mq

The WebSphere® MQ for UNIX Administration Tool

Version 2.6

14. Mai 2013

® WebSphere MQ is a registered trademark by IBM

Contents

1 Introduction

The utility **mq** has been designed to make an MQ administrator's life much easier and more reliable. Are you tired of typing a command for every single entry when manipulating a set of MQ objects the same way? Do you think **runmqsc** should allow for wild cards anywhere in an object name and not only at the end? Are you looking for a cluster agent, safely (and we mean really safely!) controlling your MQ resources? Does MQ for Solaris strain your nerves once in a while, because you cannot clean the shared memory, even though all queue managers have been stopped? Are you longing for a nice and short command to inspect a queue manager's error logs that are hidden down an endless directory path? Would you like to start and stop all queue managers on a system with one short command you can type in one second? Would you like **endmqm** to clean up your memory automatically? If you answered any of the above questions with yes, **mq** is the utility for you.

The program **mq** is a shell script calling interactively or in batch mode utilities of *IBM WebSphere MQ* and of the UNIX shell. It supports MQ administration from the shell that is much more powerful and comfortable than the original tools by IBM. The script **mq** is very flexible with regard to error logging, is robust and provides a suitable return code at a user defined timeout – regardless of the state of the system resources (e.g. SCSI timeouts of disks). Hence it can easily and reliably be used as a cluster agent (e.g. for a Veritas cluster). It is suitable for remote administration, including z/OS queue managers, almost as if the remote queue managers were on the local system.

The UNIX derivatives currently supported are: Solaris, AIX, HP-UX, and Linux. Along with the script **mq**, a man page is distributed, which basically contains the technical information supplied by the documentation at hand (cp. section 9 *Files and Installation Instructions*).

Typesetting conventions used throughout this documentation: Commands, options, and file names related to the UNIX shell are generally printed in the type face sans serif, i.e. file names and command line options are printed regular, commands and utilities in **bold face**, the actions of the **mq** command with underscore, and entries to be supplied by the user are printed *italic*.

2 Synopsis

The script **mq** is called as follows:

mq *action* [–acfnrsvxzILMPSTX] [-d *information*] [–e '*command*'] [–o '*command*'] [–w *timeout*] [–C[:*pattern*]] [*qmgr* | *pattern* …]

mq *mqsc* *object* [–a] [–e '*command*'] [–o '*command*'] [–w *timeout*] [*qmgr* | *pattern* …]

where

action	=	{ cl | kill | log[2|3] | ps | pw | run | start | stop }, see section 3 *Description of Actions.*
command	=	shell command for logging, see section 5 *Command Line Options*.
mqsc	=	{ alt[er] | clear | delete| dis[play] | ping | refresh | reset | resolve | resume | suspend | qh[andle] }, WebSphere MQ command as for **runmqsc**. Note the non-IBM keyword qhandle, cp. *object* below and section 4 *WebSphere MQ Commands*

object	=	WebSphere MQ object specification of the form (in quotes only if it contains spaces or special characters): '*type* [(*pattern*)] [*attribute* [(*pattern*)] [...]]', see section 4 *WebSphere MQ Commands* for details.
pattern	=	pattern to be matched with queue manager names, object names, attributes, or processes; case insensitive full regular expression either according to Perl or, if the original default has been changed, according to the **egrep(1)** and **regexp(5)** man pages (case sensitive for *actions* ps/pw). See section 6.6 *Default Definitions,* paragraph f) for the details of the configuration.
qmgr	=	queue manager name (case insensitive – required and case sensitive for *action* kill). Instead of a full *qmgr* name, a case-insensitive *pattern* may be specified, that identifies the queue manager(s). If with *actions* cl, ping, ps/pw, and start/stop more than one *pattern* or *qmgr* name is supplied, all queue managers or processes matching either of them are addressed. The *actions* kill, log, and run, and *mqsc* act on one single queue manager.
timeout	=	timeout in seconds (cp. section 5 *Command Line Options* below)

3 Description of Actions

The utility **mq** carries out *action* on the resources specified by the subsequent arguments:

- starts, stops or kills WebSphere MQ queue managers,
- displays a *qmgr*'s error log or WebSphere MQ processes (ps/pw),
- pings or runs WebSphere MQ commands for one *qmgr* (by **runmqsc**),
- alters, clears, deletes, or displays WebSphere MQ objects, or
- displays the number of active clients per *qmgr.*

3.1 *cl*

Displays the number of active server connections of the *qmgr*(s).

3.2 *kill*

Preemptively terminates all processes of ***one*** queue manager (**kill –9**). The *qmgr* name ***must*** be explicitly and exactly supplied with kill (case sensitive). The shared memory is cleaned up as with action stop (see section 3.8 below).

3.3 *log/log2/log3*

Opens in current default editor the corresponding circular error log of the *qmgr* or of the first queue manager identified from the pattern or, if omitted, of the default queue manager or, if no default queue manager is defined, of the first queue manager identified on the system.

3.4 *ping*

For Monitoring: Pings a queue manager by the mqsc command **ping qmgr** or **display qmgr** (on z/OS, **ping** is not known) and returns with an appropriate completion code.

3.5 ps

Displays WebSphere MQ processes, **egrep**'ed according to the *pattern*(s) (case sensitive).

3.6 pw

Same as ps but tidily wraps lines longer than 80 characters.

3.7 run

Runs WebSphere MQ commands for the *qmgr* (**runmqsc**) or, if omitted, for the default queue manager on the system, or, if no default queue manager is configured, for the first queue manager identified on the system (in /var/mqm/mqs.ini). If no local queue manager is found matching the *pattern* and a local default queue manager is defined, all available remote queue managers are compared to the *pattern*. Hence remote administration (by **runmqsc**) can be performed as if the remote queue managers were local. As a prerequisite for this, the transmission queues of all channels must have standard names, i.e. they must be named like the corresponding remote queue manager.

3.8 start/stop

If *qmgr* is omitted, **mq** starts or stops all queue managers on the system. By default (unless you changed it) starts and stops with each queue manager

- a channel initiator (standard, if configured in /var/mqm/qmgrs/*qmgr*/qm.ini),
- a command server (**strmqcsv**),
- a trigger monitor (standard, **runmqtrm**),
- a listener (**runmqlsr**) with a port associated according to the section 6.7 *Startup Automation* (if the port is not configured for the inet daemon).

The trigger monitors and listeners write their logging and error messages to the error directory of their queue manager (/var/mqm/qmgrs/*qmgr*/errors) to a file named like the corresponding process (*.log appended, e.g. runmqtrm.log).

Upon a stop request, the semaphores and the inter process communication memory of the queue manager(s) are cleaned up. If, for WebSphere MQ Version 5.2 or earlier, the memory of each queue manger shall be cleaned independent of other queue managers running on the system, the undocumented support pack **amqiclen** from IBM must be copied to the WebSphere MQ binaries directory (/opt/mqm/bin or /usr/mqm/bin). However, a guarantee that all memory is cleaned up can only be given if all queue managers and WebSphere MQ processes on a system have been stopped.

4 WebSphere MQ Commands

When called with an *mqsc* command, **mq** affects all WebSphere MQ *objects* of type *type* matching the *pattern* in the *object* specification. The where clause available for WebSphere MQ since its Version 7 is not supported by **mq**. If (*pattern*) is omitted completely, all *objects* of type *type* are affected. With no attributes as well, the quotes may be omitted. For manipulative or destructive commands (e.g. *mqsc* = alter, clear, delete), the user is prompted for confirmation before the commands are executed. For remote administration, see section 3.7 *run*.

For efficiency and ease of operation, the new "MQSC object" qhandle is introduced, which is expanded to 'qstatus type(handle)' with the default attributes 'appltag input output inquire'.

In order to address a subset of the objects matching the name pattern, a similar pattern may be specified with any attribute of the object. All objects matching both the name pattern (if specified) and the attribute pattern will be selected (cp. 7.9 *Example 9*).

The attribute name, *attribute*, does not have to be a complete MQSC attribute name. Incompletely specified names are extended to a full attribute name by **mq**, e.g. "cu" to "CURDEPTH" or "iev" to "QSVCIEV". If several attributes match the incomplete name, **mq** chooses the first one it finds (alphabetically), i.e., "max", e.g., is expanded to "MAXDEPTH", not to "MAXMSGL". Also see the examples in sections 7.4 and 7.9.

5 Command Line Options

If none of the flags –I, –L, –M, –S, and –T is specified, all the corresponding services are started according to the defaults configured (see 6.6 *Default Definitions* below).

–a –
For 'mq start': Activate backup queue manager.
For actions of type *mqsc*: Display all MQ objects, including system objects suppressed by the configuration (cp. section 6.5 Hidden System Objects).

–c –
Redefine system objects, used with 'mq start'.

–d *information*
Amount of information displayed (*information* = all | minimal | none; cp. strmqm), used with 'mq start/stop'.

–e '*command*'
Shell command (in quotes) to be executed for logging fatal errors. The message string to be reported is identified by the variable $msg (cp. section 7 *Examples* below). If several logging commands are needed, separate them by semicolons. Defaults:

- Interactive shell: standard error ('echo $msg >&2')
- Batch shell: syslog ('logger –p user.err $msg')
  (Note: The syslog daemon incl. /var/adm/user.log must be configured)

–f –
Perform recover actions, done automatically for strmqm of versions <7, used with 'mq start'.

–n –
Do not start services (equivalent to flag –ns of strmqm).

–o '*command*'
Shell command (in quotes) to be executed for the logging of informational messages. The message string to be reported is identified by the variable $msg (cp. section 7 *Examples* below). If several logging commands are needed, separate them by semicolons. Defaults:

- Interactive shell: standard output ('echo $msg')
- Batch shell: syslog ('logger –p user.info $msg')
  (Note: The syslog daemon incl. /var/adm/user.log must be configured)

–r –
Update the backup queue manager / try to reconnect reconnectalbe clients.

–s –
Switch over to a standby queue manager.

–v –
verify the commands sent to **runmqsc,** but don’t execute them (same as **runmqsc –v**). Valid only for action run.

–w *timeout*
run: Timeout in seconds for remote administration. Default: 10 seconds.

kill: Timeout in seconds at which the command **amqiclen** to clean up shared memory is terminated. Defaults as for stop (see below).

stop: Timeout in seconds at which the WebSphere MQ command **endmqm –i** is cancelled. If after *timeout* seconds in **endmqm –i** queue manager processes are still running, **mq** terminates abnormally (cp. example 7.8). Defaults:

- Interactive shell: 0 seconds = infinity
- Batch shell: 60 seconds – for **mq** to return, consider one *timeout* each for **endmqm**, **amqiclen** (if present), and the listener (if present), plus about 5–10 seconds, i.e. 3 ∗ *timeout* + 10 seconds max.

–x –
Start/stop an instance of a multi-instance queue manager.

–z –
Suppress error messages.

–C –
start or stop all configured channels of the *qmgr*(s) (except system defaults). The *qmgr*(s) are started in advance, if they are not running already. Optionally a name pattern may be added like –C:*pattern* in order to start or stop the subset of all channels matching the *pattern*.

–I –
start or stop the standard channel initiator of the *qmgr*(s).

–L –
start or stop the listener (**runmqlsr**) configured for the *qmgr*(s) (cp. section 6.1 below). The output of the listener is written to the file runmqlsr.log in the errors directory of each *qmgr* (/var/mqm/qmgrs/*qmgr*/errors).

–M –
: start the *qmgr*(s), each with its standard channel initiator (unless CHINIT=NO is specified in the QueueManagerStartup section of the file qm.ini), but without other WebSphere MQ services if not explicitly flagged by –I, –L, –S, and/or –T. With stop the *qmgr* takes down all its services except its listener.

–P –
: Enable and disable the port of the queue manager(s) in the configuration of the inet daemon at start or stop respectively. You must be the super user in order to use the option –P. The changes made to /etc/inetd.conf are double checked before they are definitely applied, and for additional safety, a backup copy /etc/inetd.bak is made in advance.

–S –
: start or stop the command server of the *qmgr*(s). With start the queue manager is also started unless already running.

–T –
: start or stop the standard trigger monitor of the *qmgr*(s) (**runmqtrm**). With start the queue manager is also started unless already running. The output of the trigger monitor is written to runmqtrm.log in the errors directory of the *qmgr* (/var/mqm/qmgrs/qmgr/errors).

–X –
: Use the *qmgr* name exactly as passed from the command line (case sensitive).

6 Configuration

The configuration section with enterprise wide defaults for the shell script **mq** is found at the top of the script, after the leading comments. System specific configurations may be supplied in an optional separate file mq.conf, which may be local to the script **mq** or at /var/mqm (cp. section 9). The local exceptions in mq.conf override the standard rules on the internal list.

6.1 TCP/IP Ports

In order to identify enterprise wide rules for assigning queue manager names to TCP/IP ports, a list of the form

```
Ports='pattern=port [ pattern=port ... ]'
```

(blank separated) can be specified, where *port* denotes a port number and *pattern* a complete queue manager name (as a regular expression, i.e. enclosed, for unequivocalness, by a leading '^' and a trailing '$') or a full regular expression according to the **egrep(1)** and **regexp(5)** man pages.

If a queue manager name can be associated to several list entries, the first occurrence is used. The same port number can be associated to several queue manager names or name patterns, if they are not running on the same system.

Exceptions to these rules may be given in the 'Ports:' section of separate configuration file mq.conf (local or at /var/mqm):

```
    Ports:
    pattern=port
[   pattern=port
        ...          ]
```

where *pattern* again may be a complete queue manager name or a full regular expression according to the **egrep(1)** and **regexp(5)** man pages. Each pattern association occupies a separate line of text. The exceptions in mq.conf override the standard rules on the internal list.

If you are using NIS/NIS+ for Solaris as a centralized services database, don't worry. The present utility is prepared for it. For any other services database, you will have to adopt the configuration variable ListServices, which holds the command to list the services (cp. section 6.6 *Default Definitions*, paragraph d).

6.2 Checking Services Configuration of Ports

The script in the section "Miscellaneous" contains an entry named ListServices with the default setting:

```
ListServices='cat /etc/services 2>/dev/null || ypcat services.byname'
```

The **mq** script tries to find the port in /etc/services and if not found it checks the NIS services. Unfortunately many administrators keep all kinds of ports configured in /etc/services even though they are not used. If an unexpected port conflict is encountered by **mq**, the check can be disabled by the setting

```
ListServices='echo'
```

6.3 Remote Administration of z/OS Queue Managers

To enable the remote administration of z/OS queue managers in your environment, the latter must be specified to **mq.** A list of name patterns of z/OS queue managers is assigned in the configuration section of mq:

ZosQmgrs='*pattern* [*pattern* ...]'

where *pattern* is a full regular expression according to the **egrep(1)** and **regexp(5)** man pages (e.g. ^CSQ[12]$ to match CSQ1 and CSQ2). Exceptions to these rules may be given in the section 'ZosQmgrs:' of the separate configuration file mq.conf (local or at /var/mqm):

```
    ZosQmgrs:
    pattern
[   pattern
      ...       ]
```

where *pattern* denotes a complete queue manager name or a full regular expression according to the **egrep(1)** and **regexp(5)** man pages. Each pattern occupies a separate line of text.

6.4 Channels to Start or Stop

The user defined channels to start or stop with the option –c are identified by assignments

AllowChannels='*pattern*'
DenyChannels='*pattern*'

where *pattern* is a full regular expression according to the **egrep(1)** and **regexp(5)** man pages. The defaults are AllowChannels='.' and DenyChannels='^SYSTEM\.', i.e. all channels are started, except the SYSTEM… defaults.

Similar to the port definitions above, local exceptions of channels to start or stop by the option –c may be identified in the sections 'AllowChannels:' and 'DenyChannels:' of the file mq.conf (local or at /var/mqm):

```
    AllowChannels:
    pattern
[   pattern
      ...     ]
    DenyChannels:
    pattern
[   pattern
      ...     ]
```

where *pattern* denotes a complete queue manager name or a full regular expression according to the **egrep(1)** and **regexp(5)** man pages. Each pattern occupies a separate line of text.

6.5 Hidden System Objects

Very similar to channels, other hidden (system) objects, accessed by command line option –a only, are identified in the top of **mq** by

```
    AllowObjects='pattern'
    DenyObjects='pattern'
```

with the defaults AllowObjects='.' and DenyObjects='^SYSTEM\.', i.e. all objects are accessed, except the SYSTEM… objects. In the file mq.conf (local or at /var/mqm) they may be configured by

```
    AllowObjects:
    pattern
[   pattern
      ...     ]
    DenyObjects:
    pattern
[   pattern
      ...     ]
```

6.6 Default Definitions

The default values documented in section 4 above are defined in the configuration at the beginning of the script **mq**, just after the introductory comments. They may be altered by the user in order to set enterprise wide defaults that differ from the original values. These global defaults may be changed for the local system by adding to the file mq.conf (local or at /var/mqm) any of the associations *parameter=value* discussed in the current section (add separate lines *parameter=value* following the header "Defaults:").

a) Service Options

Default settings of options when start or stop is activated without services options:

• Let **strmqm** start the standard channel initiator(s):	StartChiDefault=0
• start/stop a listener with each *qmgr* unless the port is configured for the inet daemon:	StartLsrDefault=1
• start a standard trigger monitor with each *qmgr*:	StartTrmDefault=1
• start a command server with each *qmgr*:	StartCsvDefault=1

All services except the listener(s) are always stopped with the queue manager.

b) *Shell Commands for Logging*

Default shell commands for the reporting of informational messages and of fatal errors. Any shell logging command may be used. The message string to be output is identified by the variable $msg:

- Interactive shell:
  - errors: TtyErr='echo $msg >&2'
  - information: TtyInfo='echo $msg'
- Batch shell:
  - errors: BatchErr='logger –p user.err $msg'
  - information: BatchInfo='logger –p user.info $msg'

c) *Timeouts*

Default timeout in seconds for stop and kill in an interactive or batch shell without specifying the option '–w *timeout*' is not specified:

• Interactive shell (0 = infinity):	TtyTimeout=0
• Batch shell:	BatchTimeout=60
• Remote administration:	AdminTimeout=10

d) *Listing the Services*

The shell command to get the list the port entries in the services configuration is given by the global constant

```
ListServices='cat /etc/services 2>/dev/null || ypcat services.byname'
```

i.e. silently look for the file and if not found, try the NIS services (Solaris). If you are using a different network services configuration tool, change this command accordingly.

e) *Maximum Log Length*

The maximum number of lines kept in the log of each service (e.g. runmqlsr, runmqtrm) is controlled by the global constant

```
MaxLogLines=1000
```

If you prefer a different length of the log histories, change this constant accordingly.

f) *Using Perl for Regular Expressions*

The global setting

UsePerlRegEx=*n*

where *n*=0 or 1 defines the enterprise wide default setting for the usage of Perl for parsing by regular expressions. If UsePerlRegEx is set to 0, shell regular expressions (**egrep**) are used, if set to 1, pattern matching is carried out using Perl.

The original default distributed is UsePerlRegEx=1, since Perl regular expressions are much more powerful, supporting, e.g., negative look ahead or negative look behind (e.g., gimme all "FOO"s not followed or not preceded by "BAR", respectively). In addition, when using Perl regular expressions, the pattern matching of object names and attributes on the command line is extended by allowing for a *leading* exclamation mark ("!") in a pattern. The latter is used for negative matching in the sense of "show me all except those matching the pattern".

The default defined within the **mq** script may be overridden by exporting a shell variable in the user's environment:

```
export  mq_UsePerlRegEx=n
```

g) System Paths

If on your system the WebSphere MQ binaries path, the path to the inet daemon configuration file, or the user binaries path that contains any binaries like, e.g., **ps, grep,** etc. should differ from the following settings, alter it accordingly to make sure that the required system files are found by **mq** in any case:

• MQ binaries Path for AIX:	AixBin='/usr/mqm/bin'
• MQ binaries Path for HP-UX:	HpUxBin='/opt/mqm/bin'
• MQ binaries Path for Linux:	LinuxBin='/opt/mqm/bin'
• MQ binaries Path for Solaris:	SunOsBin='/opt/mqm/bin'
• User binaries path:	UsrBin='/usr/bin'
• Inet daemon configuration file:	InetConf='/etc/inetd.conf'

6.7 Startup Automation

In order to have all queue managers regularly started and stopped at system startup and shutdown, hard link or copy the script **mq** to the directory *etc*/init.d and create the hard links *etc*/rc2.d/S*nn*mq and *etc*/rc0.d/K*nn*mq where *etc* denotes /etc/rc.d for Linux and /etc for all other UNIX flavors. The ownership root:mqm and the file mode u:rwx g:rwx o:––– are recommended. S*nn*mq and K*nn*mq are called with a parameter start and stop respectively, starting or stopping with standard options all queue managers on the system. Ordinals *nn* > 80 are recommended. Alternatively, you may add an entry to /etc/inittab, e.g.:

```
mqs:2:once:/usr/local/bin/mq start >/dev/console 2>&1
```

(Note: The script **mq** switches to the user mqm, 'su – mqm' not required.)

7 Examples

7.1 Example 1

Start all queue managers on a system with their standard services only (if not already running) and start all their user defined channels:

```
mq start –CM
```

7.2 Example 2

Clean up the memory after all queue managers have stopped:

```
mq stop
```

All WebSphere MQ services already stopped are ignored.

7.3 Example 3

Display the trigger setting and the current depth of all local queues beginning with 'COMPLAINT' and ending on 'IN', hosted by the remote queue manager OSIRIS:

```
mq dis 'ql(^complaint.*in$) trigger curdepth' osi
```

Note the incomplete queue manager name (requires that no other local or remote queue manager name matches 'OSI').

7.4 Example 4

Set the default persistence of all queues containing 'order' anywhere in their name, hosted by the default queue manager:

```
mq alt 'q(order) ps(yes)'
```

Note the incomplete attribute name 'ps', which is expanded to 'defpsist'. You will be prompted to confirm the whole set of commands generated and, unless 'all' is entered, for the confirmation of each individual command as well.

7.5 Example 5

Delete all channels containing either ISIS or OSIRIS anywhere in their name, hosted by the queue manager BROKER1:

```
mq delete 'chl(isis|osiris)' broker1
```

As in Example 4 above, you will be prompted to confirm the execution.

7.6 Example 6

Display the current status of all channels of the local queue manager PUBLISHER:

```
mq dis chs pub
```

Note the incomplete queue manager name (requires that no other local queue manager name matches 'PUB' and appears prior to PUBLISHER in the WebSphere MQ configuration).

7.7 Example 7

Start all queue mangers on a system, each with a rigger monitor only, sending any errors to Tivoli:

```
mq  start  –e  'postemsg  –m $msg  –r  CRITICAL  mq_error  MQM'  –MT
```

7.8 Example 8

Stop the queue manager PUBLISHER (exact name, with no other leading or trailing characters) by a Veritas Cluster with a timeout of 90 seconds , disabling the inet daemon (so no receiver channel processes are started anymore), and logging regular output as 'normal' information (tag E) and fatal errors as a 'critical error' (tag B):

```
mq  stop  –e 'halog –add B $msg'   –o 'halog –add E $msg'   –w90   –PMX   PUBLISHER
```

7.9 Example 9

Display all remote queue definitions on the local default queue manager that contain PUBLISHER in their remote queue manager attribute:

```
mq  dis  'qr  rqm(publisher)'
```

Note the incomplete attribute name ‘rqm’, which is expanded to rqmname.

8 Exit Status

In what follows, the expressions '$1' and '$2' denote variable output related to the current call, e.g. the current queue manager name ('$1'), an illegal command line option ('$1'), or the name (and path) of the error log file of the current queue manager ('$2'). The return codes of **mq** are chosen in order to prevent collisions with return codes from the IBM utilities used. Any other return code is the one from the WebSphere MQ sub-process called by **mq.**

0	No error.
201	Operating system '$1' not supported.
202	Permission denied.
203	Unexpected error in amqiclen: $1.
204	Timeout: Unable to stop queue manager '$1'.
205	Timeout: Unable to stop command server of queue manager '$1'.
206	Unable to stop the trigger monitor of queue manager '$1'.
207	Queue manager '$1' not found; exact name required with >kill.
208	No queue manager found matching '$1'
209	Command not processed: User interrupt.
210	Cannot show log in batch mode or pipe.
211	Error log '$2' of queue manager '$1' not found.
212	Illegal timeout value: '–w $1' must be numeric.
213	Unknown command line option: $1.
214	Error in mqsc command supplied: Attribute not found.
215	Option –v (verify) supported for action >run only.

9 Files and Installation Instructions

mq	The shell script for the administration of WebSphere MQ services. Copy **mq**, e.g., to /usr/local/bin with the recommended owner root:mqm and a file mode of u:rwx g:rwx o:––– (octal 770, cp. section 6.7 *Startup Automation* above).
mq.conf	Optional configuration file for **mq** (see section 6 *Configuration* above), located next to the script, on the local path, or at /var/mqm, with the recommended owner mqm:mqm and a file mode of u:rw– g:rw– o:r–– (octal 664).
mq.1	Manual entry (man page), to be copied, e.g., to the directory /usr/local/man/man1.
mq.pdf	The documentation in hand.
mqTrc	Shell script for tracing and support: May be executed on request of the author in order to generate trace output of **mq** which shall be mailed to the author for efficient support.
gpl-v2.pdf	Gnu Public License, Version 2
readme.txt	Introduction, quick installation guide, and history of changes.

10 Reporting Bugs

Post cards, suggestions, or congratulations for successfully debugging over 1500 lines of shell script code are welcome. You are invited to inform the author if you are using **mq,** and please report bugs and any comments to johannes.boehm@ti8m.ch or to j.boehm@gmx.ch.

11 Copyright