

RFHUTIL

**Display Test and Performance utilities for
IBM MQ and IBM Integration Bus**

Version 9.1

31 Dec 2018

Jim MacNair

Table of Contents

Table of Contents	ii
Notices	ix
Trademarks and service marks.....	ix
Acknowledgments.....	xi
Preface	xii
Chapter 1. Introduction	1
What the RFHUtil utility does	1
Header chaining.....	1
Chapter 2. Installation	2
Supported Environments	2
Display Requirements	2
Character set considerations.....	2
Limitations in the RFHUtil.exe utility.....	3
Installation	3
Security Considerations when using RFHUtil and RFHUtilc.....	4
Chapter 3. Using the Display and Test Utility.....	6
Starting the utility	6
Main Tab.....	6
Reading and writing from/to a file or queue	6
Queue manager and queue names	7
Using the client version (rfhutilc.exe).....	7
Selector.....	8
Read Queue button.....	8
Write Queue button	8
Writing to remote queues	8
Browse operations	8
Browse Queue button.....	8
Start Browse button.....	9
Browse Next, Browse Prev and End Browse buttons	9
Close Q button	9
Open File button.....	9
File Code Page	10
Save File Button.....	11
Clear Data and Clear All buttons	11
Display Q button.....	11
Purge Q button.....	12
Save Q button	13
Load Q button	14

Move Q button.....	15
Considerations when the Move Q button.....	16
Load Names button	17
Set Conn Id button	17
Q Depth, Queue Type and Data Size	19
Cluster Open	19
User Properties	20
Put and Get options	20
Setting the user id and application context	21
Using message groups and segmentation	22
Data tab.....	22
Data Window.....	23
Data formats	23
Numeric formats.....	24
Character formats	25
Browse Next button	26
Browse Prev button.....	26
Browse status messages	26
Copybook button	26
Search Menu Items.....	26
MQMD tab	27
MQ Message Format.....	27
Code Page, Int Fmt and PD Fmt	28
User id, Accounting Token and Backout Count	28
Put Date/Time, Expiry and Feedback	28
Original Length and message sequence number	28
Message Id, Correlation Id and Group Id.....	28
Editing the message id, correlation id and group id	28
Report options.....	29
Message flags.....	29
Application context fields.....	29
Reply to queue and queue manager	30
Reset Ids button	30
Copy msgid to correlid button.....	30
PS tab.....	30
Queue manager (to connect to).....	31
Queue Name.....	31
Topic Name and Topic	31
User Data.....	31
Subscription Correlation Id	31
Remote Queue Manager.....	31
Subscription Name.....	31
Expiry	31

Priority.....	31
Level	32
Wait Interval	32
Application Identity	32
Accounting Token	32
Subscribe button	32
Resume button.....	32
Alter Sub Button	32
Publish button	33
Get Msg button.....	33
Req Pub button	33
Save Msgs button	33
Write Msgs button	34
Close Sub button.....	36
Get Topic Names button	36
Get Sub Names button.....	37
Wildcard Option.....	37
Display type	37
Subscription Options	37
Publishing Options	38
Is Retained flag	38
User properties tab	39
Moving data from the RFH Usr tab to message properties	40
RFH tab	41
Multiple RFH headers in a single message	42
RFH fixed data	42
RFH variable data	42
Include RFH V2 Header and Include RFH V1 Header	42
V2 Folders.....	42
Message Broker PubSub tab	43
Request Type.....	44
Topic(s)	44
Filter	44
Sub Point/Stream	44
Sub Name, Identity and Data	44
Queue manager to connect to	44
Queue Name.....	44
Broker Queue Manager Name	44
Subscription Queue Manager.....	44
Subscription Queue.....	44
Pub Time.....	44
Seq No.....	45
Other Fields	45

Options.....	45
Persistence	46
Clear button	46
Save to File button	46
Write Msg button	47
pscr tab.....	47
jms tab.....	48
JMS Message Type	48
Destination	49
Reply To.....	49
Correlation Id.....	49
Group Id and Sequence	49
Timestamp	49
Priority.....	49
Expiration	49
Delivery mode	49
User Defined Fields.....	49
usr tab	49
Attributes.....	50
Repeating elements	51
other tab	51
CICS Tab.....	52
Version.....	53
Program	53
Trans id	53
Next transid.....	53
Remote sys	53
Remote tran	53
UOW	53
Link type.....	53
Start code.....	53
Conversational	54
End status	54
Function	54
ADS Descriptor	54
Attn key	54
Cursor pos	54
Return code	54
Comp code.....	54
Reason.....	54
Cancel code	54
Error ofs	54
Abend code.....	54

Facility.....	55
Fac like.....	55
Format.....	55
Reply format.....	55
Code Page	55
Wait int.....	55
Data length.....	55
Flags	55
Keep time.....	55
Input item	55
Authenticator.....	56
PD Enc.....	56
Int Enc.....	56
IMS header Tab.....	56
Include IMS header	57
Map Name	57
LTerm Override	57
Authenticator.....	57
Format.....	57
Code page	57
PD Encoding and Int Encoding.....	57
Trans Instance ID.....	57
Reply Format.....	57
Trans State	57
Commit Mode.....	58
Security Scope.....	58
Flags	58
Insert length field.....	58
Dead Letter Header (DLQ) Tab.....	58
Include DLQ header	59
Reason.....	59
Orig Dest Queue Manager and Orig Dest Queue	59
Format, Code Page and Encoding fields	59
Put Appl Name and Put Appl type	59
Date/Time on DLQ	59
Prepare for resend	59
Menu options	59
File Menu	60
Edit Menu.....	60
Search Menu.....	61
Read Menu	61
Write Menu.....	61
View Menu	61

Ids Menu	62
MQ Menu	62
Accelerator Keys.....	62
Moving between tabs using the keyboard	62
Main Tab	62
Data Tab	63
MQMD Tab	63
PS Tab.....	63
Pubsub Tab.....	64
Load Q Dialog	64
Save Q Dialog.....	64
Set Connection Id Dialog.....	64
Write Pubs Dialog	64
Capture Pubs Dialog	65
Environment variables	65
Chapter 4. Performance measurement utilities.....	66
Overview.....	66
Using a client connection	66
Differences between measurement programs.....	66
Requirements for performance measurement.....	67
Differences between the driver utilities.....	68
Using the Performance Utilities.....	68
Driving the Message Flow	68
Measuring Performance	69
Measuring performance of JMS publishing applications	69
Conflicting options in the parameters file	70
Capturing message data in files.....	70
Driver memory usage	70
Driving a remote system.....	71
Measuring performance of the message broker	71
MQ Performance considerations.....	71
Persistent and non-persistent messages.....	71
MQ log operations.....	71
Queuing disk operations.....	72
Units of work	73
Performance enhancements in recent versions.....	73
Queue buffer size.....	73
Log buffer pages	74
Connections, disconnections and queue open/close	74
Suggestions for performance testing of the Message Broker	74
Parameters file for the MQPUT2, MQPUTS and MQTEST utilities	
.....	75

Command line arguments and overrides	77
Embedding RFH headers in the message data file	77
Using more than one message data file	77
Using different parameters for different message data files	78
Multiple messages in a single data file	78
Handling of RFH and RFH2 headers	78
Specifying RFH2 folder contents.....	78
RFH parameters	79
General parameters.....	81
Storing and using MQMDs in user data files	82
Warning when using MQMDs stored with the data	83
Tune parameter for the Performance Driver Utility	83
MQMD fields.....	83
Measuring latency.....	84
Measuring latency using the MQLatency utility	85
Parameters for the MQLatency utility	85
Caveats when using the MQLatency utility	86
Using the mqreply program	86
Format of the parameters file for the data capture utilities	86
Use on systems other than Windows, Solaris, Linux and AIX ...	88
Chapter 5. Reporting problems or suggestions.....	89
Using the trace facility	89
Appendix A. Sample parameters file for MQPUT2	90
Appendix B. Sample parameters file for MQCapture.....	93
Appendix D: Using message groups and segmentation	94
Message Groups	94
Message group exercise.....	94
Message Segmentation	95
Automatic (Queue manager) segmentation	96
Application controlled segmentation	97
Appendix E: Publish and Subscribe using MQ V7.....	99
Simple publish and subscribe	99
Durable subscriptions	100
Managed Subscriptions	101
Retained publications	102
Using an administered node	103
Appendix F: Using Message Filters	105

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 been 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 IBM has reviewed each item 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.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- MQ
- WebSphere
- WebSphere Message Broker
- WebSphere Business Integration Message Broker
- WebSphere Business Integration Event Broker
- IBM Integration Bus
- IBM WebSphere
- IBM WebSphere MQ
- IBM MQ
- z/OS
- WMQ
- IBM WebSphere MQ Integrator
- IBM WebSphere MQ Integrator Broker
- IBM WebSphere Event Broker
- AIX

The following terms are trademarks of other companies:

- Microsoft Microsoft, Windows 2000, Windows XP, Microsoft Visual C++
- Sun Sun, Solaris, Sparc

Acknowledgments

The author would like to thank Tim Dunn of the IBM Hursley Lab (ideas on the performance driver utility) and Neil Kolban of the Dallas Systems Center for their assistance. The author would also like to thank Phil Coxhead and Richard Brown of the IBM Hursley Lab for his help in porting and compiling the performance utilities on Solaris.

Preface

The Message Display and Test Utility (RFHUtil) is a GUI based program to assist in the development and testing of IBM Integration Bus V9 Message Flows (applications). It can display messages in a variety of formats, including XML and COBOL copybook representations. It can read data from and write data to files as well as IBM MQ queues. It can append and interpret version 1 and version 2 Rules and Formatting headers (RFH) and CICS or IMS Information headers. Command line performance measurement and regression test utilities are also provided.

Separate client and server versions of the Display and Test Utility program are provided. The Display and Test Utility is provided as two executable programs, one for client environments and the other for server environments. The command line utilities are provided as both executable programs for the Microsoft Windows® environments and as ANSI C source programs, including the Microsoft Visual C++ project and solution files.

A version of IBM MQ must be installed on the system that this program is executed on.

Although the utilities are designed for IBM Integration Bus V9, they can be used with earlier releases. Similarly, the utilities are developed and tested using IBM MQ V9, but should work with earlier versions.

This program was originally made available as SupportPac IH03. It was released as an Apache-licensed open source project to GitHub in December 2018.

Chapter 1. Introduction

To use the utility, the data portion of a message should be entered into a file. The file data is then read and can be displayed on the data tab in several different formats. If an RFH header or CICS header is desired, then the appropriate tab(s) should be displayed and the required fields completed. The data can then be sent as a message, by filling in the queue manager and queue name fields on the main tab and pressing the *Write Q* button. A message can also be read from the queue and displayed in the data tab. The data that is read from a queue can also be saved in files.

Once data is read from a file or message, it will be held in a data buffer in memory. The user data in the data buffer cannot be edited.

In addition to the GUI utility, a number of command line utilities are also available. These utilities can be used for performance and regression testing. Utilities are provided to capture messages into files, use the captured messages to drive MQ applications and to read messages from queues and report the maximum and average message rates (in messages per second).

What the RFHUtil utility does

The RFHUtil utility program provided with this repository will read data from files and/or queues and write data to files and/or queues. It will display data in a variety of formats. The user data portion of the message can be displayed in different formats but the user data itself cannot be changed. Another program must be used to create or change the user data. The utility program will format Rules and Formatting headers (RFH) headers, CICS Information Headers (CIH), IMS Information Headers (IIH) or Dead letter headers included with messages or files it reads. It can add or remove these headers as well. The new publish and subscribe functions and message properties functions supported from Version 7 of MQ are also fully supported.

There is one exception where the user data can actually be changed. Carriage return and line feed characters can be removed when a file is read if the remove CR/LF or UNIX text options found on the *Read* menu is selected. If this data is subsequently written back to a file or a message, the resulting data may be different than the original data in the file.

The supported headers can be changed, removed or added, and the contents of these headers can be changed or created.

Header chaining

The RFHUtil utility program recognizes 5 types of MQ headers. The convention is for each message header to contain the message format, character set id and encoding values for whatever item follows each header. The MQMD fields refer to the first header, and the fields in the last header refer to the user data.

When the RFHUtil program goes to write a message, it will use the format field(s) to determine which headers are present. When a header is removed, the format, character set id and encoding values of the preceding header (or the MQMD if there is no preceding header) will be changed to refer to the item which follows the header that is being removed.

When a header is inserted, the headers will be placed in the following order.

- Dead letter queue header
- RFH version 1 header
- RFH version 2 header
- IMS header
- CICS header

Chapter 2. Installation

Supported Environments

The programs contained in this repository have been written for and tested in a 64-bit Windows 7 environment. All development was done in a Windows 7 64-bit environment using Microsoft Visual Studio 2017 and IBM MQ V9.0. The programs should work with earlier versions of IBM MQ and IBM WebSphere MQ.

The MQ client and/or server must be installed on the system that the utility is executed on.

The GUI-based test and display utilities (RFHUtil.exe and RFHUtilc.exe) run as a 32-bit windows program. It should run in either a 32-bit or 64-bit Windows installation. The performance utilities are command line based, and executables are provided for Windows. These utilities can be compiled as either 32-bit or 64-bit executables.

Display Requirements

The RFHUtil.exe utility program requires a display with a minimum resolution of 800x600 if small (normal) fonts are being used in Windows and a minimum of 1024X768 if large fonts are being used in Windows. A resolution of at least 1024X768 is recommended. When used with a resolution of 800X600, the auto hide property should be selected for the task bar.

Character set considerations

The RFHUtil.exe utility program provides limited support for national languages, including some support for Unicode (multi-byte and UCS). This support may require the installation of additional code pages in the Windows environment.

In particular, single byte code pages 437, 850 and 1250 through 1257 are recognized. The replacement of non-displayable characters with periods when message data is displayed in character format will depend on the code page specified. If the code page is not recognized, then code page 1252 will be assumed. For multi-byte code pages, code pages 932 (Japanese), 936 (simplified Chinese), 949 (Korean) and 950 (Traditional Chinese) are recognized. If these code page(s) are installed on Windows, then the appropriate choices on the data display tab will be enabled and data can be displayed as Kanji characters. Finally, for XML and parsed displays only, the two-byte Unicode code pages 1200 and 1201 are recognized. If one of the Kanji character displays is selected, then the data will be translated to the appropriate multi-byte code page and the results displayed.

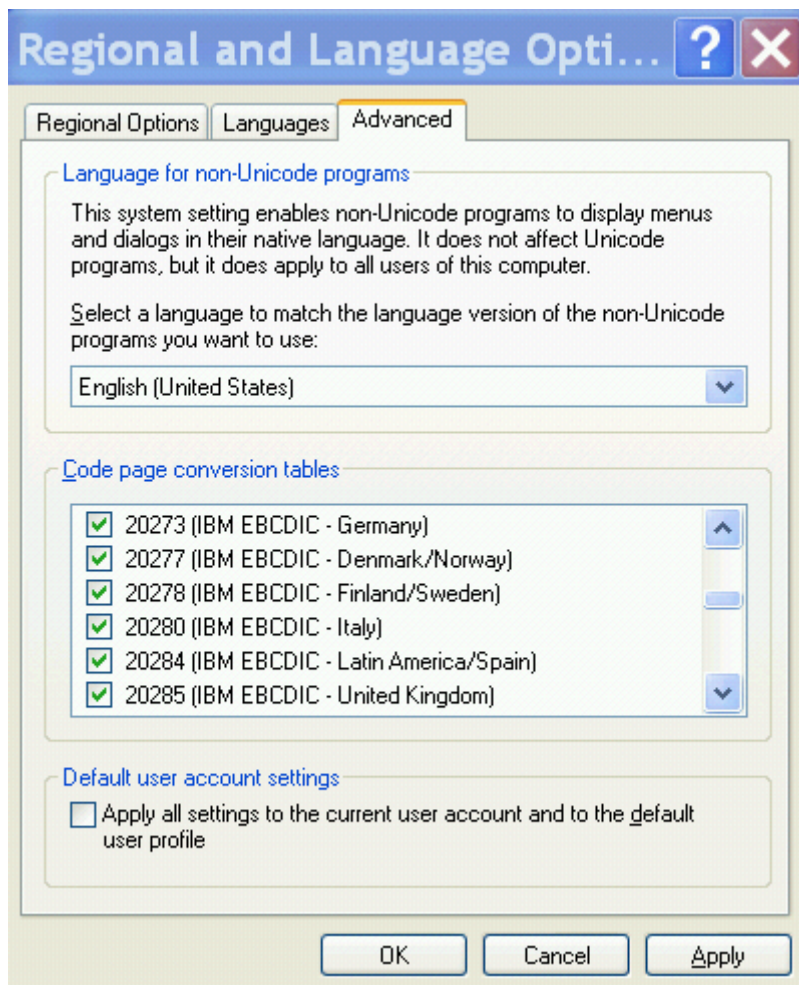
In the case of the both option, multibyte characters sequences may be split across multiple lines and will be displayed as multiple bytes rather than a single character.

Installing additional code pages on Windows XP

RFHUtil attempts to do character translations for display of EBCDIC data using Windows functions. The windows functions provide better support for national language characters. If the windows translation fails, then a built-in translation table is used, based on ASCII code page 1252 and EBCDIC code page 500.

For the windows functions to be successful, the appropriate EBCDIC code pages must be loaded into Windows. The loading of EBCDIC code pages by Windows is controlled by the Regional and Language Settings panel. To select the appropriate EBCDIC code pages, start the Control Panel and then select the Regional and Language Settings. Select the Advanced tab. Scroll down and select the desired code pages to be loaded. A screen shot is shown below.

After the desired code pages have been selected, press the OK button. The desired code pages will be loaded by Windows the next time the system is booted.



Limitations in the RFHUtil.exe utility

The RFHUtil.exe test and display utility supports most message formats and Rules and Formatting headers (RFH), there are certain limitations in what specific message types and formats are supported.

The display tabs impose certain limitations in what can be displayed or created. In particular, the following limitations should be noted:

1. For pub/sub messages, a maximum of four topics can be entered or displayed.
2. For pub/sub reply messages, a maximum of two error responses can be displayed or entered.
3. Multiple RFH2 headers or multiple RFH1 headers in the same message may not be handled correctly.

Installation

The main utility is provided as a single executable program (rfhutil.exe). A separate executable (rfhutilc.exe) is provided for use with MQ clients. The programs should be placed in an executable directory and, if desired, a program icon can be created.

Executable programs for the performance utilities are provided for Windows. The performance utilities are written in standard ANSI C and should work in most UNIX and Linux environments, assuming they are compiled for other UNIX and/or Linux environments.

In addition to the test and display utility, several command line utilities are provided to do performance measurement and testing:

- MQutT2.exe
- MQPut2c.exe
- MQTimes.exe
- MQTimes2.exe
- MQTimes2c.exe
- MQTimes3.exe
- MQTimes3c.exe
- MQCapture.exe
- MQCapone.exe
- MQCapSub.exe
- MQTest.exe
- MQReply.exe
- MQPuts.exe
- MQPutsc.exe

The MQPut2.exe program is a performance driver utility. It reads message data from one or more files into memory and then uses the data to create a test messaging workload based on the file data and a parameters file. The MQPuts program will write messages and then sleep for a specified number of milliseconds. The MQPut2 program will check the queue depth. If the queue depth is below a specified number of messages then messages will be written to top the queue up to a specified maximum. The MQPut2 program will drive applications at their maximum rate. The MQPuts will drive applications at a specified rate.

The MQTimes3.exe and MQTimes2.exe programs read messages from an output queue and reports the messages processed per second. The last two programs are special versions of the driver program. The first one writes all the test messages at once, without attempting to balance the input rate to the throughput of the test workload. Client versions are provided as well. These programs can measure latency as well, although a time stamp must be carried inserted into the message by the driver program and flow through to the measurement program.

The MQCapture.exe and MQCapone.exe utilities will read messages from queues and store them in files. The MQCapone.exe utility reads a single message and writes it to a file. The MQCapture.exe utility will read multiple messages from a queue and will store them in a file, with a delimiter string separating each message. The MQCapSub.exe utility will create a subscription and store any messages received from the subscription to a file. The author recommends that a local driver program be used if at all possible, but has provided the other two programs for cases where this is not possible. Source for these utilities is provided in the *source* sub-directory. The performance utilities should also be placed in an executable directory and program icons created if desired.

The MQTest utility is similar to the MQPut2 utility and uses the same parameter file format. However, it writes each message once. Parameters such as message count are ignored. It does not read all the files into memory but writes each file after it is read and then releases the storage before reading the next file.

The MQReply utility is provided to supply replies to request messages. It is intended to be used when a request and reply scenario is being tested. It will stand in for a back end application that would normally create the reply.

For further details on the performance utilities, please turn to the section of this document that explains their configuration and use.

In addition to the performance utilities, two additional utilities are provided that are based on the performance utilities but are not aimed at performance measurements.

Security Considerations when using RFHUtil and RFHUtilc

The RFHUtil and RFHUtilc utilities can load queue names into a drop-down list. In both cases, PCF messages are used. A temporary dynamic queue is created to receive the messages with the queue names. Messages are sent to the command queue on the queue manager.

Depending on the security that is in place, the following authorizations may be required. Note that the commands require either a -g <group> or a -p <user> option to specify either a group or a principal (user) that is being authorized.

```
setmqaut -m <QMGr> -n RFHUTIL.REPLY.QUEUE** -t q +all  
setmqaut -m <QMGr> -n SYSTEM.DEFAULT.MODEL.** -t q +all  
setmqaut -m <QMGr> -n SYSTEM.ADMIN.COMMAND.QUEUE -t q +put
```

If the above queues are not authorized for the appropriate users, then the loading of queue names will not work and the queue name drop down list will not be populated. For more information on the loading of queue names refer to the discussion in the Load Names section.

Certain options may also require explicit authorization. For example, the options to set user id or set all context require authority.

Chapter 3. Using the Display and Test Utility

Starting the utility

The RFHUtil and RFHUtilc utilities can be started from the command line or a shortcut can be created. Two positional parameters can be specified on the command line. The first parameter will be used to initialize the queue manager name on the main tab, the second parameter will be used to initialize the queue name and the third parameter will set the initial value of the remote queue manager name. Any additional parameters will be ignored.

RFHUtil <initial QM name> <initial queue name> <remote QM name>

Main Tab

Reading and writing from/to a file or queue

Five buttons are provided for the purposes of reading or writing files or messages. The program must first read data from either a file or a queue before it can do anything else. To read data from a file, press the *Open File* button. To write data press the *Save File* button. The options available on the *Read* menu should be set prior to pressing the button. To read data from a queue, enter the name of the queue. Enter the name of the local queue manager to connect to if you are not using the default queue manager or if there is no default queue manager.

Queue manager and queue names

The server version of the utility program gets the names of the queue managers defined on the system as well as their respective queue names, and then keeps this information in memory. This information is then used when the drop-down list boxes are opened to allow selection of the queue manager or queue name. This information can be refreshed by pressing the *Load Names* button. This allows queues or queue managers added since the program started to be accessed via the drop-down list boxes. A type-ahead feature (similar to command completion) is also supported for queue manager and queue names.

The implementation of the initial loading of queue names into the drop-down list is different between the client and server versions of the program. The server version executes the `dspmqr` command to get a list of available queue managers and then sends PCF commands to each running queue manager to get a list of all available queues. The client version first checks for an `MQSERVER` environment variable. If this variable is set, it is added to the dropdown list of available queue managers and an attempt is made to connect to the queue manager and use PCF commands to get a list of available queues. If the `MQSERVER` variable is not set, then the utility looks for a channel table. If a channel table is found, it is used to get a list of available queue managers but the utility does not attempt to connect to any of them. The drop-down list of available queues for any of the queue managers can be loaded with the *Load Names* button.

The *Load Names* button connects to the queue manager specified in the Queue Manager field and loads or updates the queue names for the currently selected queue manager. This works for either the client or the server version.

In all cases, queue manager and queue names can be typed directly.

In order to add cluster queues to the drop-down menu, the menu Cluster Queues menu item on the View menu must be selected and the *Load Names* button must be pressed to refresh the list of queue names.

To display the names of system queues, the System Queues menu item on the View menu must be selected. It is not necessary to refresh the list of queue names.

Using the client version (rfhutilc.exe)

The client version of the utility will use either the `MQSERVER` environment variable or an MQ client channel table. If the `MQSERVER` variable is specified, it will take precedence. In addition, the necessary connection information can be entered directly.

For each valid client connection in the MQ client channel table, an entry will be added to the drop-down list box consisting of the channel name, transport type and connection name in the same format as the `MQSERVER` environment variable.

The format of the `MQSERVER` variable is channel name, transport type and connection name, separated by forward slash characters, as shown below:

channel/transport/connection

For example, the following entry will connect to a queue manager listening on port 1414 at address 9.242.192.146 with a server connection channel called `SYSTEM.DEF.SVRCONN` and using a transport type of TCP/IP.

`SYSTEM.DEF.SVRCONN/TCP/9.242.192.146(1414)`

Please refer to the MQ client documentation for further information regarding the format of the `MQSERVER` variable.

In any case, regardless of whether the `MQSERVER` variable is set or a channel table exists, the necessary channel information can be entered in lieu of a queue manager name, using the format of the `MQSERVER` variable.

The *Load Names* button will attempt to connect to the specified queue manager. If the connection is successful, a list of queues defined on the queue manager will be requested, and this list will be used to populate the drop-down list.

Selector

The selector is a string that is used when reading or browsing a queue. It limits the messages that will be returned. The selector string itself should result in a Boolean (true or false) value. It can access properties in MQ headers and/or message properties.

Read Queue button

The Read Queue button will read a message from a queue. The name of the queue manager and queue should be filled in prior to pressing this button. The message will be loaded into a data buffer in memory. The length of the input message will be displayed in the length field on the general tab, and a message area on the bottom will indicate if the read was successful. If the MQGet fails, an error message will be displayed instead. The message data, MQMD and Rules and Formatting header (RFH) can now be displayed, using the appropriate tabs.

When a message is read into the data buffer, two radio buttons on the data tab will be set. If the code page of the input message is either 037 or 500, then the EBCDIC button will be selected. Otherwise, the data is assumed to be in ASCII and the ASCII button will be set. Similarly, if the numeric encoding format indicates host data, then the Host button will be set. Otherwise, the numeric encoding will be assumed to be in a PC format (e.g. reversed byte integers and packed decimal). The numeric encoding setting will only be used if the COBOL copybook option on the data tab is selected and the data is displayed as a COBOL copybook.

Write Queue button

The Write Queue button will write a message to the specified queue. The name of the queue manager and queue should be filled in before this button is pressed, and data should be read from either a file or another queue, using the Read File or Read Queue buttons respectively. If any settings in the MQMD other than defaults are desired, then these should be set before the data is written to a queue. In particular, if the data is in a code page or numeric format other than that of the PC that the program is running on, these fields should be set in the MQMD to match the actual data in the data buffer. If a Rules and Formatting header is to be added to the data, then the appropriate fields in the RFH page should be filled in as well. The data in the data buffer will be used for the user data portion of the message.

The message id and sequence number fields will be updated to reflect the value of the message that was written.

Writing to remote queues

There are two different ways to write to a remote queue. The first method is to create either a remote queue definition in the queue manager that the application connects to or to use a clustered queue and queue manager, which will create the remote queue definition automatically. The program can also write to a remote queue even if there is no remote queue definition on the queue manager that the application is connected to. In this case, the name of the queue manager that the queue resides on must be specified in the *Remote Queue Manager Name* field. In all cases, the appropriate channels and transmission queues must be defined on the local queue manager, either through clustering or direct definitions, and the channels must be working and available.

Browse operations

Two types of browse operations are supported. The Browse Q button will perform a simple non-destructive get operation rather than the destructive get operation performed by the Read Q button. A browse operation that spans multiple messages on the queue can also be performed. Each message in the queue can then be read in sequence.

Browse Queue button

The browse queue button is very similar to the read queue button. It will perform a non-destructive get operation rather than a destructive get. The message will remain on the

queue after the get. If the browse queue button is invoked again, the same message will be read.

Start Browse button

The start browse button is used to initiate a browse operation against a particular queue. It will read the first message from the queue, and will then enable the browse next and end browse buttons. The browse next button can be used to read successive messages from the queue.

Browse Next, Browse Prev and End Browse buttons

The browse next, browse previous and end browse buttons are normally disabled. To use these buttons, a browse operation must first be initiated with the start browse button. Once a browse operation has been started and the first message has been read, the browse next button can be used to read each successive message in the queue. After the browse next button has been used, the browse previous button is also enabled. The browse operation will be terminated when either the end of the queue is reached or the end browse button is used to terminate the browse operation.

Accelerator keys are provided for the browse next and browse previous buttons. When either the main tab or the data tab is selected, the ctrl + N key combination is equivalent to browse next, and the ctrl + R key combination is equivalent to browse previous.

The implementation of browse previous is not obvious. The MQI (MQ application programming Interface) does not support a browse previous operation. Therefore, the previous message id is retained whenever a browse next operation is performed. If a browse previous is then requested, the current browse operation is ended and a start browse operation is then performed. Messages are read until a message is found whose message id matches the saved message id. If another application is reading messages from the queue during the browse operation, it is possible that the previous message will be read before the new browse operation gets to this message. In this case, the browse previous operation will result in a no message found condition and the browse will be terminated.

Close Q button

For performance reasons RFHUtil will normally maintain the previous queue manager connection and open queue handle from the last successful operation. This is done for performance reasons. The Close Q button will close the current queue and disconnect from the queue manager.

Open File button

The *Read File* button will read the data in a file into the data buffer. The *Open File* button will read the file in binary format directly into the data buffer.

Certain menu options are provided that can change the data as it is read in. These options are located in the *Read* menu. If the *Remove Crlf* option is selected, the data will be read in binary format directly into the data buffer, and any carriage return or new line characters will then be removed. This capability allows certain types of files that contain character data only to be created and maintained in a format that allows for easier viewing with other editors, but the extra characters can then be stripped out when the data is to be used to drive an application that does not expect the extra characters.

The *Unix Text* menu option allows certain Unix formatted text files to be converted to a PC text file format. A Unix formatted text file will contain only a single new line character between lines of text, whereas a PC formatted text file expects a carriage return and new line sequence between lines of text. If a Unix formatted text file is displayed with certain PC applications (such as Notepad), then the line breaks will not be recognized properly. Once

the file has been read in with this option, it can be saved as a PC formatted file by using the *Save File* button. This operation will change the file data and cannot be easily undone

The *Remove Crlf* and *Unix Text* options should not be used to read data files in a binary format (including compressed and encrypted files), or when the message format expects carriage return or line feed characters, such as Swift FIN messages.

When the *Open File* button is pressed, a pop-up menu will allow the desired file to be specified.

When a file is read, the RFH header fields are normally reset to default values. If the *Save RFH* menu option is selected, the current values of the RFH header fields will not be changed unless an RFH header is found and the *Ignore RFH* option is not selected.

When a file is read, the program normally looks for the presence of a Rules and Formatting Header (RFH) at the beginning of the file, as described in the following paragraph. If this option is not desired, then the *Ignore RFH* option in the read menu should be selected and the program will not look for an RFH at the beginning of the file and the existing RFH fields will be preserved.

If the first four characters of data in a file are the characters “RFH” followed by a space character, then the utility will assume that the data is preceded by a Rules and Formatting header and this header will be stripped off and the appropriate fields filled in on the RFH tab. This check is performed for both ASCII and EBCDIC. The data buffer will only contain data that follows the Rules and Formatting header. The reason for this is to allow for message data that normally includes a Rules and Formatting header to be stored in a single file. Otherwise, the contents of the Rules and Formatting header would have to be recreated every time the data was read from the file.

If the program determines that there is an RFH header at the beginning of the file, then the code page and encoding for both the RFH and the message data will be set to match the RFH at the beginning of the file. The code page and numeric encoding of the user data will be assumed to match the RFH. If this is not correct, the data type on the data tab and the code page in the MQMD and/or RFH can be changed.

When data is read from a file, the ASCII and PC format radio buttons on the Data tab will be automatically selected. These buttons only affect the data displays and have no direct relationship to the data. When a message or file is written, the data that is written will be the same as the data that is in the data buffer.

When a file is read RFHUtil will attempt to associate a code page and integer encoding with the file. If a code page and/or encoding are set on the MQMD page then these values will be used. The code page associated with the file data can affect the display of the data in some cases.

If data is read from a file while a group or segment selection is in effect, then the MQMD selections and the current display options (e.g. ASCII/EBCDIC, etc) will not be reset. Otherwise, the MQMD fields and display options will be reset to default options. This preserves settings such as the persistence that cannot be changed by later messages in a group or segmented message.

File Code Page

This field allows a code page to be associated with data that is read from a file. MQMD messages have a standard field that contains the code page of the user data. If an MQMD is stored with the data then this will be used. However if a file only contains user data there is no indication of what code page the user data actually represents. This field is only used with certain display options. It is not essential for the correct operation of the program. It is only used when data is being read from a file. If this field is empty (or zero) then the MQMD ccsid field will be used. If that field is also empty (or zero) then the program will make a guess as to the code page for the data.

Save File Button

The *Save File* button will write the current contents of the data buffer into a file. If a Rules and Formatting header is specified, a valid RFH or RFH2 header will be written to the file ahead of the data in the data buffer. If just the message data is to be written to the file, then either make sure that the radio button under the *include RFH* on the RFH tab is set to *No* or select the *No RFH* option on the *Write* menu.

When the *Save File* button is pressed, a standard Windows file dialog is invoked to select the file name and location where the data will be written.

Clear Data and Clear All buttons

The *Clear Data* button will delete the contents of the data buffer only. It will not affect the contents of the MQMD nor change the Rules and Formatting Header (RFH) information. It is useful if a message has been published and a different type of publish/subscribe message is to be sent next (with no user data).

The *Clear All* button will clear the data buffer and Rules and Formatting header folders and will initialize all fields in the MQ message descriptor and Rules and Formatting header to default values.

Display Q button

The *Display Q* button will display a simple dialog. A limited amount of information is displayed about each message in the queue. If a message is a member of a group, then the G column will contain a G. If the message is a segment, then the S column will contain an S.

A message may be selected. If the *Read* radio button is selected, then a normal MQGet will be performed and the selected message will be removed from the queue. If the *Browse* radio button is selected, then a browse of the selected message will be performed and the message will remain on the queue. If the *Start Br* radio button is selected, then a start browse function will be performed, starting from the selected message.

If the *Cancel* button is selected, no read operation is performed.

When the dialog is initially displayed, the first line is assumed to be selected. If the *OK* button is pressed, and no changes are made to the dialog, a browse operation will be performed on the first message in the queue.

A maximum of 50,000 messages will be displayed. A maximum time of 15 seconds is also implemented, in which case the browsing of the queue will stop and only the messages read in that time will be displayed.

The dialog should look like the following screen shot.

Display messages in Queue TEST.OUT

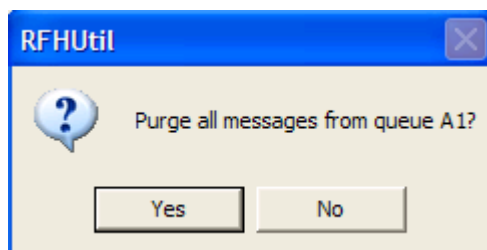
Pos	Length	Format	GS	User Id	Put Date/Time	Application	T _y
1	273	MQHRF2		jtc04	2009/07/06 16:24:33.60	BRKDEV02	8
2	273	MQHRF2		jtc04	2009/07/06 16:27:15.52	BRKDEV02	8
3	273	MQHRF2		jtc04	2009/07/06 18:37:31.07	BRKDEV02	8
4	273	MQHRF2		jtc04	2009/07/06 18:37:57.70	BRKDEV02	8
5	273	MQHRF2		jtc04	2009/07/06 18:52:24.72	BRKDEV02	8
6	273	MQHRF2		jtc04	2009/07/07 13:16:42.61	BRKDEV02	8
7	273	MQHRF2		jtc04	2009/07/07 13:30:31.52	BRKDEV02	8
8	273	MQHRF2		jtc04	2009/07/07 16:03:37.84	BRKDEV02	8
9	273	MQHRF2		jtc04	2009/07/07 16:09:55.38	BRKDEV02	8
10	273	MQHRF2		jtc04	2009/07/07 16:32:18.18	BRKDEV02	8
11	273	MQHRF2		jtc04	2009/07/07 17:02:06.60	BRKDEV02	8
12	273	MQHRF2		jtc04	2009/07/07 17:23:46.53	BRKDEV02	8
13	273	MQHRF2		jtc04	2009/07/07 17:31:04.88	BRKDEV02	8
14	273	MQHRF2		jtc04	2009/07/07 17:40:44.09	BRKDEV02	8
15	273	MQHRF2		jtc04	2009/07/07 20:22:41.46	BRKDEV02	8
16	273	MQHRF2		jtc04	2009/07/07 20:46:27.77	BRKDEV02	8
17	273	MQHRF2		jtc04	2009/07/08 13:53:55.35	BRKDEV02	8
18	273	MQHRF2		jtc04	2009/07/08 17:11:11.31	BRKDEV02	8
19	273	MQHRF2		jtc04	2009/07/08 19:46:59.08	BRKDEV02	8
20	273	MQHRF2		jtc04	2009/07/08 21:56:38.00	BRKDEV02	8
21	273	MQHRF2		jtc04	2009/07/08 21:56:44.03	BRKDEV02	8
22	273	MQHRF2		jtc04	2009/07/08 21:59:30.12	BRKDEV02	8
23	267	MQHRF2		jtc04	2009/07/08 22:01:30.60	BRKDEV02	8
24	267	MQHRF2		jtc04	2009/07/09 16:59:24.57	BRKDEV02	8
25	267	MQHRF2		jtc04	2009/07/09 16:59:56.50	BRKDEV02	8
26	273	MQHRF2		jtc04	2009/07/09 17:02:57.01	BRKDEV02	8
27	273	MQHRF2		jtc04	2009/07/09 17:05:25.80	BRKDEV02	8
28	267	MQHRF2		jtc04	2009/07/09 17:42:54.40	BRKDEV02	8
29	267	MQHRF2		jtc04	2009/07/09 18:29:24.15	BRKDEV02	8
30	273	MQHRF2		jtc04	2009/07/09 18:51:03.63	BRKDEV02	8

Read Q
Browse Q
Start Browse
Close

Please be aware that the read and browse operations use the message id of the selected message to read or browse the message. If there is more than one message with the same message id on the queue, the results of this operation are unpredictable. It is also possible that another program may have processed the message and removed it from the queue.

Purge Q button

The purge queue button will attempt to remove all messages in the current queue. It will remove the messages by opening the queue for input and then reading the queue until no more messages are found in the queue or an error is returned. The contents of any messages that are read are discarded. A warning pop-up is displayed. Select the Yes button to purge the queue.



The purge queue routine uses the syncpoint if persistent option when it reads messages. As a result a unit of work is only used for persistent messages. This is done to reduce the number of forced log writes. The number of messages is limited to either the maximum number of uncommitted messages in as specified in the properties of the queue manager or to a maximum of 2000 messages, whichever is smaller. As a result commits will be issued if there are a large number of messages to be purged. If the program fails some but not all messages may have been purged.

Save Q button

The Save Q button will save messages from a queue to one or more files. When the Save Q button is pushed, the following dialog is displayed.

The dialog box is titled "Save messages from Queue Q1 to file(s)". It contains the following fields and controls:

- File/Directory:** A text input field with a "Browse" button to its right.
- Delimiter type:** Two radio buttons: "Ascii" (selected) and "Hex".
- One/Many files:** Two radio buttons: "One file" (selected) and "File per Msg".
- Remove from Queue?:** Two radio buttons: "No" (selected) and "Yes".
- Delimiter:** A text input field containing "#@#@#".
- Properties Delimiter:** A text input field containing "\$%\$%\$".
- Start Msg:** A text input field.
- Max count:** A text input field.
- Checkboxes:** "Incl MQMD" (checked), "Incl Headers" (checked), and "Append to file" (unchecked).
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

A fully qualified file name must be provided. The Browse button can be used to navigate to a particular directory and/or file. If an existing file is used, it will be replaced.

The One file or File per msg options allow a choice between putting all messages in a single file, with delimiters between each pair of messages, or writing each message to an individual file.

If the File per msg option is selected, then the file name specified will be used, with a number inserted before any file extension for all subsequent files after the first file. The first message that is saved will use the file name that is specified.

The delimiter can be any string of characters. However, the delimiter must not exist in any data that is stored in the file, including the MQMD or other headers. The delimiter can be specified as either ASCII or hex characters. If the File per msg option is selected, the delimiter is ignored.

The Remove from Queue? options indicate whether the messages will be read and removed from the queue (select the Yes radio button) or whether the messages will be read and left on the queue (select the No radio button).

The Incl MQMD and Incl Headers checkboxes indicate whether the MQMD and any MQ headers are to be stored with the message or not. The Max count box can be used to specify a starting message number. The first message in the queue is considered to be message one. This option cannot be used if messages are being removed from the queue.

The Append to file checkbox will append messages to an existing file rather than overwriting the file if the file already exists.

If the OK button is pushed, the requested operation will be performed. If the Cancel button is pressed, the request will be ignored.

The complete msg, logical order, all avail and convert options are all honored when messages are saved. Since these options can affect the order that messages are retrieved, and the complete message option can affect the number of messages read, some care must be exercised if these options are used in conjunction with the start message number and

message count options and the queue contains messages that are segmented and/or members of groups.

Load Q button

The load Q button will write messages from one or more files to a specified queue. The individual messages will not be displayed. When the Load Q button is selected, the following dialog is displayed.

A fully qualified file name must be provided. The Browse button can be used to navigate to a particular file. If the File per msg option is selected, the name of the first file should be specified, without the message number. The message number will be inserted automatically as each file after the first is opened. If the file loading should start with a file other than the first the name of the first file should be specified in the file name with no appended suffix and the Start file number should specify the file number that was generated for the file.

The One file or File per msg options allow a choice between reading all messages from a single file, with delimiters between each pair of messages, or reading each message from an individual file. If the File per msg option is selected, then the file name specified will be used for the first file, and the file name will have an ascending number appended for each subsequent file, to make the file names unique. The delimiter can be any string of characters. However, the delimiter must not exist in any data that is stored in the file. The delimiter can be specified in either character or hex. If the File per msg option is selected, the delimiter is ignored.

The Max count box allows a maximum number of messages to be specified. Once the number specified is reached, then no further files or messages will be processed. Normally, processing will stop when all messages in either a single file or a group of files is processed.

The Write msgs once check box is used when the one file option is selected. It will write all the messages in the file once. Loading the file will stop when either all the messages in the file have been processed or the max count has been reached.

The Wait time specifies a time in milliseconds to wait after writing a batch of messages. This can be used to control the rate that messages are loaded into a queue. It is recommended that the command line performance utilities be used for performance measurements since they use more sophisticated algorithms such as keeping a queue at a specified depth as well as waiting between batches of messages. The Batch Size parameter specified the number of messages to write in a single unit of work. An MQCMIT will be issued after this number of messages has been written. If a wait time has been specified then a sleep for the specified number of milliseconds will be issued before resuming.

The Ignore MQMD will cause any MQMDs that are stored with the data to be ignored. A new MQMD will be generated. If the rate that messages are being written is being measured by the MQTimes2 utility a new MQMD must be generated at some point. The Ignore MQMD check box can be selected to use a new MQMD with a new time stamp rather than the original time stamp in the captured MQMD.

The Remove Headers options will cause any MQ headers that have been stored with the data to be removed before the messages are written. It is important to note that the RFHUtil program only recognizes dead letter headers, CICS headers and RFH version 1 and 2 headers. The Set all context option determines if any messages that were stored with an MQMD will be written with the same MQMD or if the context fields will be replaced when the message is written. The context fields include the user id and name of the application that originally wrote the message.

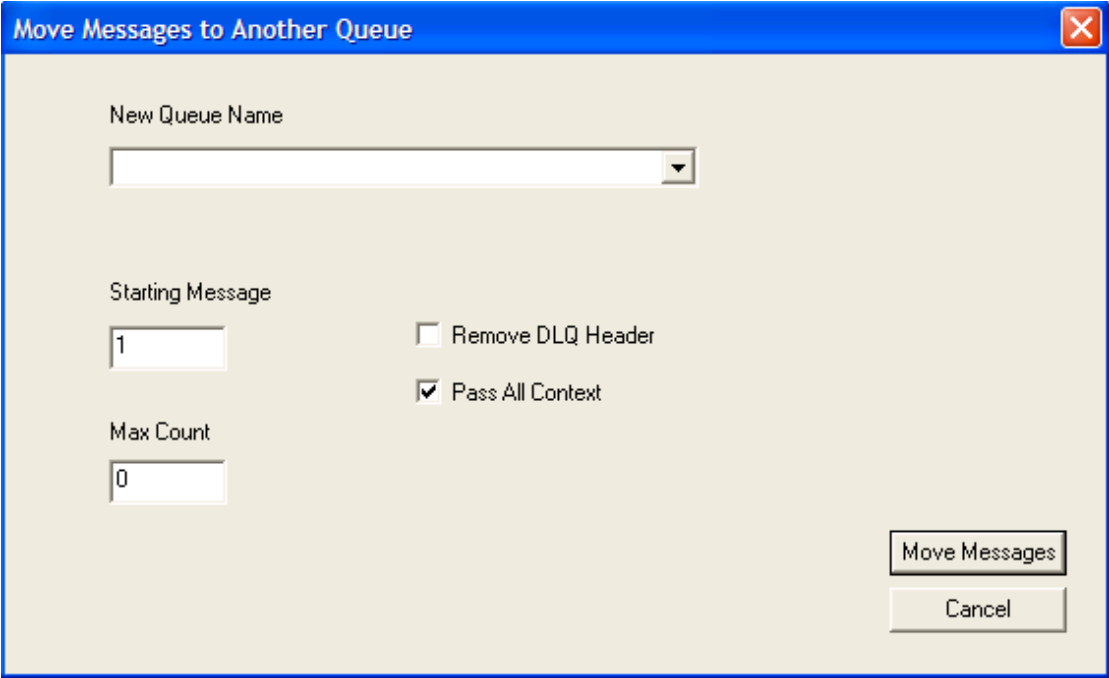
The Default values allow certain fields in the MQMD to be specified. If an MQMD is stored with the data and is used, then the default values will be ignored. If there is no MQMD stored with the data or if the MQMD is ignored, and a message header is found at the front of the data, then the format, character set id and encoding values from the removed header(s) will be used instead of the default values.

If the OK button is pushed, the requested operation will be performed. If the Cancel button is pressed, the request will be ignored.

In some cases, the messages that are written to the file may not be the same as the messages that were saved from the queue. For example, if the convert option was used when messages were saved, the messages that are written back to the file will reflect the conversion operation, and the data may be in a different code page. The Complete Msg and logical order can affect the number and ordering of messages on the queue.

Move Q button

The move Q button allows messages to be moved from one queue to another. This can involve moving all messages in one queue to another queue or some of the messages can be moved. Both queues must reside on the same queue manager. When the Move Q button is pressed the following dialog is displayed.



The dialog box is titled "Move Messages to Another Queue" and has a standard Windows-style title bar with a close button (X) in the top right corner. The main area is light gray and contains the following controls:

- New Queue Name:** A text input field with a dropdown arrow on the right side.
- Starting Message:** A text input field containing the number "1".
- Max Count:** A text input field containing the number "0".
- Remove DLQ Header:** A checkbox that is currently unchecked.
- Pass All Context:** A checkbox that is currently checked.
- Buttons:** Two buttons are located at the bottom right: "Move Messages" and "Cancel".

If the move operation involves moving all messages in the currently specified queue to another queue then the only parameter that must be specified is the new queue name. Some capability to move certain selected messages is also provided. A starting message number can be specified. The first message in the queue is considered to be message number one.

If the move operation is to start with the third message in the queue then the starting number should be set to 3. It is also possible to specify a maximum number of messages to move. If the maximum number of messages is set to zero then the move operation will continue until either the end of the queue is reached or an error is encountered.

The move operation itself is quite basic. If the starting message number is greater than one a browse operation will be started and messages will be read until the first message to be moved is found. After that messages are moved until either the number of messages exceeds the maximum number to be moved or until the end of the queue is reached.

When moving messages on a V7 or later queue manager the message properties are automatically copied. In addition any specified message filter will be used. The message filter will limit the messages that are moved. The effects of a message filter can be checked by first browsing the input queue (start browse and browse next buttons) using the specified message filter.

Certain options are used when messages are read. In particular the logical order, complete message and all available options will be used when messages are read.

Considerations when the Move Q button

It is generally problematic to move messages if messages are being added to or removed from the queue during the move operation. In particular things like the starting message number can be affected.

The movement of messages is fairly simplistic and straightforward. It reads the messages from a queue and writes them back to a different queue. By default all of the original context information is passed (the MQMD should be the same). If the starting message number is other than the first message in the queue then some number of messages will be read using a browse option but will not be moved. In a similar manner the move operation will stop if the maximum number of messages to be moved is reached. In all cases if messages are being added or removed from the queue during the queue operation or between the time the queue is displayed and the time the messages are moved then the results are hard to predict.

It is usually a good idea to display a queue before any messages are moved from that queue.

There are a number of important considerations when moving messages from one queue to another. Most of these considerations involve the selective movement of messages rather than moving all messages in the first queue.

Messages are moved using units of work. The maximum unit of work will be limited to 2000 messages, although units of work may be smaller if the queue manager has a lower limit. In most cases all messages are moved in a single unit of work. If there is an error getting or putting a message the entire operation will be rolled back and no messages will be moved. However, queue managers usually have some limit on the maximum number of get and put operations that can be contained in a single unit of work. While the queue manager defaults are usually fairly large (e.g. 10000) it is still possible to reach the limit if a large number of messages are being moved between two queues. RFHUtil will inquire to determine the maximum size of a unit of work and will issue intermediate commits if necessary. If an error is encountered getting or putting any messages after intermediate commits have been issued some but not all of the requested messages will have been moved. In no case should any messages be lost.

Messages that are part of groups and segmented messages provide different challenges. If options such as logical ordering or complete message are selected then messages in incomplete groups or segments of incomplete messages will not be moved. If these options are not selected then grouping and segmentation are ignored and physical messages are moved. Some care must be taken if a starting message number or message count is specified and options such as logical ordering are selected. In this case messages in incomplete groups will not be read and the starting number may not actually reflect the desired message. Message groups and segmentation can also change the physical order of messages in the queue if options such as logical ordering are used.

Load Names button

The load names button will refresh the list of queue manager and queue names that are kept in memory and used with the drop down lists for queue and queue manager names. This button can be used to update the dropdown menus when a new queue has been added to a queue manager, for example.

In the client version, the dropdown lists are initially empty. Pressing the Load Names button will attempt to connect to the currently selected queue manager and request a list of available queues that will be loaded into the dropdown menu. This function will only work if the queue manager is available and has an active command server. Since PCF commands are used, this will not work with Z/OS systems prior to V6.

This function requires that a connection be established with the queue manager, which requires appropriate authorization.

Set Conn Id button

The *Set Conn Id* button is used to set parameters to be used when connecting to a queue manager. The parameters include a user id and password to be entered that will be used for all client connection requests as well as SSL and security exit connection options.

This user id and password will be passed in the MQCD field, and will be made available to any authentication exits that are active. The user id and password must be set when they are required to connect to a queue manager. The user id and password are usually ignored by the channel unless a channel exit is specified for the specified channel.

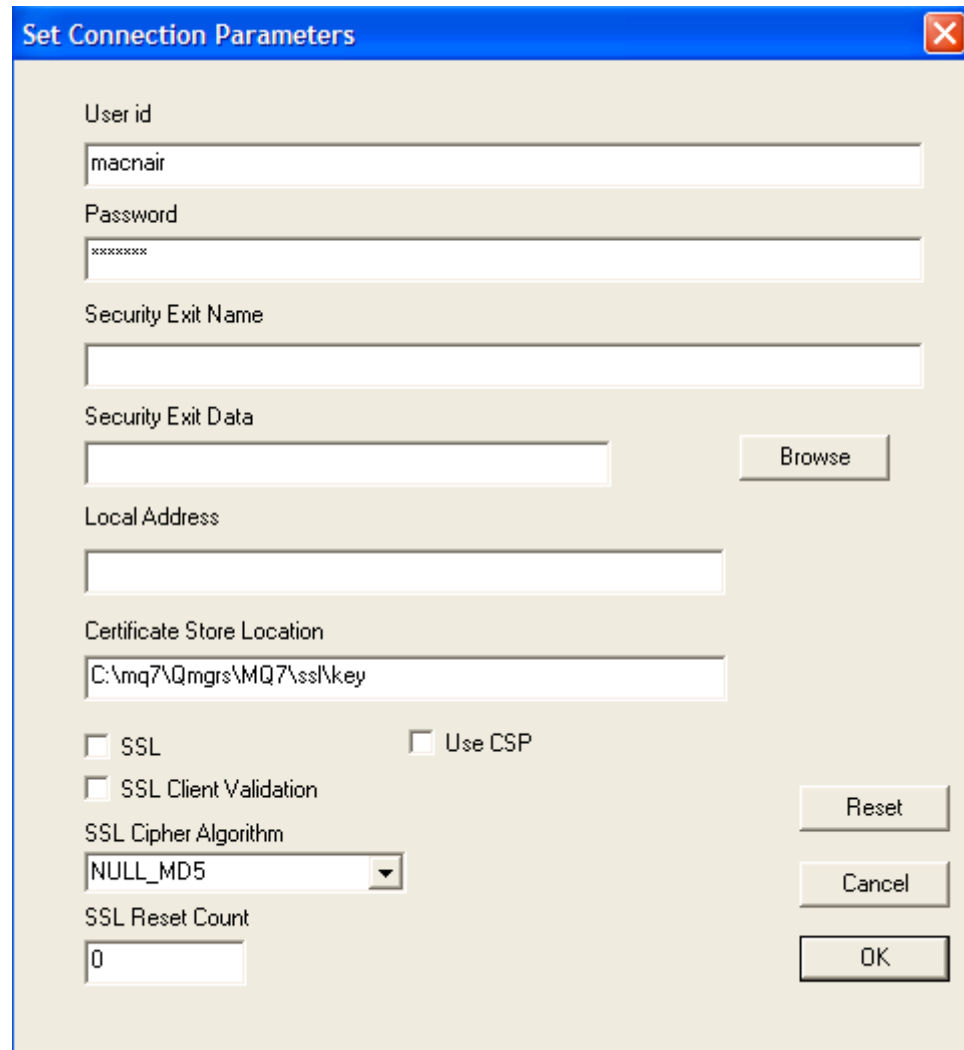
If an SSL connection is required then the SSL channel that is specified in the queue manager field must be configured to use SSL. This requires that the queue manager be configured with a key store and a certificate. The SSL cipher specification that is selected must match the configuration of the MQ server connection channel. The location of the local certificate (key) store can be specified in the dialog. The default location is taken from the MQSSLKEYR environment variable.

The SSL reset count can be set to indicate if the private session key should be regenerated. The count is the number of bytes that flow over the client channel before the key is regenerated.

The security exit name and exit data are used when a security exit is used for client connections. The exit data is ignored if no exit name is specified.

The Use CSP check box allows longer user ids and passwords to be used. If this check box is selected then user ids and passwords can be up to 512 characters long. However this requires the use of an MQCSP structure which may not be supported by all security exits.

The following dialog is displayed by the client version of RFHUtil (RFHUtilc.exe). The Browse button can be used to navigate to the location of the local certificate store.



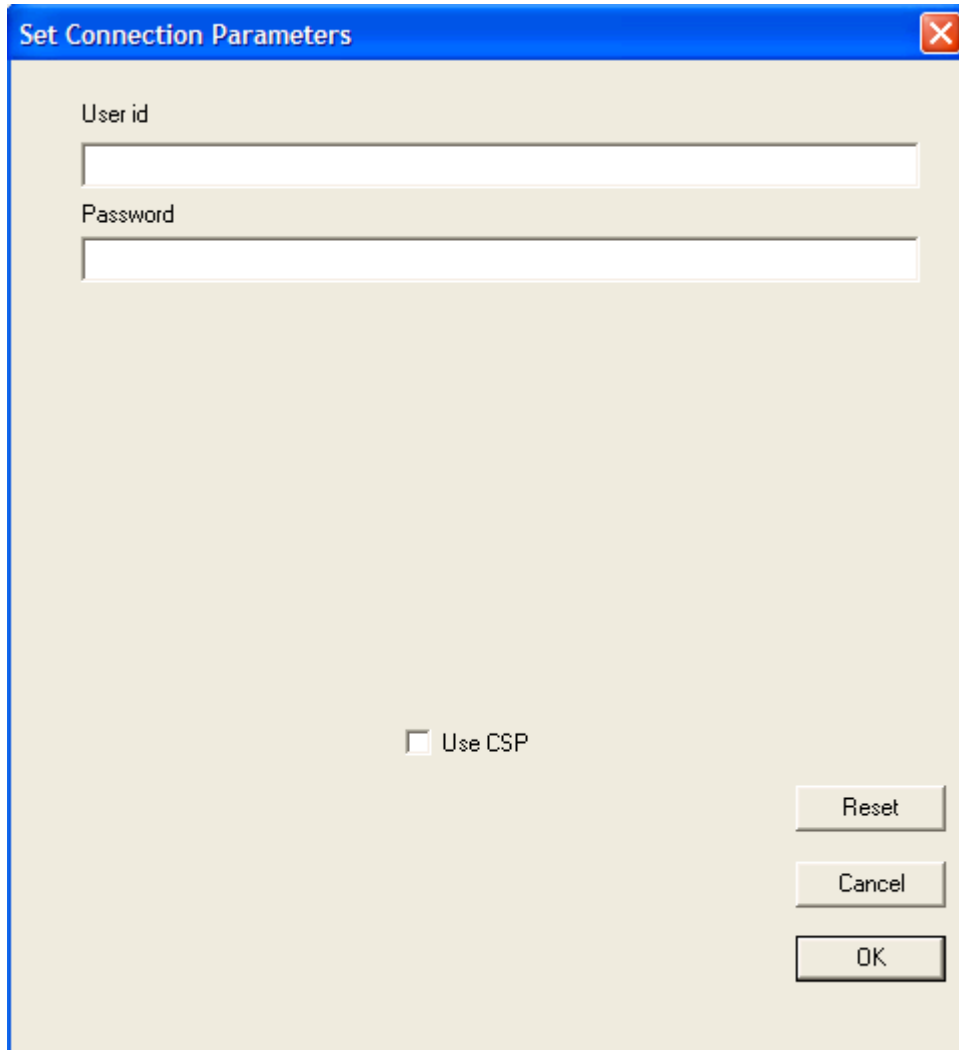
The dialog box is titled "Set Connection Parameters" and contains the following fields and controls:

- User id:** Text field containing "macnair".
- Password:** Text field containing "XXXXXXXX".
- Security Exit Name:** Empty text field.
- Security Exit Data:** Text field with a "Browse" button to its right.
- Local Address:** Empty text field.
- Certificate Store Location:** Text field containing "C:\mq7\Qmgrs\MQ7\ssl\key".
- SSL:** Unchecked checkbox.
- Use CSP:** Unchecked checkbox.
- SSL Client Validation:** Unchecked checkbox.
- SSL Cipher Algorithm:** Dropdown menu showing "NULL_MD5".
- SSL Reset Count:** Text field containing "0".
- Buttons:** "Reset", "Cancel", and "OK" buttons are located on the right side of the dialog.

Accelerator keys are available for this dialog as follows.

Ctrl + O	Browse
Ctrl + E	Reset
Ctrl + S	SSL
Ctrl + L	SSL Client Validation

The following dialog is used with the non-client version of RFHUtil. The fields that apply to client connections (TCP/IP) are not displayed.



The dialog box is titled "Set Connection Parameters" and has a blue title bar with a close button (X) in the top right corner. The main area is light beige. It contains two text input fields: "User id" and "Password". Below these fields is a checkbox labeled "Use CSP". In the bottom right corner, there are three buttons: "Reset", "Cancel", and "OK".

Q Depth, Queue Type and Data Size

The Q Depth field contains the depth of the queue after the last MQ operation was performed. The queue depth field contains the number of physical messages. If the queue contains segmented messages and the messages are read or browsed with the complete msg option, then the number of logical messages that can be read or browsed is less than the number of physical messages indicated by the queue depth field.

The queue type field will be set after all MQ operations (such as reading a message from a queue). The Data Size field contains the size of the current data that was read from a file or a message. It is for information only. The Q depth parameter is updated after a get or put operation is performed on an MQ queue.

Cluster Open

The cluster open settings will control the setting when the queue is opened. The utility currently will open the queue for each individual put, so this setting may not always produce the expected results. If there are more than one cluster queue available and there is no local copy of the queue on the queue manager that the utility is connected to, then successive messages should be sent to different queue managers.

This setting applies only to MQ put requests.

User Properties

The user properties (user props) options control the processing of MQ user properties by the application. The user properties options will be ignored if RFHUtil is connected to a V6 or earlier queue manager or client. These options require support for MQ message handles which was released in version 7 of MQ.

If the Yes option is selected message properties will be displayed on the usr props tab. A message handle will be used to extract message properties when reading a message and to include message properties with a message when writing messages.

If the none option is selected any message properties in the message will be ignored. The RFH2 option will change message properties into an RFH2 header. The properties will be placed in the usr folder. The as queue option will use the queue properties to determine whether properties should be placed in an RFH2 usr folder or should be ignored and is the default selection when RFHUtil is started.

Put and Get options

Twelve check boxes are provided to control get and put options related to the use of message groups and the segmenting of messages. Some knowledge of MQ application programming may be necessary, especially if message groups and/or segmented messages are being read or written.

The *New Msg Id* and *New Correl Id* options will set the corresponding bits when putting a message onto a queue. These options only apply to queue write operations and are ignored for any other actions.

The *Get by Msg Id*, *Get by Correlid* and *Get by Group Id* options will use the contents to the message id, correlation id and group id fields on the MQMD page to select the next message to be read. In particular, the *Get by Msg Id* option can be used to read the last message that was read by a browse operation.

If the *Segmentation allowed* option on the MQMD tab is used for MQ put operations, a single logical message may be segmented by the queue manager, if necessary. The MQ queue manager will split the message into multiple segments if necessary if the message exceeds the maximum message size specified for either the queue or the queue manager. MQ headers cannot be split between segments. If the size of a header (such as an RFH2) exceeds the maximum message size allowed, then an error (reason 2030) will be returned. This flag is carried in the MQMD and will be displayed on the MQMD tab when messages are read.

The *Logical Order* option applies to MQ message groups. When a group of messages is to be written to a queue, the *Logical Order* option should be selected on the first message. The message can then be written to the queue. After that, subsequent messages can be read from files and written to the same queue. The *Last in Group* option should be selected before the last message in the group is written. This action will complete the group. When a group of messages are to be read, the Logical Order option should be specified. This will ensure that all messages within a group will be read (or browsed) in order, even if the messages are not in physical order on the queue or other messages are interspersed on the queue.

The *Complete Msg* option will cause a single MQ get (*Read Q*) operation to return a complete message, even if the message has been segmented into multiple physical messages. If complete messages are read, the message number will indicate the number of logical messages read. The number of logical messages may be less than the number of physical messages indicated by the queue depth.

The *Convert* option will convert data from the code page in the message's MQMD to the code page specified on the MQMD tab. This option will set the convert option on the MQGET that is used to read the message. See the discussion below for more details.

The *All Avail* option will issue the MQ get request with the necessary options such that no message will be read until all segments of a segmented message are available on the local queue and/or all messages in a message group are available on the local queue.

The *Alternate User Id* option can be used to override the user id that is used to perform MQ get and put operations. If this option is selected, the user id can be specified on the MQMD page.

User properties

The User Props options will select what option is used when reading messages from a queue. These options are only used if the installed version of MQ is V7 or later.

The default option is *Yes*, which will use an MQ message handle to retrieve any message properties when reading messages from a queue. The message properties will be displayed on the *usr props* tab. In a similar fashion the *Yes* option will use a message handle to assign the message properties on the *usr props* tab to the message before writing the message to a queue. This is the only option that will display user properties on the *usr prop* tab.

The *None* option will ignore any message properties when reading messages. The *RFH2* option will convert message properties to appear in an RFH2 header. The application will see the RFH2 header at the beginning of the user data. The *Compat* (compatibility) option will either ignore MQ user properties if there are no user properties found in an RFH2 header or will add the MQ user properties to an RFH2 header if there are user properties in an RFH2 header. The *As Queue* option will use the queue specification for user properties, which can be specified as either none, RFH2 or compatibility.

Convert option and MQ data conversion considerations

The *Convert* option will set the convert option in the get message options when messages are read from a queue. The queue manager will attempt to convert the user data in the message, according to the format, character set and encoding fields in the MQMD. This is especially useful when character data is read and the format field is set to MQSTR. In fact, unless extra conversion exits or conversion formats have been added to MQ, no conversion is performed unless the format field is set to MQSTR.

The *Convert* option can be used to transform character data from one character set to another. To do this, set the desired target code page (and desired encoding, if the target code page is 16-bit (UCS2) Unicode) in the MQMD, select the *Convert* option on the main tab and read or browse the message. The converted message can then be stored as a file.

When the *Convert* option is used during a browse operation, the code page on the MQMD tab will continue to reflect the desired code page as long as the message conversion is successful. However, if the conversion is not done, then the code page will be set to the code page of the message that was read and not converted. If subsequent browse next operations are performed, a conversion to this new code page will then be attempted. In general, if a message does not have an MQ format of MQSTR, then the conversion will not be done and the code page field on the MQMD will be set to reflect the original value in the message.

The *Convert* option is ignored when the *Save Q* function is used.

Setting the user id and application context

RFHUtil supports the use of alternative user ids to get messages from a queue and the setting of a different user id when a message is written. The use of these options may require a user to have special authority options set by an administrator with the *setmqaut* utility. These options affect both the opening of a queue and the writing of a message to a queue.

The *Use Alternate Id* check box will set the alternate user id field when a queue is opened. Security checking of the open request is then performed using the alternate user id rather than the user id that the RFHUtil program is executing under. If the *Use Alternate Id* check box is selected, then the desired user id must be set in the user id field on the MQMD tab.

To use a different user id than the id that RFHUtil was started under, select any of the *Set User ID*, *Set All* or the *Use Alternate Id* check boxes. Select the MQMD tab. The user id field

should now allow input. Enter the user id to be used for the get or put operation. Return to the main tab and execute the desired MQ operation.

If the Set All check box is selected, then the application context fields can be set as well as the user id and password. If this box is selected, then the appropriate fields on the MQMD tab must be set.

For put operations, if the *Set Iden Context* or *Set All Context* options are selected, the queue will be opened with the MQOO_SET_IDENTITY_CONTEXT or MQOO_SET_ALL_CONTEXT flag selected. The user id specified on the MQMD tab will then appear in the MQMD of the output message.

If the *Use Alternate Id* check box is selected, the queue will be opened with the MQOO_ALTERNATE_USER_AUTHORITY flag selected. This applies to all types of queue operations, including purge.

Using message groups and segmentation

In most cases, the group id field in the MQMD should be left as nulls. When the group id field is null and groups or segments have been requested, the MQ queue manager will generate a unique group id and use it for each successive message. In a similar manner, the offset field will be automatically filled in when the MQ queue manager creates message segments, and the sequence number will be automatically updated when successive messages are written to the same group.

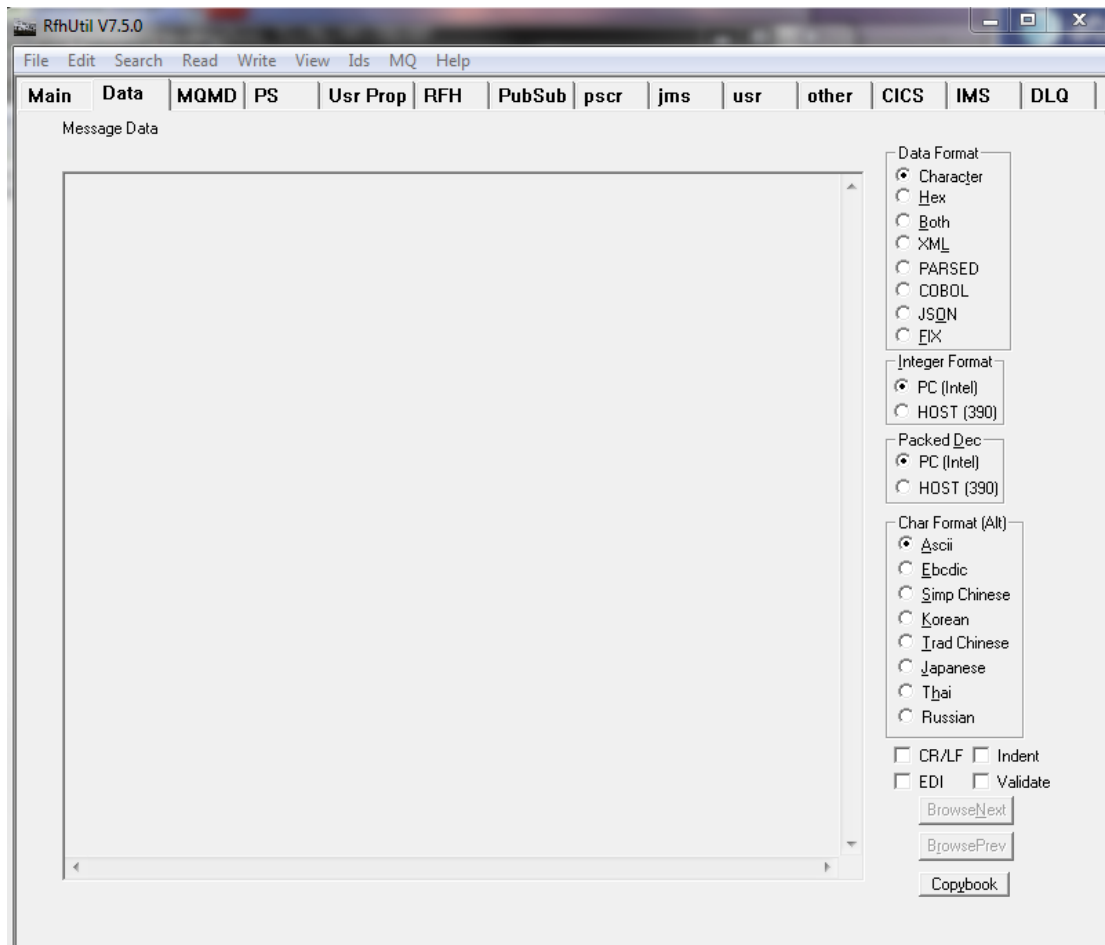
If a queue contains messages that are segmented and/or members of a group, the number of logical messages that can be read or browsed can be less than the number of physical messages on the queue. Groups and segmentation can also affect the order that messages are retrieved.

There are a number of limitations and restrictions when reading or writing a group of messages or when reading or writing a segmented message from or to a queue. A single RFHUtil session must be used to write the entire message group or segmented message. The group id should not be changed (if specified) until all messages in the group have been read. The persistence, message format, code page and encoding value must remain the same for messages within a segment, and the persistence must be the same for all messages in a group.

If persistent messages are written in a group, then a unit of work must be used. This will change the behavior of RFHUtil. Normally, messages are visible on the queue as soon as they are written. However, when a group is started and either persistence is requested or the *As Queue* option is selected, RFHUtil must open a unit of work. The unit of work will remain open until a message is written with the *Last in Group* option selected, at which time the unit of work will be committed and all the messages will now appear on the queue. If any of the MQ put operations fail for any reason or if the Close Queue button is pushed, the unit of work will be rolled back and none of the messages written in the unit of work will appear on the queue.

Data tab

The Data tab contains a large data window and a number of radio buttons that control the format that the data will be displayed in. The radio buttons will only affect the data display and do not change the contents of the data in the data buffer. The BrowseNext button is only enabled if a browse operation has been started.



Data Window

The data window displays the contents of the data buffer in the format selected by the radio buttons.

Data formats

The data in the data buffer can be displayed in six different formats. The six formats are as follows:

- Character
- Hex
- Both
- XML
- Parsed
- COBOL
- JSON
- Fix

The character, hex and both formats apply to any data format. They show the raw data in either character or hex format, or in both simultaneously. The character display will substitute periods for any non-printable characters. In all cases, an offset will be displayed on the left side, and the data will be shown in fixed length groups, with spaces inserted periodically in the display to improve legibility. Vertical scroll bars allow for large data buffers to be displayed.

If the data in the data buffer is in a valid XML format, the XML and Parsed options can be selected. The XML format will attempt to display the data as a valid XML hierarchy, with

lower level items indented. The Parsed format will display the data in a format that looks like the ESQL statements that would be used to create the given data contents. It is usually easier to see the data values when the parsed format is used rather than the XML format.

The XML and parsed formats should only be selected for data that constitutes a well-formed XML message. Attempts to use these formats for data that is not in XML format or for XML data that is not well-formed (e.g. contains XML formatting errors) will usually result in an error message, but may result in unpredictable results.

The XML and parsed formats will remove any comments and imbedded DTDs. A limited validity check is performed on the data before attempting to format the data. This checking consists of verifying that there is at least one less than sign, that the first less than sign is found before the first greater than sign and that the number of less than signs matches the number of greater than signs. Carriage returns, line feeds and tab characters found in data or attribute values will be replaced with blanks. User defined entity values found in DTDs will be ignored and the data will be displayed in its raw format.

The COBOL option will match a given data area with a COBOL copybook. To use this option, the COBOL copybook must be available on the local system. When this option is selected, a popup menu will allow the copybook to be specified. The data will then be matched against the given copybook. If a field is specified as packed decimal but the data is not in a valid packed decimal format, an error message will follow the display of the particular field. Character data will be assumed to be in the format specified (e.g. ASCII or EBCDIC). If EBCDIC is specified the data displayed in character fields will be translated from EBCDIC to ASCII before being displayed. Packed decimal and integer data will be assumed to be in the format specified by the numeric format radio buttons (e.g. Host or PC/Intel). If the wrong numeric format is selected, most packed decimal fields will generate error messages. The correct numeric format button should then be selected, and the error messages should disappear.

The JSON option will display a message that uses the JSON message format in a formatted display. This format should only be selected if the input data is in a JSON format.

The Fix format is used to format messages that are in a FIX format. Each line of data (terminated with a binary X'01" character) on a separate line. It should only be selected if the data is a FIX message.

If the ASCII button is selected, the character data will be displayed as found in the data buffer. If the EBCDIC button is selected, the data will first be translated from EBCDIC to ASCII before it is displayed. For COBOL copybook displays, data in binary or packed fields will not be translated.

The CR/LF check box will cause the display to use carriage return and new line sequences in the data buffer rather than displaying a fixed number of characters per line. There are certain types of data that are easier to read when this option is selected, such as Swift FIN messages.

If the EDI check box is selected and the standard EDI sequences are found at the front of the data area (first three characters of the data), the line termination character in the message will be used to determine the line breaks.

The indent check box is used for displays in a COBOL format. If this option is selected, then lines will be indented to reflect the hierarchy of the associated copybook.

The display data formats do not support Unicode data. The hex display can be used to view Unicode data in a raw format.

Numeric formats

The numeric format radio buttons are only used when data is to be displayed as a COBOL copybook. The integer buttons control the format of integer and floating point data, and the packed decimal buttons control the interpretation of packed decimal data. These selections will indicate whether binary and packed decimal numbers are in PC (Intel) reversed format or Host (390) / Unix format. This selection will not affect the data in the data buffer.

Three options are provided to handle the common encoding schemes for binary, packed and floating point numbers. The choices of PC, Host and Unix correspond to an encoding value of 544, 785 and 273 respectively. The difference between the Host and Unix selections is the format of floating point numbers (390 vs. IEEE). No data transformation is provided for floating point numbers. They can be displayed in hexadecimal formats only. Integer and packed decimal representations are the same.

The numeric encoding format specified in the MQ message descriptor should indicate the numeric encoding of the user data. If the data is all character data, then this setting has no meaning and is ignored.

Character formats

The character format radio buttons indicate whether the data in the data buffer should be translated from EBCDIC to ASCII before being displayed. It is used with all display formats except hexadecimal displays. This selection will not affect the data in the data buffer.

Four additional character formats can be used to select a font that is appropriate for the display of Chinese, Korean, Thai and Japanese characters. In each case, the appropriate font must be installed on the system, and the data must be in multi-byte Unicode format (code page 1208).

CR/LF selection

This selection is only used with the character display option. If this option is selected, then carriage return and line feed sequences will cause the subsequent characters to be displayed on the next line, rather than display a fixed number of characters on a line.

EDI selection

This selection is only used with the character display option. The data must be a valid EDI format, including X.12 and EDIFACT. The data must start with the characters “UNA”, “UNB” or “ISA”; otherwise this option is ignored.

Indent selection

This selection applies to COBOL copy book displays only. If selected, then successive levels will be indented in the display.

Validate

The validate option is only used with the COBOL display option. If selected, numeric fields (PIC 9(nn)) will be checked for valid data. Packed decimal data is always validated.

Data Translation

For convenience when working with EBCDIC (primarily host) data, a limited translation capability from EBCDIC characters to ASCII characters is provided. This translation is internal to the utility and does not support individual national code pages. It will translate numbers, standard letters and most special characters. This is provided to enhance the display and viewing of character data that has originated or is intended for a mainframe. The user data in a message or file is not altered by this translation. The translation of data is for display only. In addition to the data display, the header and data fields of an RFH or RFH2 header will be translated.

Binary data in either a COBOL copybook display or an RFH or RFH2 header will be transformed if the encoding parameter in the MQMD indicates that the numeric format is host (big-endian) rather than PC (little-endian) format.

Unlike MQ messages, files do not indicate the code page of the data. If data is read from a file on a PC, the data will be examined and the utility will make a guess if the data is ASCII or EBCDIC and will set the initial display options as a result of this determination. If this determination is not correct, the display option radio buttons should be changed to match the

data. To allow EBCDIC data to be stored in PC files and recognized as such, the program will assume that any data in a file with an extension of “ebc” contains EBCDIC data and the code page and display characteristics will be set for EBCDIC rather than ASCII data. In all cases, the data itself will not be transformed.

Browse Next button

The browse next button is normally disabled. It will be enabled when a browse operation has been started using the start browse button on the general tab. When enabled, the browse next button will read the next message from the queue. If there are no more messages on the queue, the browse operation will be terminated.

The Ctrl + N key combination can be used as an alternative to the Browse Next button.

Browse Prev button

The browse prev button is normally disabled. It will be enabled when a browse operation has been started using the start browse button on the general tab or the start browse button on the display queue dialog; and at least one browse next operation has been performed.

When a browse next operation is performed, the message id of the previous message is retained. When the browse previous button is pressed, the current browse operation is terminated, and a new browse operation is started from the beginning of the queue. Messages are read from the queue until a message with a matching message id is found. As before, the previous message ids are retained. If the previous message is the first message in the queue, then the browse prev button will be disabled until a browse next operation is performed. When enabled, the browse prev button will attempt to read the previous message from the queue.

If a message matching the message id of the previous message is not found, the browse operation will be terminated.

The Ctrl + R key combination can be used as an alternative to the Browse Prev button.

Browse status messages

When a browse operation is started and when a browse next or browse previous operation is performed, a status message is displayed indicating the message count and total messages on the queue. The status is displayed in both the messages box on the main dialog and in the title of the data display on the data tab. The message count is maintained within the program for the duration of the browse operation. The total messages count is updated after each browse next or browse prev operation. If another program is writing messages to the queue or reading from the queue, the number of message of messages on the queue will change. The message count is not aware of messages being read from the queue, but rather reflects the results of the current browse operation.

Copybook button

This button allows the copybook file to be changed. When the COBOL button is first selected, a popup menu is displayed that allows the copybook file to be selected. This button allows the copybook selection to be changed.

Search Menu Items

A search menu is available when the data area is displayed. There are two find options, find and find hex. The find option will accept a character string and search for the data. The find hex option is similar to find but the search argument is specified in hex characters. The go to item will move the current display line to a specific offset within the data item.

The go to and find hex items is only available for the character, hex and both display options. The find option will search the actual display string rather than the actual data buffer when the display type is set to XML.

MQMD tab

The MQMD tab contains fields that display the fields in the MQ message descriptor. In some cases, the contents of the display can be changed, such as the code page. When data is to be written to an MQ queue, the code page should be set to match the actual data in the data buffer. If the code page or numeric formats are set incorrectly, subsequent processing of the message data may fail or generate incorrect results. Changing the fields in the MQMD does not change the data in the data buffer.

Before a message is written to a queue, the code page should be set to match the first MQ header if present; or the data in the data buffer if no header is present. The encoding should be set to match the encoding sequence of any numeric (e.g. binary or packed decimal) data in an MQ header if present; or the data in the message if no MQ header is present. Finally, the message persistence should be selected. The defaults for these values are for a non-persistent message with code page set to 437 (PC ASCII) and numeric encoding set to match a PC.

MQ Message Format

If no Rules and Formatting header is to be used with this message, then the format field should be set to indicate the message format. The standard MQ message formats should be entered into this field. If the data is all character data, then the format field should be set to "MQSTR ". If the message starts with a standard MQ header, then the appropriate message format should be entered.

In all cases, the format field should reflect the format of the user data. If an RFH or RFH2 header is included with the message when written to a file or queue, then the contents of this field will actually be found in the RFH or RFH2 header. The MQ message descriptor format field in this case will contain a value of MQHRF or MQHRF2 respectively. This will be done automatically by the utility.

Code Page, Int Fmt and PD Fmt

These fields are used to set the appropriate values in the MQMD that should match the user data (or the first MQ header, if there is at least one). The code page reflects the representation of character data, while the integer (Int) and packed decimal (PD) formats reflect the byte order that is used for binary data (primarily integer, floating point and decimal data).

The byte order selections are translated to the appropriate bit settings in the encoding field. The integer format field can be set to reflect either little-endian byte ordering (typical on Intel PCs) or big-endian byte ordering (such as mainframe systems). The Host setting will set the encoding to use big-endian format.

User id, Accounting Token and Backout Count

These three fields are taken directly from the MQ message descriptor when a message is read from a queue and are for display purposes only. The user id and accounting token can be changed if the Set User Id or Set All option is selected.

Put Date/Time, Expiry and Feedback

The Put Date/Time field will display the date and time that the message was written to the queue. This field is taken from the MQ message descriptor when a message is read from a queue and is for display purposes only.

The Expiry field is used to display the expiry field in the MQ message descriptor when a message is read from a queue. It is also used to set the message expiry when a message is written to a queue and the expiry is set to a value greater than zero.

The Feedback field will display the contents of the feedback on any read operations and can be used to set the feedback on any put requests. The feedback field must be a number.

Original Length and message sequence number

These fields are loaded with the values from the MQMD when a message or a file with a stored MQMD is read. While the fields can be modified in most cases MQ will ignore the contents of these fields when writing a message. For details please consult the MQ application programming reference.

Message Id, Correlation Id and Group Id

The message id, correlation id, group id and sequence number fields are used to display the respective fields in the MQ message descriptor. These fields can be displayed in character (ASCII or EBCDIC) format or in hexadecimal format, by selecting the appropriate radio button in the *id Display* group.

The sequence number field is for display purposes only. The correlation id and group id fields will be used when messages are written to a queue, if they are set to a value other than binary zeros. The message id field can be used to set the message id of the message that is written if the New Msg Id selection on the main tab is removed.

Editing the message id, correlation id and group id

The message, correlation and group ids can be edited. The editing can be confusing, since the values can be displayed in ASCII, EBCDIC or hex, and the data may contain non-printable characters, such as nulls (binary zeros). A modified Windows edit box is used to

display the data. If the display is in ASCII or EBCDIC, no data would normally be displayed after the first null character. To show only printable characters, any null or other unprintable characters are replaced with blanks, and trailing blanks are then removed. This will display all printable characters but causes problems when the data is changed.

Standard edit controls in Windows do not support an overwrite mode. The modified edit control normally uses overwrite mode. This preserves the length of the id fields at the required 24 bytes (48 hex characters).

When the data is changed, each individual character is compared to the previous value, and only characters that have different values will be updated. If characters are deleted, by using the delete key or the backspace key, some characters after the deleted character may appear to have been changed and will be updated with the values in the edit box display. For example, some null values may be changed to a space character. This problem can be avoided if all editing is done with a hex display.

Report options

Six report options are supported as check boxes. When a message is read from a queue, the appropriate report option check boxes are set depending on the contents of the report field in the message. When a message is written to a queue, the report field is set to match the report option check boxes. If any report options are set, then the reply queue name field must be set. The reply queue manager field may also need to be set.

The supported report options are exception, expiration, confirm on arrival (COA), confirm on delivery (COD), positive activity notification (PAN), and negative activity notification (NAN) and pass correlation id. The new report options introduced in WebSphere MQ V6.0 are also supported. The pass correlation id option instructs the reply application to set the correlation id to be the same as the correlation id in the request message.

When a request message with a publish and subscribe header is sent to the WebSphere Message Broker, the PAN and NAN report options will be honored and a report message will be returned to indicate if the request was processed successfully or not.

Additional report options were introduced in WebSphere MQ V6.0. These options can be used to provide a trace route type of function. When the activity option is specified, a report message will be returned by each channel that handles the message. If the discard or discard and expiry options are selected the request message will be automatically discarded when it arrives at the ultimate destination or when the message is expired.

Message flags

The message flags indicate if a message is a segment of a message and if the message is part of a message group. The segmentation allowed flag indicates if the message can be segmented into multiple physical messages by a queue manager. This is done if the size of the message exceeds the maximum message size of a queue and or queue manager. The segmentation may be performed by a remote queue manager if the destination is a remote queue.

Application context fields

The application context consists of four fields. The four fields are application identity, name, type and origin.

These four fields display the respective fields from the MQ message descriptor after a message is read from an MQ queue. These fields are normally for display purposes only. If the *Set All* option is selected on the main tab, then these fields can be changed and the value of this field will be used for any *Write Q* operations. If the application type is set, it must be set to a number. The drop-down menu can be used for standard types. User types can be entered as a number.

Reply to queue and queue manager

These two fields contain the name of the reply to queue manager and queue, as specified in the MQ message descriptor when a message is read from a queue. These fields can also be used to set these two fields when a message is written to an MQ queue. Both names can be a maximum of 48 characters.

Reset Ids button

This button will clear the message id, correlation id and group id fields and set the values to nulls (binary zeros).

Copy msgid to correlid button

This button will copy the message id field to the correlation id field. This button is particularly useful when used with the *Get by Correlid* option on the main tab.

PS tab

The PS tab supports the new publish and subscribe functions that are contained in MQ V7. The functions allow subscriptions to be created and used and for messages to be published and received. The functions on this tab require that version 7 of MQ be installed on the system that RFHUtil is running on and that the queue manager that RFHUtil is connected to must be a V7 queue manager.

RFHUtil V7.5.0

File Edit Search Read Write View Ids MQ Help

Main Data MQMD **PS** Usr Prop RFH PubSub pscr jms usr other CICS IMS DLQ

Queue Manager (to connect to) Expiry Priority Level Wait Interval

Queue Name Application Identity

Topic Name Accounting Token

Topic

User Data

Selection string

Subscription Correlation Id

Remote Queue Manager

Subscription Name

Display type

Wildcard Option

Managed Sub Durable Sub Group Sub On Request Remove on Close Warn No Subs Set Identity Any UserId Set Correlid New only Local No Multicast Retained Pub Is Retained Not own pubs Suppress Reply To New Correl Id

Subscribe Resume Alter Sub Publish Get Msg Req Pub Save Msgs Write Msgs Close Sub

Get Topic Names

Get Sub Names

Clear All

Queue manager (to connect to)

The queue manager name is the name of the queue manager that RFHUtil will connect to. In order to use the functions on this tab both the system that RFHUtil is running on as well as the queue manager that RFHUtil is connecting to must be running at V7.

Queue Name

The queue name is the name of the queue that the subscription will use. In the case of a managed subscription the queue name is generated

Topic Name and Topic

The topic name is the name of a predefined node in the topic tree. An administrator can define certain nodes within a topic tree as administered nodes. The administered nodes can have properties assigned to them. They are generally used for security purposes. Administered nodes are given a name (up to 48 characters). If a topic name is specified the node serves as the base (root node) that the topic string starts from. The topic name does not have to have any relationship with the actual location within a topic tree.

The use of topic names is optional. The entire topic string can be entered within the topic field. If a topic name is used then the topic string should be relative to this point in the topic tree.

After a subscription is made (by pressing the Subscribe button) the topic string will be updated to reflect the full topic string and the topic name field will be cleared.

User Data

User data can be entered with a subscription request. The user data is returned if a subscription is resumed.

Subscription Correlation Id

When a subscription is made a correlation id is returned. This correlation id will be used by any messages that are published as a result of this subscription.

Remote Queue Manager

This field can be used to specify a destination queue that is defined on a different queue manager than the broker when processing a subscribe request. If the destination queue is on a different queue manager then the Get Msg button cannot be used. This field is ignored for a managed subscription.

Subscription Name

When a subscription is created it has a name associated with it. This name is used to reconnect to an existing (durable) subscription. The subscription name must be unique. The name can be up to 48 characters long and must follow the same rules as queue and queue manager names.

Expiry

If the subscription is to expire after a given time enter the amount of time in tenths of a second until the subscription should expire. For example if a subscription should expire after 10 minutes enter 6000 (10 min X 60 sec/min X 10 tenths/sec).

Priority

If the published messages are to have a specific MQ priority then enter the priority (single digit) to be used for messages published as a result of this subscription.

Level

The level is used to limit the subscribers that a message will be sent to. If the level of a published message is greater than the level of the subscription the message will not be sent to the subscriber even if the topics match. The default subscription level is 1. The level must be from 0 to 9 inclusive.

Levels can be used to intercept published messages. In this case the interceptor would subscribe at a higher level than any other subscribers. The messages could then be published at the interceptor level. The interceptor could process the message in some way and then republish it. An interceptor at level zero could recognize that no normal subscribers have an interest in the published message. This can detect situations errors or situations where messages are being published that are not necessary.

Wait Interval

The maximum time to wait for a message after the Get Msg button is selected. The wait interval should be specified in milliseconds.

Application Identity

The application identity specifies the value to be placed in the corresponding field in the MQMD for messages that are published for this subscription. This field is ignored if the Set Identity option is not selected. This field can be up to 32 characters long.

Accounting Token

The accounting token specifies the value to be placed in the corresponding field in the MQMD of messages that are published for this subscription. This field is ignored if the Set Identity option is not selected. This field can be up to 32 bytes (64 hexadecimal characters) long. It is specified as hexadecimal characters.

Subscribe button

This button will send a subscription request to the queue manager. The subscription name is used to name the subscription and is required. A topic name and/or topic (string) is also required. Subscriptions can include filter values, which are specified on the main tab.

Resume button

The resume button is used to reconnect to an existing durable subscription. The name of the subscription must be entered in the subscription name field before this button is used. After a successful resume the subscription fields will be populated from the subscription.

The Managed Sub option must be selected so as to match the original subscription. If the subscription was not managed then the queue name of the subscription queue must be specified. If the Get Sub Names button and the drop down list are used to select the subscription name then these options will be selected automatically. If the name is typed then the managed sub and queue name fields should be set to match the subscription.

It is strongly recommended that the Get Sub Names button be used before selecting the subscription name using the drop down list. The Get Sub Names button uses PCF messages to retrieve a list of subscriptions from the queue manager. In addition to retrieving the subscription name, the queue and queue manager names as well as whether the subscription is managed or not are filled in automatically from the PCF response messages. These fields must be properly set for the resume operation to complete successfully.

Alter Sub Button

The alter button modifies certain options on an existing subscription. Not all options can be modified. In order to alter a subscription the subscription must first have been created or resumed.

Only durable subscriptions can be altered, since the subscription must first be closed before the alteration can be issued and the close would remove a non-durable subscription. As a result the alter button is only enabled for durable subscriptions.

Publish button

The publish button is used to publish a message on a specified topic. The topic should be specified in the topic field. The data must be read into RFHUtil prior to using this button. Any MQMD values or other headers must also be set prior to publishing the message.

Get Msg button

The Get Msg button can be used after a subscription is established, either by creating a new subscription with the Subscribe button or resuming an existing durable subscription with the Resume button. A message will be read from the subscription queue each time this button is pushed. If no message is available the request will wait for up to the number of milliseconds specified in the wait interval field.

Req Pub button

The request publication button will send a request to the publish/subscribe broker to publish any current retained publications that match the subscription. The request publication may result in one or more retained publications being sent to the subscription queue, where they can be read by pressing the Get Msg button. Any such messages should have the MQIsRetained property set to one (true).

Save Msgs button

The Save Msgs button will display a dialog box that will save messages published to the subscription into a file. This is similar to the Save Q button on the main tab. In order to use this button a subscription must be created first using either the Subscribe or Resume buttons. When this button is pushed the following dialog is displayed.

Capture Published Messages

File/Directory:

Delimiter type:

- ☒ Ascii
- ☐ Hex

Delimiter:

Properties Delimiter:

Max count: Max Wait Time:

☒ Include Topic
☒ Include Properties
☒ Include MQMD
☒ Include Headers
☐ Append to file

The File/Directory name must specify a file name and optionally a directory name to store the captured messages in. If the Append to file check box is selected and the file exists then messages will be appended to the existing file. If the Append to file check box is not selected then any existing file will be overwritten.

The Delimiter type indicates if the Delimiter and Properties Delimiter fields are specified as ASCII or hexadecimal characters. Both fields can contain a maximum of 32 ASCII characters or 64 hexadecimal characters (32 bytes in either case). Both delimiters must contain a unique string of characters that will not occur in either the user data or in any MQ header fields (including the MQMD). The Delimiter field specifies the characters that are used to indicate the end of one message and the beginning of the next. The Properties Delimiter field specifies the characters that are used to separate the user properties from the rest of the message. This field is ignored if the Include Properties field is not checked or if the version of the queue manager or MQ client does not support user properties (e.g. is older than MQ Version 7).

The Max count specified the maximum number of messages to capture. Once this number of messages has been captured the capture operation will stop automatically. If the Max count field is zero or blank then messages will be captured until the operation is stopped using the Stop Capture button or when the Max Wait Time is exceeded. The Max Wait Time specifies a maximum number of seconds to wait before capture will stop automatically. Each time a full second elapses without any message being captured will count as one second of waiting. Once the number of seconds with no messages being captured reaches the Max Wait Time specification then the capture operation will stop automatically. If the Max Wait Time field is zero or blank then there is no time out and the capture operation will not stop based on time.

The Include Topic parameter indicates if the MQTopicString property will be captured as a user property and stored with the message. If this check box is selected a Properties Delimiter must be specified.

The Include Properties indicates if MQ user properties are to be included with the captured messages. If this check box is not selected then MQ user properties will not be captured. If this check box is selected then a Properties Delimiter must be specified.

The Include MQMD check box indicates if the MQMD associated with a message should be stored with the message data. The Include Headers check box indicates if MQ headers are to be included with the captured data. If this check box is not selected then MQ headers will be removed and not stored with the captured messages.

The Capture Msgs button will start the capture process. A file name must be specified before the capture operation can start. Once this button is pushed any messages that match the subscription topic string will be written to the specified file, with delimiters written between messages. The capture process runs as a separate thread. Once a capture operation has been started the Stop Capture button will be enabled. This button can be used to stop a capture operation at any point. A capture operation will also stop automatically once the specified number of messages or the specified wait time has been exceeded. Since MQ publish and subscribe is queue based the messages that are captured could have been published some time previous to when the capture operation is actually started.

The Cancel button will exit the dialog without starting a capture operation.

Write Msgs button

The Write Msgs button will display a dialog that will publish messages that were previously captured into a file using the Save Msgs button. This is similar to the Load Q button on the main tab.

If topics were captured with the messages they can be used when the messages are re-published. A default topic can be specified in the topic field. This topic is used when there is no captured topic for a message or if the option to ignore saved topics is selected in the dialog. If a default topic is specified it cannot use wildcards.

The File/Directory field must contain the name of the file that contains the message(s) that are to be published. Each message is separated by a delimiter sequence. The delimiter must match the delimiter that was used when the file was created. If the file contains only a single message then no delimiters are required.

If user properties were captured with the messages then the Properties Delimiter must be specified and must match the delimiter that was used when the messages were captured.

There are two user properties that are treated specially. When a message is published by MQ it sets a user property (MQTopicStr) that contains the topic that the message was published under. If the Use Topic in file check box is selected then the message will be published under the topic specified in this property rather than the default topic. This may require that the current topic be closed and a new topic object opened. This topic then effectively becomes the default topic. If the Use topic in file check box is selected and all messages contain an MQTopicStr property then no topic or topic string need be selected on the PS tab. If the Use topic in file check box is not selected then the topic and topic string combination specified on the PS tab will be used and any MQTopicStr properties in the message file will be ignored. In all cases this property will not be added to the message. The second property that has special handling is the MQIsRetained property. This user property is set if a published message came from a retained publication rather than a normal publication. In all cases this user property will be captured but will be ignored when the message is replayed.

If MQMDs were captured when the message file was created the captured MQMD will be used when the messages are replayed if the Use MQMDs in file check box is selected. If this check box is not selected then any MQMDs that were stored with the message data will be detected but will be ignored.

If user properties were captured when the message file was created then the captured properties will be added to any published messages if the Use properties in file check box is selected. If this check box is not selected then any captured user properties will not be inserted into the messages as they are published.

The Wait Time and Batch Size fields control the rate that messages are published. The Batch Size field specifies the maximum number of messages that will be published as a single unit of work as well as the number of messages to be published before issuing a wait. The Wait Time parameter specifies the number of milliseconds to wait after the specified number of messages has been published. Once this number of messages is published then

a wait for the specified number of milliseconds will be issued. At that point the cycle will repeat. If the Wait Time field is blank or zero then no wait will be issued and messages will be published as fast as possible. If topics are specified in the data and the topic strings change between messages then the existing unit of work must be committed and a new one started regardless of the batch size. In this case the actual MQ unit of work may be less than the specified batch size. However the wait will still be issued after the specified number of messages. Incomplete units of work are always committed before the wait is issued.

The warn if no match option will request notification if no subscriptions match a published message. A count of the number of messages that were published but did not match any subscription will be kept and reported after the Write Msgs operation is complete.

For example if the Batch Size parameter is set to 5 and the Wait Time is set to 100 (milliseconds) then 5 messages will be published every 100 milliseconds, for an approximate publication rate of 50 messages per second. It should be emphasized that this is not an exact number. If the batch size is large then the time to publish the messages must be taken into account when the Wait Time is specified. The Wait time parameter is aimed at preventing flooding a queue manager or network by publishing messages too fast rather than a precise method to publish messages at a specified rate.

The Publish Msgs button will start the publication operation. Once this button is selected then the publication operation will start. The publication operation uses a different thread for the duration of the publication operation. The Publish Msgs and Cancel buttons are disabled and the Stop Pub button is enabled for the duration of the publication operation. The Stop Pub button can be used to stop the publication operation at any point.

There are three ways that a publication operation will stop. It will also stop if an error is encountered during the operation. The first way to stop a publication operation is to use the Stop Pub button. This will notify the publishing thread to stop. The second method is to select a maximum number of messages to publish in the Max count field. For example if 100 is specified in the Max count field then the publication operation will stop automatically after 100 messages have been published. The final method is to select the Write msgs once check box. In this case the publication operation will stop automatically after each message in the specified file has been published once. This check box is handy when the number of messages in a file is not known and the messages in the file are to be published only once. If this check box is not selected then the publication operation will return to the beginning of the file and continue publishing again from the first message until the user stops the operation or the number of messages specified in the Max count field is reached.

The Cancel button will close the dialog box without starting a publication operation. It will return to the PS tab.

Close Sub button

When a subscription is no longer needed it should be closed by pressing the Close Sub button. If the subscription is a non-durable subscription it will be removed. If the non-durable subscription was also a managed subscription then the temporary dynamic queue that was allocated for the subscription will be deleted along with any messages still in the queue.

If the subscription is a durable subscription the subscription will be closed. It can then be resumed at a later point. If the Remove on Close option is selected however the subscription will be removed regardless of whether the subscription is durable or not and any temporary dynamic queue associated with the subscription will be deleted.

Get Topic Names button

The Get Topic Names button will send a PCF message to the broker queue manager requesting a list of topic names. Topic names are assigned when an administered node is created. The results are loaded into the topic name drop down list. This makes it easier to select a topic name when creating a subscription. If the broker queue manager does not respond to the PCF message then no names will be loaded into the drop down list.

The drop down list is optional. A valid topic name can be entered using the keyboard.

Get Sub Names button

The Get Sub Names button sends a PCF message to the broker queue manager requesting a list of durable subscriptions. The results are loading into the subscription name drop down list. This list is useful when resuming a (durable) subscription.

Wildcard Option

The wildcard option is useful for compatibility with earlier publish and subscribe implementations. The character wildcards (“*” and “?” plus the escape character “%”) were used by the publish/subscribe broker available in MQ V6 and earlier. Please see the MQ publications for further details.

Display type

The display type indicates whether the correlation id should be displayed in a hex or an ASCII format. The default is hex.

Subscription Options**Managed Sub**

The managed sub check box is used to indicate that a subscription is a managed subscription. In the case of a subscription no queue name is provided for the subscription. A temporary dynamic queue will be created for the subscription. This check box should be selected before a subscribe request is submitted. If an existing subscription is resumed using the Resume button the managed sub check box will be updated to reflect the subscription. If managed sub is selected for a new subscription then the name of the dynamic queue that was created will be returned in the queue name edit box for informational purposes.

Durable Sub

A durable subscription will remain in force even if the application that creates the subscription disconnects from the queue manager. Non-durable subscriptions are removed once the original application disconnects from the queue manager. Durable subscriptions can be resumed by the same or another application. In order to remove a durable subscription the Remove on Close option must be selected when closing the subscription.

Group Sub

The group sub option will set the corresponding option when a subscription request is sent to the broker queue manager. Grouping will consolidate multiple subscriptions that specify the same subscription level, queue name and correlation id. Only one message will be sent regardless of the number of subscriptions in the group.

Set Identity

The set identity option will use the specified application identity and accounting token for messages published by this subscription. This option applies to subscribe and alter requests. It is set when subscriptions are resumed.

Any UserId

The any user id option indicates that a durable subscription can be resumed by an application under a different user id that originally created the subscription. This option can be specified when the subscription is created or altered. If this field is being set using an alter then the userid must match the original userid.

Set CorrelId

The set correlid option will use the correlation id specified in the correl id field as the correlation id for this subscription. This allows the application to specify a particular

correlation id value when a subscription is created. This option cannot be used with managed subscriptions.

Local

The local option (scope queue manager) indicates that the subscription should be made only on the specified broker queue manager and that no proxy subscriptions should be registered with other queue managers in a publish/subscribe cluster.

On Demand

The on demand option indicates that the subscriber will not receive messages as they are published. Publications will only be sent to this subscription when a request publication is received (sent when the Req Pub button is pushed).

New Only

The new only option indicates that the current retained publication should not be sent when this subscription is created. This option is only used when creating a new subscription.

Remove on Close

The remove on close option is used when a subscription is closed. This option will remove a durable subscription when the subscription is closed. This allows durable subscriptions to be removed by a program. Durable subscriptions can also be removed by an administrator.

Publishing Options

Suppress Reply To

The suppress reply to option indicates that the reply to information should be removed from messages when they are published. This option is used when publishing messages using the publish button.

Retained Pub

The retained pub option is used when publishing messages. This option indicates that this message should be retained and should replace any previous retained publications for the topic. Each node in the topic tree can have at most one retained publication. Newer retained publications for the same topic node replace older retained publications for the same node.

Not own pubs

The not own pubs option informs the broker to not send any published messages to subscriptions using the same queue manager connection handle.

Local

The local option (scope queue manager) indicates that the publication should be published only on the specified broker queue manager and that it should not be sent to any proxy subscriptions registered with other queue managers in a publish/subscribe cluster. Note that the local option is used for both subscriptions and publications.

Warn No Subs

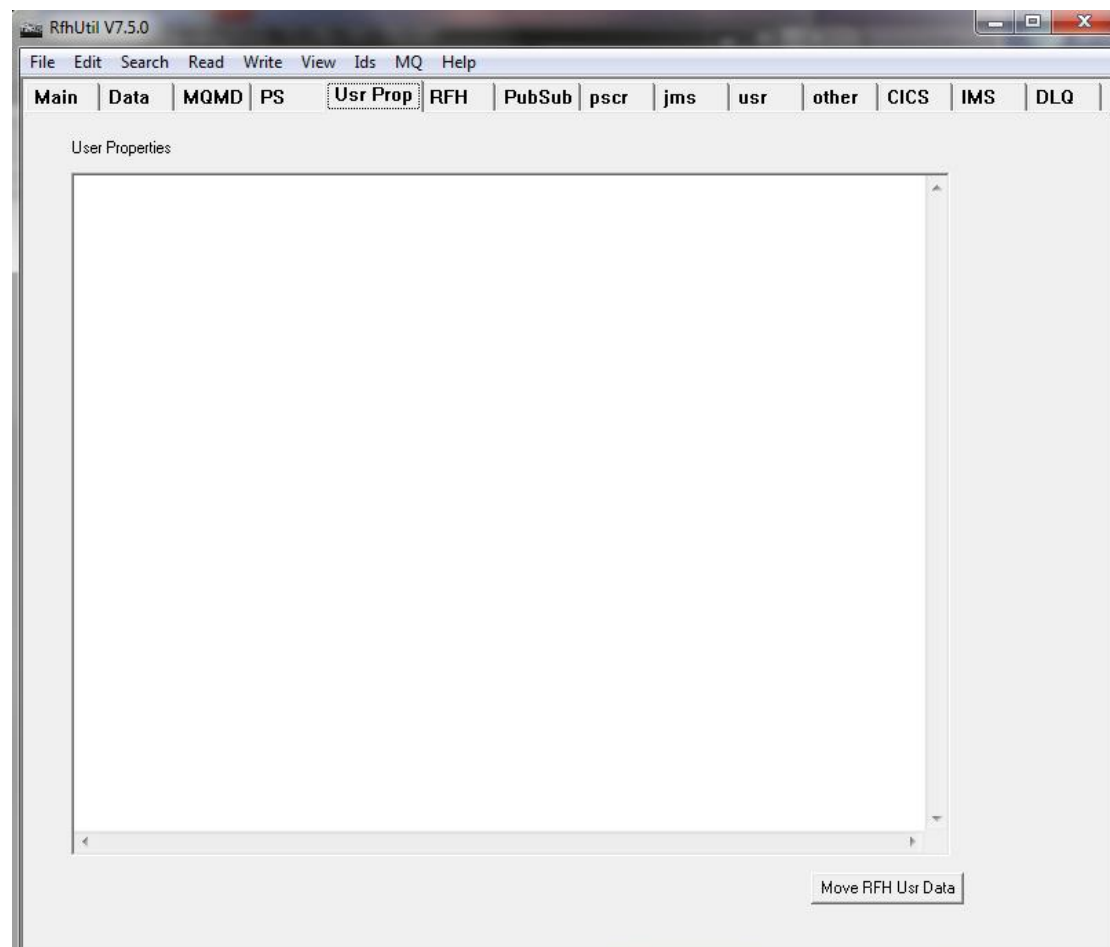
This option will issue a warning message if a message is published on a topic and there are no subscribers that match the published topic.

Is Retained flag

This flag is set after a message is retrieved using the Get Msg button. If the message contains an MQ property with a name of MQIsRetained with a value of 1 then the Is Retained box will be selected (checked). This flag cannot be modified by the user.

User properties tab

The user properties tab supports the user properties that were introduced with version 7 of WebSphere MQ.



MQ user properties are represented as name and value pairs. User properties use the same display format as the `usr` and `other` folder tabs. The name and value are separated by an equal sign. If the value is other than a string a dt attribute in parentheses will follow the name. A null value is indicated by `xsi:nil='true'` attribute rather than a dt attribute.

Although the user properties folder uses the same data format as the `usr` folder the properties in the user properties folders are more restricted than the values in the `usr` folder. User properties do not support XML groups (structures), repeating items or most attributes.

User properties can have a data type associated with the value. The following data types are recognized:

- | | |
|------------------|--|
| • i1 or byte | One byte (8-bit) integer |
| • i2 or short | Two-byte (16-bit) integer |
| • i4 or int | Four-byte (32-bit) integer |
| • i8 or i64 | 64-bit integer |
| • bin.hex | Bytes specified as pairs of hexadecimal characters |
| • boolean | Boolean value |
| • r4 or float | 32-bit floating point value |
| • r8 or double | 64-bit floating point value |
| • xsi:nil='true' | null value |

The following example illustrates the proper format for message properties.

```
User Properties
b(dt='i2')=32767
isBusinessException(dt='boolean')=TRUE
processName=3 4 5
myfield=
a(dt='i4')=12345678
c(dt='i1')=127
d(dt='r4')=2.09999990463
e(dt='r8')=2.1000000000000001
f(xsi:nil='true')=
```

Moving data from the RFH Usr tab to message properties

Values that appear on the RFH2 usr tab can be appended to the MQ user properties. Although the usr folder supports XML-like data there are representations that can appear in the usr folder that cannot be represented as MQ user properties. In particular data in the usr tab can contain attributes and a hierarchical structure. These are not allowed in MQ or JMS user properties and will not be copied. The only attributes that can be copied are the dt attribute and the xsi:nil attribute (which must have the value of “true”).

Data from the usr tab can be copied if it does not contain attributes other than data type (dt) or xsi:nil with a value of true and does not have groups (basically a structure as indicated by periods in the name). Otherwise the request will generate an error message. The message will be displayed on the main tab.

RFH tab

A Rules and Formatting header (RFH) is an optional data header that gives the message broker information about the format of the message data. The Rules and Formatting header (RFH) is contained at the front of the user data in a message.

There are two versions of the Rules and Formatting header (version 1 and version 2). Each header contains a fixed portion followed by an optional variable portion. The fixed portion identifies and describes the Rules and Formatting header itself. The variable part contains optional data fields. For a version one header, the fixed portion of the header is thirty-two bytes long, while the fixed portion of a version two header is thirty-six bytes long.

The variable portion of a version one header contains fields that can define the format of the user data (the application group and format name) and fields that are related to publish and subscribe requests.

The variable portion of a version two rules and formatting header (RFH) is divided into a number of XML folders. The mcd folder contains up to four fields that define the data type and format, namely the message domain, set, type and format. The domain indicates the parser that will process the message data. The parser determines the meaning of the other three fields. The variable portion can also contain folders related to publish and subscribe, JMS, user data and other miscellaneous user-defined folders. All folders are optional, and more than one folder may be present in the same RFH header. The desired folders can be selected using the check boxes on the RFH tab. These check boxes will be automatically set if a message or file is read that contains an RFH.

If a Rules and Formatting header is present, the format fields in the MQ message descriptor will be set to the literal “MQHRF ” or “MQHRF2 ”. This will be done automatically when the RFH type radio button is set to Version 1 or Version 2. The format field, encoding and code page fields of the Rules and Formatting header should be set to the format of the data that follows the RFH (which can be user data or another MQ header).

Multiple RFH headers in a single message

A single MQ message can contain both a version 1 and a version 2 Rules and Formatting header. The option to have both headers is selected with the Both radio button. In this case, the RFH version 1 header is normally the first header in the message, followed by the RFH version 2. The MQMD format field should identify the first RFH header, and the format field in the first RFH header should identify the second RFH header. The format field in the second RFH header should identify the type of data that follows the second RFH header. In a similar vein, the code page and encoding settings for the MQMD and RFH headers should follow a similar pattern (the setting in the MQMD should match the first header and the setting in each RFH should match the settings of the data that immediately follows the particular RFH).

RFH fixed data

The version, length and flags fields in the Rules and Formatting headers will be set automatically. The code page field and the RFH encoding radio buttons should be used to select the desired byte ordering of the binary fields in the data that immediately follows the Rules and Formatting header. The CCSID field is only used for a Rules and Formatting header that is in a version 2 format. This field denotes the code page used for the variable portion of the Rules and Formatting header. It is limited to one of four values, and should be set to code page 1208 in most cases.

RFH variable data

If the Version 1 check box is selected, then the application group and message format fields should be filled in to match the user data in the message. Similarly, if the Version 2 check box is selected, one or more of the four fields in the message format header should be selected. Data for other folders is contained on other tabs. Publish and subscribe fields are located on the *PubSub* and *pscr* tabs. The *jms*, *usr* and *other* tabs can only be used with a version two header.

Include RFH V2 Header and Include RFH V1 Header

These check boxes indicate whether a Rules and Formatting header should be included with the message or file data when it is written, and if so, what type of header should be included.

RFHUtil supports a single V2 and/or a single V1 header in a particular message.

When a message is read from either a queue or a file, the appropriate options are set automatically.

V2 Folders

The check boxes indicate which folders are to be included with an RFH2 header. They determine the contents of the variable portion of the RFH2 header. For example, if the mcd box is checked, a mcd folder will be included with the RFH2 header. The appropriate fields on the RFH tab should be filled in for the requested mcd folder.

Separate check boxes apply to an RFH version 1 header. A version 1 header can contain application group and format name fields, as well as fields associated with publish and subscribe requests and responses.

The same data can be included in both an RFH version 1 and an RFH version 2. For example, it is possible to include publish and subscribe information in both headers. While the RFHUtil program will allow this, it is very likely to cause problems with the receiving application and is considered an error. The mcd, publish and subscribe and pscr (publish and subscribe reply) information should not be selected in both headers for a single message.

A publish and subscribe tutorial is provided in an appendix. This tutorial contains examples of proper use of many of the publish and subscribe options.

Message Broker PubSub tab

The PubSub tab is used to generate publish and subscribe requests. This tab will allow most but not all publish/subscribe commands to be entered or displayed. Please be aware that different publish and subscribe brokers will not support all of the options that are available.

The RFHUtil program is intended to support three publish and subscribe environments, namely WebSphere Message Broker, WebSphere Event Broker and base MQ publish and subscribe. Some options are unique to a particular publish and subscribe environment. The base MQ publish and subscribe offering does not support RFH2 headers. Not all of the environments support all of the options and fields.

The register publisher (reg pub) and unregister publisher (unreg pub) request types are only supported by base MQ publish and subscribe and can only be used with a version 1 RFH header.

Request Type

The request type specifies the type of request. Once the request type is selected, then the appropriate input fields for that type of request will be enabled.

Topic(s)

There are four topic string fields, which allow a maximum of four topic strings to be entered, depending on the request type. Each topic is a separate complete topic.

Filter

Enter a filter string, if desired. The filter string is an ESQL expression that is used for content based filtering.

Sub Point/Stream

Enter the subscription point within the topic tree that this subscription should be entered at. Subscription points are sometimes used for versioning among other uses. If this request is being sent to a base MQ publish and subscribe environment, this field contains the stream name.

Sub Name, Identity and Data

These fields allow a subscription to be shared by multiple users. They are used by WebSphere Application Server.

Queue manager to connect to

Enter the name of the queue manager to connect to. If this field is left blank, then the default queue manager on the system will be used.

Queue Name

For publication requests, enter the name of the queue to send the publication to. This queue must have an active message flow that will process the message and create a publication, by the use of a publication node. For all other request types, the queue name is fixed at `SYSTEM.BROKER.CONTROL.QUEUE`.

Broker Queue Manager Name

Enter the name of the broker queue manager, if different from the queue manager that RFHUtil is connected to. A channel between the broker queue manager and the local queue manager that the utility is connecting to must have been defined, including an appropriate transmission queue.

Subscription Queue Manager

Enter the name of the queue manager to receive published messages.

Subscription Queue

Enter the name of the queue that will receive published messages.

Pub Time

This field is the publication time.

Seq No

Enter a sequence number, if desired.

Other Fields

This field contains any fields that are not recognized by the RFHUtil program. This field is only used with a version 2 RFH. Any unrecognized fields found in a version 1 RFH will be put in the Other Info field on the pscr tab. Unlike the other fields on this tab, this field will contain raw XML tags as well as the data value. This field must contain pairs of well-formed XML tags with the corresponding data values. The data values must use the XML escape sequences for certain special characters (such as & for the “&” character).

Options

A number of options can be selected. The actual options that are available will depend on the type of request. The options are available as check boxes, which can be selected and deselected. Options that are selected but do not apply to a particular type of request will be ignored.

The Local option applies to collectives. It indicates that the action is local to the specific broker and should not be propagated to other brokers in a collective.

If the New Only option is selected, then only new publications will be sent. Existing retained publications will not be sent, even if they match the selection criteria.

The Other Only applies to publishers that are also subscribers, and indicates that the publisher does not wish to receive its own publications.

The On Demand option is valid for registering of subscriptions, and is only effective for retained publications. It indicates that the application will issue a request publication when it is interested in receiving a publication, and that publications are not to be automatically sent to the subscribing application as they are published.

The Retain Pub option is used with publication requests and indicates that a particular message is a retained publication. The broker will maintain a copy of a retained publication until it is either replaced with a newer version or it is removed with a delete pub request.

The correlasid option uses the correlation id of the subscription message to identify a publication message. This option is primarily used when a publication queue is shared by multiple applications.

The Inf if Retain option will set an option in the RFH2 header to indicate that a published message is a retained publication. This option only applies to messages that are sent in response to subscribe or request publication requests.

The Dereg All option applies only to deregister requests, and will deregister all subscriptions for a particular publication queue.

The isRetained option is set by the broker to indicate that a particular message is a retained publication. This indication is set by the broker only in response to an Inf if Retain option being set on a subscription.

WebSphere Message Broker V6 does not support the following options.

A publish or subscribe request with the Full Resp option (full response) selected indicates that a full response is desired in reply to the request. All characteristics of the subscription will be returned with any reply message. The MQMD should specify either a message type of request or the PAN report option if this option is specified.

The Join Shared and Join Excl (exclusive) options allow a single subscription to be shared by multiple users. A subscription identity must be specified, which will be added to the identity set for this subscription. These two options are mutually exclusive. Setting one option will automatically turn off selection of the other option. WebSphere Application Server is the primary user of these options. The broker will return the Locked option if a previous subscription request has requested exclusive use of the subscription.

The Add Name, No Alter and Var User Id options are used with shared subscriptions, and are supported by WMQ Event Broker. If Add Name is specified, then a subscription name is required. The variable user id option allows a subscription to be modified or removed from a different user id than was used to create the subscription. The No Alter option is used when a new subscription is registered. If a subscription already exists, the attributes of the subscription will not be changed.

The Leave Only option applies to deregister (unsub) requests on shared subscriptions. It indicates that the subscriber identity should be deleted from the identity set for the subscription but that the subscription should be maintained even if there are no other users.

The anonymous option applies to base MQ publish and subscribe. If selected, information about the publisher will not be sent to subscribers.

The Direct Request and Dups OK options apply to base MQ publish and subscriber support. Direct Request indicates that a publication can be requested directly from the publisher. The Dups OK option is a performance optimization. If a subscriber has multiple subscriptions registered with a broker, then it is possible for a publication to match more than one subscription. The broker will normally consolidate the multiple matches and send only a single copy of the published message, regardless of the number of matching subscriptions. If the Dups Ok option is selected on a subscription request, this consolidation is not performed and a message is sent for each matching subscription.

The Incl Stream (include stream name) option indicates that the stream name is to be included in the header of any messages that are sent in response to a particular subscription. The base MQ publish and subscribe offering supports this option.

The base MQ publish and subscribe support allows publishers to register explicitly. This registration is normally optional, since a publisher will be registered implicitly when a publication is sent on a particular topic and the publisher has not explicitly registered. The No Reg option indicates that this implicit registration is not to be performed, and that the publication request will fail if the publisher has not previously registered as a publisher.

Persistence

The persistence option indicates whether published messages are sent as persistent or non-persistent messages. The default is persistence as the publication. Persistence can also be set to yes, no or as the subscriber's queue.

Clear button

The clear button will clear all data and selections on the pubsub tab.

Save to File button

The save to file button will save the current RFH header and any accompanying data to a file. This button is the same as the Save File button on the main tab. A message indicating the results of the file operation will be displayed in the message area at the bottom of the tab.

Write Msg button

The Write Msg button will send the current request to the broker, by performing an MQ put to the broker queue. A message indicating the results of the MQ put operation will be displayed in the message area at the bottom of the tab. This button is the same as the Write Q button on the main tab.

pscr tab

The pscr tab is used to display fields within a publish/subscribe response folder.

The screenshot shows the RFHUTIL V7.5.0 application window. The 'pscr' tab is selected in the tab bar. The main area contains the following fields:

- Completion:** Radio buttons for 'Ok', 'Warning', and 'Error'.
- Error Id:** Text input field.
- Error Pos:** Text input field.
- Parm Id:** Text input field.
- User id:** Text input field.
- Response 1:**
 - Reason Code: Text input field.
 - Other Response Info: Text area.
- Response 2:**
 - Reason Code: Text input field.
 - Other Response Info: Text area.
- Other Info:** Text area.

This tab will be populated with the contents of a pscr (publish/subscribe response) folder in an RFH 2 header that is found in a message or file that is read by the RFHUtil program. It is also used to display fields found in a version 1 header that are contained in broker responses. A maximum of two responses will be displayed.

The *Error Id*, *Error Pos*, *Parm Id* and *User Id* fields are only used with a version 1 RFH. The *Other Info* field may contain fields that are not recognized in a version 1 RFH.

If any entries are found in the variable section of an RFH version 1 header that are not recognized by the program, the information will be added to the *Other Info* field and the Resp check box will be selected on the RFH tab. On output operations, the contents of this field will be appended to any version 1 header that is generated if the Resp check box is selected on the RFH tab.

Responses are generally sent by the broker and not created by an application.

jms tab

The `jms` tab is used to display and set fields within a `jms` folder in an RFH version 2 header.

The `jms` tab is used to display or enter the fields found in a `jms` folder in an RFH 2 header.

There appears to be considerable confusion about the JMS standard, which actually defines only an application program interface and does not say anything about wire formats or protocols. The use of a `jms` folder and an RFH 2 header is unique to the IBM MQ implementation of JMS.

JMS Message Type

There are five types of messages supported by JMS. They are:

- text
- byte
- stream
- map
- object

The message type refers to the format and layout of the data. For example, the user data in a text message is in XML format. Byte data is an arbitrary stream of bytes and can contain any other type of data. Object data contains java objects, which can only be interpreted by other JMS java programs.

The “*none*” option can also be selected, in which case the Msd (domain) parameter in the mcd folder will be set to a value of “jms_none”.

Destination

The destination contains the name of the target JNDI object. This object is mapped to an MQ queue using the JMSAdmin command, and is equivalent to the destination queue and queue manager names.

Reply To

The replyto variable contains the name of a JNDI object that the receiving program can use to send a reply message. This object is mapped to an MQ queue using the JMSAdmin command, and is equivalent to the MQ reply to queue and queue manager names.

Correlation Id

The correlation id is an arbitrary string that can be used by an application to match a reply with a corresponding request.

Group Id and Sequence

Group id and sequence numbers allow several messages to be sent and processed as a group. This is similar to the MQ group capability.

Timestamp

This field contains the JMS timestamp field. This field contains the time when the message was created.

Priority

This variable indicates the JMS message priority. It is similar to the MQ message priority.

Expiration

The expiration variable indicates if a message should be discarded after a certain time interval, and if so, what that time interval is.

Delivery mode

The delivery mode in JMS indicates if a message is to be persistent or not.

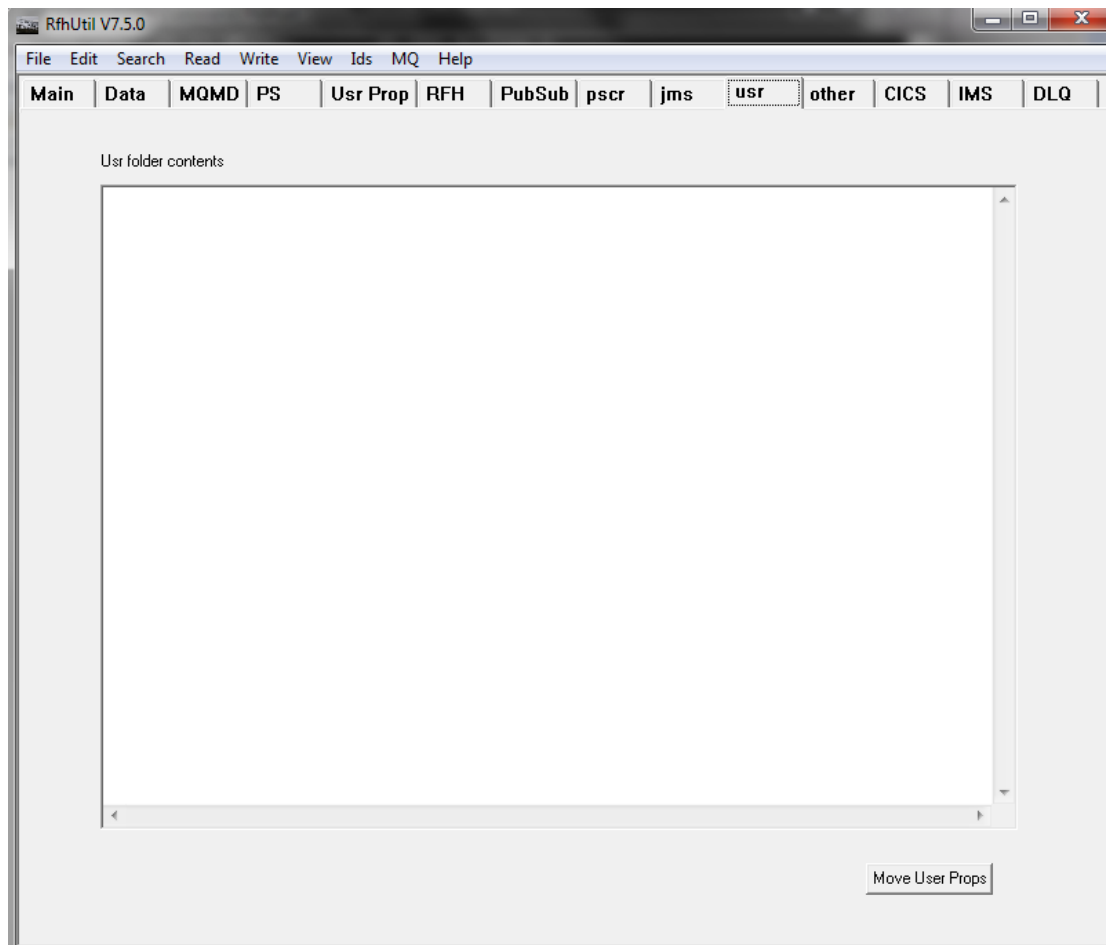
User Defined Fields

User defined fields are allowed in the jms folder. The *User Defined Fields* control should contain valid XML triplets (e.g. begin tag, data and matching end tag) that will be included in the jms folder. The contents of this field will be appended to the end of the XML stream that is generated from the defined fields and included in the jms folder. No checking of this data is performed on output.

usr tab

The usr tab is used to display and set the data in a usr folder in an RFH version 2 header. The actual data contained in RFH2 folders is described as “XML-like”. Many ordinary XML constructs are not allowed in an RFH2 folder cannot contain This would include things like

XML comments, CDATA sections, embedded DTDs and user defined entities. All user data is parsed character data.

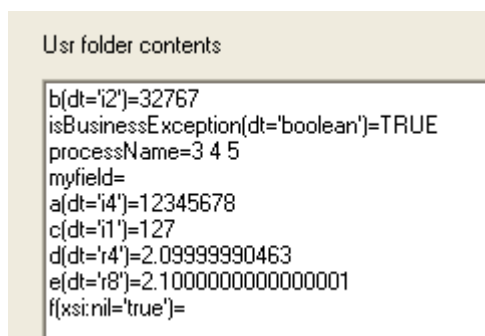


The data is displayed as name and value pairs, with an equal sign between the name and value. Each name and value is displayed on a separate line. The contents can be edited.

Attributes

Attributes are allowed in tags in the RFH2 usr area. Attributes follow the name and are enclosed in parentheses.

The following screen shot shows a sample message that contains values within a RFH2 usr folder. Note that the example shows dt attributes.



Repeating elements

Repeating elements can occur at any level of an XML document except for the root element. In some cases it is not possible to tell which element actually repeats. For example the following two lines of text could indicate that either the b or the c element repeat.

```
a.b.c=1
a.b.c=2
```

If the c element is the repeating element then this sequence should be rendered as follows:

```
<a><b><c>1</c><c>2</c></b></a>
```

However if the b element is the repeating element then the sequence would be rendered as:

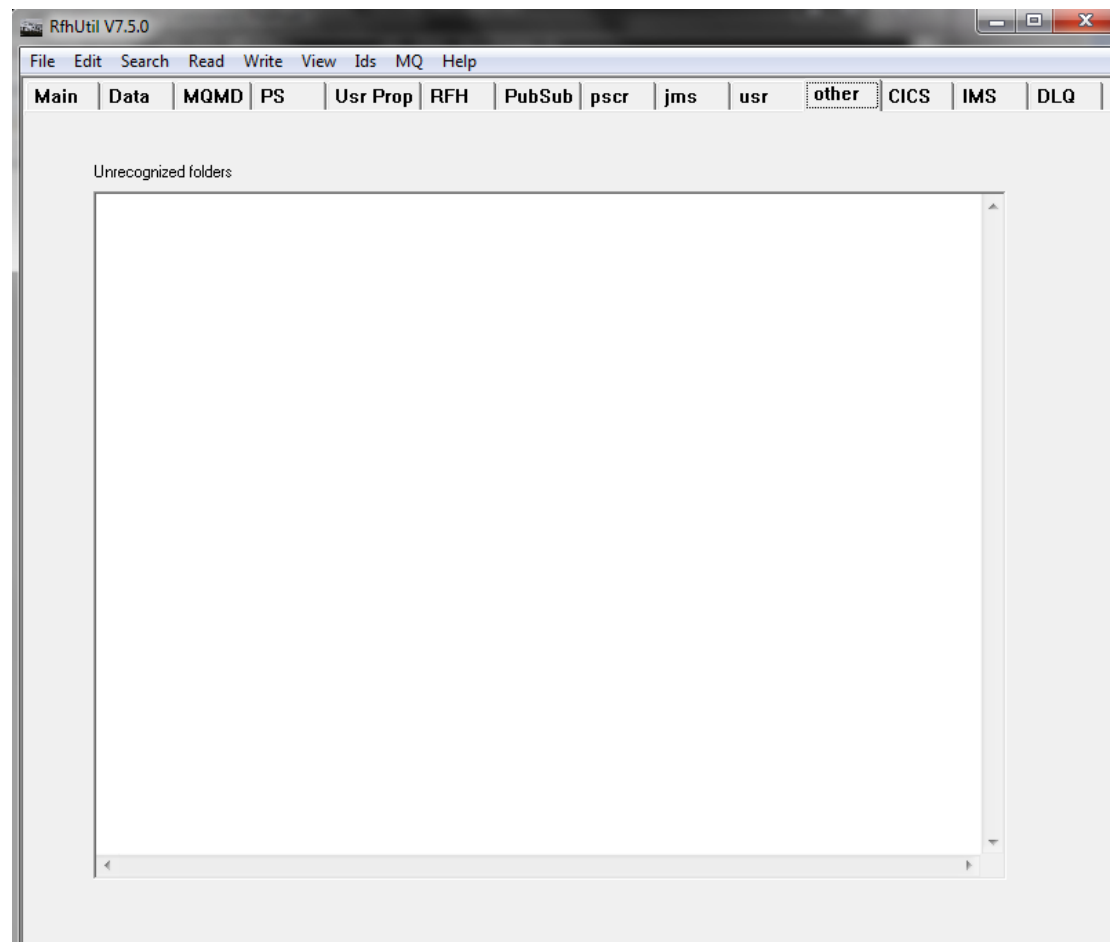
```
<a><b><c>1</c></b><b><c>2</c></b></a>
```

The situation can be resolved by using index numbers on the repeating elements. The index number should be enclosed in square brackets as shown below.

```
a.b[1].c=1
a.b[2].c=2
```

other tab

The other tab is used to display and set data in folders that are not recognized by the RFHUtil program.



The other tab uses a name and value pair format. The format is the same as the format used by the usr tab except that the highest level (root) tag name is included in the name part of the name and value pairs. See the discussion above for considerations involving repeating elements and how attributes are represented. RFHUtil does not support two successive folders with the same root tag name.

The contents of this tab are displayed in an edit box. The data can be edited and changed as desired as long as the result produces valid XML tag names and values.

The following screen shot shows three unrecognized folders. The first two folders do not contain any data.



CICS Tab

This tab displays the contents of the MQ CICS Information header (CIH), if present.

RFHUTIL V7.5.0

File Edit Search Read Write View Ids MQ Help

Main Data MQMD PS Usr Prop RFH PubSub pscr jms usr other **CICS** IMS DLQ

Version: ☒ None ☐ V1 ☐ V2

Program: Trans id: Next tranid: Remote sys: Remote tran:

UOW: ☒ Only ☐ Cont ☐ First ☐ Middle ☐ Last ☐ Commit ☐ Rollback ☐ Conversational

Start code: ☒ None ☐ Start ☐ Data ☐ Term

Link type: ☒ Prog ☐ Tran

Code Page:

End status: ☒ No Sync ☐ Commit ☐ Backout ☐ End task

Function:

ADS Descriptor: ☐ Send ☐ Receive ☐ Msg Format

Attn key: Cursor pos:

Return code: Comp code: Reason: Cancel code: Error ofs: Abend code:

Facility: Fac like: Format: Reply format:

Wait int: Data length: Flags: Keep time: Input item: Authenticator:

Encoding:

- Integer: ☒ PC ☐ Host
- Decimal: ☒ PC ☐ Host
- Float: ☐ PC ☐ Host ☐ 390 ☐ TNS

The CICS tab will allow a CICS Information Header (CIH) to be added to an MQ message. The CICS information header can be used with either the CICS 3270 bridge or the CICS DPL bridge.

In the case of the 3270 bridge, the user data is usually in the form of 3270 ADS vector structures. There is no support in the RFHUtil program to process the ADS vector structures. The data must be captured via some other means and probably stored in files.

The proper use of the fields on this tab requires an understanding of at least one of the CICS bridge programs.

Version

The version indicates the version level of the CICS header. If there is no CICS header, select none.

Program

For distributed program link (DPL) requests, a program name must be provided. The program name is actually not part of the CICS header structure, but immediately follows it in the data stream. Program name can be from one to eight characters long. It will be padded with spaces on the right when inserted into a message. Program name is required if the link type is Prog.

Trans id

Transaction id provides the name of the CICS transaction that this request should be use. This field is optional. If used, it must be from one to four characters and must match a valid CICS transaction.

Next transid

The Next transid field will be returned from CICS. It is from one to four characters in length and indicates which transaction will be executed when the next input data stream is received by CICS. This field is only used if the link type is Tran.

Remote sys

The remote sys field is used to indicate that this request should be routed to another CICS region for processing. The remote system field can be from 1 to 4 characters in length and must match a defined remote system in the CICS system that receives the initial request.

Remote tran

If the remote sys option is selected, then this field allows a different transaction id to be used. The remote transaction id must be from 1 to 4 characters in length and must be defined on the target CICS system specified in the remote sys field.

UOW

The UOW field indicates whether or not this message should be processed as a single unit of work.

Link type

Link type must be either program (for a DPL call) or tran, for a 3270 bridge call.

Start code

If a start code is desired, select the desired start code. If a transaction indicates start with data, then the CICS transaction should issue an EXEC CICS RETRIEVE to get the start data. The data must be supplied in the input data stream in a RETRIEVE vector structure.

Conversational

If the transaction is conversational, then select the check box. This field should only be used with the 3270 bridge.

End status

The end status is returned by CICS to indicate the results of the previous call.

Function

Enter the bridge function that is desired.

ADS Descriptor

This field consists of flags to indicate which types of ADS descriptors have been included in the data stream.

Attn key

Enter the code for the 3270 attention key that was used to initiate the current message data. If this field is left blank, then the default value of the enter key will be used.

Cursor pos

This field should contain the cursor position for the current 3270 bridge transaction.

Return code

This field contains the return code from the CICS bridge. It will indicate if the previous request completed successfully, and if not where the problem was detected.

Comp code

This field will contain either the MQ completion code or the CICS EIBRESP field. The contents of this field depend on the return code, which should be evaluated first.

Reason

This field can contain the MQ reason code or the CICS EIBRESP2 field, depending on the return code.

Cancel code

This field can be used to terminate a CICS transaction that has requested additional data in a conversational manner. If used, this field should be set to a four character abend code.

Error ofs

The 3270 bridge will use this field to indicate the offset within the user data where invalid data was encountered.

Abend code

This field will be set to the CICS abend code if the CICS transaction does not complete normally. The return code field should indicate a CICS transaction abend code if the CICS transaction has failed to complete normally.

Facility

The facility field is returned by CICS for 3270 bridge transactions where a keep time that is greater than zero is specified. This value should be returned on any subsequent requests that want to use the same CICS facility (terminal). This allows for proper handling of CICS facilities such as terminal user areas or other functions that depend on things like a terminal name.

Fac like

If specified, this field should be a four character terminal id that exists on the target CICS system.

Format

This field contains the format of the user data. If the user data contains only character data, then this field should be set to MQSTR. If the user data contains

Reply format

This field contains the format of the reply data returned from CICS.

Code Page

This field contains the code page of the user data.

Wait int

The length of time that the bridge will wait for a response from the CICS system is specified in this field.

Data length

This field contains the length of the user data that follows the bridge header, and is used by the DPL bridge.

Flags

This field contains flags used by the CICS bridge.

Keep time

The length of time that the 3270-bridge will remember a facility bridge token is specified in this field. This field is required to ensure that successive transactions are executed on the same simulated 3270 terminal. This is usually required for pseudo-conversational transactions. The facility identifier should be set to a null value for the first transaction in a pseudo-conversation and the same value must be returned on each subsequent transaction in the sequence.

Input item

The Input Item field contains the input item for a 3270-bridge request. This field is not intended for use by user programs and should not be set.

Authenticator

This field contains authentication information to be used by CICS to verify the identity of the user. For example, it might contain a password or pass ticket.

PD Enc

This field indicates how packed decimal fields in the user data are encoded. A value of PC indicates that the bytes in packed decimal fields are in “little-endian” order, whereas a value of Host indicates that bytes in packed decimal fields are in “big-endian” order (e.g. as on a 390).

Since the CICS bridges do not perform data translation of binary or packed decimal data, the value of the encoding field does not usually matter. It should be set to a valid non-zero value.

Int Enc

This entry indicates how binary and floating point fields in the user data are encoded. A value of PC indicates that the bytes in binary and floating-point fields are in “little-endian” order, as on an Intel processor. A value of Host or Unix indicates binary and floating-point fields are in “big-endian” order. Floating point fields will be in 390-format if a value of Host is selected, and in IEEE format if a value of Unix or PC is chosen.

IMS header Tab

This tab displays the contents of the MQ IMS information header, if present. If an IMS header is present, then the data must begin with an LLBB (length) field for the first message segment. The length fields are validated by MQ.

The screenshot shows the RfhUtil V7.5.0 application window with the 'IMS' tab selected in the top menu. The window contains several configuration fields and groups:

- Include IMS header:** A checkbox that is currently unchecked.
- Map Name:** A text input field.
- LTerm Override:** A text input field.
- Authenticator:** A text input field.
- Format:** A text input field.
- Code Page:** A text input field.
- Trans Instance ID:** A text input field.
- Reply Format:** A text input field.
- Trans State:** A group box containing three radio buttons: 'No Conversation' (selected), 'In Conversation', and 'Architected'.
- Commit Mode:** A group box containing two radio buttons: 'Commit then send' (selected) and 'Send then commit'.
- Security Scope:** A group box containing two radio buttons: 'Check' (selected) and 'Full'.
- Flags:** A group box containing two checkboxes: 'Pass Expiration' and 'No reply format', both of which are unchecked.
- Insert length field:** A checkbox that is currently unchecked.
- Encoding:** A group box containing three sub-groups:
 - Int Enc:** Two radio buttons, 'PC' (selected) and 'Host'.
 - PD Enc:** Two radio buttons, 'PC' (selected) and 'Host'.
 - Float:** Four radio buttons, 'PC' (selected), 'Host', '390', and 'TNS'.

Include IMS header

This check box indicates whether an IMS header should be included with the message if it is written, or if an IMS header was found as part of an input message. There are menu items to control whether a header is included when a message is written to a file, and whether to look for a header when the message is read from a file.

Map Name

This field contains the data MFS map name. It can be a maximum of 8 characters.

LTerm Override

This field contains the IMS logical terminal name to be used for this transaction. It can be a maximum of 8 characters.

Authenticator

This field contains a security identifier to be used by IMS to authenticate the transaction. It can be a maximum of 8 characters.

Format

This field contains the MQ format identifier for the data that follows the header. It can be a maximum of 8 characters.

Code page

This field contains the code page of any data that follows the header. It can be a maximum of 5 characters.

PD Encoding and Int Encoding

These fields are used to set the encoding values for any packed decimal or integer data contained in the user data area.

Trans Instance ID

This field contains the conversation identifier / server token from IMS for a conversational transaction. This field must be returned with the next message in a transaction. For non-conversational transactions, this field contains binary zeros. This field is displayed as hexadecimal characters and will be padded with binary zeros if necessary. This field should contain an even number of characters. It can be a maximum of 32 hexadecimal characters, which will be converted to a 16 byte value.

Reply Format

This field contains the MQ format identifier for the reply data. It can be a maximum of 8 characters.

Trans State

The transaction state can be entered using the radio buttons provided. The default is No Conversation, indicating that there is no IMS conversation in progress. If an IMS conversation is in progress, then the In Conversation radio button should be selected. The last option is Architected.

Commit Mode

The MQ commit mode can be either before or after the reply message is sent. The default is to commit the transaction before sending a reply.

Security Scope

The security scope indicates the type of security checking that is to be performed by the IMS bridge.

Flags

The Pass Expiration and No reply format check boxes allow the corresponding flags to be set in the IMS header.

Insert length field

If this check box is selected, then a 4 byte binary length field will be added to the end of the IMS header. The length field will be based on the length of the user data. This field only applies when data being written to a queue or file. It should not be used if the data already contains a length field and does not support multiple data segments.

Dead Letter Header (DLQ) Tab

This tab displays the contents of the MQ dead letter header, if present.

The screenshot shows the RFhUtil V7.5.0 application window with the 'DLQ' tab selected. The window has a menu bar (File, Edit, Search, Read, Write, View, Ids, MQ, Help) and a tab bar with the following tabs: Main, Data, MQMD, PS, Ustr Prop, RFH, PubSub, pscr, jms, usr, other, CICS, IMS, and DLQ. The DLQ tab contains the following fields and controls:

- ☐ Include DLQ header
- Reason:
- Orig Dest Queue Manager:
- Original Dest Queue:
- Format:
- Code Page:
- Put Appl Name:
- Put Appl Type:
- Date/Time on DLQ:
- Encoding section:
 - Integer: ☐ PC ☐ Host
 - Decimal: ☐ PC ☐ Host
 - Float: ☐ PC ☐ Host ☐ 390 ☐ TNS
-

Include DLQ header

This check box indicates whether or not the dead letter queue header should be included with the message if the message is written or if a dead letter queue header was found when a message was read. If this check box is not selected, then the other controls on this dialog will be disabled.

Reason

This field contains the reason code indicating why a message was placed on the dead letter queue.

Orig Dest Queue Manager and Orig Dest Queue

These fields contain the original queue and queue manager destination. Each field can contain a maximum of 48 characters

Format, Code Page and Encoding fields

These fields contain the original message format, character set id and encoding information from the MQMD. The corresponding fields in the MQMD now refer to the dead letter queue header.

Put Appl Name and Put Appl type

These fields contain the original put application name and type from the MQMD. The corresponding fields in the MQMD now refer to the application that placed the message on the dead letter queue.

Date/Time on DLQ

This field contains the date and time that the message was placed on the dead letter queue.

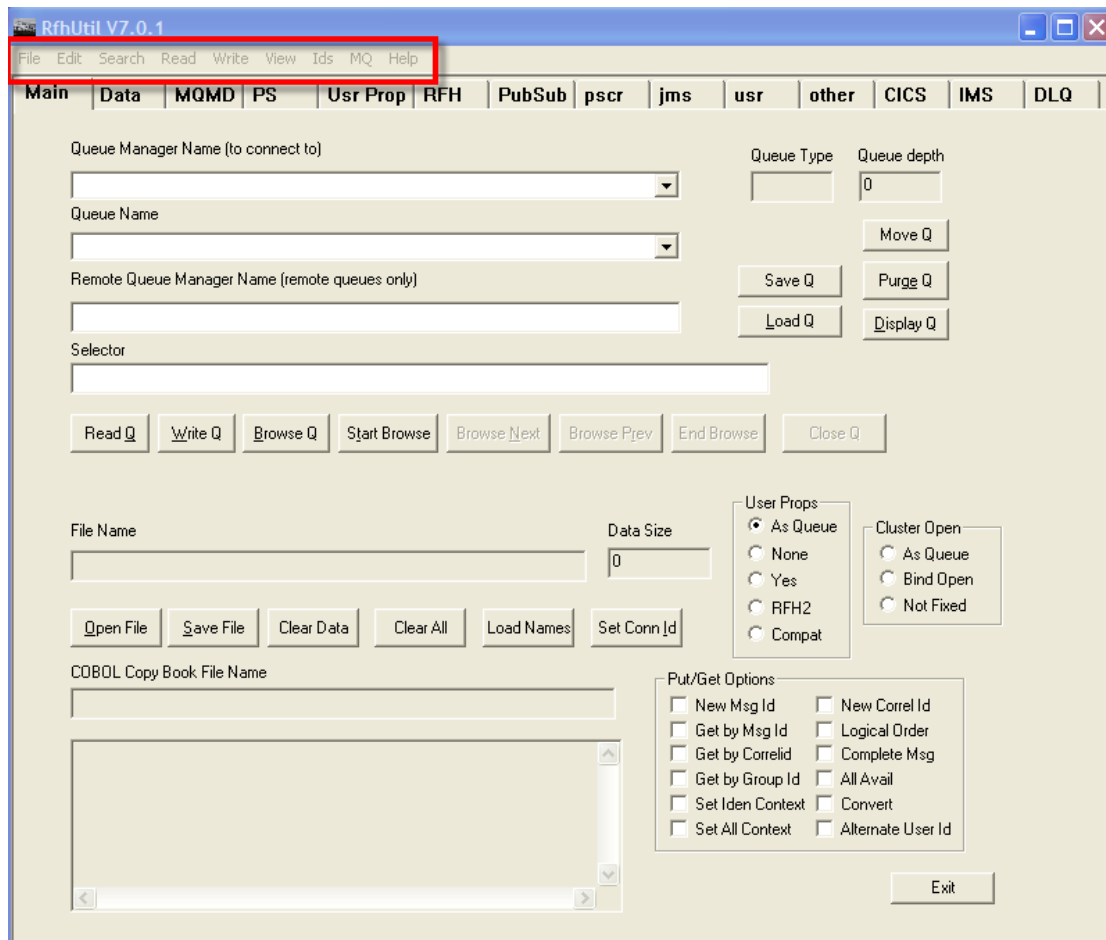
Prepare for resend

This button will copy the appropriate fields in the dead letter header into the corresponding fields in the MQMD. The destination queue and queue manager fields will be set on the Main dialog. If the destination queue manager field does not match the Queue Manager name, then the destination queue manager field will be copied to the Remote queue manager name.

If the put application name and type are to be preserved, then the message should be written with either Set All Context or Set User ID selected.

Menu options

A number of menus are available within the RFHUtil program, as shown below.



The functions available within each menu are discussed below.

File Menu

There are four functions available on the File menu. The open option will read the contents of a file into a memory buffer. It is equivalent to pressing the Read File button on the main tab. Similarly, the write option will write the contents of the memory buffer to a file. It is equivalent to pressing the Save File button on the main tab. The print option will send the current data in either the data, usr or other tab to a printer. The data will be in the same format as the current display. If more control over printing of the data is desired, then the data can be selected, copied and pasted into a word processor. Finally, the exit option will end the program and is equivalent to pressing the Exit button on the main tab.

A recent file list is also available on the file menu. The recent file list provides a quick way to read a file that has been previously read. The recent file list will hold a maximum of six file names. A least recently used algorithm determines the contents of the recent file list.

Edit Menu

The standard Windows edit functions of cut, copy, paste and select all are available on this menu. Please be aware that not all functions are supported for all fields, as many fields are read-only. In particular, user data must be read from either a file or a queue, and cannot be pasted into the data window.

The top and bottom options are active when the data tab is selected. The options will move the current display to the top and bottom of the message data respectively.

Search Menu

The search menu provides two find options and a go to option. This menu is only active when the data tab is selected. The find options will locate the next occurrence of a item and will scroll the display to show the selected line. The go to item allows the user to go to a specific offset within the data.

Read Menu

The read menu offers options that affect how file read (open) operations work. Each option can be selected and unselected using the menu. A check mark will be displayed next to any selected option(s).

The Remove CrLf option will remove all carriage return and/or line feed characters from the data when data is read from a file. This option is useful when a data file has unwanted carriage return and/or line feed characters.

The Unix Text option will replace single line feed characters with a carriage return and line feed combination. This option is useful for viewing Unix or Java text files with utilities such as notepad. The updated data can then be saved in a file. This option will change a Unix text file to a PC text file.

The Ignore Headers option will suppress the check for MQ headers when data is read from a file. RFHUtil supports the saving of MQ headers with the data in a file. When a file is read, RFHUtil will look for MQ headers at the beginning of a file. If MQ headers are found, the headers will be removed from the data and parsed. The format, character set id and encoding fields in the MQMD will be set to match the characteristics of the first MQ header.

The Save Headers menu option will leave existing MQ headers intact when new data is read from a file. Normally, all fields in the MQMD and MQ headers are reset when new data is read from a file. When this option is used, the file should not contain MQ headers since they will not be parsed.

The ignore MQMD option will suppress the normal check for an MQMD at the beginning of the data when a file is read. This option can be used if the user data were to include an MQMD at the beginning of the data, which is unlikely.

Write Menu

This menu provides additional options when writing data to a file.

The No Headers menu option will write only the user data to a file, ignoring any MQ headers. If this option is not selected and an MQ header is present, then both the user data and the MQ headers (if any) will be written to the file. This option does not affect whether an MQMD is saved with the data or not.

By default the MQMD and any MQ headers are written to the file along with the user data when a file write (save) operation is performed. If the check mark is removed from the Include MQMD menu option then no MQMD will be written.

View Menu

This menu provides one option that indicates whether system queues (queues whose name begins with SYSTEM) are displayed in the dropdown queue list. The default is to not display system queues. In either case, a system queue can be accessed by typing the full queue name.

A second option controls whether or not cluster queues that the queue manager is aware of are to be displayed. This option is off by default. Cluster queues can always be accessed by typing the name of the cluster queue rather than using the drop down list.

Ids Menu

The ids menu provides several useful options for working with message, correlation and/or group ids. All three ids are twenty-four byte binary fields. In some cases, it may be desirable to save a copy of an id and reuse it later. A particular id can be saved in an internal data area by selecting the appropriate save option. It can later be recalled by using the appropriate restore option. The restore option will copy the previously saved data into the corresponding id field in the MQMD.

In addition to the save and restore options, an option is provided to copy the message id to the correlation id.

MQ Menu

Connection and open operations are performed automatically when a queue operation is performed (such as pressing the Read Q or Write Q buttons). The queue is normally left open after an operation is performed and is not closed until a queue operation is performed to a different queue. The Close Q button allows the queue to be closed and the connection to the queue manager to be terminated.

The MQ menu provides the ability to connect and disconnect from a queue manager without performing any queue operations. It also provides the ability to open a queue for get, put or browse operations. An open queue can also be closed. This menu can be useful if the individual connection and open operations should be performed separately.

Accelerator Keys

Moving between tabs using the keyboard

You can move to the next tab by using the Alt + N key combination and the previous tab by using the Alt + P key combination.

The following accelerator keys are available. The key combinations can be used in lieu of the corresponding button.

Main Tab

Select All	Ctrl + A
Browse Q	Ctrl + B
Copy	Ctrl + C
Display Q	Ctrl + D
Purge Q	Ctrl + E
Load Names	Ctrl + H
Set User Id	Ctrl + I
Clear All	Ctrl + J
Close Q	Ctrl + K
Load Q	Ctrl + L
End Browse	Ctrl + M
Browse Next	Ctrl + N
Open File	Ctrl + O
Print	Ctrl + P
Read Q	Ctrl + Q
Browse Previous	Ctrl + R
Save File	Ctrl + S
Start Browse	Ctrl + T

Paste	Ctrl + V
Write Q	Ctrl + W
Cut	Ctrl + X
Save Q	Ctrl + Y
Clear Data Only	Alt + X
Clear All Data	Alt + Y

Data Tab

Select All	Ctrl + A
Display Both	Ctrl + B
Copy	Ctrl + C
Toggle PD setting	Ctrl + D
Find character string	Ctrl + F
Goto offset	Ctrl + G
Hex display format	Ctrl + H
Toggle Int setting	Ctrl + I
XML format	Ctrl + L
End browse	Ctrl + M
Browse Next	Ctrl + N
Open file	Ctrl + O
Parsed format	Ctrl + Q
Browse Previous	Ctrl + R
Save file	Ctrl + S
Character format	Ctrl + T
Hex search	Ctrl + Z
ASCII characters	Alt + A
EBCDIC characters	Alt + E
Fix format	Alt + F
Thai	Alt + H
Japanese	Alt + J
Korean	Alt + K
JSON	Alt + O
Simplified Chinese	Alt + S
Traditional Chinese	Alt + T
Clear data	Alt + X
Read Copybook	Alt + Y

MQMD Tab

Select All	Ctrl + A
Copy	Ctrl + C
Paste	Ctrl + V
Cut	Ctrl + X
Copy Msg to Correl ID	Alt + C
Reset IDs	Alt + J

PS Tab

Select All	Ctrl + A
Copy	Ctrl + C
Paste	Ctrl + V
Cut	Ctrl + X
Alter Sub	Alt + A
Close Sub	Alt + C
Req Pub	Alt + E
Get Msg	Alt + G
Clear All	Alt + L
Resume	Alt + R

Subscribe	Alt + S
Get Topic Names	Alt + T
Get Sub Names	Alt + U
Publish	Alt + W

Pubsub Tab

Select All	Ctrl + A
Copy	Ctrl + C
Clear	Ctrl + J
Save File	Ctrl + S
Paste	Ctrl + V
Write Msg	Ctrl + W
Cut	Ctrl + X

Load Q Dialog

Ascii delimiter	Ctrl + A
Hex delimiter	Ctrl + H
One File	Ctrl + F
File per Msg	Ctrl + M
Ignore MQMD	Ctrl + G
Remove Headers	Ctrl + R
Set all context	Ctrl + T
New Message Id	Ctrl + I
Browse	Ctrl + O

Save Q Dialog

Ascii delimiter	Ctrl + A
Hex delimiter	Ctrl + H
One File	Ctrl + F
File per Msg	Ctrl + M
Ignore MQMD	Ctrl + G
Remove Headers	Ctrl + R
Remove from queue	Ctrl + I
Don't remove	Ctrl + T
Browse	Ctrl + O

Set Connection Id Dialog

Reset	Ctrl + E
SSL	Ctrl + S
SSL Client Validation	Ctrl + L
Write Msg	Ctrl + W
Browse	Ctrl + O

Write Pubs Dialog

Ascii delimiter	Alt + A
New Correl Id	Alt + C
Hex delimiter	Alt + H
New Message Id	Alt + M
Write Messages Once	Alt + O
Use MQMD	Alt + Q
Use Properties	Alt + R
Use Topic	Alt + T
Exit (stop)	Alt + X

Capture Pubs Dialog

Ascii delimiter	Alt + A
Append to file	Alt + D
Hex delimiter	Alt + H
Include Headers	Alt + I
Capture Messages	Alt + M
Include MQMD	Alt + Q
Include Properties	Alt + R
Include topics	Alt + T
Exit (stop)	Alt + X

In most cases if the accelerator key is not visible on the button or control the key combination is shown in the tool tip help.

Environment variables

Environment variables can be used to enable auditing and trace. The auditing function will record all reading and writing activities in a text file. Each individual action is recorded in one line in the file. To enable the auditing function set the RFHUTIL_AUDIT_FILE environment variable to point to a valid file. If the file does not exist it will be created. Otherwise new lines will be appended to the end of the file. Please note that lines will continue to be added to the file as long as the variable is set. The file may need to be periodically deleted or archived so that it does not grow continuously.

The trace is primarily for problem resolution. It should only be used when reporting a problem to the author. Instructions to turn on the trace are in the problem determination section of this document (chapter 6). The trace function provides a lot of detail and should not be turned on for normal usage.

Chapter 4. Performance measurement utilities

Overview

Several command line performance measurement utilities are provided with this repository. The utilities include a driver program that will read test data from one or more files and write messages to a specified queue. This will provide a measurement workload for one or more message flows. Another program analyses the results of the message flow and calculates the number of messages processed per second. Two additional versions of the driver program are provided. They are minor variations on the MQPUT2.exe program, and are necessary when remote queues or client connections must be used. The additional driver programs are called MQPUTS.exe and MQPUTSC.exe respectively. All of the driver programs use the same parameters file and message data files for their input. The MQTEST.exe program is similar to the MQPut2 program and uses the same parameters file format. However, this utility is designed for testing environments rather than performance measurements. Each individual message in the parameters file is written once. The messages are not stored in memory.

In addition to the performance measurement utilities, capture utilities are provided that read messages either in queues or as published messages and stores the messages in files. The first utility will read messages in a queue and write them to a single file, placing a delimiter sequence between individual messages. The delimiter sequence is specified in the parameters file. The second utility will read a single message from a queue and will write the data to a file.

Using a client connection

The programs included in the repository are all linked with the mqm library which can be used for both local and client connectivity to a queue manager. If only the client components are installed on your machine, then client connectivity will be used automatically. If you have installed the server components, then the programs will attempt, by default, to connect to a local queue manager.

To force client connections to a remote queue manager, set the MQ_CONNECT_TYPE environment variable to "CLIENT". And then the usual client connectivity options (CCDT, MQSERVER environment variable) can be used.

For example

```
C:\ > set MQ_CONNECT_TYPE=CLIENT
C:\ > set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/mymachine.example.com(1414)
```

Please be sure to read the entire section entitled Requirements for performance measurement that immediately follows this section. This section details a requirement of the message flow itself that is absolutely necessary for the utilities to produce meaningful performance data.

Differences between measurement programs

Two performance measurement programs are provided, namely MQTimes2 and MQTimes3. Although they are very similar in many aspects and should generally give comparable results, it is worthwhile understanding the difference. In the case of MQTimes2, the timestamp that is put in the MQMD is used to report when the message was written. This technique can cause problems if messages are processed by a message flow and the original MQMD is passed through. The timestamp will reflect when the message was created by the original program rather than when it was processed by a message flow. This can also cause a problem when

a limited number of messages are used and the same messages are replayed repeatedly. The next section will provide a solution for this problem.

The MQTimes3 measurement program reads messages and reports the number of messages received within each second. The program watches the system clock and reports every time a new second is observed. This technique does not rely on timestamps in the MQMD. Rather it is measuring when a message is consumed by the program rather than when it was created. This technique will report accurate measurements as long as the measurement program is able to keep up with the producing program. If a single instance of the program is unable to keep up then multiple instances can be used and the results added together to produce the final result.

Requirements for performance measurement

There are a few requirements that must be met to measure the performance of one or more message flows using the MQTimes2 utility provided with this Repository.

First, the user data from one or more test messages must be captured as one or more files. The test messages must be able to be replayed repeatedly and generate the desired results each time. The message test and display utility that is included with this Repository can be used to read such messages from a queue and store them as local files. These sorts of messages are usually required for testing and debugging of the message flows prior to any attempt to measure performance.

The other key requirement is that each execution of the message flow produces at least one output message that is written to a queue. The throughput will be measured as the number of output messages per second and will be determined by the number of messages read from a particular queue. In some cases, a message flow may have to be modified to ensure that this requirement is met. Fortunately, most message flows meet this requirement.

One change is normally required to the message flow itself. This change is necessary to ensure that each message written to a queue that will be used to determine the throughput contains a new timestamp that indicates when the message was written. The message context field in the advanced properties of the MQOutput node for the queue must be changed to a value of “default” rather than the default value of “Pass All”. **This change is absolutely necessary. If the value is left as “Pass All” or some other setting, then the time the message is written will be passed from the input message, and the performance data will reflect the performance of the driver program rather than the performance of the message flow itself.** This change should be removed after the performance testing has been completed.

The change of the MQOutput parameter should not have a measurable impact to the performance of the message flow itself.

The author would like to mention one other important assumption that is usually made with regards to performance measurements. Performance measurements are usually measured in an optimized and controlled environment. In all cases, adequate real storage is assumed. For a full IBM Integration Bus V9 environment, this usually means 2 GB or more of main storage. Paging must be kept to a minimum. This is true for both the message flow environment itself as well as the performance driver utilities. If significant paging occurs during the measurement period, or if other unpredictable workloads are run on the system during the measurement period, the numbers are likely to be non-reproducible. Such results are usually regarded as useless. In most cases, the performance of IBM Integration Bus V9 message flows is CPU bound, while MQ itself may be either CPU bound (especially if non-persistent messages are used), or disk bound (if persistent messages are used). Like all assumptions, these assumptions should be validated.

In a Windows environment it is recommended that the task manager be invoked with the performance tab selected. This will display the current CPU and memory usage.

Differences between the driver utilities

Two driver utilities are provided with this Repository. The MQPut2 utility will watch the depth of the input queue and attempt to keep it within specified bounds. The program sleeps for a specified number of milliseconds and checks the queue depth. If the queue depth is less than a specified minimum then enough messages are written to bring the queue back up to a specified maximum depth. This technique will drive applications at their maximum speed. If the queue depth reaches zero then a warning message is issued. If these warning messages are observed then the results should be discarded, since the driver program is not keeping up with the application. In this case the maximum and/or minimum queue depths should be increased and the measurement repeated. A batchsize parameter specifies the maximum number of messages to write in a single unit of work. Increasing this parameter will usually allow higher message rates to be achieved.

The MQPuts utility will write messages at a specified rate. It will write the number of messages specified as the batchSize and will then sleep for the number of milliseconds specified by the thinktime parameter. For example if the batchSize parameter is set to 2 and the thinktime parameter is set to 10 (milliseconds) then the message rate should be approximately 200 per second.

Using the Performance Utilities

To use the performance driver utility, a parameters file must be created. Sample parameters files are included with this Repository (e.g. parmtst1.txt). This file contains many comments as well as examples of the various parameters. The parameters file can be edited with notepad or some other text editor. The parameters file contains the name of the queue that the input messages are to be written to. This should be the name of a queue that is referenced in an MQInput node within the message flow that is to be tested. It also must contain the name of one or more message data files, and the total number of messages that are to be written to the queue by the driver program. It is recommended to try a small number of messages first, to make sure that everything is working as expected.

If error messages are produced indicating that the queue depth has reached zero then the measurement should be run again after lowering the sleep time parameter and/or increasing the minimum and maximum queue depths. When the queue depth reaches zero it means that the driver program is not writing messages fast enough to keep up with the message flow.

Driving the Message Flow

Once the parameters and data files are available, and the message flow is deployed and ready to execute, the driver utility program (MQPUT2.exe) should be invoked from a command line prompt. The command syntax is as follows:

```
MQPUT2 -f parameter_file_name
```

The program has one required parameter. The parameter is the name of the input parameters file. This file name must be fully qualified if the parameters file is not in the current directory. The MQPUT2 program will produce some command line output, which should be reviewed for error messages and to verify that the parameters used are in fact the ones that were intended. The queue and queue manager names in the parameters file can be overridden on the command line by using the -q and -m command line arguments.

The syntax for the MQTEST utility is the same as the MQPUT2 utility.

The MQPut2 utility is designed to be used with local queues. It monitors the depth of the target queue and writes messages as needed to keep the queue depth within defined limits. The MQPuts utility does not attempt to monitor the depth of the target queue. It can therefore flood the queue with messages, which can distort the actual measurement itself. A think time parameter (and batch size) can be used to control the message rate. It is usually better to move the driver program to run on the same system as the workload.

Measuring Performance

The results should be analyzed with the performance analysis utility program (MQTIMES2.EXE or MQTIMES3.EXE). These programs should be started before the driver program. The MQTIMES.EXE program has been deprecated.

The MQTimes2 program is the preferred performance analysis tool. It is designed to run while the driver program is operating. It reads messages from the output queue in real time. By consuming the messages as soon as they are available, overflows of the queue buffer to disk are minimized, and the size of periodic queue checkpoints, where the queue buffer is flushed to disk, are reduced significantly. MQTimes2 accepts some additional parameters. The format of the MQTimes2 command is as follows:

```
MQTimes2 <-c messageCount> <-q queue> <-m queue manager> <-b nnn>
<-f parameters file>
```

The message count (-c) is the number of messages to read before automatically ending. The time out (-t) parameter is the maximum number of seconds to wait for a message to arrive before ending the program. The program will end when either it has read the expected number of messages or has waited more than the timeout interval for a message to arrive. The timeout interval should be at least 30 seconds.

The queue and queue manager names indicate which queue messages are to be read from. The queue name is required. The queue manager name is not required if the queue is defined on the default queue manager. If the drain (-d) parameter is specified, then any messages on the queue when the MQTimes2 program is started will be read and discarded. Any drained messages will not be included in the performance measurements.

The NAN and PAN reply data files are used when the consuming application must send report messages in response to a request for either positive or negative acknowledgments. Certain applications, such as JMS publishing applications, require responses from the consuming application to continue running. If a positive or negative acknowledgment is required, then a file containing the data portion of the response must be captured or created. This file must then be referenced within the parameters file. If a message requests both positive and negative acknowledgments, a positive acknowledgment will be sent.

The MQTIMES utility has been deprecated. It can be executed from a command prompt as follows:

```
MQTIMES -q queue_name <-m queue_manager_name> <-b nnn> >testout.txt
```

The performance analysis utility accepts one to three parameters. The first parameter is the name of the output queue that contains the output messages of the test message flow. The second parameter is the name of the queue manager that this queue is defined in. If the queue is defined on the default queue manager on the local system, then the second parameter can be omitted. The final parameter is the number of messages read within a single unit of work (batchsize). It helps the MQTimes program to run more quickly. The default of 25 is normally sufficient. The output will be written to the command line (stdout), and should be redirected to a file, so that the results can be saved. The results file can then be viewed with a text editor, such as notepad.

The MQTIMES utility will read all messages that are on the specified queue and will ignore all data in the messages except the timestamp in the MQ message descriptor (MQMD). This program can also be used to drain queues of unwanted messages.

The MQTIMES utility will report the number of messages that were processed within each second on a separate output line. It will also report totals as well as the average number of messages per second for all intervals except the first and last. The first and last intervals are omitted since it is likely that neither one represents a full second.

Measuring performance of JMS publishing applications

Recent versions of the IBM JMS publishing client contains a form of pacing support. The idea is to prevent a publishing application from publishing messages faster than the broker can

deal with them, and subsequently flooding a broker with messages. The implementation of this function involves setting of positive and negative acknowledgment report options periodically.

The actual implementation of this function is as follows. On the first message published, the PAN and NAN report options are set. Additional messages are then sent until the number of published messages reaches 20. On the first MQ commit after the 20th message has been published, the JMS client will first commit the unit of work, and will then perform an MQGet with the wait interval set to 30 seconds against the SYSTEM.JMS.REPORT.QUEUE. After the report message has been received, publication will resume and the cycle above is repeated. If the report message is not received within the wait interval, then the MQGet will complete with an error (reason 2033 – no message available) and an exception will be thrown indicating no response has been received from the broker.

The PAN responses normally sent by the JMS consumer must be sent by the measuring program (MQTimes2) to properly measure the performance of a JMS publishing application. In the case a jms publishing, a pscr reply message must be returned to the reply to queue name in the published message. This will be done if the positive acknowledgment file option (-p) is specified. An appropriate message can be captured and written to a file with the RFHUtil program (basically, publish a message with the PAN report option selected and the reply to queue name in the MQMD set to a user queue from which the reply can then be read and subsequently written to a file).

Conflicting options in the parameters file

It is possible to select conflicting options in the parameters file. For example, if the message type is set to 1 (request message) and the reply to queue or queue manager name is not specified, then the put requests will fail and the MQ error message will be displayed. Please correct the settings in the parameters file and rerun the performance driver program.

Capturing message data in files

Two utilities are provided with this Repository that will read messages from a queue and write the data to a file. The utilities are mqcapture.exe and mqcapone.exe respectively. Both utilities use a parameter file to control the necessary input parameters. The mqcapture.exe utility will read all the messages in a queue and write them to a single file. The mqcapone.exe program will always read a single message from a queue and write it to a file.

The command syntax is as follows:

```
MQCapture    -f parameter-file-name -o data-file-name
MQCapone     -f parameter-file-name -o data-file-name
```

The parameters file includes the name of the queue and queue manager that messages are to be read from. A parameter (MsgCount) can be specified in the parameters file to limit the number of messages that will be read by the MQCapture program. The delimiter sequence is also specified in the parameters file, either in ASCII (delimiter parameter) or in hexadecimal characters (delimiterx). The MsgCount and delimiter parameters are ignored by the mqcapone.exe utility.

A sample parameters file (parmcapt.txt) is provided with this Repository. This file should be modified to match the local configuration.

Driver memory usage

The MQPUT2 driver program reads all the messages that will be used in a performance run into memory prior to writing any messages to the designated queue. Therefore, sufficient memory must be available for not only the application(s) being driven but for the message driver program as well. If the driver program is forced to page its memory to and from disk, then the results obtained may not reflect the actual performance of the system.

On AIX systems, it may be necessary to increase the heap size to run the MQPUT2 program. The ulimit command should be used with the `-d` parameter to increase the heap size. This is necessary in cases where the program is reporting memory allocations failures.

Driving a remote system

If messages are written to a remote queue, then the real driver of the application becomes the MQ message channel agent rather than the driver program itself. The driver program is merely writing messages to a transmission queue, and the message channel agent on the remote system is the program that is actually writing the test messages to the input queue. It is recommended that the performance driver program be run on the same system as the broker (workload) to avoid this situation.

Measuring performance of the message broker

The author's experience has been that the number of messages processed per second normally is very even and level, with only an occasional dip or other deviation. The message rate can usually be calculated by just looking at the output. Select one or two ten or twenty-second intervals that look typical and then add up the number of messages processed and divide by the total number of seconds. The author has not seen the need to make lengthy runs, since the data generally seems very steady with little variation observed.

Rather than measuring a variety of workloads concurrently, the author generally drives each message flow individually. It may be more interesting to do separate measurements of different message sizes, so the effect of larger messages can be clearly understood. This makes it easier to tell what the performance characteristics of each message size are and may provide insights into how the throughput varies with different types of input. A variety of test cases should be run, so that the throughput characteristics of each message flow are fully understood. The simplicity of the measurement process makes this an attractive option.

Before discussing the details of the performance utilities, some background will be provided about IBM MQ performance.

MQ Performance considerations

Like any software system, MQ performance requires some knowledge of certain aspects of the MQ queue manager, especially with regards to log usage and other disk operations. The following sections attempt to give a bare overview of the most common performance considerations that affect MQ performance. Like any performance discussion, there is no certainty that the performance of any individual application will behave in the manner described.

Persistent and non-persistent messages

MQ messaging supports two classes of messages, namely persistent and non-persistent. The performance characteristics of these message types are often different. Persistent messages promise assured once and only once delivery and therefore must be logged to disk. Since they are logged to disk in all cases, the performance of persistent messages is often limited by disk performance. Non-persistent messages do not require any disk logging and the ultimate performance limitation is often some other resource limitation, such as CPU.

MQ log operations

MQ performs disk operations to two primary types of disk, namely a log file and a queue data file. The log file is only used for persistent messages, so the log file has no effect on the performance of non-persistent messages. Persistent messages, however, must be logged to disk before a unit of work can be completed.

When an application gets and puts persistent messages, no disk log activity is performed immediately if the message data will fit in the current log buffer. Each MQGet and checkpoint operation requires about 500 bytes of log space, and each MQPut operation requires about 500 bytes plus the length of the message and MQMD data. Thus, if a one-thousand byte message is read, then written and the application then issues a checkpoint, three log

operations are involved and a total of approximately 3500 bytes of data are placed in the log buffer. Once an application issues a commit, however, all data that is currently stored in the log buffer must be written to disk before the application can continue. The log buffer may contain data from other applications as well, so the size of the buffer write may be considerably larger or the buffer may need to be written sooner, if it becomes full.

Once the log buffer becomes full or an application issues a commit, the log buffer must be written to disk. If the applications are writing small messages, the limiting factor in performance is usually the rate that the disk can perform individual writes to the disk.

In the days of simple spinning disk drivers, each write requires the disk to spin around at least once. Since disks tend to spin at about the same speed (5400 and 7200 RPM are common), an individual dedicated disk drive generally cannot perform more than 75 to 100 writes per second. If the log disk is shared, then the number of log writes per second can be sharply reduced. If each application is performing a single MQPUT per unit of work, and the message sizes are fairly small (typically 5000 bytes or less), then the limiting factor for persistent messages is often the latency of the log file.

Triple write is the default log integrity. This can cause some extra overhead. In many cases modern disks, especially solid state and SAN disks, may be able to use single write integrity. It depends on the ability of the disk manufacturer to assure the integrity of data written to the disk.

Using a solid-state disk, SAN disk or a write cache can dramatically reduce the latency and increase the performance (in some cases by a factor of greater than ten). Certain disks with high-speed SSA connections and battery-backed fast write caches offer a more economically solution, with messages rates sometimes increasing by a factor of four or more. It must be emphasized that the above numbers have been observed in certain specific environments, and are not meant to be representative of what can be achieved in any particular situation. However, they are useful for understanding what performance could be achieved with a single queue manager and thus represent an upper bound.

Since non-persistent messages are not logged, the log disk is not a constraint. Therefore, generalized performance statements of what is possible are more difficult. In many cases, the CPU is the limiting factor for non-persistent messages. Non-persistent messages can also benefit from larger memories and large buffers. In some cases the peak rate of an application using non-persistent messages has been seen to be ten times or more that of the same application using persistent messages.

In all cases, there is wide variation in the other resources and facilities that any individual application will be using, and therefore the degree to which MQ is a constraint or even a significant consideration in the performance of the application. The applications cited in the above examples were applications that primarily used MQ facilities. If an application is constrained by some other software or hardware considerations, then changes to MQ may have little or no effect on the overall performance of the application.

Queuing disk operations

When messages are written to a queue they are stored in a buffer in memory. In fact there are two buffers, one for persistent and the other for non-persistent messages. When the in-memory buffers are full then messages will be removed from the buffer and written to a queue file on disk. Later when these messages are read they must be retrieved from disk. An in-memory index allows for rapid retrieval of messages based on message id, correlation id and/or group id. Hashing is used to reduce the amount of storage that is required.

If messages are read before the buffer in memory is filled then they are never written to the queue file (persistent messages will be logged). There can be a significant difference in performance between reading messages in an in-memory buffer and reading messages from the queue file (memory speed vs disk speed).

The size of the persistent and non-persistent buffers can be changed. Default values can be set in the qm.ini file. These values are used when a queue is created. The non persistent queue buffer size is specified using a tuning parameter with the name of DefaultQBufferSize.

The persistent queue buffer size is specified using a tuning parameter with the name of DefaultPQBufferSize.

A utility called qtune that is part of SupportPac MS0P can be used to display and change these values on an existing queue. The queue should be closed before these values are changed.

The buffer for the non persistent messages has a default size of 64K for the 32 bit queue managers (Windows, Linux32) and 128K for 64 bit Queue Managers (AIX, Solaris, HP/UX, Linux64). The buffer for persistent messages has a default size of 128K for 32 bit Queue Managers and 256K for 64 bit Queue Managers. The maximum size supported for both queue buffers is 100MB.

The buffers can be set to a value as high as 100 MB. This should be done sparingly to avoid excessive memory consumption. Increase the memory buffers for highly utilized queues. Accept default values or modest increases for other queues.

Units of work

It is recommended that all applications that are using persistent messages use explicit units of work. This is necessary to ensure the integrity of the message processing as well as for performance reasons.

Applications which process persistent messages outside of a unit of work or sync point have to wait after each and every MQGET or MQPUT for the log record to be synchronously written to the log. While that log write is taking place access to the queue is inhibited for other applications. When explicit units of work are used a shared lock is taken against the queue. This allows other applications to process messages and commit units of work in parallel.

Performance enhancements in recent versions

Recent versions of MQ continued the trend of significant performance improvements for both persistent and non-persistent messages. The processing paths for persistent messages have also been improved, but the more significant improvements have resulted from significant changes in the logging algorithms. There have been significant improvements in the ability to use multiple processors by a single queue manager. These changes have involved things like internal locking and are not visible externally.

Queue buffer size

The default buffer size for a queue is 64K. When the buffer is full the contents must be written to disk. This results in additional disk overhead when the buffer overflows and messages are first forced out to the queue file and later read back in from the queue file when the messages are eventually processed. If a message is written (put) to a queue and is then processed before the queue buffer is full then the message will never be written to the queue file on disk. Please note that logging of persistent messages is a separate operation and does not use the disk queue file.

The buffer size for a queue is determined when the queue is defined. The default value can be changed by adding a special parameter to the qm.ini file (non-windows) or in the register for Windows. Any queues that are created after this parameter is changed will use the new value rather than the default value.

If messages are being consumed rapidly then increasing the size of the queue buffer can increase performance and overall throughput. The primary case where increasing the queue buffer will help is where messages are being consumed shortly after they are created. A larger buffer can hold more messages in memory from the time they are created until they are processed.

If messages are not being read promptly after they are created, then the queue buffer will most likely have overflowed anyway and increasing the size of the queue manager will not have any affect on either performance or throughput.

To change the size of the queue buffer size you must add a TuningParameters stanza to the qm.ini file and then add a DefaultQBufferSize=nnn value under the TuningParameters stanza. The value should be a number between 64K (65536) and 1024K (1048576). For example you could specify DefaultQBufferSize = 1000000. This value only applies to new queues that are created after this value is changed (existing queues could be deleted and recreated to change the size of the queue buffer). It is a good idea to remove this parameter after any desired queues are defined or redefined.

A more convenient way to change and display the queue buffer sizes is to use the qtune utility that is provided with Repository MS0P.

Log buffer pages

The log files can have a significant affect on performance of applications that use persistent messages. The size of the memory buffer for logging operations is a property of the queue manager and can be set. The minimum number of log pages is 18 and the maximum number is 512. Each log page is 4K (4096) bytes in size. If the value is set to 0 then the queue manager will select a value, which is usually 64. This results in a log buffer that is 256K bytes in size. If persistent messages are being used and the peak message rate is significant then this value should usually be increased from the default value.

Connections, disconnections and queue open/close

Although queue manager connections and the opening and closing of queues can be very significant in many applications, they do not tend to make much difference in the case of the message broker, since the connections and open queue handles are maintained between message flows and reused by later instances. Queues are opened on first use and remain open.

Suggestions for performance testing of the Message Broker

Now that some background on MQ performance has been given, some recommendations will be given.

- 1) If the message rate is less than about 20 messages a second, then MQ performance is probably not critical. If the message rates are expected to be above 50 messages a second for persistent messages, or 200 messages a second for non-persistent messages, then tuning of MQ is probably necessary.
- 2) If persistent messages are being used, use a dedicated disk for MQ logging. If possible, use a disk with a SSA connection and a fast write cache. Increase the number of log pages to 32. Message rates above 75 will require special hard for the MQ log disk (fast write cache or solid state). Multiple queue managers and therefore multiple brokers may also be necessary.
- 3) If the primary interest is in the performance of message flows, use non-persistent messages even if the application will use persistent messages in a production environment. This tends to limit the impact that MQ itself will have on the measurements.
- 4) As a rule of thumb, message flows tend to be CPU bound. A faster CPU or more CPUs will generally increase proportionally to the relative CPU power, assuming that no base MQ or other software constraints limit the performance.
- 5) Use MQ V7.1 or V7.5, for the maximum performance.
- 6) Ensure that enough memory is available on the test system for both the application to be tested and the MQPUT2 driver program.

Parameters file for the MQPUT2, MQPUTS and MQTEST utilities

The MQPUT2 workload performance utility uses a text file to specify the values of a number of parameters. These parameters include the name of the queue and queue manager that messages are to be written to, the name of the file(s) that contains the message data, timing parameters and values for fields in the MQ message descriptor and Rules and Formatting Header (RFH) fields. The parameter names can be any combination of upper and lower case letters. Trailing blanks in the parameter values are truncated. Parameter values can be enclosed within double quotation marks (""), in which case the value will be assumed to be between the leading and trailing quotation marks.

Blank lines, as well as any line that begins with an asterisk ("*"), semi-colon (";") or a number sign("#"), are considered comments. A fully commented sample parameters file is provided.

The parameters file is divided into sections of two types. The section headers are enclosed in square brackets ("[]"). The two section types (including the square brackets) are:

- [HEADER]
- [FILELIST]

The initial section header for the HEADER section can be omitted. The section header for the file list is required.

The following fields are used to set the number of messages to be written to the queue, the queue to write to and to control the rate that the messages are written to the queue.

- Qmgr Name of the local queue manager to connect to
- Qname Name of the queue to write messages to
- RemoteQM If the queue is remote, name of remote QM
- Msgcount Total number of messages to write
- WriteOnce Ignore msgcount and write each message once, then end
- SleepTime Time in milliseconds to wait before checking queue depth (mqput2)
- Qdepth Queue depth at which messages will be written
- Qmax Maximum queue depth
- ThinkTime Delay time in milliseconds after each message is written (mqputs)
- setTimeStamp Will overwrite user data with timestamp
- UseMQMD Look for and use embedded MQMDs in data files
- ignoreMQMD Look for but ignore any MQMDs in data files
- newMsgId Generate new message identifier when using embedded MQMD
- Batchsize Maximum number of messages to write in a single unit of work
- Tune Indicator whether to adjust sleeptime automatically and report

The following fields in the message descriptor can be set:

- ReplyToQ Reply to queue name
- ReplyToQM Reply to queue manager
- Format Format of the user data
- Priority Message priority
- Persist Message persistence
- Msgtype Message type (e.g. reply, request, datagram, report, etc)
- Codepage Code page of the RFH if present or user data if no RFH
- Encoding Encoding type of the RFH if present or the user data if no RFH
- Correlid Correlation id (specified in ASCII characters)
- Correlidx Correlation id (specified in hex characters)
- Groupid Group id (specified in ASCII characters)
- Groupidx Group id (specified in hex characters)

The following fields can be used to set the Rules and Formatting Header (RFH):

- RFH Indicator if Rules and Formatting Headers are to be used
- RFH_CCSD Code page of the user data
- RFH_Encoding Encoding of the user data

- RFH_DOMAIN Message domain (V2 only)
- RFH_MSG_SET Message set (V2 only)
- RFH_MSG_TYPE Message type (V2 only)
- RFH_MSG_FMT Message fmt (V2 only)
- RFH_APP_GROUP Application group (V1 only)
- RFH_FORMAT Message format (V1 only)

The following fields can be used to set fields in the publish and subscribe folder (psc) in an RFH2 message header.

- RFH_PSC_REQTYPE Request type
- RFH_PSC_TOPIC1 Topic (up to 4 topics can be specified)
- RFH_PSC_TOPIC2 Topic (up to 4 topics can be specified)
- RFH_PSC_TOPIC3 Topic (up to 4 topics can be specified)
- RFH_PSC_TOPIC4 Topic (up to 4 topics can be specified)
- RFH_PSC_SUBPOINT Subscription point
- RFH_PSC_FILTER Filter
- RFH_PSC_REPLYQM Publish/subscribe reply to queue manager
- RFH_PSC_REPLYQ Publish/subscribe reply to queue
- RFH_PSC_PUBTIME Publication time
- RFH_PSC_SEQNO Sequence number
- RFH_PSC_LOCAL Local only
- RFH_PSC_NEWONLY New only
- RFH_PSC_OTHERONLY Other only
- RFH_PSC_ONDEMAND On demand (will use request publication)
- RFH_PSC_RETAINPUB Retained publication (set by publisher when publishing message)
- RFH_PSC_ISRETAINPUB This is a retained publication (set by broker)
- RFH_PSC_CORRELID Include correlation id in published messages
- RFH_PSC_DEREGALL De-register all (unsubscribe only)
- RFH_PSC_INFRETAIN Inform if retained publication

The request type must be one of the five valid publish and subscribe command types. The valid command types are:

- Subscribe "Subscribe", "Sub" or "1"
- Unsubscribe "Unsubscribe", "Unsub" or "2"
- Publish "Publish", "Pub", or "3"
- Request publication "ReqPub" or "4"
- Delete publication "DelPub" or "5"

Publish and subscribe options (RFH_PSC_LOCAL through RFH_PSC_INFRETAIN) are Boolean values and should be specified as either "Y" or "1" for yes and "N" or "0" for no. The default is no.

The following fields can be used to set fields in the jms folder in an RFH2 message header.

- RFH_JMS_REQTYPE JMS data type
- RFH_JMS_DEST Destination object name
- RFH_JMS_REPLY Reply to object name
- RFH_JMS_CORRELID Correlation id (JMS)
- RFH_JMS_GROUPID Group id (JMS)
- RFH_JMS_PRIORITY Message priority (JMS)
- RFH_JMS_EXPIRE Expiration time (JMS)
- RFH_JMS_DELMODE Delivery mode
- RFH_JMS_SEQ Sequence number (JMS)

The JMS data type must be one of the following values":

- text “text” or “1”
- bytes “bytes” or “2”
- stream “stream” or “3”
- object “object” or “4”
- map “map” or “5”
- none “none” or “6”

Command line arguments and overrides

In most cases the utilities use a parameters file to specify the program options. A limited number of options can also be specified on the command line as program arguments. If a value is specified on the command line it will override any value found in the parameters file.

The following command line arguments are supported.

- -f Name of the parameters file (fully qualified)
- -m Queue manager name
- -q Primary queue name
- -b Batch size (number of messages in a unit of work)
- -t Think time (delay in milliseconds between batches of messages)
- -c Message count
- -v Verbose output
- -p Purge (drain) queue before reading messages

Embedding RFH headers in the message data file

Rules and Formatting headers (RFH) can be imbedded in the user data file. However, if this is done, the message format, code page and encoding fields in the MQMD must be set to match the RFH header. If the encoding field does not match the format used in the RFH header, an MQ error (reason code 2334) will be generated.

Using more than one message data file

Support is provided for more than one message data file in a single parameters file. The driver utility program will process as many data file names as are found in the parameters file. The entire contents of each data file are read into memory during program initialization and are held in memory for the duration of the program execution. There are several important considerations.

The driver utility program will write one message of each type (file), in the sequence that the files are specified, and will then return to the first message specified in the parameters file. This process will be repeated until the number of messages written is equal to the number specified in the message count parameter. If the message count is not a multiple of the number of data files, then more of some message types written than other message types. The same file name can be specified more than once, but each instance of the file will be considered to be a different file and the file data will be read into memory more than once.

The MQ message descriptor and Rules and Formatting header fields for the file are determined by those specified in the parameters file at the time the message data file is read into memory.

In all cases, the messages will be written to a single queue. The queue and queue manager names for the output queue are those in effect when the entire parameters file has been read and processed. If the queue manager or queue names are included more than once, then the later settings will overwrite the earlier settings.

Using different parameters for different message data files

When using more than one message data file in a parameter file, it is possible to use different values for the message descriptor and Rules and Formatting header fields. This is accomplished by using more than one header and more than one file list section in the parameters file. The values for the first file are specified in the first header section. These settings are then saved with the file data. A second header section and a second file list section then follow the first file list section. The parameters in the second header section then modify the values in the MQ message descriptor and/or Rules and Formatting header. The new or changed values are then used for the files in the second file list. This process can be repeated as many times as necessary, using a different header for each data file if necessary.

Multiple messages in a single data file

Each input data file used by the performance driver program can contain one or more messages. If a file contains more than one message, then a delimiter sequence must be inserted between each pair of messages. No delimiter is needed at the beginning or end of the file. The delimiter sequence must be specified in the parameters file using the delimiter or delimiterx file.

Handling of RFH and RFH2 headers

There are three basic options for the use of Rules and Formatting headers (RFH) by the performance driver utility. The first option is “No”, which indicates that no Rules and Formatting headers are to be used. The second option is “Auto” (automatic). This option indicates that the program should look for a Rules and Formatting header at the front of the message data file. If it finds what appears to be a valid Rules and Formatting header at the beginning of the data file, then the settings for the Rules and Formatting header are taken from the message data file. The codepage and encoding parameters should still be set in the parameters file, to reflect the RFH header. The final option is to add a header based on parameters set in the parameters file. In this case, the appropriate fields in the parameters file should be set to the desired values, as described below. If an MQMD is stored with the data and the useMQMD option is selected, then the RFH parameter is ignored when an MQMD is found ahead of the data in the file.

Overall handling of Rules and Formatting headers (RFH) is controlled by the RFH parameter in the parameters files. RFH headers can be stored with the message data in a message data file or can be added based on values provided in the parameters file.

The RFH parameter can be set to one of five values. If the RFH parameter is set to “N” or “No”, then no RFH headers will be used. If the RFH parameter begins with an “A” (such as “A”, “Auto” or “Automatic”), then the program will look for an RFH at the beginning of the message data file. If the first four characters are “RFH ” (in either ASCII or EBCDIC) and the next four characters are either a binary 1 or a binary 2 (in either big-endian or little-endian format), then it is assumed that the file contains an RFH. If the parameter begins with an “X” (such as “XML” or “X”), then a special version of the RFH will be built at the beginning of the messages, to indicate that the messages are XML messages. If the parameter is set to “V2” (or “2”) or “V1” (or “1”), then a version 2 or version 1 RFH will be inserted at the beginning of the message. The individual parameters will be taken from values set in the parameters file. This parameter can be abbreviated to a single character, which can consist of “N” for No, “X” for XML, “A” for Automatic, “2” for version 2 and “1” for version 1.

Specifying RFH2 folder contents

There are three ways to specify the contents of the various folders used with Rules and Formatting headers (RFH) by the performance driver utility. The first option allows fields to be specified as individual parameters. The utility will then build the actual folder based on whichever fields are actually specified. The second technique is to specify the folder contents as a fully formatted string with the XML tags specified. The third technique is to store the RFH2 header with the data in the user data file.

If the headers are stored with the user data, then the only requirement is to set the RFH parameter to A (for automatic). The mqput2 utility will then automatically recognize the presence of an RFH header at the front of the user data and will set the format field appropriately. If the RFH parameter is set to a value of 2, then the utility will build an RFH header and insert it before the data when writing a message. The contents of the folders in the RFH header can be specified either as individual fields or as a block of text in an XML format.

If an individual field in a particular folder is specified in the parameters file, then any existing folder contents are discarded and the corresponding folder is rebuilt based on the current contents of the individual parameters. The contents of the mcd, jms and psc folders can be specified this way. This technique cannot be used for the pscr or usr folders, nor can it be used if non-standard or otherwise unsupported tags are to be used.

If a line is found that matches the XML tag for the beginning of an mcd, jms, psc, pscr or usr folder, then all text found in the parameters file is appended to the corresponding folder contents until the corresponding ending tag is found. In all cases, the beginning tag and the corresponding ending tag must be on individual lines in the parameters file. The rest of the folder contents can be on individual lines or can be on a single line in the parameters file. Any blanks, tags or new line characters found at the beginning or end of any such lines in the parameters file will be discarded, as will any lines that begin with a comment character.

For example, the following lines in the parameters file would set the contents of the psc folder.

```
<psc>
  <Command>RegSub</Command><Topic>STOCK/IBM</Topic>
</psc>
```

If a specific folder should no longer be included in the RFH2 header, the contents of the corresponding folder can be reset with a corresponding reset command, as follows:

```
RESET_MCD=Y
RESET_JMS=Y
RESET_PSC=Y
RESET_PSCR=Y
RESET_USR=Y
```

RFH parameters

If the RFH parameter is set to “V1” or “1”, then the following fields in the parameters file are used to build the RFH:

- RFH_APP_GROUP
- RFH_MESSAGE_SET
- RFH_CCSID
- RFH_ENCODING

If the RFH parameter is set to “V2” or “2”, then the following fields in the parameters file are used to build the fixed portion of the RFH header.

- RFH_CCSID Integer
- RFH_ENCODING Integer
- RFH_NAME_CCSID Integer (should be set to 1208)

If the RFH parameter is set to “V2” or “2”, then the following fields in the parameters file are used to build the mcd folder in the RFH:

- RFH_DOMAIN String
- RFH_MESSAGE_SET String
- RFH_MESSAGE_TYPE String
- RFH_MESSAGE_FMT String

The following parameters can be used to build the publish/subscribe folder (psc) in the RFH2 header:

• RFH_PSC_REQTYPE	See below
• RFH_PSC_TOPIC1	String
• RFH_PSC_TOPIC2	String
• RFH_PSC_TOPIC3	String
• RFH_PSC_TOPIC4	String
• RFH_PSC_SUBPOINT	String
• RFH_PSC_FILTER	String
• RFH_PSC_REPLYQM	String
• RFH_PSC_REPLYQ	String
• RFH_PSC_PUBTIME	String
• RFH_PSC_SEQNO	Integer
• RFH_PSC_LOCAL	Boolean
• RFH_PSC_NEWONLY	Boolean
• RFH_PSC_OTHERONLY	Boolean
• RFH_PSC_ONDEMAND	Boolean
• RFH_PSC_RETAINPUB	Boolean
• RFH_PSC_ISRETAINPUB	Boolean
• RFH_PSC_CORRELID	Boolean
• RFH_PSC_DEREGALL	Boolean
• RFH_PSC_INFRETAIN	Boolean

The publish and subscribe request type (RFH_PSC_REQTYPE) should be set to one of the options in the right hand column indicated below:

Subscribe	“Subscribe”, “Sub” or 1
Unsubscribe	“Unsubscribe”, “Unsub” or 2
Publish	“Publish”, “Pub” or 3
Request pub	“ReqPub” or 4
Delete pub	“DelPub” or 5

The Boolean fields can be selected by specifying a value of “Y” and unselected by specifying a value of “N”.

The following parameters can be used to set fields in the jms folder in the RFH version 2 header:

• RFH_JMS_REQTYPE	See below
• RFH_JMS_DEST	String
• RFH_JMS_REPLY	String
• RFH_JMS_CORRELID	String
• RFH_JMS_GROUPID	String
• RFH_JMS_PRIORITY	String
• RFH_JMS_EXPIRE	String
• RFH_JMS_DELMODE	String
• RFH_JMS_SEQ	String

The jms request type (JMS_REQTYPE) should be set to one of the options in the right hand column indicated below:

JMS text	“Text” or 1
JMS Bytes	“Bytes” or 2
JMS Stream	“Stream” or 3
JMS Object	“Object” or 4
JMS Map	“Map” or 5

If the RFH parameter is set to XML or Auto, the code page and encoding parameters for the RFH are used. If the RFH parameter is set to No, the other RFH parameters are ignored.

General parameters

Certain input parameters are global in nature and should only be specified once in the parameters file. If the parameters are specified more than once, the last value encountered in the file will be used. These parameters are as follows:

- Output queue and queue manager name
- Total number of messages to be written
- Whether embedded MQMDs are to be used if found
- Whether latency is to be measured (see discussion below)
- Time to sleep before checking the depth of the output queue (mqput2 only)
- Think time to wait after a message is written to a queue (mqputs only)
- Whether dynamic tuning of sleep time is enabled (mqput2 only)
- Report intermediate progress after specified number of messages
- Desired minimum and maximum queue depths (mqput2 only)
- Maximum number of messages to write in a single unit of work

The queue manager (QMGR) name is the name of the queue manager to connect to. This should normally be the queue manager that the queue is defined on. This is necessary if the normal version of the driver utility is used, since the program does inquiries to determine the current depth of the queue and these inquiries are not supported for remote queue definitions.

If the queue exists on a remote queue manager, then several factors must be taken into account. First, the queue can be accessed either directly by explicitly giving the name of the queue manager that the queue is defined on, or by creating a remote queue definition on the local queue manager. Although a version of the utility is provided that does not use the inquiry API to determine the queue depth, the author does not recommend this approach. The problem is that the driver program for the workload has in effect become the message channel agent on the remote system. While this may be what is desired, a detailed understanding of the inner workings of the message channel agents may be required to understand exactly what is being measured.

The MQPut2 program attempts to maintain a certain number of messages in the target queue. The minimum and maximum queue depth parameters (qdepth and qmax) are used to maintain a relatively steady number of messages in the queue. The driver utility will sleep for a specified period and will then wake up and check the depth of the queue. If the depth is below the minimum (qdepth) value, then the program will write calculate the difference between the current queue depth and the maximum desired queue depth (qmax) and write this number of messages to the queue. If the number of messages written is less than the batch size parameter, all the messages written will be in a single unit of work. Otherwise, more than one unit of work is used.

The sleep time parameter is the number of milliseconds that the program should sleep before it checks the depth of the queue. The sleep time parameter should be set to an amount that ensures the queue depth will never reach zero, but large enough so that the program does not wake up and check the queue depth more often than is necessary. The minimum value for this parameter is 10 (.01 seconds) and the maximum is 10000 (10 seconds). This parameter can be adjusted dynamically if the tune parameter is set to a one ("1"). The author recommends using the tune parameter to modify the sleep time parameter to find an optimal setting. The optimal setting allows the qdepth to drop to the desired minimum level in one or two sleep time intervals. For example, if the desired minimum queue depth is 20 and the desired maximum queue is 40, and the workload is executing at 40 messages a second, then the optimal sleep time is approximately 500 milliseconds.

If dynamic tuning is requested, the following tuning algorithm is used. The sleep time interval is increased and/or decreased after each sleep time interval until an optimal value is found. The algorithm is fairly simple. If the depth of the queue has not changed since the previous time interval, the sleep time interval will be increased by a factor of two plus one millisecond. If the queue depth is less than the amount at the beginning of the previous interval but still more than the minimum queue depth, then the sleep interval is increased by one eighth of the current value plus one millisecond. If the queue depth is zero, the sleep time interval is divided by two. In all cases, the new sleep interval cannot be less than ten milliseconds or greater than 10000 milliseconds (10 seconds). The purpose of this algorithm is to quickly find a value that is a reasonable compromise between waiting too long and having the queue become empty and checking the queue depth more often than necessary and incurring unnecessary overhead. A large number of console messages are produced if the tuning option is selected, to help to understand the optimal setting.

The author recommends that the minimum and maximum queue depth parameters and the sleep time parameter be modified and tried with the tuning parameter turned on. After the approximate optimal values for these parameters are understood, then the tuning parameter should be turned off and the desired values set in the parameters file.

In all cases, an error message will be generated if the queue depth is found to be zero after any time interval. Any runs in which the queue depth falls to zero should be discarded, since they are not measuring maximum throughput.

The `reportafter` parameter can be used to report progress while `MQPut2` is running. It will write a line to the console after the specified number of messages has been written, and will report the elapsed time to write the messages.

The `MQPUTS` program uses a different technique. An optional think time parameter (`Thinktime`) can be used to indicate the number of milliseconds that the `MQPUTS` or `MQPUTSC` program should sleep before moving on to the next message. It supports cases where messages are to be written at a specified rate. In particular, it can be used in situations where the depth of the queue cannot be queried, as in the case of remote queues. The think time parameter will force a commit after writing the number of messages specified by the batch size parameter. This allows several messages to be written as a batch, and then have the program sleep for a specified time. The think time parameter is ignored by the `MQPUT2` program.

Storing and using MQMDs in user data files

The driver utilities will look for MQMDs embedded in user data files if and only if the `UseMQMD` parameter is set in the parameters file. The program will look for the MQMD signature ("MD ") at the beginning of the user data, followed by a valid version number in binary (either a 1 or a 2). If a correct signature is found, then the data will be assumed to contain an MQMD and the presumed MQMD will be removed from the message and will be used whenever the particular message is written.

It should be noted that when an embedded MQMD is used, all MQMD field specifications in the parameters file are ignored. All MQMD fields will be set from the embedded MQMD. This includes the message, group and correlation identifiers. If the same message data is written more than once, then the identifiers may result in messages with duplicate identifiers, which can cause curious side effects. Therefore, these options should be used with care.

If MQMDs have been stored with the data but are not to be used, then the `ignoreMQMD` parameter can be used. MQMDs will be identified, but ignored. The `ignoreMQMD` and `useMQMD` parameters should not be used at the same time. If they are the `ignoreMQMD` parameter will take precedence.

If all fields in the message except for the message identifier are to be used, then the `newMsgId` parameter can be set (to either 'Y' or '1') in the parameters file, then all MQMD fields will be taken from the data file except for the message identifier.

MQMD data can be saved with the data by both the `MQCapture`, `MQCapSub` and `MQCapOne` utilities, and by the `RFHUtil` program. In either case, an option must be selected if the MQMD

is to be saved with the data. The UseMQMD parameter must be set for the MQPut2 utility to look for and use the embedded MQMD.

Warning when using MQMDs stored with the data

If MQMDs are stored with the data, some care must be taken in any message flows to create a new MQMD in the output. The reason is the measurement utilities use the time field in the MQMD to determine when a message was written. Since the time field in the MQMD will reflect the time the message was written before it was captured, it will not accurately reflect the time the message was written by the performance driver program and the messages per second reported by the MQTimes2 or MQTimes programs will not reflect the actual throughput of the system.

Tune parameter for the Performance Driver Utility

If the TUNE parameter is set to 1 in the parameters file, the program will dynamically adjust the sleep time to try to match the rate at which messages are being processed. The following algorithms are used.

If the queue depth reaches zero, the sleep time parameter is divided by two and an error message is generated. If the qdepth has not changed since the last time the sleep time interval expired, the sleep time interval will be increased by one-half of the current value plus one millisecond. If the qdepth has decreased but the value has not reached the desired value, then the sleep time interval will be increased by one-eighth of the current value plus one millisecond. If the qdepth is less than the desired value the interval will be decreased by one-eighth of the current value plus one. In no case, will the new value be less than ten milliseconds nor more than ten seconds. Every time the value is changed, a message will be written to stdout. Console messages are also written to stdout each time MQ messages are added to the queue. Tuning messages will not be written to the console if the TUNE parameter is turned off (set to zero).

The intent of the tune parameter is to find an optimal value for the sleep interval. Once that value has been determined, then the value should be set in the parameters file and automatic tuning is then turned off for the actual performance measurement runs.

MQMD fields

Certain fields in the MQ message descriptor can be set in the parameters file. All of these parameters are optional. If these fields are set in the parameters file, the values will be used for all messages that are written to the output queue. The fields that can be set are:

- Format
- Persistence
- Priority
- Reply to Queue and Queue Manager
- Correlation Id (either character or hex)
- Group Id (either character or hex)
- Message Type
- Expiry

If the above fields are set in the parameters file, the values in the parameters file will be used for all messages that are generated by the driver utility.

The format field can be up to eight characters long, and will be padded with blanks on the right. The persistence field can be 0 for non-persistent, 1 for persistent or 2 for persistence as defined by the queue. Message priority can be any value greater than or equal to zero. If the priority is set to zero, then default priority is assumed. The Reply To Queue and Queue Manager fields can be up to 48 characters in length. The message expiry can be specified as

a positive integer to create messages with a definite expiration time. If the expiry parameter is set to zero, then the messages will have no expiration time.

The message type parameter (MSGTYPE) is specified as a number. The following values are allowed for this parameter:

• Request	1
• Reply	2
• Report	4
• Datagram	8
• MQ/E fields from MQ/E	112
• MQ/E fields	113
• User	65536 to 999,999,999

The message type should be specified as the number that corresponds to the desired message type.

The correlation and group id fields can be specified as either ASCII text or as a hexadecimal string. Each field has two different keywords that indicate if the corresponding field is being specified as ASCII characters or as hexadecimal characters.

Measuring latency

Latency is the time to process one message and is measured in microseconds.

The most common measurement of system performance is throughput. Throughput is defined as the maximum number of messages that a particular application (system) can process in a given time interval and is usually specified as messages per second. However throughput does not provide any indication of how long it takes to process an individual message. The time to process an individual message can also be measured. It is usually referred to as latency.

A capability to measure latency is provided in the MQPUTS and MQTIMES2 / MQTIMES3 measurement programs. When measuring latency using these programs a timestamp is stored at some location in the original message. When an output message is read by the measurement program the original timestamp indicates when the original message was written. This is then compared to the current time value and the difference represents the latency. Minimum, maximum and average latencies are reported. In addition a histogram like report is produced that shows the relative numbers of latencies that fall within certain ranges. This can sometimes give a more accurate picture of latency, as outliers are obvious.

Latency is measured in microseconds. If latency is to be measured, the driver and measurement programs must be run on the same system, so they are using the same clock. Furthermore, some user data may be overwritten with a high-precision timestamp, which is then used by the timer program to measure the latency.

When latency is measured, a 64-bit high-precision timestamp will be placed in the original message. This timestamp must be passed along to the final message in the same location. The measurement programs will look for this timestamp and calculate the time difference. The time difference will be reported as latency.

Both the driver program and the timing program must be started with options to use the latency feature. The timestamp is generally accurate to the microsecond range, so most latencies can be measured.

Some due caution must be used when measuring latency. If the MQPUT2 program is used, the latency is probably overstated, since the messages will be placed on a queue and will not be processed immediately. When measuring latency, it is generally better to use the MQPUTS program and use the thinktime parameter to control the message rate. The batchSize parameter should be set to 1. This allows messages to be written at a given rate

and the latency to be measured. The message rate can be increased until the capacity of the system under test is reached.

To turn on the latency measurement the `setTimeStamp` parameter set to Y in the parameters file. By default the timestamp will be written to the first 8 bytes of the message and the queue manager name will be written after the timestamp. The timestamp and queue manager name will overlay the user data at these locations. An additional parameter allows the offset where the timestamp is placed to be controlled (`TimeStampOffset`). There are additional options provided for the location of the timestamp, which allow the timestamp to be stored in the correlation id, group id or accounting statistics within the MQMD or as a user property (user folder in an RFH2 header). In these cases the user data is not interfered with.

Measuring latency using the MQLatency utility

An additional utility is provided that allows the direct measure of latency. In this case the timestamp is stored locally rather than being carried in the original message.

The MQLatency utility will write messages one at a time. After each message is written the utility will read a message from a different queue. The time that elapses between the completion of the MQPUT and the matching MQGET is reported as the latency to process a message. The utility will write a specified number of messages and will report the minimum, maximum and average latency for all the messages. A parameter (`thinktime`) can be used to specify a delay in milliseconds from the time a reply message is received and the next request message is written. If this parameter is not specified then the next input message is written as soon as the previous reply message is received. The utility will continue to write messages until a maximum count is reached or until an MQ error is received. The reply messages are read and discarded.

The MQLatency utility can be used to measure latency in a number of scenarios. It can measure the time that a message flow takes to process an individual message. It can also measure the latency of request and reply scenarios. In all cases the reply queue must be on the same queue manager as the queue that messages are written to. If a message flow is being used to process a message then the MQLatency utility can be used to first drive the message flow by sending a message to the input queue and then reading the corresponding output message when it appears on the output queue. In a similar manner the utility can drive a request and reply scenario.

Parameters for the MQLatency utility

In all cases the name of a parameters file must be specified. The parameters file is the same as that used by the MQPut2 and MQPuts utilities. The batch size and queue depth parameters are ignored since the utility is sending one message at a time and then waiting for a reply. The name of the parameters file must be specified on the command line using the parameter file command line argument (`-f`).

A number of parameters can be specified on the command line when the utility is executed. The individual command line parameters can be displayed by just executing the utility with no input parameters.

The queue manager (`-m`), queue (`-q`) and reply queue (`-r`) parameters can be specified or overridden on the command line. These parameters can also be specified in the parameters file. The message count (`-c`) and think time (`-t`) parameters can be specified in the parameters file and can be overridden on the command line.

The verbose option (`-v`) will slightly increase the amount of output that is produced. The purge (`-p`) parameter will read and discard any messages found in the reply queue before any messages are sent. This prevents incorrect results from messages being present in the reply queue before the utility is started. The correlation id parameter (`-i`) will use a correlation id to match the reply message with a particular request message. If this option is used the

processing application must set the correlation id to match the original message id. This parameter can prevent the latency program from mistaking a message that is in the reply queue for the actual reply message.

Caveats when using the MQLatency utility

Latency is difficult to measure accurately. If messages are allowed to sit in a queue while other messages are processed then the measured latency can be very misleading since it is largely a function of the number of messages in a processing queue rather than the time it takes to actually process the message. In a similar fashion the use of remote queue managers with channel connections can also vary widely depending on the depths of the various transmission queues.

The MQLatency program makes no attempt to saturate the system under test. If it is desired to measure the latency of a given system at some particular level of load then other test tools such as mqputs and mqtimes2 can be used to generate workload while the MQLatency utility measures the latency of individual messages. As a system approaches saturation (maximum throughput) the latency usually increases dramatically and can become quite variable. As a result latency measurements are usually conducted on systems that are more lightly loaded.

The reply queue should be dedicated and not contain any other messages. An option is provided to purge the queue when the utility is started. This option will prevent reporting incorrect results when a message is sent and messages already in the queue are mistaken for the actual reply message. In this case abnormally low latencies will be reported.

In general group processing can conflict with the techniques used by the MQLatency utility to measure latency. Therefore group processing options are ignored if specified in the parameters file.

Using the mqreply program

The mqreply program will receive request messages and send a reply message to the reply queue that is indicated in the MQMD. The reply message type will be set to a reply message. The correlation id of the reply message will be set to the message id of the incoming request message.

There are two options for the content of the reply message. A reply message file can be specified. The contents of the file will be read into memory and used as the body of any reply messages that are sent. If the resendRFH parameter is set then any RFH header on the incoming message will be prepended to the reply message and included as part of the reply message. This option allows things like message properties to be returned to the receiving application.

A second option is to return the contents of the request message as the reply. This option is selected by using the useInputAsReply parameter.

The mqreply program will open the reply queue when the first reply is sent. If subsequent request messages have reply to queue name then the open queue will be reused. If the reply queue is different then the current reply queue will be closed and the new reply queue will be opened.

Format of the parameters file for the data capture utilities

The data captures utilities (MQCapture and MQCapone) also use a simple parameters file to specify certain attributes, such as the name of the queue and queue manager that messages are to be read from. A sample parameters file (parmcapt.txt) is provided with this Repository. Both utilities can use the same parameters file (the MQCapone utility will ignore the message count and delimiter parameters).

The following fields are used to set the maximum number of messages to be read from the queue, the queue to read from, and whether or not RFH headers are to be stripped off or saved in the file with the message data.

- Qmgr Name of the local queue manager to connect to
- Qname Name of the queue to write messages to
- StripRFH Indicates if RFH headers are to be removed (Y or N)
- ReadOnly Use non-destructive get to read messages from queue
- SaveMQMD Save MQMD with message data (Y or N)
- MaxMsgLen Maximum message length override
- MsgCount Maximum number of messages read from the queue
- IndivFiles Whether to put each message in a separate file
- MsgId Only read messages that match this message id
- MsgIdX Same as MsgId but specified in hex rather than characters
- CorrelId Only read messages that match this correlation id
- CorrelIdX Same as CorrelId but specified in hex rather than characters
- GroupId Only read messages that match this group id
- GroupIdX Same as GroupId but specified in hex rather than characters
- Delimiter Delimiter sting specified in character (ASCII) format
- Delimiterx Delimiter string specified in hexadecimal characters

The Qmgr and Qname parameters specify the name of the queue manager and queue that messages are to be read from. The StripRFH parameter indicates if RFH headers are to be removed from messages or if they are to be left at the front of the data and written to the file. The MsgCount parameter indicates the maximum number of messages that will be read from the queue and written to the file. Only one of the delimiter and delimiterx parameters should be specified in a given parameters file. The MsgCount and delimiter/delimiterx parameters are ignored by the MQCapone utility, since this version of the utility reads only a single message from the input queue.

If the SaveMQMD parameter is set to 'Y' or '1', then the MQMD will be stored in the data file at the front of the user data. If the ReadOnly parameter is set to 'Y' or '1', then a non-destructive MQGet will be used instead of the normal MQGet. This will leave the message on the queue rather than removing it.

The MaxMsgLen parameter can be used to override the maximum message length that the MQCapture and MQCapOne programs support. The default maximum message length is 16 megabytes. If a larger message is to be captured, then the MaxMsgLen parameter must be used to increase the maximum message size that can be read.

The delimiter string is used when more than one message is written to a file. The delimiter string is placed between messages so that the MQPUT2 utility or other programs that might use the file can tell where one message ends and the next message starts. The delimiter can be specified as ASCII characters (delimiter parameter) or hexadecimal characters (delimiterx parameter). The delimiter string can be from 1 to 512 bytes in length. It must be a sequence of characters that never occurs in the messages that are saved in the file. The delimiter can be a single character that never occurs in the message data (such as a binary zero if the messages contain only text data) or it can be a longer string that will not occur in normal user message data. If the sequence contains more than one character, then the sequence should contain more than one character value (e.g. it is probably better to use a sequence like "#%#" rather than "###", so that a message ending with a "#" character will be processed properly). The default delimiter sequence is "#@#@#" (without the double quotes).

The indivfiles parameter determines if all messages are stored in the same file with delimiters separating the messages or if each message is written to a separate file. If separate files are used then an incrementing number is appended to the file name for all files after the first.

If multiple messages are captured in a single file using the MQCapture program, and they are to be replayed using the MQPUT2 utility, the delimiter or delimiterx parameter specified in the

parameters file for the MQPUT2 utility should be set to match the delimiter that was used to capture the messages.

Use on systems other than Windows, Solaris, Linux and AIX

The performance utilities have been written and tested on Windows 7, various Linux distributions, Solaris and AIX platforms. Full source code, including the Microsoft Visual Studio 2017 project files, is provided for the performance utilities. The programs are written in standard ANSI C. The utilities have been ported to the AIX, Linux and Solaris environment by the author and the necessary changes have been included as conditional compilation statements. The programs have also been tested in 64-bit environments. The author believes the programs could be easily ported to other Unix environments as well, but has not done any testing to verify this.

To compile the programs in a different environment, the C source programs for the capture, driver and performance analysis programs (mqcapture.c, mqcapone.c, mqcapsub.c, mqcapsub.c, mqput2.c, mqreply.c, mqtest.c, mqtimes2.c, mqtimes3.c and mqtimes.c respectively), along with the necessary subroutines and header files (qsubs.c, qsubs.h, comsubs.c, comsubs.h, rfhsups.c and rfhsups.h, parmline.c and parmline.h, putparms.c and putparms.h, timesubs.c and timesubs.h, and int64defs.h), must be transferred to the target system.

Chapter 5. Reporting problems or suggestions

Although no official support is provided, the author is interested in hearing of any problems or suggestions for improvement for this Repository. If a bug is suspected, please create an Issue in the repository with a problem description. If possible, please attach a file with a copy of the message so that the author can reproduce the problem locally

Using the trace facility

A trace facility is included in the RFHUtil.exe and RFHUtilc.exe utilities. To enable the trace set the RFHUTIL_TRACE_FILE environment variable to a valid file name before you start the utility and then run the utility. For example you could use the following statement in a DOS command prompt and then start RFHUtil.exe or RFHUtilc.exe from the same command prompt.

```
SET RFHUTIL_TRACE_FILE=C:\Temp\RFHUtilTrace.txt
```

The trace file is written in text.

A more detailed trace is produced if the RFHUTIL_VERBOSE_TRACE variable is set to 1.

Appendix A. Sample parameters file for MQPUT2

```
[header]
* Input parameters for MQPut2 program *
*
* name of the queue and queue manager
* to write messages to
*
qname=TEMP.OUT
qmgr=QM75
*
* total number of messages to be written
* the program will stop after this number of
* messages has been written
*
msgcount=300
*
* desired qdepth for input queue
* the program will write messages until the queue depth
* is equal to the qmax parameter. When the queue depth
* reaches the desired depth, more messages will be written
* to bring the depth back to twice the desired value.
*
qdepth=15
qmax=30
*
* number of milliseconds to delay before checking
* the queue depth
*
sleeptime=50
*
* whether to dynamically adjust the sleep time or not and whether
* to write a line of output each time messages are written to the queue
* tune=0          do not dynamically adjust sleep time
* tune=1          adjust sleep time dynamically and report changes
*
tune=0
*
* maximum number of messages to write in a single unit of work
*
batchsize=20
*
* MQMD format field
*
*format= "MQSTR  "
*
* Correlation id specified in ASCII
*
*correlid="Performance Utility Test "
*
* Correlation id specified in Hex
*
*correlidx=20313233343536000037381a1b1c00
*
* Message priority
*
priority=2
*
* group id specified in ASCII
```

```

*
*groupid=group 1
*
* group id specified in hex
*
*groupidx=c1c2f1f2
*
* reply to Queue manager
*
replyqm=TEST
*
* reply to queue name
*
replyq=TEST.REMOTE
*
* message persistence
* persist no = 0
* persist yes = 1
* persist as queue def = 2
*
persist=0
*
* message type
*
* Allowed values for message type
* 1 - request
* 2 - reply
* 4 - report
* 8 - datagram
* 112 - MQE fields from MQE
* 113 - MQE fields
* 65536 to 999,999,999 - user
msgtype=1
*
* encoding for user data
* encoding=546 for PC integers, etc
* encoding=785 for host integers, etc
*
encoding=546
*
* code page for user data
* codepage=437 for PC characters
* codepage=500 for host characters
*
codepage=437
*
* the character sequence that separates messages in a file
* message delimiter parameters in ascii or hex
* only one or the other should be specified
*
* delimiter="#$%^&"
* delimiterx="0D0A"
*
* rfh usage
* rfh = N for No rfh
* rfh = A for Automatic (look for RFH at beginning of data file)
* rfh = 1 or V1 for Version 1 rfh from parameters in parm file
* rfh = 2 or V2 for Version 2 rfh from parameters in parm file
* rfh = X for special V2 rfh with fixed portion only and format=xml
*

```


RFHUTIL – Message display and test tools

- * only first character checked, except for V when second character is also checked

- *

rfh=2

- *

- * rfh data parameters - only used if rfh=1 or rfh=2

- * ignored if rfh=N, rfh=A or rfh=X

- *

- * both V1 and V2

- * encoding should be 546 for PC and 785 for host 390

RFH_CCSID=500

RFH_ENCODING=785

- * V2 only

RFH_NAME_CCSID=1208

RFH_DOMAIN=MRM

RFH_MSG_SET=DH4M6DO072001

RFH_MSG_TYPE=Customer_Root

RFH_MSG_FMT=CWF

- * V1 only

RFH_APP_GROUP=Customer_Msgs

RFH_FORMAT=Customer_Root

- *

- *

- * END OF PARAMETERS SECTION

- *

- * beginning of list of message data files

- * each line must contain a fully qualified file name

- * each file will provide the data for a separate message

- * the messages will be written in the order given below

- * only file names may follow the [filelist] entry

- *

[filelist]

msg1.dat

msg2.dat

[header]

- *

- * look for rfh with file data

rfh=auto

[filelist]

c:\v2test\rfhtest\msg3.dat

Appendix B. Sample parameters file for MQCapture

```
*
* name of the queue and queue manager
* to read messages from
*
qname=TEMP.OUT
qmgr=MQSI
*
* total number of messages to be written
* the program will stop after this number of
* messages has been written
*
MsgCount=4
*
* delimiter and delimiterx are used to
* define the message separator sequence.
*
delimiter="#@%@#"
*delimiterx="0D0A"
*
* stripafh parameter determines if RFH
* headers are to be written to the
* data file along with the message
* data or removed from the message
*
stripafh=Y
outputFileName=C:\temp\messages.dat
```

Appendix D: Using message groups and segmentation

The following exercises demonstrate the use of message groups and segmentation. They are intended to provide the reader with a basic understanding of these features and to demonstrate how to use the RFHUtil utility with these features.

Message groups are supported on MQ version 5.0 and later for all platforms except OS/390, and are supported by version 5.3 on OS/390. Segmentation is not supported on OS/390. Otherwise, segmentation is supported on the same platforms that support message groups.

Message Groups

Message groups allow a number of distinct messages to be associated, using a unique group identifier. The messages in the group are independent messages with their own data formats. There is no requirement for the messages to be of the same type, and even the code page or encoding values can be different. The messages in a group must be either all persistent or all non-persistent. The group identifier can be generated by the queue manager or can be created by the sending application. It must be unique and must be the same for all messages in a group.

While many of the features of a group could be carried in the user data area, there is one aspect that is difficult to provide in the application. The receiving application can request that all messages be available on the final target queue manager before any of the messages can be retrieved. Once the first message in the group has been read, then the subsequent messages are read in order, regardless of their physical order on the queue or the presence of other messages physically interspersed with the messages with the group. Messages in groups can also be read from the queue in physical order, but this is usually not advisable and largely defeats the purpose of using groups.

To process messages as a group, the *Logical Order* check box on the main tab should be selected. When creating a group, the Yes check box should be selected for each message within the group. The *Last* check box on the MQMD page must be selected before the last message in the group is written (the Yes check box can also be selected on the last message in a group). The All Avail check box should be checked before the first message is read. This assures that all messages are available before any of the messages are retrieved.

Message group exercise

To reinforce the discussion of message groups in the preceding section, a simple exercise is provided.

1. Create three simple text files calling the first msg1.txt, the second msg2.txt and the last one msg3.txt. Notepad or any other text editor can be used. Enter a line of distinct text for each file (e.g. "first message", "second message" and "third and last message").
2. Create a fourth file called notgroup.txt and enter a distinct line of text ("Not in Group"). A group containing the three messages will now be created, with an unrelated message after the second message.
3. Create a queue called GROUP.OUT (or use an existing empty queue).
4. Using an RFHUtil session, press the *Open File* button and select the first file (msg1.txt).
5. On the MQMD tab, select the Yes check box under *Group*.
6. Return to the Main tab. Select the *Logical Order* check box.
7. Select the queue manager. Enter GROUP.OUT for the queue name.
8. Press the *Write Q* button to write the first message.
9. On the MQMD tab, note the sequence number and group id fields, which are filled in after the message has been written. Return to the main tab.
10. Select the Read menu and then select the *Save Headers* item. This will preserve the MQ headers, including the MQMD, when the new file data is read.

11. Press the *Open File* button and select the second file (msg2.txt).
12. Select the MQMD tab and make sure the *Yes* check box under *Group* is selected.
13. Return to the Main tab. Press the *Write Q* button to write the second message.
14. Switch to the MQMD tab and note the sequence number and group id fields. Return to the main tab.
15. Start a second RFHUtil session and press the *Open File* button. Select the notgroup.txt file.
16. Select the queue manager name. Enter GROUP.OUT as the queue name.
17. Press the *Write Q* button. This will create an interspersed message in the queue that is not part of the message group. Return to the first RFHUtil session.
18. Press the *Open File* button and select the third file (msg3.txt).
19. On the MQMD tab, select the *Last* check box under *Group*.
20. Return to the Main tab. Press the *Write Q* button to write the last message.
21. Press the *Close Q* button to close the queue and disconnect from the queue manager.
22. Using MQExplorer, examine the messages on the queue. In particular, note the group identifier and the information on the Segmentation tab for each message in the group, and the logical order of all the messages in the queue

The messages will now be read back using a browse operation. They will be read first with the *Logical Order* and *All Avail* check boxes selected and the queue will then be browsed a second time without the *Logical Order* box selected.

1. Using an RFHUtil session, enter GROUP.OUT as the name of the queue and select the corresponding queue manager. Make sure that the *Logical Order* and *All Avail* check boxes are selected on the main tab.
2. Press the *Start Browse* button to read the first message in the queue. Since a browse operation is being used, the messages will not be removed from the queue after they are read. Switch to the Data tab and examine the data.
3. Press the *Browse Next* button to view the rest of the messages on the queue. The messages in the group should be seen in logical order, before any messages that are not part of the group, regardless of the physical order of the remaining messages on the queue.
4. When all the messages on the queue have been viewed, press the *Browse Next* button one more time to end the browse operation.
5. Return to the Main tab and remove the check from the *Logical Order* check box.
6. Press the *Start Browse* button.
7. Return to the Data tab to examine the data in the first message on the queue.
8. Press the *Browse Next* button to view the rest of the messages on the queue, in physical order rather than logical order. The extra message that was written to the queue in the middle of the group should now be read before the last message in the group.
9. Once the messages in the queue have been examined, they can be removed with the *Purge Q* button on the main tab.
10. It can also be interesting to repeat the loading of the group of messages into the queue from the previous exercise, but browse the queue using an additional RFHUtil session after each message is written. The effects of the various options on the order of retrieval can be easily seen.

Message Segmentation

Message segmentation is similar to message groups, in that a group of distinct physical messages are associated with each other to form a logical collection. However, in the case of segmentation, the messages are considered to form a single logical data area (single message) rather than a group of distinct but related messages. In the case of segmentation, an offset is maintained for each successive physical message, with each message having the same non-null group identifier and message sequence number. The offset of the second message matches the end of the data portion of the first message, and each successive message increases the offset by the size of the user data in the preceding message.

The *Complete Message* option on an MQGET will automatically re-assemble the segments into a single message. Segments do not have to be the same size.

Segmentation can be combined with message groups. Each individual logical message in a message group can consist of one or more message segments (physical messages).

The *complete message* option on the main tab can be used to retrieve a segmented message as a single logical message. If this option is not selected, then a single individual segment will be read each time an MQGET is issued. This option will also ensure that all message segments have arrived at the destination queue manager before any of the message segments become visible.

MQ headers such as the RFH2 header cannot be split between multiple segments.

Automatic (Queue manager) segmentation

There are two distinct ways to invoke message segmentation. Messages larger than the maximum size allowed are automatically split into as many distinct physical messages as required so that each segmented message is no larger than the maximum allowed. A maximum definition can be specified as a property of an individual queue or for the queue manager itself. Message segmentation can also be controlled by the application, as demonstrated in the next section. Messages may also be split when they are received by a receiver channel.

The following exercise will demonstrate automatic segmentation of messages by the queue manager.

1. Using MQExplorer, select the queues line under the desired local queue manager and click the right mouse button. Select New->Local Queue. Enter SEGMENT.OUT as the name of the queue and select the extended tab. Change the maximum message length parameter to 1024. Press Ok to create the queue.
2. Locate a file that is approximately 2500-3000 bytes long. The actual length of the file and the data in the file is not important. It is better if the file contains text data that can be recognized but this is not essential. If such a file cannot be easily located, then create a file using notepad (enter some text and then use the copy and paste functions of notepad to rapidly create a file of sufficient size).
3. Using an RFHUtil session, press the *Open File* button and select the file from the previous step.
4. Enter SEGMENT.OUT as the name of the queue and select the appropriate queue manager.
5. Press the *Write Q* button. An error message (MQ reason code 2030) should be generated indicating that the message is too long to write to the specified queue.
6. Select the MQMD tab and then the *Segmentation allowed* option.
7. Return to the main tab and press the *Write Q* button again. This time, the message display should indicate that a message was sent, and the queue depth should increase by the number of physical messages necessary to contain the segmented message.
8. Using MQExplorer, examine the SEGMENT.OUT queue. Display each message in the queue, and look at the Identifiers, Segmentation and Data tabs. There should be more than one physical message, with all except the last message containing 1024 bytes and the last message containing the remaining data. Note the offsets on the Segmentation tab and the Group Identifier on the Identifiers tab.

The messages we created will now be read back, both as individual physical messages and then as a single complete message.

1. Return to the RFHUtil session. Leave the queue manager and queue names set as before. Press the *Browse Q* button. The first physical message (segment) in the queue will now be read.

2. Note the data length, and examine the data and MQMD tabs. The offset will be zero, since the first physical message was read. To see all the messages, start a browse operation using the *Start Browse* and *Browse Next* buttons.
3. On the main tab, select the *Complete msg* option. Press the *Browse Q* button. This time, the entire logical message (all segments) will be read as if they were a single message. The only indication of segmentation will be the segment flags and group identifier on the MQMD tab.
4. When done examining the messages on the queue, press the *Purge Q* button to delete all messages on the queue.

When messages are segmented automatically by the queue manager, the length of all messages is generally a multiple of 16 that is less than or equal to the maximum message size allowed. Individual MQ headers must be contained within a single segment.

Application controlled segmentation

In addition to automatic segmentation by a queue manager, applications can also split a single logical message into more than one physical message. In the case of application segmentation, multiple MQ put operations must be used, and the segmentation flags must be set in the MQMD.

The following exercise will demonstrate application-controlled segmentation of a message.

1. Three files will be created to demonstrate application-controlled segmentation. The files are provided in the samples directory in the ih03 zip file or they can be created as described. The files can be created with notepad and should be named seg1.txt, seg2.txt and seg3.txt. Take a complete sentence or phrase and break it into three parts. Enter the first part as the contents of the first file, the second part as the contents of the second file and the last part as the contents of the third file.
2. Start an RFHUtil session and press the *Open File* button. Select the first file (seg1.txt).
3. Enter SEGMENT.OUT as the name of the queue and select the corresponding queue manager.
4. Select the MQMD tab. Select the *Segment Yes* check box. Make sure that the offset field is set to zero. Make sure the *Segment Last* check box is NOT selected.
5. Return to the main tab and press the *Write Q* button.
6. Select the Read menu and then select the Save RFH menu option. This will preserve the current MQMD when a file is read.
7. Press the *Open File* button and select the second file (seg2.txt).
8. On the MQMD tab, change the offset field to the length of the first file.
9. Return to the Main tab, Press the *Write Q* button.
10. Press the *Open File* button and select the third file (seg3.txt).
11. Select the MQMD tab. Select the *Segment Last* check box (it does not matter if the *Segment Yes* check box is also selected). Add the length of the second file to the offset field.
12. Select the Main tab. Press the *Write Q* button to write the last segment of the message.
13. Examine the contents of the SEGMENT.OUT queue using MQExplorer. In particular, display each message in the queue and observe the Identifiers, Segmentation and Data tabs.
14. The segments can now be read back as a single message and then individually using the same RFHUtil session. On the main tab, select the *Complete msg* check box. Press the *Browse Q* button. The entire logical message should have been read in as if it were a single message.
15. On the main tab, deselect the *Complete msg* check box. Press the *Start Browse* button. The physical message with the first segment should have been read. Examine the MQMD and data tabs.
16. On the main or data tab, press the *Browse Next* button. Examine the MQMD and data tabs.

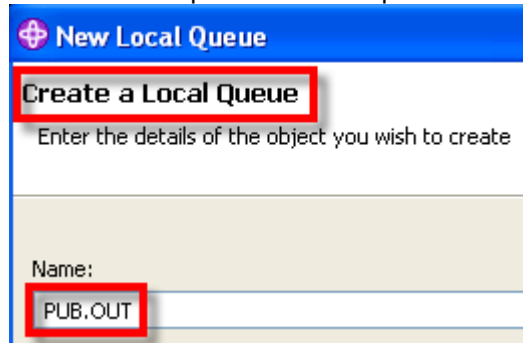
17. Repeat the previous step to view the last message segment.

Appendix E: Publish and Subscribe using MQ V7

Simple publish and subscribe

The following exercises show how to do simple publish and subscribe using publish and subscribe functions including in MQ Version 7.0. The exercises use a combination of RFHUtil and MQ Explorer. The queue manager used for these exercises must be at version 7.0 or later. In the case of client connections the client must also be version 7.0 or later.

1. Start MQExplorer.
2. Create a local queue to use for published messages. Call the queue PUB.OUT.



3. Start two RFHUtil sessions. If you are using an MQ Client connection instead of a local queue manager, then start two RFHUtilc sessions instead. One session will be used to publish messages and the other to receive the messages.
4. Select one of the RFHUtil sessions. Select the PS tab.
5. Select the queue manager that RFHUtil should connect to. This should be the same queue manager where the PUB.OUT queue was defined.
6. Enter PUB.OUT as the name of the queue. This is the queue that will receive messages that are published as a result of the subscription.
7. Enter STOCK/IBM/# as the topic. The “#” (hash) character is a multilevel wild card. This subscription should match any topics that begin with STOCK/IBM/.
8. Enter IBMSTOCK as the subscription name. The subscription name must be unique.
9. Press the Subscribe button.
10. A message indicating the subscription has been created should appear.
11. Return to the MQ Explorer session.
12. Select Subscriptions in the Navigator pane.
13. The new IBMSTOCK subscription should appear (you may need to press the refresh button).
14. Scroll to the right and examine the destination name. This is the name of the queue that will receive published messages.
15. Scroll further to the right. Examine the durable column. This subscription is not durable. It will be automatically removed when the subscribing application (in this case RFHUtil) closes the subscription or disconnects from the queue manager.
16. A message will now be published that should match the IBMSTOCK subscription.
17. Create a simple text file using notepad. This file is provided in the samples directory in the ih03 zip file. This file will be used as the data part of the published message. In this example, we will be publishing a sample stock price for IBM. The text represents a price for IBM in an XML format, with the price assumed to be in US Dollars and cents. Save the file as ibmstock (notepad will add an extension of txt).

```
<IBM><Price>00010000</Price></IBM>
```
18. Using MQExplorer, check the current depth of the PUB.OUT queue. There should not be any messages in this queue.
19. Select the second RFHUtil session.
20. Press the Open file button.
21. Navigate to a directory where you saved the file created above. Select the file and press the Open button. The contents of the file will be read.

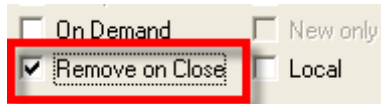
22. Select the PS tab.
23. Select the queue manager that RFHUtil should connect to.
24. Enter STOCK/IBM/PRICE as the topic.
25. Press the Publish button.
26. Return to the MQ Explorer session.
27. Select Queues in the Navigator panel. The current depth of the PUB.OUT queue should now be one. It may be necessary to press the refresh button.
28. Return to the second (publishing) RFHUtil session. Press the Publish button a second time.
29. Return to the MQ Explorer session.
30. Select Queues in the Navigator panel. The current depth of the PUB.OUT queue should now be two. It may be necessary to press the refresh button.
31. Return to the first (subscribing) RFHUtil session.
32. Press the Get Msg button.
33. The first message should have been read from the PUB.OUT queue. The queue depth of the PUB.OUT queue should now be one.
34. The contents of the message should appear in the data tab.
35. Select the MQMD tab. The contents of the MQMD should be displayed, including the correlation ID.
36. Select the Usr Prop tab. The MQ topic string of the published message should appear as a user property.
37. Return to the PS tab.
38. Press the Get Msg button two more times. The second press should generate a no messages in queue (2033) error.
39. Press the Close Sub button. The subscription is closed. Since the subscription was non-durable it will be removed.
40. Return to MQ Explorer.
41. Select Subscriptions in the navigator pane. The IBMSTOCK subscription should have been removed. Again you may have to press the refresh button.
42. Return to the second RFHUtil (publishing) session. Press the Publish button.
43. This time there is no subscription. The depth of the PUB.OUT queue should remain at zero.

Durable subscriptions

This exercise will create a durable subscription. The subscribing application will then terminate and be restarted. It will then connect to the existing subscription and receive messages, including messages published while the subscribing application was not running. This exercise assumes that the previous exercise has been done.

1. Select the first RFHUtil session. The queue manager, queue, subscription name and topic string should be filled in.
2. Select the Durable Sub check box.
3. Press the Subscribe button.
4. A message indicating the subscription has been created should appear.
5. Return to the MQ Explorer session and select the subscriptions tab (refresh if necessary.)
6. The IBMSTOCK subscription should be visible. Scroll to the right and examine the durable column. The subscription will remain in force even if the subscribing application disconnects from the queue manager and terminates. This is possible because the subscription uses a queue. In fact another application could resume the subscription instead of the original application. The subscription name is used to resume subscriptions.
7. Return to the second RFHUtil (publishing) session.
8. Press the Publish button once.
9. Return to the MQ Explorer session.
10. Select Queues in the navigator pane.
11. The depth of the PUB.OUT queue should be one.
12. Return to the first RFHUtil (subscribe) session.
13. Press the Get Msg button to read the message.

14. Press the Close Sub button to close the subscription.
15. Exit the first RFHUtil (subscribe session).
16. Return to the second (publishing) RFHUtil session. Press the Publish button.
17. Return to the MQ Explorer session. The current depth of the PUB.OUT queue should be one and the open input count should be zero. The depth should increase by one each time the Publish button is pushed on the publishing RFHUtil session, even though the subscribing application is not running.
18. Start a new RFHUtil session.
19. Select the PS tab.
20. The queue manager to connect to should already be set.
21. Press the Get Sub Names button.
22. Use the drop down menu to select the IBMSTOCK subscription. The queue name should be filled in automatically.
23. Press the Resume button (not subscribe) to resume the existing subscription.
24. Press the Get Msg button to retrieve the messages that were published.
25. Return to the second (publishing) RFHUtil session and press the publish button.
26. Return to the first (subscribing) RFHUtil session and press the Get Msg button to receive the published message.
27. Select the remove on close check box.



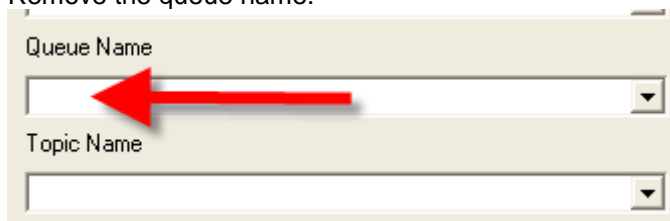
A screenshot of a checkbox labeled 'Remove on Close' which is checked. It is highlighted with a red rectangular box. Other checkboxes like 'On Demand', 'New only', and 'Local' are visible but not selected.

28. Press the Close Sub button. This time the durable subscription will be removed, since the remove option is selected when the subscription is closed.
29. Return to the second (publishing) RFHUtil session.
30. Press the Publish button. The current depth of the PUB.OUT queue should remain at zero and the subscription should no longer be visible.

Managed Subscriptions

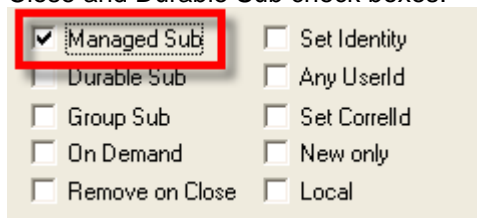
This section will demonstrate the use of managed subscriptions. It is assumed that the previous sections have been performed before this section. The RFHUtil and MQ Explorer sessions will be reused and the existing settings will be assumed.

1. Select the first RFHUtil session. The queue manager, queue and subscription names as well as the topic string should be filled in.
2. Remove the queue name.



A screenshot of a form with two dropdown menus. The top one is labeled 'Queue Name' and the bottom one is labeled 'Topic Name'. A red arrow points to the 'Queue Name' dropdown menu.

3. Select the Managed Sub check box. Remove the selection from the Remove on Close and Durable Sub check boxes.



A screenshot of a form with several checkboxes. The 'Managed Sub' checkbox is checked and highlighted with a red rectangular box. Other checkboxes include 'Durable Sub', 'Group Sub', 'On Demand', 'Remove on Close', 'Set Identity', 'Any UserId', 'Set CorrelId', 'New only', and 'Local'.


4. Make sure the topic is set to STOCK/IBM/# and the subscription name is set to IBMSTOCK.
5. Press the Subscribe button.
6. A message should indicate that the subscription has been created.

7. The name of a temporary dynamic queue should appear in the queue name field. Since the queue is created dynamically the properties are taken from the model queue (SYSTEM.NDURABLE.MODEL.QUEUE). If properties such as the maximum queue depth need to be controlled then the properties of the model queue should be changed or a predefined queue should be used rather than a managed subscription.
8. Select the MQ Explorer session.
9. Select Subscriptions in the navigator pane.
10. Locate the IBMSTOCK subscription.
11. Scroll to the right and examine the Destination Name.
12. Select the second (publishing) RFHUtil session.
13. Press the Publish button twice.
14. Return to the first (subscriber) RFHUtil session.
15. Press the Get Msg button.
16. The first message should be read.
17. Press the Get Msg button two more times. The second message should be read and then a no more messages available (2033) error message should be generated.
18. Return to the first (subscribing) RFHUtil session.
19. Press the Close Sub button.
20. The subscription will be removed since it is not durable. There is no need for the Remove on Close option, although it is allowed.

Retained publications

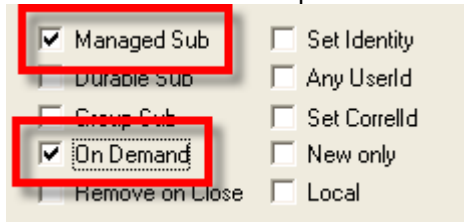
This section will demonstrate the use of a retained publication. When a message is published without the retained option the message is sent to all current subscribers and then the input message is discarded. If the retained option is specified the message is first published as usual and the message is also stored in the queue manager. If a new subscription is received the retained message can be sent immediately to the new subscriber. An existing subscriber can also request the message by sending a request publication. The retained publication is replaced by the next message published to the same topic that specifies the retain option.

1. Select the second (publishing) RFHUtil session.
2. Select the Retained Pub check box.



A screenshot of a software interface showing a checkbox labeled 'Retained Pub' which is checked. To its right are two other unchecked checkboxes: 'Not own pubs' and 'Suppress Reply To'. Below 'Retained Pub' is a disabled checkbox labeled 'Is Retained'.

3. Press the Publish button.
4. Select the first (subscribing) RFHUtil session.
5. Select the On Demand option. Leave the Managed check box selected.



A screenshot of a software interface showing two checkboxes selected: 'Managed Sub' and 'On Demand'. To the right of 'Managed Sub' are 'Set Identity' and 'Any UserId'. To the right of 'On Demand' are 'Set CorrelId' and 'New only'. Below 'Managed Sub' is a disabled checkbox 'Durable Sub'. Below 'On Demand' is a disabled checkbox 'Remove on Close'. There is also a disabled checkbox 'Group Sub' between 'Durable Sub' and 'Set CorrelId'.

6. It is not necessary to use a managed subscription to access retained publications. The options are unrelated. This exercise could equally well use a provided (unmanaged) subscription and specify an existing queue such as PUB.OUT.
7. Press the Subscribe button.
8. Press the Get Msg button. An error message should be generated indicating that there are no messages available.
9. Press the Req Pub (request publication) button. This will request that the most recent retained publication be sent to the queue specified by the subscription. It is worth noting that since a subscription can contain wildcards a publication request can in fact generate more than one message if it matches more than one topic. In this case a single request publication request can result in more than one message being

published. For this reason the topic should probably not include wildcards. It is always possible to distinguish between multiple messages by using the MQTopicName message property to see the actual topic the message was published on.

10. Press the Get Msg button to read the published message.
11. The message is received even though no message was published after the subscription was made.
12. Select the Usr Prop tab.
13. Examine the message properties. The MQIsRetained property should have a value of TRUE.
14. Return to the PS tab. The Is Retained check box should be set, indicating this was a retained rather than a new publication.
15. Press the Close Sub button to close the subscription. The subscription will be removed since it is not a durable subscription.
16. This exercise can be repeated but without selecting the On Demand check box before making the subscription. In this case the retained publication will be sent when the subscription is made. It can be retrieved immediately by pressing the Get Msg button.

Using an administered node

Administered nodes are specific locations within a publish and subscribe topic tree that have been predefined for administrative purposes (usually for security). In this exercise an administered node will be defined using MQ Explorer and will then be used to subscribe to messages using the defined node. Administered nodes are given a name of up to 48 characters.

This exercise assumes that the previous exercises have been done. It reuses the existing MQ Explorer and RFHUtil sessions.

1. Switch to the MQ Explorer session.
2. Select Topics in the navigator pane.
3. Press the right mouse button and select New->Topic from the menu.
4. Enter IBM as the topic name.
5. Press the Next button to continue.
6. Enter STOCK/IBM as the topic string.
7. Press the Finish button to create the administered node.
8. Press OK to acknowledge the message box.
9. The new topic name should now be visible as a topic.
10. Return to the first (subscribing) RFHUtil session.
11. Enter PUB.OUT as the queue name.
12. Press the Get Topic Names button.
13. Use the drop down list to select IBM as the topic name.
14. Enter # as the topic. Remember that the topic is relative to the administered node specified in the topic name.
15. Enter IBMSTOCK as the subscription name.
16. Clear any of the check boxes that are checked.

<input type="checkbox"/> Managed Sub	<input type="checkbox"/> Set Identity
<input type="checkbox"/> Durable Sub	<input type="checkbox"/> Any UserId
<input type="checkbox"/> Group Sub	<input type="checkbox"/> Set CorrelId
<input type="checkbox"/> On Demand	<input type="checkbox"/> New only
<input type="checkbox"/> Remove on Close	<input type="checkbox"/> Local

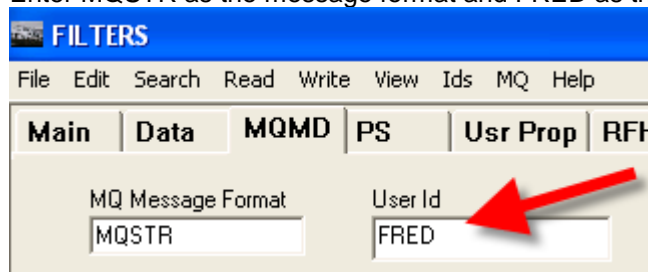
17. Press the Subscribe button.
18. A message should indicate that the subscription was created.
19. Examine the topic and topic name fields. The topic should now contain the full topic string and the topic name field should have been cleared.

20. Switch to the second (publishing) RFHUtil session.
21. The topic should be STOCK/IBM/PRICE. The topic name field should be empty.
22. Press the Publish button to publish a message.
23. Return to the first (subscribing) RFHUtil session.
24. Press the Get Msg button.
25. A message should be read from the subscriber queue.
26. Press the Get Msg button until a no messages in queue error is returned.
27. Press the Close Sub button to remove the subscription.
28. The IBM administered node can now be deleted using MQ Explorer. Select Topics and then the IBM item. Press the right mouse button and select delete from the menu. Press the Yes button to confirm the deletion and then the OK button to acknowledge the message box.

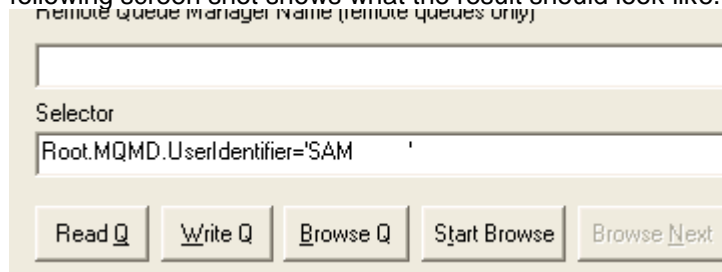
Appendix F: Using Message Filters

This section includes some simple exercises to use the message filter capability that was introduced in version 7 of MQ. Two examples will be used. The first example will create two messages in a queue with different user identifiers in the MQMD. A selector will then be used to select messages for a particular user identifier. The second example will create two messages with a user property and then select which message is read based on the value of the user property.

1. Start MQ Explorer if it is not running.
2. Create a local queue called FILTERS (the name of the queue is not important).
3. Start an RFHUtil session.
4. Press the Open File button.
5. Navigate to a suitable small file and press the open button on the file dialog. The ibmsale1.txt file in the samples directory shipped in the ih03.zip file is a good one to use but the actual data is not important.
6. Select the Set Iden Context check box.
7. Select the MQMD tab.
8. Enter MQSTR as the message format and FRED as the user id.

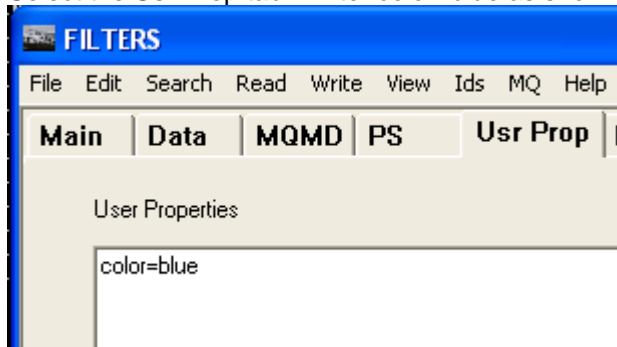


9. Return to the main tab and select the appropriate queue manager and then the FILTERS queue.
10. Press the Write Q button to write a message to the queue.
11. Return to the MQMD tab.
12. Change the user id to SAM.
13. Return to the main tab and press the Write Q button to create a second message.
14. Return to the MQ explorer. Select the FILTERS queue.
15. Press the right mouse button and select browse messages.
16. There should be a message with a user identifier of FRED followed by one with a user identifier of SAM.
17. Return to the RFHUtil session.
18. Enter Root.MQMD.UserIdentifier='SAM ' into the selector field exactly as shown. SAM should be followed by nine blanks since the user identifier field in the MQMD is 12 characters long. The name and blanks must be enclosed in single quotes. The following screen shot shows what the result should look like.

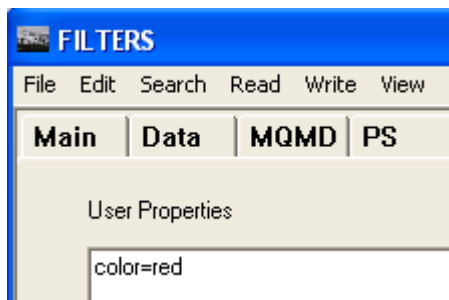


19. Press the Start Browse button. The message with a user identifier of SAM should be read, although this is actually the second message in the queue.
20. Confirm that the expected message was read by switching to the MQMD tab and checking that the user id is SAM. If no messages are found it is likely that there is an error in the selector string. Be especially careful to enclose the name followed by nine blanks in single quotations, as shown.

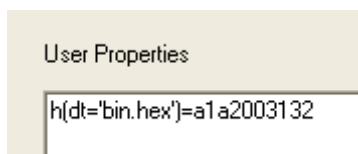
21. Press the Browse Next button. Since there is only one message with a user identifier of SAM there should be no more messages in the queue that match the selection criteria and the browse operation should end.
22. Two messages will now be added to the queue with a user property called color. Select the Usr Prop tab. Enter color=blue as shown below.



23. Select the Main tab. Select the Set Iden Context check box.
24. Select the MQMD tab.
25. Set the User Id field to FRED.
26. Select the main tab.
27. Press the Write Q button to create a message.
28. A second message will now be created with a color property of red.
29. Select the Usr Prop tab.
30. Change the color value to red.

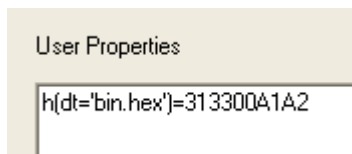


31. Select the MQMD tab.
32. Change the user id field to SAM.
33. Select the main tab.
34. Press the Write Q button to create a message.
35. There should now be four messages in the FILTERS queue. A selector will now be used to browse particular messages.
36. Enter color='red' as the text in the selector field, as shown below.
37. Press the Browse Q button to read the first message that meets the selection criteria. The message that was read is actually the fourth message in the queue but it is the first message that has a property named color with a value of red.
38. The next part of the exercise will show how to work with byte strings, integers and properties that do not exist.
39. Return to the Usr Prop tab.
40. Enter the following user property h(dt='bin.hex')=a1a2003132 as shown.

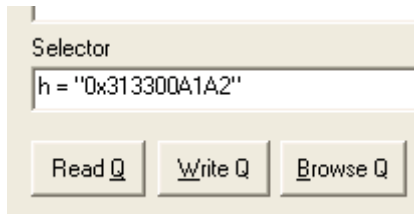


41. Return to the main tab.
42. Press the Write Q button to create a message.
43. Return to the Usr Prop tab.

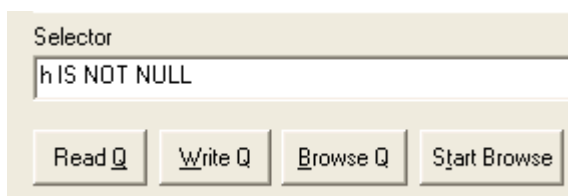
44. Enter the following user property `h(dt='bin.hex')=313300A1A2` as shown.



45. Return to the main tab.
 46. Press the Write Q button to create a message.
 47. Enter the following selection string into the Selector. `H = "0x313300A1A2"`



48. Press the Start Browse button.
 49. The last message that was written should be retrieved. This can be confirmed by examining the Usr Prop tab.
 50. On the main tab, press the Browse Next button. No more messages should match the selection criteria and the browse should end.
 51. The next part of the exercise will retrieve the two messages with a user property named "h".
 52. Change the selector to the following: `h IS NOT NULL`



53. Press the Start Browse button.
 54. This time the first of the two most recent messages should be read. This can be confirmed by examining the Usr Prop tab.
 55. On the main tab press the browse next button. The second message should be read. Again this can be confirmed by examining the usr props tab.
 56. Press the browse next button again. There should not be any more messages that meet the selection criteria and the browse operation should end.
 57. Return to the Usr Prop tab.
 58. Enter the following user property. `I4(dt="i4")=25`
 59. Return to the main tab.
 60. Press the Write Q button to create a message.
 61. Switch to the Usr Prop tab.
 62. Enter the following user property. `I4(dt="i4")=100`
 63. Return to the main tab.
 64. Press the Write Q button to create a second message.
 65. Change the selector to the following: `I4 > 50`
 66. Press the Start Browse button.
 67. Examine the Usr Prop tab. The second message with an I4 property with a value of 100 should have been read.
 68. Press the Browse Next button. There should not be any more messages that meet the selection criteria and the browse operation should end.
 69. This concludes the message filters exercise.

End of Document