MO03: WebSphere MQ Queue Load / Unload Utility
User Guide
Version 1.6

12th July 2008

Paul Clarke
WebSphere MQ Development
MP211,
IBM UK Laboratories Ltd.
Hursley
Winchester
Hants, SO21 2JN
United Kingdom

paulg_clarke@uk.ibm.com

Take Note!

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

Seventh Edition, July 2008

This edition applies to Version 1.6 of *WebSphere MQ Queue Load / Unload Utility* 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.

INTERNATIONAL BUSINESS MACHINES CORPORATION 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM 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 IBM 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:
WebSphere 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

Contents

Tables

Preface

Ever since I released my MA01 (Q Utility) SupportPac I have had periodic requests to explain how it can be used to unload, and subsequently reload, messages from a queue. The answer has always been that this is not what MA01 is for and that surely there must be a utility available. Well, after sufficient numbers of these requests I looked for a utility myself and didn't really find anything which fitted the bill. What was needed was a very simple, some would say unsophisticated, program which unloaded a queue into a text file. The notion of a text file was important because a number of users wanted the ability to change the file once it had been created. I also find that text based files are more portable and so this seemed useful if we want to unload a queue, say on Windows, and then load the messages again on a Solaris machine. The disadvantage of this approach is that the file is larger than it would be in binary mode. Storing data using the hex representation of the character rather than the character itself essentially uses twice as much space. However, in general I do not envisage people using this program to unload vast amounts of message data but a few test messages or a few rogue messages on the dead letter queue which are then changed and reloaded elsewhere.

I would like to thank Morag Hughson for writing this user guide, for testing the described functionality and for porting it to z/OS. Writing the code is the fun bit and without her offering to do the leg work I doubt whether this SupportPac would have ever seen the light of day.

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

Chapter 1. WebSphere MQ Queue Load / Unload Utility

This document describes the functions available in the SupportPac.

Overview

The *WebSphere MQ Queue Load / Unload Utility* allows the user to copy or move the contents of a queue, its messages, to a file. This file can be saved away as required and used at some later point to reload the messages back onto the queue. Messages can be copied or removed from a queue using a selection criteria based on position in the queue, a search string and/or the age of the message.

The unload file has a specific format understood by the utility, but is human-readable, so that it can be updated in an editor before being reloaded. Care should be taken not to change the format when editing fields within it. The utility will only reload a file with the correct format.

Installation

The following platforms are supplied in the zip file MO03.zip:

- Windows
- AIX
- Linux Intel
- Linux Intel 64
- Linux Power
- HP
- HP Itanium
- Solaris
- Solaris x86
- zLinux
- z/OS

Download this file (in BINARY) to a temporary directory and unzip with the option to ensure the subdirectories stored in the ZIP file are re-created). This will create subdirectories containing the items shown below. Once unzipped, transfer the appropriate executable, again in binary, to the destination system (see below for detailed z/OS installation instructions).

Windows DIRECTORY

- qload.exe – this program will run as a client or a Queue Manager program (see -l parameter).

zOS DIRECTORY

- QLOAD.SEQ – sequential file containing Queue Manager program
- QLOAD.JCL – example JCL for running in batch

Other platform DIRECTORY

- qload - this program will run as a client or a Queue Manager program (see -l parameter).

z/OS Installation Instructions

Transfer the QLOAD.SEQ file to a z/OS system using the following commands

```
ftp> binary
ftp> quote site fixrecfm 80
ftp> put zOS\QLOAD.SEQ QLOAD.SEQ
```

In tso use the following command

```
receive inds(QLOAD.SEQ)
```

when prompted for a filename, reply

```
DSN(QLOAD.LOAD)
```

QLOAD can be run on z/OS in BATCH - an example piece of JCL is provided in the zOS directory. QLOAD can also be run interactively, e.g. from the TSO/E READY prompt or the ISPF Command Shell (=6).

Chapter 2. Getting Started

The *WebSphere MQ Queue Load / Unload Utility* can be useful for a number of tasks. These are detailed below as simple examples. All of these examples can be modified with the use of a number of other parameters which are documented in “Chapter 3. Parameters” on page 5.

Example 1. Unload a Queue to a File

To save the messages that are on a queue, to a file, perhaps for archiving purposes and the possibility of later reload back onto a queue; use the following options on the command line.

```
qload –m QM1 –i Q1 –f c:\myfile
```

This takes a copy of the messages from the queue and saves them in the file specified. The format of this file is described in “Chapter 4. File Format” on page 17.

Example 2. Unload a Queue to a series of files

You can unload a queue to series of files by using an ‘insert’ character in the file name. In this mode each message is written to a new file.

```
qload –m QM1 –i Q1 –f c:\myfile%n
```

This command will unload the queue to files, myfile1, myfile2, myfile3 etc. For a complete list of insert characters please refer to File Insert Characters on page 8.

Example 3. Load a Queue from a File

To reload a queue with the messages you saved in “Example 1. Unload a Queue to a File”, use the following options on the command line. Note that the file passed to *WebSphere MQ Queue Load / Unload Utility* must be a recognised format. The recognised formats are listed in “Recognised file formats” on page 18.

```
qload –m QM1 –o Q1 –f c:\myfile
```

Example 4. Load a Queue from a series of files

You can load a queue from a series of files by using an ‘insert’ character in the file name.

```
qload –m QM1 –o Q1 –f c:\myfile%n
```

This command will load the queue files files, myfile1, myfile2, myfile3 etc. For a complete list of insert characters please refer to File Insert Characters on page 8.

Example 5. Copy the messages from one Queue to another Queue

The file parameter in “Example 1. Unload a Queue to a File” could be replaced with another queue name instead, allowing the messages from one queue to be copied to another queue, using the following options on the command line.

```
qload –m QM1 –i Q1 –o Q2
```

Example 6. Copy the first 100 messages from one Queue to another Queue

The file parameter in “Example 1. Unload a Queue to a File” could be replaced with another queue name instead, allowing the messages from one queue to be copied to another queue, using the following options on the command line.

```
qload –m QM1 –i Q1 –o Q2 –r#10
```

Example 7. Move the messages from one Queue to another Queue

A variation on “Example 3. Load a Queue from a File” would be to move the messages instead of copying them. This illustrates the distinction between using `–i` (lower case) which only browses a queue, and `–I` (upper case) which destructively gets from a queue. Use the following options on the command line.

```
qload –m QM1 –I Q1 –o Q2
```

Example 8. Move messages older than one day from one Queue to another Queue

This example shows the use of age selection. Messages can be selected which are older than, younger than or within a range of ages. Use the following options on the command line.

```
qload –m QM1 –I Q1 –o Q2 –T1440
```

Example 9. Work with the file of messages

Having unloaded the message from your queue, as in “Example 1. Unload a Queue to a File”, you may want to edit the file. You may also want to change the format of the file to use one of the display options that you did not specify at the time you unloaded the queue. You can use the *WebSphere MQ Queue Load / Unload Utility* to reprocess the file into the desired format even after the unload of the queue has taken place. Use the following options on the command line.

```
qload –f c:\oldfile –f c:\newfile –dA
```

Example 10. Display the ages of messages currently on a Queue

Use the following options on the command line.

```
qload –m QM1 –i Q1 –f stdout -dT
```

Chapter 3. Parameters

There are a number of switch parameters that can be passed to *WebSphere MQ Queue Load / Unload Utility* to control the behaviour you need. These are detailed in this chapter.

Parameters Flags

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

<table>
<tr><td>-a</td><td>Controls whether the file is opened in binary or append mode
-aa Append
-ab Binary</td></tr>
<tr><td>-c</td><td>Controls whether messages taken from a queue are converted
-c < CCSID> [: X ‘Encoding’]
For example –c850:111</td></tr>
<tr><td>-C</td><td>Context options
-CA Set all context (Default)
-CI Set identity context
-Ca Pass all context
-Ci Pass identity context
-Cd Default context
-Cn No context
Use of the ‘pass’ context options is restricted to cases where the source messages are consumed from a queue. For a fuller description of context options please see ‘Context Options’ on page 11.</td></tr>
<tr><td>-d</td><td>Display options
-da Add ASCII columns to the hex output in the file to aid readability
-dA Write ASCII lines of data wherever possible
-di Include the message index in the output
-dT Display the time the message has been on the queue
-dw<Length> Set the data width for the output</td></tr>
<tr><td>-D</td><td>Add delay, expressed in milliseconds, before writing message to output destination.
-D <+ve value> Add a fixed delay before putting message
-D500 would put each message half a second apart
-D <-ve value> Add a random delay up to the specified value before putting message
-D-10000 adds a random delay of up to 10 seconds before putting message
-D r <value> Replays the messages at a percentage of their original put speed
<table>
<tr><td>-Dr</td><td>Replays messages at original speed</td></tr>
<tr><td>-Dr50</td><td>Replays messages at twice original speed</td></tr>
<tr><td>-Dr200</td><td>Replays messages at half original speed</td></tr>
</table></td></tr>
<tr><td>-f</td><td>Specifies either the source or target file name.
For a complete description of the filename format please see ‘File Use’ on page 7.</td></tr>
<tr><td>-F</td><td>Specifies either the source or target file name. For a target file it forces output to file if it already exists. The program will not ask whether the file should be overwritten.
For a complete description of the filename format please see ‘File Use’ on page 7.</td></tr>
<tr><td>-i</td><td>Specifies an input queue to browse
For example -iQ1</td></tr>
<tr><td>-I</td><td>Specified an input queue to consume messages from
For example –IQ1</td></tr>
</table>

-l	Specifies the name of the WebSphere MQ library to run against. This parameter controls whether the program runs as a local application or as a client. -lmqm Local Application -lmqic32 Client Application
-m	Specifies the name of the Queue Manager to connect to For example –m QM1
-o	Output Queue Name For example –o Q2
-p	If set this option will cause the source queue to be purged of messages as they are copied to the target destination.
-q	Sets quiet mode. If this flag is set the program will not output it's usual summary of activity.
-r	Sets the applicable message range -r x Just the 'x'th message For example –r10 -r x..y From message 'x' to message 'y' For example –r 10..20 -r x#y Output 'y' messages starting at message 'x' -r 100#10 -r#x The first 'x' messages For example –r#100
-t	Sets the transaction message limit. For a description of the use of transactions please see 'Transactions' on page 11
-T	This parameter allows message selection based on message age. For a complete description of the options available please see 'By Message Age' on page 15.
-s	This parameter allows message selection based on the content of the message. For a complete description of message selection please see 'By Search String' on page 14.
-S	This parameter allows message selection based on the content of the message. For a complete description of message selection please see 'By Search String' on page 14.
-e	This parameter allows message selection based on the content of the message. For a complete description of message selection please see 'By Search String' on page 14.
-E	This parameter allows message selection based on the content of the message. For a complete description of message selection please see 'By Search String' on page 14.
-w	Wait Interval, in milliseconds, for consuming messages. If specified the program will wait for messages to arrive, for the specified period, before ending .
-x	This parameter allows message selection based on the content of the message. For a complete description of message selection please see 'By Search String' on page 14.
-X	This parameter allows message selection based on the content of the message. For a complete description of message selection please see 'By Search String' on page 14.

Connection Methods

You have two connection methods available to you with the *WebSphere MQ Queue Load / Unload Utility*. You can either connect directly to a local queue manager, or connect using a client connection. The default behaviour is to connect directly to the queue manager. If you have a default queue manager you can omit the `-m` switch which provides the queue manager name.

Connecting as a client

To connect to the Queue Manager via a client the normal MQ client configuration rules apply. The program can use either the MQSERVER environment variable or the Client Channel Definition Table (CCDT). The same program can connect both as a local application and as a client. The 'l' parameter controls which WebSphere MQ library is loaded at runtime, the default being to connect locally.

To run as a local application connecting directly to a local Queue Manager you would use a command such as:-

```
qload –m QM1 –iQ1
```

To run as a client you would use a command such as:-

```
qload –m QM1 –iQ1 –l mqic32
```

File Use

As already seen in "Chapter 2. Getting Started" on page 3, the `-f` flag is used to indicate the file name. This file may already exist in which case the program will ask whether you wish to overwrite it or not. If you select not to overwrite the file, no messages will be unloaded and the utility will end. You can select to overwrite the file when asked this question, or you can specify that the file to be used should be overwritten if it exists by using the `-F` (upper case) option as shown, on the command line.

```
qload –m QM1 –i Q1 –F c:\myfile
```

If you wish to combine the messages from two queues into one file, you can use your operating system services to concatenate the two files together and the format will still be acceptable to *WebSphere MQ Queue Load / Unload Utility*.

You can also use the `-f` flag to specify **`stdout`** as the output source instead of a file name. This may be useful if you simply wish to display the messages on the queue to the screen, or pipe the output another program.

z/OS File name format

Specifying the file name on z/OS can be done in a number of different ways. Examples of the most common are shown here.

To provide the name of a dataset, whether sequential file or partitioned dataset, directly, use the following file name format (note that this can be fully qualified as in the first example, or will have your TSO user ID pre-pended to the name in the second example, depending on whether you use the single quotes or not):

```
qload –m ZQM1 –i Q1 -f"//'USERID.MY.FILE'"
qload –m ZQM1 –i Q1 -f"//MY.FILE"
```

You can provide the name of a member of a partitioned dataset, using the following file name format (bearing in mind the use if single quotes mentioned above):

```
qload –m XQM1 –i Q1 -f"//'USERID.MY.PDS(MEMBER)'"
```

however, this does not work from within a parm string in your JCL because of the single quotes, so when using this from JCL, you must use the following:

```
EXEC PGM=QLOAD,PARM=('–m XQM1 –i Q1 -f"//''USERID.MY.PDS(MEMBER)''"')
```

You can also use DD cards to identify the data set you wish to use. For example, if you have a DD card thus:

```
//INFILE   DD DSN=USERID.MY.FILE,DISP=SHR
```

you can then use the following:

```
qload –m ZQM1 –i Q1 -f"DD:INFILE"
```

For more details see "Chapter 11. Performing OS I/O Operations" in *z/OS C/C++ Programming Guide* (SC09-4765-03).

File Insert Characters

The program is capable of writing and reading to multiple files in a single command. This can be useful if you wish to unload a queue where each message goes to a separate file or if you want to load multiple files to a queue. To facilitate this special insert characters can be placed in the file name which will be replaced with a value when the program runs.

Inserts are specified by preceding the character with a '%' character in the file name. For example, using the command **q –iQ1 –fQ1_Unload_%c** will actually write to the file 'Q1_Unload_061014'. In other words it will append the date to the file name. The full list of inserts is as follows:-

Index Inserts	
i	Input message number
I	Input message number padded to 5 characters eg. 00001
o	Output message number
O	Output message number padded to 5 characters eg.00001
n	Current File number
N	Current File number padded to 5 characters eg.000001
Date Inserts	
c	Current date in YYMMDD format This format is used to enable the files to be sorted alphabetically. Alternatively a different date format can be constructed using the inserts below.
C	Current date in YYYYMMDD format
d	Two digit day of month

dd	Day of month including suffix eg.1st, 2nd, 3rd …
j	Julian day of year
m	Three character month name eg.Jan,Feb,Mar….
mm	Two digit month
mmm	Full month name eg. January, February,March…
y	Four digit year
yy	Two digit year
D	Three character day of week eg.Mon,Tue,Wed…
DD	Full character day of week eg. Monday, Tuesday, Wednesday…
Time Inserts	
t	Simple time format eg. 181403
H	Two digit hour (24 hour clock)
HH	Hour (24 hour clock)
h	Two digit hour (12 hour clock)
hh	Hour (12 hour clock)
M	Two digit minutes
S	Two digit seconds
P	AM/PM
p	am/pm
Other Inserts	
%	The '%' character

Table 1: File insert characters

File Insert Examples

In the following examples assume that we have a queue called Q1 which contains 10 messages where each message contains the string which is their message number. Eg. "One", "Two","Three" etc etc

- **Adding the current date and time to an output file**
  `qload –m QM1 –iQ1 –f c:\myfile_%c_%t`

- **Writing each message on a queue to a separate file**
  `qload –m QM1 –iQ1 –f c:\myfile%n`
  Will write each message in Q1 to a separate file myfile1, myfile2, myfile3….

- **Loading a series of files onto a queue**
  `qload –m QM1 –oQ1 –f c:\myfile%n`
  Will load the files myfile1, myfile2, myfile3….
  The loading will stop as soon as a file does not exist.

- **Selectively reading messages from a queue**
  `qload –m QM1 –iQ1 –se –f c:\myfile%o`
  The –se parameter says that only messages containing the character ‘e’ should be written.
  Will write each message in Q1 to a separate file myfile1, myfile2, myfile3….

- **Selectively reading messages from a queue**
  `qload –m QM1 –iQ1 –se –f c:\myfile%i`
  The –se parameter says that only messages containing the character ‘e’ should be written.
  Will write each message in Q1 to a separate file myfile1, myfile3, myfile5,myfile7….
  Note that the %i character is the index of the **input** message. Consequently the file names correspond to the message index which caused the file to be written. In this example, therefore you get the file indexes of the numbers which contain the letter ‘e’.

- **Reformatting a selection of files**
  `qload –m QM1 –fInFile%n -da –f –fOutFile%n`
  This command will read the files, InFile1, InFile2 etc and write them reformatted to files OutFile1,OutFile2 etc. The program will stop as soon as an input filename is reached which does not exist. The number of output files will match the number of input files.

- **Reformatting a selection of files containing multiple messages**
  `qload –m QM1 –fInFile%n -da –f –fOutFile%o`
  This command will read the files, InFile1, InFile2 etc and write them reformatted to files OutFile1,OutFile2 etc. The program will stop as soon as an input filename is reached which does not exist. The number of output files will match the number of input messages – that is some of the input files may have had more than one message contained within them.

Queue access options

As already seen in “Chapter 2. Getting Started” on page 3, the `–o` flag is used to indicate the output queue, that is the queue to which messages are put; the `–i` and `–I` flags are used to indicate the input queue, that is the queue from which messages are browsed or destructively got.

If the messages on the queue being unloaded need to be converted, the `–c` flag should be used to cause the MQGET call from the input queue to specify GMO_CONVERT with the CCSID and Encoding values specified. The encoding value should be specified in hex. Use the following options on the command line.

```
qload –m QM1 –i Q1 –f c:\myfile –c 850:X’222’
```

If all that is required is to use the local code page and native encoding then simply use the following options on the command line.

```
qload –m QM1 –i Q1 –f c:\myfile –c 0
```

Transactions

By default the program will try to do a group of messages in a single transaction in order to improve performance. However, there is no guarantee that the WHOLE operation will be done in a single transaction. To do this you need to use the `-t` parameter to say how many messages can be done in a single transaction.

Option	Meaning
`-t 0`	Switch off transactions all together
`-t-1`	All messages will be done in a single transaction
`-t n`	The message operations will be split into groups of n messages, for example `–t1000` would deal with 1000 messages in a single transaction.

Table 2: Transaction flag values

Clearly you can not specify a value larger than the maximum uncommitted messages value for the Queue Manager and you need to ensure that the Queue Manager log is large enough for your transaction.

Context Options

There are two sets of context information in the Message Descriptor (MQMD), the identity context fields and the origin context fields. These are described in the *WebSphere MQ Application Programming Guide*. The default action of the *WebSphere MQ Queue Load / Unload Utility* is to set all the context information in the MQMD to that which was saved in the file being loaded. This requires the user ID under which the utility is running to have appropriate authority to set all the context fields.

The context fields can be manipulated in other ways using the various other options with the `–C` flag. “Table 3: Context options used by WebSphere MQ Queue Load / Unload Utility ” details the options that can be used. The main difference to note is that the first two options are applicable when loading from a file, and the second two options are applicable when loading one queue from another queue.

Option	Meaning
`A`	Set all the context fields in the MQMD to that which was saved in the file.
`I`	Set only the identity context fields in the MQMD to that which was saved in the file.
`a`	Pass all the context fields in the MQMD from the messages on the input queue to the messages on the output queue.
`i`	Pass only the identity context fields in the MQMD from the messages on the input queue to the messages on the output queue.
`d`	Context fields in the MQMD of messages on the output queue will represent the *WebSphere MQ Queue Load / Unload Utility*.
`n`	There will be no information in the Context fields in the MQMD of messages on the output queue.

Table 3: Context options used by WebSphere MQ Queue Load / Unload Utility

Display Options

The default way that your message data is represented in the file is hex data, as follows:

```
X 000000AE0800000100000004000000440000000DF0700000000
X 0000300000004E54314D41482020202020202020202020202020
```

When unloading messages from a queue to a file, or when manipulating a file later, you can specify one of the display options for your hex and ASCII data. What follows are three example excerpts from the file produced when using each of these options. The full format of the file is discussed in “Chapter 4. File Format” on page 17. The examples are all shown as post-processing of the file, but the options can also be used when unloading a queue to a file.

Combination of Hex and ASCII data

Use the following options on the command line.

```
qload –f c:\oldfile –f c:\newfile –da
```

This will produce a file that shows the hex values of your message data on the left and any displayable characters on the right.

```
X 000000AE0800000100000004000000440000000DF0700000000  <.............D.........>
X 0000300000004E54314D41482020202020202020202020202020  <..0...NT1MAH              >
```

Interleaved Hex and ASCII data

Use the following options on the command line.

```
qload –f c:\oldfile –f c:\newfile –dA
```

This will produce a file that shows ASCII text whenever the data is displayable and hex values otherwise.

```
X 000000AE080000010000000400000 0

S "D"
X 000000DF07000000000000

S "0"
X 000000

S "NT1MAH                                                "
```

Message Index

Use the following options on the command line.

```
qload –m QM1 –i Q1 –f c:\myfile –di
```

This will add a comment line at the beginning of each message to show the index of the message.

```
* Index 1
A RPT 0
```

This is just a comment and is useful for determining which message you are looking at in the file. If you concatenate two files together, or edit the file to remove some of the messages, then the indexing will no longer be correct. Use the following options on the command line to re-index a file after it has been edited.

```
qload –f c:\oldfile –f c:\newfile –di
```

Message Age

Use the following options on the command line.

```
qload –m QM1 –i Q1 –f stdout –dT
```

This will display the ages of the messages on the queue.

```
1. MsgId:414D51204E5450474331202020202006A937452000A905
             6 hours 16 minutes 51 seconds
2. MsgId:414D51204E5450474331202020202006A937452000D204
             35 minutes 26 seconds
```

If required message selection can be performed based on the ages of the messages. Please see Message Seelction : *By Message Age* on *Page 15*.

Summary Data

Once *WebSphere MQ Queue Load / Unload Utility* has completed executing the command with the options you have specified, it will output a summary of what it did, for example:

```
Read    - Files    1  Messages:     1  Bytes:        15
Written - Files    0  Messages:     1  Bytes:        15
```

To suppress this summary, use the quiet option, `-q`, as follows:-

```
qload –m QM1 –o Q1 –f c:\myfile –q
```

Message Selection

When using the *WebSphere MQ Queue Load / Unload Utility* to move or copy messages, or even just to manipulate the file that was produced from by unloading a queue, it may be only necessary to work with some of the messages. This can be done by selecting specific messages to process in two different ways. You can either select messages by a numeric range, or you can select messages that contain a specific string.

By Message Range

Indexing the file as described in “Message Index” on page 12, may be useful when working with ranges. You can choose to process a certain range of messages from the input source (this may be a queue or a file) in a number of different ways, which are shown in “Table 4: Message range options used by WebSphere MQ Queue Load / Unload Utility ”.

Option	Meaning
x	Process message number x only.
x..y	Process message number x through y only
x#y	Process y messages starting from message x
#y	Process y messages starting from the beginning. **Note:** This is the same as 1#y

Table 4: Message range options used by WebSphere MQ Queue Load / Unload Utility

Here are examples of some of the options. To illustrate the difference between `x..y` and `x#y`, we have a file with 10 messages in it. In order to load messages 2 through 7 inclusive onto our queue we could use either of the following options on the command line.

```
qload –m QM1 –o Q1 –f c:\myfile –r2..7
```

```
qload –m QM1 –o Q1 –f c:\myfile –r2#6
```

In both cases the *WebSphere MQ Queue Load / Unload Utility* will let you know which messages you are processing with output similar to the following.

```
QLOAD Program by Paul Clarke [ V1.0 Build:Mar 21 2005 ]
Processing message indexes 2 -> 7.
```

By Search String

You can choose to process only messages that contain a certain string. This string can be in ASCII, EBCDIC or hex.

Use the following options on the command line to only process those messages that contain the ASCII string 'SALES'.

```
qload –m QM1 –i Q1 –f c:\SalesFile –s SALES
```

You can also do the opposite search - that is for messages that do not contain a certain string. This can be done using the upper case version of the option, for example, to only process those messages that do not contain the EBCDIC string 'SALES'.

```
qload –m QM1 –i Q1 –f c:\SalesFile –E SALES
```

"Table 5: Search string options used by WebSphere MQ Queue Load / Unload Utility" lists all the search options that can be used with the *WebSphere MQ Queue Load / Unload Utility*.

	Search containing this string	Search not containing this string
ASCII Search Options	`s (lower case)`	`S (upper case)`
EBCDIC Search Options	`e (lower case)`	`E (upper case)`
Hex Search Options	`x (lower case)`	`X (upper case)`

Table 5: Search string options used by WebSphere MQ Queue Load / Unload Utility

You can use any combination of the search string options together. If more than one option is used, all search strings must match for the message to be processed. Use the following options on the command line to only process those messages that contain the ASCII string 'SALES' and do not contain the hex string 'F0F1F2'.

```
qload –m QM1 –i Q1 –f c:\SalesFile –s SALES –X F0F1F2
```

On ASCII platforms (Windows and Unix) use the `–s` option to search for a natively encoded string; on EBCDIC platforms (z/OS) use the `–e` option to search for a natively encoded string.

By Message Age

You can choose to process only messages older than a certain time interval using the **–T** flag. Time interval can be specified in Days, Hours and Minutes. The general format being **[days:]hours:]minutes**. The parameter can take one or two times, **-T [OlderThanTime][,YoungerThanTime]**

For example, here are a number of commands and there effect :-

- **display messages older than 5 minutes.**
  `qload –m QM1 –i Q1 –fstdout –T5`
- **display messages younger than 5 minutes.**
  `qload –m QM1 –i Q1 –fstdout –T,5`
- **display messages older than 1 day but younger than 2 days.**
  `qload –m QM1 –i Q1 –fstdout –T1440,2880`
- **The following command will copy messages older than 1 hour from Q1 to Q2.**
  `qload –m QM1 –i Q1 –o Q2 –T1:0`
- **The following command will move messages older than 1 week. from Q1 to Q2**
  `qload –m QM1 –I Q1 –o Q2 –T7:0:0`

The **–T** flag can be combined with the **–dT** flag to determine whether there are messages older than a certain elapsed time on the queue without actually printing the contents of the message.

Purge non-selected messages

In addition to using message selection, as described above, to control which message get copied or moved, you may also need to purge those messages that do not match the message selection criteria when moving several messages. This is best illustrated by an example.

Let's start with queue Q1 containing messages with the following message data, and Q2, which is empty.

```
one potato
two potato
three potato
four
five potato
six potato
seven potato
more
```

We use the following command:

```
qload –m QM1 –I Q1 –o Q2 –s potato
```

This would leave Q1 containing the following messages

```
four
more
```

and would move the following messages to Q2

```
one potato
two potato
three potato
five potato
six potato
seven potato
```

However, if we want to ensure that when moving those messages to Q2, any messages that do not match are also removed from the source queue, Q1, then we must use the following command:

```
qload –m QM1 –I Q1 –o Q2 –s potato -p
```

Chapter 4. File Format

The format of the file that the *WebSphere MQ Queue Load / Unload Utility* uses is deliberately human-readable to allow a user to update the file to easily make changes to the messages before loading them onto a queue, perhaps on a different system. After introducing the basic format of the file, this chapter gives an example to illustrate a possible field that you may wish to update, and then provides a reference to the full file format.

The file has a free format. Spaces and blank lines are ignored unless between quotes. The *WebSphere MQ Queue Load / Unload Utility* will add spaces to make the file more readable, but they do not affect the message format stored in that file.

The first column contains the key for each line. This can have a number of different values, some of which we have seen already. These are listed in “Table 6: Meaning of column one symbol in file format”.

Column 1 value	Meaning
S	The text shown on this line is ASCII string text
X	The text shown on this line is hex
A	The text shown on this line is an attributed in the Message Descriptor (MQMD)
*	The text on this line is a comment and will be ignored.

Table 6: Meaning of column one symbol in file format

The second column should contain a blank; this makes the file more readable.

If the first column contained the letter ‘A’, then the next three characters will represent the field that is being shown. For example, ‘FMT’ shows that this line displays the format of the message. The full set of these field labels is listed at the end of this chapter.

Example - Changing the user ID

The message descriptor (MQMD) contains a user ID which, depending on your security settings, may be used for access control when putting a message onto a queue. You may wish to change this user ID before reloading the messages onto a queue on a different system because the IDs on that system use a different naming convention. Before loading the queue from your file of messages, you would edit the file changing the field identified as USR to your desired user ID.

```
:
A RTM MQ24
A USR HUGHSON
A ACC 1A0FD4D8F2F4C3C8C9D5F1F9C6F7C1C3F3F00019F7AC30000000000000000000
:
```

Attribute Format Reference

The fields in the Message Descriptor (MQMD) are formatted in the file with a three character string representing the attribute name. “Table 7: Message descriptor attribute representations” lists the full set of these strings and which field they represent.

Message Descriptor attribute	File format representation
Report	RPT
MsgType	MST
Expiry	EXP
Feedback	FBD
Encoding	ENC
CodedCharSetId	CCS
Format	FMT
Priority	PRI
Persistence	PER
MsgId	MSI
CorrelId	COI
BackoutCount	BOC
ReplyToQ	RTQ
ReplyToQMgr	RTM
UserIdentifier	USR
AccountingToken	ACC
ApplIdentityData	AID
PutApplType	PAT
PutApplName	PAN
PutDate	PTD
PutTime	PTT
ApplOriginData	AOD

Table 7: Message descriptor attribute representations

Recognised file formats

The *WebSphere MQ Queue Load / Unload Utility* recognises the above described file format, but also recognises the file format used as output from the sample browse program AMQSBCG.