IBM MQ

MQAI program (mqpcf)

Ver 1.4.0.12
20 August 2022

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

Program Version 1.4.0.12

Tested MQ and OS version

Windows 10 64bit / IBM MQ 9.0 / 9.1 / 9.2.0 / 9.2.3
Windows 10 64bit / IBM MQ 9.1 / 9.2.1 Client
Linux RHEL Server release 7.4 (Maipo) / IBM MQ 9.0 / 9.2.3
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 J06.20.00 / IBM MQ 8.0
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 / IBM MQ 8.1

The Linux x86 and Windows edition of this program are compiled with 32 bit, so these can operate on both 64 bit and 32 bit OS. Note when running on 64bit Linux, the 32bit runtime library must be installed.

This program can be used in all versions of WebSphere MQ 5.3 and above exceptfor some functions.

It can be executed at many OS levels other than the verified environments above.

Table Of Contents

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 MQAI program (mqpcf) command is also used. For details on the MQAI program (mqpcf) command, refer to the document "MQAI program (mqpcf)".

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 no t compatible. However, operation compatibility is maintained as much as possible. A User's Guide is created separately.

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

Modification: Most of the source code has been maintained, but code has been ad ded for minor new features. Version information is added to the description of th e 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 lo wer levels will be applied at the same time. For example, if the Modification lev el 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 functi ons 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 o nly certain fixes. All previous fixes will be applied. For example, if the version i n 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 releas ed.
However, if the V.R.M requested to apply the correction is at a significantly earli er level, it may be difficult to apply the correction. In that case, we may ask yo u 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.

Starting the command server

Since this program uses MQAI, a command server must be running. The command server is started by default in the queue manager of MQ V7.0 or higher that is targeted by this program.

If it has not been started, start it as follows.

```
$ dspmqcsv <queue manager>

WebSphere MQ command server status  . . :   Stopped

$ strmqcsv <queue manager>
WebSphere MQ command server started.

$ dspmqcsv TESTQM

WebSphere MQ Command Server Status  . . :   Running
```

MQ Installation 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_PAT
H
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 mqpcf without any arguments, the usage and the parameters that can be specified are displayed.

Ex. 3.1 Display usage

```
------------------------------------
$ mqpcf
USAGE :
  mqpcf qmgr -qm Qmgr
  mqpcf qms -qm Qmgr
  mqpcf que -qm Qmgr
  mqpcf cque -qm Qmgr
  mqpcf ques -qm Qmgr
  mqpcf queh -qm Qmgr
  mqpcf chl -qm Qmgr
  mqpcf chs -qm Qmgr
  mqpcf lsnr -qm Qmgr
  mqpcf lsst -qm Qmgr
  mqpcf cqmgr -qm Qmgr
  mqpcf con -qm Qmgr
  mqpcf pngm -qm Qmgr
  mqpcf ping -qm Qmgr -c Channel
  mqpcf {put | get} {enable | disable} -qm Qmgr -q Queue
  mqpcf clr -qm Qmgr -q Queue
  mqpcf rst -qm Qmgr -c Channel
  mqpcf rslv -qm Qmgr -c Channel {commit | backout}
  mqpcf sta -qm Qmgr -c Channel
  mqpcf stp -qm Qmgr -c Channel
  mqpcf stalsn -qm Qmgr -ln Listener
  mqpcf stplsn -qm Qmgr -ln Listener
  mqpcf mqsc -qm Qmgr {-f MqscFile | -s 'Mqcmd'}
-rc : repeat count
-i  : repeat interval(sec)
-t  : display time
-wi : The maximum time(sec) that the MQAI waits for each reply message
-cu : UserId
-ci : Password
select parameters to display(e.g. mqpcf chs .. SUBSTATE MCASTAT)
```

A node-locked license is required to run this program. The license file should be located in the current directory or set the MQTOOL_LICENSE_FILE environmen t variable.

e.g. export MQTOOL_LICENSE_FILE=/home/MQHOME/license/mqtool.lic

```
$ mqpcfc
USAGE :
   mqpcfc qmgr -qm Qmgr
   mqpcfc qms -qm Qmgr
   mqpcfc que -qm Qmgr
   mqpcfc cque -qm Qmgr
   mqpcfc ques -qm Qmgr
   mqpcfc queh -qm Qmgr
   mqpcfc chl -qm Qmgr
   mqpcfc chs -qm Qmgr
   mqpcfc lsnr -qm Qmgr
   mqpcfc lsst -qm Qmgr
   mqpcfc cqmgr -qm Qmgr
   mqpcfc con -qm Qmgr
   mqpcfc pngm -qm Qmgr
   mqpcfc ping -qm Qmgr -c Channel
   mqpcfc {put | get} {enable | disable} -qm Qmgr -q Queue
   mqpcfc clr -qm Qmgr -q Queue
   mqpcfc rst -qm Qmgr -c Channel
   mqpcfc rslv -qm Qmgr -c Channel {commit | backout}
   mqpcfc sta -qm Qmgr -c Channel
   mqpcfc stp -qm Qmgr -c Channel
   mqpcfc stalsn -qm Qmgr -ln Listener
   mqpcfc stplsn -qm Qmgr -ln Listener
   mqpcfc mqsc -qm Qmgr {-f MqscFile | -s 'Mqcmd'}
-rc : repeat count
-i  : repeat interval(sec)
-t  : display time
-wi : The maximum time(sec) that the MQAI waits for each reply message
-cu : UserId
-ci : Password
The following parameters are only for client connections with mqpcfc
-x  : Connection Name
-ch : Connection Channel
-la : LocalAddress
-lb : Certificate Label
-cs : SSLCipher Spec
-er : SSLPeerName
-kr : Key Repository
select parameters to display(e.g. mqpcf chs .. SUBSTATE MCASTAT)

A node-locked license is required to run this program. The license file should be
located in the current directory or set the MQTOOL_LICENSE_FILE environmen
```

t variable.
e.g. export MQTOOL_LICENSE_FILE=/home/MQHOME/license/mqtool.lic

---

Commands in the table below can be specified.

Table 3.1 Available Commands

Command	MQAI command	Paramaters
qmgr	MQCMD_INQUIRE_Q_MGR	
qms	MQCMD_INQUIRE_Q_MGR_STATUS	
que	MQCMD_INQUIRE_Q	MQIA_Q_TYPE: MQQT_LOCAL
cque	MQCMD_INQUIRE_Q	MQIA_Q_TYPE: MQQT_CLUSTER
ques	MQCMD_INQUIRE_Q_STATUS	
queh	MQCMD_INQUIRE_Q_STATUS	MQIACF_Q_STATUS_TYPE: MQIACF_Q_HANDLE
chl	MQCMD_INQUIRE_CHANNEL	
chs	MQCMD_INQUIRE_CHANNEL_STATUS	MQIACH_CHANNEL_INSTANCE_TYPE: MQOT_SAVED_CHANNEL or MQOT_CURRENT_CHANNEL
lsnr	MQCMD_INQUIRE_LISTENER	
lsst	MQCMD_INQUIRE_LISTENER_STATUS	
cqmgr	MQCMD_INQUIRE_CLUSTER_Q_MGR	
con	MQCMD_INQUIRE_CONNECTION	MQIACF_CONN_INFO_TYPE: MQIACF_CONN_INFO_ALL or MQIACF_CONN_INFO_CONN or MQIACF_CONN_INFO_HANDLE
pngm	MQCMD_PING_Q_MGR	
ping	MQCMD_PING_CHANNEL	

Table 3.1 Available Commands

Command	MQAI command	Paramaters
put	MQCMD_CHANGE_Q	MQIA_Q_TYPE: MQQT_LOCAL MQIA_INHIBIT_PUT: MQQA_PUT_ALLOWED or MQQA_PUT_INHIBITED
get	MQCMD_CHANGE_Q	MQIA_Q_TYPE: MQQT_LOCAL MQIA_INHIBIT_GET: MQQA_GET_ALLOWED or MQQA_GET_INHIBITED
clr	MQCMD_CLEAR_Q	
rst	MQCMD_RESET_CHANNEL	
rslv	MQCMD_RESOLVE_CHANNEL	MQIACH_IN_DOUBT: MQIDO_COMMIT or MQIDO_BACKOUT
sta	MQCMD_START_CHANNEL	MQIACF_MODE: MQMODE_FORCE(option) MQIACH_CHANNEL_STATUS: MQCHS_INACTIVE(option)
stp	MQCMD_STOP_CHANNEL	
stalsn	MQCMD_START_CHANNEL_LISTENER	
stplsn	MQCMD_STOP_CHANNEL_LISTENER	
mqsc	MQCMD_ESCAPE	

Executing only the command name displays the details of the command and the parameters that can be specified.

Ex. 3.2 Display usage details by command.

$ mqpcf qmgr
USAGE : mqpcf qmgr -qm Qmgr

$ mqpcfc qmgr
USAGE : mqpcfc qmgr -qm Qmgr [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf qms
USAGE : mqpcf qms -qm Qmgr

$ mqpcfc qms
USAGE : mqpcfc qms -qm Qmgr [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf que
USAGE : mqpcf que -qm Qmgr [-q Queue]

$ mqpcfc que
USAGE : mqpcfc que -qm Qmgr [-q Queue] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf cque
USAGE : mqpcf cque -qm Qmgr [-q Queue] [-cl Cluster]

$ mqpcfc cque
USAGE : mqpcfc cque -qm Qmgr [-q Queue] [-cl Cluster] -[-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf ques
USAGE : mqpcf ques -qm Qmgr [-q Queue]

$ mqpcfc ques
USAGE : mqpcfc ques -qm Qmgr [-q Queue] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf queh
USAGE : mqpcf queh -qm Qmgr [-q Queue]

$ mqpcfc queh
USAGE : mqpcfc queh -qm Qmgr [-q Queue] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf chl
USAGE : mqpcf chl -qm Qmgr [-c Channel]

$ mqpcfc chl
USAGE : mqpcfc chl -qm Qmgr [-c Channel] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf chs
USAGE : mqpcf chs -qm Qmgr [-c Channel] [-cn Connection] [saved]

$ mqpcfc chs
USAGE : mqpcfc chs -qm Qmgr [-c Channel] [-cn Connection] [saved] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf lsnr
USAGE : mqpcf lsnr -qm Qmgr [-ln Listener]

$ mqpcfc lsnr
USAGE : mqpcfc lsnr -qm Qmgr [-ln Listener] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf lsst
USAGE : mqpcf lsst -qm Qmgr [-ln Listener]

$ mqpcfc lsst
USAGE : mqpcfc lsst -qm Qmgr [-ln Listener] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf cqmgr
USAGE : mqpcf cqmgr -qm Qmgr [-cl Cluster] [-g GenericQmgr]

$ mqpcfc cqmgr
USAGE : mqpcfc cqmgr -qm Qmgr [-cl Cluster] [-g GenericQmgr] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf con
USAGE : mqpcf con -qm Qmgr {conn | handle}[-ap ApplTag]

$ mqpcfc con
USAGE : mqpcfc con -qm Qmgr {conn | handle}[-ap ApplTag] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf pngm
USAGE : mqpcf pngm -qm Qmgr

$ mqpcfc pngm
USAGE : mqpcfc pngm -qm Qmgr [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf ping
USAGE : mqpcf ping -qm Qmgr -c Channel [-l DataLen(16-32768)]

$ mqpcfc ping
USAGE : mqpcfc ping -qm Qmgr -c Channel [-l DataLen(16-32768)] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf put
USAGE : mqpcf {put | get} {enable | disable} -qm Qmgr -q Queue

$ mqpcfc put
USAGE : mqpcfc {put | get} {enable | disable} -qm Qmgr -q Queue [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf get
USAGE : mqpcf {put | get} {enable | disable} -qm Qmgr -q Queue

$ mqpcfc get
USAGE : mqpcfc {put | get} {enable | disable} -qm Qmgr -q Queue [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf clr
USAGE : mqpcf clr -qm Qmgr -q Queue

$ mqpcfc clr
USAGE : mqpcfc clr -qm Qmgr -q Queue -x Connection [-ch ConnectChannel]

$ mqpcf rst
USAGE : mqpcf rst -qm Qmgr -c Channel [-n SeqNo(1-999999999)]

$ mqpcfc rst

USAGE : mqpcfc rst -qm Qmgr -c Channel [-n SeqNo(1-999999999)] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf rslv
USAGE : mqpcf rslv -qm Qmgr -c Channel {commit | backout}

$ mqpcfc rslv
USAGE : mqpcfc rslv -qm Qmgr -c Channel {commit | backout} [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf sta
USAGE : mqpcf sta -qm Qmgr -c Channel

$ mqpcfc sta
USAGE : mqpcfc sta -qm Qmgr -c Channel [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf stp
USAGE : mqpcf stp -qm Qmgr -c Channel [force | term] [inact] [-rm RemoteQmgr] [-cn Connection]

$ mqpcfc stp
USAGE : mqpcfc stp -qm Qmgr -c Channel [force | term] [inact] [-rm RemoteQmgr] [-cn Connection] [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf stalsn
USAGE : mqpcf stalsn -qm Qmgr -ln Listener

$ mqpcfc stalsn
USAGE : mqpcfc stalsn -qm Qmgr -ln Listener [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

$ mqpcf stplsn
USAGE : mqpcf stplsn -qm Qmgr -ln Listener

$ mqpcfc stplsn
USAGE : mqpcfc stplsn -qm Qmgr -ln Listener [-x Connection] [-ch ConnectChannel] [-la LocalAddress]

```
$ mqpcf mqsc
USAGE : mqpcf mqsc -qm Qmgr {-f MqscFile | -s 'Mqcmd'}

$ mqpcfc mqsc
USAGE : mqpcfc mqsc -qm Qmgr {-f MqscFile | -s 'Mqcmd'} [-x Connection] [-ch
ConnectChannel] [-la LocalAddress]
---------------------------------------
```

Display license information and version information

License information and version information of this program and linked library is displayed on the last line in addition to the USAGE display when -v is specified for mqpcf.

Ex. 3.3 Display license information and version information.

```
---------------------------------------
$ mqpcf -v
USAGE :
....
[ License information ]
System number     999999
Expires           2022.03.31

version 1.4.0.11 2021.09.10
library version 1.0.0.1 2021/03/10
---------------------------------------
```

Using client mode

When using in client mode, use the mqpcfc command.

Except for options for client connections, the usage is the same as mqpcf for bind mode.

mqpcfc receives the IP address or host name of the connection destination with the -x option, the connection port number, the MQI channel name with the -ch option, and the local address with the -la option. If -x or -la is specified, mqpcfc passes the connection parameters to MQCONNX().

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.

mqpcfc <command> -qm <qmgr> <-q etc..> -x "ipaddr or hostname (port)" -ch "channel name" -la "local ipaddr or hostname (sender port)[/tcp process name]"

e.g.
mqpcfc que -qm SampleQM -q SampleQ -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)

Using Using SSL/TLS Channels

When connecting in client mode using the mqpcfc command, it is possible to use SSL/TLS. See the IBM MQ product documentation for how to set up SSL/TLS channels. The following SSL/TLS parameters can be specified in mqpcfc. Specify the parameters according to the purpose and connect.

Table 3.2 SSL/TLS related parameters

Option	Value	Description
-lb	Certificate Label	Specify if you want to use a certificate label other than the default label.
-cs	SSLCipher Spec	SSL cipher spec to use.
-er	SSLPeerName	Specify the character string for verifying the name of the SSL Peer when it is necessary to confirm it.
-kr	Key Repository	Specifies the location of the key repository. For GSkit, specify <directory>/<part of key DB excluding extension>. For Openssl (MQ for HPE NonStop, etc.), specify the directory where the certificate file is located.

4. Command reference

For reference commands, some required parameters are displayed at the top and other optional parameters are displayed in alphabetical order.

Inquire Queue Manager (qmgr)

The qmgr command execute MQCMD_INQUIRE_Q_MGR MQAI command. It is equivalent to "display qmgr " of the runmqsc command.

mqpcf qmgr -qm Qmgr

Parameters in the table below are displayed.

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
QMgrName	QMNAME	MQCA_Q_MGR_NAME	Mandatory
AccountingConnOverride	ACCTCONO	MQIA_ACCOUNTING_CONN_OVERRIDE	
AccountingInterval	ACCTINT	MQIA_ACCOUNTING_INTERVAL	
MQIAccounting	ACCTMQI	MQIA_ACCOUNTING_MQI	
QueueAccounting	ACCTQ	MQIA_ACCOUNTING_Q	
MaxActiveChannels	ACTCHL	MQIA_ACTIVE_CHANNELS	[z/OS]
ActivityRecording	ACTIVREC	MQIA_ACTIVITY_RECORDING	
ActivityConnOverride	ACTVCONO	MQIA_ACTIVITY_CONN_OVERRIDE	
ActivityTrace	ACTVTRC	MQIA_ACTIVITY_TRACE	
AdoptNewMCACheck	ADOPTCHK	MQIA_ADOPTNEWMCA_CHECK	[z/OS]
AdoptNewMCAType	ADOPTMCA	MQIA_ADOPTNEWMCA_TYPE	[z/OS]

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
AlterationDate	ALTDATE	MQCA_ALTERATION_DATE	
AlterationTime	ALTTIME	MQCA_ALTERATION_TIME	
AuthorityEvent	AUTHOREV	MQIA_AUTHORITY_EVENT	
BridgeEvent	BRIDGEEV	MQIA_BRIDGE_EVENT	[z/OS]
CodedCharSetId	CCSID	MQIA_CODED_CHAR_SET_ID	
CertificateLabel	CERTLABL	MQCA_CERT_LABEL	
QSGCertificateLabel	CERTQSGL	MQCA_QSG_CERT_LABEL	[z/OS]
CertificateValPolicy	CERTVPOL	MQIA_CERT_VAL_POLICY	
CFConlos	CFCONLOS	MQIA_QMGR_CFCONLOS	[z/OS]
ChannelAutoDef	CHAD	MQIA_CHANNEL_AUTO_DEF	
ChannelAutoDefEvent	CHADEV	MQIA_CHANNEL_AUTO_DEF_EVENT	
ChannelAutoDefExit	CHADEXIT	MQCA_CHANNEL_AUTO_DEF_EXIT	
ChinitAdapters	CHIADAPS	MQIA_CHINIT_ADAPTERS	[z/OS]
ChinitDispatchers	CHIDISPS	MQIA_CHINIT_DISPATCHERS	[z/OS]
ChinitServiceParm	CHISERVP	MQCA_CHINIT_SERVICE_PARM	[z/OS]
ChannelAuthenticationRecords	CHLAUTH	MQIA_CHLAUTH_RECORDS	
ChannelEvent	CHLEV	MQIA_CHANNEL_EVENT	
ClusterWorkLoadExit	CLWLEXIT	MQCA_CLUSTER_WORKLOAD_EXIT	
ClusterWorkLoadData	CLWLDATA	MQCA_CLUSTER_WORKLOAD_DATA	
ClusterWorkLoadLength	CLWLLEN	MQIA_CLUSTER_WORKLOAD_LENGTH	
CLWLMRUChannels	CLWLMRUC	MQIA_CLWL_MRU_CHANNELS	

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
CLWLUseQ	CLWLUSEQ	MQIA_CLWL_USEQ	
CommandEvent	CMDEV	MQIA_COMMAND_EVENT	
CommandLevel	CMDLEVEL	MQIA_COMMAND_LEVEL	
CommandInputQName	COMMANDQ	MQCA_COMMAND_INPUT_Q_NAME	
ConfigurationEvent	CONFIGEV	MQIA_CONFIGURATION_EVENT	
ConnAuth	CONNAUTH	MQCA_CONN_AUTH	
CreationDate	CRDATE	MQCA_CREATION_DATE	
CreationTime	CRTIME	MQCA_CREATION_TIME	
Custom	CUSTOM	MQCA_CUSTOM	
DeadLetterQName	DEADQ	MQCA_DEAD_LETTER_Q_NAME	
DefClusterXmitQueueType	DEFCLXQ	MQIA_DEF_CLUSTER_XMIT_Q_TYPE	
DefXmitQName	DEFXMITQ	MQCA_DEF_XMIT_Q_NAME	
QMgrDesc	DESCR	MQCA_Q_MGR_DESC	
DistLists	DISTL	MQIA_DIST_LISTS	
DNSGroup	DNSGROUP	MQCA_DNS_GROUP	[z/OS]
DNSWLM	DNSWLM	MQIA_DNS_WLM	[z/OS]
ExpiryInterval	EXPRYINT	MQIA_EXPIRY_INTERVAL	[z/OS]
GroupUR	GROUPUR	MQIA_GROUP_UR	[z/OS]
IntraGroupqueuing	IGQ	MQIA_INTRA_GROUP_QUEUING	[z/OS]
IGQPutAuthority	IGQAUT	MQIA_IGQ_PUT_AUTHORITY	[z/OS]
IGQUserId	IGQUSER	MQCA_IGQ_USER_ID	[z/OS]
InhibitEvent	INHIBTEV	MQIA_INHIBIT_EVENT	
IPAddressVersion	IPADDRV	MQIA_IP_ADDRESS_VERSION	

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
LocalEvent	LOCALEV	MQIA_LOCAL_EVENT	[z/OS]
LoggerEvent	LOGGEREV	MQIA_LOGGER_EVENT	
ListenerTimer	LSTRTMR	MQIA_LISTENER_TIMER	
LUGroupName	LUGROUP	MQCA_LU_GROUP_NAME	[z/OS]
LUName	LUNAME	MQCA_LU_NAME	[z/OS]
LU62ARMSuffix	LU62ARM	MQCA_LU62_ARM_SUFFIX	[z/OS]
LU62Channels	LU62CHL	MQIA_LU62_CHANNELS	[z/OS]
MsgMarkBrowseInterval	MARKINT	MQIA_MSG_MARK_BROWSE_INTERVAL	
MaxChannels	MAXCHL	MQIA_MAX_CHANNELS	[z/OS]
MaxHandles	MAXHANDS	MQIA_MAX_HANDLES	
MaxMsgLength	MAXMSGL	MQIA_MAX_MSG_LENGTH	
MaxPropertiesLength	MAXPROPL	MQIA_MAX_PROPERTIES_LENGTH	
MaxPriority	MAXPRTY	MQIA_MAX_PRIORITY	
MaxUncommittedMsgs	MAXUMSGS	MQIA_MAX_UNCOMMITTED_MSGS	
ClusterSenderMonitoringDefault	MONACLS	MQIA_MONITORING_AUTO_CLUSSDR	
ChannelMonitoring	MONCHL	MQIA_MONITORING_CHANNEL	
QueueMonitoring	MONQ	MQIA_MONITORING_Q	
OutboundPortMax	OPORTMAX	MQIA_OUTBOUND_PORT_MAX	[z/OS]
OutboundPortMin	OPORTMIN	MQIA_OUTBOUND_PORT_MIN	[z/OS]
Parent	PARENT	MQCA_PARENT	
PerformanceEvent	PERFMEV	MQIA_PERFORMANCE_EVENT	
Platform	PLATFORM	MQIA_PLATFORM	

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
PubSubClus	PSCLUS	MQIA_PUBSUB_CLUSTER	
PubSubMode	PSMODE	MQIA_PUBSUB_MODE	
PubSubNPInputMsg	PSNPMSG	MQIA_PUBSUB_NP_MSG	
PubSubNPResponse	PSNPRES	MQIA_PUBSUB_NP_RESP	
PubSubMaxMsgRetryCount	PSRTYCNT	MQIA_PUBSUB_MAXMSG_RETRY_COUNT	
PubSubSyncPoint	PSSYNCPT	MQIA_PUBSUB_SYNC_PT	
QMgrIdentifier	QMID	MQCA_Q_MGR_IDENTIFIER	
QSGName	QSGNAME	MQCA_QSG_NAME	[z/OS]
ReceiveTimeout	RCVTIME	MQIA_RECEIVE_TIMEOUT	[z/OS]
ReceiveTimeoutMin	RCVTMIN	MQIA_RECEIVE_TIMEOUT_MIN	[z/OS]
ReceiveTimeoutType	RCVTTYPE	MQIA_RECEIVE_TIMEOUT_TYPE	[z/OS]
RemoteEvent	REMOTEEV	MQIA_REMOTE_EVENT	
RepositoryName	REPOS	MQCA_REPOSITORY_NAME	
RepositoryNamelist	REPOSNL	MQCA_REPOSITORY_NAMELIST	
RevDns	REVDNS	MQIA_REVERSE_DNS_LOOKUP	
TraceRouteRecording	ROUTEREC	MQIA_TRACE_ROUTE_RECORDING	
ChannelInitiatorControl	SCHINIT	MQIA_CHINIT_CONTROL	
CommandServerControl	SCMDSERV	MQIA_CMD_SERVER_CONTROL	
SecurityCase	SCYCASE	MQIA_SECURITY_CASE	[z/OS]
Splcap	SPLCAP	MQIA_PROT_POLICY_CAPABILITY	
SharedQQmgrName	SQQMNAME	MQIA_SHARED_Q_Q_MGR_NAME	[z/OS]

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
SSLCRLNamelist	SSLCRLNL	MQCA_SSL_CRL_NAMELIST	
SSLCryptoHardware	SSLCRYP	MQCA_SSL_CRYPTO_HARDWARE	
SSLEvent	SSLEV	MQIA_SSL_EVENT	
SSLFipsRequired	SSLFIPS	MQIA_SSL_FIPS_REQUIRED	
SSLKeyRepository	SSLKEYR	MQCA_SSL_KEY_REPOSITORY	
SSLKeyResetCount	SSLRKEYC	MQIA_SSL_RESET_COUNT	[z/OS]
SSLTasks	SSLTASKS	MQIA_SSL_TASKS	
ClusterSenderStatistics	STATACLS	MQIA_STATISTICS_AUTO_CLUSSDR	
ChannelStatistics	STATCHL	MQIA_STATISTICS_CHANNEL	
StatisticsInterval	STATINT	MQIA_STATISTICS_INTERVAL	
MQIStatistics	STATMQI	MQIA_STATISTICS_MQI	
QueueStatistics	STATQ	MQIA_STATISTICS_Q	
StartStopEvent	STRSTPEV	MQIA_START_STOP_EVENT	
EncryptionPolicySuiteB	SUITEB	MQIA_SUITE_B_STRENGTH	
SyncPoint	SYNCPT	MQIA_SYNCPOINT	
TCPChannels	TCPCHL	MQIA_TCP_CHANNELS	[z/OS]
TCPKeepAlive	TCPKEEP	MQIA_TCP_KEEP_ALIVE	[z/OS]
TCPName	TCPNAME	MQCA_TCP_NAME	[z/OS]
TCPStackType	TCPSTACK	MQIA_TCP_STACK_TYPE	
ChinitTraceAutoStart	TRAXSTR	MQIA_CHINIT_TRACE_AUTO_START	[z/OS]
ChinitTraceTableSize	TRAXTBL	MQIA_CHINIT_TRACE_TABLE_SIZE	[z/OS]
TreeLifeTime	TREELIFE	MQIA_TREE_LIFE_TIME	

Table 4.1 Parameters returned by qmgr command

Response Parameter	Display Name	Parameter ID	Note
TriggerInterval	TRIGINT	MQIA_TRIGGER_INTERVAL	
Version	VERSION	MQCA_VERSION	
XrCapability	XRCAP	MQIA_XR_CAPABILITY	

Ex. 4.1 qmgr command

< Display all parameters. >

mqpcf qmgr -qm TESTQM

1: QMNAME(TESTQM) ACCTCONO(DISABLED) ACCTINT(1800) ACCTMQI(OFF) ACCTQ(OFF) ACTIVREC(268438660) ACTVCONO(268441232) ACTVTRC(OFF) ALTDATE(2016-12-13) ALTTIME(14.19.03) AUTHOREV(DISABLED) CCSID(943) CERTLABL(ibmwebspheremqtestqm) CERTVPOL(ANY) CHAD(DISABLED) CHADEV(DISABLED) CHADEXIT() CHLAUTH(DISABLED) CHLEV(DISABLED) CLWLEXIT() CLWLDATA() CLWLLEN(100) CLWLMRUC(999999999) CLWLUSEQ(LOCAL) CMDEV(DISABLED) CMDLEVEL(800) COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) CONFIGEV(DISABLED) CONNAUTH() CRDATE(2014-09-22) CRTIME(07.21.41) CUSTOM() DEADQ(SYSTEM.DEAD.LETTER.QUEUE) DEFCLXQ(SCTQ) DEFXMITQ() DESCR() DISTL(YES) INHIBTEV(DISABLED) IPADDRV(IPV4) LOCALEV(DISABLED) LOGGEREV(DISABLED) MARKINT(5000) MAXHANDS(256) MAXMSGL(4194304) MAXPROPL(-1) MAXPRTY(9) MAXUMSGS(10000) MONACLS(QMGR) MONCHL(OFF) MONQ(OFF) PARENT() PERFMEV(DISABLED) PLATFORM(AIX) PSCLUS(ENABLED) PSMODE(ENABLED) PSNPMSG(DISCARD) PSNPRES(NORMAL) PSRTYCNT(5) PSSYNCPT(IFPER) QMID(TESTQM_2014-09-22_07.21.41) REMOTEEV(DISABLED) REPOS(REP80) REPOSNL() REVDNS(ENABLED) ROUTEREC(MSG) SCHINIT(QMGR) SCMDSERV(QMGR) SPLCAP(YES) SSLCRLNL() SSLCRYP() SSLEV(DISABLED) SSLFIPS(NO) SSLKEYR(/var/mqm/qmgrs/TESTQM/ssl/key) SSLRKEYC(0) STATACLS(QMGR) STATCHL(OFF) STATINT(30) STATMQI(OFF) STATQ(OFF) STRSTPEV(ENABLED) SUITEB(128_BIT,192_BIT) SYNCPT(YES) TREELIFE(1800) TRIGINT(999999999) VERSION(08000000) XRCAP(YES)

< Reduce a number of display parameters. >

* As an example, only the states of three parameters MONACLS, MONCHL, and

MONQ are displayed.

$ mqpcf qmgr -qm TESTQM MONACLS MONCHL MONQ
1: QMNAME(TESTQM) **MONACLS(QMGR) MONCHL(OFF) MONQ(OFF)**

* Since QMNAME is a required parameter, it is displayed even if it is not specified.

Inquire Queue Manager Status (qms)

The qms command execute MQCMD_INQUIRE_Q_MGR_STATUS MQAI command. It is equivalent to "display qms" of the runmqsc command.

mqpcf qms -qm Qmgr

Parameters in the table below are displayed.

Table 4.2 Parameters returned by qms command.

Response Parameter	Display Name	Parameter ID	Note
QMgrName	QMNAME	MQCA_Q_MGR_NAME	Mandatory
QMgrStatus	STATUS	MQIACF_Q_MGR_STATUS	Mandatory
ChannelInitiatorStatus	CHINIT	MQIACF_CHINIT_STATUS	
CommandServerStatus	CMDSERV	MQIACF_CMD_SERVER_STATUS	
ConnectionCount	CONNS	MQIACF_CONNECTION_COUNT	
CurrentLog	CURRLOG	MQCACF_CURRENT_LOG_EXTENT_NAME	
InstallationDesc	INSTDESC	MQCA_INSTALLATION_DESC	
InstallationName	INSTNAME	MQCA_INSTALLATION_NAME	
InstallationPath	INSTPATH	MQCA_INSTALLATION_PATH	
LDAPConnectionStatus	LDAPCONN	MQIACF_LDAP_CONNECTION_STATUS	
LogPath	LOGPATH	MQCACF_LOG_PATH	
MediaRecoveryLog	MEDIALOG	MQCACF_MEDIA_LOG_EXTENT_NAME	
RestartRecoveryLog	RECLOG	MQCACF_RESTART_LOG_EXTENT_NAME	
StartDate	STARTDA	MQCACF_Q_MGR_START_DATE	
StartTime	STARTTI	MQCACF_Q_MGR_START_TIME	

Ex. 4.2 qms command

```
--------------------------------------
$ mqpcf qms -qm TESTQM
1: QMNAME(TESTQM) STATUS(RUNNING) CHINIT(RUNNING)
CMDSERV(RUNNING) CONNS(22) CURRLOG() INSTDESC()
INSTNAME(Installation5) INSTPATH(/usr/mqm-mq8000gm/usr/mqm)
LDAPCONN(INACTIVE) LOGPATH(/var/mqm/log/TESTQM/active/) MEDIALOG()
RECLOG() STARTDA(2017-01-20) STARTTI(18.39.04)
--------------------------------------
```

Inquire Queue (que)

The que command execute MQCMD_INQUIRE_Q MQAI command. It is equiv alent to "display queue" of the runmqsc command.

mqpcf que -qm Qmgr [-q Queue]

Parameters in the table below are displayed.

Table 4.3 Parameters returned by que command.

Response Parameter	Display Name	Parameter ID	Note
QName	QUEUE	MQCA_Q_NAME	Mandatory
QSGDisposition	QSGDISP	MQIA_QSG_DISP	Mandatory [z/OS]
QType	TYPE	MQIA_Q_TYPE	Mandatory
QueueAccounting	ACCTQ	MQIA_ACCOUNTING_Q	
AlterationDate	ALTDATE	MQCA_ALTERATION_DATE	
AlterationTime	ALTTIME	MQCA_ALTERATION_TIME	
BackoutRequeueName	BOQNAME	MQCA_BACKOUT_REQ_Q_ NAME	
BackoutThreshold	BOTHRESH	MQIA_BACKOUT_THRESH OLD	
CFStructure	CFSTRUCT	MQCA_CF_STRUC_NAME	[z/OS]
ClusterChannelName	CLCHNAME	MQCA_CLUS_CHL_NAME	
ClusterDate	CLUSDATE	MQCA_CLUSTER_DATE	
ClusterNamelist	CLUSNL	MQCA_CLUSTER_NAMELIS T	
QMgrName	CLUSQMGR	MQCA_CLUSTER_Q_MGR_ NAME	
ClusterQType	CLUSQT	MQIA_CLUSTER_Q_TYPE	
ClusterName	CLUSTER	MQCA_CLUSTER_NAME	
ClusterTime	CLUSTIME	MQCA_CLUSTER_TIME	
CLWLQueuePriority	CLWLPRTY	MQIA_CLWL_Q_PRIORITY	

Table 4.3 Parameters returned by que command.

Response Parameter	Display Name	Parameter ID	Note
CLWLQueueRank	CLWLRANK	MQIA_CLWL_Q_RANK	
CLWLUseQ	CLWLUSEQ	MQIA_CLWL_USEQ	
CreationDate	CRDATE	MQCA_CREATION_DATE	
CreationTime	CRTIME	MQCA_CREATION_TIME	
CurrentQDepth	CURDEPTH	MQIA_CURRENT_Q_DEPTH	
Custom	CUSTOM	MQCA_CUSTOM	
DefBind	DEFBIND	MQIA_DEF_BIND	
DefInputOpenOption	DEFSOPT	MQIA_DEF_INPUT_OPEN_OPTION	
DefaultPutResponse	DEFPRESP	MQIA_DEF_PUT_RESPONSE_TYPE	
DefPriority	DEFPRTY	MQIA_DEF_PRIORITY	
DefPersistence	DEFPSIST	MQIA_DEF_PERSISTENCE	
DefReadAhead	DEFREADA	MQIA_DEF_READ_AHEAD	
DefinitionType	DEFTYPE	MQIA_DEFINITION_TYPE	[z/OS]
QDesc	DESCR	MQCA_Q_DESC	
DistLists	DISTL	MQIA_DIST_LISTS	
InhibitGet	GET	MQIA_INHIBIT_GET	
HardenGetBackout	HERDENBO	MQIA_HARDEN_GET_BACKOUT	
IndexType	INDXTYPE	MQIA_INDEX_TYPE	[z/OS]
InitiationQName	INITQ	MQCA_INITIATION_Q_NAME	
OpenInputCount	IPPROCS	MQIA_OPEN_INPUT_COUNT	
MaxQDepth	MAXDEPTH	MQIA_MAX_Q_DEPTH	
MaxMsgLength	MAXMSGL	MQIA_MAX_MSG_LENGTH	
QueueMonitoring	MONQ	MQIA_MONITORING_Q	

Table 4.3 Parameters returned by que command.

Response Parameter	Display Name	Parameter ID	Note
MsgDeliverySequence	MSGDLVSQ	MQIA_MSG_DELIVERY_SEQUENCE	
NonPersistentMessageClass	NPMCLASS	MQIA_NPM_CLASS	
OpenOutputCount	OPPROCS	MQIA_OPEN_OUTPUT_COUNT	
ProcessName	PROCESS	MQCA_PROCESS_NAME	[z/OS]
PropertyControl	PROPCTL	MQIA_PROPERTY_CONTROL	
PageSetID	PSID	MQIA_PAGESET_ID	
InhibitPut	PUT	MQIA_INHIBIT_PUT	
QDepthHighLimit	QDEPTHHI	MQIA_Q_DEPTH_HIGH_LIMIT	
QDepthLowLimit	QDEPTHLO	MQIA_Q_DEPTH_LOW_LIMIT	
QDepthHighEvent	QDPHIEV	MQIA_Q_DEPTH_HIGH_EVENT	
QDepthLowEvent	QDPLOEV	MQIA_Q_DEPTH_LOW_EVENT	
QDepthMaxEvent	QDPMAXEV	MQIA_Q_DEPTH_MAX_EVENT	
QMgrIdentifier	QMID	MQCA_Q_MGR_IDENTIFIER	
QServiceIntervalEvent	QSVCIEV	MQIA_Q_SERVICE_INTERVAL_EVENT	
QServiceInterval	QSVCINT	MQIA_Q_SERVICE_INTERVAL	
RetentionInterval	RETINTVL	MQIA_RETENTION_INTERVAL	
RemoteQName	RNAME	MQCA_REMOTE_Q_NAME	
RemoteQMgrName	RQMNAME	MQCA_REMOTE_Q_MGR_N	

Table 4.3 Parameters returned by que command.

Response Parameter	Display Name	Parameter ID	Note
		AME	
Shareability	SHRBLTY	MQIA_SHAREABILITY	
QueueStatistics	STATQ	MQIA_STATISTICS_Q	
StorageClass	STGCLASS	MQCA_STORAGE_CLASS	[z/OS]
BaseQName	TARGET	MQCA_BASE_Q_NAME	
TpipeNames	TPIPE	MQCA_TPIPE_NAME	[z/OS]
TriggerControl	TRIGCTRL	MQIA_TRIGGER_CONTROL	
TriggerData	TRIGDATA	MQCA_TRIGGER_DATA	
TriggerDepth	TRIGDPTH	MQIA_TRIGGER_DEPTH	
TriggerMsgPriority	TRIGMPRI	MQIA_TRIGGER_MSG_PRIORITY	
TriggerType	TRIGTYPE	MQIA_TRIGGER_TYPE	
Usage	USAGE	MQIA_USAGE	
XmitQName	XMITQ	MQCA_XMIT_Q_NAME	

Ex. 4.3 que command

$ mqpcf que -qm TESTQM **-q TQ**

1: QUEUE(TQ) TYPE(QLOCAL) ACCTQ(QMGR) ALTDATE(2017-01-12) ALTTIME(09.31.57) BOQNAME(BO4TQ) BOTHRESH(0) CLCHNAME() CLUSNL() CLUSTER() CLWLPRTY(0) CLWLRANK(0) CLWLUSEQ(QMGR) CRDATE(2014-09-26) CRTIME(08.36.28) CURDEPTH(0) CUSTOM() DEFBIND(OPEN) DEFSOPT (SHARED) DEFPRESP(SYNC) DEFPRTY(0) DEFPSIST(NO) DEFREADA(NO) DEFTYPE(PREDEFINED) DESCR() DISTL(YES) GET(ENABLED) HERDENBO(HARDENBO) INITQ() IPPROCS(0) MAXDEPTH(5000) MAXMSGL(4194304) MONQ(HIGH) MSGDLVSQ(PRIORITY) NPMCLASS(NORMAL) OPPROCS(0) PROCESS() PROPCTL(COMPAT) PUT(ENABLED) QDEPTHHI(80) QDEPTHLO(20) QDPHIEV(DISABLED) QDPLOEV(DISABLED) QDPMAXEV(ENABLED) QSVCIEV(NONE) QSVCINT(999999999) RETINTVL(999999999) SHRBLTY(SHARE) STATQ(ON) TRIGCTRL(NOTRIGGER) TRIGDATA() TRIGDPTH(1) TRIGMPRI(0)

TRIGTYPE(FIRST) USAGE(NORMAL)

$ mqpcf que -qm TESTQM **CURDEPTH**
1: QUEUE(ATQ) TYPE(QALIAS)
2: QUEUE(AUTTQ1) TYPE(QLOCAL) **CURDEPTH(0)**
3: QUEUE(CICS.LOCAL.QUEUE) TYPE(QLOCAL) **CURDEPTH(0)**
....

$ mqpcf que -qm TESTQM -q "SYSTEM.*" TYPE
1: QUEUE(SYSTEM.ADMIN.ACCOUNTING.QUEUE) TYPE(QLOCAL)
2: QUEUE(SYSTEM.ADMIN.ACTIVITY.QUEUE) TYPE(QLOCAL)
3: QUEUE(SYSTEM.ADMIN.CHANNEL.EVENT) TYPE(QLOCAL)
....
62: QUEUE(SYSTEM.SELECTION.VALIDATION.QUEUE) TYPE(QLOCAL)

Inquire Queue (cque)

The cque command execute MQCMD_INQUIRE_Q MQAI command with MQ QT_CLUSTER queue type parameter. It is equivalent to "display qcluster" of t he runmqsc command.

mqpcf cque -qm Qmgr [-q Queue] [-cl Cluster]

For parameters returned by the cque command, see "Table 4.4 parameters re turned by the que command".

Ex. 4.4 cque command

```
---------------------------------------
$ mqpcf cque -qm TESTQM -q CQ1A
1: QUEUE(CQ1A) TYPE(QCLUSTER) ALTDATE(2016-05-20) ALTTIME(16.33.56)
CLUSDATE(2016-05-20) CLUSQMGR(TESTQM) CLUSQT(QLOCAL)
CLUSTER(REP80) CLUSTIME(16.33.56) CLWLPRTY(0) CLWLRANK(0)
DEFBIND(OPEN) DEFPRESP(SYNC) DEFPRTY(0) DEFPSIST(NO) DESCR()
PUT(ENABLED) QMID(TESTQM_2014-09-22_07.21.41)


$ mqpcf cque -qm TESTQMA -q CQ1 CLUSTER CLUSQMGR
1: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(TESTQMC) CLUSTER(REP80)
2: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(TESTQMB) CLUSTER(REP80)
3: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(TESTQMA) CLUSTER(REP80)


$ mqpcf cque -qm TESTQM -cl "REP8*" CLUSTER
1: QUEUE(CQ1) TYPE(QCLUSTER) CLUSTER(REP80)
2: QUEUE(CQ1) TYPE(QCLUSTER) CLUSTER(REP80)
3: QUEUE(CQ1) TYPE(QCLUSTER) CLUSTER(REP80)
....
12: QUEUE(CQDUMMY) TYPE(QCLUSTER) CLUSTER(REP80B)

$ mqpcf cque -qm TESTQM -cl "REP8B" CLUSTER
1: QUEUE(CQDUMMY) TYPE(QCLUSTER) CLUSTER(REP80B)
---------------------------------------
```

Inquire Queue Status (ques)

The ques command execute MQCMD_INQUIRE_Q_STATUS MQAI command with MQIACF_Q_STATUS of the default value of MQIACF_Q_STATUS_TYPE. It is equivalent to "display qstatus" of the runmqsc command.

mqpcf ques -qm Qmgr [-q Queue]

Parameters in the table below are displayed.

Table 4.4 Parameters returned by ques command.

Response Parameter	Display Name	Parameter ID	Note
QName	QUEUE	MQCA_Q_NAME	Mandatory
QSGDisposition	QSGDISP	MQIA_QSG_DISP	Mandatory [z/OS]
StatusType	TYPE	MQIACF_Q_STATUS_TYPE	
CurrentQDepth	CURDEPTH	MQIA_CURRENT_Q_DEPTH	
OpenInputCount	IPPROCS	MQIA_OPEN_INPUT_COUNT	
LastGetDate	LGETDATE	MQCACF_LAST_GET_DATE	
LastGetTime	LGETTIME	MQCACF_LAST_GET_TIME	
LastPutTime	LPUTDATE	MQCACF_LAST_PUT_TIME	
QueueMonitoring	MONQ	MQIA_MONITORING_Q	
OldestMsgAge	MSGAGE	MQIACF_OLDEST_MSG_AGE	
OpenOutputCount	OPPROCS	MQIA_OPEN_OUTPUT_COUNT	
OnQTime	QTIME	MQIACF_Q_TIME_INDICATOR	
UncommittedMsgs	UCOM	MQIACF_UNCOMMITTED_MSGS	

Ex. 4.5 ques command

```
------------------------------------
$ mqpcf ques -qm TESTQM -q CICS.TRIG.QUEUE
1: QUEUE(CICS.TRIG.QUEUE) TYPE(QUEUE) CURDEPTH(0) IPPROCS(0)
LGETDATE() LGETTIME() LPUTDATE() LPUTTIME() MONQ(OFF) MSGAGE( )
OPPROCS(0) QTIME(-1, -1) UCOM(0)
------------------------------------
```

Inquire Queue Status (queh)

The queh command execute MQCMD_INQUIRE_Q_STATUS MQAI command with MQIACF_Q_HANDLE of the default value of MQIACF_Q_STATUS_TYPE. It is equivalent to "display qstatus type(handle)" of the runmqsc command.

mqpcf queh -qm Qmgr [-q Queue]

Parameters in the table below are displayed.

Table 4.5 Parameters returned by queh command.

Response Parameter	Display Name	Parameter ID	Note
QName	QUEUE	MQCA_Q_NAME	Mandatory
QSGDisposition	QSGDISP	MQIA_QSG_DISP	Mandatory [z/OS]
StatusType	TYPE	MQIACF_Q_STATUS_TYPE	
ApplDesc	APPLDESC	MQCACF_APPL_DESC	
ApplTag	APPLTAG	MQCACF_APPL_TAG	
ApplType	APPLTYPE	MQIA_APPL_TYPE	
ASId	ASID	MQCACF_ASID	[z/OS]
AsynchronousState	ASTATE	MQIACF_ASYNC_STATE	
OpenBrowse	BROWSE	MQIACF_OPEN_BROWSE	
ChannelName	CHANNEL	MQCACH_CHANNEL_NAME	
Conname	CONNAME	MQCACH_CONNECTION_NAME	
HandleState	HSTATE	MQIACF_HANDLE_STATE	
OpenInputType	INPUT	MQIACF_OPEN_INPUT_TYPE	
OpenInquire	INQUIRE	MQIACF_OPEN_INQUIRE	
OpenOptions	OPENOPTS	MQIACF_OPEN_OPTIONS	
OpenOutput	OUTPUT	MQIACF_OPEN_OUTPUT	
ProcessId	PID	MQIACF_PROCESS_ID	
PSBName	PSBNAME	MQCACF_PSB_NAME	[z/OS]

Table 4.5 Parameters returned by queh command.			
Response Parameter	**Display Name**	**Parameter ID**	**Note**
PSTId	PSTID	MQCACF_PST_ID	[z/OS]
QMgrUOWId	QMURID	MQBACF_Q_MGR_UOW_ID	
OpenSet	SET	MQIACF_OPEN_SET	
TaskNumber	TASKNO	MQCACF_TASK_NUMBER	[z/OS]
ThreadId	TID	MQIACF_THREAD_ID	[z/OS]
TransactionId	TRANSID	MQCACF_TRANSACTION_ID	
ExternalUOWId	URID	MQBACF_EXTERNAL_UOW_ID	[z/OS]
UOWType	URTYPE	MQIACF_UOW_TYPE	
UserIdentifier	USERID	MQCACF_USER_IDENTIFIER	

Ex. 4.6 queh command

```
-------------------------------------
$ mqpcf queh -qm TESTQM -q CICS.LOCAL.QUEUE
1: QUEUE(CICS.LOCAL.QUEUE) TYPE(HANDLE) APPLDESC() APPLTAG(mqpgf)
APPLTYPE(USER) ASTATE(NONE) BROWSE(NO) CHANNEL() CONNAME()
HSTATE(INACTIVE) INPUT(SHARED) INQUIRE(NO)
OPENOPTS(MQOO_INPUT_SHARED) OUTPUT(NO) PID(655366)
QMURID(000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000) SET(NO) TID(1)
URID(000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
```

00000000000000) URTYPE(QMGR) USERID(mq80)

$ mqpcf queh -qm TESTQM APPLTAG OPENOPTS PID
1: QUEUE(TQ) **TYPE(HANDLE) APPLTAG(mqpgf)**
OPENOPTS(MQOO_INPUT_SHARED) PID(2818152)
2: QUEUE(CICS.LOCAL.QUEUE) **TYPE(HANDLE) APPLTAG(mqpgf)**
OPENOPTS(MQOO_INPUT_SHARED) PID(655366)

Inquire Channel (chl)

The chl command execute MQCMD_INQUIRE_CHANNEL MQAI command. It is equivalent to "display channel" of the runmqsc command.

mqpcf chl -qm Qmgr [-c Channel]

Parameters in the table below are displayed.

Table 4.6 Parameters returned by chl command.

Response Parameter	Display Name	Parameter ID	Note
ChannelName	CHANNEL	MQCACH_CHANNEL_NAME	Mandatory
ChannelType	CHLTYPE	MQIACH_CHANNEL_TYPE	Mandatory
DefaultChannelDisposition	DEFCDISP	MQIACH_CHANNEL_DISP	Mandatory [z/OS]
QSGDisposition	QSGDISP	MQIA_QSG_DISP	Mandatory [z/OS]
ConnectionAffinity	AFFINITY	MQIACH_CONNECTION_AFFINITY	
AlterationDate	ALTDATE	MQCA_ALTERATION_DATE	
AlterationTime	ALTTIME	MQCA_ALTERATION_TIME	
BatchDataLimit	BATCHLIM	MQIACH_BATCH_DATA_LIMIT	
BatchHeartbeat	BATCHHB	MQIACH_BATCH_HB	
BatchInterval	BATCHINT	MQIACH_BATCH_INTERVAL	
BatchSize	BATCHSZ	MQIACH_BATCH_SIZE	
CertificateLabel	CERTLABL	MQCA_CERT_LABEL	
ClientChannelWeight	CLNTWGHT	MQIACH_CLIENT_CHANNEL_WEIGHT	
ClusterNamelist	CLUSNL	MQCA_CLUSTER_NAMELIST	
ClusterName	CLUSTER	MQCA_CLUSTER_NAME	

Table 4.6 Parameters returned by chl command.

Response Parameter	Display Name	Parameter ID	Note
CLWLChannelPriority	CLWLPRTY	MQIACH_CLWL_CHANNEL_PRIORITY	
CLWLChannelRank	CLWLRANK	MQIACH_CLWL_CHANNEL_RANK	
CLWLChannelWeight	CLWLWGHT	MQIACH_CLWL_CHANNEL_WEIGHT	
HeaderCompression	COMPHDR	MQIACH_HDR_COMPRESSION	
MessageCompression	COMPMSG	MQIACH_MSG_COMPRESSION	
ConnectionName	CONNAME	MQCACH_CONNECTION_NAME	
DataConversion	CONVERT	MQIACH_DATA_CONVERSION	
ChannelDesc	DESCR	MQCACH_DESC	
DiscInterval	DISCINT	MQIACH_DISC_INTERVAL	
HeartbeatInterval	HBINT	MQIACH_HB_INTERVAL	
KeepAliveInterval	KAINT	MQIACH_KEEP_ALIVE_INTERVAL	
LocalAddress	LOCLADDR	MQCACH_LOCAL_ADDRESS	
LongRetryCount	LONGRTY	MQIACH_LONG_RETRY	
LongRetryInterval	LONGTMR	MQIACH_LONG_TIMER	
MaxInstances	MAXINST	MQIACH_MAX_INSTANCES	
MaxInstancesPerClient	MAXINSTC	MQIACH_MAX_INSTS_PER_CLIENT	
MaxMsgLength	MAXMSGL	MQIACH_MAX_MSG_LENGTH	
MCAName	MCANAME	MQCACH_MCA_NAME	
MCAType	MCATYPE	MQIACH_MCA_TYPE	
MCAUserIdentifier	MCAUSER	MQCACH_MCA_USER_ID	
ModeName	MODENAME	MQCACH_MODE_NAME	
ChannelMonitoring	MONCHL	MQIA_MONITORING_CHANNEL	

Table 4.6 Parameters returned by chl command.

Response Parameter	Display Name	Parameter ID	Note
MsgRetryUserData	MRDATA	MQCACH_MR_EXIT_USER_DATA	
MsgRetryExit	MREXIT	MQCACH_MR_EXIT_NAME	
MsgRetryCount	MRRTY	MQIACH_MR_COUNT	
MsgRetryInterval	MRTMR	MQIACH_MR_INTERVAL	
MsgUserData	MSGDATA	MQCACH_MSG_EXIT_USER_DATA	
MsgExit	MSGEXIT	MQCACH_MSG_EXIT_NAME	
NetworkPriority	NETPRTY	MQIACH_NETWORK_PRIORITY	
NonPersistentMsgSpeed	NPMSPEED	MQIACH_NPM_SPEED	
Password	PASSWORD	MQCACH_PASSWORD	
PropertyControl	PROPCTL	MQIA_PROPERTY_CONTROL	
PutAuthority	PUTAUT	MQIACH_PUT_AUTHORITY	
QMgrName	QMNAME	MQCA_Q_MGR_NAME	
ReceiveUserData	RCVDATA	MQCACH_RCV_EXIT_USER_DATA	
ReceiveExit	RCVEXIT	MQCACH_RCV_EXIT_NAME	
SecurityUserData	SCYDATA	MQCACH_SEC_EXIT_USER_DATA	
SendExit	SENDEXIT	MQCACH_SEND_EXIT_NAME	
MsgsSent	SENTMSGS	MQIACH_MSGS_SENT	
SeqNumberWrap	SEQWRAP	MQIACH_SEQUENCE_NUMBER_WRAP	
SharingConversations	SHARECNV	MQIACH_SHARING_CONVERSATIONS	
ShortRetryCount	SHORTRTY	MQIACH_SHORT_RETRY	
ShortRetryInterval	SHORTTMR	MQIACH_SHORT_TIMER	

Table 4.6 Parameters returned by chl command.

Response Parameter	Display Name	Parameter ID	Note
SSLClientAuth	SSLCAUTH	MQIACH_SSL_CLIENT_AUTH	
SSLCipherSpec	SSLCIPH	MQCACH_SSL_CIPHER_SPEC	
SSLPeerName	SSLPEER	MQCACH_SSL_PEER_NAME	
SSLCipherSuite	SSLSUITE	MQCACH_SSL_CIPHER_SUITE	
ChannelStatistics	STATCHL	MQIA_STATISTICS_CHANNEL	
TpName	TPNAME	MQCACH_TP_NAME	
TransportType	TRPTYPE	MQIACH_XMIT_PROTOCOL_TYPE	
UseDLQ	USEDLQ	MQIA_USE_DEAD_LETTER_Q	
UserIdentifier	USERID	MQCACH_USER_ID	
XmitQName	XMITQ	MQCACH_XMIT_Q_NAME	

Ex. 4.7 chl command

$ mqpcf chl -qm TESTQM **-c TO.kuipo**
1: CHANNEL(TO.kuipo) CHLTYPE(SDR) ALTDATE(2016-12-13) ALTTIME(13.57.36) BATCHHB(0) BATCHINT(0) BATCHSZ(50) CERTLABL(SampleQM) COMPHDR() COMPMSG() CONNAME(kuipo(1414)) CONVERT(NO) DESCR() DISCINT(6000) HBINT(300) KAINT(-1) LOCLADDR() LONGRTY(999999999) LONGTMR(1200) MAXMSGL(4194304) MCANAME() MCATYPE(PROCESS) MCAUSER() MODENAME() MONCHL(QMGR) MSGDATA() MSGEXIT() NPMSPEED(FAST) PASSWORD() PROPCTL(COMPAT) RCVDATA() RCVEXIT() SCYDATA() SCYEXIT() SENDDATA() SENDEXIT() SEQWRAP(999999999) SHORTRTY(10) SHORTTMR(60) SSLCIPH() SSLPEER() STATCHL(QMGR) TPNAME() TRPTYPE(TCP) USEDLQ(YES) USERID() XMITQ(kuipo)

$ mqpcf chl -qm TESTQM -c "TO.kuipo*" **HBINT DISCINT**
1: CHANNEL(TO.kuipo) CHLTYPE(SDR) **DISCINT(6000) HBINT(300)**
2: CHANNEL(TO.kuipo2) CHLTYPE(SDR) **DISCINT(6000) HBINT(300)**

Inquire Channel Status (chs)

The chs command execute MQCMD_INQUIRE_CHANNEL_STATUS MQAI command. It is equivalent to "display chstatus" of the runmqsc command.

mqpcf chs -qm Qmgr [-c Channel] [-cn Connection] [saved]

Parameters in the table below are displayed.

Table 4.7 Parameters returned by chs command.

Response Parameter	Display Name	Parameter ID	Note
ChannelDisposition	DEFCDISP	MQIACH_CHANNEL_DISP	Mandatory
ChannelInstanceType	CHLINSTYPE	MQIACH_CHANNEL_INSTANCE_TYPE	Mandatory
ChannelName	CHANNEL	MQCACH_CHANNEL_NAME	Mandatory
ChannelStatus	STATUS	MQIACH_CHANNEL_STATUS	Mandatory
ChannelType	CHLTYPE	MQIACH_CHANNEL_TYPE	Mandatory
ConnectionName	CONNAME	MQCACH_CONNECTION_NAME	Mandatory
RemoteApplTag	RAPPLTAG	MQCACH_REMOTE_APPL_TAG	Mandatory
RemoteQMgrName	RQMNAME	MQCA_REMOTE_Q_MGR_NAME	Mandatory
StopRequested	STOPREQ	MQIACH_STOP_REQUESTED	Mandatory
SubState	SUBSTATE	MQIACH_CHANNEL_SUBSTATE	Mandatory
XmitQName	XMITQ	MQCACH_XMIT_Q_NAME	Mandatory
Batches	BATCHES	MQIACH_BATCHES	
BatchSize	BATCHSZ	MQIACH_BATCH_SIZE	
BuffersReceived	BUFSRCVD	MQIACH_BUFFERS_RCVD	

Table 4.7 Parameters returned by chs command.

Response Parameter	Display Name	Parameter ID	Note
BuffersSent	BUFSSENT	MQIACH_BUFFERS_SENT	
BytesReceived	BYTSRCVD	MQIACH_BYTES_RCVD	
BytesSent	BYTSSENT	MQIACH_BYTES_SENT	
ChannelStartDate	CHSTADA	MQCACH_CHANNEL_START_DATE	
ChannelStartTime	CHSTATI	MQCACH_CHANNEL_START_TIME	
HeaderCompression	COMPHDR	MQIACH_HDR_COMPRESSION	
MessageCompression	COMPMSG	MQIACH_MSG_COMPRESSION	
CompressionRate	COMPRATE	MQIACH_COMPRESSION_RATE	
CompressionTime	COMPTIME	MQIACH_COMPRESSION_TIME	
CurrentLUWID	CURLUWID	MQCACH_CURRENT_LUWID	
CurrentMsgs	CURMSGS	MQIACH_CURRENT_MSGS	
CurrentSequenceNumber	CURSEQNO	MQIACH_CURRENT_SEQ_NUMBER	
CurrentSharingConversations	CURSHCNV	MQIACH_CURRENT_SHARING_CONVS	
ExitTime	EXITTIME	MQIACH_EXIT_TIME_INDICATOR	
HeartbeatInterval	HBINT	MQIACH_HB_INTERVAL	
InDoubtStatus	INDOUBT	MQIACH_INDOUBT_STATUS	
MCAJobName	JOBNAME	MQCACH_MCA_JOB_NAME	
KeepAliveInterval	KAINT	MQIACH_KEEP_ALIVE_INTERVAL	
LocalAddress	LOCLADDR	MQCACH_LOCAL_ADDRESS	
LongRetriesLeft	LONGRTS	MQIACH_LONG_RETRIES_LEFT	
LastLUWID	LSTLUWID	MQCACH_LAST_LUWID	
LastMsgDate	LSTMSGDA	MQCACH_LAST_MSG_DATE	
LastMsgTime	LSTMSGTI	MQCACH_LAST_MSG_TIME	

Table 4.7 Parameters returned by chs command.

Response Parameter	Display Name	Parameter ID	Note
LastSequenceNumber	LSTSEQNO	MQIACH_LAST_SEQ_NUMBER	
MaxMsgLength	MAXMSGL	MQIACH_MAX_MSG_LENGTH	
MaxSharingConversations	MAXSHCNV	MQIACH_MAX_SHARING_CONVS	
MCAStatus	MCASTAT	MQIACH_MCA_STATUS	
MCAUserIdentifier	MCAUSER	MQCACH_MCA_USER_ID	
ChannelMonitoring	MONCHL	MQIA_MONITORING_CHANNEL	
Msgs	MSGS	MQIACH_MSGS	
NetTime	NETTIME	MQIACH_NETWORK_TIME_INDICATOR	
NonPersistentMsgSpeed	NPMSPEED	MQIACH_NPM_SPEED	
QMgrName	QMNAME	MQCA_Q_MGR_NAME	
RemoteProduct	RPRODUCT	MQCACH_REMOTE_PRODUCT	
RemoteVersion	RVERSION	MQCACH_REMOTE_VERSION	
ShortRetriesLeft	SHORTRTS	MQIACH_SHORT_RETRIES_LEFT	
SecurityProtocol	SECPROT	MQIACH_SECURITY_PROTOCOL	
SSLCertRemoteIssuerName	SSLCERTI	MQCACH_SSL_CERT_ISSUER_NAME	
SSLCertUserId	SSLCERTU	MQCACH_SSL_CERT_USER_ID	
SSLCipherSpecification	SSLCIPH	MQCACH_SSL_CIPHER_SPEC	
SSLKeyResetDate	SSLKEYDA	MQCACH_SSL_KEY_RESET_DATE	
SSLKeyResetTime	SSLKEYTI	MQCACH_SSL_KEY_RESET_TIME	
SSLKeyResets	SSLRKEYS	MQIACH_SSL_KEY_RESETS	
SSLShortPeerName	SSLPEER	MQCACH_SSL_SHORT_PEER_NAME	

Table 4.7 Parameters returned by chs command.

Response Parameter	Display Name	Parameter ID	Note
ChannelStatistics	STATCHL	MQIA_STATISTICS_CHANNEL	
MsgsAvailable	XQMSGSA	MQIACH_XMITQ_MSGS_AVAILABLE	
XQTime	XQTIME	MQIACH_XMITQ_TIME_INDICATOR	

Ex. 4.8 chs command

$ mqpcf chs -qm TESTQM **-cn "kuipo(1414)"**
1: CHLINSTYPE(CURRENT) CHANNEL(TO.kuipo) STATUS(STOPPED)
CHLTYPE(SDR) **CONNAME(kuipo(1414))** RQMNAME() STOPREQ(NO)
SUBSTATE(OTHER) XMITQ(kuipo) BATCHES(0) BATCHSZ(50) BUFSRCVD(0)
BUFSSENT(0) BYTSRCVD(0) BYTSSENT(0) CHSTADA(2016-12-13)
CHSTATI(14.04.00) COMPHDR() COMPMSG() COMPRATE(0, 0) COMPTIME(0, 0)
CURLUWID(0000000000000000) CURMSGS(0) CURSEQNO(0) EXITTIME(0, 0)
HBINT(300) INDOUBT(NO) JOBNAME(0065004600000001) LOCLADDR()
LONGRTS(999999999) LSTLUWID(0000000000000000) LSTMSGDA() LSTMSGTI()
LSTSEQNO(0) MCASTAT(NOT RUNNING) MONCHL(OFF) MSGS(0) NETTIME(0,
0) NPMSPEED(FAST) RPRODUCT() RVERSION() SHORTRTS(5) SSLCERTI()
SSLKEYDA() SSLKEYTI() SSLRKEYS(0) SSLPEER() XQTIME(0, 0)

$ mqpcf chs -qm TESTQM **saved** CHLINSTYPE
1: CHLINSTYPE(SAVED) CHANNEL(TO.TESTQM) STATUS(STOPPED)
CHLTYPE(CLUSRCVR) CONNAME(TESTQM2)
2: CHLINSTYPE(SAVED) CHANNEL(TO.TESTQM) STATUS(STOPPED)
CHLTYPE(CLUSRCVR) CONNAME(mqm80d)
....
8: CHLINSTYPE(SAVED) CHANNEL(TO.TESTQM3) STATUS(STOPPED)
CHLTYPE(CLUSSDR) CONNAME(TESTQM3)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)

Inquire Channel Listener (lsnr)

The lsnr command execute MQCMD_INQUIRE_LISTENER MQAI command. It is equivalent to "display listener" of the runmqsc command.

USAGE : mqpcf lsnr -qm Qmgr [-ln Listener]

Parameters in the table below are displayed.

Table 4.8 Parameters returned by lsnr command.

Response Parameter	Display Name	Parameter ID	Note
ListenerName	LISTENER	MQCACH_LISTENER_NAME	Mandatory
Adapter	ADAPTER	MQIACH_ADAPTER	
AlterationDate	ALTDATE	MQCA_ALTERATION_DATE	
AlterationTime	ALTTIME	MQCA_ALTERATION_TIME	
Backlog	BACKLOG	MQIACH_BACKLOG	
Commands	COMMANDS	MQIACH_COMMAND_COUNT	
StartMode	CONTROL	MQIACH_LISTENER_CONTROL	
ListenerDesc	DESCR	MQCACH_LISTENER_DESC	
IPAddress	IPADDR	MQCACH_IP_ADDRESS	
LocalName	LOCLNAME	MQCACH_LOCAL_NAME	
NetbiosNames	NTBNAMES	MQIACH_NAME_COUNT	
Port	PORT	MQIACH_PORT	
Sessions	SESSIONS	MQIACH_SESSION_COUNT	
Socket	SOCKET	MQIACH_SOCKET	
TPName	TPNAME	MQCACH_TP_NAME	
TransportType	TRPTYPE	MQIACH_XMIT_PROTOCOL_TYPE	

Ex. 4.9 lsnr command

```
------------------------------------
$ mqpcf lsnr -qm TESTQM -ln LISTENER
1: LISTENER(LISTENER) ALTDATE(2014-09-22) ALTTIME(07.55.15) BACKLOG(0)
```

CONTROL(QMGR) DESCR() IPADDR() PORT(7180) TRPTYPE(TCP)

Inquire Channel Listener Status (lsst)

The lsst command execute MQCMD_INQUIRE_LISTENER_STATUS MQAI command. It is equivalent to "display lsstatus" of the runmqsc command.

USAGE : mqpcf lsst -qm Qmgr [-ln Listener]

Parameters in the table below are displayed.

Table 4.9 Parameters returned by lsst command.

Response Parameter	Display Name	Parameter ID	Note
ListenerName	LISTENER	MQCACH_LISTENER_NAME	Mandatory
Adapter	ADAPTER	MQIACH_ADAPTER	
Backlog	BACKLOG	MQIACH_BACKLOG	
Commands	COMMANDS	MQIACH_COMMAND_COUNT	
StartMode	CONTROL	MQIACH_LISTENER_CONTROL	
ListenerDesc	DESCR	MQCACH_LISTENER_DESC	
IPAddress	IPADDR	MQCACH_IP_ADDRESS	
LocalName	LOCLNAME	MQCACH_LOCAL_NAME	
NetbiosNames	NTBNAMES	MQIACH_NAME_COUNT	
ProcessId	PID	MQIACF_PROCESS_ID	
Port	PORT	MQIACH_PORT	
Sessions	SESSIONS	MQIACH_SESSION_COUNT	
Socket	SOCKET	MQIACH_SOCKET	
StartDate	STARTDA	MQCACH_LISTENER_START_DATE	
StartTime	STARTTI	MQCACH_LISTENER_START_TIME	
Status	STATUS	MQIACH_LISTENER_STATUS	
TPName	TPNAME	MQCACH_TP_NAME	
TransportType	TRPTYPE	MQIACH_XMIT_PROTOCOL_TYPE	

Ex. 4.10 lsst command

```
--------------------------------------
$ mqpcf lsst -qm TESTQM -ln LISTENER
1: LISTENER(LISTENER) BACKLOG(100) CONTROL(QMGR) DESCR() IPADDR(*)
PID(7340256) PORT(7180) STARTDA(2017-01-23) STARTTI(21.13.49)
STATUS(RUNNING) TRPTYPE(TCP)
--------------------------------------
```

Inquire Cluster Queue Manager (cqmgr)

The cqmgr command execute MQCMD_INQUIRE_CLUSTER_Q_MGR command. It is equivalent to "display clusqmgr" of the runmqsc command.

USAGE : mqpcf cqmgr -qm Qmgr [-cl Cluster] [-g GenericQmgr]

Parameters in the table below are displayed.

Table 4.10 Parameters returned by cqmgr command.

Response Parameter	Display Name	Parameter ID	Note
ChannelName	CHANNEL	MQCACH_CHANNEL_NAME	Mandatory
ClusterName	CLUSTER	MQCA_CLUSTER_NAME	Mandatory
QMgrName	CLUSQMGR	MQCA_CLUSTER_Q_MGR_NAME	Mandatory
AlterationDate	ALTDATE	MQCA_ALTERATION_DATE	
AlterationTime	ALTTIME	MQCA_ALTERATION_TIME	
BatchDataLimit	BATCHLIM	MQIACH_BATCH_DATA_LIMIT	
BatchHeartbeat	BATCHHB	MQIACH_BATCH_HB	
BatchInterval	BATCHINT	MQIACH_BATCH_INTERVAL	
BatchSize	BATCHSZ	MQIACH_BATCH_SIZE	
ClusterDate	CLUSDATE	MQCA_CLUSTER_DATE	
ClusterTime	CLUSTIME	MQCA_CLUSTER_TIME	
CLWLChannelPriority	CLWLPRTY	MQIACH_CLWL_CHANNEL_PRIORITY	
CLWLChannelRank	CLWLRANK	MQIACH_CLWL_CHANNEL_RANK	
CLWLChannelWeight	CLWLWGHT	MQIACH_CLWL_CHANNEL_WEIGHT	
HeaderCompression	COMPHDR	MQIACH_HDR_COMPRESSION	
MessageCompression	COMPMSG	MQIACH_MSG_COMPRESSION	

Table 4.10 Parameters returned by cqmgr command.

Response Parameter	Display Name	Parameter ID	Note
ConnectionName	CONNAME	MQCACH_CONNECTION_NAME	
DataConversion	CONVERT	MQIACH_DATA_CONVERSION	
QMgrDefinitionType	DEFTYPE	MQIACF_Q_MGR_DEFINITION_TYPE	
ChannelDesc	DESCR	MQCACH_DESC	
DiscInterval	DISCINT	MQIACH_DISC_INTERVAL	
HeartbeatInterval	HBINT	MQIACH_HB_INTERVAL	
KeepAliveInterval	KAINT	MQIACH_KEEP_ALIVE_INTERVAL	
LocalAddress	LOCLADDR	MQCACH_LOCAL_ADDRESS	
LongRetryCount	LONGRTY	MQIACH_LONG_RETRY	
LongRetryInterval	LONGTMR	MQIACH_LONG_TIMER	
MaxMsgLength	MAXMSGL	MQIACH_MAX_MSG_LENGTH	
MCAName	MCANAME	MQCACH_MCA_NAME	
MCAType	MCATYPE	MQIACH_MCA_TYPE	
MCAUserIdentifier	MCAUSER	MQCACH_MCA_USER_ID	
ChannelMonitoring	MONCHL	MQIA_MONITORING_CHANNEL	
ModeName	MODENAME	MQCACH_MODE_NAME	
MsgExit	MSGEXIT	MQCACH_MSG_EXIT_NAME	
MsgRetryUserData	MRDATA	MQCACH_MR_EXIT_USER_DATA	
MsgRetryExit	MREXIT	MQCACH_MR_EXIT_NAME	
MsgRetryCount	MRRTY	MQIACH_MR_COUNT	
MsgRetryInterval	MRTMR	MQIACH_MR_INTERVAL	
MsgUserData	MSGDATA	MQCACH_MSG_EXIT_USER_DATA	
NetworkPriority	NETPRTY	MQIACH_NETWORK_PRIORIT	

Table 4.10 Parameters returned by cqmgr command.

Response Parameter	Display Name	Parameter ID	Note
		Y	
NonPersistentMsgSpeed	NPMSPEED	MQIACH_NPM_SPEED	
Password	PASSWORD	MQCACH_PASSWORD	
PropCtl	PROPCTL	MQIA_PROPERTY_CONTROL	
PutAuthority	PUTAUT	MQIACH_PUT_AUTHORITY	
QMgrIdentifier	QMID	MQCA_Q_MGR_IDENTIFIER	
QMgrType	QMTYPE	MQIACF_Q_MGR_TYPE	
ReceiveUserData	RCVDATA	MQCACH_RCV_EXIT_USER_DATA	
ReceiveExit	RCVEXIT	MQCACH_RCV_EXIT_NAME	
SecurityUserData	SCYDATA	MQCACH_SEC_EXIT_USER_DATA	
SecurityExit	SCYEXIT	MQCACH_SEC_EXIT_NAME	
SendUserData	SENDDATA	MQCACH_SEND_EXIT_USER_DATA	
SendExit	SENDEXIT	MQCACH_SEND_EXIT_NAME	
SeqNumberWrap	SEQWRAP	MQIACH_SEQUENCE_NUMBER_WRAP	
ShortRetryCount	SHORTRTY	MQIACH_SHORT_RETRY	
ShortRetryInterval	SHORTTMR	MQIACH_SHORT_TIMER	
SSLClientAuth	SSLCAUTH	MQIACH_SSL_CLIENT_AUTH	
SSLCipherSpec	SSLCIPH	MQCACH_SSL_CIPHER_SPEC	
SSLPeerName	SSLPEER	MQCACH_SSL_PEER_NAME	
ChannelStatus	STATUS	MQIACH_CHANNEL_STATUS	
Suspend	SUSPEND	MQIACF_SUSPEND	
TpName	TPNAME	MQCACH_TP_NAME	
TransportType	TRPTYPE	MQIACH_XMIT_PROTOCOL_TYPE	

Table 4.10 Parameters returned by cqmgr command.

Response Parameter	Display Name	Parameter ID	Note
UseDLQ	USEDLQ	MQIA_USE_DEAD_LETTER_Q	
UserIdentifier	USERID	MQCACH_USER_ID	
Version	VERSION	MQCA_VERSION	
xmitq	XMITQ	MQCACH_XMIT_Q_NAME	

Ex. 4.11 cqmgr command

```
-------------------------------------
$ mqpcf cqmgr -qm TESTQM -cl REP80 -g TESTQM2
1: CHANNEL(TO.TESTQM2) CLUSTER(REP80) CLUSQMGR(TESTQM2)
ALTDATE(2016-09-23) ALTTIME(11.50.57) BATCHHB(20000) BATCHINT(5000)
BATCHSZ(50) CLUSDATE(2016-12-09) CLUSTIME(14.30.51) CLWLPRTY(0)
CLWLRANK(0) CLWLWGHT(50) COMPHDR() COMPMSG()
CONNAME(remotehost(1414)) CONVERT(NO) DEFTYPE(CLUSSDRA) DESCR()
DISCINT(60000) HBINT(30) KAINT(-1) LOCLADDR() LONGRTY(999999999)
LONGTMR(1200) MAXMSGL(4194304) MCANAME() MCATYPE(THREAD)
MCAUSER(testuser) MODENAME() MSGEXIT() MRDATA() MREXIT() MRRTY(0)
MRTMR(1000) MSGDATA() NETPRTY(0) NPMSPEED(NORMAL) PASSWORD()
PROPCTL(COMPAT) PUTAUT(DEF) QMID(STSCQM2_2015-09-30_11.53.10)
QMTYPE(NORMAL) RCVDATA() RCVEXIT() SCYDATA() SCYEXIT() SENDDATA()
SENDEXIT() SEQWRAP(999999999) SHORTRTY(100) SHORTTMR(30)
SSLCAUTH(REQUIRED) SSLCIPH() SSLPEER() STATUS(INACTIVE)
SUSPEND(NO) TPNAME() TRPTYPE(TCP) USEDLQ(YES) USERID()
VERSION(08000004) XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)


$ mqpcf cqmgr -qm TESTQM -cl "REP*" -g "TEST*" CLUSQMGR
1: CHANNEL(TO.TESTQM1) CLUSTER(REP80) CLUSQMGR(TESTQM1)
2: CHANNEL(TO.TESTQM2) CLUSTER(REP80) CLUSQMGR(TESTQM2))
3: CHANNEL(TO.TESTQM3) CLUSTER(REP80) CLUSQMGR(TESTQM3)
-------------------------------------
```

Inquire Connection (con)

The con command execute MQCMD_INQUIRE_CONNECTION MQAI command. It is equivalent to "display conn" of the runmqsc command. If "conn" is specified as argument, MQIACF_CONN_INFO_CONN is used for MQIACF_CONN_INFO_TYPE and if "handle" is specified, MQIACF_CONN_INFO_HANDLE is used and if neither is specified, MQIACF_CONN_INFO_ALL is used.

USAGE : mqpcf con -qm Qmgr {conn | handle}[-ap ApplTag]

Parameters in the table below are displayed.

Table 4.11 Parameters returned by con command.

Response Parameter	Display Name	Parameter ID	Note
ConnectionId	CONN	MQBACF_CONNECTION_ID	Mandatory
ConnInfoType	TYPE	MQIACF_CONN_INFO_TYPE	Mandatory
ObjectName	OBJNAME	MQCACF_OBJECT_NAME	Mandatory(handle)
ObjectType	OBJTYPE	MQIACF_OBJECT_TYPE	Mandatory(handle)
QSGDispositon	QSGDISP	MQIA_QSG_DISP	Mandatory(handle)
ApplDesc	APPLDESC	MQCACF_APPL_DESC	(conn)
ApplTag	APPLTAG	MQCACF_APPL_TAG	(conn)
ApplType	APPLTYPE	MQIA_APPL_TYPE	(conn)
ASID	ASID	MQCACF_ASID	[z/OS](conn)
AsynchronousStat	ASTATE	MQIACF_ASYNC_STATE	(conn)(handle)
ChannelName	CHANNEL	MQCACH_CHANNEL_NAME	(conn)
ConnectionName	CONNAME	MQCACH_CONNECTION_NAME	(conn)
ConnectionOptions	CONNOPTS	MQIACF_CONNECT_OPTIONS	(conn)
Destination	DEST	MQCACF_DESTINATION	(handle)

Table 4.11 Parameters returned by con command.

Response Parameter	Display Name	Parameter ID	Note
DestinationQueueManager	DESTQMGR	MQCACF_DESTINATION_Q_MGR	(handle)
HandleState	HSTATE	MQIACF_HANDLE_STATE	(handle)
OriginName	NID	MQCACF_ORIGIN_NAME	[z/OS](conn)
OpenOptions	OPENOPTS	MQIACF_OPEN_OPTIONS	(handle)
ProcessId	PID	MQIACF_PROCESS_ID	[z/OS](conn)
PSBName	PSBNAME	MQCACF_PSB_NAME	(conn)
PSTId	PSTID	MQCACF_PST_ID	(conn)
QMgrUOWId	QMURID	MQBACF_Q_MGR_UOW_ID	(conn)
ReadAhead	READA	MQIA_READ_AHEAD	(handle)
StartUOWLogExtent	UOWLOG	MQCACF_UOW_LOG_EXTENT_NAME	(conn)
SubscriptionID	SUBID	MQBACF_SUB_ID	(handle)
SubscriptionName	SUBNAME	MQCACF_SUB_NAME	(handle)
ThreadId	TID	MQIACF_THREAD_ID	(conn)
TopicString	TOPICSTR	MQCA_TOPIC_STRING	(handle)
TransactionId	TRANSID	MQCACF_TRANSACTION_ID	[z/OS](conn)
UOWIdentifier	URID	MQBACF_EXTERNAL_UOW_ID	(conn)
UOWLogStartDate	UOWLOGDA	MQCACF_UOW_LOG_START_DATE	(conn)
UOWLogStartTime	UOWLOGTI	MQCACF_UOW_LOG_START_TIME	(conn)
UOWState	UOWSTATE	MQIACF_UOW_STATE	(conn)
UOWStartDate	UOWSTDA	MQCACF_UOW_START_DATE	(conn)
UOWStartTime	UOWSTTI	MQCACF_UOW_START_TIME	(conn)
UOWType	URTYPE	MQIACF_UOW_TYPE	(conn)

Table 4.11 Parameters returned by con command.

Response Parameter	Display Name	Parameter ID	Note
UserId	USERID	MQCACF_USER_IDENTIFIER	(conn)

Ex. 4.12 con command

$ mqpgf -qm TESTQM -q TQ MQWI_UNLIMITED MQGMO_WAIT

$ mqpcf con -qm TESTQM **-ap mqpgf**
1: CONN(414D51436F6B61716D383061202020205889D1020002001) **TYPE(ALL)**
APPLDESC() **APPLTAG(mqpgf)** APPLTYPE(USER) ASTATE(NONE) CHANNEL()
CONNAME() CONNOPTS(MQCNO_SHARED_BINDING) PID(6488090)
QMURID(000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000) UOWLOG() TID(1)
URID(000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000) UOWLOGDA() UOWLOGTI() UOWSTATE(MQUOWST_NONE)
UOWSTDA() UOWSTTI() URTYPE(QMGR) USERID(mq80)
2: CONN(414D51436F6B61716D383061202020205889D1020002001) TYPE(ALL)
OBJNAME(TQ) OBJTYPE(QUEUE) ASTATE(NONE) HSTATE(ACTIVE)
OPENOPTS(MQOO_INPUT_SHARED) READA(NO)

$ mqpcf con -qm TESTQM handle -ap mqpgf OPENOPTS
1: CONN(414D51436F6B61716D383061202020205889F36720002101)
TYPE(HANDLE) OBJNAME(TQ) OBJTYPE(QUEUE)

OPENOPTS(MQOO_INPUT_SHARED)

$ mqpcf con -qm TESTQM conn UOWSTATE
1: CONN(414D51436F6B61716D383061202020205889F36720000201) TYPE(CONN)
UOWSTATE(MQUOWST_ACTIVE)
2: CONN(414D51436F6B61716D383061202020205889F36720000101) TYPE(CONN)
UOWSTATE(MQUOWST_NONE)
....
23: CONN(414D51436F6B61716D383061202020205889F36720000801) TYPE(CONN)
UOWSTATE(MQUOWST_NONE)

Ping Queue Manager (pngm)

The pngm command execute MQCMD_PING_Q_MGR MQAI command. It is equivalent to "ping qmgr" of the runmqsc command.

USAGE : mqpcf pngm -qm Qmgr

Ex. 4.13 pngm command

```
-------------------------------------
$ mqpcf pngm -qm TESTQM
Ping Queue Manager Success. Queue Manager : TESTQM
-------------------------------------
```

Ping Channel (ping)

The ping command execute MQCMD_PING_CHANNE MQAI command. It is equivalent to "ping channel" of the runmqsc command.

USAGE : mqpcf ping -qm Qmgr -c Channel [-l DataLen(16-32768)]

Ex. 4.14 ping command

```
-------------------------------------
$ mqpcf ping -qm TESTQM1 -c TESTQM1.to.TESTQM2
Ping Channel Success. Channel Name : TESTQM1.to.TESTQM2


$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.to.TESTQM2) STATUS(RUNN
ING) CHLTYPE(SDR) CONNAME(remotehost(1414)) RQMNAME(TESTQM2) STO
PREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM2)
$
$ mqpcf ping -qm TESTQM1 -c TESTQM1.to.TESTQM2 -l 16
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], mqCommandRC=[4031]
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], mqCommandRC=[3008]
$ mqrc 4031

     4031   0x00000fbf   MQRCCF_CHANNEL_IN_USE


$ mqpcf ping -qm TESTQM1 -c TESTQM1.to.TESTQM2
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], mqCommandRC=[4010]
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], m
qCommandCC=[2], mqCommandRC=[3008]

$ mqrc 4010

     4010   0x00000faa   MQRCCF_HOST_NOT_AVAILABLE
-------------------------------------
```

Change Queue (put/get)

The put and get command execute MQCMD_CHANGE_Q MQAI command. It is equivalent to changing the PUT / GET attribute with "alter queue" of the r unmqsc command.

USAGE : mqpcf {put | get} {enable | disable} -qm Qmgr -q Queue

Ex. 4.15 put/get command

```
---------------------------------------
$ mqpcf que -qm TESTQM -q TQ PUT
1: QUEUE(TQ) TYPE(QLOCAL) PUT(ENABLED)

$ mqpcf put disable -qm TESTQM -q TQ
Put Disabled : TQ
$ mqpcf que -qm TESTQM -q TQ PUT
1: QUEUE(TQ) TYPE(QLOCAL) PUT(DISABLED)


$ mqpcf que -qm TESTQM -q TQ GET
1: QUEUE(TQ) TYPE(QLOCAL) GET(DISABLED)

$ mqpcf get enable -qm TESTQM -q TQ
Get Enabled : TQ

$ mqpcf que -qm TESTQM -q TQ GET
1: QUEUE(TQ) TYPE(QLOCAL) GET(ENABLED)
---------------------------------------
```

Clear Queue (clr)

The clr command execute MQCMD_CLEAR_Q MQAI command. It is equivale nt to "clear queue" of the runmqsc command.

USAGE : mqpcf clr -qm Qmgr -q Queue

Ex. 4.16 delete messages on a queue

```
---------------------------------------
$ mqpcf que -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QLOCAL) CURDEPTH(101)

$ mqpcf clr -qm TESTQM -q TQ
Clear Queue Success. Queue Name : TQ

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

Reset Channel (rst)

The rst command execute MQCMD_RESET_CHANNEL MQAI command. It is equivalent to "reset channel" of the runmqsc command.

USAGE : mqpcf rst -qm Qmgr -c Channel [-n SeqNo(1-999999999)]

Ex. 4.17 reset channel

```
-------------------------------------
$ mqpcf chs -qm TESTQM1 -c TESTQM1.TO.TESTQM2 STATUS CURSEQNO
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.TO.TESTQM2) STATUS(RUN
NING) CHLTYPE(SDR) CONNAME(remotehost(1414)) RQMNAME(TESTQM2) ST
OPREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM2) CURSEQNO(1)
$
$ mqpcf rst -qm TESTQM1 -c TESTQM1.TO.TESTQM2 -n 100
Channel Reset Success. Channel Name : TESTQM1.TO.TESTQM2
$
$ mqpgf -qm TESTQM1 -q RTQ -m test MQPER_PERSISTENT
[17/01/27 21:29:52] 1: message length: 4 put message : test
$
$ mqpcf chs -qm TESTQM1 -c TESTQM1.TO.TESTQM2 STATUS CURSEQNO
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.TO.TESTQM2) STATUS(RUN
NING) CHLTYPE(SDR) CONNAME(remotehost(1414)) RQMNAME(TESTQM2) ST
OPREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM2) CURSEQNO(2)

$ mqpcf stp -qm TESTQM1 -c TESTQM1.TO.TESTQM2
Channel Stop Success. Channel Name : TESTQM1.TO.TESTQM2 Connection Na
me :  Queue Manager : TESTQM2

$ mqpcf sta -qm TESTQM1 -c TESTQM1.TO.TESTQM2
Channel Start Success. Channel Name : TESTQM1.TO.TESTQM2

$ mqpcf chs -qm TESTQM1 -c TESTQM1.TO.TESTQM2 STATUS CURSEQNO
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.TO.TESTQM2) STATUS(RUN
NING) CHLTYPE(SDR) CONNAME(remotehost(1414)) RQMNAME(TESTQM1) ST
OPREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM1) CURSEQNO(2)

$ mqpgf -qm TESTQM1 -q RTQ -m test MQPER_PERSISTENT
[17/01/27 21:30:32] 1: message length: 4 put message : test

$ mqpcf chs -qm TESTQM1 -c TESTQM1.TO.TESTQM2 STATUS CURSEQNO
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.TO.TESTQM2) STATUS(RUN
NING) CHLTYPE(SDR) CONNAME(remotehost(1414)) RQMNAME(TESTQ1) STO
PREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQ1) CURSEQNO(100)
```

Resolve Channel (rslv)

The rslv command execute MQCMD_RESOLVE_CHANNEL MQAI command.. It is equivalent to "resolve channel" of the runmqsc command.

USAGE : mqpcf rslv -qm Qmgr -c Channel {commit | backout}

Ex. 4.18 resolve channel

```
-------------------------------------
$ mqpcf rslv -qm TESTQM1 -c TESTQM1.to.TESTQM2 commit
Channel Resolve Success. Channel Name : TESTQM1.to.TESTQM2

$ mqpcf rslv -qm TESTQM1 -c TESTQM1.to.TESTQM2 backout
Channel Resolve Success. Channel Name : TESTQM1.to.TESTQM2
-------------------------------------
```

Start Channel (sta)

The sta command execute MQCMD_START_CHANNEL MQAI command. It is equivalent to "start channel" of the runmqsc command.

USAGE : mqpcf sta -qm Qmgr -c Channel

Ex. 4.19 start channel

```
-------------------------------------
$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.to.TESTQM2)
STATUS(STOPPED) CHLTYPE(SDR) CONNAME(localhost(1414))
RQMNAME(TESTQM2) STOPREQ(NO) SUBSTATE(OTHER) XMITQ(TESTQM2)

$ mqpcf sta -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
Channel Start Success. Channel Name : TESTQM1.to.TESTQM2

$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.to.TESTQM2)
STATUS(RUNNING) CHLTYPE(SDR) CONNAME(remotehost(1414))
RQMNAME(TESTQM2) STOPREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM2)
-------------------------------------
```

Stop Channel (stp)

The stp command execute MQCMD_STOP_CHANNEL MQAI command. It is equivalent to "stop channel" of the runmqsc command. By specifying "force" or "term", MQMODE_FORCE or MQMODE_TERMINATE is used respectively. If " inact" is specified, the channel status after stopping will be INACTIVE.

USAGE : mqpcf stp -qm Qmgr -c Channel [force | term] [inact] [-rm RemoteQm gr] [-cn Connection]

Ex. 4.20 stop channel

$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.to.TESTQM2)
STATUS(RUNNING) CHLTYPE(SDR) CONNAME(remotehost(1414))
RQMNAME(TESTQM2) STOPREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM2)

$ mqpcf **stp** -qm TESTQM1 -c TESTQM1.to.TESTQM2
Channel Stop Success. Channel Name : TESTQM1.to.TESTQM2 Connection Name :
Queue Manager : TESTQM1

$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.to.TESTQM2)
STATUS(STOPPED) CHLTYPE(SDR) CONNAME(remotehost(1414))
RQMNAME(TESTQM1) STOPREQ(NO) SUBSTATE(OTHER) XMITQ(TESTQM1)

$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TESTQM1.to.TESTQM2)
STATUS(RUNNING) CHLTYPE(SDR) CONNAME(localhost(1414))
RQMNAME(TESTQM2) STOPREQ(NO) SUBSTATE(MQGET) XMITQ(TESTQM2)

$ mqpcf stp -qm TESTQM1 -c TESTQM1.to.TESTQM2 force inact
Channel Stop Success. Channel Name : TESTQM1.to.TESTQM2 Connection Name :
Queue Manager : TESTQM1

$ mqpcf chs -qm TESTQM1 -c TESTQM1.to.TESTQM2 STATUS
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008],
mqCommandCC=[2], **mqCommandRC=[3065]**
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008],
mqCommandCC=[2], mqCommandRC=[3008]

$ mqrc 3065

3065 0x00000bf9 MQRCCF_CHL_STATUS_NOT_FOUND

$ mqpcf chs -qm CLB -c TO.CLB STATUS
1: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(111.111.111.111) RQMNAME(CLA)**
STOPREQ(NO) SUBSTATE(RECEIVE)
2: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(222.222.222.222) RQMNAME(CLC)**
STOPREQ(NO) SUBSTATE(RECEIVE)
3: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(333.333.333.333) RQMNAME(STSCQM2)**
STOPREQ(NO) SUBSTATE(RECEIVE)

$ mqpcf **stp** -qm CLB -c TO.CLB
Channel Stop Success. Channel Name : TO.CLB Connection Name : Queue Manager : CLB

$ mqpcf chs -qm CLB -c TO.CLB STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) **STATUS(STOPPING)**
CHLTYPE(CLUSRCVR) CONNAME(111.111.111.111) RQMNAME(CLA)
STOPREQ(YES) SUBSTATE(RECEIVE)
2: CHLINSTYPE(CURRENT) CHANNEL(TO.CLLB) **STATUS(STOPPING)**
CHLTYPE(CLUSRCVR) CONNAME(222.222.222.222) RQMNAME(CLC)
STOPREQ(YES) SUBSTATE(RECEIVE)
3: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) **STATUS(STOPPING)**
CHLTYPE(CLUSRCVR) CONNAME(333.333.333.333) RQMNAME(CLD)
STOPREQ(YES) SUBSTATE(RECEIVE)

$ mqpcf chs -qm CLB -c TO.CLB STATUS
1: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(111.111.111.111) RQMNAME(CLA)**
STOPREQ(NO) SUBSTATE(RECEIVE)
2: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(222.222.222.222) RQMNAME(CLC)**

STOPREQ(NO) SUBSTATE(RECEIVE)
3: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(333.333.333.333) RQMNAME(STSCQM2)**
STOPREQ(NO) SUBSTATE(RECEIVE)

$ mqpcf stp -qm CLB -c TO.CLB **-rm CLA**
Channel Stop Success. Channel Name : TO.CLB Connection Name : Queue Manager : CLA
$ mqpcf chs -qm CLB -c TO.CLB STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) STATUS(RUNNING)
CHLTYPE(CLUSRCVR) CONNAME(111.111.111.111) RQMNAME(CLC)
STOPREQ(NO) SUBSTATE(RECEIVE)
2: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) **STATUS(STOPPING)**
CHLTYPE(CLUSRCVR) CONNAME(222.222.222.222) **RQMNAME(CLA)**
STOPREQ(YES) SUBSTATE(RECEIVE)
3: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) STATUS(RUNNING)
CHLTYPE(CLUSRCVR) CONNAME(333.333.333.333) RQMNAME(CLD)
STOPREQ(NO) SUBSTATE(RECEIVE)

$ mqpcf chs -qm CLB -c TO.CLB STATUS
1: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(111.111.111.111) RQMNAME(CLA)**
STOPREQ(NO) SUBSTATE(RECEIVE)
2: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(222.222.222.222) RQMNAME(CLC)**
STOPREQ(NO) SUBSTATE(RECEIVE)
3: CHLINSTYPE(CURRENT) **CHANNEL(TO.CLB) STATUS(RUNNING)**
CHLTYPE(CLUSRCVR) **CONNAME(333.333.333.333) RQMNAME(CLD)**
STOPREQ(NO) SUBSTATE(RECEIVE)

$ mqpcf stp -qm CLB -c TO.CLB **-cn 333.333.333.333**
Channel Stop Success. Channel Name : TO.CLB **Connection Name : 333.333.333.333**
Queue Manager :

$ mqpcf chs -qm CLB -c TO.CLB STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) STATUS(RUNNING)
CHLTYPE(CLUSRCVR) CONNAME(222.222.222.222) RQMNAME(CLC)

STOPREQ(NO) SUBSTATE(RECEIVE)
2: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) **STATUS(STOPPING)**
CHLTYPE(CLUSRCVR) **CONNAME(333.333.333.333)** RQMNAME(CLD)
STOPREQ(YES) SUBSTATE(RECEIVE)
3: CHLINSTYPE(CURRENT) CHANNEL(TO.CLB) STATUS(RUNNING)
CHLTYPE(CLUSRCVR) CONNAME(111.111.111.111) RQMNAME(CLA)
STOPREQ(NO) SUBSTATE(RECEIVE)

Start Channel Listener (stalsn)

The stalsn command execute MQCMD_START_CHANNEL_LISTENER MQAI command.

It is equivalent to "start listener" of the runmqsc command.

USAGE : mqpcf stalsn -qm Qmgr -ln Listener

Ex. 4.21 start channel listener

```
---------------------------------------
$ mqpcf lsst -qm HM8A -ln LISTENER
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]
```

* 2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME is returned as a reason code when the status of the channel listener is queried by the command lsst if t he listener is not started.

```
$ mqpcf stalsn -qm HM8A -ln LISTENER
Listener Start Success. Listener Name : LISTENER

$ mqpcf lsst -qm HM8A -ln LISTENER
1: LISTENER(LISTENER) STATUS(RUNNING) PID(22884) STARTDA(2021-07-01)
 STARTTI(16.33.25) DESCR() TRPTYPE(TCP) CONTROL(MANUAL) IPADDR(*)
PORT(1414) BACKLOG(100)
---------------------------------------
```

Stop Channel Listener (stplsn)

The stplsn command execute MQCMD_STOP_CHANNEL_LISTENER MQAI command.

It is equivalent to "stop listener" of the runmqsc command.

USAGE : mqpcf stplsn -qm Qmgr -ln Listener

Ex. 4.22 stop channel listener

```
-------------------------------------
$ mqpcf lsst -qm HM8A -ln LISTENER
1: LISTENER(LISTENER) STATUS(RUNNING) PID(22884) STARTDA(2021-07-01)
 STARTTI(16.33.25) DESCR() TRPTYPE(TCP) CONTROL(MANUAL) IPADDR(*)
PORT(1414) BACKLOG(100)

$ mqpcf stplsn -qm HM8A -ln LISTENER
Listener Stop Success. Listener Name : LISTENER

$ mqpcf lsst -qm HM8A -ln LISTENER
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]
-------------------------------------
```

Escape (mqsc)

The mqsc command execute MQCMD_ESCAPE MQAI command. Specify the MQSC script file with the "-f" option, or specify "-s" followed by character strings of the runmqsc command directly.

USAGE : mqpcf mqsc -qm Qmgr {-f MqscFile | -s 'Mqcmd'}

Ex. 4.23 Send MQSC command to a remote queue manager.

```
-------------------------------------
$ mqpcfc mqsc -qm CLA -s "dis ql(C*)" -x "remotehost(1414)"
1: AMQ8409: Display Queue details.
   QUEUE(CLUS_Q1)                          TYPE(QLOCAL)


$ mqpcfc mqsc -qm CLA -f mqsc.scr -x "remotehost(1414)"

MQSC Command [1] : dis ql(c*)

1: AMQ8409: Display Queue details.
   QUEUE(CLUS_Q1)                          TYPE(QLOCAL)


MQSC Command [2] : dis chs(*)

1: AMQ8417: Display Channel Status details.
   CHANNEL(TO.CLB)                          CHLTYPE(CLUSSDR)
   CONNAME(111.111.111.111(1414))           CURRENT
   RQMNAME( )                               STATUS(RETRYING)
   SUBSTATE( )                              XMITQ(SYSTEM.CLUSTER.TRANS
MIT.QUEUE)

2: AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)                   CHLTYPE(SVRCONN)
   CONNAME(222.222.222.222)                 CURRENT
   STATUS(RUNNING)                          SUBSTATE(RECEIVE)


MQSC Command [3] : dis lsstatus(*)

1: AMQ8631: Display listener status details.
   LISTENER(LISTENER)                       STATUS(RUNNING)
   PID(9996)
-------------------------------------
```

5. Additional parameters

Repeat Count (-rc)

The number of times to repeat the command. It is available with all commands.

mqpcf <cmd> -qm <qmgr>… **-rc <nnn>**

Ex. 5.1 Repeat chs command repeat by a specified interval and specified number of times and also display execution time.

```
---------------------------------------
$ mqpcf chs -qm TESTQM -rc 3 -i 2 -t MSGS
[17/01/30 20:41:40] 1: CHLINSTYPE(CURRENT) CHANNEL(SSLCHL) STATUS(S
TOPPED) CHLTYPE(SVRCONN) CONNAME() RAPPLTAG() STOPREQ(NO) SUB
STATE(OTHER) MSGS(0)
[17/01/30 20:41:40] 2: CHLINSTYPE(CURRENT) CHANNEL(TO.kuipo) STATUS(S
TOPPED) CHLTYPE(SDR) CONNAME(kuipo(1414)) RQMNAME() STOPREQ(NO)
SUBSTATE(OTHER) XMITQ(kuipo) MSGS(0)
[17/01/30 20:41:42] 1: CHLINSTYPE(CURRENT) CHANNEL(SSLCHL) STATUS(S
TOPPED) CHLTYPE(SVRCONN) CONNAME() RAPPLTAG() STOPREQ(NO) SUB
STATE(OTHER) MSGS(0)
[17/01/30 20:41:42] 2: CHLINSTYPE(CURRENT) CHANNEL(TO.kuipo) STATUS(S
TOPPED) CHLTYPE(SDR) CONNAME(kuipo(1414)) RQMNAME() STOPREQ(NO)
SUBSTATE(OTHER) XMITQ(kuipo) MSGS(0)
[17/01/30 20:41:44] 1: CHLINSTYPE(CURRENT) CHANNEL(SSLCHL) STATUS(S
TOPPED) CHLTYPE(SVRCONN) CONNAME() RAPPLTAG() STOPREQ(NO) SUB
STATE(OTHER) MSGS(0)
[17/01/30 20:41:44] 2: CHLINSTYPE(CURRENT) CHANNEL(TO.kuipo) STATUS(S
TOPPED) CHLTYPE(SDR) CONNAME(kuipo(1414)) RQMNAME() STOPREQ(NO)
SUBSTATE(OTHER) XMITQ(kuipo) MSGS(0)
---------------------------------------
```

Interval (-i)

Execution interval in seconds when commands are repeatedly executed. You can also specify decimal places. It is available with all commands. This can be specified up to 10 digits.

mqpcf <cmd> -qm <qmgr>… -rc <nnn> **-i <nnn>**

Display Time (-t)

Displays the command execution time. It is available with all commands.

mqpcf <cmd> -qm <qmgr>… -rc <nnn> -i <nnn> **-t**

Wait Interval (-wi)

This data item specifies the maximum time in milliseconds that the MQAI should wait for each reply message. The time interval must be zero or greater; the default is ten seconds. The mqExecute call completes either when all of the reply messages are received or when the specified wait interval expires without the expected reply message having been received. It is available with all commands.

mqpcf <cmd> -qm <qmgr>… -rc <nnn> -i <nnn> -t -wi <nnn>

For example, if the SSL/TLS setting information is refreshed while the channel is connected, it may take some time to receive the refresh execution result from the command server. In that case, the return of the response from the command server is delayed, and 2033 (MQRC_NO_MSG_AVAILABLE) is displayed as shown below after the default response wait time of 10 seconds has passed.

MQExecute : Message Get Fail(no msg available).CompCode=[2], ReasonCode=[2033]

CSP User Id (-cu)

Specify a user id that a queue manager uses for connection authentication. Used with "CSP password (-cp)".

mqpcf <cmd> -qm <qmgr>… -cu <user id> -cp <password>

CSP Password (-cp)

Specify a password that a queue manager uses for connection authentication. Used with "CSP password (-cp)".

mqpcf <cmd> -qm <qmgr>… -cu <user id> -cp <password>

Certificate Label (-lb)

Specify when connecting to a channel using SSL/TLS in client mode. Use if you need to specify a label other than the queue manager's default certificate label.

SSLCipher Spec (-cs)

Specify when connecting to a channel using SSL/TLS in client mode. Specifies the name of the SSL cipher spec to use.

SSLPeerName (-er)

Specify when connecting to a channel using SSL/TLS in client mode. Specify a string to verify the SSL Peer name.

Key Repository (-kr)

Specifies where the key repository is located when connecting to a channel using SSL/TLS in client mode. For GSkit, specify <directory>/<key DB extension>, and for Openssl (MQ for HPE NonStop, etc.), specify the directory where the certificate file is located.

Conclusion

If you find any defects in this program, or if you have any questions and reques ts about this program, please contact us.

Pulsar Integration Inc.
https://www.pulsarintegration.com
https://www.pulsarintegration.jp
e-mail: support@pulsarintegration.com