MQMONA
Command Server Agent for IBM MQ
User Guide

Version 9.3.0

28th April 2023

Paul Clarke

MQGem Software Limited
support@mqgem.com

Take Note!

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

First Edition, April 2023

This edition applies to Version 9.3.0 of the MQMONA Command Server Agent for IBM MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

MQGEM SOFTWARE LIMITED PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

The information contained in this document has not be submitted to any formal test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by MQGem Software for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:
IBM MQ
IBM
z/OS

The following terms are trademarks of the Microsoft Corporation in the United States and/or other countries:
Windows 95,98,Me
Windows NT, 2000,XP

Table of Contents

History

The MQMONA Command Server for IBM MQ agent was a simple 'C' program that was created in the same time frame as the MO71 IBM MQ GUI Administration SupportPac to mitigate problems with latency and bandwidth on slow networks. It was retired when the IBM MQ product added the client read-ahead feature which provided a different solution to the same problem, and it seemed that the agent was no longer necessary.

In recent times, we have been requested by some users to revive the agent as use of the read ahead feature does not always provide the necessary relief.

This product is loosely based on the original simple 'C' program, but has had a number improvements made from a serviceability and reporting perspective over and above what the original program did.

I hope you find the program useful. As always I welcome your comments, both good and bad. Please feel free to e-mail me at support@mqgem.com with any bug reports or suggestions.

Chapter 1. MQMONA Command Server Agent for IBM MQ

1.1. Overview

The MQMONA Command Server Agent for IBM MQ is a program that can significantly improve the performance of MQ administrative tools in environments with low bandwidth, high latency, and/or large numbers of MQ objects. The agent achieves this by listening for MQSC and PCF commands issued by programs such as MO71 and MQSCX. It forwards them onto the IBM MQ Command Server and when it receives the reply messages, it compresses and consolidates them into a few, perhaps even only one, IBM MQ message, to be sent to the original application's reply queue, and retrieved by the application over the client network link.

When the IBM MQ Command Server responds to display commands, which result in multiple different objects being returned as the output, it does so with a single reply message for each object. By consolidating these many reply messages into one message, network latency becomes less of an issue, because there are fewer MQGET calls made by the original application. Read more about consolidation in Chapter 4. on page 8.

The IBM MQ Command Server tends to blank pad strings to their full length. A queue name, for example, is 48 characters long, and unless you have very long queue names, this tends to be a majority of blank characters. MQMONA strips those spaces to compress the data into a smaller message, and so bandwidth becomes less of an issue. Read more about compression in Chapter 5. on page 10.

1.2. Changes from previous version

The main changes from the original version of the program are:

1. **Message Consumers**
   MQMONA has been changed to use the IBM MQ Asynchronous Consumer interface instead of polling MQGET calls.

2. **Object Handle Cache**
   MQMONA will cache the object handles that it uses to improve the efficiency of subsequent requests as it is likely that a requester will come back again with another request.

3. **Reply when timed out**
   Commands that get no reply from the IBM MQ Command Server are handled more visibly. See Chapter 8. Timing out commands on page 13

4. **MQMONA can be Triggered**
   To ensure MQMONA is always available to process your command messages, you can define MQMONA to run as a SERVICE object (see Chapter 6. on page 11) and/or triggered upon the arrival of command messages (see Chapter 7. Triggering on page 12).

5. **Status Reporting**
   MQMONA reports status on a number of items that are interesting. For more details see Chapter 9. Status Reporting on page 14.

6. **Licensing**
   MQMONA is licensed based on your pre-existing MQGem product licenses. For more details see Chapter 2. Licensing on page 4.

7. **Configuration File**
   In addition to the main program parameters, MQMONA has a number of other optional configuration attributes supplied to it in a configuration file which are detailed in Chapter 12. Configuration File on page 17.

8. **Numbered Error Messages**
   When MQMONA needs to report some information or an error, it does so by writing a message with a message number. See Chapter 15. Messages on page 21 for a full catalogue of the error messages output by MQMONA.

9. **Error messages written to JESMSGLG on z/OS**
   When running MQMONA as a batch job or started task on z/OS, error messages are written to the JESMSGLG.

10. **Error Log file**
    On distributed platforms, and on z/OS when running in z/OS UNIX, MQMONA will write error messages to a rotating error log file. See Chapter 14. Error reporting on page 20 for full details.

11. **Stopping MQMONA**
    The MQMONA agent can be asked to stop gracefully with a STOP command, issued either by running the MQMONA executable with a simple stop command, or by using the MVS STOP command.

1.3. Installation

Installation has been made as simple as possible. Click the download button on the web site for the platform(s) you are interested in. This will download a zip file to your download location. Extract the files from the zip file using the appropriate tools for the platform listed below.

Platform	Unzip commands
AIX	gzip -d tar -xvf
Intel Linux	tar -xvzf
Power Linux	tar -xvzf
Windows	Extract using your favourite zip utility
z/OS	Extract using your favourite zip utility before transferring to a z/OS system. See 1.3.2 z/OS Installation Instructions below for more details

1.3.1. Linux, Unix and Windows

Once unzipped you should have the **mqmona** executable. Copy this file into the directory where you wish to run the program from. Normally this would be somewhere in your PATH. Remember that if you transfer the file between machines you should make sure you do so in binary.

1.3.2. z/OS Installation Instructions

Once unzipped, transfer the ***MQMONA.SEQ*** file to a z/OS system using the following commands.

```
ftp> binary
ftp> quote site recfm=FB lrecl=80 blksize=3120 blocks primary=1000
ftp> put MQMONA.SEQ
```

Once the ***MQMONA.SEQ*** file is successfully FTPed to your z/OS system, from TSO issue the following:-

```
receive inds(MQMONA.SEQ)
```

When prompted for a filename, reply

```
DSN(USER.LOAD)
```

MQMONA can be run on z/OS in BATCH (including as a Started Task). Example pieces of JCL are provided in the zip file. MQMONA can also be run interactively, e.g. from the TSO/E READY prompt, or the ISPF Command Shell (=6). It can also be run in z/OS UNIX (see below).

1.3.3. z/OS UNIX Installation Instructions

If you wish to run MQMONA in z/OS UNIX, you can copy the MVS executable module that you have installed in the previous section, to a z/OS UNIX executable file using the following command.

```
TSO OPUT 'GEMUSER.USER.LOAD(MQMONA)' '/u/gemuser/bin/mqmona' BIN
```

Chapter 2. Licensing

MQMONA is licensed as a result of your existing MO71 and/or MQSCX licenses. Whichever applications you want to use with MQMONA, ensure the MQMONA program can find the licences for those programs.

2.1. Licence File Location

When you bought an MQGem Software licence, you were sent an ***mqgem.lic*** file. All you need to do is place this licence file in the appropriate place for the MQMONA program to find it as detailed in the table.

Platform	Location
Windows and Linux	Same directory as the MQMONA program
AIX and z/OS UNIX	Current directory
z/OS	DD:MQGEML

Alternatively you can set environment variable **MQGEML** to point to the directory path where the licence file can be found (in which case the name will be assumed to be ***mqgem.lic***), or MVS file or DD name of the licence file. For example, if you use the program in all of TSO, z/OS UNIX and from JCL, you can have one copy of the licence file saved either as a z/OS UNIX file or in an MVS dataset, and refer to it from any environment.

2.2. Multiple licences

If you have multiple licences, for example one for MO71 and one for MQSCX, then they must be concatenated into a single **mqgem.lic** file for MQMONA to find. This can be done using simple OS commands such as **copy** or by using your favourite editor.

2.3. Reporting found licences

MQMONA will report which licenses it found, and thus which programs will be able to use the MQMONA agent. You will see output like the following when MQMONA starts up.

```
[13:38:42] MQGA221I MO71 Licence details: Valid - expires 09/09/23
[13:38:42] MQGA221I MQSCX Licence details: Valid - expires 09/09/23
```

If no licence file is found, then instead you will see output like this when MQMONA starts up. In this situation any command messages that arrived on the MQMONA command queue will be forwarded on to the IBM MQ Command Server queue and all MQMONA functionality will be switched off.

```
[02:39:05] MQGA220E No valid licence found
```

Chapter 3. Getting Started

Running MQMONA is quite simple. It needs one new queue definition in order to operate. By default it will try to use a queue named `MQGEM.MQMONA.COMMAND.QUEUE` as its input queue. If you prefer to have a differently named queue, you can do that and provide its name with the `-q` parameter flag. For this chapter we will define the default queue.

```
DEFINE QLOCAL(MQGEM.MQMONA.COMMAND.QUEUE)
       DESCR('MQGem Command Server Agent queue')
```

Now issue the following command to run MQMONA against a queue manager called MQG1.

```
mqmona -m MQG1
```

You will see a number of informational messages to show you that MQMONA has got up and running, and the names of the various queues it is using.

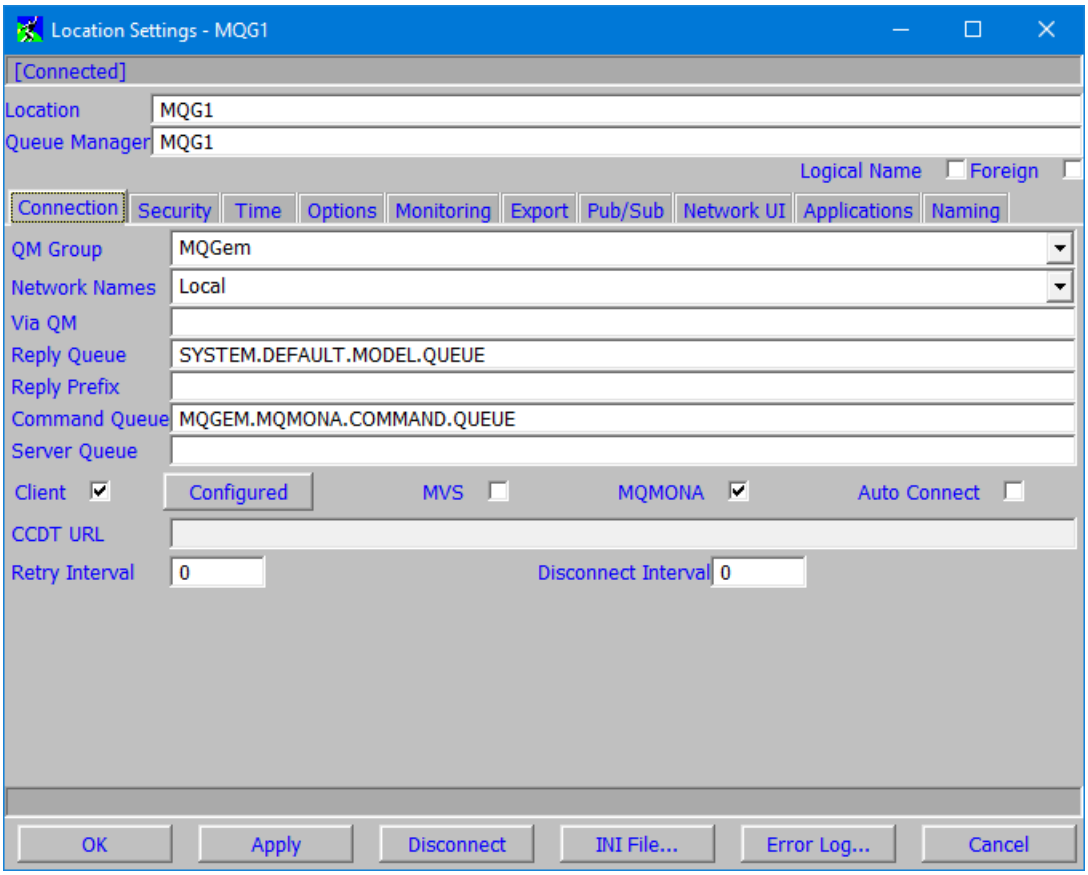
```
[13:38:42] MQGA001I MQGem Software MQMONA : Command Server Agent for IBM MQ
  Version V9.3.0 Build Date:Apr 28 2023
  (C) Copyright MQGem Software 2023,2023. (https://www.mqgem.com)
[13:38:42] MQGA221I MO71 Licence details: Valid - expires 09/09/23
[13:38:42] MQGA221I MQSCX Licence details: Valid - expires 09/09/23
[13:38:42] MQGA004I Parameters in effect:
  Queue Manager         : MQG1
  Input Queue           : MQGEM.MQMONA.COMMAND.QUEUE
  Reply Queue           : SYSTEM.DEFAULT.MODEL.QUEUE
  Actual Reply Queue    : MQGEM.MQMONA.6301D47321206C01
  Verbose Level         : Low detail
```

You can see the aforementioned input queue, `MQGEM.MQMONA.COMMAND.QUEUE`, and the reply queue it is using is a MODEL queue, the `SYSTEM.DEFAULT.MODEL.QUEUE`. The actual name of the dynamic queue created as its reply queue is also shown with a name that begins with `MQGEM.MQMONA`.

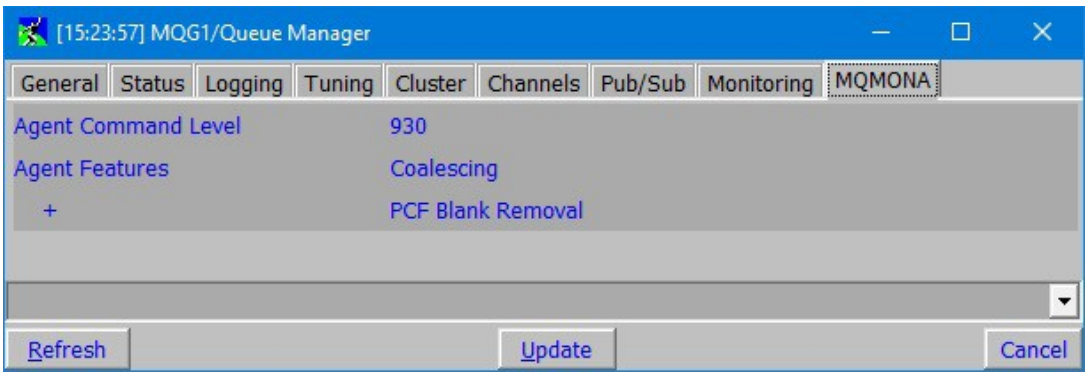
3.1. Configuring MO71 to use MQMONA

To use MQMONA with MO71, you must be using MO71 at V9.3.1 or higher.

In order to set up MO71 to use MQMONA, you need to make one small change to the location. Open the location dialog for the queue manager, and on the Connection tab, check the “MQMONA” check-box which will change the “Command Queue” field to `MQGEM.MQMONA.COMMAND.QUEUE`. Now press Apply and issue your commands through MO71 as normal.



When you connect to a queue manager using the MQMONA agent, viewing the queue manager dialog will show an additional tab with some MQMONA details.



3.2. Configuring MQSCX to use MQMONA

To use MQMONA with MQSCX, you must be using MQSCX at V9.3.1 or higher.

In order to set up MQSCX to use MQMONA, you can either add an extra parameter flag when you start up the program as follows:-

```
mqscx -m MQG1 -q MQGEM.MQMONA.COMMAND.QUEUE
```

Or alternatively you can use an `=conn` command like this.

```
=conn qm(MQG1) cmdq(MQGEM.MQMONA.COMMAND.QUEUE)
```

This can be very useful to put into a synonym along with client connection details and then you can quickly use the synonym instead of typing all this in. This is our recommended way of using MQMONA with MQSCX connections.

```
=syn(MQG1) cmd(=conn qm(MQG1) channel(MQGEM.SVRCONN)
conname(gemmvs1.mqgem.com(1414)) cmdq(MQGEM.MQMONA.COMMAND.QUEUE) compmsg(RLE))
```

Use of message compression is advised for channels where MQSC traffic rather than PCF traffic is used. For example, MO71 connected to a z/OS queue manager, or channels for MQSCX. See 5.3. String padding in MQSC on page 10 for a full explanation.

When you connect to a queue manager using the MQMONA agent, viewing the MQSCX help screen (using F1 or `=show scrn(help)` will show some MQMONA details.

```
Administering Queue Manager
---------------------------
Qm            : MQG1
Command Level: 930
Platform      : Unix
MQMONA Level  : 930
```

Chapter 4. Consolidation

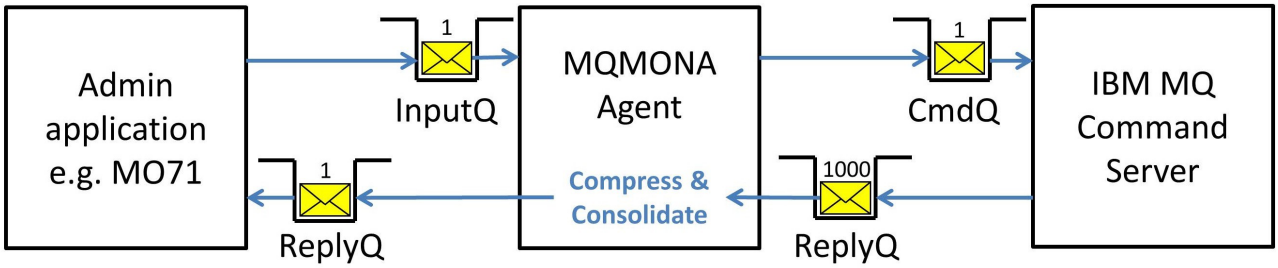
The MQMONA agent improves your IBM MQ administrative command experience when network latency is high, by consolidating the responses from the IBM MQ Command Server into fewer messages and thus reducing the number of MQGET calls an administrative application must make to retrieve all the answers.

4.1. IBM MQ Commands and Responses

When you issue a command to the IBM MQ Command Server, such as the MQSC command to display all queues, or the PCF command to Inquire Channel Status, you place a single command message on the `SYSTEM.ADMIN.COMMAND.QUEUE`[1], and provide the name of your ReplyToQ in the Message Descriptor. The IBM MQ Command Server sends the response messages that answer your command, to your nominated ReplyToQ.

The responses take the form of a single message for each item returned. This means if you displayed 1000 queues, your administrative application would be sent 1000 messages to retrieve from it's ReplyToQ.

Using MQMONA as an agent, means that your administrative application actually puts its command message onto the `MQGEM.MQMONA.COMMAND.QUEUE`, and MQMONA picks it up, changes the ReplyToQ and sends it on to the IBM MQ Command Server. This means that MQMONA receives the 1000 separate messages on its ReplyToQ, and when it does, it consolidates those reply messages together into several, or perhaps even just one larger reply message which it then puts to the original ReplyToQ for your administrative application.



By reducing the number of messages, we reduce the number of MQGET calls made over a client, which means we reduce the number of line turn-arounds, which are the expensive part of the system when on a network with high latency.

4.2. Consolidated Message Size

When MQMONA compresses and consolidates the responses from the IBM MQ Command Server into fewer, larger messages, it will keep the total size below a configured maximum. This is 1MB by default. You can change this by using configuration file setting `maxmsgsize` (see Chapter 12. Configuration File on page 17 for more details on how to set it).

[1]Or the SYSTEM.COMMAND.INPUT on z/OS

For client connected MQ administrative applications, MQMONA will inquire the reply queue to discover its Maximum Message Length and if it is discovered to be smaller than the currently set `maxmsgsize` it will reduce the size used for this specific reply queue. Message **MQGA304I** will be written to the error log when this happens as this suggests that this specific MQ administrative application is not getting the full benefit of MQMONA consolidation.

For MQ administrative applications that are connecting in “via” mode, the above check is not possible, and you must ensure that all queues along the route between MQMONA and the MQ administrative application have a Maximum Message Length that can accommodate messages up to the size specified by `maxmsgsize`.

Chapter 5. Compression

The MQMONA agent improves your IBM MQ administrative command experience when bandwidth is low, by removing unnecessary padding from your PCF response messages. The process of consolidation described in the previous chapter, also has a small compressive effect because the MQCFH header is not repeated for every response when they are consolidated into a single message. The MQCFH header is only 36 bytes long, but that's an additional saving of 30K when displaying 1000 queues.

5.1. IBM MQ string padding

The string fields in the response messages to an IBM MQ command are generally blank padded to their full length. Exceptions to this, thankfully, are the following 10K long fields.

- MQCA_TOPIC_STRING
- MQCACF_SUBSCRIPTION_NAME
- MQCACF_SUBSCRIPTION_USER_DATA
- MQCACF_SUB_SELECTOR

Object name fields are mostly 48 characters in length, but most people have queue names that are far shorter than that. The response message from the IBM MQ Command Server will always contain 48 characters even if over half of them are blanks.

5.2. MQMONA compression

MQMONA removes all these blanks at the end of strings, and adjusts the lengths in PCF messages accordingly. This has a fairly major compression impact. Consider the length of the response messages to just display the SYSTEM queues on a V9.3.0 queue manager.

The command server responds with 55 reply messages with a total size of 11.2 KB, which MQMONA consolidates and compresses into a single response message to the original application, with a total size of 6.86 KB. This gives a compression rate of 38%.

Compression rates will be higher when the responses have a high percentage of string fields. So, as an example, if you display all the SYSTEM queues and ask for the Backout Requeue Name; Cluster; Description; Process Name; and Stream Queue Name, then you get 55 reply messages with a total size of 26.27 KB, which MQMONA consolidates and compresses into a single response message with a total size of 12.9KB, a compression rate of 50%.

5.3. String padding in MQSC

MQSC response messages from z/OS are fully blank padded too. However, because the responses to MQSC commands are not quoted, these blanks are actually important to parsing the output. For this reason MQMONA does not strip off the blanks from MQSC responses messages, and we advise the use of message compression on client connections to reduce the client payload further.

```
ALTER CHANNEL(MQGEM.SVRCONN) CHLTYPE(SVRCONN) COMPMSG(RLE)
```

Chapter 6. Running MQMONA as a SERVICE

You can setup MQMONA to run as an IBM MQ SERVICE object so that it is always started up when your queue manager is started, and is ready and waiting to process your command messages.

Here is an example command showing how to configure this.

```
DEFINE SERVICE(MQGEM.MQMONA) SERVTYPE(SERVER) CONTROL(QMGR) +
       STARTCMD('c:\MQGem\mqmona.exe') +
       STARTARG('-m +QMNAME+ -f C:\MQGem\MQMONA.CFG -k') +
       STOPCMD('c:\MQGem\mqmona.exe')  STOPARG('-m +QMNAME+ -c STOP') +
       DESCR('MQGem Command Server Agent')
```

This service object can now be started and stopped using the IBM MQ commands:-

```
START SERVICE(MQGEM.MQMONA)
```

and

```
STOP SERVICE(MQGEM.MQMONA)
```

Additionally, you can check that it is running using the following command:-

```
DISPLAY SVSTATUS(MQGEM.MQMONA)
```

Chapter 7. Triggering

The MQMONA Agent can be trigger started. This may be useful instead of, or as well as, running it as a SERVICE which was described in the previous chapter. This chapter shows a full working example of triggering MQMONA. You may already have a trigger monitor in place, and so can skip some of the steps that follow.

7.1. Set up a trigger monitor

Define an initiation queue, and a service object to automatically run your trigger monitor, then start it. The `DEFINE SERVICE` command example below is for Windows, the bin directory is in a slightly different place on Unix, please adjust the `STARTCMD` and `STOPCMD` values accordingly (remove the '64').

```
DEFINE QLOCAL(MQGEM.INITQ) DESCR('Initiation Queue')
```

```
DEFINE SERVICE(MQGEM.TRIGGER.MONITOR) SERVTYPE(SERVER) CONTROL(QMGR) +
       DESCR('Trigger Monitor Service') +
       STARTCMD('+MQ_INSTALL_PATH+bin64/runmqtrm') +
       STARTARG('-m +QMNAME+ -q MQGEM.INITQ') +
       STOPCMD('+MQ_INSTALL_PATH+bin64/amqsstop') +
       STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+') +
       STDOUT('+MQ_Q_MGR_DATA_PATH+errors/MQGemTrigMonOut.txt') +
       STDERR('+MQ_Q_MGR_DATA_PATH+errors/MQGemTrigMonErr.txt')
```

```
START SERVICE(MQGEM.TRIGGER.MONITOR)
```

7.2. Describe MQMONA, the triggered application

It is assumed at this point, that you have already successfully run MQMONA manually from a command line and that the queue MQMONA requires to run has already been defined as described in Chapter 3. Getting Started on page 5. The example `DEFINE PROCESS` command below is for Windows. Adjust the path of the executable and configuration file for your system.

```
DEFINE PROCESS(MQGEM.MQMONA.PROC) DESCR('MQGem Command Server Agent') +
       APPLICID('c:\MQGem\mqmona.exe') USERDATA('-f c:\MQGem\MQMONA.CFG -k')
```

```
ALTER QLOCAL(MQGEM.MQMONA.COMMAND.QUEUE) INITQ(MQGEM.INITQ) +
      PROCESS(MQGEM.MQMON.PROC) TRIGGER TRIGTYPE(FIRST)
```

Now that you have the trigger configuration in place, you can easily test that the triggering is correctly set up by sending a command message to MQMONA. If you get your responses back, then you know it worked.

When MQMONA is trigger started, it will report this in its error log with message **MQGA003I**.

Triggering is useful to combine with the configuration file keyword `discint`, whose default value is 60 minutes. This means the MQMONA program will gracefully end after this time if there is no work for it to do.

Chapter 8. Timing out commands

There are error conditions when the IBM MQ Command Server will never reply, often when the original command message has been put to the Dead-letter Queue.

In this situation, MQMONA will wait for the response to arrive for a configurable period of time, which is by default 10 seconds. You can adjust this interval using the `maxage` keyword in the configuration file.

If this time period elapses for a command without any response from the IBM MQ Command Server, MQMONA will take the following actions.

- It will send a response message to the original application to tell it that there was no response from the IBM MQ Command Server.
- It will write message **MQGA303E** to the error log to record the message ID of the command that has now been discarded.

Chapter 9. Status Reporting

MQMONA will periodically report some status about how much work it has done, and what it has achieved for you. This chapter details what those status values are.

These status reports will be written to the error log file, by default, every 60 minutes.

You can change this interval using the `reportint` keyword in the configuration file.

Additionally, they can be requested at any time by sending a simple command, REPORT, to MQMONA. See Chapter 11. Simple Commands on page 16 for details.

Status	Meaning
Commands Received	The number of commands (including MQSC, PCF and simple string commands) received on the MQMONA input queue
Command Bytes Received	The command messages received totalled this many bytes
Replies Received	The number of reply messages received from the IBM MQ Command Server
Reply Bytes Received	The reply messages received totalled this many bytes
Responses Sent	The number of responses sent to original application reply queues. This number reflects the messages after they have been consolidated.
Response Bytes Sent	The response messages sent totalled this many bytes. This number reflects the messages after they have been compressed.
Compression Achieved	A comparison between Response Bytes Sent and Reply Bytes Received to show how much compression has been achieved.
Failed Puts	The number of puts which were not successfully.
Put Successfully Retried	The number of puts, that initially failed, but after some number of retries were eventually successful.

Chapter 10. Parameters

There are a number of parameters that can be passed to MQMONA to control the main behaviour you need. Additionally there is a configuration file containing optional attributes for more detailed control of various behaviours. This is described in the next chapter.

10.1. Parameters Flags

Parameters are passed to the program as flags on the command line. The following parameters are available:

Flag	Meaning
-c	<Simple Command> This parameter can be used to put a very simple string command onto the MQMONA input queue. See Chapter 11. Simple Commands on page 16 for the list of valid commands. The program puts the string command onto the queue and then immediately ends.
-f	<configuration file name> This parameter provides the name of the configuration file. If not specified, MQMONA will look for a file called MQMONA.CFG in the same directory as the program (Windows and Linux) or in the current directory (AIX and z/OS UNIX). For more details about this configuration file, see the next chapter.
-k	Run as a SERVICE
-m	<Queue Manager name> Specifies the name of the Queue Manager to connect to if not the default.
-q	<input queue name> Specifies the input queue to read from. By default this will use a queue called MQGEM.MQMONA.COMMAND.QUEUE.
-v	Verbose Level Controls which informational or error messages are written to the log. Review Chapter 15. Messages on page 21 to see which messages are written at which verbose levels. Value can be Quiet, Low, Medium, or High. Only the first character need be provided.

Chapter 11. Simple Commands

You can send some very simple string commands to MQMONA.

You can do this in a number of ways.

- Put a simple string format message to the MQMONA input queue, which is, by default `MQGEM.MQMONA.COMMAND.QUEUE`, using
  - A simple MQ sample program such as `amqsput`, or the MO71 Put Message dialog
  - the MQMONA program itself with the -c flag
- On z/OS, you can also use the MVS `MODIFY` command, putting the command in as the `APPL=` string. e.g.

```
/MODIFY MQG1MONA,APPL=REPORT
```

- On z/OS, you can also use the MVS `STOP` command, e.g.

```
/STOP MQG1MONA
```

Valid commands are:-

- `STOP`
  MQMONA will end when it receives this command
- `REPORT`
  MQMONA will write out a status report when it receives this command. See Chapter 9. Status Reporting on page 14.

Chapter 12. Configuration File

In addition to the main parameters shown in the previous chapter, the configuration file can be used to supply additional optional configuration of how the program operates.

12.1. Example Configuration File

To demonstrate the format of the MQMON configuration file, here is an example one.

```
################################################################
# MQMONA configuration file
################################################################
reportint = 90                      # 1 hour 30 minutes
replyqinq = no                      # This one is a yes|no switch
hobjttl = 60
logpath = "C:\temp\+QMNAME+"
tempqpfx = "MQGEM.MQMONA.MQGEMUSR.*"
discint = 5
```

12.2. Configuration File keywords

Keywords are specified in the file with an equals sign and then the value, unless the keyword is described as a switch with no value.

Keyword	Meaning
`commandq`	The IBM MQ Command Server queue name. By default MQMONA will use `SYSTEM.ADMIN.COMMAND.QUEUE` unless the queue manager is discovered to be a z/OS platform queue manager in which case it will use `SYSTEM.COMMAND.INPUT` If you need a different queue than the above, you can use this keyword. If you do need to use this, please get in touch to let us know so that we can improve MQMONA to recognise other command queue names.
`replyq`	The reply queue to use. This can be a local queue, or a model queue. If it is a model queue, MQMONA will create a dynamic queue with the prefix `MQGEM.MQMONA.*` To use a different prefix, please use `tempqpfx` described below. By default MQMONA will use `SYSTEM.DEFAULT.MODEL.QUEUE` for this queue.
`tempqpfx`	The prefix used when MQMONA creates a dynamic reply queue from the model queue name provided in the `replyq` keyword. If not specified, MQMONA will use a prefix of `MQGEM.MQMONA.*`
`reportint`	The interval, in minutes, that status is reported to the log. You can also force status to be reported with a simple string command. See the previous chapter for more details.
`compress`	This `yes|no` switch can be used to turn off compression which is on by default. Read more about compression in Chapter 5. on page 10.

`coalesce`	This `yes|no` switch can be used to turn off consolidation which is on by default. Read more about consolidation in Chapter 4. on page 8.
`replyqinq`	This `yes|no` switch can be used to disallow MQMONA from inquiring the ReplyToQ in a bid to discover the Maximum Message Length.
`logpath`	The location where the rotating error log files will be written. If not specified, MQMONA will use the same directory as the program (Windows and Linux) or in the current directory (AIX and z/OS UNIX). A special value `+QMNAME+` can be used as a substitution for the queue manager name, allowing the ability to use the same configuration file for multiple queue managers. An example of this is shown in the example configuration file above. This keyword is not used if MQMONA is running in batch or as a started task on z/OS as error messages are written to the JESMSGLG instead.
`discint`	If MQMONA has found nothing to do for this time period, in minutes, then it will end and await being trigger started again should any more command messages arrive. The default value is 60 minutes. This keyword only applies if MQMONA was triggered started. It is ignored otherwise.
`retrycount`	The number of times to re-attempt a failed MQPUT of a message before giving up. The default is 10 times.
`retryint`	The time period, in seconds, between re-attempted MQPUTs of failed messages. The default value is 1 second.
`maxage`	The maximum time period, in seconds, that MQMONA will hold onto a request waiting for an answer from the IBM MQ Command Server. The default value is 10 seconds. Once this time period is exceeded and no response has been received, MQMONA will throw away the request, writing message **MQGA303E** to the log.
`maxmsgsize`	The maximum size of response message, in bytes, that MQMONA will create when consolidating the replies from the IBM MQ Command Server. The default value is 1MB (1048576 bytes). Read more about consolidation in 4 Consolidation on page 8.
`hobjttl`	The time period, in seconds, that an object handle will remain in MQMONAs cache without having been used. After this time the handle will be closed. The default is 300 seconds.
`numlogfiles`	The number of rotating error log files MQMONA will write to. The default value if this is not specified is three.
`maxlogsize`	The maximum size, in bytes, that an error log file can grow to before MQMONA will start writing to a new error log file. The default value is 1MB.
`logfailfatal`	This `yes|no` switch controls whether MQMONA should continue running if it has a failure when working with its error log files. By default it will stop running.

Chapter 13. Performance Comparisons

Clearly the benefit you receive from using MQMONA will depend hugely on the latency and speed of your network.

We deliberately tested this on a poorly performing network connection, hopefully a much worse connection than you have to deal with.

So, the numbers that follow are simply an example where we have a particularly high latency to a z/OS queue manager on the other side of the planet (New Zealand to United Kingdom) and low bandwidth. We measured the network latency reported by TCP/IP ping which averages around 315 ms. FTP moving large files gave a throughput of around 56 Kbytes/sec.

The tests below are all retrieving a list of 1000 alias queue definitions with their Inhibit Put and Inhibit Get attributes using MO71 over a non-TLS client connection. When connecting to a z/OS queue manager using MO71, the command type in use is MQSC.

#	Test Description	# Responses	Time Taken
1	MQ Client Read-ahead is disabled	1000	5m 48s
2	MQ Client Read-ahead is enabled	1000	19s
3	MQMONA is used with maxmsgsize = 100,000	2	9s
4	MQMONA is used with maxmsgsize = 900,000	1	6s

These tests also illustrate the benefits that read-ahead can bring you when using a network with high latency. Hopefully queues such as the `SYSTEM.DEFAULT.MODEL.QUEUE` which are often used to create dynamic reply queues for administrative tools already have `DEFREAD(ON)` set, so you will not experience the numbers in test #1!

MQ Client Read-ahead can only go so far though, and that's where MQMONA can offer additional help. Rather than speeding up how fast an MQ Client can consume a 1000 small messages, MQMONA changes the pattern so that the MQ Client is consuming a very small number of much larger messages. The comparison between test #2 (which is where we hope everyone is at) and test #4 is a three times improvement.

Chapter 14. Error reporting

On z/OS, when running as a batch job or a started task, MQMONA will write informational and error messages to the JESMSGLG. In all other environments (z/OS UNIX, AIX, Linux, Windows) these messages are written to a rotating set of error log files.

These files will be created either in a default location or a location you specify.

The default location is in the same directory as the executable when running on Windows or Linux and the current directory when running on AIX on z/OS UNIX.

You can specify a location to over-ride this default by using the `logpath` keyword in the configuration file. This keyword has a special substitution value `+QMNAME+` to allow the creation of a configuration file that can be used by multiple queue managers.

The error log file will be named `MQMONA_Log<nn>.txt` where *nn* is the number of the log file. By default there will be three rotating error log files, very similar to IBM MQ. When the first error log gets full, then it will be renamed to the next number and a new error log will be created.

The number of error log files can be configured using the `numlogfiles` keyword in the configuration file. The size at which an error log is considered to be full can be configured using the `maxlogsize` keyword.

The current error log being written to is always called `MQMONA_Log01.txt`.

Chapter 15. Messages

MQMONA writes a number of informational and error messages to the error log files, or JESMSGLG on z/OS. This chapter details those messages.

Some messages are only shown when you are running MQMONA with a specific verbosity level (-v parameter flag). This is detailed in each message.

MQGA001I MQGem Software MQMONA : Command Server Agent for IBM MQ
Version *version-number* Build Date: *date*
(C) Copyright MQGem Software 2023,*current-year*. (https://www.mqgem.com)

A welcome banner message showing the *version-number* and *date* the program was built.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA002I Current Date: *date*

When an error message is written to the error log on a new date, this message precedes it in the file. This message is not written to the JESMSGLG as every message has the date recorded.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA003I Program Triggered

The MQMONA program was started as a result of a trigger message.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA004I Parameters in effect:
Queue Manager : *qmgr-name*
Input Queue : *input-q-name*
Reply Queue : *reply-q-name*
Actual Reply Queue : *actual-reply-q-name*
Verbose Level : *verbose-level*

A brief report of the parameters in use. The *actual-reply-q-name* will be different from the *reply-q-name* if the *reply-q-name* was the name of a MODEL queue.

Minimum Verbosity Level: Always shown

User Action Required: If these do not match what you expect, ensure you have used the correct parameter flags.

MQGA005I Default overrides from file '*config-file*' in effect: *details*

The *details* will show the list of overrides read from the *config-file*. If there are keywords in the file that do not apply to the current mode of running, for example attributes that only apply if the program was triggered, they will not be shown.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA006E Usage: MQMONA <Optional flags as below>

```
  [-m <Queue Manager Name>]
  [-q <Input Queue Name>  ]
  [-f <Config File Name>  ]
  [-k                     ] Running as a service
  [-c <Simple command>    ]
  [-v {Quiet|Low|Med|High}] Verbosity level
```

The MQMONA program usage statement.

Minimum Verbosity Level: Always shown

User Action Required: If you did not request to see the usage statement, this means that you have incorrectly specified some parameter flags on your invocation of the program. Review any instances of error message **MQGA007E** to discover what was wrong.

MQGA007E Unrecognised parameter '*parm-flag*'

You have specified an unrecognised parameter flag 'parm-flag'. Review the invocation of MQMONA and correct any flags that are not supported by MQMONA. A full description of the parameter flags can be read in Chapter 10. on page 15.

Minimum Verbosity Level: Always shown

User Action Required: Correct the parameter flags.

MQGA020I Disconnect Interval, *n* minutes, expired

MQMONA is stopping because there has been no work for it to do for at least *n* minutes. Disconnection only takes place when MQMONA is trigger started, so it will now be in a position to be restarted by a trigger message when more work arrives on the queue.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA021I STOP command received

MQMONA is stopping because it received a STOP command, either from a simple string message contain the text “STOP” on it's input queue or from an MVS STOP command.

Minimum Verbosity Level: Always shown

User Action Required: None – unless you did not expect a STOP command to have been issued.

MQGA080I MQMONA Status Report

This multi-line message will report a number of status attributes. For full details see Chapter 9. Status Reporting on page 14. This message is written at regular timed intervals; when prompted to do so with a REPORT simple command; and just before ending.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA099I MQMONA Agent shutting down

The MQMONA agent is shutting down. See preceding messages to understand the reason why it is shutting down.

Minimum Verbosity Level: Always shown

User Action Required: None

MQGA100E Error loading MQAPI mqm.lib rc=*return-code*

MQMONA failed to successfully load the MQ libraries. The *return-code* provides more information.

If the reason for the failure is not obvious, you can set the environment variable, `MQACCESS_DEBUG` for more information.

Minimum Verbosity Level: Always shown

User Action Required: Ensure the MQ environment is set up correctly, for example by using `setmqenv` or by having a primary installation.

MQGA101E MQCONN to queue manager '*qmgr*' failed with reason *mqrc* (*string*)

MQMONA was unable to connect to queue manager *qmgr*, The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. Remember that there is often additional information in the queue manager error log `AMQERR01.LOG` especially if the failure is `MQRC_NOT_AUTHORIZED`.

MQGA102E MQOPEN of queue manager object failed with reason *mqrc* (*string*)

MQMONA was unable to open the queue manager object for inquire. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. Remember that there is often additional information in the queue manager error log `AMQERR01.LOG` especially if the failure is `MQRC_NOT_AUTHORIZED`.

MQGA103E MQINQ of queue manager object failed with reason *mqrc* (*string*)

MQMONA was unable to inquire the queue manager object. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. Remember that there is often additional information in the queue manager error log `AMQERR01.LOG`.

MQGA104E MQOPEN of queue '*q-name*' failed with reason *mqrc* (*string*)

MQMONA was unable to open one of the queues it needs to operate, *q-name*. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. Remember that there is often additional information in the queue manager error log `AMQERR01.LOG` especially if the failure is `MQRC_NOT_AUTHORIZED`.

MQGA105E MQCB for queue '*q-name*' failed with reason *mqrc* (*string*)

MQMONA was unable to set up a call-back function to read from queue *q-name*. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. Remember that there is often additional information in the queue manager error log `AMQERR01.LOG`.

MQGA106E MQCTL on queue manager '*qmgr*' failed with reason *mqrc* (*string*)

MQMONA was unable to either start or stop the message consumers. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. Remember that there is often additional information in the queue manager error log `AMQERR01.LOG`.

MQGA107E MQPUT to queue '*q-name*' failed with reason *mqrc* (*string*)

MQMONA was unable to put a message to the queue *q-name*. The reason code *mqrc* and its *string* equivalent provide more information about the failure. This message will be seen after message retry has been attempted and all retry attempts have been exhausted. No more attempts will be made to retry this message and it will be discarded.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem.

MQGA108E MQPUT to queue '*q-name*' failed with reason *mqrc* (*string*) - Retry #*retry-attempt* after *retry-interval* seconds

MQMONA was unable to put a message to the queue *q-name*. The reason code *mqrc* and its *string* equivalent provide more information about the failure. This put will be reattempted after *retry-interval* seconds have passed. The next put attempt will be number *retry-attempt*.

If all retry attempts are exhausted, error message **MQGA107E** will be written to the error log.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem.

MQGA109E MQCLOSE of queue '*q-name*' failed with reason *mqrc* (*string*)

MQMONA attempted to close the object handle of queue *q-name* in the cache, and was unable to. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

The handle will still be removed from the cache.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem.

MQGA110E MQINQ of reply queue '*q-name*' failed with reason *mqrc* (*string*)

MQMONA attempted to inquire details about reply queue *q-name*, and was unable to. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem. To stop MQMONA from inquiring reply queues, use the `replyqinq` switch in the configuration file. See Chapter 12. Configuration File on page 17 for more details.

MQGA120E Queue 'q-name' has been GET(INHIBITED)

MQMONA has received notification that queue *q-name* has been inhibited for get. This will cause MQMONA to shut down.

Minimum Verbosity Level: Always shown

User Action Required: None – unless you did not expect the queue to have been get inhibited.

MQGA199E MQMONA received reason code *mqrc* (*string*)

The MQMONA consumer has received reason code *mqrc*. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Look up the *mqrc* in IBM Docs, or with the IBM MQ `mqrc` tool. Follow the guidance in IBM Docs to solve the problem.

MQGA201E Can not allocate *n* bytes for a message!

MQMONA is trying to allocate a memory buffer to store an MQ message in, but was unable to allocate a buffer of *n* bytes in length.

Minimum Verbosity Level: Always shown

User Action Required: Unless you are running in a very memory restricted environment, report this issue to MQGem Software support.

MQGA202E Can not allocate *n* bytes for a '*struct-name*' structure

MQMONA is trying to allocate a memory buffer to store s structure called *struct-name* in, but was unable to allocate a buffer of *n* bytes in length.

Minimum Verbosity Level: Always shown

User Action Required: Unless you are running in a very memory restricted environment, report this issue to MQGem Software support.

MQGA203E Unable to open config file '*file-name*' rc=*errno* (*string*)

MQMONA was unable to open the provided configuration file, file-name. The reason code *errno* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: If the *string* equivalent of the *errno* does not immediately make the reason for the failure obvious, look up the *errno* in your platform documentation and follow the guidance there to solve the problem.

MQGA204E Unable to open error log file '*file-name*' rc=*errno* (*string*)

MQMONA was unable to open the error log file, *file-name*. The reason code *errno* and its *string* equivalent provide more information about the failure.

MQMONA will shut down. To force MQMONA to continue in this situation, use the `logfailfatal` switch in the configuration file. See Chapter 12. Configuration File on page 17 for more details.

Minimum Verbosity Level: Always shown

User Action Required: If the *string* equivalent of the *errno* does not immediately make the reason for the failure obvious, look up the *errno* in your platform documentation and follow the guidance there to solve the problem.

MQGA205I Deleting old error log file 'file-name'

MQMONA is rotating its error log files and is deleting the oldest one, file-name.

Minimum Verbosity Level: High – this informational message is only seen when running with high verbosity.

User Action Required: None, unless your error log files are rotating too frequently, in which case consider setting their maximum size to a larger value. For more details read Chapter 14. Error reporting on page 20.

MQGA206E Unable to delete old error log file '*file-name*' rc=errno (*string*)

MQMONA was unable to delete the error log file, *file-name*. The reason code *errno* and its *string* equivalent provide more information about the failure.

MQMONA will shut down. To force MQMONA to continue in this situation, use the `nologfailfatal` switch in the configuration file. See Chapter 12. Configuration File on page 17 for more details.

Minimum Verbosity Level: Always shown

User Action Required: If the *string* equivalent of the *errno* does not immediately make the reason for the failure obvious, look up the *errno* in your platform documentation and follow the guidance there to solve the problem.

MQGA207I Renaming old error log file '*file-name*' to '*new-file-name*'

MQMONA is rotating its error log files and is renaming an old error log file, *file-name*, to have a new name, *new-file-name*.

Minimum Verbosity Level: High – this informational message is only seen when running with high verbosity.

User Action Required: None, unless your error log files are rotating too frequently, in which case consider setting their maximum size to a larger value. For more details read Chapter 14. Error reporting on page 20.

MQGA208E Unable to rename old error log file '*file-name*' to '*new-file-name*' rc=*errno* (*string*)

MQMONA was unable to rename the error log file, *file-name* to have a new name, *new-file-name*. The reason code *errno* and its *string* equivalent provide more information about the failure.

MQMONA will shut down. To force MQMONA to continue in this situation, use the `logfailfatal` switch in the configuration file. See Chapter 12. Configuration File on page 17 for more details.

Minimum Verbosity Level: Always shown

User Action Required: If the *string* equivalent of the *errno* does not immediately make the reason for the failure obvious, look up the *errno* in your platform documentation and follow the guidance there to solve the problem.

MQGA209E Error log File access with *func* failed. rc=*errno* (*string*)

Access on the current error log file with function *func* failed. The reason code *errno* and its *string* equivalent provide more information about the failure.

MQMONA will shut down. To force MQMONA to continue in this situation, use the `logfailfatal` switch in the configuration file. See Chapter 12. Configuration File on page 17 for more details.

Minimum Verbosity Level: Always shown

User Action Required: If the *string* equivalent of the *errno* does not immediately make the reason for the failure obvious, look up the *errno* in your platform documentation and follow the guidance there to solve the problem.

MQGA210I Logging to '*file-name*'

This informational message is only written to `stdout` and is simply there to inform you the full location of the error log file, *file-name*, MQMONA is using to write out informational and error messages.

Minimum Verbosity Level: Always shown

User Action Required: None.

MQGA211E __console() failed, rc=*errno* (*string*)

MQMONA used the system function `__console()` which failed. The reason code *errno* and its *string* equivalent provide more information about the failure. This error message is only seen on MVS.

Use of MVS `MODIFY` or MVS `STOP` commands will not be operational on this instance of MQMONA.

Minimum Verbosity Level: Always shown

User Action Required: Report this error to MQGem Software support.

MQGA212E MQMONA does not recognise the simple command '*string*'

A simple command has been entered using the MVS MODIFY command, but it was not one that MQMONA recognised. The command entered was *string*.

Minimum Verbosity Level: Always shown

User Action Required: Use a recognised simple command as detailed in Chapter 11. Simple Commands on page 16.

MQGA220E No valid licence found

MQMONA searched for MQGem Software licenses as described in Chapter 2. Licensing on page 4, but did not find any valid licences.

MQMONA will run in pass-through mode where all messages that arrive on the input queue will be passed straight onto the IBM MQ Command Server queue and MQMONA will not be involved with them further.

Minimum Verbosity Level: Always shown

User Action Required: Ensure your valid MQGem Software licences can be found by MQMONA.

MQGA221I *product-name* Licence details: *details*

MQMONA found a valid licence for *product-name*, and will allow messages from that software product to be consolidated and compressed before returning the replies to the application. Licence details are shown in *details*.

Minimum Verbosity Level: Low – this informational message is seen with a verbosity of at least low.

User Action Required: None.

MQGA222E Application '*product-name*' is unlicensed - request for ReplyQ '*q-name*' passed through

A command message from *product-name* arrived on the MQMONA input queue using a reply queue name of *q-name*. However, *product-name* is unlicensed so the command message was passed through to the IBM MQ Command Server queue and MQMONA will not be involved with that command further.

Minimum Verbosity Level: Always shown

User Action Required: Ensure your valid MQGem Software licences can be found by MQMONA.

MQGA301E Can not compress PCF type *n*

MQMON can compress all the PCF types that are found in IBM MQ PCF commands. If a new PCF type is used in a command that MQMONA does not currently cater for, no compression will take place for that command attribute, but the command responses will still be returned to the application.

Minimum Verbosity Level: Always shown

User Action Required: If this error is still seen when using the latest version of MQMONA, contact MQGem Software support to have an improvement made to MQMONA to handle the reported PCF type *n*.

MQGA302E Unrecognised reply received MsgId: *message-id*

MQMONA has received a reply message for a command it has no record of sending. This is most likely to happen if the response to the command was delayed in some way, for example because of a retrying channel, or because it spent some time on a Dead-letter queue. MQMONA does not remember about the commands it has issued forever, instead ageing them out after a configurable number of seconds.

Minimum Verbosity Level: Always shown

User Action Required: You should see the same message id reported in an instance of error message **MQGA303E** which will record the Reply queue that was linked to this message. Determine whether the administration tool using that Reply queue is using a “via connection”, i.e. one going across MCA channels, and consider increasing the time period after which commands are aged out of MQMONA's memory. You can do this using the `maxage` keyword in the configuration file.

If this is not a message using a via connection, investigate whether reply messages went via a Dead-letter queue.

If this message is regularly seen, consider increasing `maxage` as noted above.

MQGA303E Ageing out request for ReplyQ '*q-name*' issued at *time* MsgId: *message-id*

MQMONA is ageing out, that is deliberately forgetting about, a command request that was issued at *time* but for which no response have yet been returned. This command used Reply queue *q-name* and had a message of *message-id*.

This is most likely to happen if the response to the command was delayed in some way, for example because of a retrying channel, or because it spent some time on a Dead-letter queue. If a response message later turns up for a command MQMONA has deliberately forgotten about, error message **MQGA302E** will be reported and you can tie this error message to that one using the message id.

Minimum Verbosity Level: Always shown

User Action Required: See error message **MQGA302E**

MQGA304I ReplyQ '*q-name*' (*deftype*) has smaller MaxMsgLen (*q-max-len*) than configured maxmsgsize (*mqmona-max-len*). Consolidation size reduced accordingly

MQMONA inquired reply queue *q-name* and discovered that its Maximum Message Length was smaller than the MQMONA configured size, *mqmona-max-len*, set using `maxmsgsize` (see Chapter 12. Configuration File on page 17 for more details on how to set it). As a result, MQMONA will use a smaller consolidation size for messages sent to this reply queue.

Minimum Verbosity Level: Always shown

User Action Required: The application using reply queue *q-name* will continue to operate successfully but will not be getting the same efficiency savings as other applications whose reply queues have a larger Maximum Message Length. It is suggested that you increase the Maximum Message Length of the reply queue to be at least that which is used by MQMONA, *mqmona-max-len*. If the reply queue is a dynamic queue (which is shown by *deftype*) then you will have to make the change to the MODEL queue it was created from.

MQGA401I Received PCF '*pcf-command*' command message of length *n* bytes

MQMONA has received a PCF command message for command *pcf-command* on its input queue. It has passed this message onto the IBM MQ Command Server queue after changing the reply queue in the MQMD.

Minimum Verbosity Level: Medium – this informational message is only seen when running with medium or high verbosity.

User Action Required: None.

MQGA402I Received MQSC command message of length *n* bytes

MQMONA has received an MQSC command message on its input queue. It has passed this message onto the IBM MQ Command Server queue after changing the reply queue in the MQMD.

Minimum Verbosity Level: Medium – this informational message is only seen when running with medium or high verbosity.

User Action Required: None.

MQGA403I Received reply message of length *n* bytes

MQMONA has received a reply message on its reply queue. It will compress and consolidate this reply message with others for the same command before sending a large reply message onto the application's reply queue.

Minimum Verbosity Level: Medium or High – this informational message is only seen when running with high verbosity for commands that have multiple replies. It will be seen at Medium verbosity when a command only has one reply in total.

User Action Required: None.

MQGA404I Received *count* reply messages of total length *n* bytes

MQMONA has received a *count* reply messages on its reply queue. It will compress and consolidate these reply messages with others for the same command before sending a large reply message onto the application's reply queue.

Minimum Verbosity Level: Medium – this informational message is only seen when running with medium verbosity. When running at High verbosity, message **MQG403I** is issued multiple times, once for each reply message received.

User Action Required: None.

MQGA405E PCF Message failure *mqrc* (*string*)

While processing a PCF format IBM MQ message, a failure occurred. The reason code *mqrc* and its *string* equivalent provide more information about the failure.

Minimum Verbosity Level: Always shown

User Action Required: Report the problem to MQGem Software support.

End of Document