MO71
WebSphere MQ for Windows GUI Administrator User Guide

Version 7.0.2
3rd July 2011

Paul Clarke
WebSphere MQ Development

paulg_clarke@uk.ibm.com

MP211
IBM UK Laboratories Ltd
Hursley
Winchester
Hampshire
SO21 2JN
United Kingdom

Take Note!

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

Twentieth Edition, July 2011

This edition applies to Version 7.0.2 of *WebSphere MQ for Windows GUI Administrator* and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

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

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

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you. References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

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

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

The information contained in this document has not 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 each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

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

WebSphere MQ
IBM
AIX
OS/400
MVS
z/OS

The following terms are trademarks of the Microsoft Corporation in the United States and/or other countries:

Windows 95, 98, Me
Windows NT, 2000, XP

Table of Contents

Figures

Preface

This SupportPac was originally written as a monitoring program for the 1996 Olympic Games in Atlanta. The requirement was for an OS/2 PM program which would actively monitor a number of remote Queue Managers and give visual and audible feedback if a problem was detected. At the time I was interested in gaining experience of auto (Programmable Command Format) messages. Since the program already had the basic structure required I decided to add command support to it allowing you to issue virtually any MQSC command against any number of Queue Managers in a dialog.

The program was written largely in my own time and, as such, many facilities I would like to add have not yet been written. In addition it has not had the benefit of any formal testing. I can therefore not vouch for the correctness of the program and would welcome any comments, both positive and negative. Either way I hope you find the program useful and a big usability improvement over MQSC.

My thanks go to Morag Hughson for considerable assistance with this manual, for numerous usability suggestions, for keeping me sane when nothing ever seems to go right, as well as tirelessly testing many buggy versions of the program! She was also the writer of the API Exerciser which I'm sure many of you will find useful.

My thanks to Jan Fluitsma for providing "*Appendix C:Displaying National Language Characters*".

My thanks to Stefan Raabe for his many suggestions and testing of the programmable filters.

I would also like to express my gratitude to the many people over the years who have written to me to express their thanks for the program and make suggestions. It makes all the difference in the world to know that people are using the program and finding it useful. When I first wrote the program there were few alternatives to MQSC for Queue Manager Administration but now there are many packages available. Although this is a spare time activity I shall try to ensure that updates are made to this program as long as people tell me it is useful to them.

Main changes from previous version

The introduction of the Queue Manager list in many of the list dialogs has had a profound effect on the code. The data model has had to change significantly which is one of the main reasons the time between releases has been so long. While every effort has been made to try and ensure that previous function is working correctly it would be surprising if some problems didn't leak out. I would recommend that users keep their previous version of MO71 handy so that they can revert to it if a bug is found. Needless to say, please report any incompatibilities to me if they are found.

The main changes since version 7.0.1 are:-

1. **MQ 7.0.1 Support**
   There weren't a great many changes to the MQ commands and objects in MQ 7.0.1.

2. **Multiple Queue Manager list support**
   Most objects can now shows objects for multiple Queue Managers.
   New filter functions 'qm()' and 'qmloc()' added for this feature.

3. **List dialog areas are now sizeable.**
   This allows the section of the list dialog which displays the fields at the top of the dialog to be resized.

4. **New Preference options**
   - Show object name in entry field rather than a dropdown list.
   - Confirm on update if object name has changed since refresh.
   - Show QM list by default.
   - Don't show red crosses.
   - Default message display range.
   - Reposition list on sort.
   - Select the first item of list, after refresh, if no current selection.
   - Auto update of lists if actions may have changed the contents
   - Auto update of browse queue screen if actions may have changed contents

5. **Updates to MO71 authorities**
   Changes to 'cluster queue manager' and 'compare' authorities.

6. **Message selector field added to message browse window.**
   This allows the dialog to use the power of message selectors added in MQ V7.0.

7. **New set() filter function**
   Effectively allows the user to change the default field values for a dialog.

Chapter 1. WebSphere MQ GUI Administrator

This document describes the functions available in the SupportPac.

Overview

The main window displays a list of Queue Managers. Typically, this list would be the local Queue Manager and a number of other 'remote' Queue Managers that the machine has access to via WebSphere MQ channels. The user can then perform the following functions against the Queue Managers in the list:-

1. **Issue commands**
   By selecting a Queue Manager in the list and invoking one of the Command menu options the user can issue commands against that Queue Manager. Note that the Queue Manager can be any platform that has a command server (including z/OS).
   Commands may also be entered in MQSC format from the MQSC window described later.

2. **Filtering**
   When dealing with a list of objects as the result of an issued command, it can be useful to filter out some of these objects to reduce the size of the list displayed.

3. **Queue Manipulation**
   The queue being accessed can be either on the local Queue Manager or a remote Queue Manager provided a program like 'MQMONNTP' is running on the remote Queue Manager. Connecting to the Queue Manger as a client will allow queue manipulation on any remote Queue Manager since, as far as MQMONNTP is concerned, the Queue Manager is local.

4. **Communication**
   By selecting Talk from the menu, a dialog is shown which allows the user to type a text message that can be sent to the target Queue Manager. The receiving Queue Manager must be running the monitoring program, in which a dialog shows the text that the original Queue Manager sent.

5. **Monitoring Queue Managers**
   With monitoring enabled, the program will periodically send 'monitor' messages to any enabled Queue Manager. You can monitor a Queue Manager that the application is directly connected to but it is more useful to monitor a remote Queue Manager. This essentially involves sending a message to a named queue ‘the monitor queue’ on the remote machine and waiting for the message to be returned.

6. **Monitoring Queue Manager objects**
   WebSphere MQ provides quite a number of 'event' messages which are generated when certain key events occur during the life of the Queue Manager. For example, an event message can be generated when the depth of a queue reaches a certain threshold value. WebSphere MQ itself does not act upon these messages, they are purely generated for some monitoring tool to receive and perform an appropriate action. The appropriate action will vary depending on the installation, the object involved and the event type generated.
   MO71 can be configured to read these event messages from the event queues and can be configured to perform a variety of actions such as issue commands or write to a log file

7. **Try out MQI verbs**
   The API Exerciser allows the user to try out some of the major MQI verbs (MQCONN, MQOPEN, MQPUT, MQGET etc) and see their behaviour.

Installation

Create a directory, say MQMON, and copy the following files into it:-

- MQMONNTP.EXE

Note that the directory where you place the .EXE program is the default location where the program will store its configuration file MQMON.CFG and the object definition file MQMON.DFN. You generally need not worry about these files but you should be aware that they are written and therefore the program needs write access to the directory. Alternatively you can use the '-f' parameter to the program to give the directory name the program should use for these files.

Once the MQMONNTP program is in the path it can be run just by typing 'MQMONNTP' on the command line. No further set-up at this point is required. The program will dynamically load either the WebSphere MQ client or WebSphere MQ Queue Manager libraries as required. Note that to successfully connect to a Queue Manager at least one of these WebSphere MQ products must be installed. The windows WebSphere MQ client is available for free download at:-

http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24009961&loc=en_US&cs=utf-8&lang=en2

Users will probably find it more convenient to set up a desktop icon for the Administrator program.

Chapter 2. Getting Started

This section will explain how to run the Administrator program and provide an example to add the first Queue Manager to be administered.

Running the Administrator

To run the Administrator enter the following:-

- MQMONNTP
- Or click on the desktop icon created at installation time

The program does not require any parameters but you can pass a Queue Manager location in on the command line if you wish. See *"Parameters"* on page 97 for more information.

Once started, the user will be presented with a main application window with a number of menu options. If this is the first time the program has been run then the main window itself will be empty and entries for each Queue Manager must be added. By default the application receives replies from the Queue Manager via the standard model queue SYSTEM.DEFAULT.MODEL.QUEUE but it may be preferable to define an explicit queue for MQMON to use. See *"Reply Queue Details*" on page 96 for further details.

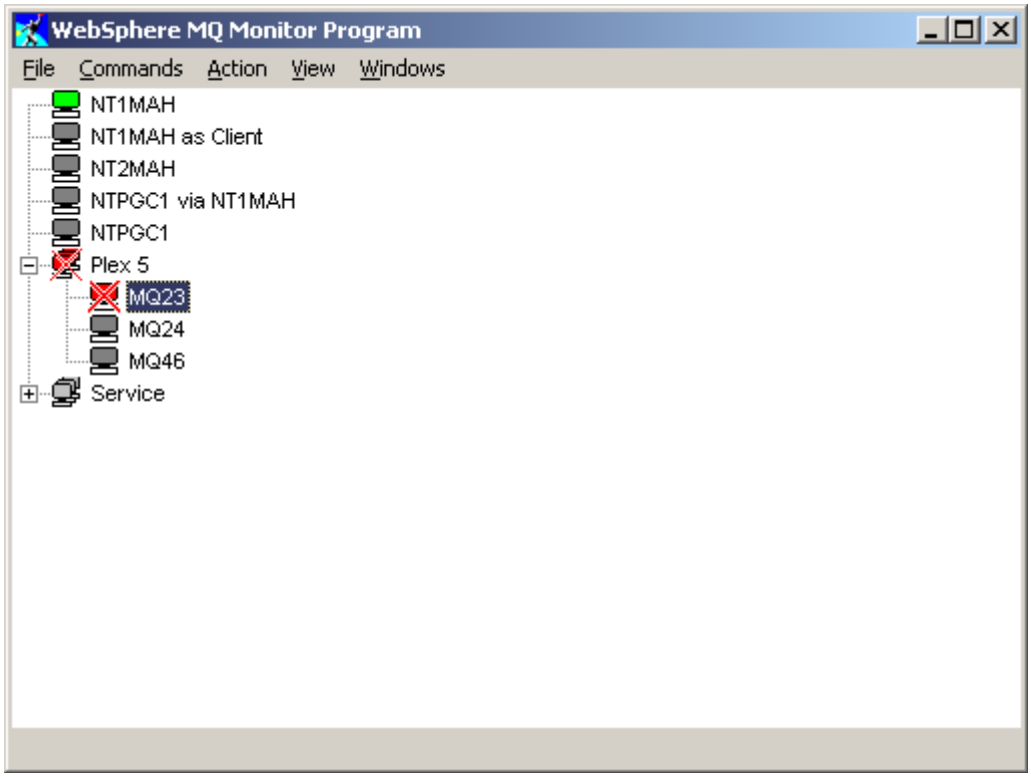


Figure 1: Main Window

To successfully issue commands then the Queue Manager and its respective command server (e.g. strmqcsv <QMNAME>) must be running.

Adding a Queue Manager Entry

To add an entry for a new Queue Manager in the Administrator use *'Add Location'* in the File menu. This will bring up a location dialog.

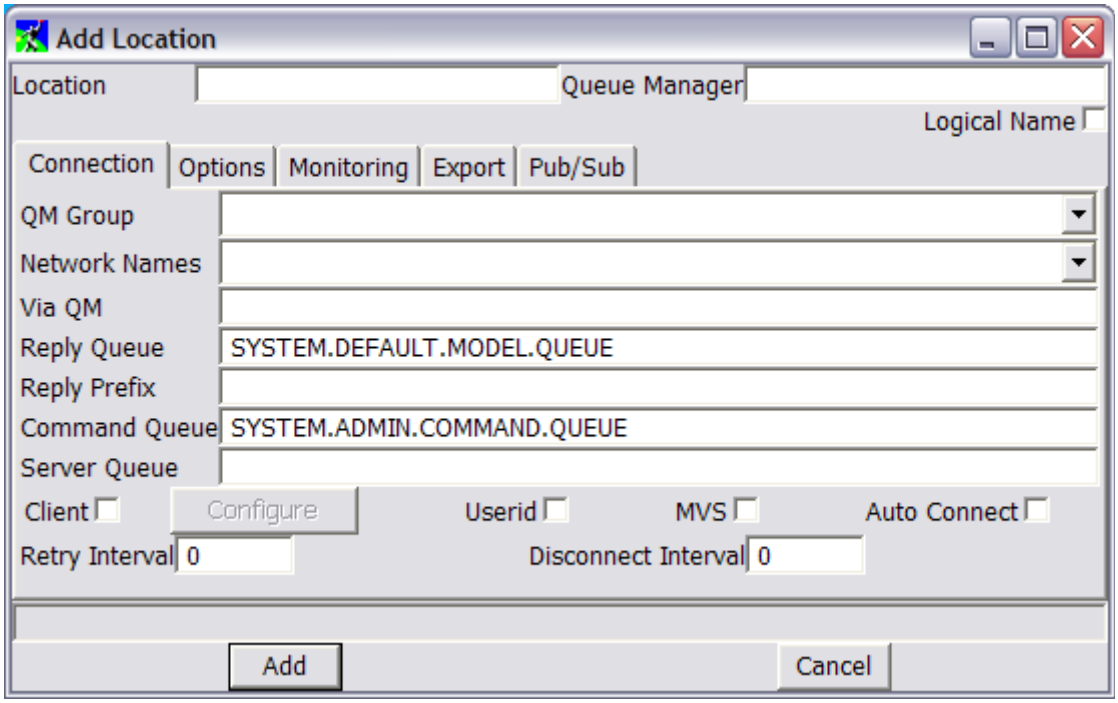


Figure 2: Add Location Dialog

You can administer a local Queue Manager by simply entering its name in the Queue Manager field and entering a text description of the location in the location field. To administer a remote Queue Manager you can either connect via a client or via another Queue Manager. The following examples will show the additional configuration needed to use either of these methods.

Connecting via a Client

To directly connect to the Queue Manager via a client, select the Client check box and click on the Configure button to bring up a dialog to define the client connection channel. In this dialog fill in at least the Channel Name and the Connection Name.

Connecting via a different Queue Manager

Connecting via a different Queue Manager can be useful in certain circumstances when a client connection to the required Queue Manager is either not feasible or possible.

For example suppose we wish to remotely administer Queue Manager QM2 but wish all our messages to go to QM2 via QM1. To configure this set up you must perform two steps.

1. If not already defined we must first define a location for the 'via' Queue Manager QM1. We can either define it as a local Queue Manager (if it's on windows) or as a client.

2. Next we define the location we actually want to administer, QM2. In the 'Via QM' field of the location dialog we put the name of the Queue Manager where we are routing messages via. In this case it is QM1. We must also remove the Reply Queue since it is not needed for connection by this method.

   Note that in addition to defining these locations you must have WebSphere MQ communications (for example Sender/Receiver channels) in both directions between these Queue Managers. If the transmission queues are not called the same as the queue manager they are going to then it will also be necessary to define Queue Manager Alias definitions in accordance with WebSphere MQ remote configuration. Please refer to the

WebSphere MQ Intercommunication manual for details in configuring remote messaging between two Queue Managers.

For details of all the fields in the location dialog see “Location Dialogs” on page 88.

Chapter 3. Issuing Commands

MQMON provides a graphical interface to the most common commands one can issue to a Queue Manager. This interface has the advantage that the user does not need to remember the command syntax and generally reduces typing by keeping relevant fields on the dialog. To issue a command against one of the Queue Manager's via the dialogs the user must select the Queue Manager from the list and then select the item from the Commands menu that most closely approximates the command required[1].

The commands presented depend on the version of the Queue Manager at the remote location. For example clustering commands are only presented if the remote Queue Manager supports clustering.

The 'Commands' menu can be accessed either from the menu bar, or by pressing mouse button 2, after selecting the appropriate Queue Manager. Pop-up menus are available in most dialogs by pressing mouse button 2.

The menu contains commands such as :-

- a single object dialog for any Queue Manager object type
- a list dialog for any Queue Manager object type
- a single dialog and list dialog for Channel status
- queue statistics
- queue manipulation (See the section below on Queue Manipulation)
- cluster commands (where appropriate)

There are two styles of dialog for each of the main Queue Manager object types, for example, Queue and Queue List. The former style is designed to allow operations on a single object, on input of the object name. The list dialogs allow the display of, and operation on, multiple objects of that type.

The Command window

There are two types of command window, the list and single object display. Both window types allow the user to issue a command by pressing the command button if displayed at the bottom of the screen or by selecting the appropriate menu option from the pop-up menu that is displayed when you press mouse button 2. Note that only the common operations are presented as buttons, there are often more options available in the pop-up menu.

The command windows are split into the following sections :-

Fields

These fields display the current value of the objects attribute. The user can move the entry field to any attribute (except certain read-only attributes) and change its value. This can be done by clicking anywhere on the field or its description with the mouse, using the up/down buttons or using the TAB keys. For fields that have a choice of values the drop down list must be displayed (PF4) and then the value selected using up/down keys.

- If the field is a name of another object, for example a transmission queue in a sender channel dialog then double clicking on the field will cause a dialog of that object to be displayed. This allows a quick method of displaying all the objects associated with a primary object.
- It is possible for an object dialog to contain more than one key field value. This is done by selecting more than one entry of a list dialog and then opening the definition or by ending the key field name with a '+' sign to indicate that you want to type further key values. When an object does have more than one key field specified then any command issued will be applied to all the objects in the list. Note that wherever possible the commands available will reflect the objects selected or displayed but it may be that the command selected is not applicable to all the objects. In this case an error message will be displayed. Because a single command could now generate many responses, one for each object, the response window at the bottom of the dialog is a drop

[1]An alternative way of entering commands to a Queue Manager is to use the MQSC window. This is available from the main menu and is described later. The MQSC window has the advantage that **any** command supported by the Queue Manager may be issued but it has the disadvantage that the output of the command is not formatted or post-processed and the user must know the exact syntax of the command.

down list which allows all the responses to be checked.

For example to start (or stop) many channels at once. First display a list of channels. Select the required channels using standard controls e.g. <CTRL> + Mouse Click for individual selections, <SHIFT> + Mouse Click for selecting a range. Click Mouse Button 2 then select the command required. The command will be issued against all the selected objects in sequence and the command response is given in the drop down list at the bottom of the dialog.

- For list windows the fields are used to limit the amount of data requested from the command server. For example, on the Queue List specifying a Queue Name of "SYS*" will only request queues starting with the string "SYS".
  If space is limited the field area can be reduced by dragging the bar at the base of the fields or they can be hidden completely by selecting the *'Hide Fields'* pop-up menu option or clicking the 'Show Fields' tool in the toolbar.

Status window

The status window is used to relay information about the current status of the dialog, including any error messages. This is a drop down list so a history of responses can be seen. Note that the history will be abbreviated. Not all messages are written to the history and the same entry will not be entered twice in a row.

Action buttons

Some of the commonly used options are displayed as action buttons to the user. These are equivalent to the options on the pop-up menu.
If space is limited the buttons can be hidden by selecting the *'Hide Buttons'* pop-up menu option or the 'Hide Buttons' tool in the toolbar.

Command Keys

Most of the commands can be issued using a keystroke such as *Alt*-r. If a command has a key shortcut the letter will be underlined in the context menu.

List Windows

- **Object List**
  A list box containing the returned objects. Double clicking on an entry will cause a dialog for that object to be shown.
- **Queue Manager List**
  By default a list dialog is associated with just a single Queue Manager. However, most of the lists can be associated with more than one Queue Manager. This allows, for example, a single dialog to show the queues for multiple queue managers. Or perhaps to monitor the depth of the dead letter queue on a group of Queue Managers. There are two ways to associate multiple Queue Managers with a list.

  1. **Select 'Show Queue Manager List' or click the 'Show Queue Manager List' tool from the toolbar**
     A pane showing the available Queue Managers will appear on the left side of the list dialog. Each Queue Manager can be (de)selected by clicking on the selection button. Immediately to the right of this button is a database symbol which indicates whether data is available for this queue manager or whether the data is refreshing. By clicking on this button a request is sent to just this single queue manager for a refresh of the data.
  2. **Using the qm() and qmloc() filter functions**
     For example, a filter of *qm("NT*")* would associate all queue managers starting with the characters **NT** with the dialog. Care should be used with these functions. A filter of *qm("*")* will associate all Queue Managers with the dialog. This can cause a large number of requests for data to be sent when the dialog is refreshed and the dialog can also use a significant amount of storage if there are a large number of Queue Managers and Queues.

- **Split List**
  If required the list window can be split into two separate panes by selecting the 'split list' option from the context menu or clicking the ‘Split List’ tool in the toolbar. For example, the object name such as Channel Name can be kept in view in the left hand pane while the channel attributes are scrolled in the right hand pane.
- **Filter button and filter entry field**
  This entry field allows the user to perform filtering on the data returned by the command server before displaying. A filter can be added by typing a filter expression in the filter window and pressing the '*' filter button. The filters are stored in the list portion of the filter field, this allows previous filter values to be selected from the drop down list. Please see “Filtering” on page 18
  If space is limited the filter can be hidden by selecting the *‘Hide Filter”* pop-up menu option or pressing the ‘Show Filter’ tool in the toolbar.
- **List Titles**
  When a list is refreshed the titles of the various fields in the list are displayed in the list titles window. The list of attributes displayed can be chosen from the ‘*Alter List’* pop-up menu option. If a particular attribute is not returned for any object in the returned list then that column is not displayed, thereby allowing more space to be used for other fields. Clicking on the list titles window will sort the list in alternatively ascending and descending order. An arrow pointing either upwards or downwards in a column title shows which field is being used to sort by and whether it’s ascending or descending respectively.

  The edges of each column can be dragged via the mouse to re-size the columns. The user specified widths can be cancelled by selecting the *‘Reset Column Widths”* pop-up menu option or by pressing the reset width toolbar icon.

Dialog Examples

Display a list of Queues

1. Select the appropriate Queue Manager
   - Choose *'Queue List...'* from Commands menu
   - Press *Refresh* to cause the list to be shown

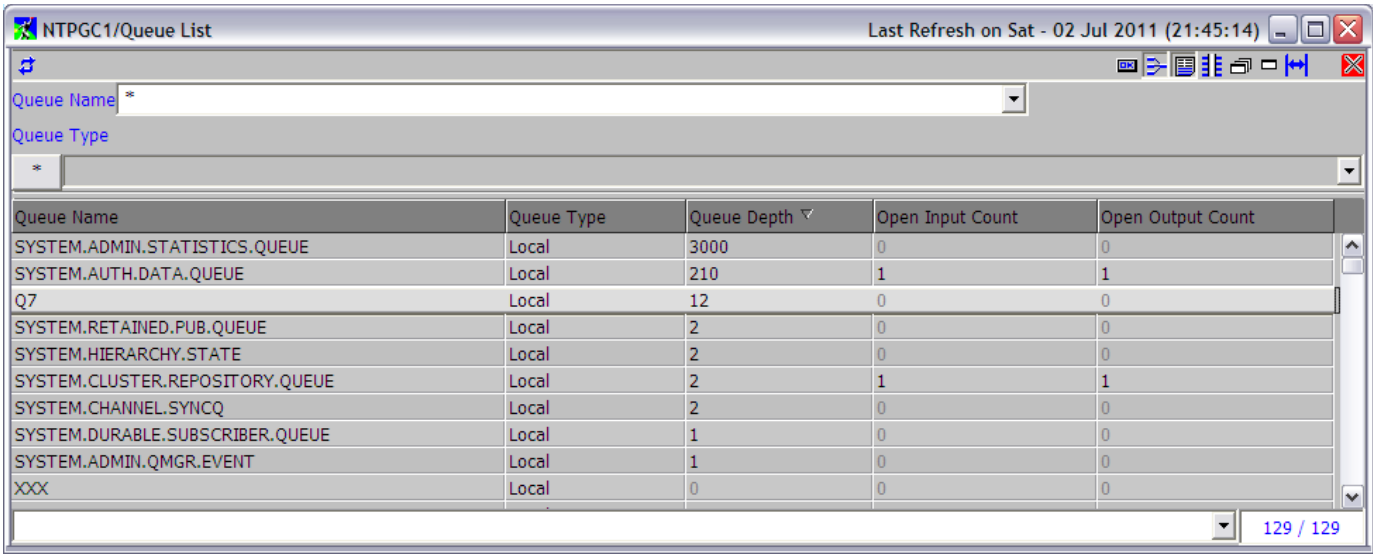


Figure 3: Queue List Dialog

Displaying Queues for multiple Queue Managers

1. Display the multi Queue Manager by clicking the 'Show Queue Manager list' tool in the toolbar or selecting the menu from the right-click pop-up menu Option.
   - Select the additional Queue Managers(s) you want to display
   - If necessary click the database icon to refresh the information

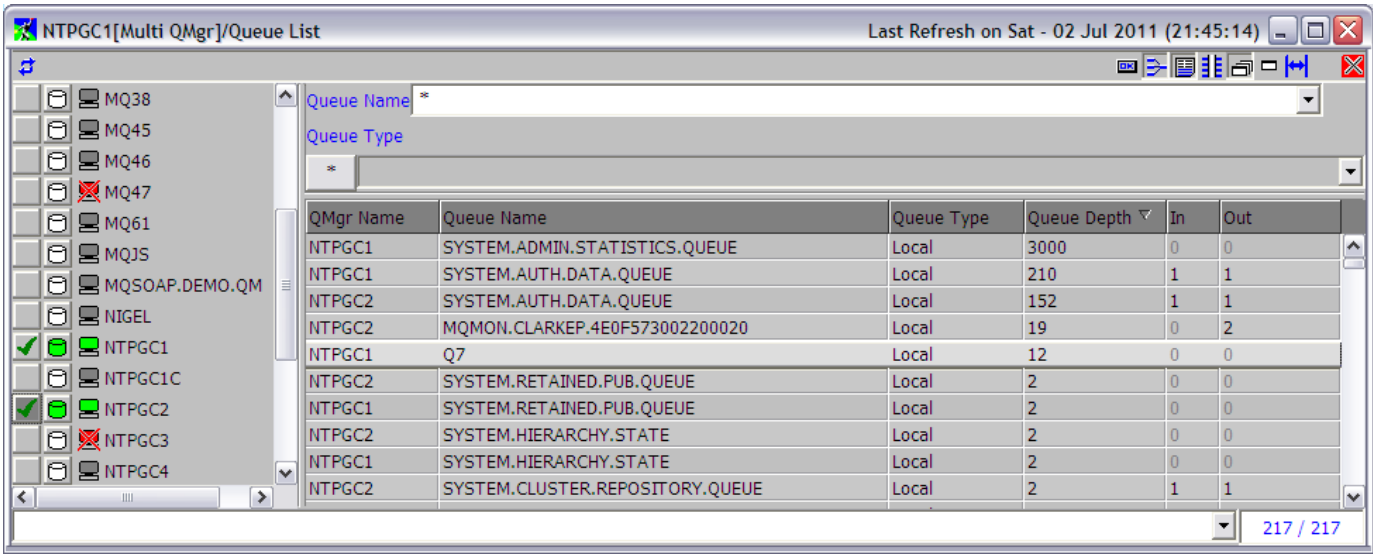


Figure 4: Multi Queue Manager List

The user should be sensible about how many Queue Managers are selected in the left-hand pane. If a very large number of Queue Managers are selected and each Queue Manager has a large number of queues then the performance of the dialog could be affected. In addition care should be taken with the 'REFESH' button since this will send a REFESH command **to each** of the Queue Managers potentially causing a lot of unnecessary message traffic.

Defining a Queue

1. Select the appropriate Queue Manager
   - Choose *'Queue...'* from Commands menu
   - Fill in the queue attributes required, for example *Queue Name*
     Any blank fields will inherit the default values
   - Press *Create* to cause the queue to be defined
   - Pressing *Refresh* will show the complete queue definition

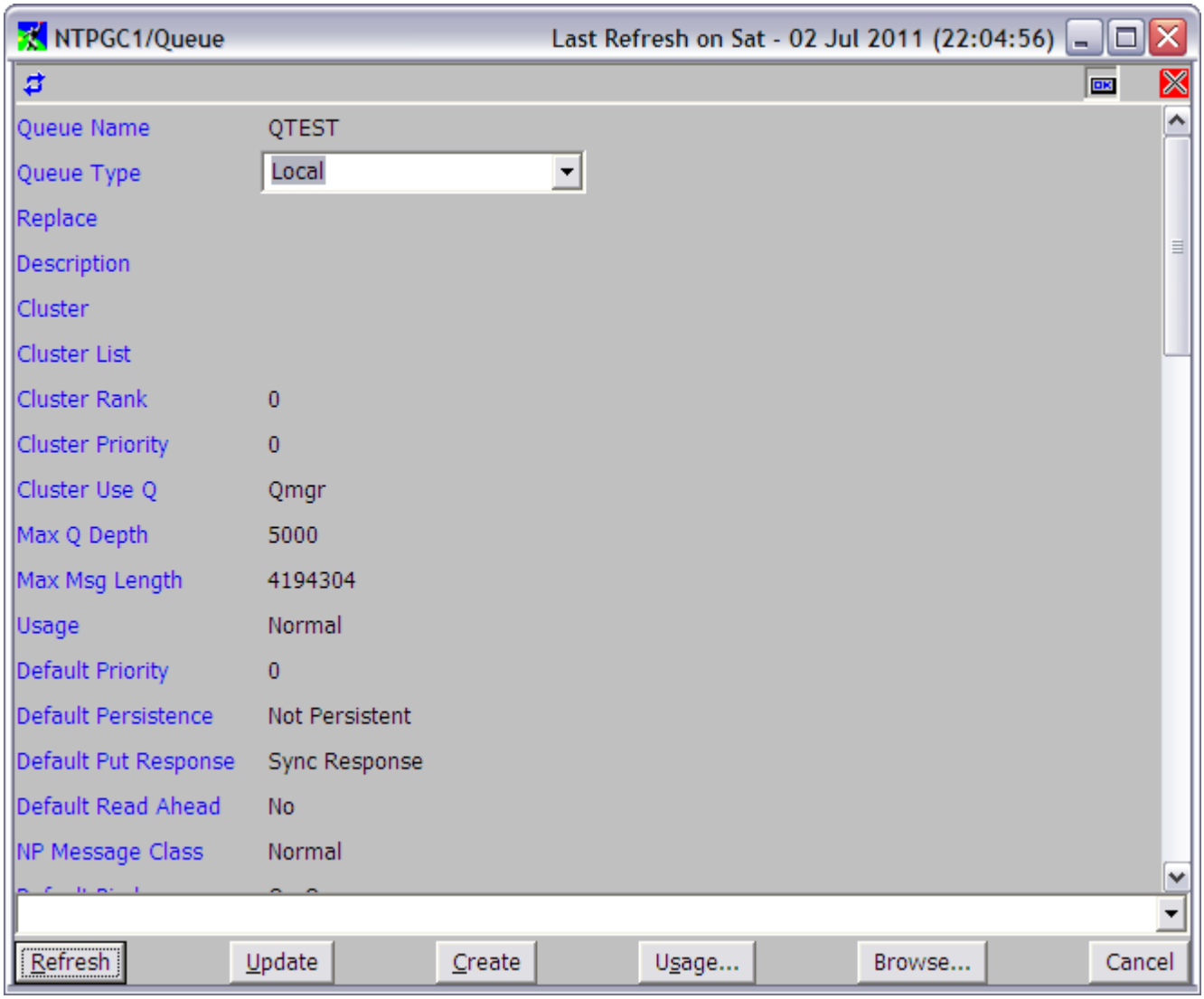


Figure 5: Queue Dialog

Defining a Queue like another Queue

1. Select the appropriate Queue Manager
   - Choose *'Queue...'* from Commands menu
   - Enter queue name to copy
   - Press *Refresh* to show complete queue definition
   - Change any queue attributes required, which must include the *Queue Name*
   - Press *Create* to cause the queue to be defined

Starting a Channel

The following examples illustrate the two different ways you could start a channel.

1. Select the appropriate Queue Manager
   - Choose *'Channel List...'* from Commands menu
   - Press *Refresh* to cause the list to be shown

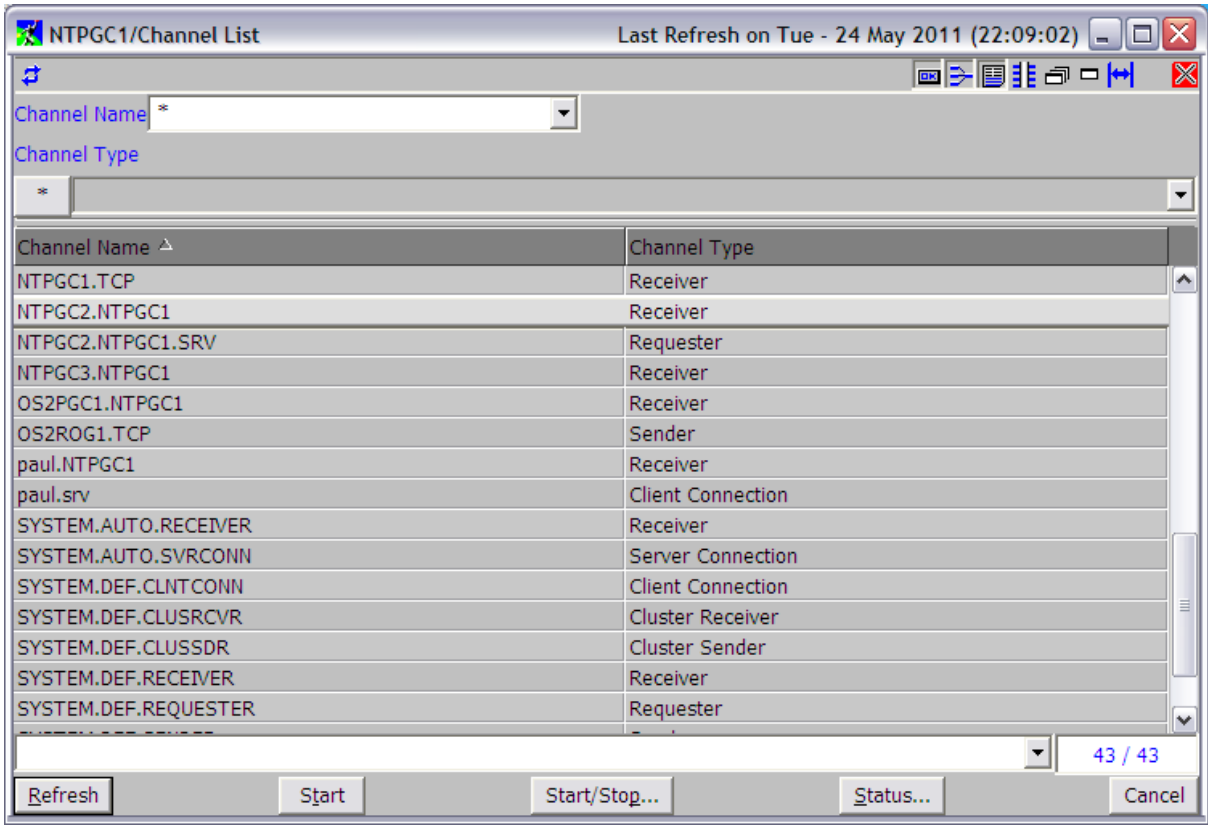


Figure 6: Channel List Dialog

   - Find the Channel in question and select it
   - Select *'Start/Stop...'* from the pop-up menu or button
   - Press *'Start'* in the Start/Stop Channel dialog

2. Select the appropriate Queue Manager
   - Choose *'Channel...'* from Commands menu
   - Type the name of the Channel to start
   - Press *Refresh* to cause the channel definition to be shown
   - Select *Start/Stop...* from the pop-up menu or button
   - Press *'Start'* in the Start/Stop Channel dialog

Clearly if there are a number of operations to be performed against objects of the same type then the list option involves less typing for the user.

The Pop-up menu

- All the commands available are contained in a pop-up menu that is displayed when mouse button 2 is pressed. Care should be taken with destructive commands such as *'Delete'* and *'Clear Queue'* if confirmation dialogs are not active (see “Preferences Dialog” on page 78 to control confirmation dialogs). This pop-up menu may also contain a *‘Print...’* option which is described in “Printing” on page 48, and an *‘Export...’* option which is described in “Exporting Definitions” on page 51.

The pop-up menu also contains an *‘Options’* sub menu. By selecting this the user is shown a number of options which change the look and behaviour of the dialog. Note that the list displayed will depend on the dialog. Pop-up menus for the message dialogs have other sub-menus in addition to this one. For details see “Queue Manipulation” on page 32.

- **Alter list**
  Each list has a predetermined set of attributes that are displayed for each object. For example the Queue list displays the Queue name, Queue type and the current depth. If required, the user can change this to a list of his/her choice in two ways. Firstly the lists can be changed globally for all locations by using the option under the *‘View/Set Default Lists’* menu on the main window. Secondly the list of attributes can be changed only for this location by selecting the *‘Alter list’* option from the pop-up menu. See “Alter list dialog” on page 15 for how to operate the dialog.
- **Split List**
  By selecting this option the object list can be split into two separate panes, each capable of displaying the entire data. This is useful when displaying a large number of attributes and the user wishes to keep one set of attributes on the screen, such as the object name, while the other attributes are scrolled from left to right.
- **Reset Column Widths**
  Selecting this options will cancel any user defined column width and the system will choose what it considers to be the best width for the column, The user can set the required column width by dragging the edge of the column title.
- **Make Filter Default**
  Objects lists can be filtered via a Boolean expression (see “Filtering” on page 18) to restrict the number of objects displayed in the list. Selecting this menu item will associate the current filter with this list type. Each time the dialog is initially displayed it will have this filter. Selecting this menu with the filter window empty will disassociate any previous default filter.
- **Initial Refresh**
  Selecting this menu item will cause the list to be automatically retrieved from the command server whenever a dialog of this type is opened. This should be used if the list filtering options are rarely used.
- **Hide System Objects**
  When selected all objects starting with “SYSTEM.” are removed from the displayed list. The object count at the bottom of the window will reflect the fact that not all objects are being displayed.
- **Update Predefined Definition**
  By selecting this option a predefined dialog definition is either created or updated depending on whether the dialog was started from a predefined dialog definition. Note that if this option is used to create a predefined definition the ‘description’ field will be blank. In this case the *Predefined Dialog List* dialog should be used to select the newly created definition and add a description.
- **Auto Refresh**
  This displays a dialog which allows you to set an auto-refresh time interval and optionally ask for the returned information to be exported to a file. For more information please see “Auto Refresh Dialog” on page 14.
- **Highlight difference from default**
  This menu option is only available on object types which have the notion of a default object. For example, queues have a default definition for each type of queue. When this option is selected attribute values which are

the same as the value in this objects corresponding default object will be displayed in a subdued colour[2]. In other words, the differences will be highlighted.

This menu will be greyed out if the default object definitions have not yet been retrieved.

- **Display difference from default**
  As above, this menu is only available on object types which have the notion of a default object. When this option is selected then only those columns which contain a difference from the default object definitions will be displayed. Note that which columns are displayed is therefore heavily dependant on which objects are displayed.
- **Highlight difference fields**
  This menu is only available on 'compare' type dialogs. When selected the fields (attributes) which are the same on both Queue Managers are displayed in the list lowlight.
- **Display only different fields**
  This menu is only available on 'compare' type dialogs. When selected only field columns which differ between the two Queue Managers are displayed. The columns displayed are therefore heavily dependant on which objects are displayed.
- **Display objects**
  This menu is only available on 'compare' type dialogs. It allows a quick way of selecting a subset of the objects based on the categories :-

Qmgr A defined only	Those objects defined only on Queue Manager A
Qmgr B defined only	Those objects defined only on Queue Manager B
Matching entries	Those objects which are defined the same on both Queue Manager A and Queue Manager B
Non-matching entries	Those objects which are not defined the same on both Queue Manager A and Queue Manager B

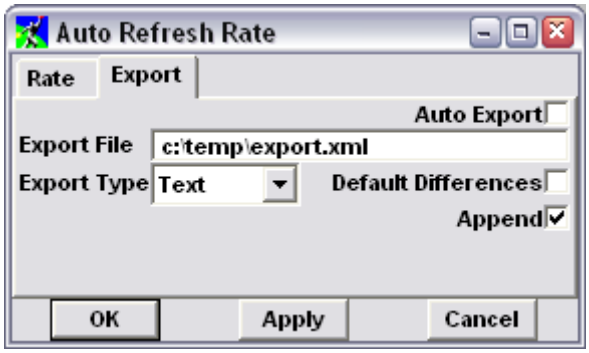
- **Hide/Show Buttons**
  This option allows the user to Hide/Show the buttons of the dialog to make best use of the available screen area.
- **Hide/Show Toolbar**
  This option allows the user to Hide/Show the toolbar. The toolbar allows quick selection of common operations.
- **Hide/Show Fields**
  This option allows the user to Hide/Show the fields of the dialog to make best use of the available screen area.
- **Hide/Show Filter**
  This option allows the user to Hide/Show the filter windows of the dialog to make best use of the available screen area.
- **Hide/Show Queue Manager List**
  This option allows the user to Hide/Show a pane of Queue Managers to the left of many of the list dialogs. This pane allows the dialog to be associated with more than just a single Queue Manager and therefore display command results from multiple locations.
- **Reset Queue Manager List**
  Selecting this option will reset the dialog back to being associate with the single Queue Manager that was selected when the dialog was started.

[2] The colour with which 'lowlight' items are displayed in a list can be set in the colour dialog by adjusting the value for 'List lowlight Foreground'.

Auto Refresh Dialog

This dialog allows you to set an auto-refresh time for a command dialog. This allows an administrator to constantly monitor, for example, the depth of the Queues or the status of Channels.

It is displayed when you select *'Options-Auto Refresh'* from any object or list dialog. You can set a different refresh time interval per command dialog type. Auto Refresh is also available on the MQSC window. The window title will display the refresh rate if auto refresh is active. Different refresh intervals can be specified for each type of dialog.



It is possible to set a very small refresh interval, for example 1 second, which is fine for particular tests on a development machine but probably not a good idea on a production system. Remember that refreshing a dialog involves message exchanges with the command server and is therefore not an insignificant cost particularly when there are a lot of objects involved in the request.

The Auto Refresh dialog also gives you the ability to have an export file generated each time the data is refreshed. This is a rather specialised use of export but can be useful when you wish to capture a series of object status history. The resultant file can then be post-processed by another program to look at the trends. As usual with export you can choose from four file formats, for post processing the formats of CSV or XML are likely to be the most useful.

Auto Refresh Export Fields

The fields related to auto refresh export and their meanings are as follows

- **Auto Export**
  Controls whether the dialog will write to the auto export file or not.
- **Export File**
  The export file format can contain character inserts which are described in "Figure 19:Export File Inserts" on page 92.
- **Export Type**
  Determines what type of export file will be generated.
- **Default Differences**
  Selecting this option will cause only the attributes which are different from the default object to be written. Note that this option only applies to objects which have the notion of a default object, other objects will have their complete definition written even if this option is selected.
- **Append**
  When selected any data will be appended to the end of the file if it exists. If not selected then any existing file will be over-written.

Alter list dialog

This dialog allows the user to change the list of attributes displayed in a list window. It is invoked either by selecting the *'View/Set Default Lists'* menu on the main window or by selecting *'Alter list'* from the pop-up menu on a list window. The first of these sets the fields 'globally' for all locations. The second invocation is only 'Local' to this location and will not affect the lists displayed by other locations.

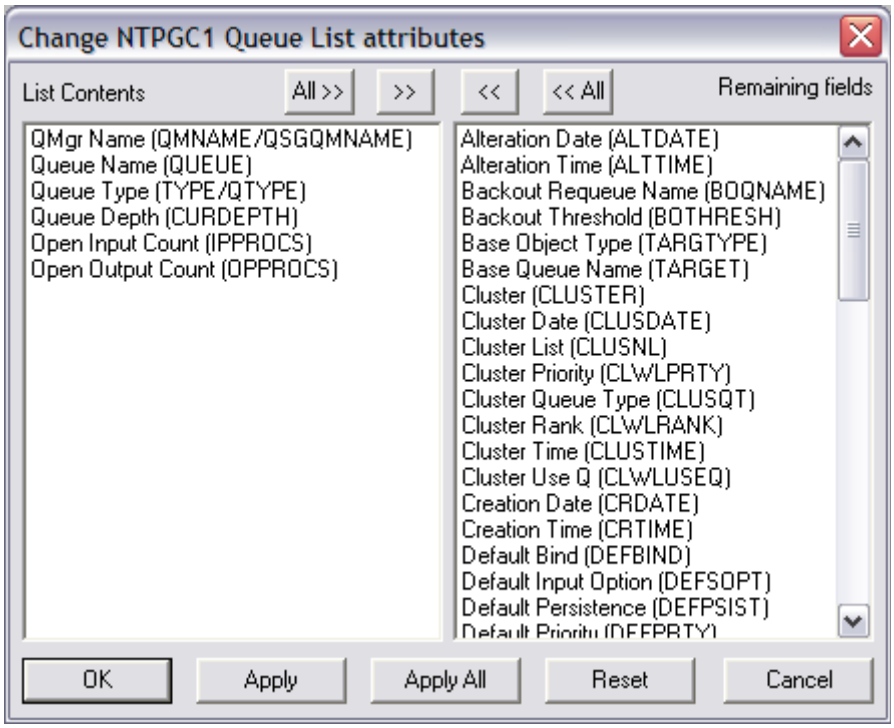


Figure 7: Alter List Dialog

The dialog displayed shows two sets of the attributes, the **List Contents** in the left hand column and the **Remaining fields** in the right hand column. Attributes are moved either by selecting them and pressing the arrow buttons or double clicking on items. To move all the attributes use the *'All >>'* and *'<< All'* buttons.

The attributes in these lists have a text description followed by an identifier in brackets. This identifier is the name, usually the MQSC name, which should be used in filter expressions (see "Filtering" on page 18 for more information).

The order of attributes can be set by selecting an item in the List Contents column on the left before moving the attributes across from the Remaining fields in the right hand column. Items will be added before the first selected field. If no items are selected, attributes will be added to the end of the list.

Pressing the *'OK'* or *'Apply'* buttons will cause the new attributes to be adopted by the dialog list. By default the first attribute will be used as the sort field for the list. However, if an item is selected in the List Contents column when the *'OK'* or *'Apply'* button is pressed then that field will be used as the sort field. This simple technique is overridden if the user uses SORT functions in the filter window or clicks on the list titles.

The *'Apply All'* button will update all instances of this type of list. For a global change this means that all locations will have their current list replaced by the new list in the dialog. For a local change the change will be 'saved' for this location. Subsequent invocations of MQMONNTP will display this new list for this location. This method replaces the *'Make List Default'* option available in previous releases.

As the number of columns is increased you may find that the complete field title is not displayed in order to safe space on the dialog. By hovering the mouse over the column title a tooltip will be displayed which will give the complete name of the field and the MQSC attribute name which can be used in filter expressions.

QMNAME or QSGQMNAME field

This field is special in that it only appears if it is appropriate to do so. The dialog allows the user to position the field within the list, or remove it completely, but to save space in the list dialog it won't always be displayed. For example,

if the dialog is associated with more than one Queue Manager then this column will display to allow the user to see which Queue Manager an object belongs to. If the dialog is only associated with the dialogs main Queue Manager then the column is unnecessary and is not displayed. It is recommended that this field always be included and be the first element in the list.

MQSC Window

The MQSC window is provided to allow the user to issue commands not provided by MQMONNTP, for example, commands like **DISPLAY DQM** on z/OS. The screen is split into three main areas, the entry field at the top where you enter an MQSC command; the central area where the command server responses are displayed; and status messages, as usual, are shown at the bottom of the screen.

To make life a little easier the entry field is a drop down list box. The last 20 commands entered are kept as a history and can be recalled either by dropping down the list box or by using the up and down arrow keys. The command history will also be saved across invocations of MQMONNTP.

As with nearly all the dialogs the *'Auto Refresh'* option is available. This allows the user to type a command on the command line and have the program execute the command every few seconds. Care should be taken when doing this, however, since the response window maintains a list of **all** the responses from the command server since the start of the dialog, therefore this should be used in conjunction with the **cls** command below.

The command entered in the command field is interpreted by the program before being passed to the command server. The following additional capabilities are allowed :-

Command	Meaning
Cls	**Clear screen command**. This command will delete all data in the main window
;	**Command separator**. Allow multiple commands to be entered on a single line Note that commands themselves should not contain this character
/	**Remove search string**. A find command with no following text will remove the highlighting of the search text.
/text	**Find text**. Issuing this command will find the next occurrence of 'text'. All instances of 'text' will be highlighted. This command stays in effect until the search string is removed by the command below
file(infile)	**Execute MQSC script file**. Any '?' at the end of the *infile* will bring up a file selection dialog for the path, if any, specified in *infile*.
file(infile,outfile)	**Execute MQSC script file and write the result to the output file**. Any '?' at the end of either the *infile* or *outfile* will bring up a file selection dialog for the path, if any, specified in the filename.
>	**Execute command and write the result to the output file**. For example *dis q(*) > myfile.txt*
>>	**Execute command and append the result to the output file**. For example *dis q(*) >> myfile.txt*

Examples

To clear the screen, display the queue and channel called ‘FRED’ and then highlight all instances of the word ‘FRED’, use the following command :-

cls;dis q(FRED);dis chl(FRED);/FRED

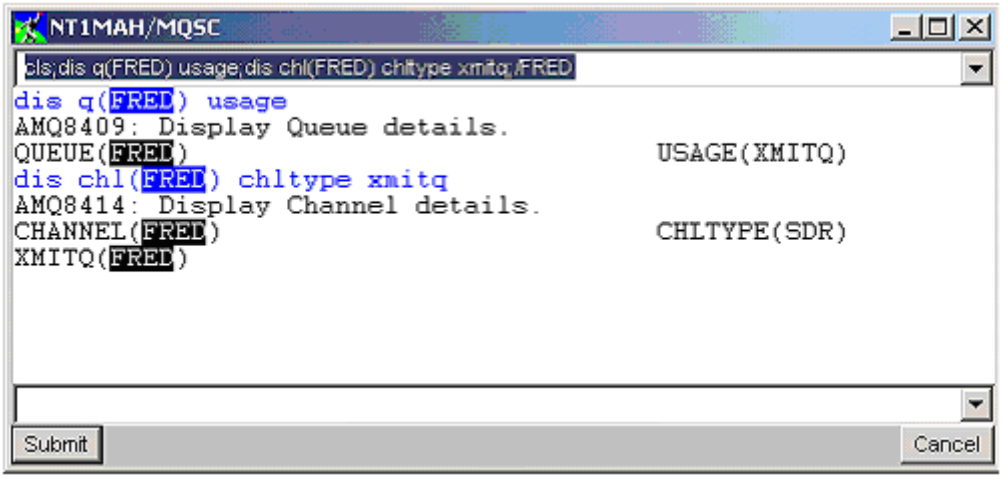


Figure 8: MQSC Window

With auto refresh enabled the following could be useful to constantly display the state of Distributed Queuing on a z/OS Queue Manager :-

cls; dis dqm

Clipboard

Two menu items are provided for copying the displayed data to the clipboard.

1. **Copy to clipboard**
   This option will copy just what is displayed on the window to the clipboard
2. **Copy all to clipboard**
   This option will copy all of the data currently in the MQSC window history to the clipboard

Chapter 4. Filtering

On Queue Managers where there are many definitions of the same object type it can be difficult to 'see the wood from the trees'. What is needed is a way of limiting the objects displayed to the user, i.e. filtering. Filtering allows the user to provide a Boolean expression to determine which objects are displayed.

Note that, where possible, filtering is performed against the data cached from the command server. In other words, typing in an expression and pressing the '*' filter button will not necessarily cause a request to the command server. If all the information needed to satisfy the expression is already stored locally the filter will be performed on that information. Consequently the display of data is much faster. However, as a consequence the data displayed will not necessarily be the latest up-to-date data and users should periodically hit the *'Refresh'* button to download the latest information from the server. If the filter expression contains attributes that are not cached then a request for more information will be sent to the command server.

It is possible, in fact common, for a filter expression to contain field names that are not applicable to all entries in the list. If this is the case then the value of the field for the particular entry will be FALSE.

At the bottom right hand corner of the screen are two numbers. The first number gives the number of objects displayed in the list, and the second number gives the number of objects returned from the queue manager. If filtering is not active, these two numbers will be the same, however, with filtering there may be fewer objects in the list than were returned from the queue manager.

Filter Expressions

The expression rules are as follows:-

- **The expression is free-format**
  e.g. "2+4" and "3 + 4" are both acceptable
- **Statements**
  A filter can consist of multiple statements joined together using a ';' character. The value of the statements is the last statement in the list.
  For example the filter **status("Hello World");0** has the value 0.
- **Numbers**
  Numbers can be entered as:
  - Integer 12,1234
  - Real 1.3,12345.5
  - Hex 0xFAB,0x10
- **Strings**
  Strings can be any sequence of characters between ' or " characters. Special characters can be included in the string using the '\<value>' convention.
  - \\ Backslash (Note should be used when specifying filenames with back slashes in them. For example use **c:\\temp\\file.dat** not **c:\temp\file.dat**
  - **\n** New line
  - **\"** Double quote
  - \' Single quote
  - **\a** Alert
  - **\b** Back space
  - **\f** Form feed
  - **\r** Carriage return
  - **\t** Tab
  - **\v** Vertical tab

For example :-

- "SYS.*"
- 'ABC'
- "This string has a \" in it"

- **Constants**
  - Colours (used in the fg() and bg() functions)
    white, black, blue, red, pink, green, cyan, yellow, brown, gray, dblue, dred, dpink, dgreen, dgray
    The constants starting with 'd' are darker versions of the colour.
  - Booleans
    true, false
  - Console priorities (used in the csl() function)
    info, low, medium, high

List Variables

Any attribute name of the object being displayed may be used in the expression. The attribute name is normally the MQSC name of the attribute and only enough of the name to ensure uniqueness need be specified. To see the identifier that should be used look in the *'Alter List'* dialog where the complete list of attributes and their identifiers are given. Alternatively if it's a displayed field you can hover the mouse over the column title and a tooltip window will be displayed giving the full name of the field and the MQSC attribute name which should be used in filter expressions.

It is not necessary to only use fields from the currently displayed list. Note that the identifier name is not always the MQSC name since some fields don't have one. For example, Trigger Control doesn't have an actual MQSC attribute name it only has a value TRIGGER or NOTRIGGER. In this case I have introduced a value of TRIGCTL to identify this field. The same is true of fields like Harden Get Backout and Shareability.

It is often useful to be able to temporarily see a value in the list without going through the process of actually changing the list items using *'Alter List'* described earlier. In this case you can follow the variable name with a '#' (hash) character. This will cause that field to be displayed if it's not already in the list. The columns will be displayed after the defined columns for the list in the order they were written in the expression. As soon as a filter is entered without this field followed by a hash then the field will be removed from the display

For example, for a Queue List the following are valid variables :-

Queue	Queue Name
Qtype	Queue type
Usage	Usage value
Usage#	Usage Value and display Usage in the list display
Curdepth	Current Queue depth
Cur	Current Queue depth
Cu	Current Queue depth

System Variables

System variables are predefined variables which are initialized to a value before being used in the filter.

- **_first** An integer which is set to 1 (or true) each time the filter is matched against a set of entries. This integer allows you to do some initial processing. Note that the filter should set the value of **_first** back to 0

once initial processing has completed with the statement.
For example, **_first := 0;**

- **_last** An integer which is set to 1 (or true) only for the last entry in the list. The **_last** system variable is most useful for reporting results.
  For example, **if (_last) status("All done now");**
- **_time** An integer which contains the current time. This can be useful for calculating the different in time between successive invocations of the filter or for performing different processing in the filter depending on the day of the week or time of day.
- **_index** The index of the current item in the list starting at 0.
- **_items** The number of items in the list
- **_qmname** The name of the Queue Manager
- **_locname** The name for this location
- **_scan** The scan instance. This field can be used to discover how many times this current data has been scanned. This can be used in conjunction with **_rescan** to examine all data in scan 0 before deciding what to return in scan 1.
- **_rescan** Whether another scan should be performed. A filter can set this value to true to force a re-scan of the current entries. This can be useful to collate information about the data in the first scan, then during the re-scan the filter can use this collated data to make decisions about the data.

User Variables

User variables begin with the character '@'. They may be used to store information either between items in a list or between separate invocations of the list filter. Different data types of user variables are allowable by following the '@' by an optional character.

- **@x := 1** Declares x as an integer variable
- **@@x := 3.2** Declares x as a real or float variable
- **@$x := "Hello"** Declares x as a string variable
  String variables may only be a maximum of 50 characters.

As shown in the example above, assignments are made using the '**:=**' operator. Note the use of the colon character to distinguish the assignment from a comparison.

For example, **@x = 3** is comparing **x** with the value 3 whereas **@x := 3** is assigning **x** the value 3.

Sometimes it may be useful to have a variable for each object in the list. In these cases you can use the insert %o in the variable name. The object name will be substituted in the variable name before it is used. For example, **@depth %o := cur** will assign the queue depth to a unique variable. Note that the uniqueness of the variable is only as unique as the object name itself. For example, when used in a queue list it is possible for there to be more than one entry with the same queue name. A local queue and a cluster queue can appear in the same list. Similarly it is possible to have multiple entries in a channel status list each with the same channel name.

Operators & Flow control

Normal operator operation and precedence apply :-

+, -, *, /	Addition, Subtraction, Multiplication, Division
mod	Modulus
=,>,>=,<,<=,<>,!=	Boolean comparisons
&,|,!	Boolean AND, OR and NOT
&&,||	Bitwise AND and OR[3]
==	String wild card matching The second operand is specified as a string containing the following wild card characters: - '*****' Matches any number of characters (including 0) '**?**' Matches exactly 1 character.

Note that the '**+**' operator is used to concatenate strings.

- **Conditional branches**
  It is possible to take conditional branches using the following construct:-
  if (expression) {statements} [else {statements}]
- **Coercion**
  If two different operand types are involved in a sub expression the type of one or more of the operands will be changed. Most notably if strings are used in expressions other than comparisons then the string length is what is used. For example, the filter **"desc"** is TRUE only if the object has a non-blank description. So for example the filters **queue == "???"** and **queue = 3** will both display the list of queues with only three character names. As another example, **SORT(queue)** will sort the list in alphabetic queue name order but **SORT(queue+1)**, since we're forcing the queue to be treated as a number, will display the list in order of the length of their name. A similar effect is achieved by using **SORT(+queue)**.
- **Invoking other filters**
  If a filter has been given a name then it can be invoked from another filter using the expression **$<filter_name>**. Note that this means that filter names can not contain blank characters.

Functions

- **alarm(Exp)**
  Causes the application to alarm, always returns TRUE. If 'Exp' evaluates to TRUE then the application will issue an alarm BEEP every few seconds provided alarms are switched on for the application.
- **all(Exp1,Exp2,Exp3,.....)**
  Takes 1 or more parameters. The **all** function is used for temporarily adding a new column to a list display. For example a filter of **'all(maxdepth#)'** will add max depth to the list of columns displayed but will still display all queue definitions regardless of whether maxdepth is applicable to that type of queue.
- **any(Exp1,Exp2,Exp3,.....)**
  Takes 1 or more parameters. The **any** function is useful for displaying fields which can contain any value. For example a filter of **'trigctl#'** will display a column for Trigger Control but will only display those items for which the value of Trigger Control is non-zero, it is equivalent to 'trigctl# <> 0'. There are times when you want to see the value displayed regardless of what it is. A filter of **'any(trigctl#)'** will display those queue definitions for which Trigger Control is a valid attribute regardless of what that value is.
- **beep(Exp)**
  Causes the application to beep, always returns TRUE. If 'Exp' evaluates to TRUE then the application will issue a BEEP.
- **bg(Exp, Colour)**
  Sets background colour, always returns TRUE

[3] Please note that the bitwise and logical operators are the other way round from languages such as 'C'. This is due to the fact that most of the time the single operator, logical version, is required.

If 'Exp' evaluates to TRUE then it sets the background colour of the object in the list
The colour can be specified using the colour constant or can be an expression.

- **bgcell(Exp,Colour,Cellname)**
  Sets background colour of a single cell, always returns TRUE
  If 'Exp' evaluates to TRUE then it sets the background colour of just the specified cell in the list. For example, bgcell(cur, yellow, queue) will set the background colour of just the Queue Name for any queues which have messages on them.
  The colour can be specified using the colour constant or can be an expression.

- **csl(Exp,Priority,Error Identifier)**
  This function allows the user to write items to the console window.

  The expression parameter controls whether this error is shown or removed from the console. If the expression evaluates to TRUE the error will be shown on the console, if it evaluates to FALSE it will be removed from the console.

  The priority parameter sets the priority value to be used for the entry in the console. Constant values such as **info, low, medium** and **high** may be used for this parameter.

  The Error Identifier parameter can either be a WMQ error code or a string error message. This error identifier, or its string equivalent, will be shown on the console window.

  For example, suppose we have a queue list and wish to have the queues which have more than 100 messages on them displayed on the console we could use a filter of :-

  csl(curdepth > 100,medium,"Loaded queues")

- **day(time)**
  Returns day of the month represented by the time parameter. Returned value from 1->31.

- **date$([time [,format]])**
  Returns the date as a formatted string.
  This function takes 0, 1 or 2 parameters. If no parameters are given the current time is returned in the default format. If just the time parameter is passed then that time is returned in the default format. If both a time and format is given then the returned value is the time in a formatted string according to the following values for the format string.

H	Two digit hour (24 hour clock)
HH	Hour (24 hour clock)
h	Two digit hour (12 hour clock)
hh	Hour (12 hour clock)
M	Two digit minutes
S	Two digit seconds
d	Two digit day of month
dd	Day of month including suffix eg.1st, 2nd, 3rd …
j	Julian day of year (zero based)
J	Julian day of year (one based)
m	Three character month name eg.Jan,Feb,Mar….
mm	Two digit month
mmm	Full month name eg. January, February,March…
P	AM/PM
p	am/pm

y	Four digit year
yy	Two digit year
D	Three character day of week eg.Mon,Tue,Wed…
DD	Full character day of week eg. Monday, Tuesday, Wednesday…
t	Simple time format eg. 18:14:03
\\<char>	Escape character sequence. eg. \\m will print 'm'

The default format is : **H:M d/mm/y** eg. 22:10 05/10/2006

- **day$(time,type)**
  Returns day, as a string, represented by the time parameter. The type parameter controls the format of the returned string.
  - **Type = 1**
    Sun,Mon,Tue,Wed,Thu,Fri,Sat
  - **Type = 2**
    Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
- **fclose(fd)**
  Closes the file descriptor.
- **fg(Exp, Colour)**
  Sets foreground colour, always returns TRUE
  If 'Exp' evaluates to TRUE then it sets the foreground colour of the object in the list.
  The colour can be specified using the colour constant or can be an expression.
- **fgcell(Exp,Colour,Cellname)**
  Sets foreground colour of a single cell, always returns TRUE
  If 'Exp' evaluates to TRUE then it sets the foreground colour of just the specified cell in the list. For example, fgcell(cur, red, queue) will set the foreground colour of just the Queue Name for any queues which have messages on them.
  The colour can be specified using the colour constant or can be an expression.
- **flash(Exp)**
  If 'Exp' evaluates to TRUE then it will periodically flash the line, always returns TRUE
- **flashcell(Exp,Cellname)**
  If 'Exp' evaluates to TRUE then it will periodically flash specified cell, always returns TRUE
- **fopen(FileName [, filemode])**
  Returns file descriptor which can be passed into file routines, or 0 if open failed.
  If the file name contains the '\' character remember to use '\\' instead.

  Optional file mode parameter controls how the file is opened. Valid values are :-
  - **"r"** Open for reading
  - **"w"** Open for writing (Default if filemode not specified)
  - **"a"** Open for append
  - **"r+"** Open for reading and writing but file must exist
  - **"w+"** Open for reading and writing. Any current file will be overwritten.
  - **"a+"** Open for reading and appending
  - **"b"** Open in binary mode
  - **"t"** Open in text mode [Default if neither "b" or "t" specified]
- **fprint(File,Exp1,Exp2,Exp3,…….)**
  Writes the expressions to a file.

The file parameter can either be a file name or a file descriptor returned from **fopen()**
If the file name contains the '\' character remember to use '\\' instead.

- **fprintf(File,Fmt, Exp1,Exp2,Exp3,…….)**
  Writes the expressions to a file a formatted string.
  The file parameter can either be a file name or a file descriptor returned from **fopen()**
  If the file name contains the '\' character remember to use '\\' instead.

  For example the filter **fprintf("c:\\temp\\mo71.out","%50s %d\n",queue,cur)** on a queue list dialog will write a file containing the queue name and current depth of all local queues.

- **hour(time)**
  Returns the hour portion of the time parameter

- **max(Exp1, Exp2)**
  Returns maximum value of the two expressions

- **min(Exp1, Exp2)**
  Returns minimum value of the two expressions

- **minute(time)**
  Returns the minute portion of the time parameter

- **month(time)**
  Returns month of the year represented by the time parameter. Returned value from 0->11.

- **mon$(monthindex,type)**
  Returns month of the year, as a string, represented by the month index (0->11) parameter. The type parameter controls the format of the returned string.

  - **Type = 1**
    Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
  - **Type = 2**
    January,February,March,April,May,June,July,August,September,October,November,December

- **mqtime(Date, Time)**
  Accepts date in YYYY-MM-DD format and time in HH.MM.SS format and returns number of seconds since 1970 as a long.

- **qm(Pattern1,Pattern2,….)**[4]
  Associates the list dialog with all queue managers names matching the patterns. Wildcards are supported. Passing no parameters will reset the Queue Manager selection back to the original single Queue Manger.

- **qmloc(Pattern1,Patter2,…)**[4]
  Associates the list dialog with all queue manager location values matching the patterns. Wildcards are supported. Passing no parameters will reset the Queue Manager selection back to the original single Queue Manger.

- **second(time)**
  Returns the second portion of the time parameter

- **set(field,Exp)**[4]
  Sets the list field to a particular value. This can be useful to change the default values for a list dialog. For example, by default the connection list dialog shows 'connection' based values, but by using filter 'set(type,handle)' the handle based values will be displayed. By saving this filter as the default the defaults for the dialog can effectively be changed.

[4] This function is unusual in that it is not run against each entry in a returned list – instead it is run once, initially, when the filter is defined.

- **sortd(Exp)**
  Sorts the list in descending order of the expression. Note that since this is an explicit sort it will effectively disable sorting by the user by clicking on the list titles.
- **sort2(Exp)**
  Specifies the secondary sorting expression.
- **sound(Exp,Wave file)**
  Causes the application to play wave file, always returns TRUE. If 'Exp' evaluates to TRUE then the application will play the wave file.
- **status(Exp1,Exp2,Exp3,…….)**
  Writes the expressions to the status line of the dialog.
- **statusf(Fmt, Exp1,Exp2,Exp3,…….)**
  Writes the expressions to the status line in a formatted string.
- **str$(Exp1)**
  Return a string representation of whatever parameter is passed in.
- **system(Exp, Command String)**
  If the expression is TRUE then the Command is executed, always returns TRUE. This is useful if you want to check for out of line situations for example a Channel being in STOPPED state or a Queue Depth being larger than 100,000. The command is the name of an EXE program on the local machine and can be followed by parameters. The program will be started as a background process.
  e.g. command(cur>100000,"LOG.EXE -t \"Queue %o is in trouble!\" ")
  For more information refer to “Command Strings” on page 52.
- **weekday(time)**
  Returns day of the week represented by the time parameter. Returned value from 0->6 where 0 represents Sunday.
- **yearday(time)**
  Returns day of the year represented by the time parameter. Returned value from 0->365.

Output Format String

Some functions, such as **fprintf** and **statusf**, allow data to be output in a formatted string. The format string is modelled closely on the 'C' language printf format string.

%[flags][width][.precision][type]

where :-

flags

- **-** Left aligned
- **+** Prefixes the output a + or – sign
- **0** Prefixes the output number with '0'
- **#** For **o,x**, or **X** fields with prefix with 0,0x or 0X respectively.
  For **g** or **G** fields it forces the output to contain a decimal point.

width

An optional positive integer specifying the minimum number of characters printed for this field.

precision

An optional positive integer specifying the number of decimal places, the number of significant places or the number of characters to be printed depending on the data type.

type

- **d,i** Signed integer
- **o** Octal integer
- **u** Unsigned integer
- **x** Hexadecimal integer using lowercase characters
- **X** Hexadecimal integer using uppercase characters
- **e** Real number in the format [-]d.dddd e [- | +]ddd
- **E** Real number in the format [-]d.dddd E [- | +]ddd
- **f** Real number in the format [-]dddd.dddd
- **g** Real number in either **e** or **f** format depending on which is smaller
- **G** Real number in either **E** or **f** format depending on which is smaller
- **S** String

Examples :-

Specification	**Output**
statusf("%d" , 2)	2
statusf("%05d" , 2)	00002
statusf("%.7s","This is a test")	This is
statusf("%s = %d","Value",7)	Value = 7
statusf("%s = %d","Value")	Value = %d *Note that there aren't enough parameters to fill the inserts*

Filter Examples

On the basis that the best way to see what effect filters have is the try them, here are a number of examples (some of use and others purely for demonstration purposes). Note that the following filters are examples of filtering that can be done on the Queue List. Similar filters can be made on any of the other lists.

- **queue = "Q1"**
  Show only queue "Q1"
- **queue == "Q1*"**
  Show only queues with a name beginning "Q1"
- **queue == "*Q1"**
  Show only queues with a name ending "Q1"
- **queue == "*Q1*"**
  Show only queues with "Q1" in their name
- **queue == "????"**
  Show only queues with 4 character names
- **queue = 4**
  Show only queues with 4 character names
- **queue < 8**
  Show only queues with less than 8 character names
- **cur**
  Show only queues which have messages on them
- **cur > 100**
  Show only queues which have more than 100 messages on them
- **ipprocs# | opprocs#**
  Show only queues which are currently in-use.
  The hash characters force the relevant columns to be displayed
- **bg(usage#=xmitq,red)**
  Show all queues but highlight the transmission queues in red.
  The hash character will force the usage column to be displayed
- **fg(qtype=local,red) & fg(qtype=remote,blue)**
  Show all queues but highlight local and remote queue types in red and blue respectively.
- **(qtype=local) | (qtype=remote)**
  Show only local and remote queues
  Note that quotes are **not** used.
- **qtype==”*o*”**
  Show queues of a type which contains a ‘o’ character such as local, remote and model.
- **!initq**
  Show queues which do not have an initiation queue defined.
- **trigctl#**
  Show local queues that have trigger enabled
  The hash will force the trigger control column to be displayed
- **!trigctl#**
  Show local queues that have trigger disabled
  The hash will force the trigger control column to be displayed

- **!trigctl# | !initq# | (!process# & usage# != xmitq) | !trigtype#**
  Show all the queues which have triggering disabled for whatever reason, and display any of the fields provided they have a value.
- **sortd(queue)**
  Sort the entries based on the Queue name in descending order
- **sortd(+queue)**
  Sort the entries based on the length of the Queue name in descending order
- **alarm(cur>100)**
  Switch on alarm if any queue has a depth greater than 100
- **beep(!desc)**
  Beep if there's a queue without a description
- **system(curdepth>maxdepth*0.8,"q -oQ1 -M\"%o is getting full\"")**
  If the queue is more than 80% full then issue the given command. The command given above is actually a call to my Q program (WebSphere MQ SupportPac MA01) which sends a text message to a queue. This demonstrates, in a simple way, how you can monitor the attributes of your objects and have commands issued should certain conditions arise.
- **bg(qtype=local,yellow) & fg(usage#=xmitq,red) & !(queue=='SYSTEM*') & sort(queue)**

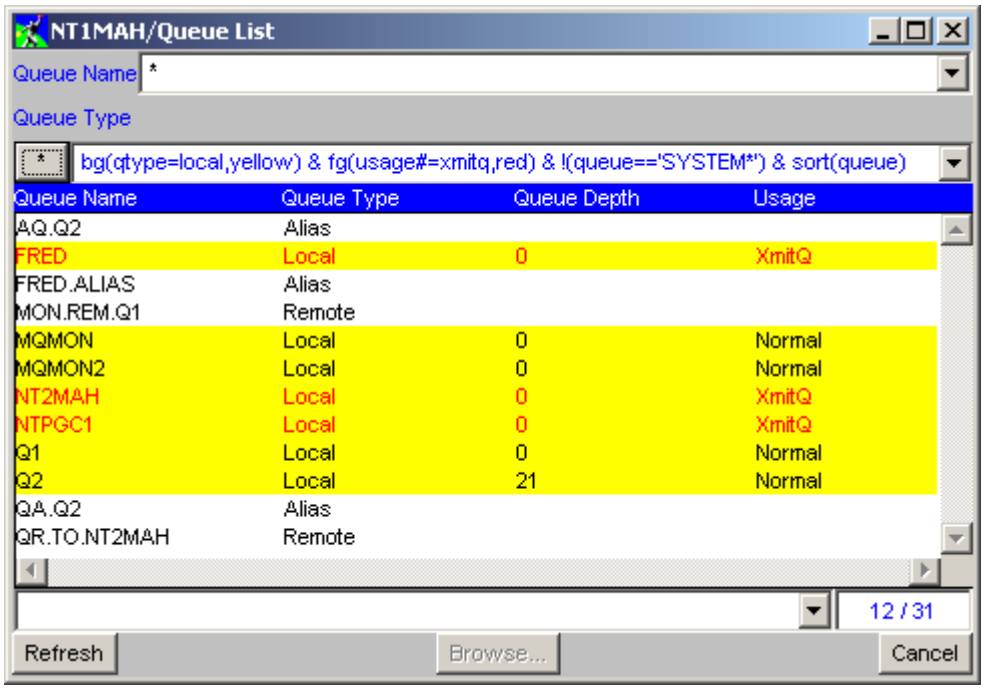


Figure 9: Queue List with filter

- **@time := mqtime(crdate,crtime); @$d# := day$(weekday(@time),2);1**
  On a queue list dialog display a new column showing the day of the week the queue was created.
- **Depth History**
  To demonstrate the use of a user variable object insert, the following example will change the colour of a queue list entry depending on the depth of the queue. However, what makes this example different is that the depth of the queue is only considered significant if depth for this queue has been non-empty for multiple refreshes of the filter.

```
if (cur) @depth%o := @depth%o + 1;
   else  @depth%o := 0;
if (@depth%o)
{
  if (@depth%o > 3) bg(1,red);
```

```
                else bg(1,yellow);
}
else
{
  bg(1,green)
}
sortd(@depth%o);
```

- **Total number of messages on all queues**

  This example adds up the cur (or curdepth) value for each queue in a queue list. It then displays the total on the status bar after processing the last entry.

```
if (_first)
{
  @total := 0;
  _first := false
}
@total := @total + cur;
if (_last) status("Total messages = ",@total);
1;
```

- **Fullest Queue**

  The following example shows how you can collate information about all the list items and then display the result. This simple filter calculates which queue is fullest and displays it to the status bar.

```
if (_first)
{
  _first:=false;
  @$name := "";
  @depth := 0;
}
if (curdepth > @depth)
{
  @depth := curdepth;
  @$name := queue;
}

if (_last)
{
  if (@depth) status("Fullest queue is ",@$name," with ",@depth," messages.");
        else status("All queues are empty");
}
1;
```

- **Top two depths**

  This example shows the ability of requesting a rescan. This example uses scan 0 to find out what the top two current depths of a queue are (an expansion on the fullest queue example). A rescan is then requested and in scan 1 this data is used to show only the queues which have greater or equal to the top two depths of all queues.

```
if (_first)
{
  if (!_scan)
  {
    @depth1 := 0;
    @depth2 := 0;
    _rescan := 1;
  }
  _first := 0;
}
if (_scan)
{
  (@depth1 & (cur >= @depth1)) |
  (@depth2 & (cur >= @depth2))
}
else
{
  if (cur > @depth2)
  {
    @depth1 := @depth2;
    @depth2 := cur;
  }
  else
  {
    if (cur > @depth1) @depth1 := cur;
  }
}
```

Filter Troubleshooting

There are a number of common mistakes, particularly with the more complicated filters, which can often lead to unexpected results when using filters. Check your filter against the list below if you see odd behaviour.

- **Did you remember to return a Boolean ?**
  Remember that as well as processing some data the filter must also return a Boolean value indicating whether the entry should be shown or not. In a number of cases above this is achieved simply by adding the characters **;1** to the end of the filter. This ensures that the filter always returns true and therefore the entry will always be displayed.
- **Are you using files ?**
  Remember that a '\' character must be represented by '\\' in a file name.
- **Remember to use := not = for assignment**
  If you're using user variables then you must use := to assign a value. Both x := 3 and x = 3 are syntactically correct but the first is assigning the value 3 to x and the second is comparing the value of x with 3. Make sure you use the right one.
- **A displayed user variable and one that isn't aren't the same variable**
  The act of displaying a variable actually changes where the variable is stored which therefore changes the value. In other words @x# (which is displayed on the screen) is not the same variable as @x. You can prove this with simple test filters
  - **@x# := 5; @x := 2 ; 1**
    This filter will generate a column for **x** all containing 5
  - **@x# := @x# + 1; 1**
    This filter shows a column of all 1's. Why do we not see the number increasing for each entry ? Well, the displayed field is set to zero initially for each entry.

    If we wanted to see an increasing number we'd have to use the following filter.
  - **@x := @x +1; @x# := @x; 1**
    Here we see different numbers and they seem to be increasing but there are not in order. The reason for this is that the order displayed in the dialog is not necessarily the order that the objects are returned from the command server.
- **Remember to switch _first off**
  It is very common to want to do some initialization in your filter like reset counters to zero. To do this you put it inside an **if (_first)** block. However, remember that you yourself must set **_first** to false. If you don't your counters will be reset to zero for every entry in the list.
- **Remember operator precedence**
  For example, suppose you wish to see the queues that don't match the pattern "*OBJ*". At first glance you might think is was **! queue == "*OBJ*"**. However, because the **!** operator is higher in precedence than == it will be executed first which will give a completely different result. The solution is to use brackets. The expression **! (queue == "*OBJ*")** will work fine.

Filter Manager

There may be a number of filters which are either used frequently or that are fairly complicated. For these filters it may be more convenient to define the filter globally and assign a name to it. The name should not contain blank characters. To access the filter manager use the *'Action-Filter...'* main menu item or by pressing **<Ctrl-F>**. This will display the following dialog.

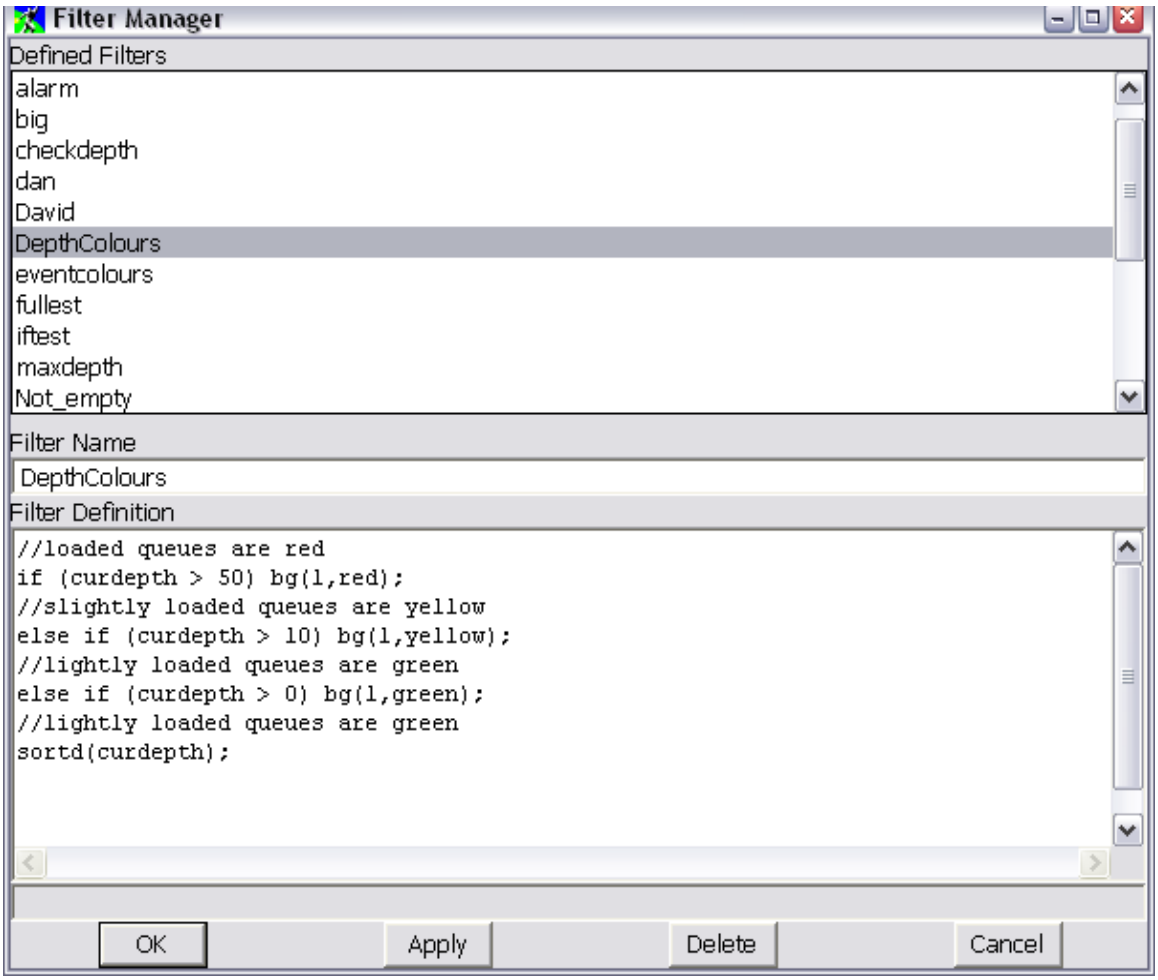


Figure 10: Filter manager Dialog

In this example a filter has been defined called 'DepthColours' which will set the background colour of a queue in a queue list window to a colour representing its current depth. To use this filter a filter of '$DepthColours' should be specified in a queue list window.

Some filters can be used with all list types but since the example shown uses the variable 'curdepth' it should only be used on queue list displays. If this filter were used in say a Channel list display and error message saying 'Unrecognised keyword 'curdepth'' would be displayed.

Changing a filter and then pressing OK or Apply will cause any dialogs using this filter to update their display.

Chapter 5. Queue Manipulation

To manipulate messages MQMON needs direct access to the queue since no support is provided by the command server. It is possible to browse messages on other queue managers but you need a program on the remote system that understands the PCF messages generated by MQMONNTP. MQMONNTP itself clearly does understand the flows so it is quite possible to have, for example, two NT machines each running MQMONNTP browsing each others queues[5].

There are three ways of displaying the contents of a Queue.

1. **Browse Queue**

   This dialog will show the list of messages on the specified Queue

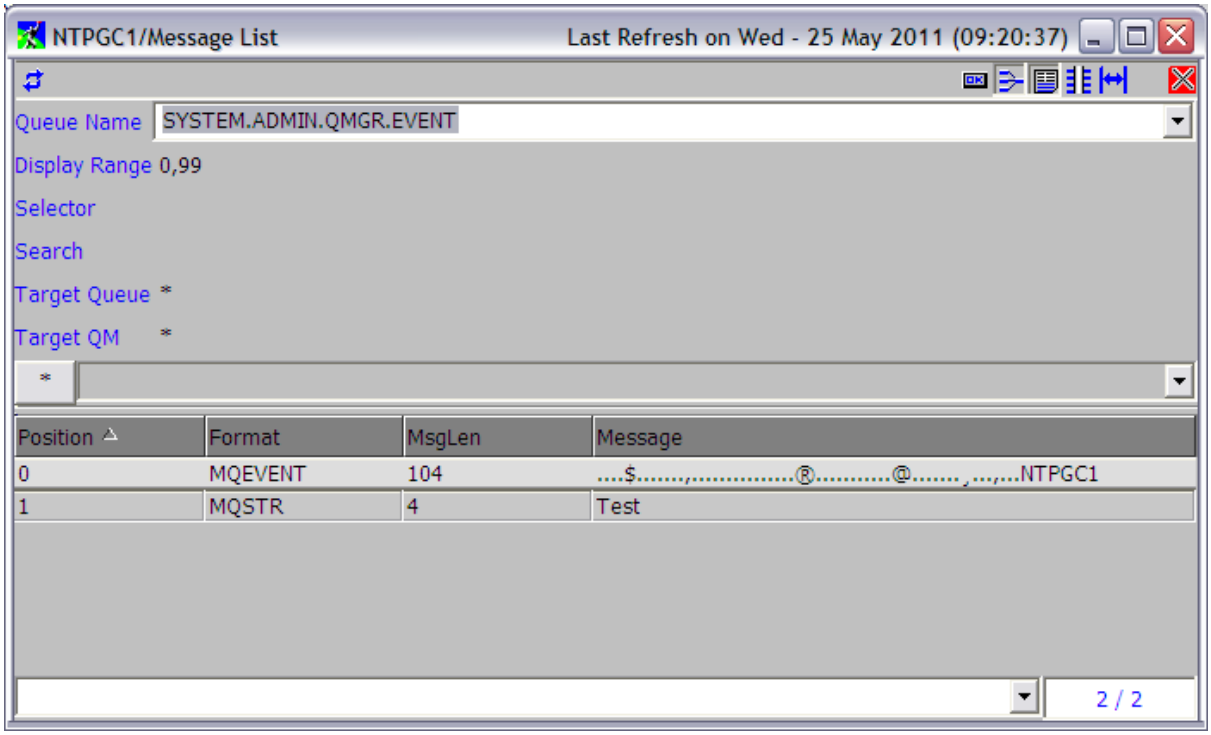


Figure 11: Message List Dialog

When performing an operation against one or more messages you can use the 'Message Selection' to determine how the application matches the selected messages against the messages on the queue. The problem is that WebSphere MQ does not have to assign a unique Message Id to each message it is under application control. Therefore when selecting a message to move the Message Id may not be sufficient, checking the position on the queue and the Message Id is much safer. Of course, even this is still not guaranteed to identify the same message.

2. **Browse Message**

   This dialog will show the contents of the message formatted into a human readable form. If some characters do not display as expected, it may be necessary to adjust the locale setting. For more information see "Appendix C:Display National Language Characters" on page 112.

   The application only understands standard WebSphere MQ formats. But it allows the user to look at messages on Dead Letter Queues, Transmission Queues and Command Queues for example.

   This is the most complex dialog for queue browsing. There are a number of options on the Pop-up menu which let you control what data is displayed.

[5] I do not provide a description of the messages but they are fairly easy to work out. It should not be too difficult to write an application on your target system which reads the queue and generates the correct responses. For simplicity, however, I suggest that most people stick to browsing queues only on the Queue Manager the application is connected to directly or via a client connection.

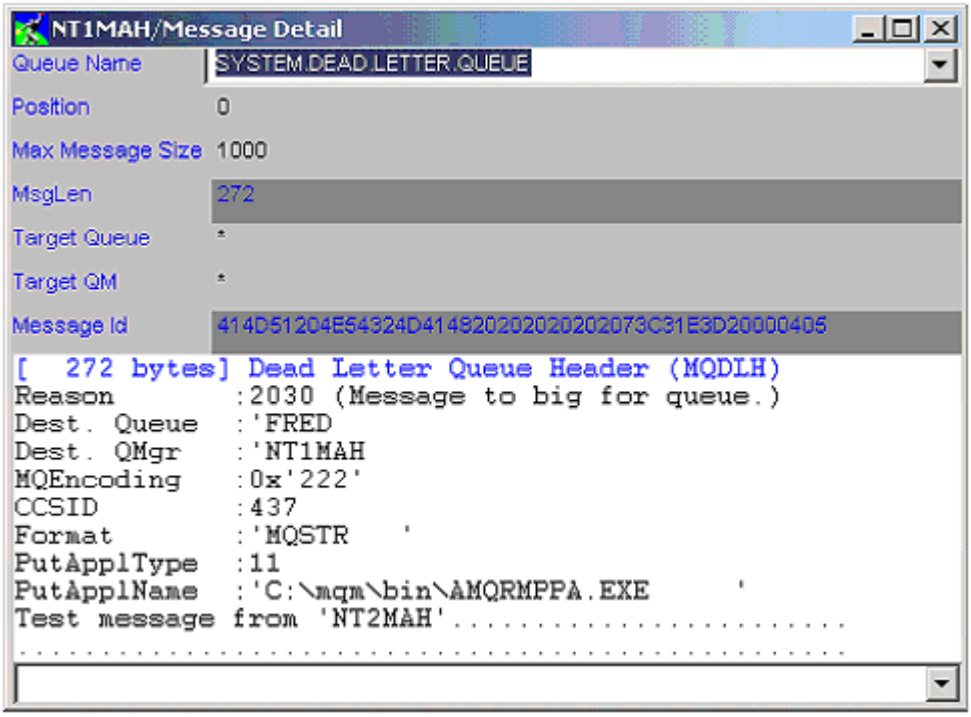


Figure 12: Message Dialog

3. Message Detail

This dialog will show the message descriptor fields and the first few bytes of the message for a particular message on a queue

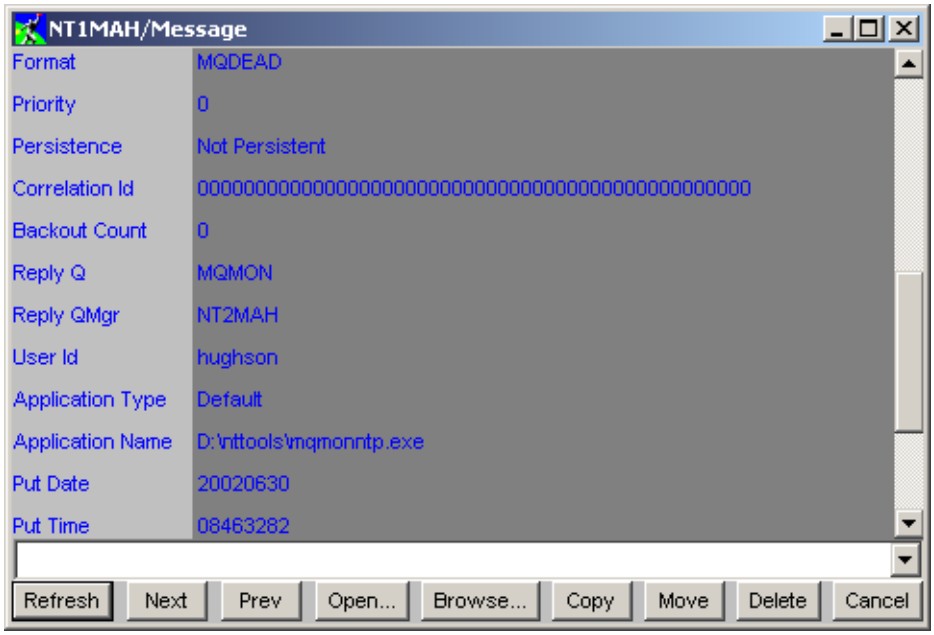


Figure 13: Message Detail

Identification of a message is controlled by either or both of its position and its Message Id. The position is purely an index of the message when the queue was browsed last. If messages are being added and removed from the queue while you're browsing the queue then the same message may well be given a different position number the next time you refresh the display.

Fields

The fields at the top of the dialogs allow control over the messages displayed and also controls when targeting a message to another queue.

Display Range

When a list of messages is requested you have the option to specify the range of messages you're interested in. By default the range is '0,99', which means the first 100 messages. This prevents accidentally displaying thousands of messages if the queue contains more messages than you expected !

However, if you find that you frequently want to browse a different number of messages than the first 100 you can change this setting in the preferences dialog. See "Preferences Dialog" on page 78 for more details.

Selector[6]

The selector field allows the user to enter an SQL92 expression which is executed at the server. Descriptions of the format of the expression can be found in the Message Selector Syntax section of the WebSphere MQ documentation.

Simples examples would be:

- **Root.MQMD.Persistence=1**
  Show only the persistent messages
- **Root.MQMD.UserIdentifier = 'clarkep '**
  Show only messages put by a particular user
- **myprop > 10**
  Show only messages where the value of property 'myprop' has a value greater than 10.

Note that all field names in selectors are case sensitive.

Search

The search field allows the user to enter a search string for the messages. Only messages which contain this case sensitive search string will be displayed. Note that only the message itself, not the message descriptor is checked for the string. When a search string is used the display range fields control which of the matching messages are displayed. For example, 3,5 will display the 4th, 5th and 6th matching messages. Note that the '**Search Offset**' setting controls whether the portion of the message shown starts at the point where the search string is found or whether it is the start of the message.

Max Message Size

When displaying message detail only the first 1000 bytes are shown by default. This prevents very large messages causing network delays etc. However, if necessary this value can be increased to whatever size is required. If you frequently browse messages larger than 1000 bytes and find yourself constantly increasing this value then it might be worth increasing the default maximum browse message size in the preferences dialog. See "Preferences Dialog" on page 78 for more details.

If the message being browsed is larger than this maximum message size then clearly the message displayed will be truncated. This may cause error messages to be displayed at the end of the message reporting that the data is incomplete. A common type of message to have this problem is XML since XML messages are often fairly large and any truncation will truncate the end-tags making the message incomplete.

Target Queue Manager

When copying or moving messages to another queue you can optionally identify the target queue manager for the message, by default it has the value '*' which has the following effect :-

If the message has a MQXQH or MQDLH header then the queue manager is that which is specified in the header

If the message doesn't have a header then it is the local queue manager

[6] Available on Queue Managers MQ V7.0 and later

Target Queue

When copying or moving messages to another queue you can optionally identify the target queue for the message, by default it has the value '*' which has the following effect :-

If the message has a MQXQH or MQDLH header then the queue is that which is specified in the header

If the message doesn't have a header then this is an error and a message will be displayed asking for the target queue name to be identified.

Pop-up Menu

The operations that may be available are listed below. Some of the most common actions can also be found as buttons.

Next

Will skip to the next message, if any, on the queue.

Prev

Will skip to the previous message, if any, on the queue.

Display Format

This option will allow the user to control the amount and format of the data that is displayed.

- **Formatted Message**
  If selected the application will attempt to format the message into human readable text. Once it no longer recognises the format it will display the remainder in hex.
- **Hex Message**
  If selected the message will be printed in hex.
- **Message Descriptor**
  If selected the contents of the message descriptor are displayed before the message.
- **Display Offset**
  In a HEX display the offset of the bytes from the beginning of the data is displayed.
- **Ascii Column**
  In a HEX display this option controls whether a column of ASCII characters is displayed on the right of the display. This can be useful to make sense of all the HEX characters.
- **Auto Detect XML Message**
  When selected the application will examine the first few bytes of the message to see whether it contains XML tags. If it does the message will be formatted as though it were an XML message. If not selected only message with the MQRFH2 format will be treated as XML.
- **Display XML shortform**
  When selected, any XML message will be displayed in an abbreviated form where not all tags and tag characters are displayed to aid readability.
- **Display Hex as chars**
  When selected, the hex fields in the MQ headers will be displayed as character strings as well as in hex. This can be useful if the Message Id or Correlation Id, for example, are character strings.

Detail Level

Will allow the user to control how much detail is shown when structures are displayed.

- **Low** Only the major fields of the structures are shown
- **Medium** The majority of structure fields are shown
- **High** All structure fields are shown

Copy To Clipboard

Will copy all the message text, as displayed, to the clipboard. Only the visible portion of the message is copied. To copy the entire window to the clipboard, the standard windows action is <Alt>+<PrtSc>. This will work for practically any window.

Copy All To Clipboard

Will copy the entire message to the clipboard, including the data that is not currently visible.

Copy

Will copy the selected messages to the queue identified by the Target Queue and Target Queue Manager fields.

This option is not available if the context setting is 'pass'.

Move

Will move the selected messages to the queue identified by the Target Queue and Target Queue Manager fields.

Delete

Will delete the selected messages from the queue.

Put Message

This will present a put message dialog which allows the user to put a simple message. For more information see "Put Message" on page 76.

Load/Unload Messages

This will present a dialog which will allow the user to load or unload messages between a queue or file or between two queues. For further information please see "Queue load/unload facility" on page 38.

New

This will present a blank dialog of a type suitable to define a new object of the type displayed in the list. Not all list types will have this option.

Convert

By default browse operations will convert the message to the code page and encoding values of the workstation. The codepage used to convert to can be changed in the preferences dialog if the windows codepage is not suitable for the message being browsed. By the deselecting the *'Convert'* menu option messages will not be converted when they are browsed.

Note that message copy, move and delete operations never convert the message.

Properties

This menu controls whether message properties are returned in a header at the front of the user message (see MQGMO_PROPERTIES_FORCE_MQRFH2). Currently this is the only way in which message properties can be viewed when browsing a queue.

Strip Headers

Messages on a WebSphere MQ queue may have a transmission queue or dead letter queue header pre-pended to the actual application message. By default it is assumed that these headers should be maintained if the message is moved or copied to another queue[7]. This helps to ensure that the full message is maintained. However, there are times when these headers should be stripped off, for example, when moving a message from a transmission queue or Dead Letter Queue to a new target.

[7] Earlier versions of the SupportPac had this option on as default. However, having 'strip' off is regarded as a safer option.

Ignore CR/LF

This option controls whether carriage and line feed characters in the message data should be adhered to or not. By switching on ignore the data will flow in a single line.

Multi Transaction

Normally all copy, move and delete operations will be done under one transaction, this avoids the problem that a system crash or some sort of failure will leave the operation only partially completed. All the operations will either complete or the transaction will be backed out. There is a queue manager attribute, max uncommitted messages, which dictates the maximum number of messaging operations allowable by an application in a single transaction. If you are trying to move or copy more messages than this limit then clearly it can not be done in a single transaction. In these cases you must select the *'Multi Transaction'* option and take the small risk that the operation may only partially complete.

Search Offset

When using a search string in a message list display this setting controls whether the 100 byte message portion that is displayed is from the start of the message or from where the search string is found within the message.

Message Selection

Operations such as Copy, Move and Delete will apply to the current message if only a single message is displayed or the selected messages if a list of messages is displayed.

A third option is to have the operation applied to all the messages on the queue. This is achieved by selecting *'Apply to all messages'* in the *'Message Selection'* menu. If this menu is selected then the action buttons will change to reflect the 'all messages' status. Because this is a potentially hazardous operation the 'All message' mode is removed as soon as any action is selected.

Context

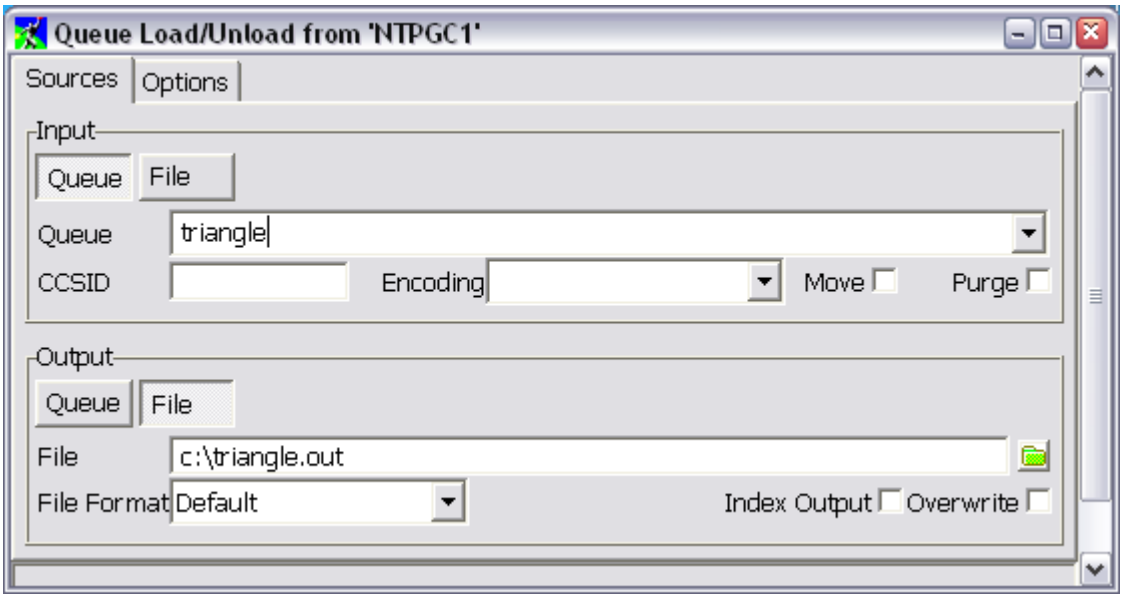
When copying or moving messages between queues the context of the message will also be copied or moved by default. However, there are other options :-

- **Set all (default)**
  The context information of the copied or moved messages will be same as the original. This does require a high level of authority on the target queue.
- **Pass**
  Will pass all the context from the source message to the target message. Pass context is only valid on a move operation not copy since pass context is not valid after a browse.
- **Default**
  Default context is used. The identity and origin context of the new messages will be that of the MQMONNTP program itself.

Chapter 6. Queue load/unload facility

From various places, such as the queue list and message browse dialogs a menu option is provided which will present the queue load/unload dialog. This allows the user to unload messages from a queue and put them in a file, copy messages from one queue to another, or load messages from a file to a queue. This can be useful for a variety of reasons such as backing up your queues, saving a set of messages for later re-use, unloading a queue so that you can do some editing of the message or message attributes or for sending someone else a copy of your messages.

The load/unload facility is essentially the same code as in my MO03 command line queue load/unload SupportPac but presented in a window. As a consequence unloaded files can be shared between them.



The dialog is split in two halves. For both input and output a choice has to be made about whether to use a file or a queue. So, for example, if the user chose an input of a queue and an output of a file then this would move messages from a queue to a file. In other words, it would unload the queue to the file. Note that it will not destructively get the messages; rather a copy of the messages will be taken.

Both input and output can be of the same type. If they are both queues then you can do a copy of messages from one queue to another. If both are files then it allows you to reformat the output of the file.

The various options are:-

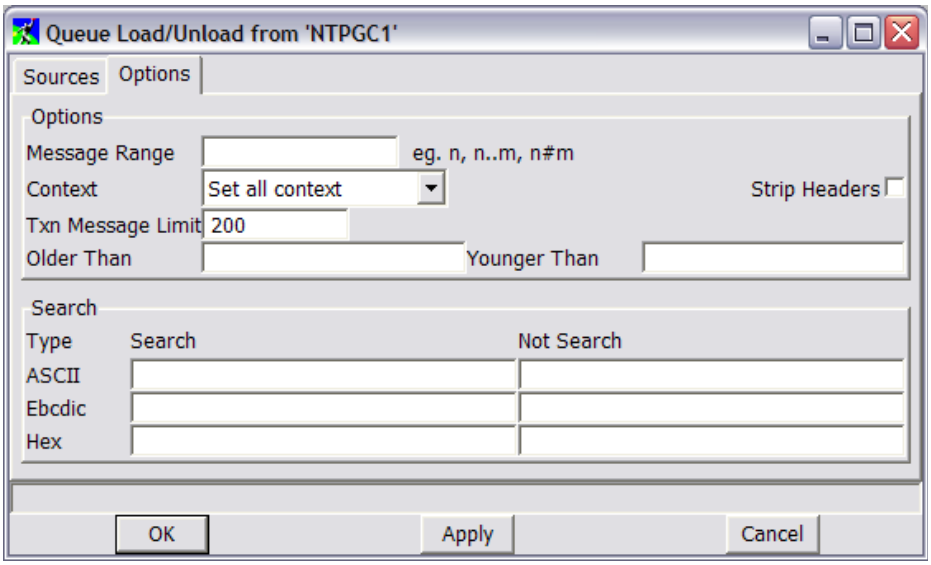
- **CCSID**
  If specified the messages will be 'got' from queue converted to the requested codepage.
- **Encoding**
  If specified the messages will be 'got' from the queue using the byte encoding requested.
- **Move**
  If this checkbox is checked then the messages will be removed from the source queue otherwise they will be copied.
- **Purge**
  This option is only in effect when the move option is specified and some form of filtering is in effect in the options tag. When selected it says that messages not matching the selection should be removed from the queue (and are therefore lost since they **will not** be written to the target). If this options is not selected then messages not matching the selection will remain on the source queue.

- **File Format**
  The following file formats are available:-
  - **Default**
    The message format is just a hex string output. This is the most compact format but not too friendly to the human eye.

    ```
    X 54657374204D65737373616765
    ```
  - **ASCII**
    Message is presented in ASCII as much as possible. This is useful if the message contains a fair degree of ASCII which needs to be edited.

    ```
    S "Test Messsage"
    ```
  - **ASCII Column**
    Message is presented in hex but with a column of ASCII characters to the right-hand side. Any character in the message which can not be represented by a printable ASCII character will be displayed as a '.' (dot).

    ```
    X 54657374204D65737373616765                    <Test Messsage>
    ```
- **Index Output**
  When selected the produced file will contain comments which indicate which message index each message is. This is useful when reloading only part of a file to identify each message by its index value.
- **Overwrite**
  By default the application will prompt the user if the output file already exists. If this option is checked then any existing file will be overwritten without requiring user confirmation.



A number of options are available to control which message are loaded/unloaded and what message attributes should be taken.

- **Message Range**
  This option allows only a certain number of the messages to be processed. For example
  - **5** Would only process message number 5
  - **5..8** Would only process messages 5 through 8, inclusive.
    To process all messages from message 5 onwards enter a string like **5..999999**
  - **5#8** Would only process 8 messages starting with number 5
  - **#8** Would process the first 8 messages.
- **Context**
  Context setting which control how the origin and identity context is transferred between the input and output source. The value of 'set all context' will ensure that all of the context fields are transferred and would

therefore tend to be the most obvious choice. Note, however, that 'set all context' requires the most authority.

- **Strip Headers**
  This option controls whether headers, such as transmission queue and Dead Letter Queue headers should be stripped before the message is moved or copied.
- **Transaction Message Limit**
  If you are moving message from one queue to another then this must be done under a transaction to be totally reliable (in the case of persistent messages it is also much more efficient). However, there is a limit to how many operations you can perform in a single transaction so this value allows you to say how many messages form part of the transaction before the transaction is committed.
- **Older Than**
  This option allows selection based on the age of the message. The parameter can be entered free format using the keywords, **seconds, minutes, hours, days, weeks, years**. Only the initial parts of the keywords need be specified. For example, **1 hour 15 minutes** or **1 h 15 m** are equivalent.
- **Younger Than**
  This options allows selection based on the age of the message. The format of this parameter is the same as the **'Older Than'** field.
- **Search**
  You can choose to process only messages that contain a certain string. This string can be in ASCII, EBCDIC or hex. You can also do the opposite search - that is for messages that do not contain a certain string. You can use any combination of the search string options together. If more than one option is used, all search strings must match for the message to be processed.

File Format

The format of the file that MO71 uses is the same as SupportPac MO03. The file is deliberately human-readable to allow a user to update the file to easily make changes to the messages before loading them onto a queue, perhaps on a different system.

The file has a free format. Spaces and blank lines are ignored unless between quotes. When generated the file may contain spaces to make the file more readable, but they do not affect the message format stored in that file.

The first column contains the key for each line. This can have a number of different values, some of which we have seen already. These are listed in "Table 1: Meaning of column one symbol in file format".

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

Table 1: Meaning of column one symbol in file format

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

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

Example - Changing the user ID

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

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

Attribute Format Reference

The fields in the Message Descriptor (MQMD) are formatted in the file with a three character string representing the attribute name. “Table 2: Message descriptor attribute representations” lists the full set of these strings and which field they represent.

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

Table 2: Message descriptor attribute representations

Recognised file formats

MO71 recognises the file format described above but also recognises the file format used as output from the sample browse program AMQSBCG. In other words, you can take an AMQSBCG output file and pass it as an input file to the load/unload dialog.

Chapter 7. Network View

The network view is by far the most complicated dialog in MQMONNTP. The general idea is to try and relate the various objects in the various Queue Managers. It is started by the menu option *'View-View Network'* from the main window. By selecting *'Action-Auto Start'* from the network dialog menu itself the Network view can be set to start automatically whenever MO71 starts up.

Both manually defined objects and cluster objects will be displayed in the Network View.

The main window can either display one or two windows. If two windows are displayed a sliding bar allows the user to set the proportion of the main window given to each sub-window.

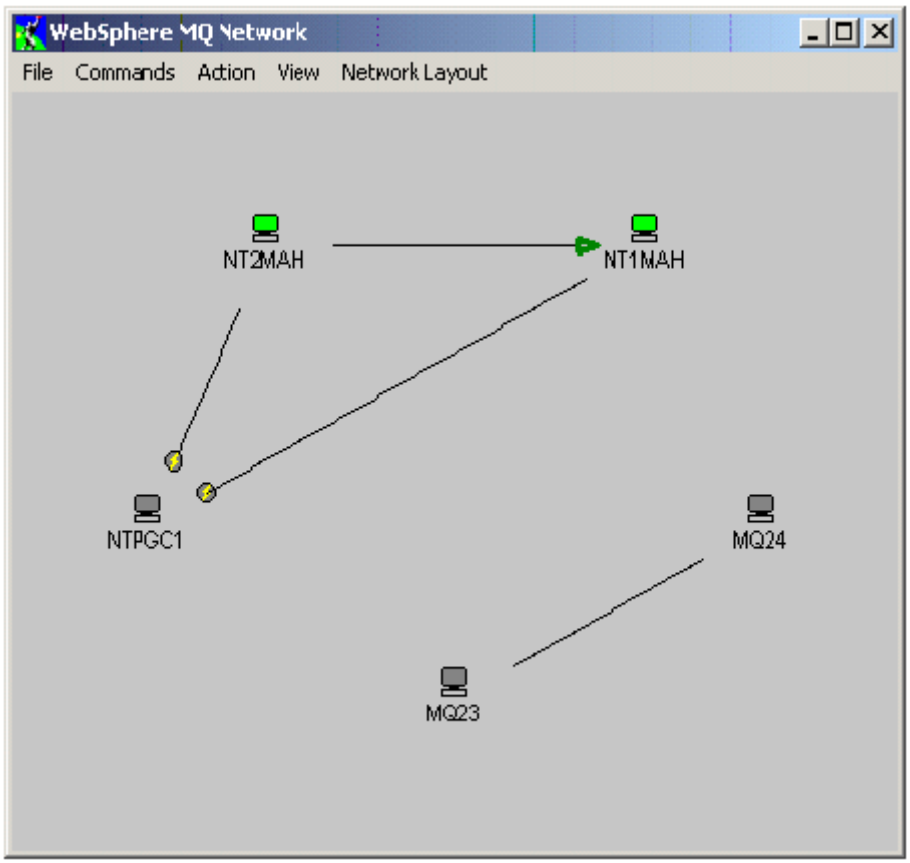


Figure 14: Network Window

A number of the windows display a hierarchical tree view known as a container. These containers are very similar to Windows container views displayed in applications such as Windows Explorer. However, an explanation of the keys available is included later in this document.

The sub-windows that can be displayed are described below.

Network Display

The Network display shows the topology of the Queue Manager network according to the channel definitions. I discovered while writing this code that displaying a set of interconnected nodes in a reliable, eye-pleasing fashion is actually very difficult. For small networks a simple circle is hard to beat. For large networks the problem becomes very difficult very quickly. Ultimately there is probably no substitute for allowing the manual placement of the Queue Managers in the display. I have therefore initially allowed three types of display selectable from the *'Network Layout'* menu.

- **Circle layout**
  This simply displays all Queue Managers in circle. All Queue Managers in the network are displayed.
- **Diagram layout**
  This layout uses the links between the Queue Managers to try an devise a reasonable looking topology. Only interconnected Queue Managers are displayed.

- **User layout**
  Allows the user to manually position each Queue Manager by clicking on the Queue Manager[8], holding the mouse button and moving it to the desired location. Note that it is much easier to do this with *'Scale'* switched off otherwise you can get some very strange results.

The other menu options available in the *'Network Layout'* are **:-**

- **Show location name/Show Queue Manager name**
  Allows control over which name is used in the display
- **Scale**
  When checked all Queue Managers will be scaled to fit on a single page. With scaling switched on moving Queue Manager at the extremes of the window can produce unexpected results.
- **Show Grid**
  When selected an array of dots is displayed. The size of the grid is determined by the Grid size specified in the Preferences dialog.
- **Snap all to Grid**
  When selected all Queue Managers will snap to their nearest grid co-ordinate. This can be useful to align Queue Managers for neatness.
- **Highlight Channels**
  When selected the channel lines from the selected Queue Manager are displayed with a thicker line in the highlight colour to make it easier to see which Queue Managers the selected Queue Manager is connected to.
- **Centre QM**
  Will move the selected Queue Manager to the centre of the network display. This can be useful if the current position of the Queue Manager is unknown.
- **Save layout**
  When you are happy with the layout for your Queue Managers you can save the current settings by selecting this option. If you want to back out subsequent changes pressing the 'User layout' option will go back to this save point. Note that the save is purely to memory. You must then press 'Save Configuration' on the main window if you want your values written to disk. As usual all changes will be written to disk if the application ends normally.
  If the layout is changed and the dialog is ended the application will automatically prompt the user as to whether the current layout should be saved.

Display Format

The network display will show the Queue Managers that have been selected as part of the 'network'. If a pair of Queue Managers have at least one channel pairing between them then a line will be drawn between the two Queue Managers showing that messages can flow between them. Only a single line will be shown regardless of how many channel pairs are defined between the Queue Managers.

To indicate the state of the channels between the Queue Manager pair the ends of the lines may have a number of different symbols. Since a single line could represent a number of different channels it is necessary to define some order of precedence. This list is therefore given in precedence order.

- **Arrow** At least one channel is running in direction of arrow
- **Lightening Bolt** No running channels and at least one channel in retrying or binding
- **No entry symbol** Channel stopped
- **No symbol displayed** Channel inactive

See "Appendix A:Icons" on page 109 for examples of what these icons look like.

In the preferences dialog you can set how often the channel status information is retrieved from the Queue Managers. See "Preferences Dialog" on page 78 for more details.

[8]Note that moving a Queue Manager manually while in any layout will change the layout to User.

Verify Display

This window will display the definitions that MQMONNTP has stored for each of the Queue Managers in the network[9]. Each Queue Manager contains a hierarchical display. The levels of the hierarchy are expanded by pressing on the horizontal arrow, and collapsed by pressing on the vertical arrow. The complete hierarchy is shown in the 'levels display' but essentially it contains the list of queues and channels defined for the Queue Manager.

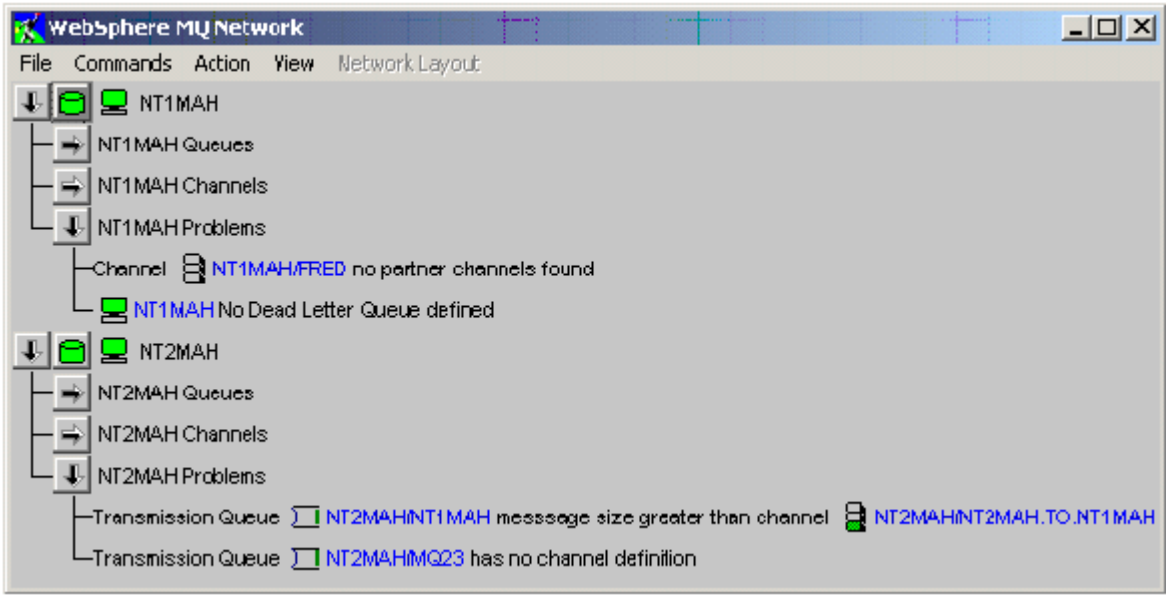


Figure 15: Verify Window

Some of the objects will also have sub-trees to relate each object to other objects in the Queue Manager and other Queue Managers in your network. For example a remote queue will expand to show which transmission queue it resolves to, which in turn will expand to show the remote Queue Managers that the transmission queue services. Each of those Queue Managers will expand to show the channels used to get to them from this transmission queue. Once at the target Queue Manager the path resolution process continues until ultimately it will show the local queue that the remote queue resolves to.

- Similarly, local queues will expand if they are resolved to from another queue in the network.
  Path resolution can be a complicated, time-consuming task and as such can be switched off in the Preferences dialog. For details see "Preferences Dialog" on page 78.

The channel objects will expand if a partner channel (i.e. one with the same name) is detected in the network.

- An additional ***problems*** sub tree may be displayed after the channels sub tree if enabled in the Preferences dialog (see "Preferences Dialog" on page 78). The Problems sub tree contains a list of all the possible problems detected by the application in this Queue Managers configuration. Note that this is just a guide and a highlighted problem may not in fact be a problem. For example, the application expects each Queue Manager to have a defined Dead Letter Queue which is a strong MQ recommendation. There are, however, good reasons why the user may not wish to use a Dead Letter Queue. For this reason the user may use the problem selection window (see "Problem Selection" on page 47) to disable the checks which are not wanted. It should be pointed out that the problems displayed are not an exhaustive test, while I do intend to increase the number of consistency checks over time, there is no guarantee whatsoever that all problems will be detected.

In the object hierarchy various queue manager names, queue names and channel names will be displayed. These are really a form of push button. Clicking on the name with a mouse or pressing <space> while the object has focus will bring the up a dialog of that object allowing the user to examine its attributes and make changes if necessary.

Verify Display Search

It is possible to search the verify data for certain string, including wildcards. By selecting 'Search' from the *'View"* menu a search field is displayed above the verify data. By selecting 'Search Status' an additional line is displayed which shows exactly what is being search.

[9]Note that the Queue Manager will only appear in the list when definitions are received from the Queue Manager

The format of the search string is fairly straightforward. A comma (or space) separated list of search strings can be entered. The search strings can contain the normal wildcards '*' or '?'. '*' represents 0 or more characters, '?' represents exactly one character. If a list is specified the results show the results or either search; in other words the searches are additive.

When data is entered into the search field the search will applied after the configured delay interval. This is set in the 'Network' tab of the preference dialog.

Automatic Search Wildcards

A preference option, in the Network tab, controls whether '*' are added to the entered search string automatically. For example, with the option selected, entering 'abc' will actually search for '*abc*'. In other words any entry containing the string 'abc'. By enclosing the search string in double quote '"' characters the automatic wildcard addition can be suspended. Entering '"abc"' will just search for the string 'abc' without any additional wildcards. In fact, the double quote can be entered independently at each end of the search string. So, if one wishes to search for entries which start with the string 'abc' one can enter "abc.

Case Sensitivity

Whether the search is case sensitive or not is controlled by a preference option in the Network tab.

Search Qualifiers

It is possible to qualify the search so that only certain objects or fields are searched. This allows the search results to be significantly reduced, for example, only show the queues containing this string. Qualifiers are entered as a list between '<' and '>' brackets. So, for example the search string **abc <q c>** will just search for queues and channels containing the string 'abc'. The search string **abc <q c desc>** will only search the description field of queues and channels for the string 'abc'. The full list of qualifiers which can be used are:

Qualifier	Qualifier Shortform	Short	Effect
Channel	**chl**	**c**	Channel Objects
Queue		**q**	Queue Objects
Process	**prc**		Process objects
Qmgr		**m**	Queue Managers
Namelist	**nl**		Namelists
Cluster	**cl**		Cluster field
Topic		**t**	Topic Object
Description	**desc**	**ds**	Description field
Name	**nm**		Name of object
Application	**app**		Application Name field
TopicStr	**ts**		Topic String field
Reference	**ref**		Reference For example, the base queue of an alias queue definition.

The abbreviated names, if given, can be used instead of the full name.

It can be useful, in some cases, to effectively use the qualifier on its own. For example, the search string *** <q>** will just show entries relating to queue objects. Similarly **?* <q desc>** will show only queues containing descriptions.

Queue Managers Display

This window displays all the configured Queue Managers and allows the user to indicate which Queue Managers are part of the 'network'. The 'network' of Queue Managers is the subset of Queue Managers which will be acted upon by the Network View. Not all of your configured Queue Managers might be considered part of your immediate network and they can therefore be excluded from analysis. Each Queue Manager can either be part of or excluded from the network.

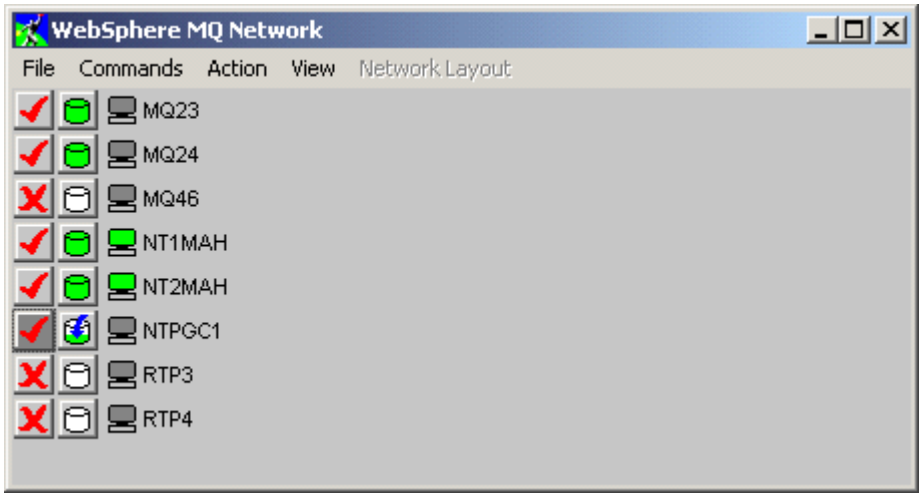


Figure 16: Queue Manager Window

A Queue Manager is part of the network if a 'tick' appears next to its name. A Queue Manager is not part of the network if a 'cross' appears next to its name[10]. Clicking on the icon (or pressing <space> while it has focus) will alternate between the two choices. Each location can be defined as *associated* with a number of network names. This allows many Queue Managers to be included or excluded from the network view in a single mouse click. See "Network Names" on page 47 for a description of this.

The next icon after the tick/cross indicates whether object definitions have been received from that Queue Manager. The icon can either be empty (white), full (green), overdue (red) or requesting (half full with arrow). At any time the user can press this button and the application will get a new set of definitions from the Queue Manager. See "Appendix A:Icons" on page 109 to see what these icons look like. In the 'Network' tab of the preference dialog there are values which can set by the user to control the automatic retrieval of the object definitions and what constitutes 'out of date' definitions.

Levels Display

The Verify window can contain quite a complicated hierarchy. This 'Levels' display shows the various elements of this hierarchy. By enabling or disabling various levels elements can be removed/displayed in the Verify window. Selecting and removing levels in the hierarchy only affects what is displayed, no change is made to the internal data and no data is retrieved from the remote Queue Managers.

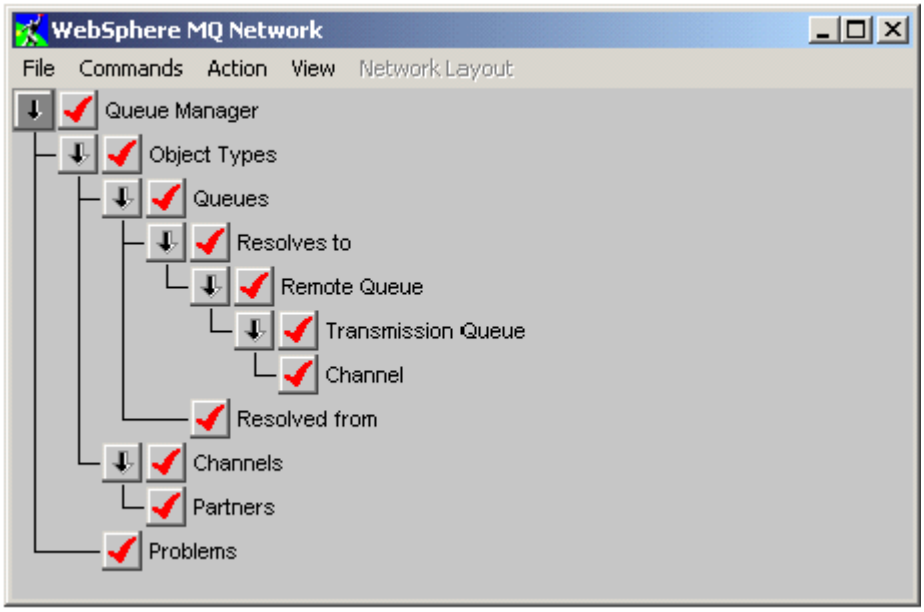


Figure 17: Network Levels Display

[10] A preference option is available which will just display a blank rather than a cross.

Problem Selection

- This window displays the complete list of problems the application tries to detect (see “Appendix B:Problem Selection List” on page 111 for the current list). By default all problems will be checked. However, by setting the icon to a ‘cross’ next to a problem the application will no longer check for this situation. If no problem detection at all is required a single check box in the Preferences Dialog allows all checking to be switched off (see “Preferences Dialog” on page 78 for details).

The list of problems is clearly an area which could be expanded in the future. I would be interested in hearing anyone's suggestions for problem checks. In particular I am interested in problems which are not easily seen from a single object definition but rely on the interaction between two or more objects.

Network Names

Each location definition has a field to specify a comma (or space) separated list of Network Names. The names themselves can be anything which makes sense for the particular Queue Manager organisation. For example, you could have each location be part of three network groups. The first identifying the type of machine, the second identifying the geographical location, and the third identifying the business function. So, we could specify a value of **‘iSeries, MidWest, DataCenter’**. Each name can contain any alphanumeric character but not a space since a space indicates the start of the next name.

Each unique name is displayed in the Network Names window together with a check mark. Selecting the check mark will then either include or exclude all Queue Managers with that network name from the network view. This allows easy inclusion of many Queue Managers in a single mouse click if analysis and displays only required on a subset of the Queue Managers. In the example above, for instance, it would be easy to only display the machines located in the MidWest or only the iSeries machines.

In addition to the defined network names the standard Network Name of ‘All’ is always available which, as its name suggests, applies to all Queue Managers.

Note that Queue resolution and problem analysis applies to only those Queue Managers actually in the network view. Consequently if some Queue Managers are excluded full path and channel resolution is not possible. This may lead to problems being reported, for example, “No partner channels found”, which are caused purely by the resolving Queue Manager not being included in the analysis.

Chapter 8. General Actions

Printing

The main window and the object dialogs have a *'Print...'* menu item allowing the user to send the information in the dialog to the printer.

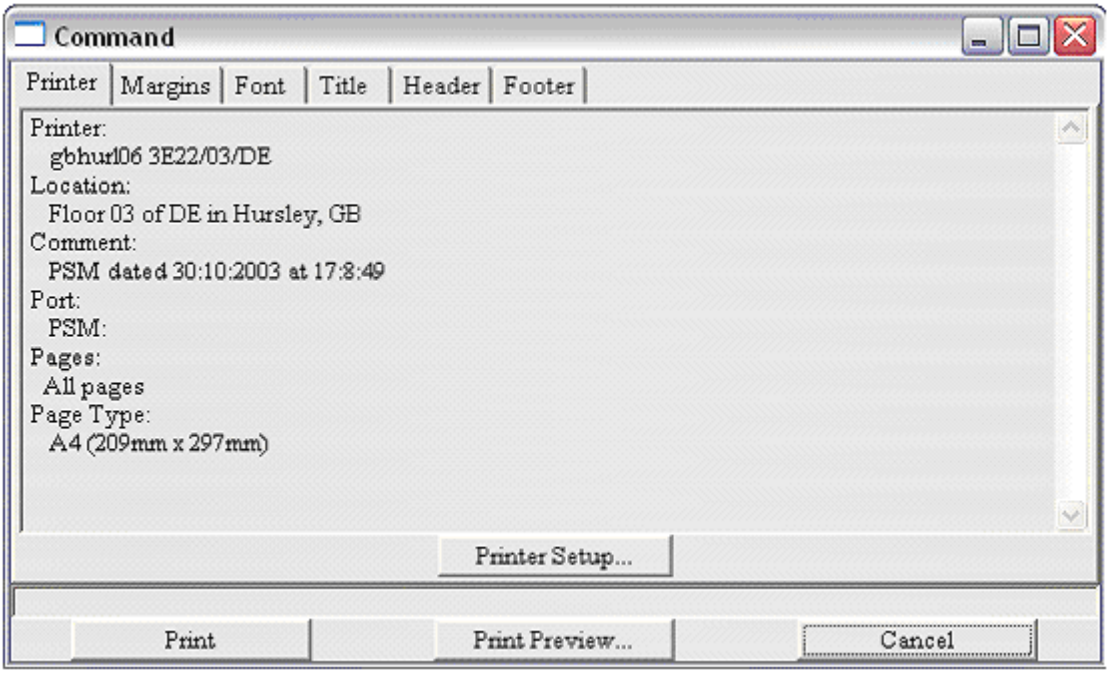


Figure 18: Print Dialog

Selecting the *'Print...'* menu item will present a print dialog box. This dialog displays the general parameters which will be used for printing.

- **Printer Details**
  Displays information about the selected printer and paper size
  To change the printer and the general values chosen you can select the 'Printer Setup...' button.
- **Printer Setup**
  Pressing this button will bring up the system printer dialog. The exact format of this dialog is system dependent. However, it should allow the user to select the printer, the paper size, the paper orientation, pages to print and probably a whole host of printer specific options.

  On some systems the button to apply the changes in the printer dialog is 'Print'. No printing will actually take place when this button is pressed. The printer values will be stored and returned to MQMONNTP. Only when the user presses 'Print' in MQMONNTP's print dialog will any data be sent to the printer.
- **Print Preview**
  Pressing this button will display a simple window showing what the print output would look like with the current settings. This allows the user to modify various values like margins, font size, paper orientation and line spacing to ensure the data fits neatly on the page. As values are changed in the main print dialog the preview will change immediately showing the effect of the value[11]. The print preview menu allows the user to select how many pages to display and what the size of each page is relative to the size of the window.
- **Print Button**
  Only when this button is pressed will any data be sent to the printer. The button will change to a cancel print button which allows the user to cancel the print operation if necessary.

[11]On Windows 95,98,Me the preview window is a monochrome bit map so the print colour will not be reflected in the displayed output. This is due to a bit map size limitation on these platforms.

The rest of the options are selected in various Tabs.

Margins

This Tab will show graphically the printable area of the page. There are two types of setting:-

- **Margins**
  These fields allow the user to set the vertical and horizontal margins on the paper. Values are in millimetres.
- **Overlap**
  When printing large dialogs it could be that to print all the information the output will need to be spread across multiple pages. In this case the program prints each page as though it were part of one large logical piece of paper. After printing the pieces can then be placed side by side to see the whole output.

  If multiple page output is required the Overlap values tell the program how much of an overlap is required between each page. An overlap ensures that a relatively seamless join can be made between the multiple sheets. Overlap values are specified in millimetres.

Font

- **Choose Font**
  This button will display a dialog allowing the user to select the required font for printing. This font is not related in any way to the fonts selected for dialog screen display. The currently selected font is displayed in the box to the left of the Choose Font button.
- **Line Spacing**
  This field allows the user to change the spacing between successive lines in the print output. The value is expressed as a percentage of the currently selected font height. For example a value of 50 would leave a gap of half (50%) of the height of the font between each line.

Title, Header, Footer

These Tabs allow a user specified string to be printed along with the printed text. The title is printed once, across all the top most pages. Headers and Footers are printer at the top and bottom of each page respectively.

It is useful to be able to put values rather than just a known string at the top and bottom of the page. For example, you may wish to print out the date and time as a footer. To allow this, inserts may be specified in the text. These are as follows :-

Insert	Meaning
%m	Queue Manager Name
%l	Location Name
%d	Day short form e.g. Sun, Mon, Tue
%dd	Day e.g. Sunday, Monday, Tuesday
%D	Day of the month e.g. 16
%DD	Day of the month e.g. 16th
%M	Month number e.g. 11
%MM	Month short form e.g. Jan, Feb, Mar
%MMM	Month e.g. January, February, March
%n	Page number
%N	Total print pages
%Y	Year short form e.g. 03
%YY	Year e.g. 2003
%T	Time e.g. 22:18
%TT	Time e.g. 22:18:35

Not all of these values are available all the time. For example the Queue Manager may be meaningless for some printouts. Suppose one wished to have a header that printed

Queue Manager: QMNAME

This would be fine for printouts that had a Queue Manager but on the others you would just get :-

Queue Manager:

This is not very desirable. We need a way of switching off text if a variable is not present. The answer is to enclose the text in brackets.

By specifying a header of '%m{ Queue Manager :%m}' we are saying only print out the contents of the brackets if you have a value for %m.

Printing Lists

When printing lists the application will print titles for the fields at the top of the upper pages. The titles shown will honour the ‘List Titles’ choice made in the list dialog window. However, there can be a slight difference with the ‘Auto’ setting. The ‘Auto’ list title setting tells the dialog to use short or full titles depending on whether there is room on the dialog. In printing there isn’t quite the same concept of ‘room on the dialog’. For printing therefore ‘Auto’ means

- **Full title** if field data length is greater than full title length
- **Short title** if full title is greater than the all the data lengths

Exporting Definitions

I was asked for an option to write the displayed dialog information to a file. There are essentially two reasons why this may be useful.

1. You want to embed the information into some other document
   In this case the information needs to be exported in human readable text form.
2. You want to capture the information so the object can be defined on another Queue Manager
   In this case you really want the data exported in MQSC 'define' format.

On most of the dialog displays an export option is available in the pop-up menu. The dialog displayed allows the user to select the type of export, a file name to export to if required and control what type of data should be exported.

- **Export Type**
  As described above this option allows the user to choose between straight text format, comma separated values (CSV), XML or 'MQSC DEFINE' format
- **All Objects**
  When using export from a list the default operation will export only those objects selected in the list. However, by checking the 'All Objects' box all entries in the list will be exported.
- **Export to File or Export to Clipboard**
  A choice is presented as to whether the data should be exported to the file name given below or directly to a clipboard for pasting into another application.
- **File Name**
  The name of the file the program will write to
- **Overwrite**
  When checked the program will overwrite any previous version of the file. Without this option checked the program will prompt the user for an overwrite decision if the file already exists.
- **Append**
  By default the program will overwrite any existing file. However, with the append box checked the data is written to end of any existing information in the file.

Export MQSC format

When data is exported only the data currently contained in the dialog is exported. This can be really useful when exporting text and only certain fields of an object should be exported. However, care should be used when exporting lists in MQSC format. Because only the data in the dialog is exported it is quite possible to export an invalid MQSC definition. To ensure a complete definition is exported select *'Alter List'* in the list window first. Ensure that all fields are in the left hand window i.e. in the **List Contents**. Select *'OK'*. This will retrieve all information about all objects. You can now export the objects from the list knowing that all data for all fields is available. Alternatively, if all objects of a particular type are required you can export them from the location dialog directly. Please see " Location Dialogs" on page 88 for more information.

Chapter 9. Queue Manager Monitoring

There are three types of Queue Manager monitoring:-

- **Direct Connection**
  If MQMONNTP is directly connected to the Queue Manager either via local bindings or via a client connection then a message is sent to the reply queue and then immediately got back again.
- **Normal monitoring**
  A program on the receiving Queue Manager should send the monitor messages back to the monitor program. In other words, it should use the Reply Queue and Reply Queue Manager of the monitor message. It must set the Reply Queue and Reply Queue Manager to its own values. In addition, it must change the first character of the message from 'm' to 'r'.
- **Loop back**
  In loop back monitoring the monitor queue at the remote location must 'point' back to the Reply Queue defined in the local Queue Manager's location settings. Using this method it is not necessary to have any code other than the base WebSphere MQ product on the target system which allows very simple monitoring of any WebSphere MQ platform. Since you must define a remote queue at the remote location pointing back to the Reply Queue you can not use loop back monitoring if your reply queue is a model queue.

Using any of the methods will cause a WebSphere MQ message to make a complete round trip between the monitoring program and the target Queue Manager. The application keeps track of when it needs to send a message and whether a reply is outstanding. If a message is not received within a reasonable time then the application will give a visible and, if required, an audible alert that a Queue Manager is not responding. In this way it is simple for an operator to check whether all of the Queue Managers and the Channels between them are functioning correctly.

Command Strings

A command string is a command you wish MQMONNTP to issue to the operating system. The command is always submitted as a background process. The actual format of the command and what capabilities you have depend on the OS.

The two places where you can specify a command string are :-

- **In the location dialog**
  Here you are given the opportunity to give two commands. The idea is that, during monitoring of a remote location, monitoring will either succeed or fail. As it makes the transition from one to the other the appropriate command will be issued. This allows the user to have a downstream application which is driven from MQMONNTP which takes some action when it is told that, for instance, monitoring to a remote location is now failing.
- **In the filter string using the system() function call**
  The system() function, in the filter, takes two parameters, an expression and a command string. If the expression is evaluated to TRUE the command will be issued. This allows the user to check for virtually any combination of object attributes and if they match a certain criteria then a command will be issued. Note that the system() function is executed for each item in the list returned. So, for example, if a filter of system(1,"myprog.exe") is used on a queue list then as many instances of myprog.exe will be started as there are queues in the list. You should use the expression to make sure that the program is only started as many times as necessary.

Format

The format of the command string is whatever the underlying OS supports. Essentially it should be a program name followed by zero or more parameters which are governed by the program. The Administrator will treat everything up to the first space as the program name and pass everything else as parameters. I have had varying degrees of success starting REXX and batch files. Sometimes it is necessary to start the command interpreter (for example

CMD.EXE on Windows) and pass the command you want executed as the parameter to it e.g. “cmd /c rexxcmd”. Remember that if you need embedded double quotes in a string you can precede it by a backslash.

Substitution Characters

A substitution character is a way of simply modifying the parameters passed depending on the object generating the command. Substitution will occur anywhere in the command string. The following characters are supported:-

%l : will be replaced by the location name of the location

%m : will be replaced by the Queue Manager name of the location

%o : will be replaced by the object name in the list. If the command being issued is not related to a list then it will be ignored and removed from the command string

Examples

"myprog.exe"

"myprog.exe -m OS2PGC1"

"myprog.exe -t "Failure for object %o on Queue Manager %m"

Chapter 10. Event Monitoring

The previous chapter dealt with monitoring Queue Managers as a whole and detecting whether they were reachable. However, even though the Queue Manager is available and working it does not necessarily imply that all is well within it. What we need is some way to monitor the individual objects of the Queue manager, most notably Channels and Queues. To facilitate this WebSphere MQ Queue Managers have the notion of event messages. Event messages are specially formatted messages which are written to 'known' queues when particular key events occur within the Queue Manager. For example, an event message can be generated when the depth of a queue reaches a certain predetermined level. Why, I hear you ask, rather than MQ just raising a message doesn't the Queue Manager just fix the problem? The point here is that different installations may want to perform different actions for the same event; the same event for different objects may well be handled differently; and for a number of these situations it is not at all clear what 'the right' thing to do is. For example, in the case earlier where a queue has reached a certain depth, is the right thing to do just increase the maximum depth of the queue; should one inhibit the queue for puts to prevent any further messages from being put to it; or should one start another instance of the application which is supposed to be servicing this queue. The answer may well depend on what the queue is and what the installation policy is. So it is an advantage that MQ does not enforce a certain policy for these situations and instead generates an event message which provides the most flexible method of informing an application that something has happened.

The downside, of course, is that something has to be written to process these event messages and do something with them. For some time now I have been having more and more requests for this type of functionality to be put into MO71. The problem is that I too am faced with the problem that what each person wants to do may well be different. This is a big area and there are many companies who have developed entire products whose sole purpose is to monitor and act upon these events. Sadly I have neither the resource nor knowledge to generate a product which integrates with other monitoring tools. What I have attempted to do is come up with a fairly simple but flexible way of monitoring for these event messages and acting upon them.

Perhaps the best way of describing monitoring is to do an example.

Event Example

Let's take the case above of a queue reaching a predefined limit. First we determine what the MQ event message is that will be generated and which queue it will be written to. To do this we look at the *'Event Monitoring'* book and find the chapter on queue depth events. We see that there is an event *'QueueDepthHigh'* which is written to the ***SYSTEM.ADMIN.PERFM.EVENT*** queue. Note that we need to switch on performance events for the Queue Manager and the High Depth Event for this queue. So, now we should have the Queue Manager generating this message, all we have to do is configure MO71 to look for it.

From the main menu select *Action-Event* this will bring up a dialog with quite a number of fields which we'll ignore for now. For the moment just concentrate on the fields we must fill in.

Queue Manager Name

Clearly you place in here the name of the Queue Manager, which must be a location already defined to MO71, i.e. you have a location for this Queue Manager on the main window.

Event Queue Name

For now let's just put the name of the queue we're monitoring **'SYSTEM.ADMIN.PERFM.EVENT'**

Type

We now need to tell MO71 what event we're looking for. Press 'F4*'* to drop down the list. Select 'Queue High'.

The rest of the fields are mainly concerned with what we want to happen when we see one of these messages. Let's keep things simple and just write a message to the console. Tab down to the 'Console Priority' field and change 'None' to, say, 'Medium'.

Now press the create button and check the application says an event has been created. The message may well say it has created *2* events. We'll see the reason for this soon.

Now go back to the main window but this time select 'Action-Event List'. This will show a familiar looking list window, pressing refresh will show the event we've just defined. Notice though that it only shows the Queue Name and not the actual event type. By changing the 'Display Type' to 'Detail' and pressing refresh we see the event type we created. We also see another event, called 'Default'. This explains why the message said 2 events created but what is it for? The default event is there because the Queue High Event we're looking for isn't the only message which may be put to this event Queue. If MO71 reads a message off the queue and it is not a Queue High Event what should it do with the message? This default event allows you to specify that behaviour. But for now, in time honoured tradition, let's ignore it.

The other thing we notice on our event list display is that the status is stopped. What does this mean? Well it means that MO71 is not at this moment reading this queue. Any messages put to the event queue will not be noticed. So, we need to start the event, select either of the events for this queue, right-click and press start. You should see the status change from Stopped to Running. If you don't there should be a 'status reason' field stating why it didn't work.

Now the exciting bit, we get to see whether it actually works. We need to cause a Queue High Event. Choose a queue which has a fairly low value for 'Max Q Depth' and 'Depth High Limit' and put messages to it. Using something like my SupportPac MA01 which allows you to specify the number of messages to put can help here but you can use MO71 itself by selecting *'Action-Put Message'*. Put a bunch of messages to the queue and keep watching the event list display.

You should see the count of events processed change to 1 as MO71 receives and processes the message.

Now, if you remember this far back, when we defined the event we changed the console priority to 'Medium'. What this said was that when one of these messages was received MO71 should write a message to the console. So let's go see.....

From the main menu select *'View-View Console'*. You will probably see an info message telling you the time MO71 started and now we also see *a 'Queue High'* message stating that whatever queue name you chose has reached its *Queue High* limit.

Now let's see what happens when the queue depth becomes low again. Use MO71 to browse the queue. Select 'Message Selection-Apply to all messages'. The 'Delete' button now changes to 'Delete All'. Press the 'Delete All' button. Now it we take a look at the console you see we still have a record there saying Queue High even though we know it's empty. There are two things we forgot to do; firstly if we've enabled Queue High events for the Queue it probably makes sense to also enable Queue low events. Secondly, in our event list we need to add an entry to look for Queue Low. So, go and do that now.

Right, hopefully now you have a event list which shows three entries, Queue High, Queue Low and Default. Let's try the same exercise. Use the 'Action-Put Message' to put a bunch of messages to the queue. You should see the event list for Queue High now read 2. Let's take a quick look at the console. That's strange; we only see one record for Queue High. Why is this? The reason is the 'Console Type' we specified when we defined the event. We set 'cumulative'. What this says is that the console should only display the latest entry if there are two events of the same type, for the same object from the same Queue Manager. If you like you can change this to 'History' which will write a new entry to the console every time it happens.

Anyway, now we want to go and delete the messages on the queue. So do the same trick of 'Delete-All' from the browse window.

The first thing to notice is that in the event list we now correctly have a count of one next to Queue Low event. The second thing to notice in the console window rather than getting a console entry saying 'Queue Low' event instead the 'Queue High' event has gone. Why is this? Well for pretty much the same reason we didn't get two Queue High events. Since we've asked for a cumulative console we're asking for it to display the sum of the events that have occurred. Since a Queue Low cancels a Queue High event it seems logical not to display anything. Most events do not have an obvious opposite but there are a few such as Channel Started/Channel Stopped and Channel not Activated/Channel Activated that operate in the same way.

By now you should understand the basic principals of event monitoring so let's go through all the fields you can specify on an event object.

Event Fields

The following fields are available on an event dialog.

Queue Manager Name

In its simplest form this is just the Queue Manager name for which this event applies to but it can take wildcards. '*' implies one or more characters. '?' implies a single character. This can be useful for defining an event on all Queue Managers starting with the string 'NT*' for example.

Location

This field is largely for convenience but it can useful in some circumstances where you have more than one location defined for the same Queue Manager.

Type

This is the event type the event is monitoring for. You cannot define a 'default' event nor can you delete the default event unless it is the only event for a particular event queue.

Filter

There may be occasions when you may want different actions depending on the actual contents of the event message. The most likely of these would be the object name that generated the event. For example, suppose you wanted to only write console messages for channels starting with the string 'NT' you could specify a filter of **'channel == NT*'** For a discussion of what can be specified in filter please see “Filtering” on page 18.

Filter Priority

This field can be used to force the correct ordering of events. Suppose you have two events defined for catching the Queue High event. You may have a filter on one of them for just queues called 'ABC'. The second event may have no filter as a catch-all mechanism. For this to work you need to ensure that the filtered event is processed first. To achieve this define the filtered event with a filter priority which is higher than the general event i.e. priority 2 filters are processed before priority 1.

To enable you to make easy additions later it may be a good idea to define your filters with priority such as 10,20,30 and so on so that you can insert another filter between them if required.

Discard

This value essentially switches off the event definition. If Discard is set to 'yes' the event is essentially disabled. Any other events which match the event, including the default filter, will still be processed.

Command

Here you can type the name of a command to be issued if this event is driven. For example, it might run a script which calls the operators pager.

The following inserts are available:

- %m Queue Manager Name
- %l Location Name
- %o Object Name
- %r Reason Code
- %R Reason Code String e.g. ”Queue Full”

Log File

If required a hard copy record of the events which have been caught can be written to file. This can be useful as an audit trail or perhaps to post process and see how often certain events are being generated. Enter the full path of the file name to write to.

Log Line

The format of each line in the log file.

The following inserts are available:

- %m Queue Manager Name
- %l Location Name
- %o Object Name
- %t Time 11.14.12 format
- %d Date in 2003-11-23 format
- %r Reason Code
- %R Reason Code String e.g. “Queue Full”
- %([\n] [\t] [\m] [*] [MQSC Name])
  where
  - \n prints a new line between each attribute
  - \t precedes the attribute with its name (or title)
  - \m precedes the attribute with its MQSC name
  - * prints all attributes of the message
  - MQSC name prints only the given attribute

Requeue

One of the problems with processing event messages is that there may be occasions where you want more than one program to monitor events. The simplest way of achieving this is to daisy chain the monitors. Have one monitor processing the 'real' event queue and when it's finished it then puts the message to another queue for another program to then process. This field allows you to specify where MO71 should put the message once it has finished processing it. If no destination is specified the message will be discarded.

Log Event

This field is currently not supported.

Auto Start

As its name suggests it allows for the event monitor to reading the queue as soon as MO71 is started. If set to **'No'** the event must be started manually. Note that starting any event for the same queue will start all of the events defined for that queue. It is not possible to only start one.

Console Priority

This field tells MO71 whether the event should be logged to the console. A value of 'None' implies that no console entry will be made.

Console Type

Some events have an equivalent but opposite event. For these events it can make sense to allow them to cancel one another out so that only the negative situation is ever presented on the console. The value for this is 'cumulative'. If however you want every instance of this event to be logged to the console you can specify 'History' so that a history of all of these events is maintained in the console.

Operational Considerations

In order to monitor event messages MO71 must have a connection to the Queue Manager and open the event queue. It is written to spend its whole time sitting in an MQGET. Consequently the more event queues which need to be monitored the more connections are required to the Queue Manager. Different queues can be useful if you have one type of monitoring application monitoring one set of events and another monitoring a different set. However, if you have a program, such as MO71, which can deal with different events from different queues it is far more efficient to merge the event queues. The simplest way of doing this is to redefine the event queues as alias queues pointing to a central event queue. A simple MQSC script to do this would be :-

```
delete ql(SYSTEM.ADMIN.PERFM.EVENT)   purge
delete ql(SYSTEM.ADMIN.QMGR.EVENT)    purge
delete ql(SYSTEM.ADMIN.CHANNEL.EVENT) purge

define ql(GENERAL.EVENT)

define qa(SYSTEM.ADMIN.PERFM.EVENT)   targq(GENERAL.EVENT) replace
define qa(SYSTEM.ADMIN.QMGR.EVENT)    targq(GENERAL.EVENT) replace
define qa(SYSTEM.ADMIN.CHANNEL.EVENT) targq(GENERAL.EVENT) replace
```

Once this script is run all events will be pointed at the general event queue 'GENERAL.EVENT' and only a single connection to the Queue Manager is required. This mechanism could even be extended to collect events from remote queue managers to a single, central event queue, by defining the event queues as remote queues.

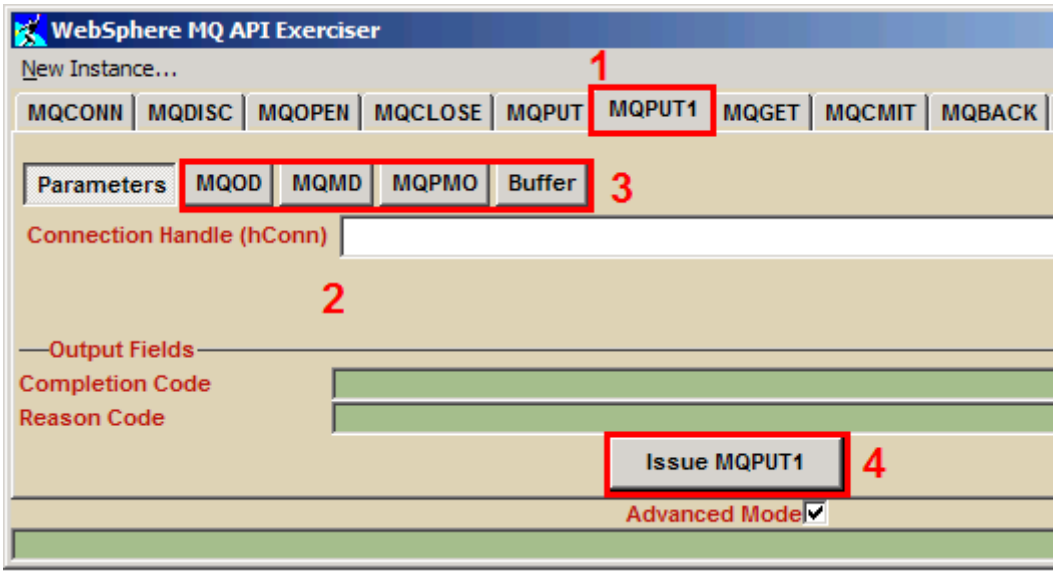
Chapter 11. API Exerciser

The API Exerciser is provided to allow the user to make API calls through a graphical interface making testing of specific options and details, and inspection of reason codes generated, quick and easy prior to writing the code for an application. Clearly it is not an appropriate tool to write an application for a production system, but is excellent for trying things out in a development or sandbox arena.

You can start the API Exerciser from the main MO71 window, via the Action menu, or by pressing F9.

Main Layout

Each MQ API verb is provided on a tab.



1. Switch to the tab for the verb you need
2. Fill in the appropriate fields within that tab
3. If there are structures associated with the verb, there will be a section for “Parameters” and a separate section for each structure. Click on each of the structures and fill in appropriate fields within the structure.
4. Press the button to issue the verb.

There are a number of points that are common on each tab, regardless of which verb is it used for.

Each tab will have a button labelled, “Issue MQ*verb*” which you press when you are ready for the verb to be issued using all the information you have provided in the various fields within the tab.

This button, along with the Output Fields for the verb (for example Completion Code and Reason Code) are shown at the bottom of the window whichever structure you are looking at.

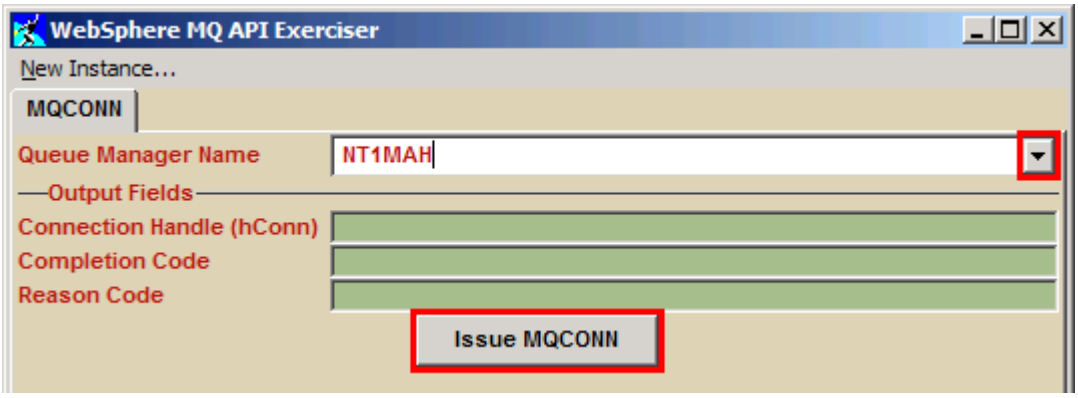
There is an Advanced Mode check box at the bottom of the window. When this is checked, all verbs and options are available to you to try even if they don’t make sense. More details later.

There is a status bar at the very bottom of the window where messages will be written to indicate any obvious errors detected prior to the user pressing the “Issue MQ*verb*” button. For example, unknown structure version numbers will be indicated here.

Walk-through

As an example, we'll step through a few of the verbs to illustrate how the exerciser is used. We will assume that Advanced Mode is not checked throughout this walk-through.

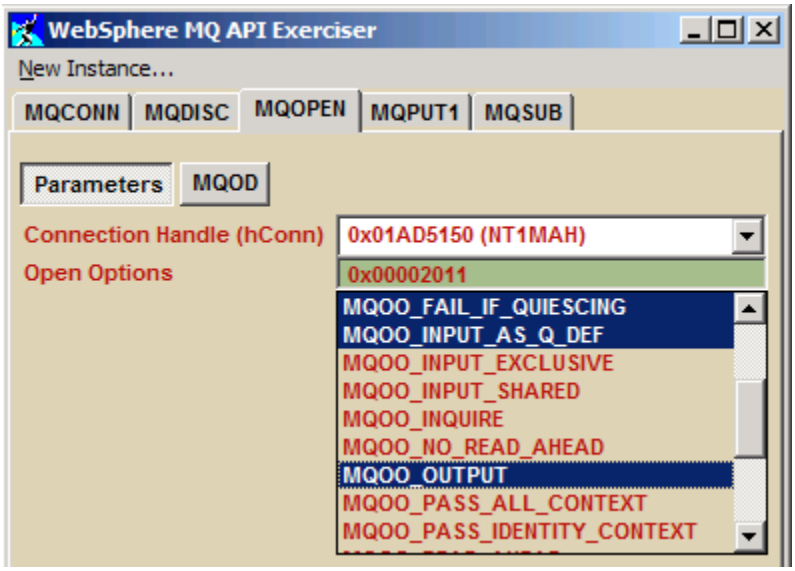
MQCONN



Starting on the MQCONN tab (which to begin with is the only one available), choose a queue manager name from the drop-down which contains all the queue managers defined to MQMON. Having chosen your queue manager name, press the "Issue MQCONN" button and you will see the output fields are filled in, with a completion code and reason code. If you have successfully connected to the queue manager, a connection handle will also be returned to you. You will see that there are a number of additional tabs now visible.

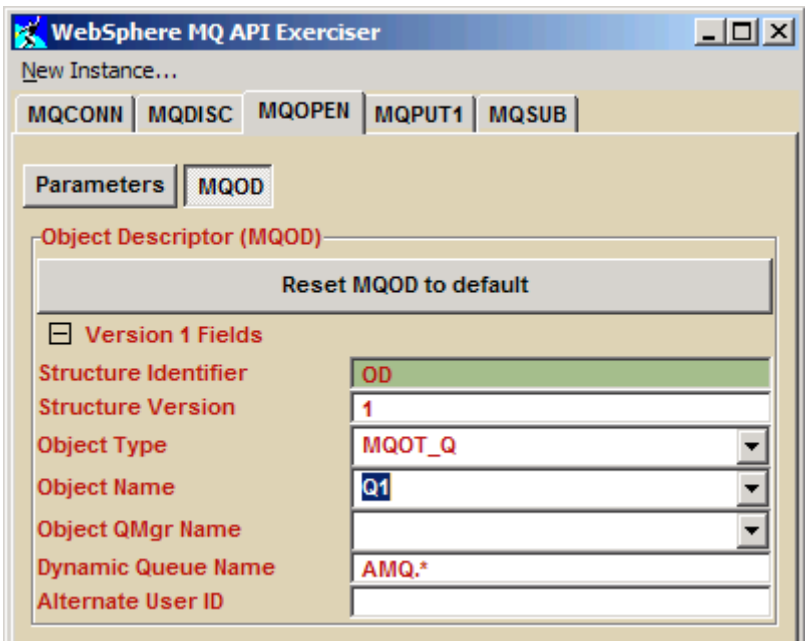
You can double click on the queue manager name field when it is populated with a queue manager name and MQMON will display the queue manager attributes dialog.

MQOPEN



Now we switch to the MQOPEN tab. We can see that the connection handle that was returned to us on the MQCONN call is already chosen for us on the MQOPEN tab. If you make several connections to queue managers, this drop-down box allows you to choose the connection handle for the queue manager you want. The API Exerciser will label your connection handle with the name of the queue manager it is for to aid you. You will see this name in parentheses after the connection handle hexadecimal value.

Select the open options you wish to use, holding down the <CTRL> button in order to select more than one. You can see the total number being added up for you in the field above the list of options. Keep an eye on the status line at the bottom of the window. If the API Exerciser detects a combination that is going to fail with MQRC_OPTIONS_ERROR it will provide information there.



Before clicking on the "Issue MQOPEN" button, we will add some information in the Object Descriptor.

Object Descriptor (MQOD)

Now click on the MQOD button to provide fields in the Object Descriptor structure. This

structure comes pre-filled in, in the API Exerciser, with the fields that would be in the C-structure MQOD_DEFAULT if you were writing a C program. In the simplest case, all you need to do here is select an Object Name from the drop-down list and press the “Issue MQOPEN” button.

There are of course more fields that you can optionally use, and longer versions of the Object Descriptor which you can view by editing the Structure Version entry field.

If you make lots of changes to the Object Descriptor and then want to start again, you can press the “Reset MQOD to default” button which will reset the fields to the values from the MQOD_DEFAULT C-structure again.

You can double click on the Object Name field and MQMON will display the object details dialog. If you have typed in the name of a queue that doesn’t exist (rather than selecting it from the drop-down list) this will bring up an empty dialog allowing you to define the queue through MQMON. If you choose an Object QMgr Name as well, and it is a different queue manager to the one you are connected to (i.e. you are opening a fully qualified remote queue), the queue attributes dialog displayed will show you the local queue definition on that remote queue manager (so long as it is known to MQMON).

At the time of writing, the fields in a version 2 Object Descriptor are not supported by the API Exerciser. Those in versions 1, 3 and 4 of the Object Descriptor are supported.

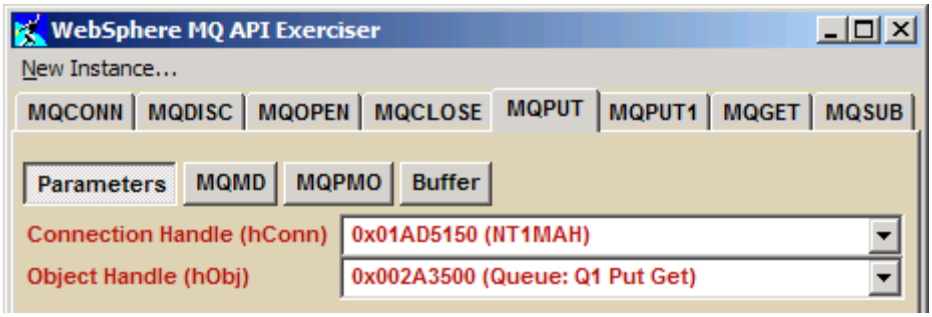
Using other connection handles

If you have made a number of MQCONN calls, the Connection Handle drop down box will contain all the hConns. The most recently created hConn will be pre-selected for you, but you can choose any other as required.

Opening other object types

Of course you can issue an MQOPEN call on objects other than queues. When doing this, I suggest first filling in the Object Descriptor with the Object Type and Object Name, then switching back to the Parameters section where only those options appropriate for the Object Type you have selected will be shown. This will make it easier to select the options you need.

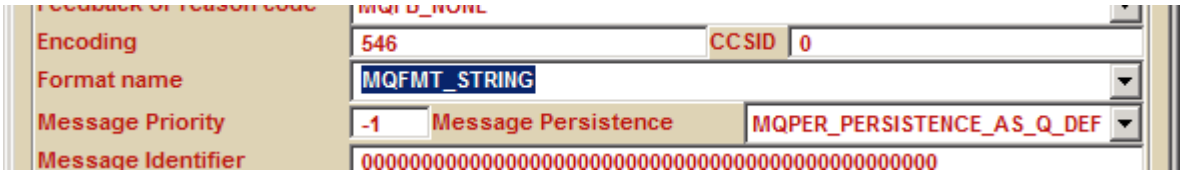
MQPUT



If the object we opened in the previous step was a queue and we opened it with the MQOO_OUTPUT option, the MQPUT tab will now be available to use. Switching to that tab we see the object handle that was returned to us on the MQOPEN call is already chosen for us on the MQPUT tab. If you open several queues (or topics) for output, this drop-down box allows you to choose the object handle for the object you want. The API Exerciser will label your object handle with the name of the object it is for, and the options it was opened with to aid you. You will see these details in parentheses after the object handle hexadecimal value.

Message Descriptor (MQMD)

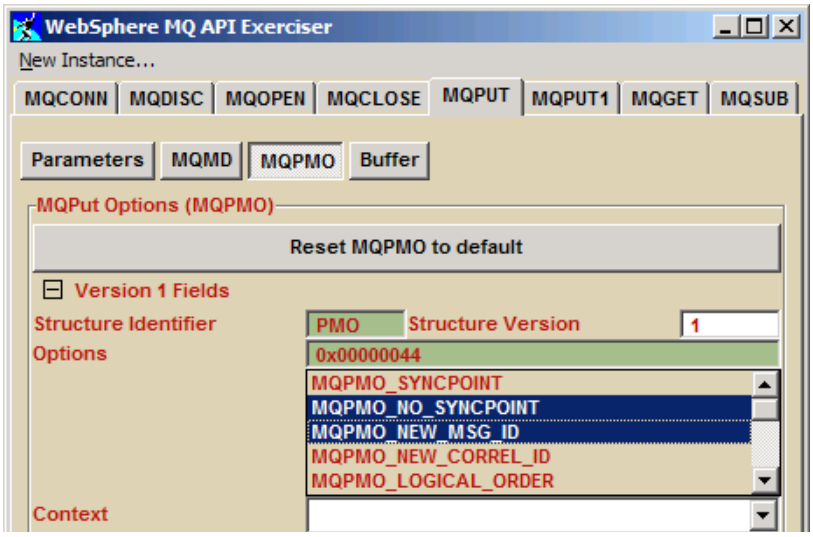
Now click on the MQMD button to provide fields in the Message Descriptor structure. This structure comes pre-filled in, in the API Exerciser, with the fields that would be in the C-structure MQMD_DEFAULT if you were writing a C program. We will make one simple change. We will choose MQFMT_STRING in the Format name drop down.



There are of course lots of fields that you can optionally use, and a second version of the Message Descriptor which you can view by editing the Structure Version entry field.

If you make lots of changes to the Message Descriptor and then want to start again, you can press the “Reset MQMD to default” button which will reset the fields to the values from the MQMD_DEFAULT C-structure again.

Put Message Options (MQPMO)



Now click on the MQPMO button to provide fields in the Put Message Options structure.

Select the put message options you wish to use, holding down the <CTRL> button in order to select more than one. You can see the total number being added up for you in the field above the list of options. Keep an eye on the status line at the bottom of the window. If the API Exerciser detects a combination that is going to fail with MQRC_OPTIONS_ERROR it will provide information there.

At the time of writing, the fields in a version 2 Put Message Options are not supported by the API Exerciser. Those in versions 1 and 3 of the Put Message Options are supported. Message Handles are not yet supported either.

Before clicking on the “Issue MQPUT” button, we will fill in our message buffer.

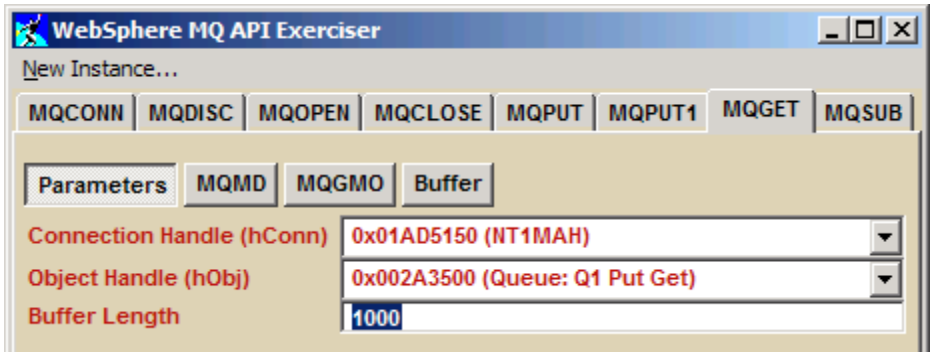
Message Buffer

Now click on the Buffer button and type your message text in the Buffer entry field. The Buffer Length field will be automatically filled in with the length of the text you type, although you may edit it to be longer or shorter if you wish.

Press the “Issue MQPUT” button, and check out the output fields that are now filled in, in your Message Descriptor (MQMD) and Put Message Options (MQPMO) structures when the verb is issued successfully.

MQGET

If the object we opened in the earlier step was a queue and we opened it with one of the MQOO_INPUT_* options, the MQGET tab will now be available to use. Switching to that tab we see the object handle that was returned to us on the MQOPEN call is already chosen for us on the MQGET tab. If you open several queues for input, this drop-down box allows you to choose the object handle for the object you want. The API Exerciser will label your object handle with the name of the queue it is for, and the options it was opened with to aid you. You will see these details in parentheses after the object handle hexadecimal value.



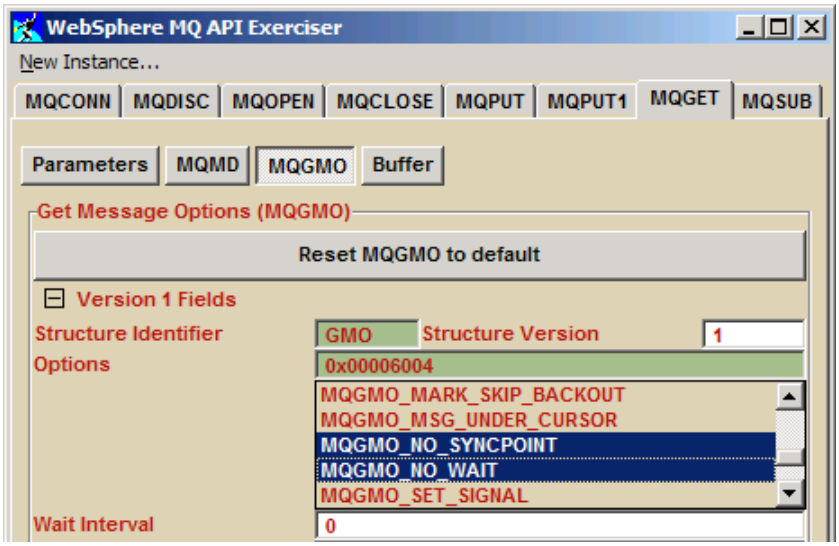
Fill in a value in the Buffer Length field. Before clicking on the “Issue MQGET” button, we will add some information in the Message Descriptor and the Get Message Options.

Message Descriptor (MQMD)

Now click on the MQMD button to provide fields in the Message Descriptor structure. This structure comes pre-filled in, in the API Exerciser, with the fields that would be in the C-structure MQMD_DEFAULT if you were writing a C program. The default Message Descriptor will do for our purposes this time. If this isn’t the first MQGET you have issued, you might want to press the “Reset MQMD to default” button which will reset the fields to the values from the MQMD_DEFAULT C-structure again.

Get Message Options (MQGMO)

Now click on the MQGMO button to provide fields in the Get Message Options structure.

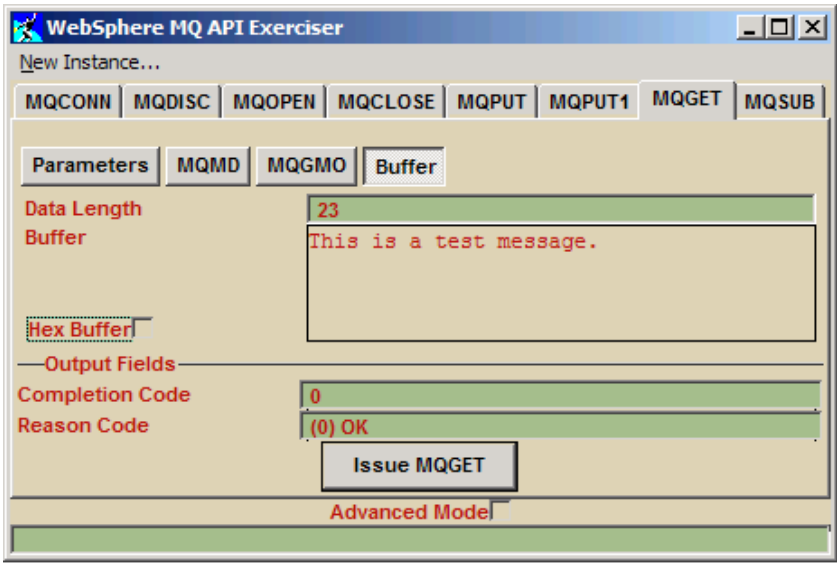


Select the get message options you wish to use, holding down the <CTRL> button in order to select more than one. You can see the total number being added up for you in the field above the list of options. Keep an eye on the status line at the bottom of the window. If the API Exerciser detects a combination that is going to fail with MQRC_OPTIONS_ERROR it will provide information there.

At the time of writing, the field in a version 4 Get Message Options structure (a Message Handle) is not supported by the API Exerciser. Those in versions 1, 2 and 3 of the Get Message Options are supported.

Message Buffer

Now click on the Buffer button and chose whether you want to display your message as a Hex Buffer or not. Press the “Issue MQGET” button, and view your message contents and check out the output fields that are now filled in, in your Message Descriptor (MQMD) and Get Message Options (MQPMO) structures when the verb is issued successfully.



Walkthrough with multiple instances

The API Exerciser can be used with multiple instances of the Exerciser window running at the same time. Either by starting one from the Action menu multiple times, or by selecting New Instance from menu bar on the first instance started. As an example of this, we’ll step through an example of subscribing in one window and publishing in another window.

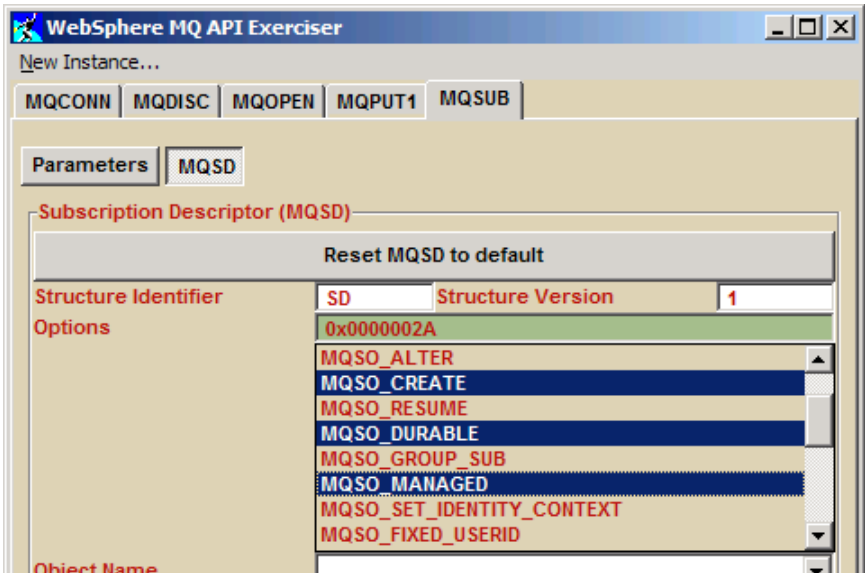
Start by connecting to your queue manager in each window.

MQSUB

Now we switch to the MQSUB tab. We can see that the connection handle that was returned to us on the MQCONN call is already chosen for us on the MQSUB tab. We are going to select the MQSO_MANAGED option in a moment, so we don’t need to select anything in the Object Handle field. Before clicking on the “Issue MQSUB” button, we will add some information in the Subscription Descriptor.

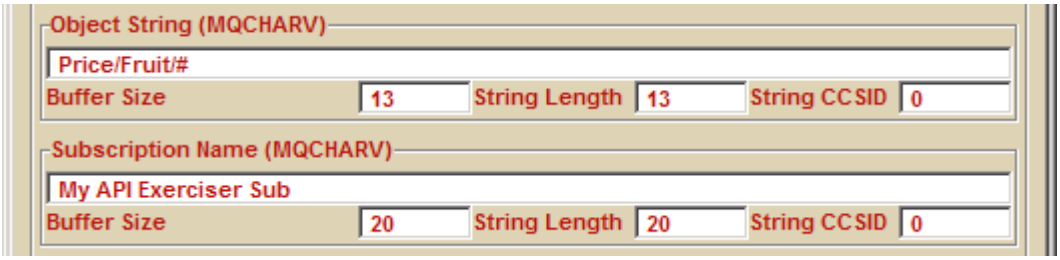
Subscription Descriptor (MQSD)

Select the subscription options you wish to use, holding down the <CTRL> button in order to select more than one. You can see the total number being added up for you in the field above the list of options. Keep an eye on the status line at the bottom of the window.



If the API Exerciser detects a combination that is going to fail with MQRC_OPTIONS_ERROR it will provide information there.

Fill in a topic string to subscribe on in the Object String field and, since we have chosen the MQSO_DURABLE option, also fill in a Subscription name.



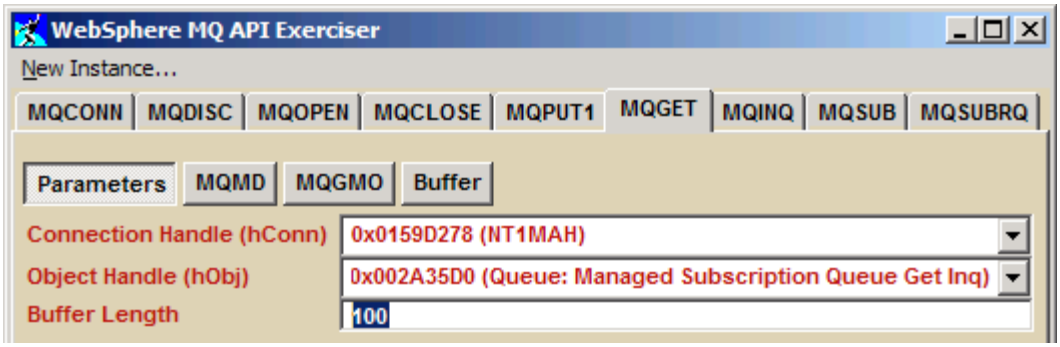
Press the “Issue MQSUB” button, and check out the output fields that are now filled in, in your Subscription Descriptor (MQSD) structure when the verb is issued successfully. You will also see at the bottom of the window that this verb has returned you two handles, an Object Handle and a Subscription Handle.

Using MQCHARV data types

A number of the fields in the MQSD structure (and in other structures) have a data type of MQCHARV. These are variable length structures. The String Length field will be automatically filled in with the length of the text you type, although you may edit it to be longer or shorter if you wish. If the MQCHARV field in question is an output field, you should provide a non-zero value in the Buffer Size field in order to be returned the contents on output of the verb.

MQGET from a handle returned from MQSUB

Now switch to the MQGET tab. We see that one of the handles that were returned to us on the MQSUB call is already chosen for us on the MQGET tab. Provide a Buffer Length, in the MQGMO, select MQGMO_WAIT and a WaitInterval of 60000 ms, and press the “Issue MQGET” button. We will come back to this window later.

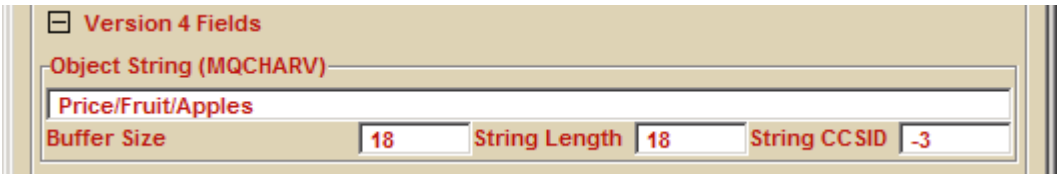


We are now going to use our second window to publish to this topic. We already saw an example of MQOPEN and MQPUT, so now we shall use MQPUT1.

MQPUT1

On the Parameters tab the Connection Handle we were returned from the MQCONN in this window is pre-filled in for us. Note that you cannot use Connection Handle from the other instance of the API Exerciser. Think of each instance as a separate application.

The MQPUT1 verb has three structures and you will see a button for each one, MQOD, MQMD, MQPMO and Buffer. You have seen all of these in the earlier walkthrough. Fill in an Object String in the MQOD structure that will match the subscription you just made (note my example used a wildcard #). You will need to change the structure version number to 4 in order to see these fields and change the Object Type to MQOT_TOPIC.



Provide any attributes you want in the MQMD (for example Format set to MQFMT_STRING) and MQPMO structures and fill in a message in the Buffer. Now press the “Issue MQPUT1” button.

So long as your wait interval of 60 seconds has not already elapsed on the MQGET, you should see the message immediately appear in the other window, if it has already elapsed you’ll have to press the “Issue MQGET” button again.

MQCLOSE

On the subscriber API Exerciser instance, now switch to the MQCLOSE tab. There are two choices in the drop down for the Object Handle, chose each one in turn, selecting MQCO_REMOVE_SUB for the subscription handle one, and press the “Issue MQCLOSE” button for each one.

The MQI Verb set

At the time of writing the API Exerciser does not support the full MQI Verb set. The table below shows which verbs are currently supported by the API Exerciser and which are not yet supported. Comments about support for specific versions of API structures are noted earlier in the chapter as each verb is discussed.

MQI Verbs provided in the API Exerciser	MQI Verbs not yet provided in the API Exerciser
MQCONN	MQCONNX
MQDISC	MQCTL
MQOPEN	MQCRTMH
MQCLOSE	MQDLTMH
MQPUT	MQSETMP
MQPUT1	MQDLTMP
MQGET	MQINQMP
MQCMIT	MQBEGIN
MQBACK	MQCB
MQINQ	MQSET
MQSUB	MQSTAT
MQSUBRQ	MQBUFMH
	MQMHBUF

Advanced Mode

When Advanced Mode is checked, all verbs and options are available to you to try even if they don't make sense. For example, you can attempt to issue an MQOPEN before having issued an MQCONN. This may be helpful when exercising verbs to get particular reason codes. When unchecked, the exerciser attempts to discourage you from doing anything that will obviously bring about an error. For example, the MQOPEN tab is not visible until you have issued an MQCONN verb.

We will detail the differences when running in Advanced Mode in this section.

MQCONN

In addition to choosing a queue manager name from the drop-down which contains all the queue managers defined to MQMON, you can alternatively type in a queue manager name and if it is unknown to MQMON, an attempt will first be made to connect to it using local bindings, and if that fails, a client connection will be attempted. If the queue manager name selected from the list is a "via" location, this pattern to attempt to connect will also be used. All MQCONNX style details, for example client channel definition or connection security parameters (MQCSP) are configured in MO71. The API Exerciser does not support the provision of any MQCONNX fields at this time.

Connection Handle fields

In MQOPEN, and indeed all verbs which have a connection handle as an input field, in addition to being able to chose the connection handles create by previous uses of the MQCONN verbs you have issued, you can also choose, "0xFFFFFFFF (MQHC_UNUSABLE_HCONN)" or type in any number yourself. These are of course only useful for testing error conditions!

Options fields

In advanced mode, all possible options for a particular verb are shown, even when some of them are not appropriate, for example, they will cause MQRC_OPTION_NOT_VALID_FOR_TYPE if used. Additionally, you may type in an options numeric total in the entry field above the list box instead of selecting the options you want from the list. This will highlight in the list below the field which options are chosen from the number you typed in.

Troubleshooting

Why are my object name drop-down lists empty?

In order to populate the drop-down list with relevant queue (or other object) names, the API Exerciser first needs to know the queue manager name. If no connection handle is selected, or the MQHC_UNUSABLE_HCONN is chosen, then there is no context within which to populate these drop-down lists.

Why is the object handle I just opened not in the drop-down list?

There are two main reasons for this.

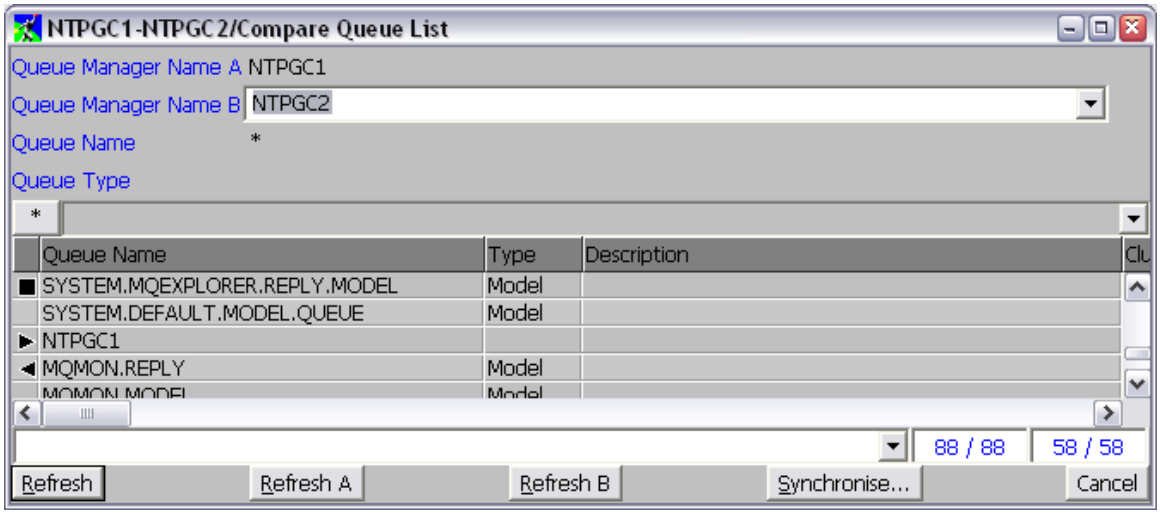
1. If the tab you are on has no connection handle selected or has a different connection handle selected, then the object handle will not be in the drop-down list until the correct connection handle is first selected.
2. If the object handle was made without MQOO_OUTPUT, it will not show up in the MQPUT object handle drop-down list since it is not valid for use with that verb. Using advanced mode will allow all object handles for this connection – even those with incorrect options - to show in the drop-down list.

Why is the queue I just defined not in the drop-down list?

The drop-down lists of object names are populated from MQMON's database of information about the queue manager, rather than going to the queue manager and requesting a list of queue names at the point when the drop-down list needs them. This means that you will need to tell MQMON to refresh it's database of object names if you define any new queues, in order for them to show up in the list (remember that you can also just type in the queue name too). In order to refresh this database, go to the main MQMON window, select your queue manager from the list, and press File->Refresh Information (or <CTRL>+<R>).

Chapter 12. Comparing Queue Manager Objects

By selecting the *'Action-Compare Definitions'* main menu item from the main window a dialog is displayed which allows the objects from two different Queue Managers to be compared. This dialog operates in a similar way to the normal object dialogs and most of the normal dialog operations are available. However, there are a few key differences.



The first difference is that the dialog itself is not permanently tied to a particular Queue Manager. Instead there are two Queue Managers involved and these must be entered at the top of the dialog. The first Queue Manager is referred to as Queue Manager 'A' and the second one as Queue Manager 'B'. Nothing will be displayed until the dialog contains data for both Queue Managers. Note that the action *'Refresh'* will refresh **both** Queue Managers, actions are provided to refresh the data for just one Queue Manager as well.

Another difference is that the first column is an indicator of where the object is defined and whether the definition differs between the two Queue Managers. The icons are:-

- Left pointing triangle — Object is only defined on Queue Manager A
- Right pointing triangle — Object is only defined on Queue Manager B
- Black Square — Object is defined on both Queue Managers and they are different
- <Nothing> — Object is defined on both Queue Managers and they are identical

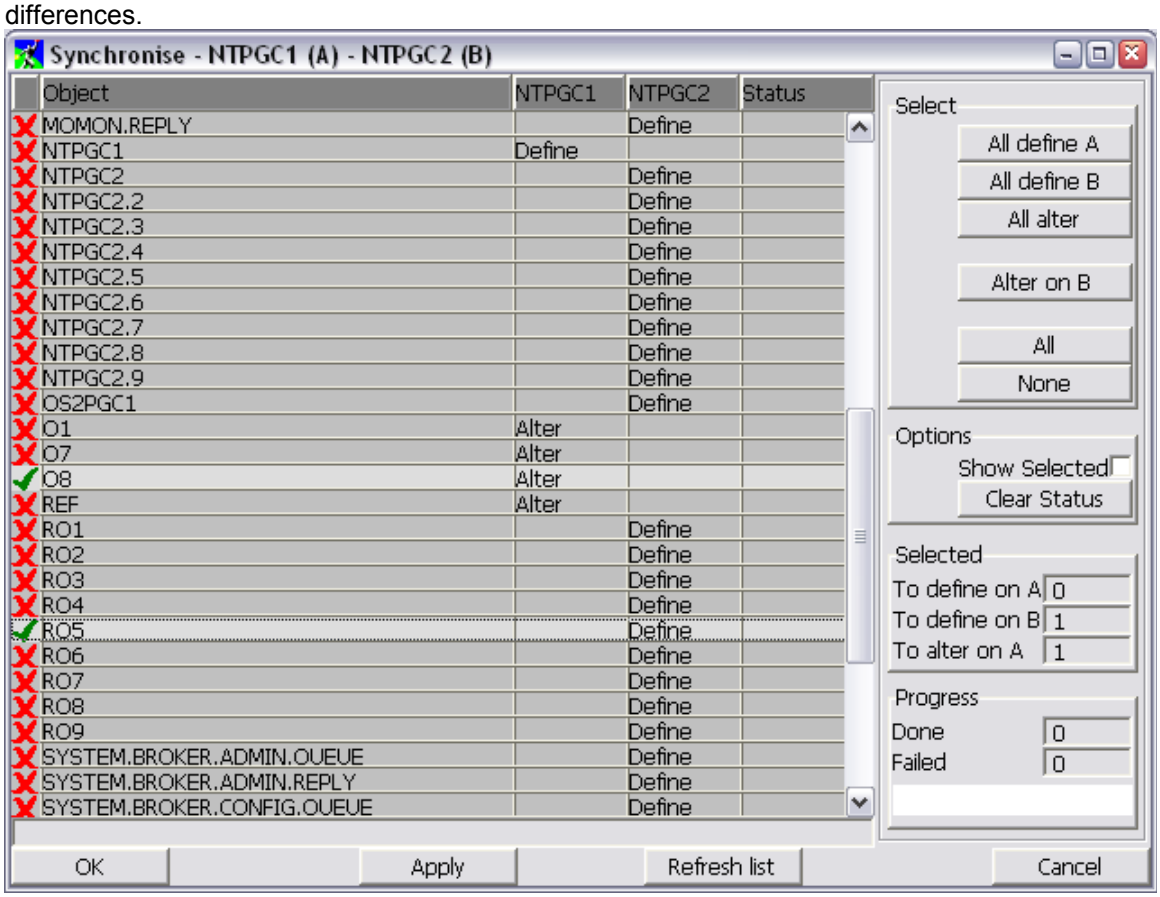
The list output itself is essentially two lists presented side by side. The lists are ordered so that the data for the same object name in the two Queue Managers are aligned horizontally. As with the other dialog lists you can change the fields displayed using the *'Alter list…'* menu option and the lists can be split into two display areas using the *'Split list'* menu option. However, there are a few new menu options which are particular to the compare dialogs.

- **Highlight difference fields**
  This is only applicable to those objects which are defined on both Queue Managers but are different. The fields which match between the two definitions are displayed using the list lowlight colour making the different fields more obvious.
- **Display only different fields**
  This option is useful to reduce the amount of information shown. When selected only those columns which contain different data between the two objects are displayed. Note that the columns displayed will probably vary greatly depending on how many objects are displayed. To further reduce the columns it may well be worth using the object name/type dialog field, entering a filter or using the *'Display objects'* menu item.
- **Display objects**
  This menu option provides a quick and simple way of displaying only a subset of the data based on which of the four categories the object falls in. Just selecting a menu item will deselect all the other items. So for example, clicking on *'Qmgr A defined only'* will only show objects which are defined on Queue Manager A. However, suppose you want to see these plus the objects defined only on Queue Manager B; by holding down the *<Ctrl>* button when selecting the menu option *'Qmgr B defined only'*, this option will be added to the current selection.

In other words, the <*Ctrl*> button allows the selection status of the menu item to be toggled without affecting the selection status of the other object menu items in a similar way to selection in list boxes.

Synchronise

One of the most powerful features of the compare dialog is that it allows you to synchronise the definitions of the two Queue Managers. By selecting the *'Synchronise'* menu option a dialog is presented listing the object differences.



At first glance this appears like a complicated dialog but it is relatively straight-forward. The selected objects are the only ones which will be changed. In the example only two queues will be updated, **Q8** and **RQ5**. **Q8** exists on both Queue Managers so the dialog shows the action needed is to alter the definition on **NTPGC1** to match the definition on **NTPGC2**. Queue **RQ5** is only defined on **NTPGC1** so the dialog shows the action needed is to define it on **NTPGC2**. For objects defined on both Queue Managers clearly one could alter the definition of either to match the other. The synchronise dialog will always assume that you wish to alter Queue Manager A to match Queue Manager B. However, by pressing the *'Alter on B'* button the dialog will then alter the definitions on Queue Manager B to match the definitions on Queue Manager A. It is not possible to alter some objects on A and some on B in a single operation. If you wish to do this then you would have to select the first set of objects, make your changes, then select you different set of objects, change where the alter happens and then make the second set of changes.

The *'Selected'* group of fields shows the number of changes which will be made when the *'Apply'* or *'OK'* buttons are pressed. By selecting the *'Show Selected'* checkbox the list will offer further confirmation by only showing those objects which will be changed. Once the user is happy with the selection they should press either the *'Apply'* or *'OK'* button.

Updates are made and a progress indicator shows how far through the process the dialog is. In addition, the status column next to each object will display the result of the operation. Note that it is possible for an update to fail for a variety of reasons. For example, a queue may be defined when the initial object list is obtained but by the time the synchronisation command is issued someone else has independently deleted the object. Clearly the alter command will fail if there's no base object to alter.

Chapter 13. Web Access

MO71 can be configured to listen for connections from browsers and is capable of returning the information about the configured queue managers as web pages. Currently this information is only read-only, no support is provided for issuing commands other than display. By default MO71 does not listen for HTTP connections. In order to enable the feature start the listener from the HTTP tab in the preference dialog (please see HTTP on page 87 for more information).

The most important thing to ensure is that the 'Root Path' is set to the directory where the HTML files which came with MO71 are located. MO71 is capable of delivering virtually any files and can therefore also be used as a very simple http server.

Getting Started

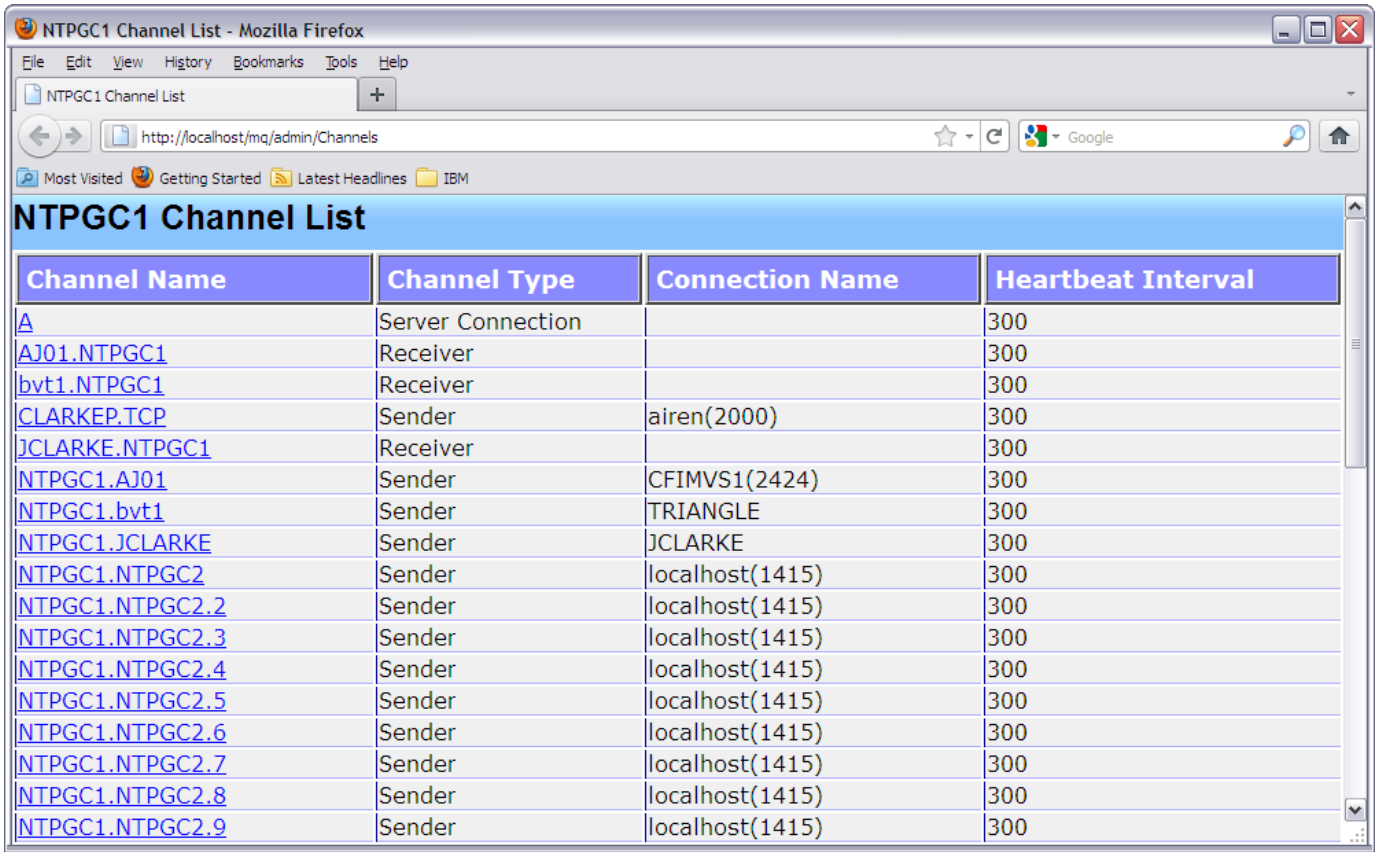
Once the listener is started choose a Queue Manager location that you wish to access from a browser. In the options tab of the location dialog select both **'HTTP Allowed'** and **'HTTP Default'**. This means that any HTTP request which doesn't explicitly state the queue manager name will get data from this location.

The general form of the URL is :-

http://<Host Address>/**mq/admin/**<Object Type> '/' [<Queue Manager Name>] ['/' <Object Name>] ['?' <Dialog Parameters>]

As mentioned before the Queue Manager part of the URL is optional but if not specified there must be a location which is identified as the HTTP default location.

For example, a simple query for the defined channels might look like this.....



NTPGC1 Channel List

Channel Name	Channel Type	Connection Name	Heartbeat Interval
A	Server Connection		300
AJ01.NTPGC1	Receiver		300
bvt1.NTPGC1	Receiver		300
CLARKEP.TCP	Sender	airen(2000)	300
JCLARKE.NTPGC1	Receiver		300
NTPGC1.AJ01	Sender	CFIMVS1(2424)	300
NTPGC1.bvt1	Sender	TRIANGLE	300
NTPGC1.JCLARKE	Sender	JCLARKE	300
NTPGC1.NTPGC2	Sender	localhost(1415)	300
NTPGC1.NTPGC2.2	Sender	localhost(1415)	300
NTPGC1.NTPGC2.3	Sender	localhost(1415)	300
NTPGC1.NTPGC2.4	Sender	localhost(1415)	300
NTPGC1.NTPGC2.5	Sender	localhost(1415)	300
NTPGC1.NTPGC2.6	Sender	localhost(1415)	300
NTPGC1.NTPGC2.7	Sender	localhost(1415)	300
NTPGC1.NTPGC2.8	Sender	localhost(1415)	300
NTPGC1.NTPGC2.9	Sender	localhost(1415)	300

The object type indicates the type of information which should be returned in the page, the table below shows all the available types and gives an example of the full URL syntax.

Examples URLs

Examples of the URLs are given below for each type of display. Most of the links should work against your Queue Manager but some won't because they reference an object name that you won't have in your definitions. Note also that the examples assume that MO71 is configured to listen on the default HTTP port of 80. If a different port is selected then all URLs will need to be modified accordingly.

Display Type	Example URL
Auth Info	http://localhost/mq/admin/Authinfo//SYSTEM.DEFAULT.AUTHINFO.CRLLDAP
Auth Info List	http://localhost/mq/admin/Authinfos
CF Struct	http://localhost/mq/admin/CFStruct//APPLICATION1
CF Struct List	http://localhost/mq/admin/CFStructs
Channel	http://localhost/mq/admin/Channel//SYSTEM.DEF.SVRCONN
Channel List	http://localhost/mq/admin/Channels
Channel Status	http://localhost/mq/admin/Chstatus//NTPGC1.NTPGC2
Channel Status List	http://localhost/mq/admin/Chstatuses
Cluster Queue Manager	http://localhost/mq/admin/Clusqmgr//NTPGC2
Cluster Queue Managers	http://localhost/mq/admin/Clusqmgrs
Connection	http://localhost/mq/admin/Conn//414D51434E5450474331202020202020205E28FF4D20015301
Connections	http://localhost/mq/admin/Conns
Console	http://localhost/mq/admin/Console//0
Console List	http://localhost/mq/admin/Consoles
Message	http://localhost/mq/admin/Msg//SYSTEM.ADMIN.QMGR.EVENT?pos=0
Message List	http://localhost/mq/admin/Msgs//SYSTEM.ADMIN.QMGR.EVENT
Namelist	http://localhost/mq/admin/Namelist//SYSTEM.DEFAULT.NAMELIST
Namelist List	http://localhost/mq/admin/Namelists
Process	http://localhost/mq/admin/Process//SYSTEM.DEFAULT.PROCESS
Process List	http://localhost/mq/admin/Processes
Queue	http://localhost/mq/admin/Queue//SYSTEM.DEFAULT.LOCAL.QUEUE
Queue List	http://localhost/mq/admin/Queues
Queue Manager	http://localhost/mq/admin/Qmgr
Queue Status	http://localhost/mq/admin/Qstatus//SYSTEM.ADMIN.COMMAND.QUEUE
Queue Usage List	http://localhost/mq/admin/Qusage
Storage Class	http://localhost/mq/admin/StgClass//DEFAULT
Storage Class List	http://localhost/mq/admin/StgClasses
Subscription	http://localhost/mq/admin/Sub//bob
Subscription List	http://localhost/mq/admin/Subs
Topic	http://localhost/mq/admin/Topic//SYSTEM.BASE.TOPIC
Topics	http://localhost/mq/admin/Topics

Dialog Fields

It is possible to qualify the URL by specifying the values of the dialog fields.

- http://localhost/mq/admin/Queues/?qtype=model
  Will show the list of model queues on the default HTTP queue manager.
- http://localhost/mq/admin/Channels/?type=svrconn
  Will show all the SVRCONN channels.

In addition there are other parameter values you can specify which modify the data retrieved or how it is displayed. These are equivalent to setting the same option in the message dialog. The list of parameter fields you can add are:

filter	to set the value of the filter field on list dalogs
pos	to set the required position on the queue when browsing
xmlauto	display xml formatted if recognised
xmlshort	display xml in short form
hex	display message data in hex
formatted	display message in formatted form
mqmd	display the message descriptor
low	display low level of detail
med	display medium level of detail
high	display high level of detail
offset	display offsets next to any hex data
msgrange	display a particular range of messages when browsing
maxmsgsize	to set the maximum message size which can be browsed
search	to set the search string for the message browse
message	to return the full message on browse regardless of size

Essentially any dialog field value can be set by sending the name of the field and an appropriate value.

Examples

- http://localhost/mq/admin/Queues/?filter=”cur > 0”
  Will show only queues with messages on them.
- http://localhost/mq/admin/Msg//SYSTEM.DEFAULT.LOCAL.QUEUE?pos=0;mqmd
  Will show the first message on the queue along with the message descriptor.
- http://localhost/mq/admin/Msg//SYSTEM.DEFAULT.LOCAL.QUEUE?pos=0;mqmd;high
  Will show the same information but with ‘high’ detail.

HTML Files

Although the program itself generates HTML constructs to report the data it embeds this data inside HTML files stored on disk. This means that by altering the contents of the HTML files you can achieve a significant amount of customisation about the way the data is reported. For example, colour schemes, layout, logos and additional links can be added. The files supplied with MO71 are:-

- **index.html**
  This is the page that would be delivered for the URL http://localhost
- **mqadminindex.html**
  This is the page that would be delivered for the URL http://localhost/mq/admin
- **object.html**
  This is the html template which is used whenever a single object is returned.
- **objects.html**
  This is the html template which is used whenever a list of objects is returned.

- **msg.html**
  This is the html template which is used whenever a single message is displayed.
- **<object>.html**
  These templates can be used to show a different look and feel for each object type. For example, by have a **queues.html** the display just for the queue list can be changed.
- **errors.html**
  This template will be displayed for general errors, for example queue manager not available.
- **error<error number>.html**
  By having html files with these names you can report back your own error messages for HTTP return codes. For example, a file **error400.html** will be returned if the inbound URL is badly formed.

HTML Inserts

In addition to being able to modify the html files themselves you can also use inserts in the html files. These are:-

- **%[include <filename>]**
  The text will be replaced with the identified include file. This is useful if you want the same html to be included in many different html templates, for example to include a logo or stylesheet. Include files can, themselves, include other files.
- **%[content]**
  The point at which the actual content of the page should be inserted. For example, in an object html file it's where the table containing the object definition should be placed.
- **%[qmlist [<qm name mask>]]**
  This insert will be replaced by a list of queue manager object links. The optional queue manager name can be used to select which queue managers are shown. Wildcards can be used where '*' matches zero or more characters. '?' matches any single character.
  - **%[qmlist]** will show all queue managers selected for HTTP access
  - **%[qmlist %[qm]]** will show just the queue manager being displayed**.** This is useful to show additional links to other objects for this queue manager.
  - **%[qmlist *WIN*]** will show all queue managers with 'WIN' in their name.

  Use of this insert requires that the simple java script which enables expanding areas is included in the template. See the contents of **mqadminindex.html** for examples.
- **%[qm]**
  This insert is replaced by the name of the queue manager.
- **%[loc]**
  This insert is replaced by the location name of the queue manager.
- **%[prev]**
  This insert is replaced by the URL of the previous message. This should only be used in the **msg.html** template.
- **%[next]**
  As above but the next URL.
- **%[objtype]**
  This insert is replaced by the name of the object type. For example 'queue'.
- **%[errormsg]**
  This insert is replaced by the error message. This should be used in the **error.html** template.

Chapter 14. Customization and Reference

Menu options

Many of the menu options act on the selected Queue Manager in the main window. The menu options available are therefore dependent on which Queue Manager is selected. Make sure that the correct Queue Manager is selected before issuing the command.

The menu options available are:-

File

- **Refresh Information**
  Causes the application to refresh its cached information about the selected Queue Manager. This should be used to update the information shown in the tree view of the main window.
- **Refresh Default Objects**
  Causes the application to refresh the definitions of the default objects for the selected Queue Manager. Note that an initial refresh of the default objects can be performed automatically by selecting the appropriate option in the location definition.
- **Open Location**
  Displays a dialog showing the values set for the location allowing the current configuration to be changed. See “Location Dialogs” on page 88 for a description of the fields.
- **Copy Location**
  Displays a dialog to allow a new location to be added to the list. The dialog will be pre-filled with the values of the selected location. See “Location Dialogs” on page 88 for a description of the fields.
- **Add Location**
  Displays a dialog to allow a new location to be added to the list. See “Location Dialogs” on page 88 for a description of the fields.
- **Delete Location**
  Deletes the current selected location.
- **Save Configuration**
  The current configuration will be written to the configuration file, MQMON.CFG.
- **Preferences**
  Displays a dialog to allow the user to change some global parameters on the way the program operates, such as saving dialog positions. See “Preferences Dialog” on page 78 for a description of the dialog.
- **Print**
  Print the list of Queue Manager and location names. See “Printing” on page 48 for more information.
- **About**
  The About dialog gives version and build date information.
- **Help**
  Will try to display this manual. Note that the manual file should be placed in the same directory as the program or its location identified in the *‘Help file location’* preference field.
- **Exit**
  Exits the program[12].
  The current configuration will automatically be written to the configuration file.

[12]The first time the user requests to end the program it will disconnect from all the connected Queue Managers before shutting down. It is possible that this process will hang because, say, the network connection has been lost. Selecting ‘Exit’ again will end the program immediately bypassing the disconnect processing.

Commands

Various menu items to invoke a command dialog against the selected Queue Manager location. The list of commands available will depend on the platform and release level of the selected location.

Action

- **MQSC Window**
  If commands are allowed for the selected location an MQSC window will be presented where MQSC commands can be entered and responses from the command server displayed. This has the advantage that any MQSC command supported by the command server can be entered without requiring MQMONNTP to support it directly. No parsing of the command or response is made other than formatting the data to fit the screen. See "MQSC Window" on page 16 for a description of how to use this window.
- **Predefined Dialog list**
  Shows a dialog containing the currently defined predefined dialogs.
- **Predefined Dialog**
  Shows a dialog allowing the creation of a predefined dialog. See "Predefined Dialogs" on page 94 for a description of predefined dialogs.
- **Event List**
  Shows a dialog of the currently defined Events. See "Event Monitoring" on page 54 for a description of how to use this window.
- **Event**
  Shows a dialog allowing the creation of an event. See "Event Monitoring" on page 54 for a description of how to use this window.
- **Filters**
  Displays the Filter Manager dialog. See "Filter Manager" on page 31 for a description of how to use this window.
- **Compare Definitions**
  Displays a dialog comparing the object definitions of two different Queue Managers. See "Comparing Queue Manager Objects" on page 69 for a description of how to use these dialogs.
- **Enable Alarms**
  When checked the application will alarm whenever an 'alarm' situation is detected. No audible alerts will be given without this option set.
- **Reset Alarm**
  Select this option to stop the alarm sounding. The alarm will sound again if another 'alarm' situation is detected.
- **Reset Location**
  If an attempt is made to contact a location which is not available the icon for that location will show an error. This error condition is normally only reset when connection is made. However, by selecting this menu option the location can be manually reset.
- **Reset All Locations**
  As above but this menu applies to all locations in error.
- **Enable Monitoring**
  When checked the application will send messages periodically to any location which is not disabled for monitoring. See "Queue Manager Monitoring" on page 52 for more details on monitoring.
- **Monitor all**
  Force the sending of monitor messages to all non-disabled locations.
- **Put Message**
  Will display a simple dialog for the selected location to allow a simple message to be put to a target queue. The message can only be a string ('MQSTR' format) but the user can choose how many messages are put and

whether persistence or syncpoint is used. The user can also choose whether the message content comes from the dialog string or a file.

- **Publish Message**
  Very similar to the 'Put Message' above but in this case the message is published to the given Topic.
- **API Exerciser**
  Will display a dialog allowing the user to issue direct verbs to WebSphere MQ. This is useful to test the effects of certain verbs with certain options. See 'API Exerciser' on page 59 for more defaults.
- **Load/Unload Queue**
  Displays a dialog allowing the user to load/unload a queue from/to a file. See "Queue load/unload facility" on page 38 for more details on how to do this.
- **Talk**
  Presents a dialog which allows the user to type a text message and send it to the given Queue Manager. The remote location must be running another instance of MQMONNTP or other such program which will display the message.
- **Disconnect**
  Normally the application will disconnect from a Queue Manager based on the disconnect interval specified in the location dialog. However, if the user wishes to force disconnection earlier than this then this menu may be used.

View

- **View Network**
  Displays the network view dialog. See "Network View" on page 42 for a description of how to use this window.
- **View Console**
  Displays the console window.
- **Queue Manager name**
  Identifies the location by its Queue Manager name.
- **Location**
  Identifies the location by its location description.
- **Set Font**
  There are two fonts which can be set within the program. One used for 'Windows' and one for displaying 'Data'.

  The windows font is used as the font for the main window and the command windows.
  The data font is used as the font for displaying messages on queues and in the MQSC window. It is recommended that the user select a non-proportional font, such as Courier, for the data font since displays of this type are easier to read when the columns of text are aligned on multiple rows.
- **Set Colours**
  Displays a dialog allowing the setting of the colour of various parts of the dialog windows.
- **Set Default Lists**
  Under this menu are menus for each of the list types supported by the program. Selecting a menu will present a dialog allowing the user to modify the attributes displayed when a list is selected for a location. Note that the user will be able to choose from all the attributes appropriate for **all** the locations added to MQMONNTP. However, when a list for a particular location is displayed only those attributes that are appropriate for this particular location will be shown. If an additional location is added which is a later level of Queue Manager those already defined then it may be appropriate to add new fields to the lists. See "Alter list dialog" on page 15 for an explanation of this dialog.
- **List view**
  Displays the list of locations in a list box.

- **Tree view**
  Displays the list of locations in a tree view. If the Preference setting “Display objects in main window” is selected then each location can be expanded to include the major objects for that Queue Manager. For channels and queues the icon changes depending on the state of the object (see “Appendix A:Icons” on page 109 to see what the icons look like).

  Note that the state of the object displayed will be the state the last time the information for that queue manager was refreshed. To refresh information for a location select the location and press *‘Refresh Information’* from the File menu or use the refresh icons in the network display.

- **Non-responders only**
  When this option is active only the locations which are not responding to monitor requests will be displayed. This is useful if you have many, many locations and just want to show the ones with problems.

- **Show MQ Version**
  When this option is active the MQ Version (if known) of the location is added to end of the displayed entry.

Preferences Dialog

The preferences dialog allows the user to set the behaviour of various aspects of the application. The dialog is split into a number of different parts. The first being a set of on/off switches in a scrollable list. The item is on if a tick is displayed next to the item.

General

- **Show confirmation dialogs**
  When checked the application will display a confirmation dialog whenever a potentially harmful command such as “delete queue” is issued.

- **Confirm message operations**
  When checked destructive message operations such as ‘copy’ and ‘delete’ will cause a confirmation dialog to be displayed.

- **Confirm on object change**
  When checked the application will show a confirmation dialog if the user tries to update an object definition after changing the object name from the one whose details are shown. This can prevent accidental changing of the object definition.

- **Auto update lists**
  When checked the application will refresh list displays if actions may have caused the contents to change. For example, creating or deleting a queue would update the queue list display. The application can not always determine which lists may have changed however so the user should periodically refresh the display or set auto-refresh on the dialog if you want to be certain you are looking at the latest data.
  Auto updates can be disabled per location in the location dialog.

- **Auto update browse lists**
  When checked the application will refresh browse queue displays if actions may have caused the contents to change. For example, putting, copying, moving or deleting messages would update the queue browse display. The application can not always determine which lists may have changed however so the user should periodically refresh the display or set auto-refresh on the dialog if you want to be certain you are looking at the latest data.
  Auto updates can be disabled per location in the location dialog.

- **Show objects in main window**
  If checked then the locations in the main window can be expanded to show the main objects of the Queue Manager at that location. Most users use the command menu to get the lists which provide a much richer set of facilities with the returned objects.

- **Predict Hexadecimal characters**
  When browsing a queue the user can request that hexadecimal fields such as the Message Id are displayed in character form as well as in hexadecimal. It is not clear, however, whether these fields will be in ASCII or EBCDIC format. If this item is checked the application will display ASCII or EBCDIC depending on the number of characters falling into the alphanumeric range in each of the character sets. If not checked the value will be assumed to be ASCII.
- **Save object definitions**
  With this option checked the application will save the key fields in the Queue Manager objects in a file called MQMON.DFN in the program path directory. This allows the application to start with a list of the objects defined on each Queue Manager. The user must select refresh information or hit the refresh button in the network display to update this data.
- **Refresh object definitions when first displayed**
  When selected the application will automatically refresh the object definitions the first time the location is expanded in the main window. This requires that the 'Show objects in main window' is also selected.
- **Show frequent menus**
  When checked the command menu will display the menus most frequently used. The number of menu items shown is controlled by the *'Menu items'* setting in the Preference 'Display' tab.
- **Show recent menus**
  When checked the command menu will display the menus most recently used. The number of menu items shown is controlled by the *'Menu items'* setting in the Preference 'Display' tab.
- **Show list menus**
  When checked the command menu will display the menus concerned with showing lists of objects.
- **Show object names in menus**
  When checked the dialog names displayed in the *'Windows'* main menu will contain any appropriate object name.

Dialogs

- **Save Dialog Positions**
  When checked the positions of the command dialogs will be saved
- **Save Dialog Sizes**
  When checked the sizes of command dialogs will be saved
- **Show object name field as a dropdown list**
  When checked the main object name in a dialog is show in a dropdown list. This has the advantage that the list of recently used names can be scrolled through. However, it does have the disadvantage that the value can be inadvertently changed by moving the scrollbar. If required this option can be unchecked which will display the field in a normal entry field.
- **Integral monitor positioning**
  Normally MQMON will try to ensure that new dialogs are initially presented so that they do not span between multiple monitors, even if that was the last saved position of the dialog. By unchecking this option this behaviour is switched off and dialogs will be presented exactly where they were last.
- **Adjust Dialog positions for dual monitors**
  When this option is checked MQMON remembers the position and size of dialogs independently for when the programs is running with one or two monitors. This reduces the chance that a new dialog is placed on the second monitor which isn't there any more.[13]
- **Reflect dialog changes to predefined dialog definition**
  When checked changing the attributes of a dialog created using a predefined dialog definition will be reflected

[13] Note that MQMON does actually know whether a second monitor is physically connected or not. If the Windows settings says there are two monitors but there is only one connected then dialogs will still be placed on this non-displayed monitor.

back to that definition. For example, re-sizing a predefined window will change the size of the values in the definition of the pre-defined dialog. Please see “Predefined Dialogs” on page 94 for information on predefined dialogs.

- **Double click should *Switch* on predefined dialogs**
  When checked double clicking on a predefined list will switch focus to the instance of the predefined dialog rather than bring up the definition of predefined dialog. Whatever the setting of this value holding **‘Alt’** while double clicking will get ‘the other’ behaviour. Please see “Predefined Dialogs” on page 94 for information on predefined dialogs.
- **Autostart predefined dialogs**
  When checked predefined dialogs will be started automatically when MQMON itself is started.
- **Restore running dialogs across restarts**
  Normally, any dialogs running when the application is ended will be restarted automatically when the application is restarted. This option should be unchecked if this behaviour is not required.
- **Display dialogs on taskbar**
  By default the dialog windows are displayed on the windows taskbar. However, if you have many of them then this can use up valuable taskbar space. By deselecting this option the dialogs will not be on the taskbar but can be easily accessed via the ‘Windows’ menu item on the application main window.
- **Show existing dialog by default**
  When the user requests a dialog, for example Queue List, this option controls whether any existing Queue List dialog for this location should be displayed or whether a new one should be created. Holding down the <Shift> key when the dialog menu is selected will give the alternate behaviour than specified in the preferences dialog. For example, suppose we already have a Queue List dialog for location X. With ‘Show existing dialog by default’ selected then pressing the Queue List menu for location X will display the existing Queue List dialog. However, holding down the <Shift> key and then selecting the Queue List menu item will cause a new Queue List dialog to be created.
- **Double click should browse on queue lists**
  By default double clicking on an object on a list window will show its definition. In the case of browsable queues though it may be more convenient to have double click bring up the browse screen. Whatever is chosen as the default action, definition or browse, holding the '**ALT**' key and double clicking will perform the other 'non-default' action.
- **Up/Down keys should tab between fields**
  Versions of MO71 prior to 5.3.3 used the up/down keys to tab between fields in a command dialog. The disadvantage of this is that to select from a pull-down list required actually bring the pull-down list down. By disabling this option the user can scroll through the available choices of a pull-down with the Up/Down buttons. The tab button should be used for tabbing between fields.
- **<esc> should end dialog**
  When selected the escape key will end the current dialog.
- **Ticks should be green**
  When selected, tick symbols  (sometimes known as check marks) are displayed as green. When not selected the symbol will be displayed as red. Remember than until you press ‘OK’ or ‘Apply’ no change will take place.
- **Don’t show red crosses**
  If this option is checked the red cross icons are replaced with a blank square. There is a separate option that covers the icons in the Network dialog.
- **Subkey dialog check**
  Some objects do not have unique names. For example, it is possible to have multiple cluster queues with the same name or multiple channel status objects for the same channel. In these cases the application uses a subkey to determine which object is which. For example, in the case of a cluster queue the cluster queue

manager name is used to uniquely identify the objects returned from the Queue Manager. By selecting this option this subkey checking can be switched off.

- **Show refresh time display**
  By default the dialogs will display the last time they were refreshed on the right hand side of the titlebar. By checking this option this behaviour can be switched off.
- **Show tooltips as balloons**
  When a list dialog contains more columns than can be displayed across the screen the titles of the columns are compressed. Hovering the mouse over the column title will display the full name of the field in a tooltip. This option controls whether the tooltip is displayed as a balloon or a simple rectangle. Note that changing this option will only take effect the next time the application is started.
- **Allow selection of read only fields**
  This options changes whether read only fields in the object dialogs can be selected. Allowing them to be selected means there can be more fields to tab over but it also means that you can select text to copy/paste into other non-read only fields.
- **Use global MQSC command history**
  When selected commands entered into any queue manager are available in the drop down list on any other queue manager. If not selected only the history of commands entered on ‘this’ queue manager are shown. Press ‘alt’ when dropping down the list will show the ‘other’ list.
- **Check for printable characters on input**
  It is possible to enter unprintable characters in to a windows entryfield. As a consequence these characters can be ‘accidentally’ entered into, say, an object description. By checking this option the application will check that all characters on input are printable before adding them to the object. Input is truncated to the last printable character.
- **Disable read ahead**
  Read Ahead can give significant performance advantages particularly over slow networks. However, it can also cause more data to be retrieved from the server than necessary. When checked this option switches off read ahead for all locations, both the reply queue processing and browsing message queues. For more granular control the user can also disable read ahead by location in the option section of the location dialog. For more information please see “Location Dialogs“ on page 88.
- **Default Msg Display Range**
  This option allows the user to set the default browse range when browsing a queue. By default it has the value ‘0,99’ which is the first 100 messages. Any value can be set but bear in mind that large values could cause large numbers of messages to be retrieved from the server with the related impact on performance.
- **Default Max browse message size**
  When browsing messages in a queue the message size is limited to a maximum size which is displayed on the dialog. This prevents accidentally downloading very large messages from the queue and enables quick navigation of the messages. The default maximum message size is 1000 bytes but if you frequently want to browse messages larger than this you might like to increase this default size.
- **History of filter expression**
  How many filter expressions should be kept in the dropdown history
- **History of Object Names**
  How many object names should be kept in the dropdown history

Lists

- **Show QM list initially**
  When checked the Queue Manager list pane is show by default on appropriate list dialogs.

- **Lists should be striped**
  When checked list dialogs are candy striped. The main colour and the stripe colour can be set in the View-Set Colours… menu item. Please see “Colour Dialog” on page 96 for more information.
- **Show rule lines on lists**
  When checked list dialogs will be displayed with a ruled line underneath each item displayed in the list. The colour of this line will be the list selection background colour.
- **3D List Titles**
  When checked the titles in list dialogs appear like three dimensional buttons. Regardless of this setting the titles will always behave like buttons and allow sorting based on the field title selected.
- **3D List Sort Arrow**
  For most colour schemes the arrow displayed in the list titles to identify the sorted field looks better when displayed in a 3D fashion. However, for some colour schemes it may be preferable to switch off the 3D attribute.
- **Reposition list on sort**
  If checked the list is repositioned, after a sort, to try and display the same entry as displayed before the sort. If not checked the top of the list is show after sorting.
- **Select first entry on refresh**
  If checked the first entry of the lists will be selected when the list is refreshed if the list has no current selection.
- **Filled List Sort Arrow**
  For the same reason as the previous option it may be preferable to colour fill the sort arrow with the foreground colour.
- **Size columns by data portion only**
  When displaying a list of objects the column width are automatically set by the greater of the title width and data width. Some fields, generally numerics, tend to be quite narrow despite having fairly large titles. By selecting this option the column width will only be sized based on the data rather than the title. Note that this can make the title unreadable.
- **Raised, Bump, Sunken, Etched, Flat, One dimensional, Half, Softer**
  These options can be used to change the way the list items are displayed. These options control calls to the windows DrawEdge function and the exact display results depending on the implementation of that system. In particular, in my experiments, **Softer**, has no effect on most of the border types.
  The **Half** option causes the border to be drawn only on the bottom and right hand sides of the box. This means that although the appearance is softer it also reduces the 3D look of the border.
- **3D Selection**
  When selected the selected items in the dialog listbox will be drawn in a 3D fashion. The effectiveness of this option will depend on the colour scheme being used.
- **Show multi-fields in list dialog**
  Some object values can be lists such as the names in a namelist object or channel exits in a channel definition. When displaying a list of these objects the application needs to know whether to display all of the values in the list or just the first one. Normally you would want this option checked but if you have channel definitions with a large number of channel exits the list might be excessively large and so it might be worth switching this off.

Connection

- **Auto start command server**
  With this option checked the application will check whether the command server is running when it connects to a local Queue Manager**.** If it isn’t it will be started with the ‘strmqcsv’ command.
- **Uppercase MODEL Queue userid**
  When using a model queue as a location reply queue the application will construct the name of the queue based on the application name and the userid running the program. For some security profiles it may be useful to always have the userid folded to uppercase.

- **Use MQSET to free reply queue**
  When MQMON tries to open a reply queue for a location it might find that queue already in use. This may happen because a previous instance of MQMON was not ended cleanly. By default, MQMON will then inhibit get on the reply queue forcing any outstanding get to return. This then allows the current instance of MQMON to open the reply queue successfully. The ability to issue MQSET against a queue does, however, require a higher level of authority which may be inconvenient. This MQSET processing may be switched off by unchecking this option.
- **Auto Connect Delay**
  This parameter sets the frequency, in seconds, an auto connect location will try to connect to its target Queue Manager. Please see 'Time Interval Format' on Page 97 for the format.
- **Command Expire Interval**
  This parameter sets the expiry interval of the command messages sent by the MQMON program. Expiring messages is useful to prevent a build-up of messages on, for example, a transmission queue which can't connect to a remote location. Please see 'Time Interval Format' on Page 97 for the format.

Network

- **Resolve Queue Paths**
  In the Network Display the application will try to resolve the paths for all queues in the Queue Manager definition. For large numbers of queues and large numbers of Queue Managers this may take some time. If this information is not needed then this value may be unchecked.
- **Resolve SYSTEM Objects**
  Normally the objects starting with the string 'SYSTEM.' are only used as default definitions and are therefore not used as real objects by applications. As such you would normally not want the SYSTEM objects resolved.
- **Check for Problems**
  The Network View display can display possible problems with the current configuration of Queue Manager, for example, an alias queue that targets an undefined queue. If this information is not required then this option may be unchecked to improve performance.
- **Auto-Analyse Network**
  Normally checked, this tells MO71 to automatically reanalyse the relationships between the Queue Manager objects in the Network view. They may be some occasions though where the user always wishes to manually request analysis and therefore this option may be unchecked.
- **Display problem priority icon**
  Each problem is given a priority based on how serious the situation is considered to be. Red is high priority and should really be fixed. Yellow implies it may be problem depending on the circumstances. Green implies that this is not really a problem but may be something you should consider changing. Each problem type can be switched off in the problem selection window.
- **Wildcard Search**
  This option controls whether automatic '*' wildcards are added to the Network view search string. When selected an entered string of 'abc' is actually treated as '*abc*'.
- **Case Sensitive Search**
  This option controls whether the Network view search is treated in a case sensitive or case insensitive manner.
- **Don't show red crosses**
  If this option is checked the red cross icons are replaced with a blank square.
- **Network Snap**
  The network display allows the user to move individual Queue Manager icons. The movement can 'snap' to certain locations :-
  - **None** no snap is performed
  - **Qmgr** the icon is snapped to connected Queue Managers
  - **Grid** the icon is snapped to a fixed size grid

- **Network Snap Size**
  The option allows the user to set the size of snap grid in pixels.
- **Auto Channel Status Update**
  The application may be configured to periodically query the running state of the channels. Which Queue Managers are queried is set by this value
  - **Network (only connected)** Only Network Queue Managers already connected to
  - **Network** All Network Queue Managers
  - **All connected** All Queue Managers currently connected to
  - **All** All Queue Managers
  - **None** Don't perform automatic channel status update
- **Auto Channel Status Update Rate**
  Sets the interval, in seconds, for channel status update. Please see 'Time Interval Format' on Page 97 for the format.
- **Object Refresh Rate**
  This interval controls how often MQMON should try to automatically refresh the objects in the verify window. Please see 'Time Interval Format' on Page 97 for the format.
- **Object overdue interval**
  This interval sets the point at which definitions in the verify window are considered stale. If the set of object definitions is overdue it is identified by a red icon. Overdue Please see 'Time Interval Format' on Page 97 for the format.
- **Object retry rate**
  This sets how often an attempt should be made to automatically refresh the objects if the initial attempt fails. Please see 'Time Interval Format' on Page 97 for the format.
- **Search update delay**
  This option sets the delay between data being entered in to the search field and the search being applied to verify data. The interval is specified in milliseconds. Please see 'Time Interval Format' on Page 97 for the format.

Sounds

- **Warning Sound**
  The sound played when the application wishes to issue a warning. For example an invalid value has been entered or a command failed.
  The value can be beep, none, or .WAV file.
  Depending on the OS it may be necessary to enter the full path to the .WAV file
  The application will play the sound when the user double-clicks on the entry field or presses the 'Play' button.
- **Alarm Sound**
  The sound played for the application alarm.
  The value can be as above for the warning sound.
- **Sound Times**
  Allows the specification of how long and how often the sounds should be generated. The format of the field is **X [: Y [: Z]]** where :-
  - **X Minimum alarm interval**
    Minimum time which must have elapsed before sounding the alarm again
  - **Y Maximum alarm period**
    Maximum time to sound the alarm for
  - **Z Minimum sound interval**
    Minimum time which must have elapsed between making any sound

Display

- **Display menu icons**
  When checked some of the menu items will have simple bitmaps drawn next to them to aid in visual recognition. Changing this option may only have an effect the next time the menu is created. It may be necessary to restart the program to see this option change reflected in all menus.
- **Alter main icon when QM failure**
  When checked the main application icon will go red if any of the Queue Managers are in error.
- **Track selection in container view**
  By default the application high lights the Queue Manager entry as the mouse is moved over the main window. By checking this option this behaviour can be turned off.
- **Use DOS font**
  There are slight differences between the Windows and DOS versions of the same codepage. The codepage itself is identified later in the preferences dialog in the locale field. This preference option just controls the characters displayed when browsing messages on queues. The right value is probably best found by trial and error.
- **Flash main window on alarm**
  Check this option for a visible indication of when the alarm is sounding.
- **Toolbar Raised,Highlight,Highlight Border**
  Selects the style that tools are displayed on the toolbar.
- **Below Edge**
  Determines the type of edge which should be used on the bottom of the toolbar
- **Above Edge**
  Determines the type of edge which should be used on the top of the toolbar
- **Locale**
  Any valid Windows locale string can be entered in this field. The value specified will control the characters displayed in dialogs such queue browsing. It will also control which characters are considered as printable. A value of 'Default' returns the application to the default windows setting.
  The drop down list contains the basic settings for this field. Selecting one and pressing apply will then display the locale selected by windows. The number at the end of the locale is the codepage. This can be changed if required provided the entire string is recognised by windows as a valid locale string. For more information see *"Appendix C:Display National Language Characters"* on page 112.
- **Browse CCSID**
  This field allows you to specify a codepage to convert messages to when browsing messages on queues. In general if you change the value of 'locale' you would want to change this CCSID to match the codepage number. A value of 'Default' will cause the application to use the standard Windows codepage setting. For more information see *"Appendix C:Display National Language Characters"* on page 112.
- **Menu Items**
  The number of menu items which should be displayed at the top level when using the preference options frequent or recent menu items.
- **CSV Separator**
  The character to use for comma separated lists generated by the export command.

Events

- **Events can update queue lists**
  If the user has defined event listeners for queue events this preference option controls whether the events should be used to drive a refresh of Queue List displays. The events that drive the refresh process are :-
  - **Queue Depth High**
  - **Queue Depth Low**

- **Queue Full**

- **Events can update channel status lists**
  If the user has defined event listeners for channel events this preference option controls whether the events should be used to drive a refresh of Channel status List displays. The events that drive the refresh process are :-
  - **Channel Activated**
  - **Channel Not Activated**
  - **Channel Started**
  - **Channel Stopped**
  - **Channel Stopped by User**

- **Reconnect Interval**
  If an event has been defined and started but can not connect to the Queue Manager it will attempt a re-connect. This value allows the user to set the retry frequency in seconds. Please see 'Time Interval Format' on Page 97 for the format.

- **Event Averaging Interval**
  As each event message is caught a smoothed average arrival frequency/ per interval is kept. This value allows the set the interval in seconds. A large interval allows a smoothed average over a large amount of time; a small value will only average over a small period. Please see 'Time Interval Format' on Page 97 for the format.

Console

- **Maximum items**
  This value controls how many items are maintained in the console. If the console reaches this limit the items in the console will be removed in priority order - lowest first. Within each priority items will be discarded oldest first.

Time

- **Large Format, Medium Format, Small Format**
  These values control what is displayed on the right hand side of the dialog titlebar to display the last refresh time for the dialog. Any character string may be entered with the addition of certain inserts preceded by a '%' character. The inserts characters are :-

H	Two digit hour
M	Two digit minutes
S	Two digit seconds
d	Two digit day of month
m	Three character month name eg.Jan,Feb,Mar….
y	Four digit year
D	Three character day of week eg.Mon,Tue,Wed…
t	Simple time format eg. 18:14:03
%	Percent character

- **Time Zone**
  Only available if the user is showing local times and not using the system setting for local timezone. Enter the number of hours (positive or negative) from GMT using a real number. For example, a value of -1.5 would represent 1 hour, 30 minutes before GMT.

- **Show local times**
  When checked the values for GMT times, such as put time in a message browse will be displayed in local time rather than GMT.

- **Use System settings**
  When checked the value for the local time-zone will be taken from the windows system. This value must be unchecked to be able to specify your own time zone.

HTTP

- **Auto start HTTP Server**
  When checked the HTTP listener will be started as soon as the program starts.
- **Below root only**
  When checked the program will limit file requests to those below the HTTP root path
- **Send XML by default**
  MO71 is capable of sending either HTML or XML as the result of a query. If the HTTP requesting application doesn't specify which of these it wants then this setting decides which format the data should be returned in.
- **Show HTTP Dialogs (Debug)**
  When checked this option causes the dialog windows used to serve the HTTP pages to be visible on the machine running MO71. Normally you would expect this value to be unchecked. It can be useful, though, to determine what dialogs and values are set when processing an inbound URL.
- **Root Path**
  The root directory for HTML pages
- **TCP/IP Port**
  The port number to listen on
- **Incl. Filetype**
  The list of file types which may be served. If blank then there are no restrictions.
- **Excl. Filetype**
  The list of file types which are excluded from being served. If blank then there are no restrictions.
- **Clients**
  The current number of clients connected.
- **Peak Clients**
  The peak number of connected clients.
- **Start**
  A button which will start the HTTP listener.
- **Stop**
  A button which will stop MO71 accepting new connections but any existing connections will be maintained.
- **Close**
  A button which will terminate the HTTP listener and any existing connections.

Help

- **Help file location**
  Enter the location of this manual PDF file. The file is displayed when the user presses 'F1'.

Location Dialogs

Similar dialogs are displayed when the user chooses to Add or Open a location. A description of the fields is given below. Note that the Add dialog does not contain all the fields given.

- **Location**
  A free format text description of the location.
  This field is to allow the user to assign a more meaningful name.
- **Queue Manager**
  The Queue Manager name that should be used by MQMONNTP when sending messages to the remote location.
- **Logical Name**
  This checkbox should be selected if the Queue Manager name is merely a logical name and not the actual name of the Queue Manager. This is useful when using client connections; it has the effect that a '*' is pre-pended to the Queue Manager name when connection is made[14].

Connection

- **QM Group**
  A free format group name for this location
  The effect of this value is that the location will be displayed in the tree view 'under' the group name. This allows all your z/OS machines, for example, to be contained in a group such as "z/OS Queue Managers".
  Since objects are placed in the tree view in alphabetic order you can place all of your groups at the top of the main window by having a group name with a leading blank.
- **Network Names**
  You can give a comma or space separated list of Network names with which you want this location to be associated with. This allows groups of Queue Managers to be associated with particular network name. In the network view you can then include or exclude groups of Queue Managers associated with a particular name. See Network Names on page 47 for more information.
- **Via QM**
  If the administrator should not try to connect directly to the Queue Manager name given above but instead should send the messages via a different Queue Manager then this field allows the user to tell the application which Queue Manager it should use.
- **Reply Queue**
  Only required if this location is connected to directly i.e. no Via QM is specified.
  This is the name of the predefined reply queue on the above Queue Manager. It defaults to the name SYSTEM.DEFAULT.MODEL.QUEUE. See "Reply Queue Details" on page 96 to see how another name can be used.
- **Reply Prefix**
  If a model queue is given in the Reply Queue field then the Reply Prefix field can be used to specify the prefix which should be use to construct the actual temporary queue name. See "Model Reply Queue" on page 96 to see how another name can be used.
- **Command Queue**
  The name of the queue at the location which processes WebSphere MQ commands. This value is set automatically to the appropriate name depending on whether the MVS check box is selected or not. It can be changed to another queue if required.
- **Server Queue**
  The name of the queue at the remote machine where MQMONNTP command messages should be sent. It is not required if this location directly connects to the queue manager. But if you want to browse a queue at a

[14] For those interested look at the section on Queue Manager groups in the MQ clients book for a description of this behaviour.

remote location which you are not directly connected to, then you must have a program, for example MQMONNTP, at that location to service those requests. This is the name of the queue which that program is reading.

- **Client**
  If checked this indicates that MQMONNTP should connect to this location via a client connection rather than directly.
- **Configure(d)**
  Only available if 'Client' is checked
  If the location is to be connected to via a client then the client definition to be used can be entered in a dialog presented when this button is pressed. If a separate client channel definition is not entered then the client connection will use the normal MQSERVER or Client Channel mechanisms to connect to the server.
  If the target location is a multi-instance Queue Manager then the connection list field should be a comma separated list of the IP addresses of the locations where the Queue Manager can be located.
  The name of this button will either be 'Configure' or 'Configured' depending on whether there is currently a client configuration specified.
- **Userid**
  Check this item if you wish the program to prompt for a userid/password to be passed over on the connection. This option requires at least MQ Version 6 to be installed on both the client and the server.
  The program will remember the userid across invocations of the program but not the password. Each time the program is started and a connection is attempted the user will be prompted for the password. However, providing the application stays running repeated disconnects and reconnects does not require the user to re-enter the password. By unchecking and rechecking the userid option the user can force the program to 'forget' this cached password. This is useful, for example, if the incorrect userid/password was specified or a password expires.
- **MVS**
  Select this if the remote location is z/OS. This will cause the name of the command queue to change. When checked, messages will be sent to the command queue in MQSC, rather than PCF format.
- **Auto Connect**
  When checked the application will always try to ensure that it is connected to the Queue Manager. If the button is unchecked then connection to the Queue Manager will be made 'when necessary'. In other words the connection attempt will only be made when a monitor message or command is issued against this Queue Manager.
- **Retry Interval**
  If the connection to the remote queue manager fails for some reason (for example, the Queue Manager is ended) then this field allows the specification of a number of seconds before a reconnect attempt is made. No automatic reconnection attempt will be made if the value is 0.
- **Disconnect Interval**
  This field allows the specification of the number of seconds of inactivity before the administrator disconnects from this location. A value of 0 means that the administrator will never disconnect.

Options

- **Disable Commands**
  When checked the administrator will not send commands to that particular location. The administrator will 'beep' if the user attempts to issue a command. Monitoring to that location is still permitted. This option is useful if the remote location does not have a command server and generating commands would result in messages being put to the Dead Letter Queue or worse causing the channel to end.
- **Disable Auto Updates**
  Auto updating lists can be a useful convenience. However, refreshing lists on some locations can be expensive,

for example a client connection over a slow link. In these cases this option can be used to switch it off just for a particular location.

- **Single Thread**
  By default the application will communicate with a connected queue manager using two threads. One thread is used for outbound messages put to the target queues and a second thread sits in an MQGET waiting for the answers to those requests and any messages from other MQMONNTP instances. In some cases the use of a second thread may be too wasteful of system resource, for example over a client connection. In those cases you can check this box and the application will run in single thread mode. This does have some disadvantages though, most notably that messages need to be 'expected'. It would be most inefficient if every few seconds the application issued an MQGET on the off chance of retrieving a message. Consequently it will only issue the MQGET if it has recently issued a command to the Queue Manager. Provided that you are just using MQMONNTP to issue commands to a Queue Manager and display the results you should not notice too much difference running in single thread mode. Responses may take slightly longer to be displayed since the application must now 'poll' for the response. If the command server does not respond within about 10 seconds or so it may be necessary to issue another command, say Refresh, to see the responses.

- **Download Default Objects**
  When selected the definitions of the default objects will be downloaded soon after connection is made with the Queue Manager. The default objects are necessary if the options to display differences from the default objects are required.

  A fresh copy of the default object definitions for any Queue Manager can be requested manually by selecting the *File-Refresh Default Objects'* menu option on the main window.

- **Disable Read Ahead**
  When selected this option will disable read ahead when browsing queues from this location. The option only has an effect when the location is a client connection. Read ahead can give a significant speed improvement when browsing queues, particularly over a slow network. However, it can mean that more messages than necessary are sent to the client from the server. For this reason the user is able to disable read ahead processing.
  An option in the preference dialog allows the user to disable read ahead for all locations, please see "Preferences Dialog" on page 78 for more information.

- **HTTP Allowed**
  This option controls whether MO71 will allow this location to be viewed via a browser through the HTTP Web access. Please see 'Web Access' on Page 71 for a description of this feature.

- **HTTP Default**
  This option controls whether this location is considered the 'default' location for the purposes of Web access. In other words the location which is used if no particular Queue Manager name is provided in the URL. Please see 'Web Access' on Page 71 for a description of this feature.

- **Uppercase Options**
  These options allow the user to say whether, by default, characters should be entered into the object names, exit name or namelist names in upper case. When selected pressing the *'shift'* key will allow entry of lower case characters.

Monitoring

- **Monitor Queue**
  The name of the queue at the remote location which will receive the monitor messages.

- **Disable Monitoring**
  This option, when checked, allows the user to disable monitoring for this location.

- **Enable Alarm**
  When selected an 'alarm' event will be signalled during monitoring if this location does not respond within a given time period.

- **Loop Monitoring**
  When selected it implies that the monitor queue name is actually just a remote queue at the far location which points back to the applications reply queue.
- **Log Console**
  When selected and monitoring is enabled an entry will be added to the console to indicate whether monitoring to this location is successful or not.
- **Monitor Interval**
  This value determines how frequently, in seconds, monitor messages should be sent to this location. The field value can be a number of different fields, the format is as follows :-

  [OK Monitor Interval ['(' OK Expire ')'] [',' Bad Monitor Interval ['(' Bad Expire ')']]]

  The first interval gives the number of seconds between each monitor message provided responses are being received. The second interval gives the number of messages between monitor messages if responses are not being received. This can prevent a large build up of messages if a remote location is not available.

  By default the monitor messages are set to expire a few seconds after a further monitor message is to be sent out. This can be overridden, however, with an explicit value specified in the braces.

  For example the value '**30 (60),3600 (7200)**' has the following meaning :-
  If the target location is available and responding send a monitor message every 30 seconds, each with an expiry of 60 seconds. If the remote location does not respond, send a monitor message every 3600 seconds (1 hour) each with an expiry of 7200 seconds (2 hours).

  If a remote location is not available then the monitor message will probably be in a queue waiting to be delivered to that target queue manager so it is not necessary to continually send new monitor messages since the first ones are likely to be delivered eventually. Bear in mind that monitor messages are non-persistent though, so they will be lost if an intermediate queue manager is ended.

  The default value is 60,1800, note that values can not be lower than a pre-set value.
- **Last Send**
  The time the last monitor message was sent.
- **Last Receive**
  The time the last monitor message was received.
- **Average Response**
  The average response time for monitor messages.
- **Reset Average**
  Resets the average response time calculation.
- **Last Response**
  The number of seconds the last monitor message round trip took.
- **Command OK**
  You can enter here a command string will which be executed whenever MQMONNTP detects that monitoring to the remote location is successful. Note that the command will only be executed when there is a change of state. For example, the first time MQMONNTP receives a response from a remote location it will issue the command but each successful monitor message from there on will not issue the command. However, should monitoring then fail the next successful monitor will cause the command to be issued.
- **Test Command OK / Test Command Fail**
  These buttons cause MQMONNTP to issue the command regardless of the state of the location. This allows testing of the command and/or syntax.

- **Command Fail**
  You can enter here a command string will which be executed whenever MQMONNTP detects that monitoring to the remote location fails. Note that the command will only be executed when there is a change of state as above.

Export

This tab allows the user to specify values so that the object definitions of a Queue Manager are saved on a regular basis. This can be useful to be able to re-create the Queue Manager definitions on another machine or on the same machine in the case of disaster recovery situations.

- **Auto Export**
  A simple checkbox controlling whether MO71 should automatically export objects based on the export frequency or only do it manually when the **'Export Now'** button is pressed.
- **Objects**
  A list of the object types which can be exported. Only those object types which have a 'tick' next to them will be exported.

  For an MQSC export In the special case of the Queue Manager the export is of the form of an ALTER rather than a DEFINE.
- **Export File**
  Enter the name of the file which should be used to store the object definitions. Inserts may be used in the filename, after a % character, which give additional behaviour.

Insert	Effect
q	Queue Manager name
l	Location name
i<*n*>	Cyclic index number For example **file%i3** will write to files file1,file2,file3,file1,file2…. etc
H	Two digit hour
M	Two digit minutes
S	Two digit seconds
d	Two digit day of month
m	Three character month name eg.Jan,Feb,Mar….
y	Four digit year
D	Three character day of week eg.Mon,Tue,Wed…
t	Simple time format eg. 18.14.03
%	Percent character

Figure 19:Export File Inserts

- **Export Type**
  Select one of the four export types:-
  - **MQSC** An MQSC define (or alter) command for the object
  - **Text** A text representation of the object suitable for inclusion in a report
  - **CSV** A comma separated value list suitable for import into anther program such as a spreadsheet
  - **XML** An XML representation which is useful if you wish to have another program which will parse and process the data.
- **Frequency**
  Enter the frequency with which you would like an automatic save of the Queue Manager objects to be performed. The frequency can be entered in one of two ways

- **Explicit time**
  An explicit time takes the form of a day followed by a time. So, for example :-
  - **Monday 09:00**
  - **Everyday 15:00**

  Only the first 2 characters of the day need be specified.

- **Time Interval**
  The units of time are Weeks, Days, Hours and Minutes. So, for example, enter values such as
  - **1 day**
  - **12 hours**
  - **1 week 2 days 4 hours**

- **Default Objects**
  A simple checkbox controlling whether default objects are exported.
- **System Objects**
  A simple checkbox controlling whether system objects (those starting "**SYSTEM.**") are exported.
- **Default Differences**
  When selected only the differences between the current object and the default object definition is exported. This can be useful to export only the key values and allow installation defaults to still apply when the exported script is executed.
- **Last Attempt**
  A simple time display of the last time (since MO71 started) that an attempt was made to export the object definitions.
- **Last Export**
  A simple time display of the last time the objects were successfully exported.
- **Export Now**
  When pressed MO71 will run the export process now rather than waiting for the export frequency to be reached.
- **Status**
  A simple status field reporting what part of the export process has currently been reached.

Pub/Sub

This tab allows you to set up the configuration options to allow you to administer the Publish/Subscribe objects in the broker.

- **Manage Publish/Subscribe**
  This checkbox controls whether Publish/Subscribe commands should be available in the application for this location
- **All Identities**
  Objects such as such as Publishers and Subscribers can be added to the broker anonymously and as such will normally not be returned when a list of subscribers are requested. By selecting the 'All Identities' option even the anonymous Publishers and Subscribers are reported. This option does require greater authority to the Queue Managers, for further information refer to the Publish/Subscribe User Guide.
- **Broker Queue**
  The name of the primary broker queue which should be used when sending control messages.

Buttons

- **Monitor Button**
  This button can be used to force a monitor message to be sent immediately.

Predefined Dialogs

Accessible from the main menu under the '*Action'* menu the user can create predefined dialog definitions. These definitions allow the user to create a set of dialogs of various types which can be started either when the application itself starts or from the predefined dialog list dialog.

The attributes which can be associated with a predefined dialog are:-

Dialog Type
The most important attribute which is the type of dialog you want created.

Description
This field is a free format description of the dialog for reference purposes. This field can be used in conjunction with the **-p** parameter to start a defined subset of dialogs on application start. By including a word in the description, say MONITORMVS, the application can be started with the command, **mqmonntp –p MONITORMVS**, and all the dialogs with MONITORMVS in their description.

Queue Manager
The Queue Manager for which the dialog should be started. A special value of **<Prompt>** may be specified. If **<Prompt>** is used the application will prompt for the name of the Queue Manager every time a predefined dialog of this type is to started.

Status
Shows whether a dialog of this type is currently active or not.

Id
A unique numerical id which identifies this dialog definition. The id of a definition is not guaranteed to be constant across multiple invocations of the program.

Filter
Use this field to set an initial value for the filter in the dialog.

Auto Start
This field defines whether the dialog should be started automatically upon program start.

Initial Refresh
This field defines whether the dialog should refresh its display automatically on start. Specifying initial refresh **'yes'** displays the dialog already containing information but does not allow any selection fields to be specified.

Refresh Rate
If the dialog should be refreshed at regular intervals this field can specify the time interval in seconds.

Restore
This field determines how the dialog should be displayed:-

Normal	Normal, sizable window
Minimize	Minimized to the taskbar or window menu
Maximize	Maximized

X Position, Y Position, Width, Height
The initial position and size of the dialog.
A preference option controls whether re-sizing and moving the predefined dialog should update the values in the dialog definition.

Show Buttons, Show Fields, Show Filter
The settings that should be used to determine which parts of the dialog should be displayed.
A preference option controls whether changing these settings while the dialog is running should be reflected back to the dialog definition.

Auto Export, Export Type, Export Default Differences, Export File, Export Append
These fields allow you to start a predefined dialog which will automatically write to an export file whenever it refreshes its data. For a description of these fields please see "Auto Refresh Export Fields" on page 14.

Menu

- **Create**
  The create menu item will take whatever fields are presented on the dialog and create a new predefined dialog entry. The predefined dialog list dialog will automatically update to show the new entry.
- **Update**
  The update menu item will take whatever fields are presented on the dialog and update the dialog entry with the id given by the id field.
- **Open**
  The open menu item will bring up the predefined dialog to allow the values to inspected or changed. This can be the double-click default action depending on a preference option.
- **Start**
  The start menu item will start a new instance of the selected dialog regardless of whether an instance of this dialog is already running.
- **Switch**
  The switch menu item will check whether there is an instance of the selected dialog already running. If there is the dialog is brought to the foreground, if there isn't a new instance of the selected dialog is created. This can be the double-click default action depending on a preference option.
- **Delete**
  The delete menu item will delete selected dialog definition.

Container Windows

The Network Display shows a number of windows which are 'containers'. A container is a window suitable for showing text, bitmaps and buttons in a tree view. It is similar to the standard windows container control but I had a number of problems using the standard control. Firstly it was very difficult to add buttons to the control. Secondly the standard windows control is rather slow to update and paint, particularly when it has thousands of entries. For this reason I decided to write my own container control from scratch which allows me full control over the interface, features and performance.

The container can be driven either by mouse or keyboard. For keyboard control the item in focus is shown either with a slightly different background colour or with a focus rectangle around the text.

Keyboard Control

The following keys can be used to navigate the container

Key sequence	Action
Arrow Keys	To move the focus from one item to another.
Space	To activate a button, selectable text or a sub tree arrow
Home	Move to top of container data
End	Move to end of container data
PageDown	Move one page down through container data
PageUp	Move one page up through container data
Shift+Home	Move to the far left of container data
Shift+End	Move to the far right of container data
Shift+PageDown	Move one page to the right in the container data
Shift+PageUp	Move one page to the left in the container data
F9	Toggle Expand/Contract state of current line
F9+Shift	Expand current line and all sub trees of current line
F9+Ctrl	Contract current line and all sub trees of current line
F9+Alt[15]+Shift	Expand all lines in container
F9+Alt+Ctrl	Contract all lines in the container

[15]Depending on keyboard mapping it may be necessary to use the 'Alt Gr' button rather than just the 'Alt' button.

Reply Queue Details

The Administrator uses a reply queue to receive all of its reply messages from a Queue Manager it directly connects to. By default the name of this queue is SYSTEM.DEFAULT.MODEL.QUEUE[16] but can be overridden by changing the Reply Queue name in the location definition. The local or model queue of whatever name you choose must already exist on the respective Queue Manager.

If more than one instance of the Administrator is run against the same Queue Manager then each instance must use a different local reply queue or a model queue.

Model Reply Queue

Using a model queue is useful if many instances of MQMONNTP will be run against the same configuration file at the same time. The disadvantage of using a model queue as the reply is that an instance of MQMONNTP has no 'fixed address'. In other words the reply queue name is not predetermined. This in turn means that you can not use loop back monitoring since that requires a predefined remote queue.

When a model queue is used as a reply queue the actual queue name used is governed by the DynamicQName specified in the object descriptor. The Reply Prefix field of the location dialog allows the user to specify the value for DynamicQName field.

The field can be any sequence of characters providing they are valid Queue name characters. Two special characters have special significance:-

* An asterisk at the end of the string indicates that MQ should add a sequence of hex characters to ensure that the Queue name is unique.

%u This sequence indicates that the current userid should be inserts at this part of the name.

If the reply prefix is empty the value of ***MQMON.%u.**** will be used. On my own machine using this default setting, with a userid of CLARKEP, the queue names I get are of the form **MQMON.CLARKEP.412EE8F603370020**

Colour Dialog

The colours of most of the elements of the application can be configured in the colour dialog available under the 'View' menu in the main window..

The dialog displays all the dialog elements in a list and a number of action buttons. The list of elements shows the colour of element when the dialog was started, a text description of the element, the colour of the selected scheme for this element and the current colour of the element that would be applied if the 'OK' or 'Apply' button was pressed.

The element list allows multiple selection using the standard extended selection scheme. The action buttons will either apply to those items selected or to all the elements in the list.

For example, to change to use the Grey colour scheme use the following sequence.

1. Select the Grey scheme in the colour scheme drop down list
2. Press 'Reset All to scheme' button
3. Press 'Apply' button

Colour Schemes

Colour schemes are a fast and simple way to set all the colours in the application to a predefined set of values. However, each user may have different preferences. If you create a colour scheme which is significantly different than the provided set of schemes I encourage you to send me your scheme so it can be included in the next release of the SupportPac. To do this run the program with the -d flag, for example **'mqmonntp -d'**. In the colour dialog you will now see a **'write'** button. By pressing this the program will write a file called **c:\temp\schemes.txt.** Please send me this file, together with the name you'd like to use for the scheme. I appreciate you taking the time to help improve the SupportPac.

[16]Earlier versions of the program used to set a default value of MQMON but this required that MQMON was predefined on the Queue Manager before the administrator would run against the Queue Manager.

Extended Selection

By holding down the **'Alt'** button at the same time a selection is made from the colour list the application will select (or deselect) not only the particular item in question but all the items in the colour list which have the same current colour. This allows the user to easily change all the items that use the same colour to now use a different colour. On some systems the two **'Alt'** buttons behave slightly differently. One is additive to the current selections and the other will replace any previous selection.

Time Interval Format

A number of configuration options are some form of time interval. These parameters use a common input format. The format allows for a list of value and unit pairs. The units are **milliseconds(ms), seconds, minute, hour, day, week.** Generally speaking only the first letter of the unit need be used. The following inputs are examples of valid input.

- 1 day 3 hours 25 minutes
- 1d3h25m
- 35 ms
- 25

In cases where no unit is specified the default unit for the particular field is used. Note that not all units are valid for all field types. Generally speaking fields which have a default unit of milliseconds do not allow units larger than seconds. The user should always press the 'Apply' button and verify that the output field value matches the value which was required. Note that the output format will not match exactly the input but will be in the standard value. For example, entering **48h** will be output as **2 days**.

Parameters

The Administrator program does not require any parameters but if you wish you have the option to provide a single location via command line flags. By default this location will be available for the duration of the program but will not be saved to the configuration file. You can use the **-k** flag to ask the program to store the definition in the configuration file if you wish.

The complete list of flags are :-

- **-m** < *Queue Manager Name* >
  This parameter is mandatory if you wish to create a location from the command line. All other parameters are optional and allow you to change the definition that is created.
- **-q** < *Queue Name* >
  Specify the name of the 'monitor' and 'reply' queue that should be used. If not specified the queue name **SYSTEM.DEFAULT.MODEL.QUEUE** will be assumed.
- **-l**
  Tells the program to create a client connection to this Queue Manager.
- **-a**
  Tells the program to auto-connect. In other words, the program will attempt to always have a connection to this location.
- **-s**
  Identifies the location as an MVS machine.
- **-t**
  Tells MQMONNTP to connect to the Queue Manager with only a single thread
- **-v** < *Via Queue Manager* >
  Tells the program to route all messages to this location via this Queue Manager definition. The 'Via Queue Manager' must be already defined in the programs CFG file.
- **-c** < *Channel Name* >
  For client connections you can specify explicitly the channel name to use on the connection. If this option is

used then program will assume it's a TCP/IP connection and you must specify the TCP/IP Host name or address of the machine below.

- **-h** < *TCP/IP Host Name* >
  Host name of the target machine for client connections. The Channel name parameters must be specified for this parameter to take effect.
- **-g** < *Group Name* >
  Tells the program which logical group the definition belongs in
- **-k**
  By default the location created from command line options will not be saved when the program ends. However, by using this flag you can ask the program to **keep** the definition.
- **-f** <*Configuration File Name*>
  By default the program will use file MQMON.CFG in the directory where it finds the program. However, by using this parameter you can override this and give the name and location of the configuration file to use.
- **-n**
  Don't autostart any predefined dialogs.
- **-p** <*Filter*>
  Start any predefined dialogs which have the filter string in their description fields.
- **-r**
  Don't write to CFG file, open for read-only.

Chapter 15. Authorities

Defining authorisations for MQMONNTP users gives some level of control over what the application can do. For example, suppose you want some users to be able to display queues but you don't want them to go near your channels or vice versa. You can remove menu items, such as **talk** or **mqsc**, from the main window

Authorities are provided using a separate configuration file - MQMON.AUT. This file must be created manually (using your favourite editor) and placed in the same directory as the MQMON.CFG file. If this file is not present then all options will be made available to all users. The MQMON.AUT file can be write protected since it is constructed and updated only by the administrator setting up the authorities. Providing the application can read the file it can be protected as required. It should be noted that this mechanism is not intended to be fully secure; the serious user can always find ways to circumvent such mechanisms if required. For the majority of these kinds of situations, however, a simple way of preventing the full range of options being presented to the user is sufficient.

Authorisations can either be specified additive, for example **queue_display**, or subtractive, for example **! queue_display**. The advantage of the subtractive it that it future proofs for future securities. For example, suppose you want a user to be able to do anything with a queue except delete them then you could use the authority **queue_all !queue_delete**.

The best way of explaining the format of the file is to show an example:-

```
# Set authorisations for users of MQMON
#
# Global Authorisations
queue_display location_display

# Per Queue Manager Authorisations
QM: NTPGC1
        all
QM: NTPGC2
        location_change
QM: NTPGC3
        queue_change
```

The format of the file is entirely free-format. White space can be added/removed wherever the user chooses. Comments can be added from the first ';' or '#' character to the end of the line.

The file is structured into global authorisations and authorisations on a per queue manager basis. The global authorisations appear before the first 'QM:' keyword. The 'QM:' must be followed by the name of the Queue Manager and then zero or more authorisation keywords. Authorisations specified in the Queue Manager section are additive to the global authorisations. They do not replace the authorisations. So a user running with this file can:-

- Do all operations against NTPGC1
- Display queues and display and change the location for NTPGC2
- Display and change queues and display the location for NTPGC3
- Display the queue and location against all other queue managers

Note that the only queue manager that this user could display channels, processes and the queue manager object itself for is NTPGC1.

A full list of keywords for authorisations is given below.

location_create	Allows the user to create new locations
location_change	Allows the user to update this location
location_display	Allows the user to display this location
location_delete	Allows the user to delete this location
queue_create	Allows the user to create new queues
queue_change	Allows the user to change defined queues

queue_delete	Allows the user to delete queues
queue_stats	Allows the user to view queue statistics
queue_usage	Allows the user to view queue usage
queue_display	Allows the user to display queues and queue lists
queue_admin	Allows the user to issue clear queue
msg_create	Allows the user to use the put message dialog
msg_copy	Allows the user to copy messages from one queue to another
msg_move	Allows the user to move messages from one queue to another
msg_display	Allows the user to browse queues
msg_delete	Allows the user to delete messages from queues
namelist_create	Allows the user to create new namelists
namelist_change	Allows the user to change defined namelists
namelist_display	Allows the user to display namelists and namelist lists
namelist_delete	Allows the user to delete namelists
process_create	Allows the user to create new process objects
process_change	Allows the user to change process objects
process_display	Allows the user to display process objects and process lists
process_delete	Allows the user to delete process objects
channel_create	Allows the user to create new channels
channel_change	Allows the user to change existing channels
channel_display	Allows the user to display existing channels and channel lists
channel_delete	Allows the user to delete channels
channel_admin	Allows the user to issue administration commands such as reset and resolve channel.
channel_start	Allows the user to start a channel
channel_stop	Allows the user to stop a channel
channel_ping	Allows the user to ping a channel
clusqm_display	Allows the user to display cluster queue manager objects
authinfo_create	Allows the user to create new authinfo objects
authinfo_change	Allows the user to change authinfo objects
authinfo_display	Allows the user to display authinfo objects
authinfo_delete	Allows the user to delete authinfo objects
qmgr_mqsc	Allows the user to get an MQSC window. ***Care should be taken with this option since it essentially gives the user access to the full range of MQSC commands. No parsing of the command is done by the MQMONNTP program.***
qmgr_change	Allows the user to change the Queue Manager definition
qmgr_display	Allows the user to display the Queue Manager definition
qmgr_admin	Allows the user to issue administration commands against the Queue Manager such as the cluster commands RESET, REFRESH, SUSPEND and RESUME cluster.
network_display	Allows the user to display the network view
location_all	Allows the user to do anything to the location objects
msg_all	Allows the user to do anything to messages
queue_all	Allows the user to do anything to queue objects
channel_all	Allows the user to do anything to channel objects
authinfo_all	Allows the user to do anything to authinfo objects
qmgr_all	Allows the user to do anything to the Queue Manager definition
all_create	Allows the user to create objects of all types
all_change	Allows the user to change objects of all types
all_display	Allows the user to display objects of all types
all_delete	Allows the user to delete objects of all types

all	Allows the user to do anything to all object types.
nomenu_refresh_information	Removes the named option from the menu
nomenu_refresh_default_objects	“ “ “
nomenu_open_location	“ “ “
nomenu_copy_location	“ “ “
nomenu_add_location	“ “ “
nomenu_delete_location	“ “ “
nomenu_save_configuration	“ “ “
nomenu_preferences	“ “ “
nomenu_print	“ “ “
nomenu_mqsc	“ “ “
nomenu_predefined_dialog	“ “ “
nomenu_predefined_event	“ “ “
nomenu_filters	“ “ “
nomenu_compare	“ “ “
nomenu_monitoring	“ “ “
nomenu_put_message	“ “ “
nomenu_publish_message	“ “ “
nomenu_qload	“ “ “
nomenu_talk	“ “ “
nomenu_disconnect	“ “ “
nomenu_view_network	“ “ “
nomenu_view_console	“ “ “
nomenu_default_lists	“ “ “
nomenu_list_view	“ “ “
nomenu_font	“ “ “
nomenu_colours	“ “ “
nomenu_view	“ “ “
nomenu_alterlist	“ “ “
nomenu_splitlist	“ “ “
nomenu_defaultfilter	“ “ “
nomenu_initialrefresh	“ “ “
nomenu_predefined	“ “ “
nomenu_autorefresh	“ “ “
nomenu_defaultobjs	“ “ “
nomenu_listtitles	“ “ “
nomenu_export	“ “ “
nomenu_api_exerciser	“ “ “
nomenu_context	Removes any context menus (those displayed when button 2 is pressed) from the application.

Chapter 16. MQMONA Agent

From a performance point of view the way in which a list of objects is returned from the command server is not optimal. Essentially the command server returns each object as a separate message. This can have a significant performance impact particularly if the connection to the command server is over a client connection. The inefficiency is most noticeable across a slow network, for example a dial-up connection, but even in a fast LAN environment you may find MQMONA speeds things up.

The essential principal of MQMONA is to work like a 'proxy' command server. It runs local to the Queue Manager and reads a 'command' queue. By default this is **MQMONA** which must be created before the MQMONA program is run. Commands directed to the MQMONA queue are redirected to the command server queue and responses from the command server are then received by the MQMONA program. The MQMONA program then collates all the responses from the command server and sends only the collated response to the originating application. The net result of all this is that, although the Queue Manager is handling more MQI requests, there are far fewer messages being flowed across the client link[17] from the originating application.

The only applications which are currently able to handle the responses from MQMONA are my two Administration SupportPacs MO71 (version 5.3.4 and later) and MO72 (version 1.2 and later).

Configuration

MQMONA

Configuration of the MQMONA program is only via command line parameters. A full list of these can be seen by entering MQMONA -? (or any incorrect usage will also display the full list).

- **-c <Command Server Queue>**
  The name of the command server queue that MQMONA should forward messages to. By default either SYSTEM.ADMIN.COMMAND.QUEUE or SYSTEM.COMMAND.INPUT depending on the platform.
- **-m <Queue Manager Name>**
  The name of the Queue Manager MQMONA should connect to
- **-n**
  MQMONA will try to compress the values in the command responses. For example, removing all the unnecessary blanks. This compression can be switched off using this flag.
- **-q <Queue Name>**
  The name of the command queue that MQMONA should read. By default MQMONA
- **-p {parm=<value> [,] }** Parameter String

Field Name	Description	Default
maxage	Maximum age of request	10
maxsize	Maximum size of a reply message	100000
cntdelay	Minumum interval before displaying message counts. Can be used to reduce amount of diagnostic information written	0
reqwait	How long we should wait for a request before checking for replies	300
reqcheck	How long we should wait for new requests when we're waiting for a reply	0
repwait	How long we should wait for a reply when we're expecting one	1
repcheck	How long we should wait for a reply when we aren't expecting one	0

[17] There is nothing to stop you running MQMONA when the application is locally connected but at the moment there is no benefit. This configuration can be useful to help set up the system and ensure MQMONA is working correctly.

For example : mqmona –pmaxage=8,maxsize=50000

- **-v { value }** Verbosity level
  - **l** Low Detail
  - **m** Medium Detail
  - **h** High Detail
  - **c** Message Counts

    Useful for seeing what messages MQMONA is processing. Statistics for three message types are printed.

    Cmd Command messages sent from the application into MQMONA
    Rep Reply message received from the command server
    Res Response message sent by MQMONA back to application

    In each case the number of messages and the number of bytes are printed. This allows you to see the reduction in both messages and bytes, the difference between the Rep and Res values.
  - **q** Quiet

For example : mqmona -vqc

MO71

Configuring MO71 to use MQMONA could not be simpler. In the each location definition there is a 'command queue' value. Just change this to be the queue that MQMONA is reading, this is by default MQMONA.

MO72

Configuring MO72 to use MQMONA is as simple as changing the command queue. This is done using the **–q** command line parameter. For example:-

mqsc –q MQMONA

Security

MQMONA is just a normal MQI application and, as such, does not pose a security risk. Messages sent to it are passed on to the command server using the 'pass all context' MQI option. Pass all context requires that MQMONA is run under a userid that has this authority for the command server.

Source Code

The MQMONA program is provided in source code format for a number of reasons. The main one is so that I don't have to provide multiple versions of the program executable, one for each different server. It also has the advantage that the users can see exactly what the program is doing and even extend it (or fix it) if required. The program is just a normal MQI application and can therefore be built using the instructions for building the sample programs on your particular platform.

Clearly releasing source code is hazardous from an authors point of view since there will always be people who will not like the way it's written, the style, the comments etc. I welcome comments and suggestions but please bear in mind that I don't really have the time to beautify the code completely.

Futures

Clearly having an agent process running on the server introduces the possibility of adding many more features such as remote monitoring, browsing messages etc without the need for a client connection.

While it would be nice to provide these features I have no current plans to do so. It's more a question of time than anything else. If anyone extends the program to include these features I would be interested in seeing them.

Chapter 17. Trouble Shooting

Because of the nature of messaging the application can not guarantee that replies from remote Queue Managers will be returned. If you find that a command window continually waits for a response the most common reasons are:-

1. The Command server for the Queue Manager is not running
2. The channel to the remote Queue Manager is not running
3. The channel from the remote Queue Manager is not running
4. The user does not have security access to the Queue Manager

   The usual symptom of this is that messages will build up on the remote Queue Manager's Dead Letter Queue. The 'Q' program (SupportPac MA01) can be used to browse the messages on the Dead Letter Queue to see the reason code.

   The most common cause of this is that the userid in the incoming message does not have a security profile on the receiving machine. For Windows it will be the logged on userid. Therefore, the receiving Queue Manager, be it OS/400, z/OS, AIX or whatever, must have a security profile for this id. Alternatively, Channel exits can be used to change the userid as the message is being transferred.

Chapter 18. Migration from previous versions

I always try to ensure that, as each version is shipped, all the features that were working on the previous versions remain intact. However, since this is a spare time activity I can't test all functions individually. When installing a new version I always recommend you backup the previous MQMONNTP program and the configuration file MQMON.CFG. If you do find a problem then please send me a description and I'll try to fix it and send you a replacement.

Migrating from a version prior to Version 7.0.2

Regressions

The introduction of the Queue Manager list in many of the list dialogs has had a profound effect on the code. The data model has had to change significantly which is one of the main reasons the time between releases has been so long. While every effort has been made to try and ensure that previous function is working correctly it would be surprising if some problems didn't leak out. I would recommend that users keep their previous version of MO71 handy so that they can revert to it if a bug is found. Needless to say, please report any incompatibilities to me if they are found.

MO71 Authorities

By request there was a slight change made to MO71 authorities so that cluster queue managers had their own authorities, previously they were controlled by the channel authorities. If you use MO71 authority files it may be necessary to make a change to the file to ensure that cluster queue manager dialogs are displayed correctly.

API Exerciser

In the previous version the API Exerciser could be displayed by pressing PF9 on any screen in the application. In this version PF9 will only show the API Exerciser when the main window has focus.

Frequently Used Menu Commands

The algorithm used to calculate the frequently used menu commands has changed which means that the application must re-learn the commands you use regularly. If you use this option you will notice that the first few times you use the application the menus will take a little while to re-sort themselves.

Migrating from a version prior to Version 7.0.1

Strip Headers

The default value for strip headers has been changed to off. This is because it was considered safer not to strip headers by default.

API Exerciser

The API Exerciser feature is a fairly powerful feature which is really designed for experimentation on non-production Queue Managers. It may be worth disabling this feature (see Authorities) when MO71 is used in a production environment or by non-programmer users.

Object Histories

This version of MO71 keeps a separate history for transmission queues and initiation queues than for 'other' queues. This change in behaviour may be noticeable but hopefully users will regard the new behaviour as preferable.

Migrating from a version prior to Version 7.0.0

Exporting in XML

The format of the XML format has been altered slightly to include an encompassing tag.

Migrating from a version prior to Version 6.0.3

Filter Names

Because filters can now invoke each other it was necessary to introduce the restriction that filter names can not contain blank characters. If you have embedded blanks in your filter names then they will be replaced with the underscore (_) character the first time MQMONNTP is run on version 6.0.3 or higher.

Migrating from a version prior to Version 6.0.0

New dialog colours

There have been a variety of new separately colourable dialog parts added in this release.
These are :-

- Default Object Background
- Default Object Foreground
- Default Object Title Background
- Default Object Title Foreground
- Protected Field Title Background
- Protected Field Title Foreground
- List Lowlight Foreground

The program will make an educated 'guess' at what good values for these colours might be for your current colour scheme but it is unlikely to get it right in all situations. If you find that certain effects, like highlighting default objects, are difficult to see then it may be necessary to alter the value of these colours manually using the *'View-Set Colours…'* menu item from the main window menu.

Migrating from a version prior to Version 5.3.4

Filters and the % operator

Prior to Version 5.3.4 the % character was used as the modulus operator in filters. Version 5.3.4 introduces the concept of user variables and also user variable inserts. To keep consistent with other substitution characters in the program the % character is now used to indicate a substitution string. The modulus operator is now **mod**. For example, to calculate the modulus of 4 and 3 specify **4 mod 3** which will yield the result 1.

Migrating from a version prior to Version 5.3.2

A number of changes may have a potential affect on the operation of the program. Wherever possible the application will try to make sensible choices but it may be necessary to modify a configuration value.

List Selection Colour

Previous versions displayed selected item in the list dialogs using the reverse colours of the list entries themselves. The new version allows this colour to be set independently. It may be necessary to alter the colour setting for list selections to see the selections clearly.

Client Configuration

Previous versions always used a standard version 4 MQCD structure to make MQCONNX connections to the Queue Manager. In addition, chains of send/receive channel exits and SSL were not supported. The application will now dynamically choose the structure version. If you have trouble connecting to a location over a client connection

which worked in the previous version it would be worth ensuring that your client connection definition has defined only those fields supported by your installed WebSphere MQ client.

Autorefresh resolution changed to 1 second

It may be necessary to change your AutoRefresh settings if you have any active.

Change default reply queue

To make life easier for the first time user the default reply queue is now SYSTEM.DEFAULT.MODEL.QUEUE. This may mean that anyone adding locations by starting MQMONNTP via the parameter settings may need to pass the '-q' parameter to set the value back to 'MQMON'.

Migrating from a version prior to Version 5.2

Options

In previous versions there used to be an Options menu which allowed you to set options such as whether the dialog positions should be saved across restarts. These configuration options are now available in the 'Preferences' dialog.

Objects in the main window

Previous versions used to allow the expansion of objects in the main window by default so that the list of queues and channels, for example, could be seen. By default this is turned off in the 5.2 version but can be selected in the preferences dialog.

List and Tree View

The default view has been changed from 'list' to 'tree'. Most users find the tree view much more flexible and appealing.

Migration from a version prior to Version 5.0

Prior to Version 5.0 the application connected only to one Queue Manager and all messaging to other Queue Managers were made via this single Queue Manager. This proved restrictive and awkward since channels must be up and running between all the Queue Managers. Version 5.0 has under gone a major internal revision. Essentially what was a two threaded application has become a multithreaded application. Two connections are made to each directly connected application from two different threads. One thread is used for putting and the other for getting.

Another significant change is the ability to configure a client channel configuration as part of the location information.

The above two changes have meant that the application no longer needs to be told at start-up the name of the Queue Manager, the Queue name and/or whether to connect directly or as a client since all of these are now specified in the location information. This has meant that the format of the MQMON.CFG file has been changed. For example there is now no longer the need to have multiple 'AP:' entries. Most of this will be transparent to the user however it will be necessary to go through the locations when the program is run for the first time and say for each location whether the Queue Manager is accessed directly, via another Queue Manager or as a client. My apologies for this but it is unavoidable since I do not have the information to hand.

Migration from a version prior to Version 4.0

Functionality

The general look and feel of Version 4.0 was virtually the same as previous versions however the internal workings of the code was changed quite extensively.

Main changes

The main differences from previous versions are:-

- Main window can be displayed as a container/tree view
  The Queue Manager objects are displayed if the location is expanded.

- Loop-back monitoring
  This means that it is no longer necessary to have user code at the monitored location.
- Positions, Sizes, Filters and display options of dialogs can be saved
- User can ask for confirmations of sensitive commands
- Configuration is saved per Queue Manager
- Connections to the Queue Manager can be made via a client
- Configuration is automatically saved on application termination.

Configuration

Version 4.0 introduced a changed configuration file format. The simple line orientated file in MQMONP.LST has been replaced by a keyword orientated file called MQMON.CFG. The old file is not converted to the new file, you must add your Queue Manager locations again in the version 4.0 program. The MQMON.CFG file is an editable file and its format is given in an Appendix at the end of this document, so if you have a large number of Queue Managers and you're feeling brave you can create the file in an editor rather than going through all the dialogs. Perhaps the simplest solution would be to run the program once, add a single Queue Manager entry, and then end it. This will create a file with one Queue Manager entry. You can then replicate the lines to create the other destinations you need.

Appendix A: Icons

These are the icons used in MQMONNTP, together with brief descriptions of their meaning.

Table 1: Queue Manager Icons

	Queue Manager - unknown state
	Queue Manager - Available state
	Queue Manager - Unavailable state
	Queue Manager - Waiting reply
	Queue Manager Group - unknown state
	Queue Manager Group - No queue managers are unavailable
	Queue Manager Group - One or more queue managers are unavailable

Table 2: Queue Icons

	Queue List
	Local queue - Unknown depth
	Local queue - less than 20% full
	Local queue - less than 40% full
	Local queue - less than 60% full
	Local queue - less than 80% full
	Local queue - over 80% full
	Alias queue
	Model queue
	Remote queue

A small ‘c’ character will be displayed on the icon if the queue is representing a cluster queue definition.

Table 3: Channel Icons

	Channel - unknown status
	Channel - STOPPED status
	Channel - RUNNING status
	Channel - other status (e.g. RETRYING)

Table 4: Other Object Icon

	Process
	Empty Namelist
	Namelist with one name
	Namelist with multiple names
	Topic

Table 5: Network View Icons

	Queue Manager definition list empty
	Queue Manager definition list populated
	Queue Manager definition list is populated but the definitions are ‘old’. The setting of what is regarded as old is set in the ‘Network’ tab of the preference dialog.
	Queue Manager definition list refreshing
	One or more channels running
	One or more channels retrying (none running)
	One of more channels stopped (none running or retrying)

Table 6: Container Icons

	Sub tree expanded
	Sub tree expanding/contracting
	Sub tree contracted
	Option selected
	Option deselected

Figure 20:Icons

Appendix B: Problem Selection List

This appendix contains the current list of problems that can be detected using the Verify Network feature in MQMONNTP. In the descriptions shown below, '%q' and '%r' represent queue names, '%c' and '%d' represent channel names, '%m' represents a queue manager name and '%l' represents either a cluster name or a cluster namelist.

- Queue %q target queue does not exist
- Queue %q is an alias to an alias queue
- Queue %q is an alias to a model queue
- Queue %q no transmission queue
- Queue %q invalid transmission queue
- Queue %q invalid Queue Manager name
- Channel %c no partner channels found
- %m No Dead Letter Queue defined
- Channel %c message size greater than DLQ %q
- Queue %q invalid initiation queue %r
- Queue %q uses an invalid process
- Transmission Queue %q has no channel definition
- Transmission Queue %q message size greater than channel %c
- %m Default transmission queue %q circular definition
- Channel %c has invalid transmission queue
- Channel %c has queue which is not a transmission queue
- Channel %c has message size greater than partner channel %d
- Queue %q initiation queue not found
- %m Invalid Dead Letter Queue defined
- %m Default transmission queue invalid
- Queue %q could not resolve to target queue
- Only one repository for cluster %l
- No repository found for cluster %l
- CLUSSDR missing for cluster %l
- CLUSRCVR missing for cluster %l
- Queue %q has an invalid clusnl %l
- Queue %q has an invalid cluster %l
- Channel %c has an invalid clusnl %l
- Queue Manager has an invalid reposnl %l
- Channel %c has a different Heartbeat interval than partner channel %d
- Channel %c has a different NPM Speed than partner channel %d
- Channel %c has a different BatchSize than partner channel %d

Appendix C: Display National Language Characters

In displaying a formatted message there are 3 codepages involved.

- The codepage of the message stated by the CCSID
- The codepage for the program reading and displaying the message, depending on the language of Windows
- The codepage used by the font to display the contents of a message, depending on the locale settings

In most cases when only ASCII characters (characters < x'80') are used these settings are not critical, however when other characters, like umlauts and the Euro-sign, are used these setting become more critical. As an extra handicap, with the introduction of the Euro Microsoft simply added the Euro to their codepages while IBM created new codepages.

The necessary translations from the program to the display are controlled by the locale Preferences field, selecting the correct language will be sufficient in most cases. The translation between the message and the program is controlled by the Browse CCSID Preference option and Convert option.

The table shows the recommended setting which will suffice in most cases.

Language	Codepage	
	Windows NT/2000[1)]	Browse CCSID[2)]
Arabic	1256	5352
Brazilian	1252	5348
Chinese (Simplified)	936	936
Chinese (Traditional)	950	950
Chinese (Hong Kong)[3)]	950	950
Czech	1250	5346
Danish	1252	5348
Dutch	1252	5348
English	1252	5348
Finnish	1252	5348
French	1252	5348
German	1252	5348
Greek[3)]	1253	5349
Hebrew	1255	5351
Hungarian	1250	5346
Italian	1252	5348
Japanese	932	932
Korean	949	949
Norwegian	1252	5348
Polish	1250	5346
Portuguese[3)]	1252	5348
Russian	1251	5347
Spanish	1252	5348
Swedish	1252	5348
Turkish	1254	5350

Figure 21:Codepages

1. For non-Western European language versions of Windows NT/2000 (codepage <> 1252), this value can be changed during installation of Windows NT/2000 and can be different from the value shown here.
   This information is derived from
   www.microsoft.com/globaldev/reference/oslocversion.mspx
4. This information is derived from
   www-3.ibm.com/software/ts/mqseries/support/euro/eurotabl.html
5. These languages are not supported in Windows NT

Appendix D: Configuration File

The configuration is file is a human readable, editable file called MQMON.CFG which contains a number of keyword value pairs. It is possible to construct or modify this file by hand. However, unless there is good reason to manually change the file I suggest you make all the changes using the application itself. It is well worth making a backup copy of the configuration file if it contains many settings since it is non-recoverable if it gets damaged or lost.

Note that not all keywords are used in the Windows NT version.

The format is described in a pseudo-meta language.

{}	Zero or more repetitions of what's contained in the braces
[]	optional element
|	Alternatives
<>	Supplied value
""	Literal value

MQMON.CFG := Instance
{GlobalDialog}
[QObjList]
[ProcObjList]
[ChlObjList]
[NLObjList]
{QMgrBlock [ClientBlock] [MQSCBlock] {DialogBlock} }

Instance := "AP:"

[x = <pos>]	X position of main window
[y = <pos>]	Y position of main window
[cx = <pos>]	Width of main window
[cy = <pos>]	Height of main window
[vx = <pos>]	X position of verify window
[vy = <pos>]	Y position of verify window
[vcx = <pos>]	Width of verify window
[vcy = <pos>]	Height of verify window
[prnsp = <value>]	Print spacing
[pmvm = <value>]	Print vertical margin setting
[prnhm = <value>]	Print horizontal margin setting
[prnvo = <value>]	Print vertical overlap setting
[prnho = <value>]	Print horizontal overlap setting
[cntr]	Show container/tree view
[qms]	Display Queue Manager names, otherwise show location description
[name]	Show 'name' container view
[icon]	Show 'icon' container view
[mini]	Show 'mini-icons' container view
[tree]	Show main window as 'tree'
[list]	Show 'main window as ‘list'
[alarm]	Alarms are active
[monitor]	Monitoring is active
[confirms]	Show confirm dialogs
[mainobjs]	Show objects in main window
[nphex]	No hex character prediction
[nsaveobj]	No Queue Manager object saving
[nauto]	No auto starting of the command server
[nmulti]	Don’t show string lists in list dialogs
[nres]	No Queue path resolution
[nprob]	Don’t check Queue Manager definitions for problems
[naro]	Don’t auto-refresh object list when main window expanded

[ntb]	Don’t display dialog windows on taskbar
[nexist]	Don’t display existing dialog window by default
[savepos]	Save command dialog positions
[savesize]	Save command dialog sizes
[sys]	Resolve SYSTEM objects
[chsrate = <value>]	Channel status refresh rate
[chstype = <value>]	Channel status refresh type
[layout = <value>]	Network view layout type
[dsplev = <value>]	Verify dialog display levels
[npfont = <value>]	Nonproportional font
[snapsz = <value>]	Verify window snap size
[pom= <value>]	Preference options mask value
[po= <value>]	Preference options
[dspwnd = <value>]	Verify dialog windows displayed
[font = <fontname>]	Set window font name
[fontsize = <fontsize>]	Set window font size
[npfont = <value>]	Nonproportional font
[npfont size = <value>]	Nonproportional font size
[pfont = <value>]	Printer font
[pfont size = <value>]	Printer font size
[loc = <value>]	Current locale setting
[ccsid = <value>]	Current character set setting
[expfile = <value>]	Last used Export file
[wsnd = <value>]	Warning sound .WAV
[asnd = <value>]	Alarm sound .WAV
{ clrn = <RGB number> }	Set area colour

GlobalDialog := "GDG:" <DialogName>
{ListBlock}

QObjList:= “OBQ:” {<Queue Name>}

ProcObjList := “OBP:” {<Process Name>}

ChlObjList:= “OBC:” {<Channel Name>}

NLObjList:-= “OBN:” {<Namelist Name>}

QMgrBlock := "QM:" <QMgrName> <Description> <Machine Type> <Command Level>

[viaqm = <value>]	Via Queue Manager for this location
[rq = <value>]	Set reply queue name for location if not 'MQMON'
[mq = <value>]	Set monitor queue name for location if not 'MQMON'
[cq = <value>]	Set command queue name for location if not normal value
[sq = <value>]	Server queue name for the location
[mvs]	Location is an MVS system
[alarm]	Alarms are active for this location
[auto]	Automatically connect
[client]	Connect as a client
[loop]	Use loop-back monitoring
[nc]	No commands allowed
[network]	Location is part of the 'network'
[single]	Connect with a single thread
[grp = <value>]	Group name for this location
[nnames = <value>]	Network names for this location
[disabled]	Monitoring is disabled for this location
[retryint = <value>]	Set retry interval, if not default

	[discint = <value>]	Set disconnect interval, if not default
	[interval = <value>]	Set monitoring interval, if not default
	[cmdok = <value>]	Set command ok value for this location
	[cmdfail = <value>]	Set command fail value for this location
ClientBlock:=	“CLN:”	
	{ PCFID = Value }	PCF ID and value pairs for client definition
MQSCBlock:=	“MQS:” {<MQSC Command>}	
DialogBlock :=	"DLG:" <DialogName>	
	[x = <pos>]	X position of this dialog type
	[y = <pos>]	Y position of this dialog type
	[cx = <pos>]	Width of this dialog type
	[cy = <pos>]	Height of this dialog type
	[refresh = <value>]	Automatic refresh rate for this dialog type
	[buttons]	Display buttons on this dialog type
	[fields]	Display fields on this dialog type
	[filter]	Display filter fields on this dialog type
	[initrefresh]	Automatically refresh this dialog type
	[dftfilter = <value>]	Default filter for this list
	[ListBlock]	
	[FilterBlock]	
ListBlock :=	"LST:" <Number Attribute Pairs> { Attribute Pairs }	
FilterBlock :=	"FLT:" { <FilterString> }	

- **Figure 22:MQMON.CFG format**

Index

Q

R

S

T

V

W

X

Y

Z