SampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949866 tid=50704] MQGET start hcon:20971530 Options:0x00000000
[18/08/07 17:56:19.949924 tid=47424] MQGET stop hcon:20971528 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949959 tid=47424] 1: message length: 4 get message：test
[18/08/07 17:56:19.949996 tid=47424] MQCMIT start hcon:20971528
[18/08/07 17:56:19.950041 tid=50704] MQGET stop hcon:20971530 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950083 tid=50704] 1: message length: 4 get message：test
[18/08/07 17:56:19.950118 tid=50704] MQCMIT start hcon:20971530
[18/08/07 17:56:19.950521 tid=44144] MQCMIT stop hcon:20971526 CompCd=00 ReasonCd=00
MQCMIT success：CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950600 tid=44144] MQCLOSE start hcon:20971526 Options:0x00000000
[18/08/07 17:56:19.950649 tid=47424] MQCMIT stop hcon:20971528 CompCd=00 ReasonCd=00
MQCMIT success：CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950714 tid=47424] MQCLOSE start hcon:20971528 Options:0x00000000
[18/08/07 17:56:19.950761 tid=50704] MQCMIT stop hcon:20971530 CompCd=00 ReasonCd=00
MQCMIT success：CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950825 tid=50704] MQCLOSE start hcon:20971530 Options:0x00000000
[18/08/07 17:56:19.950871 tid=44144] MQCLOSE stop hcon:20971526 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950908 tid=44144] MQDISC start hcon:20971526
[18/08/07 17:56:19.950955 tid=47424] MQCLOSE stop hcon:20971528 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950989 tid=47424] MQDISC start hcon:20971528
[18/08/07 17:56:19.956575 tid=50704] MQCLOSE stop hcon:20971530 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.956619 tid=50704] MQDISC start hcon:20971530

```
[18/08/07 17:56:19.957617 tid=44144] MQDISC stop hcon:-1 CompCd=00 ReasonCd
=00
[18/08/07 17:56:19.958623 tid=47424] MQDISC stop hcon:-1 CompCd=00 ReasonCd
=00
[18/08/07 17:56:19.962197 tid=50704] MQDISC stop hcon:-1 CompCd=00 ReasonCd
=00
-------------------------------------
```

Enable api trace (-tr)

Displays a simple API trace for MQI calls. When "-nt" or "-ni" is specified and
mqpgf is executed in multi-thread, the Thread ID("tid") of the thread that called
MQI is also displayed. The notations "start" and "stop" mean the start and
completion of calling MQI respectively. It displays the option specified at start,
completion code, reason code on completion.

The file for synchronization start (-sf)

With this option, after every MOCONN() or MQCONNX() and subsequent opening
of the I/O queues, the specified file will be opened and checked for existence at 1ms
intervals, and processing will start after the file is successfully opened. This can be
used when starting multiple mqpgf(c) processes or specifying multiple queue
managers in the -qm option, and wanting to synchronize the start of processing
after connecting to the queue manager and opening the input/output queues. This is
useful when performing performance tests in some environments where process
startup takes a long time and processing starts inconsistently. This option is invalid
when "Number of threads that call MQCONN/MQDISC internally (-ni)" is specified.

Connection loop count (-c)

This parameter is the number of iterations from MQCONN(X)() to MQDISC(). When
used with "Wait time to next processing (-wp)", you can specify the repetition interval.

Skip MQDISC (-sd)

This option makes MQDISC() from being invoked.

Wait time to next processing (-wp)

This is the interval time from MQDISC() to the next MQCONN(X)() when used with "Connection loop count(-c)" and the retry interval for MQCONN(X)() when used with "The number of connection retry(-cr)". Specify in ms units.

Continue processing after MQI fails (-ca)

It is basically used with "connection repeat count(- c)". If this option is specified, the next iteration starting with MQCONN() will continue even if each MQI fails.

Applend the counter to message automatically (-ac)

This option is used with "Input message (-m)". When this option is specified, an 8-digit message counter ("nnnnnnnn:") is automatically added at the beginning of each sent message.

The number of connection retry (-cr)

This parameter is the number of times MQCONN(X)() was recalled if MQCONN (X)() failed. It reconnects at the interval specified in "Wait time to next processing (-wp)".

5.2 Platform-specific options

Using Global UOW for NSK (-gt)

This option is only available with HPE NonStop. For the single thread versio
n (mqpgfs, mqpgfcs), the TMF transaction API, BEGINTRANSACTION / END
TRANSACTION / ABORTTRANSACTION, is used, and for the multithreaded
version (mqpgf, mqpgfc), PUT_BEGINTRANSACTION / PUT_ENDTRANSACTI
ON / PUT_ABORTTRANSACTION is used. Even if multiple QMGRs are specif
ied, the TMF API will be called only once in the entire process.

Using Global UOW for NSK(TMFAPI: per PUT/GET) (-gti)

This option is only available with HPE NonStop. For the single thread versio
n (mqpgfs, mqpgfcs), the TMF transaction API, BEGINTRANSACTION / END
TRANSACTION / ABORTTRANSACTION, is used, and for the multithreaded
version (mqpgf, mqpgfc), PUT_BEGINTRANSACTION / PUT_ENDTRANSACTI
ON / PUT_ABORTTRANSACTION is used. If multiple QMGRs are specified, t
he TMF API will be called for each QMGR. If -oq (Queue to Queue) is specified
 with -n or -r, BEGINTRANSACTION / ENDTRANSACTION is called every MQG
ET to MQPUT. If -iq (input queue name) (Send and Receive) is specified with -n,
BEGINTRANSACTION / ENDTRANSACTION is called for each MQPUT and MQ
GET.

5.3 Specify the MQI function to be called

MQSET (-set)

Invoke MQSET function for a specified queue.

MQINQ (-inq)

Invoke MQINQ() to query attributes of a specified queue (local, remote, alias), name list, process and queue manager.

MQSETMP (-smp)

Put message with specifying message properties of arbitrary data type

5.4 MQCD Fields

Hereinafter, the notation of the title is the field name (option) (data type) (default value).

ConnectionName (-x) (MQCHAR264)(-)

The ConnectinName of the channel definition structure.

ChannelName (-ch) (MQCHAR20)(-)

The ChannelName of the channel definition structure.

LocalAddress (-la) (MQCHAR48)(-)

Specify LocalAddress of MQCD channel definition structure. If you specify this v alue, mqpgf automatically sets MQCD_VERSION_7 to MQCD.Version as an excepti on.

CertificateLabel (-cl) (MQCHAR64)(-)

Specify CertificateLabel of MQCD channel definition structure. It is necessary to specify MQCD_VERSION_11 or higher. It is available when trying an SSL connect ion in client mode.

SSLCipherSpec (-cs) (MQCHAR32)(-)

Specify SSLCipherSpec of MQCD channel definition structure. It is necessary to specify MQCD_VERSION_7 or higher. It is available when trying an SSL connecti on in client mode.

SSLPeerName (-er) (-)(-)

Specify the string for validating the SSL Peer name within MQ_SSL_PEER_NA ME_LENGTH (1024) bytes. mqpgf automatically sets mqcd.SSLPeerNamePtr and

SSLPeerNameLength of the MQCD channel definition structure from the specified string. It is necessary to specify MQCD_VERSION_7 or higher. It is available when trying an SSL connection in client mode.

5.5 MQMD Fields

In this section, the fields that specifing arbitrary values (numeric value, character strings and binary) that are not constant (such as MQFB_*) in MQMD field are described.

Ex. 5.5.1 Example of specifying MsgId and CorrelId

$ mqpgf -qm TESTQM -q TQ -f input.txt **-mi 0x0123456789abcedfABCDEF -ci Co**
rrelId
[16/12/07 20:01:29] 1: put from input.txt
$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
.... **MsgId[0x0123456789ABCEDFABCDEF0000000000000000000000000] CorrelId**
[0x436F7272656C49640000000000000000000000000000000000000] ....

Ex. 5.5.2 Example of specifying UserIdentifier, AccountingToken and
ApplIdentityData.

$ mqpgf -qm TESTQM -q TQ -f input.txt **-ui testuser -at 0x6163636F756E74696E**
67746F6B656E -ap applidentitydata
[16/12/07 20:29:37] 1: put from input.txt

$ mqpgf -qm TESTQM -q TQ -br
message number: 1
.... **UserIdentifier[mq80   ] AccountingToken[0x0332343200000000000000000000**
0000000000000000000000000000000000000000000006] ApplIdentityData[
            ] ....

$ mqpgf -qm TESTQM -q TQ -f input.txt **-ui testuser -at 0x6163636F756E74696E**
67746F6B656E -ap applidentitydata MQPMO_SET_IDENTITY_CONTEXT
[16/12/07 20:33:03] 1: put from input.txt
MQPUT fail : TQ CompCd=02 **ReasonCd=2096**

$ mqrc 2096

    2096   0x00000830   **MQRC_NOT_OPEN_FOR_SET_IDENT**

$ mqpgf -qm TESTQM -q TQ -f input.txt -ui testuser **-at 0x6163636F756E74696E**
67746F6B656E -ap applidentitydata MQPMO_SET_IDENTITY_CONTEXT MQOO_
SET_IDENTITY_CONTEXT
[16/12/07 20:36:47] 1: put from input.txt
> mqpgf -qm TESTQM -q TQ -br

message number: 1
.... UserIdentifier[testuser   ] AccountingToken[0x6163636F756E74696E67746F6B
656E0000000000000000000000000000000000] ApplIdentityData[applidentitydata
                ] ....

Ex. 5.5.3 Example of specifying PutApplName, PutDate, PutTime, ApplOriginDat
a.

> mqpgf -qm TESTQM -q TQ -f input.txt **-pn testapl -pd 20140101 -pt 00112233
-ao orig** MQPMO_SET_ALL_CONTEXT MQOO_SET_ALL_CONTEXT MQAT_WIN
DOWS
 [16/12/07 20:45:56] 1: put from input.txt
> mqpgf -qm TESTQM -q TQ -br
message number: 1
.... **PutApplType[9] PutApplName[testapl   ] PutDate[20140101]
 PutTime[00112233] ApplOriginData[orig]** ....

   Hereinafter, the notation of the title is the field name (option) (data type) (default
value).

Expiry (-ex) (MQLONG)(MQEI_UNLIMITED)

Message exiry in 100 ms increments.

Encoding (-ec) (MQLONG)(MQENC_NATIVE)

Numeric encoding of message data.

Ex. 5.5.4 Example of specifying the encoding of MQRFH2 header.

$ mqpgf -qm TESTQM -q TQ -m "test" **-ec 0x222** -re 273 -rc 1208 -rf MQFMT_S
TRING -fg 100 -nc 1208 -nd "test1,test22,test333" MQFMT_RF_HEADER_2
[17/01/06 20:09:54] 1: message length: 4 put message : test

$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] **Encoding**

[546] CodedCharSetId[943] **Format[MQHRF2]** ....

....
*StrucId[RFH] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[1208]
Format[MQSTR   ] Flags[100] NameValueCCSID[943]
NameValueLength[8] NameValueData[test1   ]
NameValueLength[8] NameValueData[test22   ]
NameValueLength[8] NameValueData[test333]
data length: 76
00000000: 7465 7374   'test   '

```
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
```
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] **Encoding**
[546] CodedCharSetId[943] **Format[MQHRF2]** ....

data length: 76
00000000: 5246 4820 **0200 0000 4800 0000 1101 0000**   'RFH ....H.......'
00000010: **B804 0000** 4D51 5354 5220 2020 **6400 0000**   'ﾘ...MQSTR   d...'
00000020: **B804 0000 0800 0000** 7465 7374 3120 2020   '........test1   '
00000030: **0800 0000** 7465 7374 3232 2020 **0800 0000**   '....test22 ....'
00000040: 7465 7374 3333 3320 7465 7374   'test333 test   '

CodedCharSetId (-cc) (MQLONG)(MQCCSI_Q_MGR)

Coded Character Set ID of message data.

Priority (-pr) (MQLONG)(MQPRI_PRIORITY_AS_Q_DEF)

Priority of a message.

MsgId (-mi) (MQBYTE24)(MQMI_NONE_ARRAY)

Identifier of a message.

CorrelId (-ci) (MQBYTE24)(MQCI_NONE_ARRAY)

Correlation Identifier of a message.

ReplyToQ (-rq) (MQCHAR48)("")

Name of a response queue.

ReplyToQMgr (-rm) (MQCHAR48)("")

Name of a response queue manager.

UserIdentifier (-ui) (MQCHAR12)("")

User Identifier which is a part of Identity Context.

AccountingToken (-at) (MQBYTE32)(MQACT_NONE_ARRAY)

Accounting Token which is a part of Ideneity Context.

ApplIdentityData (-ap) (MQCHAR32)("")

Application Identity Data which is a part of Ideneity Context.

PutApplName (-pn) (MQCHAR28)("")

Application Name put a message which is a part of Origin Context.

PutDate (-pd) (MQCHAR8)("")

Date when the message was put which is a part of Origin Context.

PutTime (-pt) (MQCHAR8)("")

Time when the message was put which is a part of Origin Context.

ApplOriginData (-ao) (MQCHAR4)("")

This field is defined by the application suite and can be used to provide additional information about the source of the message. This is a part of Origin Context.

5.6 MQMD Version 2 Fields

This program uses MQMD_VERSION_1 by default. (As an exception, when browse option (-br / -brv) is specified, MQMD_VERION_ 2 is used by default.) When using the field of MQMD Version 2 except when browsing, it is necessary to specify MQMD_VERSION_2 constant.

Ex. 5.6.1 Example of specifying MQMD Version 2

$ mqpgf -qm TESTQM -q TQ -f input.txt **-gi 0x67726F757069640000000000000000 0000000000000000000 -ms 3 -of 100 -ol 1000 MQMD_VERSION_2 MQMF_SEGM ENT MQMT_REPORT MQMF_MSG_IN_GROUP**
[16/12/23 00:11:41] 1: put from input.txt

$ mqpgf -qm TESTQM -q TQ -brv
....
GroupId[0x67726F7570696400000000000000000000000000000000000000] MsgSeqNumbe r[3] Offset[100] MsgFlags[10] OriginalLength[1000]
....

* MQMF_SEGMENT: 0x00000002
* MQMF_MSG_IN_GROUP: 0x00000008

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

GroupId (-gi) (MQBYTE24)("")

Group Id of a message.

MsgSeqNumber (-ms) (MQLONG)(1)

Message Sequence Number of logical message in group.

Offset (-of) (MQLONG)(0)

The Relative position of a physical message data from the start point of a logical message.

OriginalLength (-ol) (MQLONG)(MQOL_UNDEFINED)

The length of a message segment to which a report message relates.

5.7 MQRFH2 Fields

This program creates MQRFH2 header after MQMD if MQRFH2 field is specified.

Ex. 5.7.1 Example of specifying MQRFH2 fields.

Encoding：273
CodedCharSetId：1208
Format：MQFMT_STRING
flags：100
NameValueCCSID：1208
NameValueData："test1,test22,test333"

```
--------------------------------------
$ mqpgf -qm TESTQM -q TQ -m "test" -re 273 -rc 1208 -rf MQFMT_STRING -fg
  100 -nc 1208 -nd "test1,test22,test333" MQFMT_RF_HEADER_2
[16/12/23 00:22:25] 1: message length: 4 put message：test

$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[MQHRF2  ] Priority[0] Persistence[0] MsgId[0
x414D51206F6B61716D3830612020202057D9E32620003F06] CorrelId[0x000000000
00000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                          ] ReplyToQMgr[TESTQM
             ] UserIdentifier[mq80          ] AccountingToken[0x0332343
2000000000000000000000000000000000000000000000000000006] ApplIdentityDat
a[                                  ] PutApplType[6] PutApplName[mqpgf
          ] PutDate[20160915] PutTime[08041209] ApplOriginData[    ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[RFH ] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[1208]
Format[MQSTR   ] Flags[100] NameValueCCSID[1208]
NameValueLength[8] NameValueData[test1   ]
NameValueLength[8] NameValueData[test22  ]
NameValueLength[8] NameValueData[test333 ]
data length: 4
00000000:  7465 7374                                  'test          '
--------------------------------------
```

Hereinafter, the notation of the title is a field name (option) (data type) (default

value).

Encoding（-re）（MQLONG)(MQENC_NATIVE)

Numeric encoding of the data following NameValueData field.

Constants in the table below can also be specified.

Table 5.7.1 MQENC_ *

Constant Name	Value	location	Operation and etc.
MQENC_NATIVE	0x00000111 0x00000222(Windows, Linux x86)	MQRFH 2. Encodin g	OR (default)
MQENC_INTEGER_UNDEFI NED	0x00000000	Same as above	Same as above
MQENC_INTEGER_NORMAL	0x00000001		
MQENC_INTEGER_REVERS ED	0x00000002		
MQENC_DECIMAL_UNDEFI NED	0x00000000		
MQENC_DECIMAL_NORMAL	0x00000010		
MQENC_DECIMAL_REVERS ED	0x00000020		
MQENC_FLOAT_UNDEFINE D	0x00000000		
MQENC_FLOAT_IEEE_NOR MAL	0x00000100		
MQENC_FLOAT_IEEE_REVE RSED	0x00000200		
MQENC_FLOAT_S390	0x00000300		
MQENC_FLOAT_TNS	0x00000400		

Table 5.7.1 MQENC_ *

Constant Name	Value	location	Operation and etc.
MQENC_NORMAL	MQENC_FLOAT_IEEE_NORMAL | MQENC_DECIMAL_NORMAL | MQENC_INTEGER_NORMAL		
MQENC_REVERSED	MQENC_FLOAT_IEEE_REVERSED | MQENC_DECIMAL_REVERSED | MQENC_INTEGER_REVERSED		
MQENC_S390	MQENC_FLOAT_S390 | MQENC_DECIMAL_NORMAL | MQENC_INTEGER_NORMAL		
MQENC_TNS	MQENC_FLOAT_TNS | MQENC_DECIMAL_NORMAL | MQENC_INTEGER_NORMAL		
MQENC_AS_PUBLISHED	(-1)		

Ex. 5.7.2 Example of specifying Encoding field.
```
--------------------------------------
$ mqpgf -qm TESTQM -q TQ -m test -re MQENC_FLOAT_IEEE_REVERSED:MQENC_DECIMAL_REVERSED:MQENC_INTEGER_REVERSED MQFMT_RF_HEADER_2
[17/01/06 08:53:00] 1: message length: 4 put message : test
$ mqpgf -qm TESTQM -q TQ -m test -re MQENC_REVERSED MQFMT_RF_HEADER_2
[17/01/06 08:53:06] 1: message length: 4 put message : test
$ mqpgf -qm TESTQM -q TQ -m test -re 0x222 MQFMT_RF_HEADER_2
[17/01/06 08:53:11] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -brv -r
message number: 1
....
```

*StrucId[RFH] Version[2] StrucLength[36] **Encoding[546]** CodedCharSetId[943] Fo
rmat[   ] Flags[0] NameValueCCSID[1208]
....

message number: 2

....
*StrucId[RFH] Version[2] StrucLength[36] **Encoding[546]** CodedCharSetId[943] Fo
rmat[   ] Flags[0] NameValueCCSID[1208]
....

message number: 3

....
*StrucId[RFH] Version[2] StrucLength[36] **Encoding[546]** CodedCharSetId[943] Fo
rmat[   ] Flags[0] NameValueCCSID[1208]
....

no message available : TQ CompCd=02 ReasonCd=2033

CodedCharSetId (-rc) (MQLONG)(MQCCSI_INHERIT)

Coded Character Set ID of the data following NameValueData field.

Constants in the table below can also be specified.

Table 5.7.2 MQCCSI_*			
Constant Name	**Value**	**location**	**Operation and etc.**
MQCCSI_UNDEFINED	0	MQRFH2.CodedCharSetId	Overwrite
MQCCSI_DEFAULT	0	Same as above	Same as avbove
MQCCSI_Q_MGR	0		(default)
MQCCSI_INHERIT	(-2)		
MQCCSI_EMBEDDED	(-1)		
MQCCSI_APPL	(-3)		
MQCCSI_AS_PUBLISHED	(-4)		

Format（-rf）(MQCHAR8)(MQFMT_NONE_ARRAY)

Format name of the data following NameValueData.

Constants in the table below can also be specified

Table 5.7.3 MQFMT_*

Constant Name	Value	location	Operation and etc.
MQFMT_NONE	" "	MQRFH2.Format	Overwrite (default)
MQFMT_ADMIN	"MQADMIN "	Same as above	Same as above
MQFMT_CHANNEL_COMPLETED	"MQCHCOM "		
MQFMT_CICS	"MQCICS "		
MQFMT_COMMAND_1	"MQCMD1 "		
MQFMT_COMMAND_2	"MQCMD2 "		
MQFMT_DEAD_LETTER_HEADER	"MQDEAD "		
MQFMT_DIST_HEADER	"MQHDIST "		
MQFMT_EMBEDDED_PCF	"MQHEPCF "		
MQFMT_EVENT	"MQEVENT "		
MQFMT_IMS	"MQIMS "		
MQFMT_IMS_VAR_STRING	"MQIMSVS "		
MQFMT_MD_EXTENSION	"MQHMDE "		
MQFMT_PCF	"MQPCF "		
MQFMT_REF_MSG_HEADER	"MQHREF "		
MQFMT_RF_HEADER	"MQHRF "		
MQFMT_RF_HEADER_1	"MQHRF "		
MQFMT_RF_HEADER_2	"MQHRF2 "		
MQFMT_STRING	"MQSTR "		
MQFMT_TRIGGER	"MQTRIG "		

Table 5.7.3 MQFMT_*			
Constant Name	**Value**	**location**	**Operation and etc.**
MQFMT_WORK_INFO_HEADER	"MQHWIH　"		
MQFMT_XMIT_Q_HEADER	"MQXMIT　"		

Flags (-fg) (MQLONG)(MQRFH_NONE)

The Default value of Flags is MQRFH_NONE.

NameValueCCSID (-nc) (MQLONG)(1208)

Coded Character Set ID of the data in NameValueData field.

NameValueData (-nd) (MQCHARn)(-)

The Variable length field containing folder with name / value pairs or message properties.
　e.g. -nd "data1,data2,data3"

5.8 MQCSP Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

CSPUserId (-cu) (-)(-)

This parameter is a user ID that the authorization service can use to identify and authenticate the user. If you use the MQCSP structure to pass credentials, the maximum length of a user ID is 1024

When the user and password to be set in MQCSP are specifid by "- cu" and "- cp" , mqpgf sets automatically the pointer to each character string to MQCSP.CSPUserIdPtr, MQCSP.CSPPasswordPtr and sets the length of the string to MQCSP.CSPUserIdLength, MQCSP.CSPPasswordLength. In order to perform a uthentication, it is necessary to specify MQCSP_AUTH_USER_ID_AND_PWD f or MQCSP.AuthenticationType. When MQCSP_AUTH_ * is specified as an argument, mqpgf sets it to MQCSP.AuthenticationType and sets a pointer to MQCNO.SecurityParmsPtr automatically. Note that this MQCNO.SecurityParmsPtr specification requires MQCNO_VERSION_5 or more. The default for MQCNO is MQCNO_VERSION_1. If MQCNO version is less than MQCNO_VERSION_ 5, authentication will not be performed.

CSPPassword (-cp) (-)(-)

This parameter is password that the authorization service can use to identify and authenticate the user. (See "MQCSP field - CSPUserId (- cu)" above)

Ex. 5.8.1 Connect with MQCSP UserId, Password.
```
---------------------------------------
$ mqpgfc -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnnn) -ch Sa
mpleQM.MQICHL -cu pulsar -cp correctPW MQCSP_AUTH_USER_ID_AND_PWD
MQCNO_VERSION_5
 [2018/08/23 14:50:46.286] 1: message length: 4 put message: test

 $ mqpgfc -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnnn) -ch S
ampleQM.MQICHL -cu pulsar -cp wrongPW MQCSP_AUTH_USER_ID_AND_PWD
 MQCNO_VERSION_5
 MQCONNX fail : SampleQM CompCd=02 ReasonCd=2035
 !!! Queue Manager Connect Fail SampleQM !!!

 $ mqrc 2035

2035 0x000007f3 MQRC_NOT_AUTHORIZED
```

$ mqpgfc -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnnn) -ch S
ampleQM.MQICHL -cu pulsar -cp wrongPW MQCSP_AUTH_USER_ID_AND_PWD
 [2018/08/23 14:51:02.293] 1: message length: 4 put message: test

* In this example, although an incorrect password is specified, since MQCNO_VE
RSION_ 5 is not specified, the default MQCNO_VERSION_ 1 is used and authen
tication of the user and password specified in MQCSP was not performed, and a
s a result MQPUT is successful.

5.9 MQOD Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

ObjectQMgrName (-om) (MQCHAR48)("")

The Name of the queue manager in which the object is defined.

Ex. 5.9.1 Example of specifying an object queue manager.

```
-------------------------------------
$ mqpcf cque -qm QMB -q CQ1 TYPE CLUSQMGR
1: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(QMA)
2: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(QMB)
3: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(QMC)

$ mqpcf que -qm QMB -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(0)

$ mqpcf que -qm QMC -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(0)

$ mqpgf -qm QMB -q CQ1 -m "test" -n 3
[16/12/23 00:45:19] 1: message length: 4 put message : test
[16/12/23 00:45:19] 2: message length: 4 put message : test
[16/12/23 00:45:19] 3: message length: 4 put message : test
$ mqpcf que -qm QMB -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(3)
$ mqpcf que -qm QMC -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(0)

$ mqpgf -qm QMB -q CQ1 -m "test" -om QMC -n 3
[16/12/23 00:46:40] 1: message length: 4 put message : test
[16/12/23 00:46:40] 2: message length: 4 put message : test
[16/12/23 00:46:40] 3: message length: 4 put message : test
$ mqpcf que -qm QMB -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(3)
$ mqpcf que -qm QMC -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(3)
-------------------------------------
```

AlternateUserId (-au) (MQCHAR12)("")

Identifier of an alternate user.

Ex. 5.9.2 Put or get a message with AlternateUserID parameter.

$ setmqaut -m TESTQM -t qmgr -p guest +altusr
The setmqaut command completed successfully.
$ dspmqaut -m TESTQM -t qmgr -p guest
Entity guest has the following authorizations for object TESTQM:
        inq
        set
        connect
        altusr
        setid
        setall
$ dspmqaut -m TESTQM -t q -n TQ -p guest
Entity guest has the following authorizations for object TESTQM:
$ id
uid=243(guest) gid=1(staff)

$ mqpgf -qm TESTQM -q TQ -m "test message" -au mqm
MQOPEN fail : TESTQM TQ CompCd=02 ReasonCd=2035

$ mqpgf -qm TESTQM -q TQ -m "test " **-au mqm MQOO_ALTERNATE_USER_A
UTHORITY**
[16/12/23 01:01:17] 1: message length: 4 put message : test
$ mqpgf -qm TESTQM -q TQ **-au mqm MQOO_ALTERNATE_USER_AUTHORIT
Y**
[16/12/23 01:01:33] 1: message length: 4 get message : test

ObjectRec(MQOR) (-or) (-)(-)

The Record of queue objects.
e.g. -or <queue1[:qmgr1]>,<queue2[:qmgr2]>,<queue3[:qmgr3]>,...

DynamicQName (-dq) (MQCHAR48)("AMQ.*")

Specify the name of the dynamic queue. When using dynamic queue to send
message, you can specify model queue name with "-q". When using dynamic q
ueue to receive response you need specify model queue name with "-iq".

Ex. 5.9.3 Create a temporary dynamic queue and write a message on it
----
* Create a dynamic temporary queue named "DYNAMICQS1" using the model qu
eue (SYSTEM.DEFAULT.MODEL.QUEUE), write a message, and stop processing
before calling MQCLOSE().

$ mqpgf -qm SampleQM **-q SYSTEM.DEFAULT.MODEL.QUEUE** -m dyntest **-dq
"DYNAMICQS1" -s MQCLOSE**
[18/05/31 15:12:00] 1: message length: 7 put message: dyntest
MQCMIT success : CompCd=00 ReasonCd=00
stop before calling MQCLOSE().
Hit Any Key!!!

* In other terminals, confirm that the dynamic temporary queue is created and t
he message is on the queue and get the message.
$ mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE CURDEPTH
1: **QUEUE(DYNAMICQS1)** TYPE(QLOCAL) **CURDEPTH(1) DEFTYPE(TEMPDY
N)**

$ mqpgf -qm SampleQM -q DYNAMICQS1
[18/05/31 15:20:13] 1: message length: 7 get message : dyntest

* If you call MQCLOSE () by entering an arbitrary key at the terminal you use
d first, the dynamic temporary queue will be deleted each time.

$ mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE CURDEPTH
1: QUEUE(DYNAMICQS1) TYPE(QLOCAL) CURDEPTH(1) DEFTYPE(TEMPDY
N)
/home/okaqm7/work/cprog/mqpgf: mqpcf que -qm SampleQM -q DYNA* TYPE DE
FTYPE C>
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], **mqCommandRC=[2085]**
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], mqCommandRC=[3008]

$ mqrc 2085

    2085   0x00000825   MQRC_UNKNOWN_OBJECT_NAME

Ex. 5.9.4 Create a temporary dynamic queue and wait for a message to arrive
----

* mqpgf command waits for reception of messages from the queue (SampleQ) rep
eatedly. The received message is written to the queue set in MQMD.ReplyToQ wi
th specifying MQMT_REPLY for MQMD.MsgType on the secondary side. mqpgf w
rites a message to MQMD.ReplyToQ if "*" is specified for "-oq".

$ mqpgf -qm SampleQM -q SampleQ **-oq "*"** MQGMO_WAIT MQWI_UNLIMITED
 -dp -r **-ss MQMT_REPLY**

* In another terminal, create a dynamic temporary queue ("DYNAMIC *") for rec
eiving the response and write 3 messagese with the created queue name to MQ
MD.ReplyToQ and MQMT_REQUEST to MQMD.MsgType to the queue.
Then, processing stops before calling MQCLOSE().

$ mqpgf -qm SampleQM -q SampleQ -m "dynamic" -iq SYSTEM.DEFAULT.MOD
EL.QUEUE -rm SampleQM **MQMT_REQUEST** MQPMO_NO_SYNCPOINT -n 3 -i
1000 -ss MQGMO_WAIT MQWI_UNLIMITED MQGMO_NO_SYNCPOINT **-dq DY
NAMIC*** -s MQCLOSE
[18/05/31 15:51:55] 1: message length: 7 put message : dynamic
[18/05/31 15:51:55] 1: message length: 7 get message : dynamic
stop before calling MQCLOSE().
Hit Any Key!!!
[18/05/31 15:51:59] 1: message length: 7 put message : dynamic
[18/05/31 15:51:59] 1: message length: 7 get message : dynamic
stop before calling MQCLOSE().
Hit Any Key!!!
[18/05/31 15:52:01] 1: message length: 7 put message : dynamic
[18/05/31 15:52:01] 1: message length: 7 get message : dynamic
stop before calling MQCLOSE().
Hit Any Key!!!

* Every time you press any key, a dump of the received message and the respon
se message sent are displayed on the other terminal.message number: 1

*StrucId[MD] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x41
4D512053616D706C65514D202020205B0F79E22000360A] CorrelId[0x000000000000
0000000000000000000000000000000000000000] BackoutCount[0] **ReplyToQ[DYNAMIC5
B0F79E220003609   ]** ReplyToQMgr[SampleQM
                    ] UserIdentifier[mqm   ] AccountingToken[0x05
34343033310000000000000000000000000000000000000000000000000000006] ApplIdent
ityData[   ] PutApplType[13] PutApplName[mqpgf
            ] PutDate[20180531] PutTime[06515552] ApplOriginData[
 ]

GroupId[0x00000000000000000000000000000000000000000000000] MsgSeqNumbe

r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 7
00000000: 6479 6E61 6D69 63   'dynamic   '

[18/05/31 15:51:55] 1: message length: 7 put message : dynamic
MQCMIT success : CompCd=00 ReasonCd=00
message number: 2
*StrucId[MD] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x41
4D512053616D706C65514D202020205B0F79E22000360C] CorrelId[0x000000000000
00000000000000000000000000000000000000] BackoutCount[0] **ReplyToQ[DYNAMIC5
B0F79E22000360B**   ] ReplyToQMgr[SampleQM
                         ] UserIdentifier[mqm   ] AccountingToken[0x05
3434303331000000000000000000000000000000000000000000000006] ApplIdent
ityData[   ] PutApplType[13] PutApplName[mqpgf
            ] PutDate[20180531] PutTime[06515956] ApplOriginData[
 ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 7
00000000: 6479 6E61 6D69 63   'dynamic   '

[18/05/31 15:51:59] 2: message length: 7 put message : dynamic
MQCMIT success : CompCd=00 ReasonCd=00
message number: 3
*StrucId[MD] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x41
4D512053616D706C65514D202020205B0F79E22000360E] CorrelId[0x000000000000
00000000000000000000000000000000000000] BackoutCount[0] **ReplyToQ[DYNAMIC5
B0F79E22000360D**   ] ReplyToQMgr[SampleQM
                         ] UserIdentifier[mqm   ] AccountingToken[0x0
53434303331000000000000000000000000000000000000000000000006] ApplIden
tityData[   ] PutApplType[13] PutApplName[mqpgf
            ] PutDate[20180531] PutTime[06520139] ApplOriginData[
 ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumbe
r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 7
00000000: 6479 6E61 6D69 63   'dynamic   '

[18/05/31 15:52:01] 3: message length: 7 put message : dynamic

* Each time a response is received and the dynamic temporary queue is closed, the queue is deleted. All three created dynamic temporary queues have been deleted.
$ mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE CURDEPTH
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommandCC=[2], mqCommandRC=[2085]
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommandCC=[2], mqCommandRC=[3008]

$ mqrc 2085

    2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME

5.10 MQPMO Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

PutMsgRec(MQPMR) (-mr) (-)(-)

Record of a message to put.
e.g. -mr <msgId>:<correlId>:<groupId>:<feedback>:<accountingtoken>,...

5.11 MQGMO Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

WaitInterval (-wi) (MQLONG)(0)

Approximate time in milliseconds to wait for a message to arrive.

Ex. 5.11.1 Wait for the specified time until a message arrives when getting the message.
```
-------------------------------------
$ date; mqpgf -qm TESTQM -q TQ -wi 5000 MQGMO_WAIT
Fri Jan  6 22:11:42 JST 2017
[17/01/06 22:11:44] 1: message length: 4 get message : test
-------------------------------------
```


MsgToken (-mt) (MQBYTE16)(MQMTOK_NONE_ARRAY)

Specify "Message Token (MsgToken)". Only hexadecimal notation can be used. MQGMO_VERSION_3 or higher must be specified at the same time. Usually used with MQMO_MATCH_MSG_TOKEN.

5.12 MQIMPO Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

RequestedEncoding (-pe) (MQLONG)(MQENC_NATIVE)

Encoding to convert property values to be queried when specifying MQIMPO_CONVERT_VALUE or MQIMPO_CONVERT_TYPE.

RequestedCCSID (-pc) (MQLONG)(MQCCSI_APPL)

CCSID of property values to be queried when specifying MQIMPO_CONVERT_VALUE or MQIMPO_CONVERT_TYPE.

5.13 MQCB Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

Operation (-op) (MQLONG)(-)

The Operation of MQCB. Possible values are MQOP_ *. If you want to specify more than one option at the same time, specify multiple times like "-op MQOP_REGISTER -op MQOP_SUSPEND". MQCB is only called if you specified the CallBack function in the MQCBD field using the "-cf" option. (See "Field of MQCBD - CallbackFunction (- cf) (MQPTR) (-)")

Table 5.13.1 MQOP_*

Constant Name	Value	location	Operation and etc.
MQOP_START	0x00000001	Operation which is an argument of MQCB()	OR
MQOP_START_WAIT	0x00000002	Same as above	Same as above
MQOP_STOP	0x00000004		
MQOP_REGISTER	0x00000100		
MQOP_DEREGISTER	0x00000200		
MQOP_SUSPEND	0x00010000		
MQOP_RESUME	0x00020000		

5.14 MQCBD Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

CallbackFunction (-cf) (MQPTR)(-)

The callback function name. The callback function currently available is "EventHandler" only. The EventHandler is useful when testing Automatic client reconnection, and displays what was set in Context.Reason when the event handler was called back and the value of Context.ReconnectDelay.

5.15 MQSCO Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

KeyRepository (-kr) (MQCHAR256)(-)

Specifies the location of the key repository when using SSL/TLS in client mode. For GSkit, specify <directory>/<the part excluding the file extension of the key DB>. For Openssl(MQ for HPE NonStop, etc.), specify the directory where the certificate file is located.

5.16 MQAIR Fields

Hereinafter, the notation of the title is a field name (option) (data type) (default value).

OCSPResponderURL (-ru) (MQCHAR256)(-)

When using SSL/TLS in client mode and connecting to the OCSP responder to validate the certificate, specify the URL to connect to the OSCP responder. The connection string must start with "http://".

6. Available Constants

This program allows many constants as arguments.

6.1 MQMD Parameters

MQMD_*

MQMD_* are set into MQMD.Version.

Constants of the table below can be specified.

Table 6.1.1 MQMD_*				
Constant Name	**Value**	**location**	**Operation and etc.**
MQMD_VERSION_1	1	MQMD.Version	Overwrite (default)
MQMD_VERSION_2	2	Same as above	Same as above
MQMD_CURRENT_VERSION	2		

Ex. 6.1.1 Get a message with MQMD_VERSION_*.

$ mqpgf -qm TESTQM -q TQ -m test **-gi GID -ms 3 -of 100 -ol 1000 MQMD_VE RSION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP**
[16/12/28 19:57:08] 1: message length: 4 put message : test

$ mqpgf -qm TESTQM -q TQ -brv **MQMD_VERSION_2**
message number: 1
*StrucId[MD　] **Version[2]** Report[0] MsgType[4] Expiry[-1] Feedback[0] ....
....
GroupId[0x474944000000000000000000000000000000000000000000] MsgSeqNumbe r[3] Offset[100] MsgFlags[10] OriginalLength[1000]
....

$ mqpgf -qm TESTQM -q TQ -brv **MQMD_VERSION_1**
message number: 1
*StrucId[MD　] **Version[1]** Report[0] MsgType[4] Expiry[-1] Feedback[0] ....
....

GroupId[0x000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[MDE] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[943] Format[   ] Flags[0] GroupId[0x474944000000000000000000000000000000000000000000] MsgSeqNumber[3] Offset[100] MsgFlags[10] OriginalLength[1000]

....

MQRO_*

MQRO_* are set into MQMD.Report.

Constants of the table below can be specified.

Table 6.1.2 MQRO_*

Constant Name	Value	location	Operation and etc.
MQRO_EXCEPTION	0x01000000	MQMD.Report	OR
MQRO_EXCEPTION_WITH_DATA	0x03000000	Same as above	Same as above
MQRO_EXCEPTION_WITH_FULL_DATA	0x07000000		
MQRO_EXPIRATION	0x00200000		
MQRO_EXPIRATION_WITH_DATA	0x00600000		
MQRO_EXPIRATION_WITH_FULL_DATA	0x00E00000		
MQRO_COA	0x00000100		
MQRO_COA_WITH_DATA	0x00000300		
MQRO_COA_WITH_FULL_DATA	0x00000700		
MQRO_COD	0x00000800		
MQRO_COD_WITH_DATA	0x00001800		
MQRO_COD_WITH_FULL_DATA	0x00003800		
MQRO_PAN	0x00000001		
MQRO_NAN	0x00000002		

Table 6.1.2 MQRO_*

Constant Name	Value	location	Operation and etc.
MQRO_ACTIVITY	0x00000004		
MQRO_NEW_MSG_ID	0x00000000		
MQRO_PASS_MSG_ID	0x00000080		
MQRO_COPY_MSG_ID_TO_CORREL_ID	0x00000000		
MQRO_PASS_CORREL_ID	0x00000040		
MQRO_DEAD_LETTER_Q	0x00000000		
MQRO_DISCARD_MSG	0x08000000		
MQRO_PASS_DISCARD_AND_EXPIRY	0x00004000		
MQRO_NONE	0x00000000		(default)

Ex. 6.1.2 Example of specifying report options

$ mqpgf -qm TESTQ -q TQ -f input.txt **MQRO_EXCEPTION MQRO_COA_WITH_FULL_DATA**
[16/12/27 20:27:47] 1: put from input.txt
MQPUT fail : TQ CompCd=02 ReasonCd=2027
$
$ mqrc 2027

        2027   0x000007eb   MQRC_MISSING_REPLY_TO_Q

$ mqpgf -qm TESTQ -q TQ -m "test" **-rq RQ MQRO_EXCEPTION MQRO_COA_WITH_FULL_DATA**
[16/12/27 20:33:35] 1: message length: 4 put message : test
$ mqpgf -qm TESTQ -q TQ -brv -hex
message number: 1
*StrucId[MD] Version[2] **Report[0x01000700]** MsgType[8] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[943] Format[   ] Priority[0] Persistence[0] MsgId[0x414D51206F6B61716D3830612020202058624CD720002287] CorrelId[0x0000000000000000000000000000000000000000000000000] BackoutCount[0] **ReplyToQ[RQ   ]** ....
....

* MQRO_EXCEPTION : 0x01000000
* MQRO_COA_WITH_FULL_DATA : 0x00000700

MQMT_*

MQMT_* are set into MQMD.MsgType.

Constants of the table below can be specified.

Table 6.1.3 MQMT_*			
Constant Name	**Value**	**location**	**Operation and etc.**
MQMT_SYSTEM_FIRST	1	MQMD.MsgType	Overwrite
MQMT_REQUEST	1	Same as above	Same as above
MQMT_REPLY	2		
MQMT_DATAGRAM	8		(default)
MQMT_REPORT	4		
MQMT_MQE_FIELDS_FROM_MQE	112		
MQMT_MQE_FIELDS	113		
MQMT_SYSTEM_LAST	65535		
MQMT_APPL_FIRST	65536		
MQMT_APPL_LAST	999999999		

Ex. 6.1.3 Example of specifying a message type

```
$ mqpgf -qm TESTQM -q TQ -m test MQMT_REQUEST
[16/12/27 21:17:22] 1: message length: 4 put message : test
MQPUT fail : TQ CompCd=02 ReasonCd=2027
$
$ mqrc 2027

    2027  0x000007eb  MQRC_MISSING_REPLY_TO_Q

$ mqpgf -qm TESTQM -q TQ -m test MQMT_REQUEST -rq RQ
[16/12/27 21:17:38] 1: message length: 4 put message : test
$
```

```
 mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[          ] Priority[0] Persistence[0] MsgId[0x41
4D51206F6B61716D3830612020202058624CD720002503] CorrelId[0x0000000000000
0000000000000000000000000000000000] BackoutCount[0] ReplyToQ[RQ
                                    ] ....
------------------------------------
```

MQEI_*

MQEI_* are set into MQMD.Expiry.

Constants of the table below can be specified.

Table 6.1.4 MQEI_*

Constant Name	Value	location	Operation and etc.
MQEI_UNLIMITED	(-1)	MQMD.Expiry	Overwrite (default)

MQFB_*

MQFB_* are set into MQMD.Feedback.

Constants of the table below can be specified.

Table 6.1.5 MQFB_*

Constant Name	Value	location	Operation and etc.
MQFB_NONE	0	MQMD.Feedback	Overwrite (default)
MQFB_SYSTEM_FIRST	1	Same as above	Same as above
MQFB_QUIT	256		
MQFB_EXPIRATION	258		

Table 6.1.5 MQFB_*

Constant Name	Value	location	Operation and etc.
MQFB_COA	259		
MQFB_COD	260		
MQFB_CHANNEL_COMPLETED	262		
MQFB_CHANNEL_FAIL_RETRY	263		
MQFB_CHANNEL_FAIL	264		
MQFB_APPL_CANNOT_BE_STARTED	265		
MQFB_TM_ERROR	266		
MQFB_APPL_TYPE_ERROR	267		
MQFB_STOPPED_BY_MSG_EXIT	268		
MQFB_ACTIVITY	269		
MQFB_XMIT_Q_MSG_ERROR	271		
MQFB_PAN	275		
MQFB_NAN	276		
MQFB_STOPPED_BY_CHAD_EXIT	277		
MQFB_STOPPED_BY_PUBSUB_EXIT	279		
MQFB_NOT_A_REPOSITORY_MSG	280		
MQFB_BIND_OPEN_CLUSRCVR_DEL	281		
MQFB_MAX_ACTIVITIES	282		
MQFB_NOT_FORWARDED	283		
MQFB_NOT_DELIVERED	284		
MQFB_UNSUPPORTED_FORWARDING	285		
MQFB_UNSUPPORTED_DELIVERY	286		
MQFB_DATA_LENGTH_ZERO	291		
MQFB_DATA_LENGTH_NEGATIVE	292		
MQFB_DATA_LENGTH_TOO_BIG	293		
MQFB_BUFFER_OVERFLOW	294		

Table 6.1.5 MQFB_*

Constant Name	Value	location	Operation and etc.
MQFB_LENGTH_OFF_BY_ONE	295		
MQFB_IIH_ERROR	296		
MQFB_NOT_AUTHORIZED_FOR_IMS	298		
MQFB_IMS_ERROR	300		
MQFB_IMS_FIRST	301		
MQFB_IMS_LAST	399		
MQFB_CICS_INTERNAL_ERROR	401		
MQFB_CICS_NOT_AUTHORIZED	402		
MQFB_CICS_BRIDGE_FAILURE	403		
MQFB_CICS_CORREL_ID_ERROR	404		
MQFB_CICS_CCSID_ERROR	405		
MQFB_CICS_ENCODING_ERROR	406		
MQFB_CICS_CIH_ERROR	407		
MQFB_CICS_UOW_ERROR	408		
MQFB_CICS_COMMAREA_ERROR	409		
MQFB_CICS_APPL_NOT_STARTED	410		
MQFB_CICS_APPL_ABENDED	411		
MQFB_CICS_DLQ_ERROR	412		
MQFB_CICS_UOW_BACKED_OUT	413		
MQFB_PUBLICATIONS_ON_REQUEST	501		
MQFB_SUBSCRIBER_IS_PUBLISHER	502		
MQFB_MSG_SCOPE_MISMATCH	503		
MQFB_SELECTOR_MISMATCH	504		
MQFB_NOT_A_GROUPUR_MSG	505		
MQFB_IMS_NACK_1A_REASON_FIRST	600		
MQFB_IMS_NACK_1A_REASON_LAST	855		

Table 6.1.5 MQFB_*

Constant Name	Value	location	Operation and etc.
MQFB_SYSTEM_LAST	65535		
MQFB_APPL_FIRST	65536		
MQFB_APPL_LAST	999999999		

Ex. 6.1.4 Example of specifying MQFB_*.

```
--------------------------------------
$ mqpgf -qm TESTQM -q TQ -m test MQFB_COA
[16/12/27 21:23:35] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[259] ....
....
--------------------------------------
```

MQENC_*

MQENC_* are set into MQMD.Encoding.

MQCCSI_*

MQCCSI_* are set into MQMD.CodedCharSetId.

MQFMT_*

MQFMT_* are set into MQMD.Format.

Constants of the table below can be specified.

Table 6.1.6 MQFMT_*

Constant Name	Value	location	Operation and etc.
MQFMT_NONE	" "	MQMD.Format	Overwrite (default)
MQFMT_ADMIN	"MQADMIN "	Same as above	Same as above
MQFMT_CHANNEL_COMPLETED	"MQCHCOM "		
MQFMT_CICS	"MQCICS "		
MQFMT_COMMAND_1	"MQCMD1 "		
MQFMT_COMMAND_2	"MQCMD2 "		
MQFMT_DEAD_LETTER_HEADER	"MQDEAD "		
MQFMT_DIST_HEADER	"MQHDIST "		
MQFMT_EMBEDDED_PCF	"MQHEPCF "		
MQFMT_EVENT	"MQEVENT "		
MQFMT_IMS	"MQIMS "		
MQFMT_IMS_VAR_STRING	"MQIMSVS "		
MQFMT_MD_EXTENSION	"MQHMDE "		
MQFMT_PCF	"MQPCF "		
MQFMT_REF_MSG_HEADER	"MQHREF "		
MQFMT_RF_HEADER	"MQHRF "		
MQFMT_RF_HEADER_1	"MQHRF "		
MQFMT_RF_HEADER_2	"MQHRF2 "		
MQFMT_STRING	"MQSTR "		
MQFMT_TRIGGER	"MQTRIG "		
MQFMT_WORK_INFO_HEADER	"MQHWIH "		
MQFMT_XMIT_Q_HEADER	"MQXMIT "		

Ex. 6.1.5 Example of specifying as MQMD.Format and MQRFH2.Format.

$ mqpgf -qm TESTQM -q TQ -m test **MQFMT_RF_HEADER_2 -rf MQFMT_STRI**

NG

[16/12/27 21:44:59] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] **Format[MQHRF2]** ....
....
StrucId[RFH] Version[2] StrucLength[36] Encoding[273] CodedCharSetId[943] **Fo
rmat[MQSTR   ]** Flags[0] NameValueCCSID[1208]
....

MQPRI_*

MQPRI_* are set into MQMD.Priority.

Constants of the table below can be specified.

Table 6.1.7 MQPRI_*			
Constant Name	**Value**	**location**	**Operation and etc.**
MQPRI_PRIORITY_AS_Q_DEF	(-1)	MQMD.Priority	Overwrite (default)
MQPRI_PRIORITY_AS_PARENT	(-2)	Same as above	Same as above
MQPRI_PRIORITY_AS_PUBLISHED	(-3)		
MQPRI_PRIORITY_AS_TOPIC_DEF	(-1)		

MQPER_*

MQPER_* are set into MQMD.Persistence.

Constants of the table below can be specified.

Table 6.1.8 MQPER_*

Constant Name	Value	location	Operation and etc.
MQPER_PERSISTENCE_AS_PARENT	(-1)	MQMD.Persistence	Overwrite
MQPER_NOT_PERSISTENT	0	Same as above	Same as above
MQPER_PERSISTENT	1		
MQPER_PERSISTENCE_AS_Q_DEF	2		(default)
MQPER_PERSISTENCE_AS_TOPIC_DEF	2		

Ex. 6.1.6 Example of specifying persistent attributes.

```
-------------------------------------
$ mqpgf -qm TESTQM -q TQ -inq MQIA_DEF_PERSISTENCE
[16/12/28 19:07:54] 1: DEFPSIST(NO)
$
$ mqpgf -qm TESTQM -q TQ -m test MQPER_PERSISTENT
[16/12/28 19:08:58] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding
[273] CodedCharSetId[943] Format[          ] Priority[0] Persistence[1] ....
....
-------------------------------------
```

MQMI_*

MQMI_* are set into MQMD.MsgId.

Constants of the table below can be specified.

Table 6.1.9 MQMI_*

Constant Name	Value	location	Operation and etc.
MQMI_NONE	0x00(24Byte)	MQMD.MsgId	Overwrite (default)

MQCI_*

MQCI_* are set into MQMD.CorrelId.

Constants of the table below can be specified.

Table 6.1.10 MQCI_*

Constant Name	Value	location	Operation and etc.
MQCI_NONE	0x00(24Byte)	MQMD.CorrelId	Overwrite (default)
MQCI_NEW_SESSION	"AMQ!NEW_SESSION_CORRELID"		

MQACT_*

MQACT_* are set into MQMD.AccountingToken.

Constants of the table below can be specified.

Table 6.1.11 MQACT_*

Constant Name	Value	location	Operation and etc.
MQACT_NONE	0x00(32Byte)	MQMD.AccountingToken	Overwrite (default)

MQACTT_*

MQACTT_* are set into the last byte of MQMD.AccountingToken.

Constants of the table below can be specified.

Table 6.1.12 MQACTT_*

Constant Name	Value	location	Operation and etc.
MQACTT_UNKNOWN	0x00	MQMD.AccountingToken(last 32nd byte)	Overwrite (default)
MQACTT_CICS_LUOW_ID	0x01		
MQACTT_OS2_DEFAULT	0x04		
MQACTT_DOS_DEFAULT	0x05		
MQACTT_UNIX_NUMERIC_ID	0x06		
MQACTT_OS400_ACCOUNT_TOKEN	0x08		
MQACTT_WINDOWS_DEFAULT	0x09		
MQACTT_NT_SECURITY_ID	0x0B		
MQACTT_USER	0x19		

Ex. 6.1.7 Example of specifying MQACTT_* and '-at'.

$ mqpgf -qm TESTQM -q TQ -m test **MQACTT_CICS_LUOW_ID -at 0x03323432 0000000000000000000000000000000000000000000000000000** **MQPMO_SET_ALL _CONTEXT MQOO_SET_ALL_CONTEXT**

[17/01/10 14:21:38] 1: message length: 4 put message : test
$ mqpgf -qm TESTQM -q TQ -brv -r
message number: 1
*StrucId[MD   ] .... **AccountingToken[0x03323432000000000000000000000000000000 0000000000000000000000001]** ....
....

MQAT_*

MQAT_* are set into MQMD.PutApplType.

Constants of the table below can be specified.

Table 6.1.13 MQAT_*

Constant Name	Value	location	Operation and etc.
MQAT_UNKNOWN	(-1)	MQMD.PutApplType	Overwrite Same as above
MQAT_NO_CONTEXT	0	Same as above	(default)
MQAT_CICS	1		
MQAT_MVS	2		
MQAT_OS390	2		
MQAT_ZOS	2		
MQAT_IMS	3		
MQAT_OS2	4		
MQAT_DOS	5		
MQAT_AIX	6		
MQAT_UNIX	6		
MQAT_QMGR	7		
MQAT_OS400	8		
MQAT_WINDOWS	9		
MQAT_CICS_VSE	10		
MQAT_WINDOWS_NT	11		
MQAT_VMS	12		
MQAT_GUARDIAN	13		
MQAT_NSK	13		
MQAT_VOS	14		
MQAT_OPEN_TP1	15		
MQAT_VM	18		
MQAT_IMS_BRIDGE	19		
MQAT_XCF	20		
MQAT_CICS_BRIDGE	21		

Table 6.1.13 MQAT_*

Constant Name	Value	location	Operation and etc.
MQAT_NOTES_AGENT	22		
MQAT_TPF	23		
MQAT_USER	25		
MQAT_BROKER	26		
MQAT_QMGR_PUBLISH	26		
MQAT_JAVA	28		
MQAT_DQM	29		
MQAT_CHANNEL_INITIATOR	30		
MQAT_WLM	31		
MQAT_BATCH	32		
MQAT_RRS_BATCH	33		
MQAT_SIB	34		
MQAT_SYSTEM_EXTENSION	35		
MQAT_MCAST_PUBLISH	36		
MQAT_DEFAULT	6		
MQAT_USER_FIRST	65536		
MQAT_USER_LAST	999999999		

Ex. 6.1.8 Example of specifying an application type.

$ mqpgf -qm TESTQM -q TQ -m test **MQAT_CICS MQPMO_SET_ALL_CONTEXT MQOO_SET_ALL_CONTEXT**
[16/12/28 19:30:53] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding [273] .... **PutApplType[1]** ....
....

MQMF_*

MQMF_* are set into MQMD.MsgFlags.

Constants of the table below can be specified.

Table 6.1.14 MQMF_*

Constant Name	Value	location	Operation and etc.
MQMF_SEGMENTATION_INHIBITED	0x00000000	MQMD.MsgFlags	OR
MQMF_SEGMENTATION_ALLOWED	0x00000001	Same as above	Same as above
MQMF_MSG_IN_GROUP	0x00000008		
MQMF_LAST_MSG_IN_GROUP	0x00000010		
MQMF_SEGMENT	0x00000002		
MQMF_LAST_SEGMENT	0x00000004		
MQMF_NONE	0x00000000		(default)

Ex. 6.1.9 Example of specifying message flags.
```
-------------------------------------
$ mqpgf -qm TESTQM -q TQ -m test MQMF_MSG_IN_GROUP MQMF_LAST_M
SG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMD_VERSION_2
[16/12/28 20:19:23] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -brv -hex
message number: 1
*StrucId[MD  ] Version[2] Report[0x00000000] MsgType[8] Expiry[-1] ....
....
GroupId[0x414D51206F6B61716D3830612020202058638EF520002703] MsgSeqNum
ber[1] Offset[0] MsgFlags[0x0000001E] OriginalLength[4]
....

The following bits (flags) are turned on in MsgFlags.
* MQMF_MSG_IN_GROUP: 0x00000008
* MQMF_LAST_MSG_IN_GROUP: 0x00000010
* MQMF_SEGMENT: 0x00000002
* MQMF_LAST_SEGMENT: 0x00000004
```

If MQMF_LAST_MSG_IN_GROUP is set, the queue manager turns on MQMF_M SG_IN_GROUP in the copy of MQMD that is sent with the message. So, in this case MQMF_MSG_IN_GROUP is optional.

If MQMF_LAST_SEGMENT is set, the queue manager turns on MQMF_SEGME NT in the copy of MQMD that is sent with the message. So, in this case MQMF _SEGMENT is optional.

6.2 MQRFH2 Parameters

MQRFH_*

MQRFH_* are set into MQRFH2.Flags.

Constants of the table below can be specified.

Table 6.2.1 MQRFH_*

Constant Name	Value	location	Operation and etc.
MQRFH_NONE	0x00000000	MQRFH2.Flags	Overwrite (default)
MQRFH_NO_FLAGS	0	Same as above	Same as above

6.3 MQCNO Parameters

MQCNO_* (exceot MQCNO*VERSION*, MQCNO_STRUC_ID)

MQCNO_* are set into MQCNO.Options.

Constants of the table below can be specified.

Table 6.3.1 MQCNO_* (except MQCNO*VERSION*, MQCNO_STRUC_ID)

Constant Name	Value	location	Operation and etc.
MQCNO_STANDARD_BINDING	0x00000000	MQCNO.Options	OR
MQCNO_FASTPATH_BINDING	0x00000001	Same as above	Same as above
MQCNO_SERIALIZE_CONN_TAG_Q_MGR	0x00000002		
MQCNO_SERIALIZE_CONN_TAG_QSG	0x00000004		
MQCNO_RESTRICT_CONN_TAG_Q_MGR	0x00000008		
MQCNO_RESTRICT_CONN_TAG_QSG	0x00000010		
MQCNO_HANDLE_SHARE_NONE	0x00000020		
MQCNO_HANDLE_SHARE_BLOCK	0x00000040		
MQCNO_HANDLE_SHARE_NO_BLOCK	0x00000080		
MQCNO_SHARED_BINDING	0x00000100		
MQCNO_ISOLATED_BINDING	0x00000200		
MQCNO_LOCAL_BINDING	0x00000400		
MQCNO_CLIENT_BINDING	0x00000800		
MQCNO_ACCOUNTING_MQI_ENABLED	0x00001000		
MQCNO_ACCOUNTING_MQI_DISABLED	0x00002000		
MQCNO_ACCOUNTING_Q_ENABLED	0x00004000		
MQCNO_ACCOUNTING_Q_DISABLED	0x00008000		
MQCNO_NO_CONV_SHARING	0x00010000		
MQCNO_ALL_CONVS_SHARE	0x00040000		

Table 6.3.1 MQCNO_* (except MQCNO*VERSION*, MQCNO_STRUC_ID)

Constant Name	Value	location	Operation and etc.
MQCNO_CD_FOR_OUTPUT_ONLY	0x00080000		
MQCNO_USE_CD_SELECTION	0x00100000		
MQCNO_RECONNECT_AS_DEF	0x00000000		
MQCNO_RECONNECT	0x01000000		
MQCNO_RECONNECT_DISABLED	0x02000000		
MQCNO_RECONNECT_Q_MGR	0x04000000		
MQCNO_ACTIVITY_TRACE_ENABLED	0x08000000		
MQCNO_ACTIVITY_TRACE_DISABLED	0x10000000		
MQCNO_NONE	0x00000000		(default)

Ex. 6.3.1 Example of connecting with SHARED_BINDING connect option.

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN
stop before calling MQOPEN().
Hit Any Key!!!

$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003301) TYPE(CONN)
CONNOPTS(MQCNO_SHARED_BINDING)

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN **MQCNO_STANDARD_BINDING**
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003501) TYPE(CONN)
CONNOPTS(MQCNO_SHARED_BINDING)

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN **MQCNO_SHARED_BINDING**
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003502) TYPE(CONN)
CONNOPTS(MQCNO_SHARED_BINDING)

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN **MQCNO_NONE**

stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003503) TYPE(CONN)
CONNOPTS(MQCNO_SHARED_BINDING)

Ex. 6.3.2 Example of connecting with FASTPATH_BINDING connect option.

$ id
uid=xxx(mq80) gid=xxx(mqm) groups=1(xxxx)
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN **MQCNO_FASTPATH_BINDING**
MQCONNX fail : TESTQM **CompCd=02 ReasonCd=2012**
!!! Queue Manager Connect Fail !!!
$ mqrc 2012

      2012   0x000007dc   MQRC_ENVIRONMENT_ERROR

$ su mqm
$id
uid=xxx(mqm) gid=xxx(mqm) groups=1(xxxx)

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN **MQCNO_FASTPATH_BINDING**
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
$ 1: CONN(414D51436F6B61716D3830612020202058746D1420003901) TYPE(CON
N) **CONNOPTS(MQCNO_FASTPATH_BINDING)**
$

Ex. 6.3.3 Example of connecting with ISOLATED_BINDING connect option.

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN **MQCNO_ISOLATED_BINDING**
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003C01) TYPE(CONN)
 CONNOPTS(MQCNO_ISOLATED_BINDING)

6.4 MQOPEN Options

MQOO_*

MQOO_* are set into the argument "Options" of the MQOPEN(). The options that are automatically set vary depending on the type of subsequent MQI call, MQGET () / MQPUT () / MQSET () / MQINQ (). (See the table below) Constants of the table below can be specified.

Table 6.4.1 MQOO_*

Constant Name	Value	location	Operation and etc.
MQOO_BIND_AS_Q_DEF	0x00000000	Options which is an argument of MQGET().	OR
MQOO_READ_AHEAD_AS_Q_DEF	0x00000000	Same as above	Same as above
MQOO_INPUT_AS_Q_DEF	0x00000001		Used when calling MQGET() and no MQOO_BROWSE or MQOO_INPUT * is specified.
MQOO_INPUT_SHARED	0x00000002		
MQOO_INPUT_EXCLUSIVE	0x00000004		
MQOO_BROWSE	0x00000008		Automatically set when browsing is specified.
MQOO_OUTPUT	0x00000010		Automatically set when calling MQPUT().
MQOO_INQUIRE	0x00000020		Automatically set when calling MQINQ().
MQOO_SET	0x00000040		Automatically set when calling MQSET().

Table 6.4.1 MQOO_*

Constant Name	Value	location	Operation and etc.
MQOO_SAVE_ALL_CONTEXT	0x00000080		
MQOO_PASS_IDENTITY_CONTEXT	0x00000100		
MQOO_PASS_ALL_CONTEXT	0x00000200		
MQOO_SET_IDENTITY_CONTEXT	0x00000400		
MQOO_SET_ALL_CONTEXT	0x00000800		
MQOO_ALTERNATE_USER_ AUTHORITY	0x00001000		
MQOO_FAIL_IF_QUIESCING	0x00002000		
MQOO_BIND_ON_OPEN	0x00004000		
MQOO_BIND_ON_GROUP	0x00400000		
MQOO_BIND_NOT_FIXED	0x00008000		
MQOO_CO_OP	0x00020000		
MQOO_NO_READ_AHEAD	0x00080000		
MQOO_READ_AHEAD	0x00100000		
MQOO_NO_MULTICAST	0x00200000		
MQOO_RESOLVE_LOCAL_Q	0x00040000		
MQOO_RESOLVE_LOCAL_TOPIC	0x00040000		
MQOO_RESOLVE_NAMES	0x00010000		

6.5 MQOD Parameters

MQOT_*

MQOT_* are set into MQOD.ObjectType.

Constants of the table below can be specified.

Table 6.5.1 MQOT_*

Constant Name	Value	location	Operation and etc.
MQOT_NONE	0	MQOD.ObjectType	Overwrite Same as above
MQOT_Q	1	Same as above	(default)
MQOT_NAMELIST	2		
MQOT_PROCESS	3		
MQOT_Q_MGR	5		
MQOT_TOPIC	8		
MQOT_CLNTCONN_CHANNEL	1014		

MQOD_* (MQOD*VERSION*)

MQOD_* are set into MQOD.Version.

Constants of the table below can be specified.

Table 6.5.2 MQOD_* (MQOD*VERSION*)

Constant Name	Value	location	Operation and etc.
MQOD_VERSION_1	1	MQOD.Version	Overwrite (default)
MQOD_VERSION_2	2	Same as above	Same as above
MQOD_VERSION_3	3		
MQOD_VERSION_4	4		
MQOD_CURRENT_VERSION	4		

6.6 MQPMO Parameters

MQPMO_* (MQPMO*VERSION*)

MQPMO_* are set into MQPMO.Version.

Constants of the table below can be specified.

Table 6.6.1 MQPMO*VERSION*

Constant Name	Value	location	Operation and etc.
MQPMO_VERSION_1	1	MQPMO.Version	Overwrite (default)
MQPMO_VERSION_2	2	Same as above	Same as above
MQPMO_VERSION_3	3		
MQPMO_CURRENT_VERSION	3		

MQPMO_* (except MQPMO*VERSION*, MQPMO_STRUC_ID, MQPMO_LENGTH_*)

MQPMO_* are set into MQPMO.Options.

Constants of the table below can be specified.

Table 6.6.2 MQPMO_* (except MQPMO*VERSION*, MQPMO_STRUC_ID, MQPMO_LENGTH_*)

Constant Name	Value	location	Operation and etc.
MQPMO_SYNCPOINT	0x00000002	MQPMO. Options	OR
MQPMO_NO_SYNCPOINT	0x00000004	Same as above	Same as above
MQPMO_DEFAULT_CONTEXT	0x00000020		
MQPMO_NEW_MSG_ID	0x00000040		
MQPMO_NEW_CORREL_ID	0x00000080		

Table 6.6.2 MQPMO_* (except MQPMO*VERSION*, MQPMO_STRUC_ID, MQPMO_LENGTH_*)			
Constant Name	Value	location	Operation and etc.
MQPMO_PASS_IDENTITY_CONTEXT	0x00000100		
MQPMO_PASS_ALL_CONTEXT	0x00000200		
MQPMO_SET_IDENTITY_CONTEXT	0x00000400		
MQPMO_SET_ALL_CONTEXT	0x00000800		
MQPMO_ALTERNATE_USER_AUTHORITY	0x00001000		
MQPMO_FAIL_IF_QUIESCING	0x00002000		
MQPMO_NO_CONTEXT	0x00004000		
MQPMO_LOGICAL_ORDER	0x00008000		
MQPMO_ASYNC_RESPONSE	0x00010000		
MQPMO_SYNC_RESPONSE	0x00020000		
MQPMO_RESOLVE_LOCAL_Q	0x00040000		
MQPMO_WARN_IF_NO_SUBS_MATCHED	0x00080000		
MQPMO_RETAIN	0x00200000		
MQPMO_MD_FOR_OUTPUT_ONLY	0x00800000		
MQPMO_SCOPE_QMGR	0x04000000		
MQPMO_SUPPRESS_REPLYTO	0x08000000		
MQPMO_NOT_OWN_SUBS	0x10000000		
MQPMO_RESPONSE_AS_Q_DEF	0x00000000		
MQPMO_RESPONSE_AS_TOPIC_DEF	0x00000000		
MQPMO_NONE	0x00000000		(default)
MQPMO_PUB_OPTIONS_MASK	0x00200000		

MQPMRF_*

MQPMRF_* are set into MQPMO.PutMsgRecFields.

Constants of the table below can be specified.

Table 6.6.3 MQPMRF_*

Constant Name	Value	location	Operation and etc.
MQPMRF_MSG_ID	0x00000001	MQPMO.PutMsgRecFields	OR
MQPMRF_CORREL_ID	0x00000002	Same as above	Same as above
MQPMRF_GROUP_ID	0x00000004		
MQPMRF_FEEDBACK	0x00000008		
MQPMRF_ACCOUNTING_TOKEN	0x00000010		
MQPMRF_NONE	0x00000000		(default)

6.7 MQGMO Parameters

MQGMO_* (MQGMO*VERSION*)

MQGMO_* are set into MQGMO.Version.

Constants of the table below can be specified.

Table 6.7.1 MQGMO*VERSION*

Constant Name	Value	location	Operation and etc.
MQGMO_VERSION_1	1	MQGMO.Version	Overwrite (default)
MQGMO_VERSION_2	2	Same as above	Same as above
MQGMO_VERSION_3	3		
MQGMO_VERSION_4	4		
MQGMO_CURRENT_VERSION	4		

MQGMO_* (except MQPMO*VERSION*, MQPMO_STRUC_ID, MQPMO_LENGTH_*)

MQGMO_* are set into MQGMO.Options.

Constants of the table below can be specified.

Table 6.7.2 MQGMO_* (except MQGMO*VERSION*, MQGMO_STRUC_ID, MQGMO_LENGTH_*)

Constant Name	Value	location	Operation and etc.
MQGMO_WAIT	0x00000001	MQGMO.Options	OR Same as above
MQGMO_NO_WAIT	0x00000000	Same as above	(default)
MQGMO_SET_SIGNAL	0x00000008		

Table 6.7.2 MQGMO_* (except MQGMO*VERSION*, MQGMO_STRUC_ID, MQGMO_LENGTH_*)

Constant Name	Value	location	Operation and etc.
MQGMO_FAIL_IF_QUIESCING	0x00002000		
MQGMO_SYNCPOINT	0x00000002		
MQGMO_SYNCPOINT_IF_PERSISTENT	0x00001000		
MQGMO_NO_SYNCPOINT	0x00000004		
MQGMO_MARK_SKIP_BACKOUT	0x00000080		
MQGMO_BROWSE_FIRST	0x00000010		
MQGMO_BROWSE_NEXT	0x00000020		
MQGMO_BROWSE_MSG_UNDER_CURSOR	0x00000800		
MQGMO_MSG_UNDER_CURSOR	0x00000100		
MQGMO_LOCK	0x00000200		
MQGMO_UNLOCK	0x00000400		
MQGMO_ACCEPT_TRUNCATED_MSG	0x00000040		
MQGMO_CONVERT	0x00004000		
MQGMO_LOGICAL_ORDER	0x00008000		
MQGMO_COMPLETE_MSG	0x00010000		
MQGMO_ALL_MSGS_AVAILABLE	0x00020000		
MQGMO_ALL_SEGMENTS_AVAILABLE	0x00040000		
MQGMO_MARK_BROWSE_HANDLE	0x00100000		
MQGMO_MARK_BROWSE_CO_OP	0x00200000		

Table 6.7.2 MQGMO_* (except MQGMO*VERSION*, MQGMO_STRUC_ID, MQGMO_LENGTH_*)

Constant Name	Value	location	Operation and etc.
MQGMO_UNMARK_BROWSE_CO_OP	0x00400000		
MQGMO_UNMARK_BROWSE_HANDLE	0x00800000		
MQGMO_UNMARKED_BROWSE_MSG	0x01000000		
MQGMO_PROPERTIES_FORCE_MQRFH2	0x02000000		
MQGMO_NO_PROPERTIES	0x04000000		
MQGMO_PROPERTIES_IN_HANDLE	0x08000000		
MQGMO_PROPERTIES_COMPATIBILITY	0x10000000		
MQGMO_PROPERTIES_AS_Q_DEF	0x00000000		(default)
MQGMO_NONE	0x00000000		
MQGMO_BROWSE_HANDLE	MQGMO_BROWSE_FIRST | MQGMO_UNMARKED_BROWSE_MSG | MQGMO_MARK_BROWSE_HANDLE		
MQGMO_BROWSE_CO_OP	MQGMO_BROWSE_FIRST | MQGMO_UNMARKED_BROWSE_MSG | MQGMO_MARK_BROWSE_CO_OP		

Ex. 6.7.1 Example of converting user data by specifying the MQGMO_CONVERT option.

$ mqpgf -qm TESTQM -q TQ -m "1234" **-cc 943 MQFMT_STRING**

[17/01/10 20:49:49] 1: message length: 4 put message : 1234

$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD] .... CodedCharSetId[943] Format[MQSTR   ] ....

data length: 4
00000000: 3132 3334   '1234   '

$ mqpgf -qm TESTQM -q TQ -br -cc 930 MQGMO_CONVERT
message number: 1
*StrucId[MD] .... CodedCharSetId[930] Format[MQSTR   ] ....

data length: 4
00000000: F1F2 F3F4   '□□   '

* CCSID(930=EBCDIC)

Ex. 6.7.2 Example of simultaneously converting RFH2 headder and user data by
specifying MQGMO_CONVERT optiion.

$ mqpgf -qm TESTQM -q TQ -ec 273 -cc 943 -mx 8eb18eb28eb3 -rf MQFMT_ST
RING -fg 100 -nd "test1,test22,test333" MQFMT_RF_HEADER_2 -rc 5050
[17/01/13 15:04:06] 1: message length: 6 put message : 0x8EB18EB28EB3

$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD] .... Encoding[273] CodedCharSetId[943] Format[MQHRF2] ....

*StrucId[RFH] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[5050]
Format[MQSTR   ] Flags[100] NameValueCCSID[1208]
NameValueLength[8] NameValueData[test1   ]
NameValueLength[8] NameValueData[test22   ]
NameValueLength[8] NameValueData[test333]
data length: 78
00000000: 8EB1 8EB2 8EB3   '竺軸宍   '

$ mqpgf -qm TESTQM -q TQ -br MQGMO_CONVERT -ec 546 -cc 930
message number: 1
*StrucId[MD] .... Encoding[546] CodedCharSetId[930] Format[MQHRF2] ....

data length: 75
00000000: D9C6 C840 0200 0000 4800 0000 2202 0000   'ﾙﾆﾈ@....H..."...'

6-7-4

```
00000010:  A203 0000 D4D8 E2E3 D940 4040 6400 0000   '「...ヤリ粤ル@@@d...'
00000020:  B804 0000 0800 0000 7465 7374 3120 2020   'ク.......test1   '
00000030:  0800 0000 7465 7374 3232 2020 0800 0000   '....test22   ....'
00000040:  7465 7374 3333 3320 8182 83               'test333 ≠.      '
```

* The character code of MQRFH2 itself has been converted from SHIFTJIS (CCS ID 943) to EBCDIC (CCSID 930).
* 00000000-00000003：D9C6 C840 = EBCDIC "RFH "

* Encoding of MQRFH 2 itself is converted to Little Endian (Encoding 546).
00000004-00000007：0200 0000

* MQRFH2.Encoding is set to 546 (Little Endian)
0000000C-0000000F：2202 0000= 0000 0222(Big Endian) = 546(Decimal)

* MQRFH2.CodedCharSetId is set to EBCDIC (CCSID 930)
00000010-00000011：A203 0000 = 0000 03A2(Big Endian) = 930(Decimal)EBCDI C CCSID

* MQRFH2.NameValueCCSID remains as UTF-8 (CCSID 1208) (note that NameV alueData is not target to conversion)
00000020-00000023：B804 0000 = 0000 04B8(Big Endian) = 1208(Decimal)UTF-8 CCSID

* The user data is converted from eucJP to Japanese EBCDIC.
00000048-0000004a：8182 83 = Japanece EBCDIC half-width characters "ｱｲｳ"

Ex. 6.7.3 Example of code conversion of PCF message by specifying MQGMO_CONVERT.

$ mqpgf -qm TESTQM -q TQ -pcf sample3.def MQCCSI_EMBEDDED
Command：99
Id：1111, MQCFT_BYTE_STRING：10, 1234567890
Id：2222, MQCFT_BYTE_STRING_FILTER：1 5 1A2B3
Id：3501, MQCFT_STRING：943 3 ｱｲｳ
Id：4444, MQCFT_STRING_FILTER：2 1208 9 .ｽｱ.ｽｲ.ｽｳ
Id：5555, MQCFT_STRING_LIST：930 5 3 [□□],[□□],[□□],[□□],[□.]
Id：6666, MQCFT_INTEGER：1234567890
Id：7777, MQCFT_INTEGER_FILTER：6 -3
Id：8888, MQCFT_INTEGER_LIST：3 [1234],[-1],[5678]
Id：9999, MQCFT_INTEGER64：9223372036854775807
Id：1234, MQCFT_INTEGER64_LIST：5 [4294967294],[4294967291],[-5],[9223372 036854775803],[4294967290]

```

[17/01/13 16:51:13] 1: put from sample3.def

$ mqpgf -qm TESTQM -q TQ -brv -hex
message number: 1
*StrucId[MD  ] .... Encoding[273] **CodedCharSetId[-1] Format[MQPCF   ]** ....

....
*MQCFH(MQCFT_USER) ....

....
(MQCFT_STRING) Type[4] StrucLength[24] Parameter[3501] **CodedCharSetId[943]** StringLength[3] **String[0xB1B2B3]**
(MQCFT_STRING_FILTER) Type[14] StrucLength[36] Parameter[4444] Operator [2] **CodedCharSetId[1208]** FilterValueLength[9] **FilterValue[0xEFBDB1EFBDB2EFBDB3]**
(MQCFT_STRING_LIST) Type[6] StrucLength[40] Parameter[5555] **CodedCharSetId[930]** Count[5] StringLength[3] **String1[0xF1F1F1] String2[0xF1F2F3] String3[0xF3F3F3] String4[0xF5F6F7] String5[0xF5F5F5]**

....
* CCSID 943(SHIFT_JIS) 0xB1 B2 B3 are half-width characters "ｱｲｳ"
* CCSID 1208 (UTF-8) 0xEFBDB1 EFBDB2 EFBDB3 are half-width characters "ｱｲｳ"
* CCSID 930 (EBCDIC) 0xF0 F1 F2 F3 ... F9 are half-width characters "0123 ... 9"

$ mqpgf -qm TESTQM -q TQ -brv MQGMO_CONVERT -cc 5050 -hex
message number: 1
*StrucId[MD  ] .... **CodedCharSetId[5050] Format[MQPCF   ]** ....

....
*MQCFH(MQCFT_USER) ....

....
(MQCFT_STRING) Type[4] StrucLength[28] Parameter[3501] **CodedCharSetId[5050]** StringLength[6] **String[0x8EB18EB28EB3]**
(MQCFT_STRING_FILTER) Type[14] StrucLength[32] Parameter[4444] Operator [2] **CodedCharSetId[5050]** FilterValueLength[6] **FilterValue[0x8EB18EB28EB3]**
(MQCFT_STRING_LIST) Type[6] StrucLength[40] Parameter[5555] **CodedCharSetId[5050]** Count[5] StringLength[3] **String1[0x313131] String2[0x313233] String3[0x333333] String4[0x353637] String5[0x353535]**

....

* All parameters CodedCharSetId are set to 5050 (eucJP).
* 0x8EB1 8EB2 8EB3 are eucJP (5050) half-width character "ｱｲｳ"
* 0x30 31 32 33 ... 39 are eucJP(5050) half-width character "0123 ... 9"
------------------------------------

## MQWI_*

MQWI_* are set into MQGMO.WaitInterval.

Constants of the table below can be specified.

| Table 6.7.3 MQWI_* | | | |
| --- | --- | --- | --- |
| **Constant Name** | **Value** | **location** | **Operation and etc.** |
| MQWI_UNLIMITED | (-1) | MQGMO.WaitInterval | Overwrite |

## MQMO_*

MQMO_* are set into MQGMO.MatchOptions.

Constants of the table below can be specified.

| Table 6.7.4 MQMO_* | | | |
| --- | --- | --- | --- |
| **Constant Name** | **Value** | **location** | **Operation and etc.** |
| MQMO_MATCH_MSG_ID | 0x00000001 | MQGMO.Match Options | OR |
| MQMO_MATCH_CORREL_ID | 0x00000002 | Same as above | Same as above |
| MQMO_MATCH_GROUP_ID | 0x00000004 | | |
| MQMO_MATCH_MSG_SEQ_NUMBER | 0x00000008 | | |
| MQMO_MATCH_OFFSET | 0x00000010 | | |
| MQMO_MATCH_MSG_TOKEN | 0x00000020 | | |
| MQMO_NONE | 0x00000000 | | |

Ex. 6.7.4 Get a specific message by specifing MQMO_MATCH_MSG_ID.
```

$ mqpgf -qm TESTQM -q TQ -m test -n 3
[17/01/10 21:29:58] 1: message length: 4 put message : test
[17/01/10 21:29:58] 2: message length: 4 put message : test
[17/01/10 21:29:58] 3: message length: 4 put message : test
```

$ mqpgf -qm TESTQM -q TQ -br -r
message number: 1
*StrucId[MD  ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005
E03] ....
message number: 2
*StrucId[MD  ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005
E04] .....
message number: 3
*StrucId[MD  ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005
E05] ....

* It gets 2nd message with its MsgId specified. (It is not mandatory to specify M
QMO_MATCH_MSG_ID because MQGMO_DEFAULT has MQMO_MATCH_MSG_ID.)
$ mqpgf -qm TESTQM -q TQ **-mi 0x414D51206F6B61716D3830612020202058746**
**D1420005E04 MQMO_MATCH_MSG_ID**
[17/01/10 21:35:34] 1: message length: 4 get message : test

$ mqpgf -qm TESTQM -q TQ -br -r
message number: 1
*StrucId[MD  ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005
E03] ....
message number: 2
*StrucId[MD  ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005
E05] ....
no message available : TQ CompCd=02 ReasonCd=2033
-------------------------------------

# 6.8 MQCLOSE Option

## MQCO_*

MQCO_* are set into the argument "Options" of MQCLOSE(). The Default value of this option is MQCO_NONE. When two or more MQCO_ * are specified, each is turned on.

| Table 6.8.1 MQCO_* | | | |
|---|---|---|---|
| Constant Name | Value | location | Operation and etc. |
| MQCO_IMMEDIATE | 0x00000000 | Options which is an argument of MQGET(). | OR |
| MQCO_NONE | 0x00000000 | Same as above | Same as above |
| MQCO_DELETE | 0x00000001 | | |
| MQCO_DELETE_PURGE | 0x00000002 | | |
| MQCO_KEEP_SUB | 0x00000004 | | |
| MQCO_REMOVE_SUB | 0x00000008 | | |
| MQCO_QUIESCE | 0x00000020 | | |

Ex. 6.8.1 Example of deleting permanent dynamic queue when calling MQCLOSE ()
--------------------------------------
* Specify SYSTEM.DURABLE.MODEL.QUEUE to create a permanent dynamic queue.

$ mqpgf -qm SampleQM -q **SYSTEM.DURABLE.MODEL.QUEUE** -dq DYNAMICQ -s MQCLOSE MQCO_DELETE
no message available : SYSTEM.DURABLE.MODEL.QUEUE CompCd=02 ReasonCd=2033
MQCMIT success : CompCd=00 ReasonCd=00
stop before calling MQCLOSE().
Hit Any Key!!!

* Confirm permanent dynamic queues created from other terminals.

$ mqpcf que -qm SampleQM -q DYNAMICQ TYPE DEFTYPE
1: QUEUE(DYNAMICQ) TYPE(QLOCAL) **DEFTYPE(PERMDYN)**

* Enter an arbitrary key in first terminal, continue processing and call MQCLOS
E().

$ mqpcf que -qm SampleQM -q DYNAMICQ TYPE DEFTYPE
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], **mqCommandRC=[2085]**
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], mqCommandRC=[3008]

mqrc 2085

      2085   0x00000825   **MQRC_UNKNOWN_OBJECT_NAME**

* Permanent dynamic queue has been deleted.
-------------------------------------

# 6.9 MQSETMP Option

## MQPD_*

It is set to MQPD.Context.

Constants of the table below can be specified.

Table 6.9.1 MQPD_*

| Constant Name | Value | location | Operation and etc. |
|---|---|---|---|
| MQPD_USER_CONTEXT | 0x00000000 | MQPD.Context | Overwrite |
| MQPD_NO_CONTEXT | 0x00000001 | MQPD.Context | Overwrite |

Ex. 6.9.1 Example of re-queuing while inheriting input identification, origin, user context
--------------------------------------
* First, to associate a message property with a user context, it is necessary to specify MQPD_USER_CONTEXT.
$ mqpgf -qm SampleQM -q SampleQ -m "test1" -smp "MQTYPE_BOOLEAN:boolean1:TRUE,MQTYPE_BYTE_STRING:byteString:0102feff" MQPMO_VERSION_3 **MQPD_USER_CONTEXT**
[18/02/02 09:38:14] 1: message length: 5 put message : test1

* When MQPD_USER_CONTEXT is set, "context = 'user'" is added to the XML attribute of the message property.
$ mqpgf -qm SampleQM -q SampleQ -brv
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[819] Format[MQHRF2  ] Priority[0] Persistence[0] MsgId[0x414D512053616D706C65514D202020205A73B22B20002412] CorrelId[0x0000000000000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[                                ] ReplyToQMgr[SampleQM              ] **UserIdentifier[mqm**          ] AccountingToken[0x0534343033331000000000000000000000000000000000000000000000000006] ApplIdentityData[                            ] PutApplType[13] PutApplName[mqpgf          ] PutDate[20180202] PutTime[00381410] ApplOriginData[    ]

GroupId[0x000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[RFH ] Version[2] StrucLength[164] Encoding[273] CodedCharSetId[819] Format[          ] Flags[0] NameValueCCSID[1208]

NameValueLength[124] NameValueData[<usr><boolean1 dt='boolean' **context='user'** >1</boolean1><byteString dt='bin.hex' **context='user'**>0102FEFF</byteString></usr> ]
data length: 169
00000000:  7465 7374 31                                    'test1           '

* Specify MQGMO_PROPERTIES_IN_HANDLE at the time of re-queuing so that int ernal message property is enabled by MsgHandle. If you do not specify the MQOO_S AVE_ALL_CONTEXT option, the context information of the original message will not be saved. To pass all the contexts, specify MQOO_PASS_ALL_CONTEXT and MQPM O_PASS_ALL_CONTEXT. mqpgf sets the handle of the input queue to MQPMO.Cont ext if MQPMO_PASS_ * is specified.
$ mqpgf -qm SampleQM -q SampleQ -oq SampleQ2 **MQGMO_PROPERTIES_IN_ HANDLE** MQGMO_VERSION_4 **MQOO_SAVE_ALL_CONTEXT** -oo **MQOO_PASS_ ALL_CONTEXT MQPMO_PASS_ALL_CONTEXT** MQPMO_VERSION_3
[18/01/31 19:35:23] 1: message length: 5 get message : test1
[18/01/31 19:35:23] 1: message length: 5 put message : test1


$ mqpgf -qm SampleQM -q SampleQ2 -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding [273] CodedCharSetId[819] Format[MQHRF2 ] Priority[0] Persistence[0] MsgId[0 x414D512053616D706C65514D202020205A73B22B20002412] CorrelId[0x000000000 000000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[
                                       ] ReplyToQMgr[SampleQM
                 ] **UserIdentifier[mqm           ]** AccountingToken[0x05343 4303331000000000000000000000000000000000000000000000000000000006] **ApplIdentity Data[                                ]** PutApplType[13] **PutApplName[mqpgf**
          **]** PutDate[20180202] PutTime[00381410] ApplOriginData[    ]

GroupId[0x0000000000000000000000000000000000000000000000000] MsgSeqNumbe r[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[RFH ] Version[2] StrucLength[164] Encoding[273] CodedCharSetId[819] Format[         ] Flags[0] NameValueCCSID[1208]
NameValueLength[124] NameValueData[<usr>**<boolean1 dt='boolean' context='user'** >1</boolean1>**<byteString dt='bin.hex' context='user'>0102FEFF</byteString>**</usr> ]
data length: 169
00000000:  7465 7374 31                                    'test1           '

MQCMIT success : CompCd=00 ReasonCd=00
-------------------------------------

# 6.10 MQINQMP Options

## MQIMPO_*

MQIMPO_* are set into MQIMPO.Options.

Constants of the table below can be specified.

Table 6.10.1 MQIMPO_*

| Constant Name | Value | location | Operation and etc. |
|---|---|---|---|
| MQIMPO_CONVERT_TYPE | 0x00000000 | MQIMPO. Options | OR |
| MQIMPO_QUERY_LENGTH | 0x00000001 | Same as above | Same as above |
| MQIMPO_INQ_FIRST | 0x00000000 | | (default) |
| MQIMPO_INQ_NEXT | 0x00000008 | | This value is automatically set for the second and subsequent property calls. |
| MQIMPO_INQ_PROP_UNDER_CURSOR | 0x00000010 | | |
| MQIMPO_CONVERT_VALUE | 0x00000020 | | |
| MQIMPO_NONE | 0x00000000 | | |

Ex. 6.10.1 Example of converting property data by specifying MQIMPO_CONVERT_VALUE.
```

$ mqpgf -qm mqm90a -q TQ -m "test" -smp "MQTYPE_STRING:string:123,MQTYPE_INT32:int32:1" MQPMO_VERSION_3
[17/01/12 20:26:24] 1: message length: 4 put message : test

$ mqpgf -qm mqm90a -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VERSION_4 -br
message number: 1
*StrucId[MD] .... Encoding[546] CodedCharSetId[1208] ....
```

****Message properties****

  **string**：**'123'**
  **int32**：**1**
MQINQMP faild：CompCd=02 ReasonCd=2471
....

$ mqpgf -qm mqm90a -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VER
SION_4 -br **MQIMPO_CONVERT_VALUE -br -pe 273 -pc 930** -hex
message number: 1
*StrucId[MD  ] .... Encoding[546] CodedCharSetId[1208] ....

****Message properties****

  **0xABB39B717668**：**'0xF1F2F3'**
  **0x7176B3F3F2**：**16777216**
MQINQMP faild：CompCd=02 ReasonCd=2471

....
* When MQIMPO_CONVERT_VALUE is specified, property names are also conve
rted.
* F1F2F3 are EBCDIC numeric characters '123'.
* 16777216 is 0x01 decimal notation with 32 bit reverse endian.(0x01000000=167
77216)

* MQINQMP() is invoked repeatedly until the reason code: 2471 (MQRC_PROPE
RTY_NOT_AVAILABLE) is returned.
------------------------------------

# 6.11 MQCRTMH Options

## MQCMHO_*

MQCMHO_* are set into MQCRTMH.Options.

Constants of the table below can be specified.

Table 6.11.1 MQCMHO_*

| Constant Name | Value | location | Operation and etc. |
|---|---|---|---|
| MQCMHO_DEFAULT_VALIDATION | 0x00000000 | MQCRTMH.Options | Overwrite (default) |
| MQCMHO_NO_VALIDATION | 0x00000001 | Same as above | Same as above |
| MQCMHO_VALIDATE | 0x00000002 | | |
| MQCMHO_NONE | 0x00000000 | | |

Ex. 6.11.1 Example of validating a property name by MQSETMP().
----------------------------------------
$ mqpgf -qm TESTQM -q TQ -m "test" -smp "MQTYPE_STRING:*:test" MQPMO_VERSION_3 **MQCMHO_VALIDATE**
MQSETMP faild : CompCd=02 ReasonCd=2442
$ mqrc 2442
$

     2442   0x0000098a   **MQRC_PROPERTY_NAME_ERROR**

$
$ mqpgf -qm TESTQM -q TQ -m "test" -smp "MQTYPE_STRING:*:test" MQPMO_VERSION_3 MQCMHO_NO_VALIDATION
$
[17/01/11 18:38:39] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VERSION_4 -br
message number: 1
....
****Message properties****

  * : 'test'

MQINQMP faild：CompCd=02 ReasonCd=2471
....
--------------------------------------

# 6.12 MQCBD Parameters

## MQCBT_*

MQCBT_* are set into MQCBT.CallbackType.

Constants of the table below can be specified.

Table 6.12.1 MQCBT_*

| Constant Name | Value | location | Operation and etc. |
|---|---|---|---|
| MQCBT_MESSAGE_CONSUMER | 0x00000001 | MQCBD.Callback Type | Overwrite (default) |
| MQCBT_EVENT_HANDLER | 0x00000002 | Same as above | Same as above |

## 6.13 MQAIR Parameters

### MQAIR_* (MQAIR*VERSION*)

MQAIR*VERSION* are set into MQAIR.Version.

Constants of the table below can be specified.

Table 6.13.1 MQAIR_*

| Constant Name | Value | location | Operation and etc. |
|---|---|---|---|
| MQAIR_VERSION_1 | 1 | MQAIR.Version | Overwrite (default) |
| MQAIR_VERSION_2 | 2 | Same as above | Same as above |
| MQAIR_CURRENT_VERSION | 1 | Same as above | Same as above |

### MQAIT_*

MQAIT * are set into MQAIR.AuthInfoType.

Constants of the table below can be specified.

Table 6.13.2 MQAIT_*

| Constant Name | Value | location | Operation and etc. |
|---|---|---|---|
| MQAIT_CRL_LDAP | 1 | MQAIR.AuthInfo Type | Overwrite (default) |
| MQAIT_OCSP | 2 | Same as above | Same as above |

# Conclusion

If you find any defects in this program, or if you have any questions and requests about this program, please contact us.

Pulsar Integration Inc.
https://www.pulsarintegration.com
https://www.pulsarintegration.jp
e-mail: support@pulsarintegration.com


The HP NonStop version of this program uses the sha2 signature program, which is copyrighted below.

```
* AUTHOR: Aaron D. Gifford - http://www.aarongifford.com/
*
* Copyright (c) 2000-2001, Aaron D. Gifford
* All rights reserved.
*
```