MQSeries Everyplace for Multiplatforms – MQe_Explorer

User Guide: v1.27a

SupportPac Category 3 – December 2002

Barry Aldred
IBM Corporation
Hursley Park
Winchester
UK
SO21 2JN

barry_aldred@uk.ibm.com

Take Note!

Before using this report be sure to read the general information under "Appendix C: Notices" on page 201.

License warning

MQSeries Everyplace – MQe_Explorer version 1.27a is supplied under the terms of the International Program License Agreement, a copy of which is reproduced in “Appendix D” on page 211. Defect correction will be provided under that agreement for users holding valid MQSeries Everyplace deployment license(s) until the end of service date, June 30, 2003.

Please refer to *http://www.ibm.com/software/mqseries* for details of the license conditions pertaining to the MQSeries Everyplace product.

Eight Edition, December 2002

This edition applies to version 1.27a of *MQSeries Everyplace for Multiplatforms – MQe_Explorer* and to all subsequent releases and modifications unless otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2000, 2001, 2002. All rights reserved. Note to US Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

Table of contents

Figures vii
Summary of amendments ix
Preface xi
*** Upgrade notice *** xi
Bibliography xii
Related material xii
Web references xiii
Download sites xiii
Newsgroups xiii
1 Introduction 1
1.1 Further reading 1
1.2 Support of MQSeries 1
2 Installation 2
2.1 Environments 2
Local execution and management environment 2
Remote management environments 2
2.2 Contents 2
2.3 Installation 4
Preparation 4
MQe_Explorer executable - new installation 5
MQe_Explorer executable - upgrade 5
Installing the MQe_Explorer classes 6
The MQSeries Classes for Java 6
3 Getting started 7
3.1 Configuring a first queue manager 7
3.2 Main window layout 9
3.3 Navigation and control 11
3.4 Setting properties 11
3.5 Creating and deleting objects 12
3.6 A simple network 13
3.7 Remote administration 13
3.8 Gateway administration 14
4 Main window menus 15
4.1 File 15
New 15
Open 57
Close 58
Properties 58
Exit 64
4.2 Edit 65
Clear 65
Cut 65
Copy 65
Delete 65
Paste 65
Select all 65
4.3 View 65
Console 65
Details 66
Hide empty 66
List 66
Off-line admin 66
Refresh 66
Small icons 66
Trace 66
Translate admin 66
Use multiple 67
4.4 Action 68

© Copyright IBM Corp. 2000, 2001

Get new credentials 68
Ping 68
Renew credentials 69
Start 69
Stop 69
Trigger local 69
Send File 70
Receive File 71
4.5 Tools 73
Configure remote 73
Demo mode 75
Load 76
Options 77
4.6 Window 90
Close all 90
Minimize all 90
Restore all 90
4.7 Help 91
About MQe_Explorer 91
Supervisor mode 91
5 The list view pane 92
5.1 Connections 93
5.2 Fields 94
5.3 Local queue managers 94
5.4 Messages 95
5.5 MQ bridges 96
5.6 MQ client connections 96
5.7 MQ listeners 97
5.8 MQ queue manager proxies 97
5.9 Queues 98
5.10 Queue manager folders 99
5.11 Remote queues 99
6 Off-line administration 100
6.1 Message properties 103
7 Off-line administration menus 104
7.1 Edit 104
Clear 104
Delete 104
Select All 104
7.2 File 104
Exit 104
Resend 104
7.3 View 104
Detail 104
Refresh 104
Requests 104
Replies 104
Others 105
8 Console 106
9 Console menus 108
9.1 Action 108
Increase priority 108
Decrease priority 108
Interrupt 108
9.2 File 108
Exit 108
9.3 Tools 108
Load 108
9.4 View 108
Refresh 108

© Copyright IBM Corp. 2000, 2001, 2002

10 Trace....109
11 Trace menus....111
11.1 Action....111
Call to debug....111
Debug....111
Error....111
Exception....111
Information....111
Security....111
System....112
Warning....112
11.2 Edit....112
Clear....112
Copy....112
Cut....112
Delete....112
Find....112
Paste....112
Select all....113
11.3 File....113
Exit....113
Log system.err....113
Save as....113
11.4 Format....113
Font....113
11.5 View....113
Object names....113
System.err....113
System.out....113
Thread names....113
Timestamp....114
12 Advanced topics....115
12.1 Configuration files....115
12.2 Initialization....116
Initialization files....116
Connection style....118
Default configuration....121
12.3 Invocation....122
Invocation options....122
Invocation parameters....122
12.4 Managing remote queue managers....123
Connections....123
Queues....123
Connectivity with remote clients....124
Connectivity with remote peers....124
Connectivity with remote servers and gateways....124
Automatic remote configuration....125
12.5 MQe_Explorer messages....125
12.6 Operation....126
Object cache....126
13 Sample scripts....129
13.1 Concepts and objects....130
Overview....130
Script....130
Resetting the script....144
13.2 Basic messaging....145
Overview....145
Scenario (direct & indirect routing)....145
Script....146
Resetting the script....152

© Copyright IBM Corp. 2000, 2001

13.3 Advanced messaging....153
Overview....153
Scenario....153
Script....154
Resetting the script....158
13.4 Gateway configuration and usage....159
Overview....159
Scenario....159
Script....159
14 Programming MQe_Explorer applications....168
14.1 Windows application shell....168
14.2 Non-windows application shell....169
14.3 Handling console interrupts....170
15 Appendix A: Performance tool....171
15.1 Menus and operations....172
16 Appendix B: Reference section....176
16.1 Icons and buttons....176
MQe object icons....176
Buttons....178
16.2 Initialization file sections....180
[MQe_Explorer section]....180
[QueueManager] section....181
[Registry] section....181
16.3 MQe_Explorer external variables....182
16.4 Accessibility features....183
Shortcuts....185
16.5 MQe classes....186
Class types....186
Class details....188
17 Appendix C: MQeFileTransferMsg class....202
17.1 Constructor....202
17.2 Methods....203
17.3 Usage notes....208
Sending files....208
Receiving files....208
18 Appendix D: Notices....209
Trademarks....210
19 Appendix E: International Program License Agreement....211
Part 1 - General Terms....211
Part 2 - Country-unique Terms....213

© Copyright IBM Corp. 2000, 2001, 2002

Figures

Figure 3-1: New MQe_Explorer window....7
Figure 3-2: New queue manager creation – General tab....8
Figure 3-3: Main MQe_Explorer window....8
Figure 3-4: Root expanded in main MQe_Explorer window....9
Figure 3-5: Queues folder expanded in main MQe_Explorer window....10
Figure 3-6: Property pages for the *FirstQM* queue manager....12
Figure 3-7: Main MQe_Explorer window of a simple network....13
Figure 4-1: New connection dialog....15
Figure 4-2: New message dialog....18
Figure 4-3: New MQ bridge dialog....20
Figure 4-4: New MQ client connection dialog....21
Figure 4-5: New MQ listener dialog....23
Figure 4-6: New MQ queue manager proxy dialog....25
Figure 4-7: New queue dialog....27
Figure 4-8: New queue dialog – General tab....29
Figure 4-9: New queue dialog – Properties tab....31
Figure 4-10: New queue dialog – Storage tab....32
Figure 4-11: New queue dialog – MQ bridge tab....33
Figure 4-12: New queue dialog – Security tab....34
Figure 4-13: Modify queue dialog – Certificates tab....36
Figure 4-14: New queue dialog – Destinations tab....37
Figure 4-15: New queue dialog – Aliases tab....38
Figure 4-16: New queue manager dialog....39
Figure 4-17: New queue manager dialog – General tab....42
Figure 4-18: Modify queue manager dialog – Summary tab....43
Figure 4-19: New queue manager dialog – Detail tab....44
Figure 4-20: New queue manager dialog – Aliases tab....46
Figure 4-21: New queue manager dialog – Comms tab....47
Figure 4-22: New queue manager dialog – Registry tab....49
Figure 4-23: New queue manager dialog – Security tab....50
Figure 4-24: Modify queue manager dialog – Certificates tab....53
Figure 4-25: New queue manager dialog – Class aliases tab....54
Figure 4-26: New queue manager dialog – Pre-loads tab....55
Figure 4-27: New queue manager dialog – Bridges tab....56
Figure 4-28: New queue manager dialog – Permissions tab....57
Figure 4-29: Typical connection property page....58
Figure 4-30: Typical MQ bridge property page....59
Figure 4-31: Typical MQ client connection property page....60
Figure 4-32: Typical MQ listener property page....61
Figure 4-33: Typical MQ queue manager proxy property page....62
Figure 4-34: Typical queue manager property page....63
Figure 4-35: Typical queue property page....64
Figure 4-36: Send file dialog....70
Figure 4-37: Receive file dialog....71
Figure 4-38: Remote configuration dialog – Local qMgr tab....73
Figure 4-39: Remote configuration dialog – Remote qMgr tab....74
Figure 4-40: Load Java class dialog....76
Figure 4-41: Options dialog – Classes tab....80
Figure 4-42: Options dialog – Connections tab....81
Figure 4-43: Options dialog – Fields tab....82
Figure 4-44: Options dialog – Local qMgrs tab....83
Figure 4-45: Options dialog – Messages tab....84
Figure 4-46: Options dialog – MQ bridges tab....85
Figure 4-47: Options dialog – MQ client conns tab....86
Figure 4-48: Options dialog – MQ listeners tab....87
Figure 4-49: Options dialog – MQ proxies tab....88
Figure 4-50: Options dialog – Queues tab....89
Figure 4-51: Options dialog – Properties tab....90

© Copyright IBM Corp. 2000, 2001

Figure 4-52: System information panel....91
Figure 6-1: Off-line administration – modify queue manager....100
Figure 6-2: Off-line administration – admin request assignment....101
Figure 6-3: The off-line administration window....101
Figure 6-4: Off-line request detail....102
Figure 8-1: The console window....106
Figure 10-1: The trace window....110
Figure 12-1: The *FirstQM* initialization file....117
Figure 12-2: The options dialog – Open qMgrs tab....118
Figure 12-3: The options dialog – Advanced-1 tab....121
Figure 12-4: Loading status....126
Figure 12-5: The options dialog – Advanced-2 tab....127
Figure 13-1: Basic messaging scenario....145
Figure 13-2: Advanced messaging scenario....153
Figure 13-3: MQSeries Explorer view....162
Figure 14-1: Basic Windows application shell....169
Figure 14-2: Standard application shell....170
Figure 14-3: Console interrupt programming....170
Figure 15-1: The performance tool – General tab....172
Figure 15-2: The performance tool – Test tab....173
Figure 15-3: The performance tool – Results tab....175
Figure 16-1: [MQe_Explorer] section entries....180
Figure 16-2: [MQe_Explorer] entry presence....180
Figure 16-3: General keystroke aids to navigation....183
Figure 16-4: Restricted keystroke aids to navigation....184
Figure 16-5: Keystroke shortcuts....185

© Copyright IBM Corp. 2000, 2001, 2002

Summary of amendments

Date	Changes
6 November 2000	Version 1.0 (initial release)
1 May 2001	Version 1.20 This version supports the creation of queue managers without the use of initialization files. It provides for the remote set-up of other queue managers for administration and configuration. Password security is handled. Full support is provided for both on-line and off-line management of MQe objects. MQe_Explorer can now be invoked with parameters or can be called from applications. There have been substantial functional and usability enhancements over the previous version. A tool is included for basic performance characterization.
14 May 2001	Version 1.21 Replacement of embedded MQe v1.20 classes with v1.21 classes.
10 August 2001	Version 1.23 Addition of support for the creation and management of the MQe bridge to MQSeries queue managers (MQe gateway capability). Replacement of embedded MQe v1.21 classes with v1.23 classes. Support for the mapping of the registry into a chosen storage adapter. Miscellaneous minor changes.
1 October 2001	Version 1.25 Addition of console support. Display of system information. Improved display of trace information. Additional support for the creation of new queue managers and the modification of existing local queue managers. Miscellaneous fixes & minor changes. Embedded MQe v1.23 classes replaced with v1.25 classes.
18 December 2001	Version 1.26 Additional properties supported for local queue managers. Services folder replaced by Bridges folder. Additional connection type and related changes. Accessibility improvements. Addition of a script for gateway configuration. Support for a description property for all bridge-related objects. Support for the message store property of queues. Addition of a ping capability. SupportPac classification upgraded to Category 3. Withdrawal of the MQe_ExplorerX.exe option. Miscellaneous fixes and minor changes.

© Copyright IBM Corp. 2000, 2001

14 June 2002	Version 1.27
	Additional support for the WTLS mini-certificate server and for WTLS mini-certificates. Additional properties available when creating a queue manager. Support for *MQeMQMsgObject* class test messages. File transfer capability added using either *MQeMQMsgObject* or *MQeFileTransferMsg* class messages; the latter providing segmented file transfer support. Full support for destinations in store and forward queues. Programming examples added for MQe_Explorer applications. Extensions to the performance tool to allow measurements to MQSeries destinations. Miscellaneous fixes and minor changes, including exploitation of the bridge-enabled queue manager property introduced in MQe 1.27.
1 December 2002	Version 1.27a
	Inclusion of support for client connections to an MQSeries queue manager, removing the need to install the MA88 SupportPac for local gateway function. Miscellaneous fixes and minor changes.

© Copyright IBM Corp. 2000, 2001, 2002

Preface

The MQSeries Everyplace MQe_Explorer is a management tool for MQSeries Everyplace (MQe). It allows MQe queue managers and their associated objects, such as queues, connections and bridges, to be locally or remotely configured. Similarly, messages on queues can be inspected and test messages sent to validate the operation of the network. MQe_Explorer also provides a simple way of creating local queue managers, which can then be further configured to meet the needs of applications.

A tool for gathering basic performance information is included.

*** Upgrade notice ***

If upgrading from a previous version of MQe_Explorer, please note:

- This version pre-reqs MQSeries Everyplace version 1.27 – you must install that level (or later) on any machine that is to run MQe_Explorer. Version 2 is not supported in this release.

- If you intend to use the local gateway function, i.e. the bridge to MQSeries, there is now **no** requirement to install the *MQSeries Classes for Java*. Such a local bridge can only operate in **client mode** however; there is no support for **bindings mode** in a Microsoft JVM.

- It is strongly recommended that all attached client, peer, server and gateway queue managers in the MQe network are also upgraded to MQe v1.27. MQe_Explorer will always send a *MQeBridgesAdminMsg* to any MQe queue manager which is at a lower version level; if upgraded, this message is only sent to gateway queue managers.

© Copyright IBM Corp. 2000, 2001

Bibliography

- *MQSeries Everyplace Version 1.2: Introduction,* IBM Corporation, SC34-5843
- *MQSeries Everyplace Version 1.2: Java Programming Guide,* IBM Corporation, SC34-5845
- *MQSeries Everyplace Version 1.2: Java Programming Reference,* IBM Corporation, SC34-5846
- *Websphere MQ Everyplace SupportPac EC01: MQSeries Everyplace: Configuration Guide*
- *Websphere MQ SupportPac MA88: MQSeries Classes for Java.*

Related material

- *Websphere MQ Everyplace SupportPac EA01: MQSeries Everyplace - XML conversion utility*
- *Websphere MQ Everyplace SupportPac EAP1: MQSeries Everyplace - Device code for Palm OS*
- *Websphere MQ Everyplace SupportPac ED01: MQSeries Everyplace - Get Started*
- *Websphere MQ Everyplace SupportPac ED02: Using MQSeries Everyplace with WebSphere Everyplace Server.*
- *Websphere MQ Everyplace SupportPac EP01: MQSeries Everyplace – Performance report*
- *Websphere MQ Everyplace SupportPac ES03: MQSeries Everyplace – WTLS Mini-Certificate Server*
- *Websphere MQ Integrator SupportPac ID03: MQSeries Integrator – Working with MQSeries Everyplace*

© Copyright IBM Corp. 2000, 2001, 2002

Web references

The following URLs provide useful resources for both MQSeries Everyplace and MQe_Explorer:

Download sites

IBM Websphere (MQSeries SupportPacs):

http://www.ibm.com/software/ts/mqseries/txppacs/

IBM Boulder (MQSeries Everyplace downloads):

http://www6.software.ibm.com/dl/mqsem/mqsem-p

IBM Visual Age Micro Edition (Java stacks & related technologies):

http://www.embedded.oti.com

Microsoft Corp. (JVM downloads):

http://www.microsoft.com/java/download.htm

Newsgroups

IBM Software Group (MQSeries Everyplace newsgroup):

news://news.software.ibm.com/ibm.software.websphere.mqeveryplace

© Copyright IBM Corp. 2000, 2001

1 Introduction

MQe_Explorer is a management tool for MQSeries Everyplace (MQe). It runs on Windows 2000 and other Windows platforms and can be used to manage one or more target MQe queue managers. Although the tool itself executes only on Windows platforms, the managed targets can be on any platform supported by MQe. MQe_Explorer has the Windows look and feel – mimicking the appearance of the *Windows Explorer* – and its use should be intuitive for those familiar with Windows applications. It is similar in some respects to the MQSeries version 5 Explorer used to manage distributed MQSeries, though it differs substantially in many details.

MQe_Explorer allows MQe queue managers and their associated objects, such as queues, connections and bridges, to be locally or remotely configured. Similarly, messages on queues can be inspected and test messages sent to validate the operation of the network. It provides a simple way of creating queue managers, which can then be further configured to meet the needs of applications. MQe_Explorer can be used to launch multiple MQe applications against a local queue manager in such a way that all applications share the same JVM. It can also remotely modify the configuration of existing queue managers such that they become amenable to MQe_Explorer inspection and change.

MQe_Explorer provides one of the fastest and easiest ways of getting started with MQSeries Everyplace. It provides a powerful visualization of an MQe network and is a useful educational tool both for understanding the relationships between MQe objects and the functionality of MQe.

1.1 Further reading

The *Websphere MQ Everyplace SupportPac EC01: MQSeries Everyplace Configuration Guide* provides detailed information and advice on MQe queue manager and network configuration. It also explains the mechanisms through which the MQe_Explorer controls MQe and provides sample code to allow similar functions to be embedded in application programs.

1.2 Support of MQSeries

MQe_Explorer includes full support for the configuration and management of gateway queue managers, i.e. those MQe queue managers that can bridge to MQSeries queue managers and queues. Those gateway queue managers created by MQe_Explorer *and* running in the same JVM, are restricted to interfacing to MQSeries queue managers through the MQSeries client interface (not the bindings interface). This capability is included with MQe_Explorer and does not require installation of additional software.

MQe_Explorer does not support the configuration of the MQSeries queue managers themselves – they should be configured using the various MQSeries management tools and protocols.

© Copyright IBM Corp. 2000, 2001

2 Installation

2.1 Environments

MQe_Explorer executes as an application running in a Java virtual machine. It can manage (and even create) its own local MQe queue manager and manage any remote MQe queue managers for which its local queue manager has MQe network connectivity.

Local execution and management environment

MQSeries Everyplace MQe_Explorer version 1.27a requires a Microsoft Windows environment[1] with the following characteristics:

- *Microsoft Java Virtual Machine* (at a level of version 5.00.3155 or later[2]).
- *Microsoft Foundation Classes* (MFC) – these are always present on the supported Windows operating systems.

Windows 2000 systems meet these requirements as does Windows NT 4.0 and Windows 98, with the correct level of Microsoft JVM installed. New installations of Windows XP do not include the JVM by default and therefore it must be downloaded from the Microsoft web site (see related footnote below). Upgrades to Windows XP preserve an existing JVM. If a choice of operating system is available, Windows 2000 or XP is preferred.

MQe_Explorer version 1.27a also requires a local installation of:

- *MQSeries Everyplace version 1.27 or a later version 1 release.*

Remote management environments

All environments (i.e. both Windows and non-Windows systems) supported by MQSeries Everyplace versions 1.0 – 1.27 may be remotely administered by MQe_Explorer version 1.27a.

2.2 Contents

In addition to this User Guide, the MQe_Explorer package includes the files:

- *MQe_Explorer.exe*

  The MQe_Explorer executable.

- *MQe_ExplorerC.exe*

  A self-extracting file containing MQe_Explorer classes and resources (that allows MQe_Explorer and its associated functions to be invoked by other Java classes).

---

[1] On Windows 95, 98 and ME there are minor functional deficiencies. Tooltips are not provided – affecting the provision of hover help for toolbar buttons, queue & message object class feedback on node selection in the object tree, etc.

[2] To determine the level of the Microsoft JVM type the command JVIEW at a DOS prompt. If the command is not recognized then your machine does not have a Microsoft JVM installed and any other JVM is not acceptable. JVM SDK downloads to install or upgrade are available from http://www.microsoft.com/java/download.htm.

© Copyright IBM Corp. 2000, 2001, 2002

Install the *MQe_Explorer.exe* executable to run the MQe_Explorer from a program icon, the Windows *Run* menu, or from a DOS command line. Additionally (or alternatively), install *MQe_ExplorerC.exe* to be able to invoke MQe_Explorer from a Java application or to gain access to the classes in the package *com.ibm.mqe.mqe_explorer*.

© Copyright IBM Corp. 2000, 2001

2.3 Installation

The following instruction is important:

> Capitalization is significant to both MQe and MQe_Explorer. At all times, both during installation and subsequent use, ensure that any text input matches the capitalization given in this document – this instruction applies to all names and strings (i.e. class names, alias names, directory names, file paths and names etc). Descriptive text is the only exception.

Before installation check that you have a Microsoft JVM installed on your machine and that it is at the correct level – see the details given in *Local execution and management environment* on page 2[3].

Preparation

The instructions below assume that the standard MQe v1.27 installation defaults have been adopted[4]. Thus, as a brief check on the MQe installation, confirm that a directory *C:\Program Files\MQe\Java\com\ibm\mqe* containing various *.class* files and other sub-directories (such as *\adapters*, *\administration* etc) exist. Also the directory *C:\Program Files\MQe\Java\examples* should also be present, again with many sub-directories (for example *\administration*). If these directories and files are missing you do not have a complete MQe installation and must re-install.

It is important to ensure that the *classpath*[5] variable has been set in accordance with the MQe installation instructions. On Windows systems there is a choice available between either setting the *classpath system variable,* or setting the *classpath user variable*. The system variable (illustrated below) will apply to all users of the machine; the user variable applies only to the selected user. Note that any value present for the user variable will over-ride the system variable value for that user.

On Windows 2000 & XP:

From the *Windows Start* button, go to *Settings*, then *Control Panel* and click on the *System* icon to bring up the *System Properties* panel. Click the *Advanced* tab followed by the *Environment Variables…* button. In the resulting *System Variables* section check the *classpath* definition. It should appear as:

classpath C:\Program Files\MQe\Java

Alternatively, the string "C:\Program Files\MQe\Java" should be present in the value.

If *classpath* is not present, click *New*…, then add "classpath" in the *Variable Name* input field and "C:\Program Files\MQe\Java" in the *Variable Value* field; click *OK*, then exit the previous panel via the *OK* button.

If *classpath* is present with the wrong value, then select *classpath*, click *Edit*…, and append the string ";C:\Program Files\MQe\Java" to the existing *Variable Value*. Click *OK*, then exit the previous panel via the *OK* button.

---

[3] If on starting MQe_Explorer you get the error message "Unable to start the application -- the Java Virtual Machine cannot be loaded. Class not registered", then the JVM is not at the correct level.
[4] Non-default MQe installations are acceptable if the appropriate name substitutions are made for those given in this document.
[5] Once MQe_Explorer is running, the value of the *classpath* variable is shown in *System Information*, accessed from the *Help→About MQe_Explorer* menu item.

© Copyright IBM Corp. 2000, 2001, 2002

On Windows NT:

From the *Windows Start* button go to *Settings*, then *Control Panel* and click on the *System* icon to bring up the *System Properties* panel. Click the *Environment* tab. In the *System Variables* section check the *classpath* definition. It should appear as:

classpath C:\Program Files\MQe\Java

Alternatively, the string “C:\Program Files\MQe\Java” should be present in the value.

If *classpath* is not present, click an item in the *System Variables* section, then add “classpath" in the *Variable* input field and “C:\Program Files\MQe\Java” in the *Value* field; click *SET*, then exit via the *OK* button.

If *classpath* is present with the wrong value, then select *classpath*, and append the string “;C:\Program Files\MQe\Java” to the existing string in the *Value* field. Click *SET*, then exit via the *OK* button.

On Windows 98:

From the *Windows Start* button go to *Run*… In the program name input field type “sysedit” and click *OK*. The *System Configuration Editor* appears; click on the C:\AUTOEXEC.BAT window to bring it to the front. In this file the line:

set classpath=C:\Program Files\MQe\Java

should be present, or alternatively the string “C:\Program Files\MQe\Java” should be present in the *classpath* value. If *classpath* is missing add the line shown above; if *classpath* is present with the wrong value, append the string “;C:\Program Files\MQe\Java” to the existing value. Go to *File*, click *Save*, then *Exit*.

If the *classpath* system variable is not correctly set MQe_Explorer (and any other MQe applications) will not be able to locate the MQe classes and will not execute.

The sample Windows NT authenticator (supplied with MQe) is used later in the sample scripts in this guide to demonstrate queue security. Ensure that you have made all the changes necessary for this authenticator to execute as per the MQe v1.27 installation instructions.

MQe_Explorer executable - new installation

The MQe_Explorer executable allows you to run the MQe management tool from the Windows desktop. If you have not previously installed a version of MQe_Explorer then:

1. Create a directory *C:\Program Files\MQe\Java\MQe_Explorer* and move the file MQe_Explorer.exe into it.

2. Create a desktop shortcut to MQe_Explorer by locating the *MQe_Explorer.exe* file in the *Windows Explorer*, right click it and select *Create Shortcut*. Drag the resulting shortcut to the desktop. Rename if desired.

MQe_Explorer executable - upgrade

If you have previously installed a version of MQe_Explorer executable, then:

1. In the directory *C:\Program Files\MQe\Java\MQe_Explorer* replace the existing *MQe_Explorer.exe* with the new file.

© Copyright IBM Corp. 2000, 2001

2. If you have created existing queue managers then leave their initialization files (**.ini*) and associated sub-directories. If you have no further use for any of these queue managers, then delete their files and sub-directories.

3. If you have extensive MQe_Explorer options created that you wish to save, then keep any configuration files named *MQe_Explorer.cfg* or other **.cfg* files that you have created. If you keep these files MQe_Explorer v1.27a will upgrade them – however you will not have the same class names and other values available as will new users. If in any doubt, delete the existing files.

Installing the MQe_Explorer classes

The MQe_Explorer classes allow you to launch and run the MQe management tool from within a Java application. To install the classes double click the file *MQe_ExplorerC.exe*. You will be prompted for a file location; if you have used the default MQe install then accept C:\Program Files\MQe\Java (or modify accordingly).

In the default case the install will add class files and resource files to *C:\Program Files\MQe\Java\com\ibm\mqe\mqe_explorer* and to *C:\Program Files\MQe\Java\examples\mqe_explorer*. Sub-directories will be created if needed.

The MQSeries Classes for Java

The *MQSeries Classes for Java* provide a Java interface to MQSeries queue managers, either locally (in bindings mode) or remotely (through an MQSeries client channel). When an MQe is bridge-enabled, i.e. configured as a gateway queue manager, it normally uses the *MQSeries Classes for Java* in order to exchange information with MQSeries queue managers. Current levels of these classes no longer support the Microsoft JVM, however MQe_Explorer 1.27a includes the necessary function for local MQe gateway queue managers to be created and to communicate with MQSeries queue managers over client channels. In this case there is no need to install the *MQSeries Classes for Java* and there are no associated *classpath* changes to be made.

© Copyright IBM Corp. 2000, 2001, 2002

3 Getting started

3.1 Configuring a first queue manager

Start MQe_Explorer by clicking on the desktop shortcut .

If this is a new installation (or an upgrade and you have deleted the old configuration file) then a message will appear indicating that no saved options have been found and that defaults will be used.

Click *OK* in the message box.

The main MQe_Explorer window then appears:

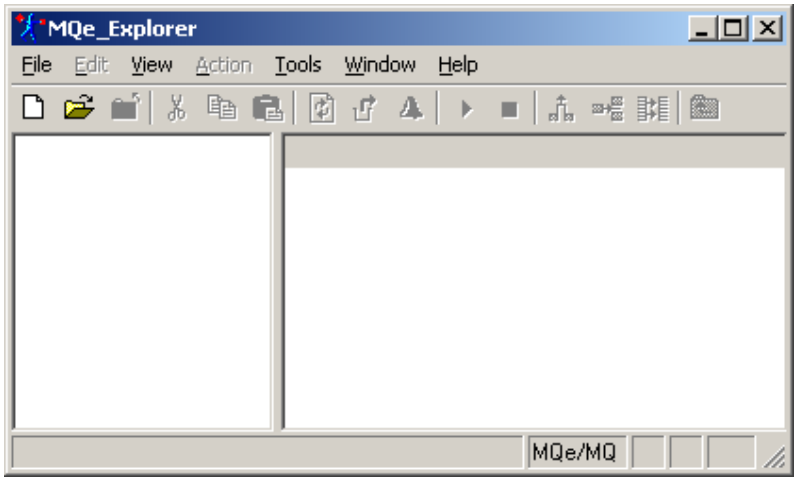


Figure 3-1: New MQe_Explorer window

© Copyright IBM Corp. 2000, 2001

Click the file *New* toolbar icon to create a new queue manager.

The window for queue manager creation appears:

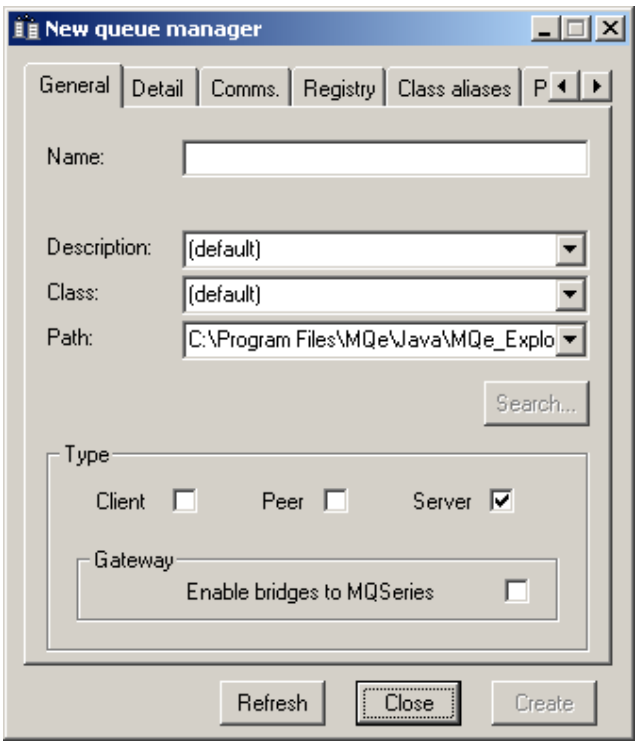


Figure 3-2: New queue manager creation – General tab

Type in the string "FirstQM" as the name of the new queue manager and click the *Create* button. A message box appears displaying the name of an initialization file – this name is important because, if in the future you want to restart the queue manager, you will need to open this file with MQe_Explorer.

Click OK to accept the message (making a note of the full file path for future use).

The queue manager is created and the main MQe_Explorer window changes appearance:

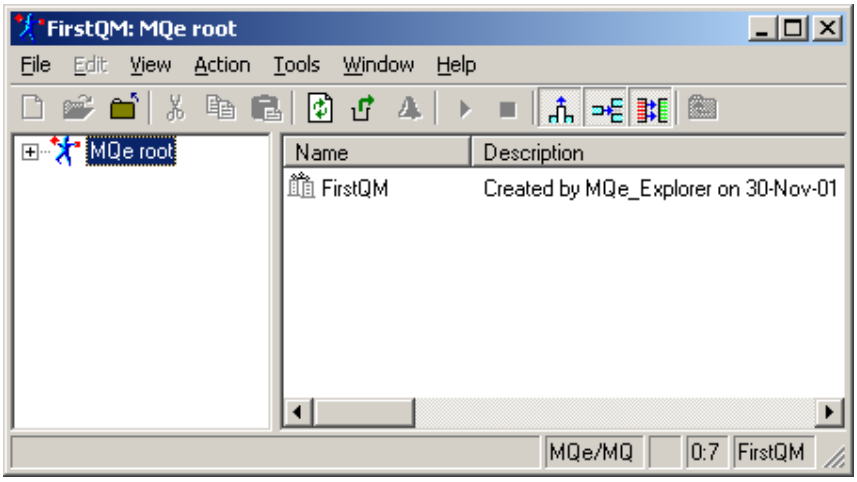


Figure 3-3: Main MQe_Explorer window

© Copyright IBM Corp. 2000, 2001, 2002

The new queue manager is now running; configured and loaded (by default) as a server. MQe_Explorer is running in the same JVM and as an application of that queue manager – other applications may now be loaded into that same JVM if required. If MQe_Explorer is subsequently shut down the fully configured MQe queue manager will still exist on the file system and may be later restarted. A structure of sub-directories has been created to store the configured queue manager and these are located with the initialization file in the file system.

If you are anxious to get started and are familiar with Windows, you may choose to skip the rest of this chapter and run the first sample script *Concepts and objects* on page 130.

3.2 Main window layout

The principal elements of the main window are a menu bar at the top, a toolbar below, two large panes (a tree view pane on the left, a list view pane on the right) and a status bar at the bottom. Individual menu options are described in a later chapter; the various toolbar buttons offer shortcuts to commonly used menu items.

The tree view pane presents a tree view of the MQe network.

Expand the ⊞ sign next to the *MQe root* icon.

The next level of the tree is visible:

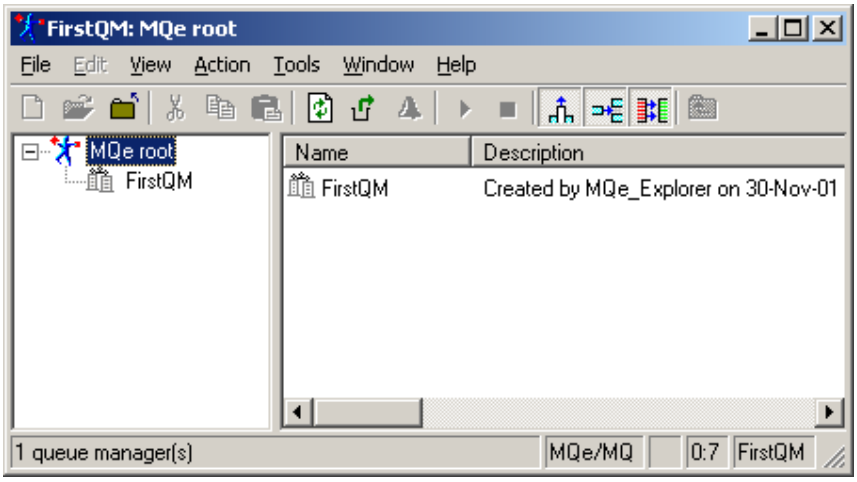


Figure 3-4: Root expanded in main MQe_Explorer window

Expanding the ⊞ symbol next to the queue manager icon reveals the next level, and so on. The queue manager icon has four forms: when a queue manager has responded with its details it has the appearance (unselected) (selected); however if no response has been received then it is shown as (unselected) (selected).

Other nodes can be expanded, just as were the *MQe root* and the *FirstQM* queue manager. For example, closed folders represented by this icon, can be selected and opened.

© Copyright IBM Corp. 2000, 2001

Select *FirstQM*. Click the ⊞ icon that appears next to *FirstQM* (note: a double click of a tree node has the same effect as a select followed by a click).

Click the folder *Local queues*. Expand the main window by pulling the right hand edge. Adjust the spacing between the tree view and list view panes by dragging the separator.

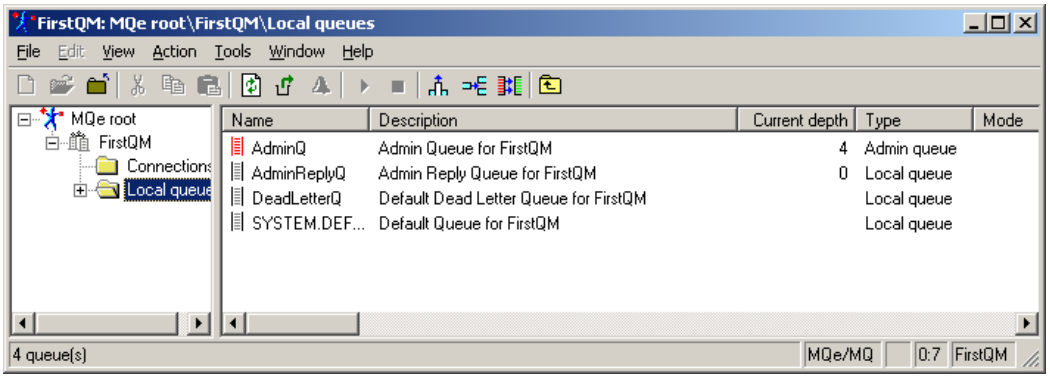


Figure 3-5: Queues folder expanded in main MQe_Explorer window

Besides folders and queue managers, many other tree node icons may be expanded, including connections, queues, messages, fields-type fields and many bridge-related icons. The window title changes to reflect the currently selected element in the tree. At the same time the list view pane shows the details of the child objects corresponding to the selected object in the tree. In the example above, the title indicates that the *FirstQM* queue manager is hosting MQe_Explorer, and that the *FirstQM* local queues folder is selected. The list view shows the four queues owned by *FirstQM*. Clicking the ⊟ symbol contracts the tree display.

The list view comprises rows and columns. Rows may be selected; in some views only a single row is selectable – in others multiple rows can be selected. Columns can be re-ordered by dragging the column header to a new position (that is, over other headers); columns can be re-sized by dragging the column boundaries in the header. Column re-ordering is not remembered when a view changes; however column resizing is remembered on a per object basis (that is, the width is associated with the selected object in the tree). Permanent control of column order, the columns to be displayed and their default widths, are all controlled through a *Tools→Options* menu panel – see page 73. Clicking a column header sorts the data by that column; re-clicking re-sorts in the reverse sequence. Double clicking a row in the list view pane is identical to selecting the equivalent object in the tree view pane.

The overall MQe_Explorer window may be resized and the divider between the tree and list view panes can be dragged to change the relative pane widths.

The status bar contains five panels; these are, from left to right:

- *Status and information message area*.
- *Evnvironment*.
- *Number of objects selected in the list view pane* (when more than one selected).
- *Object cache status* (see *Operation* on page 126).
- *Name of the local queue manager hosting MQe_Explorer*.

The status bar may be double clicked; this displays a panel with more information on object cache status.

© Copyright IBM Corp. 2000, 2001, 2002

3.3 Navigation and control

MQe_Explorer supports the standard Windows conventions for selection, shortcuts, context menus, tabbing through fields, keyboard operations, clipboard, and dragging and dropping.

Multiple object selection is supported (where appropriate) using shift key and control key combinations with the mouse select button. *Control A* selects everything in the list pane (when multiple selections are enabled). Keyboard shortcuts are displayed on the menu items where applicable. Right clicking on an object will reveal context menus for that object – and is supported for all list view and tree view objects. *Tab* will move to the next object in a panel or window – for ease of data entry. *Cut*, *copy* and *paste* functions to/from the clipboard is supported for message objects. Likewise *drag and drop* of message(s) from the list view pane to the tree view pane is permitted; be aware that only certain objects in the tree are valid drop targets. *Shift* held down whilst dragging indicates a move operation; *Control* held down, a copy operation. The complete set of supported keyboard operations is listed in *Accessibility features* on page 183.

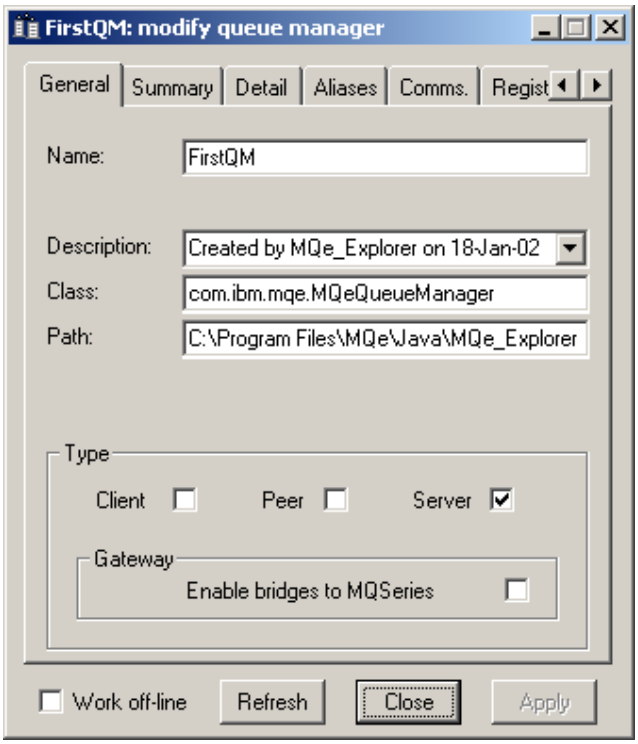
Hover help is available on many of the displayed objects. Of particular note is that, when a queue is selected in the tree view, hover help shows its class name. Similarly, when a message object (or a fields object) is selected in the tree, hover help shows the class name of the message object.

3.4 Setting properties

Properties are changed through the use of property pages.

Right click on *FirstQM* in the tree. In the context menu that appears, click the *Properties* sub-menu item.

The following window appears, displaying the property pages for the *FirstQM* queue manager:



© Copyright IBM Corp. 2000, 2001

Figure 3-6: Property pages for the *FirstQM* queue manager

The tabs each hold logical grouping of properties. Property values may be changed by editing the currently displayed value (where permitted); followed by clicking the *Apply* button to action the updates. The *Refresh* button causes the properties to be re-queried and displayed; any drop-down lists are re-loaded. *Close* removes the window. Property windows (and indeed most other MQe_Explorer windows) do not need to be closed immediately after use; they can either be left on the screen or minimized. MQe_Explorer is not modal in its operations and so for example, the main window can be used even when other windows are present. The *Window* menu item provides for control of open windows.

3.5 Creating and deleting objects

In a similar manner, objects are created through property pages. To create an object select the parent object, in either the tree view pane or the list view pane, and then use the *New* context menu item or the *File→New* menu item. Most objects can be created in this way but certain restrictions exist. For queue managers, only the local queue manager object can be created and only when no queue manager is currently loaded. Extra fields in messages cannot be directly created. Finally the bridges object (which enables a queue manager to be configured as a gateway) is not created explicitly – but by setting the type of queue manager (via *File→New→Queue Manager* or *File→Properties*).

Most other MQe objects can be freely created, including queues, connections, messages, MQ bridge objects, MQ queue manager proxies, MQ client connections and MQ listeners. In the majority of cases the parent object must be first selected – for example for a queue, the parent is the *Local queues* folder. However there are many other cases supported to aid usability, e.g. test messages can be sent to target queues by selecting almost any of the elements in the tree view pane. Likewise a *Test message* property page, however instantiated, is capable of sending a message to any queue in the network.

Objects are deleted by selecting them and pressing the *Delete key* (or by using the *Edit→Delete* option). One or more messages can be selected and deleted in a single operation; messages can also be cut, copied and pasted as well as dragged and dropped

When working off-line, i.e. when connectivity to the remote queue manager is not available, administration of queue managers, queues and connections is still possible. Since the parent object or the object to be managed cannot always be displayed, MQe_Explorer enables the relevant operation on a displayable ancestor object – thus to delete a queue, the off-line delete queue operation is enabled on the *Local queues* folder.

© Copyright IBM Corp. 2000, 2001, 2002

3.6 A simple network

The *Concepts and objects* script on page 130 creates a simple network. The figure below shows the MQe_Explorer main window view of this network shortly after creation:

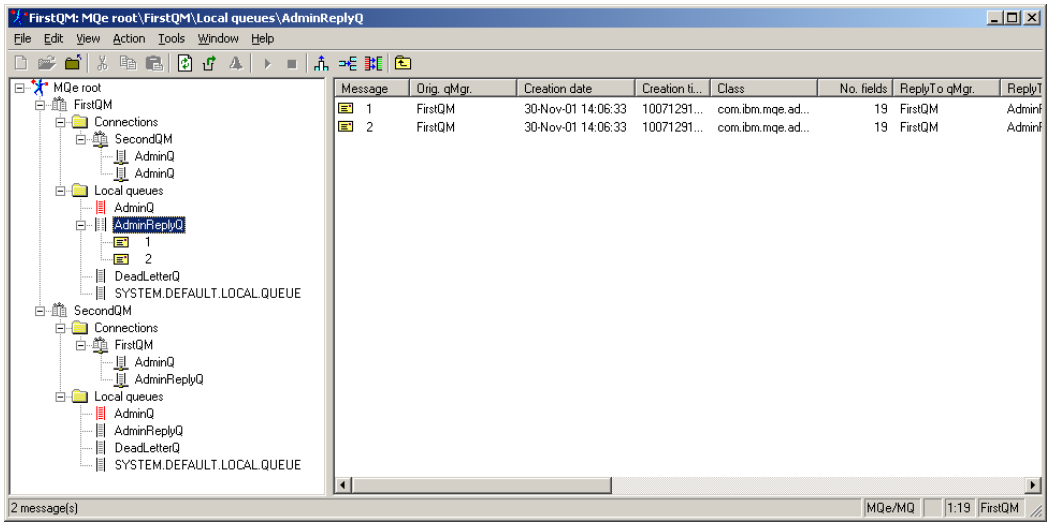


Figure 3-7: Main MQe_Explorer window of a simple network

The tree view is now more complicated than that shown previously – the *MQe ro*ot node has two child queue manager objects, representing the two queue managers in the network. Each of these has a number of local queues; each queue manager also has a connection defined to the other, and each knows a number of queues on the other.

The node selected is identified in the title bar as *MQe root\FirstQM\Local queues\AdminReplyQ*, that is, a local queue belonging to *FirstQM* called *AdminReplyQ*. This queue contains two messages, as indicated by the message in the status bar. Six properties, *Origin queue manager*, *Creation date*, *Creation time, Class, Number of fields* and *Reply to Queue Manager* are visible in the list view pane – with more properties present, but not visible, to the right (as apparent from the horizontal scroll bar in that pane). The object cache information is shown as “1:19” (which means that the properties of nineteen objects have been requested and one reply is still outstanding. The hosting queue manager for MQe_Explorer is *FirstQM*. The local environment supports gateway function.

3.7 Remote administration

As well as supporting the creation and management of a local queue manager, MQe_Explorer is also able to manage remote queue managers. To do this it requires MQe connectivity with those remote queue managers; and likewise they must be able to communicate with the local queue manager hosting MQe_Explorer. This involves the presence of both connection definitions and queue definitions on both queue managers. Management itself can be either take place on-line, which is the common case, and where the results of an admin operation are immediately executed and available; or off-line, where the execution of the admin operation occurs at some later time. In this book on-line administration is assumed to be the norm; the off-line aspects are deferred until the section *Off-line administration* on page 100. The details of the queue manager configuration required for MQe_Explorer administration are described in *Managing remote queue managers* on page 123.

© Copyright IBM Corp. 2000, 2001

3.8 Gateway administration

MQe_Explorer supports the creation and administration of gateway queue managers that have the ability to link an MQe network with an MQSeries network. The facilities are described under the relevant menu topics. A script, *Gateway configuration and usage* on page 159, illustrates setting up such a gateway from both MQe and MQSeries perspectives.

It is now recommended that you run the first script *Concepts and objects* on page 130. The information in the following chapters is more detailed and may be more useful after the first script has given an appreciation of MQe_Explorer operation. Subsequent scripts are concerned with setting up different messaging topologies and exploring more advanced MQe functions.

© Copyright IBM Corp. 2000, 2001, 2002

4 Main window menus

4.1 File

New

This command has eight sub-commands – the available one is determined by the nature of the object selected in either the tree view pane or the list view pane:

- *Connection* (select a *Connection folder*).
- *MQ bridge (*select a *Bridges object).*
- *MQ client connection* (select an *MQ queue manager proxy).*
- *MQ listener (*select an *MQ client connection).*
- *MQ queue manager proxy (*select an *MQ bridge.)*
- *Queue* (select either a *Local queues folder* or a *Connection*).
- *Message* (select a *Queue manager*, *Local queues folder*, *Local queue*, *Remote queue* or a *Connection*).
- *Queue manager* (requires that a local queue manager has not been loaded – no prior selection required).

Connection

File→New→Connection enables the *New connection* definition dialog (sometimes known as a new remote queue manager definition):

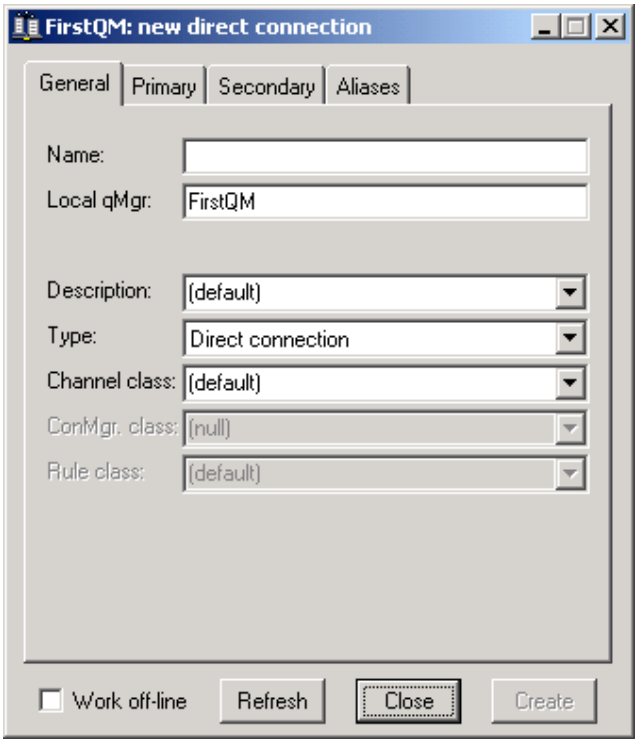


Figure 4-1: New connection dialog

© Copyright IBM Corp. 2000, 2001

Various types of connection can be created:

- *Alias-only*

  An alias-only connection is a specialized type of connection that is used to give alias names to remote queue managers that feature as destinations in store and forward queue definitions. No other parameters are required because the store and forward queue itself handles any associated delivery aspects.

  Alias-only connections share property pages with MQ connections.

- *Direct*

  A direct connection supplies the information needed for the local queue manager to create channels directly to another queue manager elsewhere in the MQe network. The connection name matches the name of that remote queue manager. Direct here means that the channels go to the remote queue manager without passing through an intermediate queue manager.

- *Indirect*

  An indirect connection (or via connection) indicates that messages destined for a remote queue manager elsewhere in the MQe network are to be sent to an intermediate queue manager (for which a direct connection will also exist). The connection name matches the name of the remote queue manager.

- *Local*

  This is a specialized type of connection that is actually created as a queue manager connection to itself. Local connections are best regarded as system definitions used by MQe to store local queue manager aliases and/or a peer listener definition. It should not be necessary to explicitly create local connections when using MQe_Explorer; they are automatically created as required. By the same token, local connections should not normally be deleted.

- *MQ*

  This is a specialized type of connection that identifies a remote queue manager as an MQSeries queue manager, as opposed to an MQe queue manager. MQ connections share property pages with Alias-only connections.

- *Peer listener*

  This is a specialized type of connection that is used to define a peer channel listener. For local queue managers created by MQe_Explorer it should not normally be necessary to define such a connection; MQe_Explorer will automatically create/delete one as required by the type setting of the local queue manager.

The dialog presents a number of different tabs, each relating to a related set of queue characteristics. The tabs are shared with the *Connection properties* dialog, used to change an existing queue. The complete set of tabs is:

- *General*

  The name of the connection, local queue manager and other properties generally supported by most connection types.

- *Primary*

  The network details and other related parameters.

© Copyright IBM Corp. 2000, 2001, 2002

- *Secondary*

  This tab refers to a secondary adapter to be used in association without the primary adapter to realize the connection. MQe does not yet support secondary adapters.

- *Aliases*

  Alternative names that should be resolved to this connection.

The following input fields may be present (note that many of the properties available depend upon the type of connection and the values of other properties):

- *General* tab:
  - *Name* – the name of the new connection (and therefore the name – or an alias – of the remote queue manager to be connected).
  - *Local queue manager* – the name of the queue manager owing this definition.
  - *Description* – any UNICODE string.
  - *Type* – connection type (see above).
  - *Channel class (or alias)* – the class that handles data transfers to the remote queue manager.
    - For a local connection defined as a listener: *(default)* is mapped to "com.ibm.mqe.MQePeerChannel".
    - For a remote direct connection: *(default)* is mapped to "com.ibm.mqe.MQeChannel".
    - For an Alias-only/MQ connection: *(default)* is mapped to null.
  - *Connection manager class (or alias)* – (*this property not yet supported by MQe)*.
  - *Rule class (or alias)* – (*this property not yet supported by MQe)*.

- *Primary* tab:
  - *IP address* – the numeric or string IP address of the machine hosting the remote queue manager (available only for direct connections).
  - *Via queue manager* – the name of the queue manager through which the connection should be routed (available only for indirect connections).
  - *IP port* – the port number used by the remote queue manager to service incoming connection requests (available only for indirect, remote connections).
  - *Adapter class (or alias)* – the class of the communications protocol adapter. Each protocol adapter supports a specific IP protocol, for example HTTP, TCP or UDP.
    - *(default)* is mapped to "com.ibm.mqe.adapters.MQeTcpipHistoryAdapter".

© Copyright IBM Corp. 2000, 2001

- *Options* – a series of ASCII options to be passed to the communications protocol adapter. The format for specifying options is to enclose each option value within angled brackets.
  - *(default)* is mapped to "<PERSIST><HISTORY>" when the "com.ibm.adapters.MQeTcpipHistoryAdapter" adapter class or the alias "FastNetwork" is specified; otherwise *(default)* is mapped to "".
  - *(none)* is mapped to "".
- *Parameters* – an ASCII parameter string to be passed to the primary adapter, for example for an HTTP connection, the name of the servlet.
  - *(default)* and *(none*) are both mapped to "".
- *Encoded parameters* – (*this property not yet supported by MQe)*.
- *Rule data* – (*this property not yet supported by MQe)*.

- *Secondary tab:*
  - MQe does not yet support secondary adapters; the *Connection manager class* should be set to default and no data should be entered into any input fields.

- *Aliases tab:*
  - *Alias names* – alias names for this connection. The *Add* button adds additional alias names entered in the input field; the *Delete* button deletes selected alias names.

The *Create* button creates the new connection and updates the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window. *Work off-line* allows connection definitions to be created on remote queue managers that are not currently available on the network.

Message

File→New→Message enables the *Send test message* dialog. By default it will generate and send a message of class *com.ibm.mqe.MQeMsgObject*; optionally it will use the *com.ibm.mqe.mqemqmessage.MQeMQMsgObject* class instead.

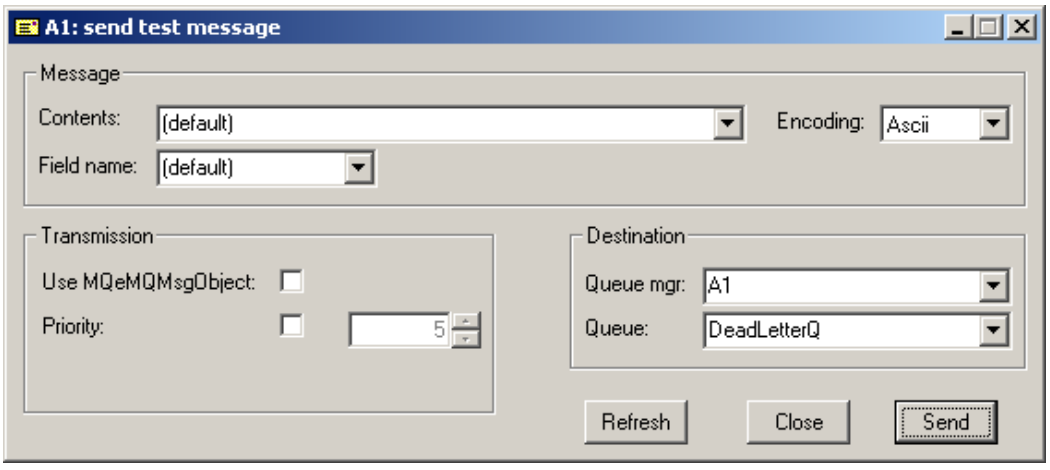


Figure 4-2: New message dialog

© Copyright IBM Corp. 2000, 2001, 2002

The following input fields are present:

- *Message* group:
  - *Message* – the text string of the message.
    - *(default)* generates a message that automatically includes the name of the sending queue manager, the date and time.
  - *Field name* – an ASCII string specifying name of the field containing the message text (only applicable when the message class is *com.ibm.mqe.MQeMsgObject*).
    - *(default)* is mapped to the string “Message”.
  - *Encoding* – the encoding to be used (Ascii or Unicode).
    - *(default)* is mapped to “ASCII”.
- *Destination* group:
  - *Queue manager* – the name of the target queue manager. This may be entered or selected from the drop-down list. Any name must identify a queue manager to which addressability exists from the queue manager hosting MQe_Explorer, i.e. it must be one of the following: the name of the hosting queue manager (or an alias); a connection name (or an alias); a destination name owned by a *store and forward queue* (or an alias).
  - *Queue* – the name of the target queue. This may be entered or selected from the drop-down list. Any name entered must either be known to the queue manager hosting MQe_Explorer (in conjunction with the target queue manager name) as a queue name (or alias), or otherwise will be assumed to identify a synchronous remote queue.
- *Transmission* group:
  - *Use MQeMQMsgObject* – the message object class used is *com.ibm.mqe.mqemqmessage.MQeMQMsgObject* (this class may be useful when sending to MQ queue managers as it is supported by the default bridge transformer).
  - *Priority* – if the box is unchecked the message will not contain a priority field; if checked a priority field with the selected priority (0 – 9) is added to the message.

The *Send* button sends the new message; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

© Copyright IBM Corp. 2000, 2001

MQ bridge

File→New→MQ Bridge enables the *New MQ bridge* definition dialog:

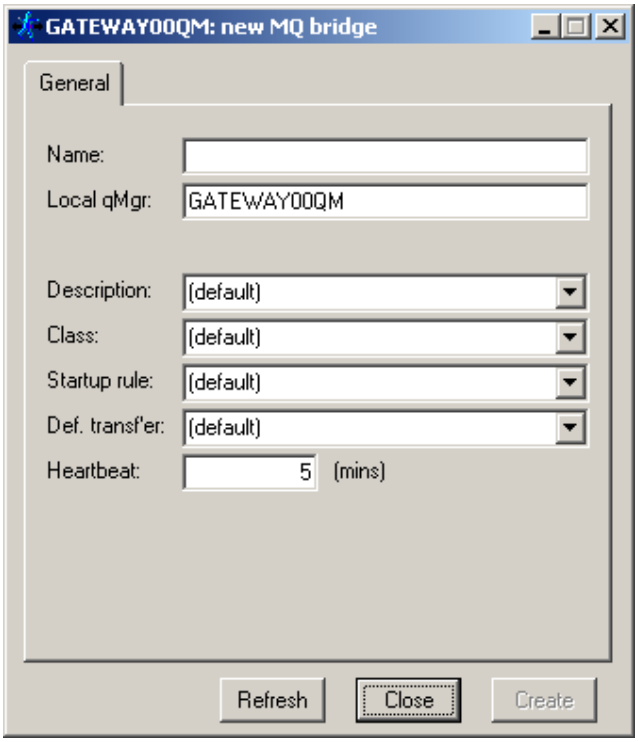


Figure 4-3: New MQ bridge dialog

The dialog presents a single tab with the MQ bridge characteristics. The tab is shared with the *MQ bridge properties* dialog, used to change an existing bridge.

The following input fields may be present:

- *General* tab:
  - *Name* – the name of the new MQ bridge.
  - *Local queue manager* – the name of the queue manager owing this definition.
  - *Description* – any UNICODE string..
  - *MQ bridge class (or alias)* – the class that implements the MQ bridge.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeMQBridge".
  - *Start-up rule class (or alias)* – the class that determines whether the MQ bridge is started at object creation time or when the owning queue manager is opened.
    - *(default)* is mapped to “com.ibm.mqe.mqbridge.MQeStartupRule”.
    - *(none)* is mapped to “” (i.e. no rule class). Objects are created in the stopped state.

© Copyright IBM Corp. 2000, 2001, 2002

- *Default transformer class (or alias)* – this class sets the transformer class used for message conversion – if not otherwise specified by either (a) the target queue (for messages being sent from MQe to MQSeries), and/or (b) the MQ transmission queue listener (for messages being sent from MQSeries to MQe).
  - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeBaseTransformer".
- *Heartbeat* – defines the frequency of pulses issued to other bridge-related objects. These pulses initiate housekeeping operations, such as the purging of the sync. queue. The value is in minutes within the range 1 – 60. The default is 5.
- *Status* – indicates whether the MQ bridge is in the stopped or started state.

The *Create* button creates the new MQ bridge and updates the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

MQ client connection

File→New→MQ Client Conn. enables the *New MQ client connection* definition dialog:

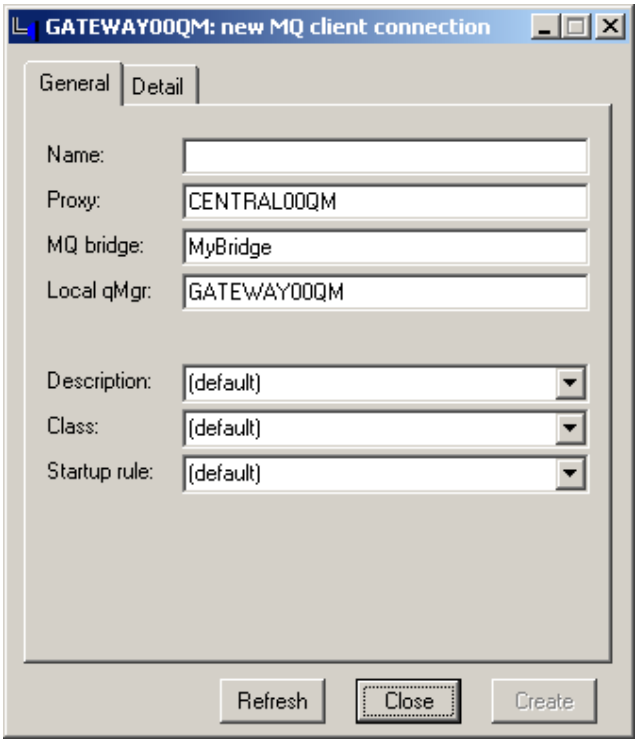


Figure 4-4: New MQ client connection dialog

The dialog presents tabs with the MQ client connection characteristics. These tabs are shared with the *MQ client connection properties* dialog, used to change an existing client connection.

© Copyright IBM Corp. 2000, 2001

The following input fields may be present:

- *General* tab:
  - *Name* – the name of the new MQ client connection.
  - *Proxy* – the name of the MQ proxy owning this definition.
  - *MQ bridge* – the name of the MQ bridge owning this definition.
  - *Local queue manager* – the name of the queue manager owing this definition.
  - *Description* – any UNICODE string..
  - *MQ client connection class (or alias)* – the class that implements the MQ client connection.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeClientConnection".
  - *Start-up rule class (or alias)* – the class that determines whether the MQ client connection and its parents are started at object creation time or when the owning queue manager is opened.
    - *(default)* is mapped to “com.ibm.mqe.mqbridge.MQeStartupRule”.
    - *(none)* is mapped to “” (i.e. no rule class). Objects are created in the stopped state.
  - *Status* – indicates whether the MQ client connection is in the stopped or started state.

- *Detail* tab:
  - *Port* – the port number used by the target MQSeries queue manager. If bindings mode is being used, this value should be set to “”.
  - *MQ adapter class (or alias)* – the class used to move messages from MQe to the target MQSeries queue manager.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeMQAdapter".
  - *User id* – the MQSeries user id.
    - *(default)* is mapped to “".
  - *Password* – the MQSeries password (note that the value typed is indicated only by * characters).
  - *Send exit* – used to match the send exit specified at the remote end of the MQSeries client channel.
    - *(default)* is mapped to “".
  - *Receive exit* – used to match the receive exit specified at the remote end of the MQSeries client channel.
    - *(default)* is mapped to “".
  - *Security exit* – used to match the security exit specified at the remote end of the MQSeries client channel.
    - *(default)* is mapped to “".
  - *CCSID* – the MQSeries CCSID property. The value must be a zero or positive integer; the default is 0.
  - *Sync. queue* – the name of the synchronization queue on the MQSeries queue manager that is used by the MQ bridge. This

© Copyright IBM Corp. 2000, 2001, 2002

queue keeps track of messages moving from MQe to MQSeries and, if the listener state store is also set to use MQSeries, also keeps track of messages moving from MQSeries to MQe.

- *(default)* is mapped to “MQE.SYNC.DEFAULT".

- *Purger rule class (or alias)* – the class that is used when a message on the sync. queue indicates a failure of MQe to confirm a message.
  - *(default)* is mapped to ""com.ibm.mqe.mqbridge.MQeSyncQueuePurgerRule".
- *Purger interval* – the time between successive purges of the synchronization queue. The value is in minutes and must be a zero or positive integer; the default is 60.
- *Max. idle* – the time after which an idle connection to MQSeries is discarded and the resources returned to the pool. The value is in minutes within the range 0 – 720. The default is 5.

The *Create* button creates the new MQ client connection and updates the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

MQ listener

File→New→MQ Listener enables the *New MQ listener* definition dialog:

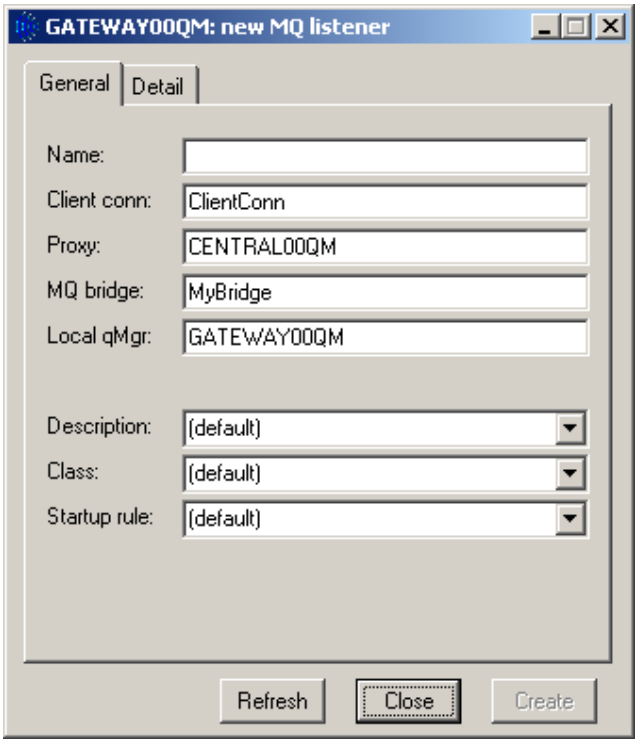


Figure 4-5: New MQ listener dialog

The dialog presents tabs with the MQ listener characteristics. These tabs are shared with the *MQ listener properties* dialog, used to change an existing MQ listener.

© Copyright IBM Corp. 2000, 2001

The following input fields may be present:

- *General* tab:
  - *Name* – the name of the new MQ listener.
  - *Client connection* – the name of the MQ client connection owning this definition.
  - *Proxy* – the name of the MQ proxy owning this definition.
  - *MQ bridge* – the name of the MQ bridge owning this definition.
  - *Local queue manager* – the name of the queue manager owing this definition.
  - *Description* – any UNICODE string..
  - *MQ listener class (or alias)* – the class that implements the MQ client connection.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeListener".
  - *Start-up rule class (or alias)* – the class that determines whether the MQ listener and its parents are started at object creation time or when the owning queue manager is opened.
    - *(default)* is mapped to “com.ibm.mqe.mqbridge.MQeStartupRule”.
    - *(none)* is mapped to “” (i.e. no rule class). Objects are created in the stopped state.
  - *Status* – indicates whether the MQ listener is in the stopped or started state.
- *Detail* tab:
  - *Dead letter queue* – the MQSeries queue used to hold messages that cannot be delivered from MQSeries to MQe.
    - *(default)* is mapped to "SYSTEM.DEAD.LETTER.QUEUE".
  - *State store* – specifies the permanent storage used to hold state information as messages are moved from MQSeries to MQe. The storage can be defined on either MQe or MQSeries.

    If MQe is to be used the value has the format of an MQe storage adapter, followed by a colon, followed by a path name to either a file or a directory. If a directory is specified then a file will be created there with the name *<listener name>-listener.sta*. If no parameter is specified a file will be created in the current working directory.

    If MQSeries is to be used then the MQ adapter should be specified with no parameters (i.e. “com.ibm.mqe,mqbridge.MQeMQAdapter”).
    - *(default)* is mapped to "com.ibm.mqe.adapters.MQeDiskFieldsAdapter".
  - *Undelivered rule class (or alias)* – the class determining the action taken when a message cannot be delivered from MQSeries to MQe.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeUndeliveredMessageRule".

© Copyright IBM Corp. 2000, 2001, 2002

- *Transformer class (or alias)* – the class defining the transformation used for message conversion for messages from MQSeries to MQe).
  - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeBaseTransformer".
  - *(null)* is mapped to a null value (i.e. use the default transformer defined in the MQ bridge).

The *Create* button creates the new MQ listener and updates the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

MQ queue manager proxy

File→New→MQ QMgr. Proxy enables the *New MQ queue manager proxy* definition dialog:

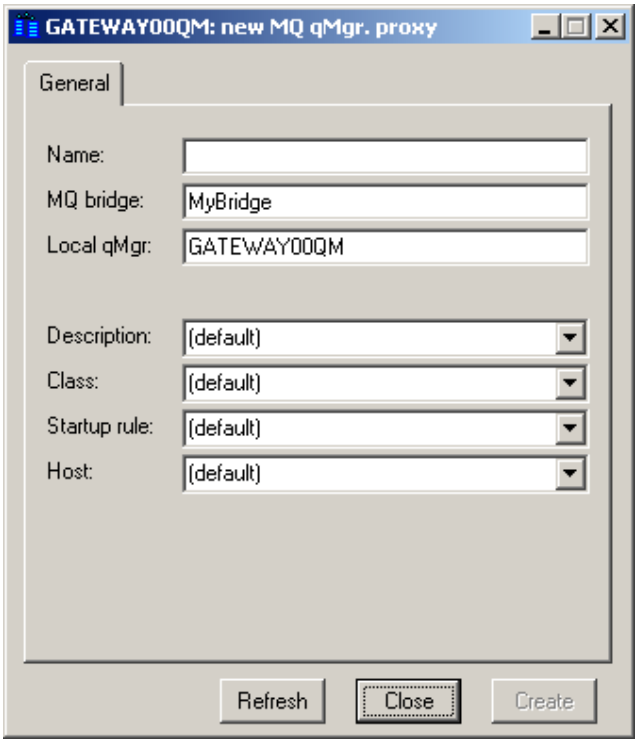


Figure 4-6: New MQ queue manager proxy dialog

The dialog presents a single tab with the MQ queue manager proxy characteristics. The tab is shared with the *MQ queue manager proxy properties* dialog, used to change an existing MQ queue manager proxy.

© Copyright IBM Corp. 2000, 2001

The following input fields may be present:

- *General* tab:
  - *Name* – the name of the new MQ queue manager proxy.
  - *MQ bridge* – the name of the MQ bridge owning this definition.
  - *Local queue manager* – the name of the queue manager owing this definition.
  - *Description* – any UNICODE string..
  - *MQ queue manager proxy class (or alias)* – the class that implements the MQ proxy.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeMQQMgrProxy".
  - *Start-up rule class (or alias)* – the class that determines whether the MQ queue manager proxy and its parents are started at object creation time or when the owning queue manager is opened.
    - *(default)* is mapped to “com.ibm.mqe.mqbridge.MQeStartupRule”.
    - *(none)* is mapped to “” (i.e. no rule class). Objects are created in the stopped state.
  - *Host* – the IP name or numeric address of the MQSeries queue manager. If “” is specified then the interface to MQSeries uses bindings mode, otherwise client mode.
    - *(default)* is mapped to "".
  - *Status* – indicates whether the MQ queue manager proxy is in the stopped or started state.

The *Create* button creates the new MQ queue manager proxy and updates the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

© Copyright IBM Corp. 2000, 2001, 2002

Queue

File→New→Queue enables the *New queue* definition dialog:

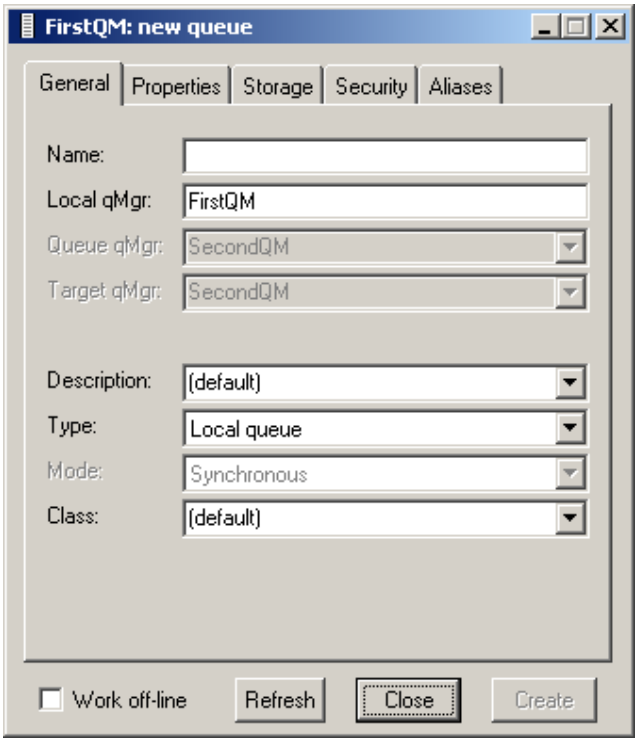


Figure 4-7: New queue dialog

Various types of queue can be created:

- *Admin*

  An admin queue is a queue that accepts admin messages. It processes those messages, executing their contained instructions. It may send replies to such messages. All queue managers should normally have one admin queue defined; otherwise the queue manager itself cannot be configured. The admin queue is normally created at the time the queue manager itself is created (see *Tools→Options→Advanced-1*). It is a convention that the queue is named 'AdminQ'. If off-line administration is also required it is a convention that the admin queue be given an alias name of 'OfflineAdminQ'. The contents of Admin queues are not accessible to applications.

- *Bridge*

  A bridge queue is a proxy for a remote queue on an MQSeries queue manager. Bridge queues are only defined on MQe gateways.

- *Home server*

  A home server queue is a local queue that points to a store and forward queue elsewhere in the MQe network. Home server queues pull messages destined for their local queue manager from those remote queues and deliver them to the relevant local queues. Home server queues should be regarded as system queues – messages are never addressed to home server queues and their contents are not accessible to applications.

© Copyright IBM Corp. 2000, 2001

- *Local*

  A local queue (sometimes, and more appropriately, called an application queue) stores messages. Its contents are accessible to applications – and most messages are addressed to local queues. The number of messages stored on the queue is available as the *current depth* property.

- *Remote*

  A remote queue definition is a proxy for a local queue on a remote queue manager elsewhere in the MQe network. This remote queue manager is known as the *queue queue manager*. The proxy sends messages to the remote local queue; additionally, if the proxy is defined as asynchronous, then it will temporarily store the messages locally until it is able to send them. Applications are not able to view messages awaiting transmission; attempts to browse remote queue definitions are interpreted as a browse of the remote local queue itself. The number of messages awaiting transmission is available as the *current depth* property.

- *Store and forward*

  A store and forward queue is a queue that collects those messages on its local queue manager that are destined for elsewhere in the MQe network. A single store and forward queue can collect messages addressed to one or more remote queue managers – these queue managers are referred to as *destination queue managers*. A store and forward queue can optionally send the messages it has collected to a remote queue manager – this named queue manager is called the *target queue manager*, i.e. the queue is forwarding messages to the target queue manager. In some cases it is desirable that the messages are just collected and not forwarded – the queue is now just a store, and there is no target queue manager defined. Store queues normally have messages pulled from them by home server queues. Store (and forward) queues should be regarded as system queues – messages are never addressed to store and forward queues and their contents are not accessible to applications. The number of messages awaiting transmission is available as the *current depth* property.

The dialog presents a number of different tabs, each relating to a related set of queue characteristics. The tabs are shared with the *Queue properties* dialog, used to change an existing queue. The complete set of tabs is:

- *General*

  The name of the queue, local queue manager and other properties generally supported by most queue types.

- *Properties*

  Additional detailed properties of a queue, the relevance depends upon the queue type.

- *Storage*

  Properties related to the storage of messages, specific to certain queue types.

- *MQ bridge*

  Properties specific to bridge queues.

- *Security*

  Queue security properties.

© Copyright IBM Corp. 2000, 2001, 2002

- *Certificates*

  Details of the WTLS mini-certificates stored in the registry relevant to the queue.

- *Destinations*

  A list of destination queue manager names, applicable only to store and forward queues.

- *Aliases*

  Alternative names that should be resolved to this queue.

The following input fields may be present (note that many of the properties available depend upon the type of queue, whether the queue is new or already exists, and the values of other properties):

The *General* tab has the appearance:

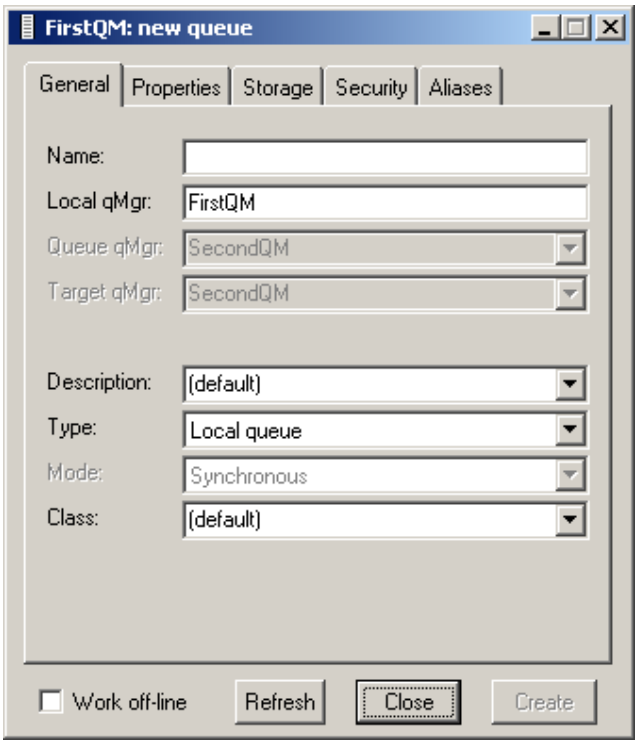


Figure 4-8: New queue dialog – General tab

© Copyright IBM Corp. 2000, 2001

The following input fields may be present:

- *General* tab:
  - *Name* – the name of the new queue.
  - *Description* – any UNICODE string.
  - *Local queue manager* – the name of the queue manager owing this definition.
  - *Type* – the type of queue (*admin, bridge, home server, local, remote, store and forward*).
  - *Queue queue manager* – this is available for *remote queues, home server queues* and *bridge queues* and is the remote queue manager that owns the (remote) physical queue.
  - *Target queue manager* – this is available for *store and forward queues* and is the queue manager to which the messages should be forwarded – for store queues this value should be set to *(none)*.
  - *Mode* – determines whether access is synchronous or asynchronous.
  - *Class (or alias)* – the class that implements the queue.
    - *(default)* is mapped as follows, depending upon the type of queue:

      admin: "com.ibm.mqe.MQeAdminQueue"

      bridge: "com.ibm.mqe.mqbridge.MQeMQBridgeQueue"

      home server: "com.ibm.mqe.MQeHomeServerQueue"

      *local: "*com.ibm.mqe.MQeQueue"

      remote: "com.ibm.mqe.MQeRemoteQueue"

      store and forward: "com.ibm.mqe.MQeStoreAndForwardQueue"
  - *Created* – the date/time that this queue was created.

© Copyright IBM Corp. 2000, 2001, 2002

The *Properties* tab has the appearance:

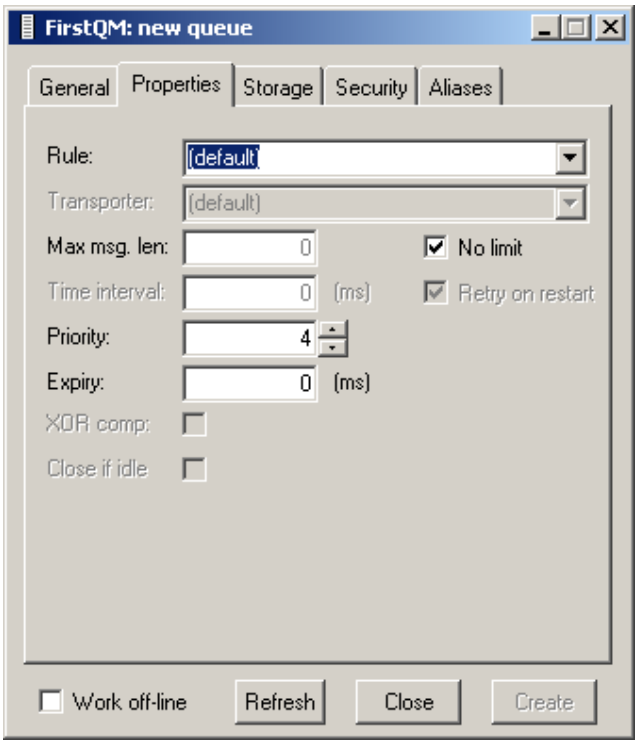


Figure 4-9: New queue dialog – Properties tab

The following input fields may be present:

- *Properties* tab:
  - *Rule class (or alias)* – the rule class to be associated with the queue.
    - *(default)* is mapped to “com.ibm.mqe.MQeQueueRule”.
    - *(null)* is mapped to a null value (i.e. no rule class).
  - *Transporter class (or alias)* – the transporter class to be used (handles the getting and putting of messages to/from remote queues).
    - *(default)* is mapped to "com.ibm.mqe.MQeTransporter".
  - *Maximum message length* – the maximum message length property, used by the queue rule. It must be a positive integer. The *No limit* check box takes priority and allows messages of unlimited length.
  - *Time interval* – the timer interval in milliseconds.
    - For an admin, if a positive integer, it is the time between successive attempts, for those cases where an attempt fails because resources are unavailable. The *Retry on restart* check box (equivalent to setting the time interval to zero) specifies that admin messages that cannot be actioned immediately should be retried when the queue manager is re-opened.
    - For a home server queue, if a positive integer, it is the time between the getting of pending messages. If a value of zero is specified then messages will only be got when a trigger

© Copyright IBM Corp. 2000, 2001

transmission event occurs (subject to the queue manager rule).

- *Priority* – the default priority associated with messages on the queue; the value must be a positive integer in the range 0 – 9.
- *Expiry* – the time in milliseconds after which messages on the queue expire, and are then removed. A value of zero indicates that messages never expire.
- *XOR compress* – if checked, the transporter will use XOR compression.
- *Close if idle* – if checked, the transporter will be closed once all the pending messages have been transmitted.
- *Status* – indicates whether the queue is active or inactive.

The *Storage* tab has the appearance:

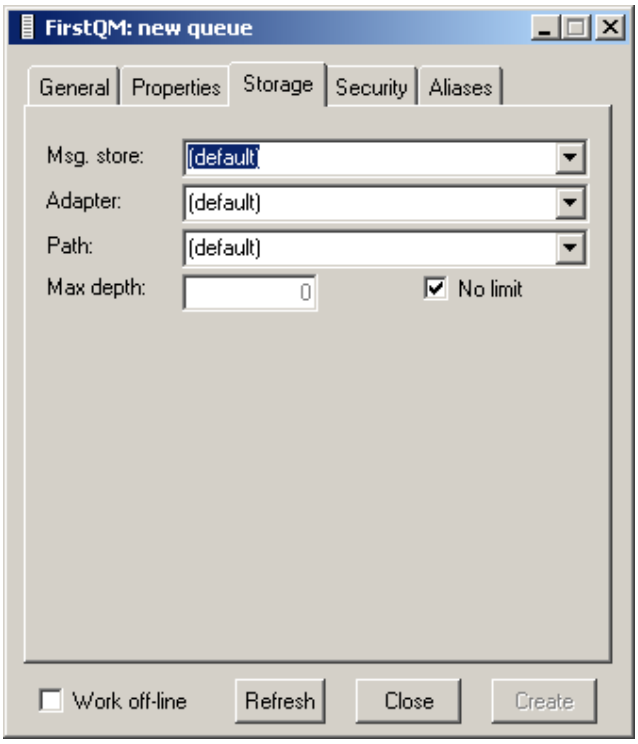


Figure 4-10: New queue dialog – Storage tab

© Copyright IBM Corp. 2000, 2001, 2002

The following input fields may be present:

- *Storage* tab:
  - *Message store (or alias)* – the message store class that determines the way that the queue adapter stores messages. If the value *(default)* is used for the *message store*, the *adapter* and the *path*, then the default message store, adapter and path associated with the queue manager is used; otherwise *(default)* is mapped to "com.ibm.mqe.messagestore.MQeMessageStore".
  - *Adapter (or alias)* – the queue adapter class that determines the backing store associated with the queue. If the value *(default)* is used for the *message store*, the *adapter* and the *path*, then the default message store, adapter and path associated with the queue manager is used; otherwise *(default)* is mapped to "com.ibm.mqe.adapters.MQeDiskFieldsAdapter".
  - *Path* – the path locating the physical storage of the queue – passed to the queue adapter. If the value *(default)* is used then the default path associated with the queue manager is used.
  - *Maximum queue depth* – the maximum number of messages allowed on the queue, used by the queue rule. The *No limit* check box takes priority and allows an unlimited number of messages.
  - *Current depth* – for an active queue only: the number of messages on the queue. For a store and forward or a remote queue: the number of messages awaiting transmission.

The *MQ bridge* tab has the appearance:

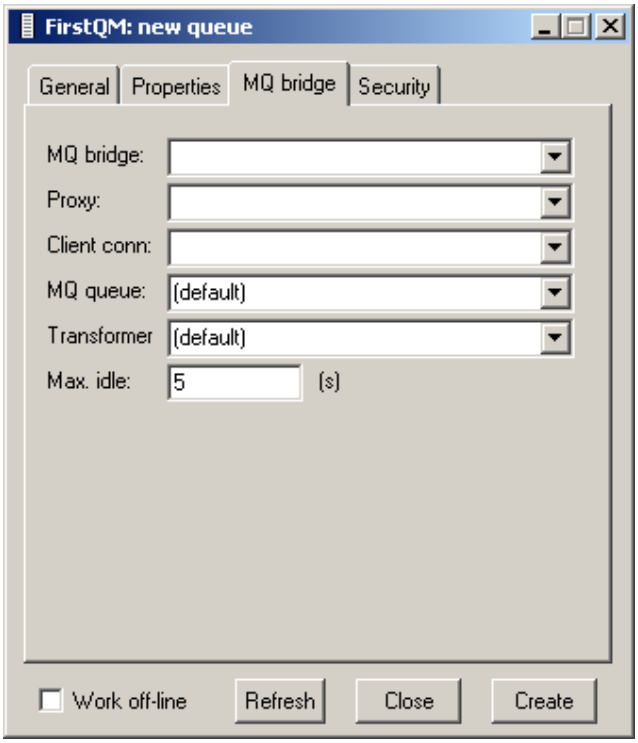


Figure 4-11: New queue dialog – MQ bridge tab

© Copyright IBM Corp. 2000, 2001

The following input fields may be present:

- *MQ bridge* tab:
  - *Bridge* – the bridge associated with the queue.
  - *MQ queue manager* – the name of the MQSeries queue manager associated with the queue.
  - *Client connection* – the name of the MQ client connection associated with the queue.
  - *MQ queue* – the name of the MQSeries queue associated with the queue.
    - *(default)* indicates that the queue name on the MQSeries queue manager has the same name as this bridge queue definition.
  - *Transformer class (or alias)* – the class that transforms messages being sent from MQe to MQSeries.
    - *(default)* is mapped to "com.ibm.mqe.mqbridge.MQeBaseTransformer".
    - *(null)* is mapped to a null value (use default transformer defined in the MQ bridge).
  - *Max. idle* – the maximum time (in seconds) that an MQ bridge queue is allowed to hang on to an idle connection before it is returned to the connection pool. The value must be zero or a positive integer; the default is 5.

The *Security* tab has the appearance:

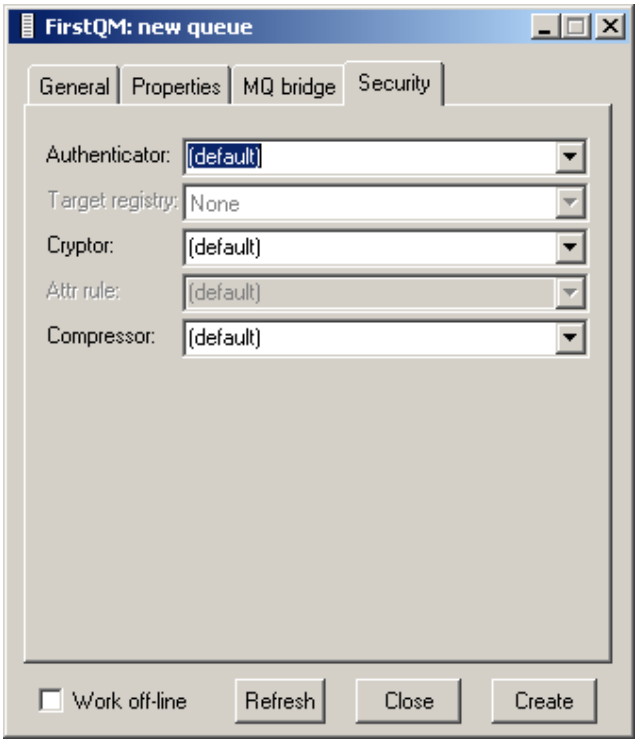


Figure 4-12: New queue dialog – Security tab

© Copyright IBM Corp. 2000, 2001, 2002

The following input fields may be present:

- *Security* tab*:*
  - *Authenticator class (or alias)* – the authenticator class associated with the security attribute assigned to the queue.
    - *(default)* is mapped to a null value.
    - *(null)* is mapped to a null value.

    If *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* is specified, mini-certificate based authentication will be used. The certificate used depends upon the value of the *target registry* property. If queue-based authentication is selected this will cause auto-registration with the MQe mini-certificate server. The mini-certificate server requires prior set-up such that it is allowed to issue a mini-certificate to the queue. Certificates are stored in the private registry and have a pre-determined lifetime, set when issued. They can be renewed through the *Actions→Renew Credentials* menu item[6].

    For more details see also the *Security* tab of the *Queue manager* on page 39.
  - *Target registry* – the target registry to be used by the authenticator. If queue registries are enabled, it has one of values: “None”, “Queue”, “Queue manager”; otherwise the 'Queue' value is not valid.
    - for the *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* class, *(default)* is mapped to “Queue manager”; for all others it is mapped to "None".

    For more details see also the *Security* tab of the *Queue manager* on page 39.
  - *Cryptor class (or alias)* – the cryptor class associated with the security attribute assigned to the queue.
    - *(default)* is mapped to a null value.
    - *(null)* is mapped to a null value.
  - *Attribute rule class (or alias)* – the rule class associated with the security attribute assigned to the queue.
    - *(default)* is mapped to "examples.rules.AttributeRule".
  - *Compressor class (or alias)* – the compressor class associated with the security attribute assigned to the queue.
    - *(default)* is mapped to a null value.
    - *(null)* is mapped to a null value.

[6] For more details of MQe security and auto-registration see the *MQSeries Everyplace Configuration Guide*.

© Copyright IBM Corp. 2000, 2001

The *Certificates* tab has the following appearance:

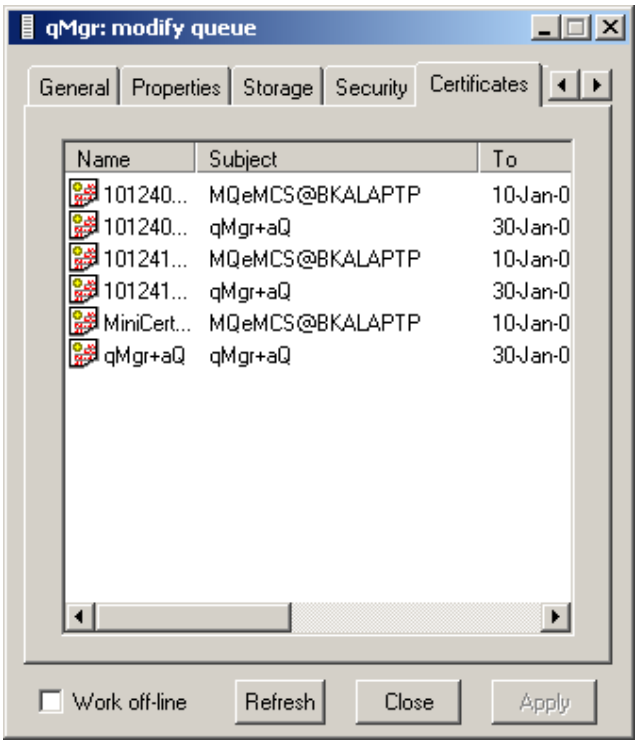


Figure 4-13: Modify queue dialog – Certificates tab

- *Certificates* tab:
  - *Name* – the name of the certificate[7]. Names that start with a number are old certificates that have been renamed.
  - *Subject* – the name of the queue authenticatable entity that owns the certificate.
  - *To* – the date after which the certificate is invalid.
  - *From* – the date from which the certificate is valid.
  - *Issuer* – the name of the entity issuing the certificate.

[7] MQe_Explorer removes the suffix "_MiniCertificate".

© Copyright IBM Corp. 2000, 2001, 2002

The *Destinations* tab has the appearance:

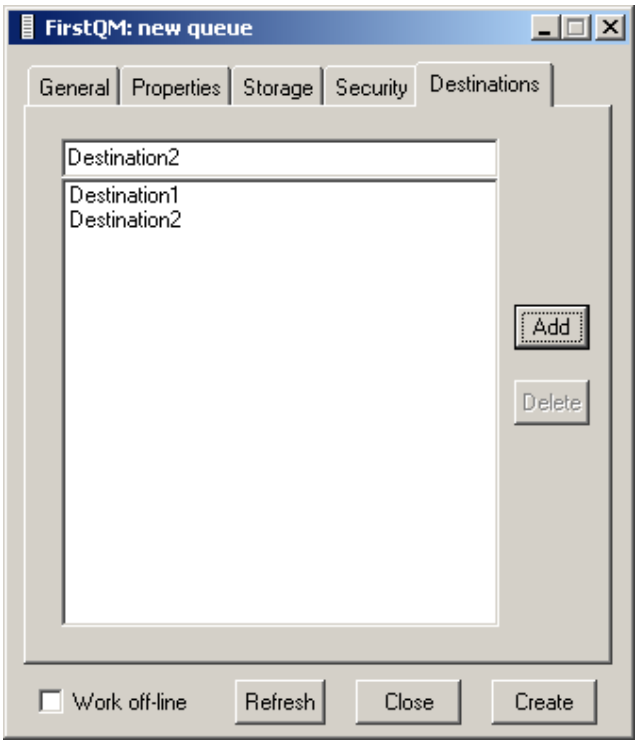


Figure 4-14: New queue dialog – Destinations tab

The following input fields are present:

- *Destinations* tab*:*
  - *Destination queue managers* – applicable only to store and forward queues and specifies the list of queue manager names for which this queue will hold messages. The *Add* button adds additional destinations entered in the input field; the *Delete* button deletes selected destinations.

© Copyright IBM Corp. 2000, 2001

The *Aliases* tab has the appearance:

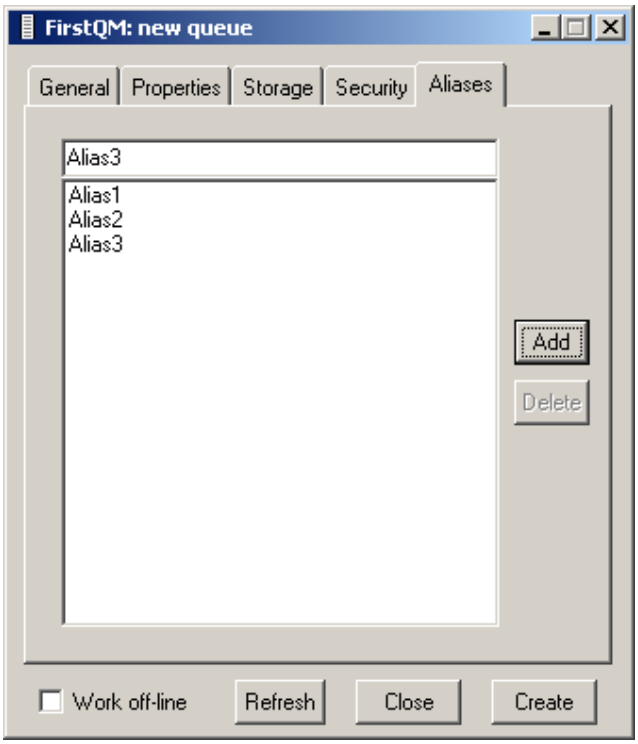


Figure 4-15: New queue dialog – Aliases tab

The following input fields are present:

- *Aliases* tab*:*
  - *Alias names* – alias names for this queue. The *Add* button adds additional alias names entered in the input field; the *Delete* button deletes selected alias names.

The *Create* button creates the new queue and updates the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window. *Work off-line* allows queue definitions to be created on remote queue managers that are not currently available on the network.

© Copyright IBM Corp. 2000, 2001, 2002

Queue manager

File→New→Queue Manager enables the *New queue manager* dialog that allows local queue managers to be created.

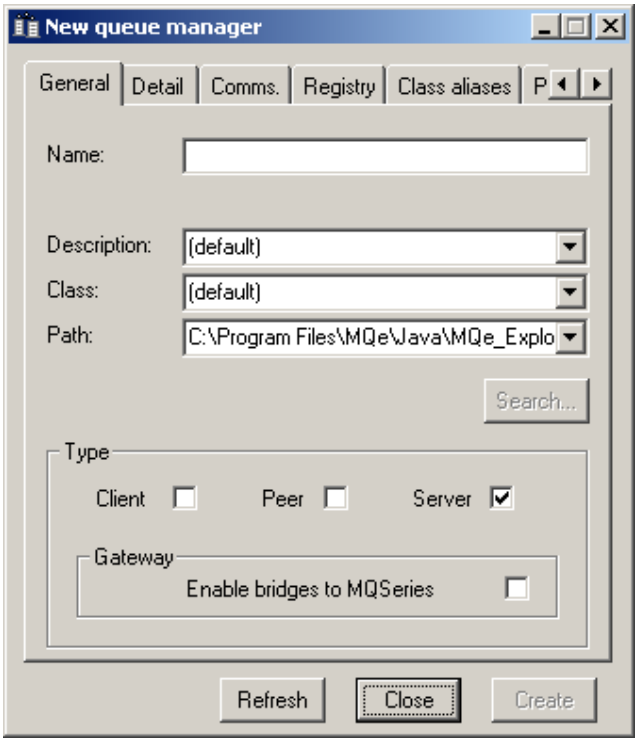


Figure 4-16: New queue manager dialog

The following queue manager types are supported:

- *Client*

  A queue manager that connects to one or more *server* queue manager(s) using client/server channels. It is able to initiate multiple outgoing connections but cannot accept incoming connection requests. Any data transfer over a client/server channel is initiated by the client.

- *Peer*

  A queue manager that connects to one or more *peer* queue manager(s) using peer channels. It is able to initiate multiple outgoing connections but can only accept one incoming connection request at a time. Any data transfer over a peer channel can be initiated by either peer. Note that a peer also has the ability to act as a client; thus it can also connect to servers via client/server channels and initiate data transfers to those servers.

- *Server*

  A queue manager that is connected to one or more *client* queue manager(s) using client/server channels. As a server is not able to initiate outgoing connections to a client but can accept multiple incoming connection requests from clients. Any data transfer over a client/server channel is initiated by the client. Note that a server also has the ability to act as a client; thus it can also connect to other servers and initiate data transfers to those servers.

© Copyright IBM Corp. 2000, 2001

- *Gateway*

  A queue manager that has all the properties of a *server* (as described above) but with the additionally capability of being able to exchange messages with MQSeries queue manager.

MQe_Explorer keeps track of the intended type of a queue manager – provided that it was originally created through MQe_Explorer[8]. If such a queue manager is subsequently modified such that its type is changed (e.g. a peer channel listener is added to a client queue manager), MQe_Explorer will reset the queue manager the next time it is re-started. The following rules are enforced:

- *Client*

  The following components are deleted, if present: peer channel listener, client/server channel listener, channel manager or a bridges object. A local connection will be retained.

- *Peer*

  The following components are deleted, if present: client/server channel listener, channel manager or a bridges object. A local connection will be retained or added; upgraded if necessary to include a peer channel listener.

- *Server*

  A bridges object are deleted, if present. A client/server channel listener and channel manager will be added, if not already present. No changes will be made to a peer channel listener.

- *Gateway*

  A bridges object, client/server channel listener and channel manager will be added, if not already present. No changes will be made to a peer channel listener.

The dialog presents a number of different tabs, each relating to a related set of queue manager characteristics. The tabs are shared with the *Queue manager properties* dialog, used to change an existing queue manager. The complete set of tabs is:

- *General*

  The name of the queue manager, description, class, path of the associated initialization file and the type details.

- *Summary*

  A high level view of the objects owned by the queue manager, e.g. queues, connections, listeners, etc.

- *Detail*

  A number of detailed properties of the queue manager, e.g. channel time-out, channel attribute rule classes etc.

- *Aliases*

  Alternative names that should be resolved to this queue manager.

[8] Full function is available when MQe_Explorer v1.26 or later created the queue manager.

© Copyright IBM Corp. 2000, 2001, 2002

- *Comms.*

  The information needed by another queue manager to connect to this queue manager (operating in its designated type), i.e. this tab describes incoming communications (as seen by the remote queue manager). It includes IP address and port number, channel type, protocol, options etc.

- *Registry*

  Details of the queue manager registry type, including the registry adapter.

- *Security*

  Security data associated with the registry.

- *Certificates*

  Details of queue manager WTLS mini-certificates stored in the registry.

- *Class aliases*

  Alternative names that can be used locally to reference Java classes.

- *Preloads*

  Java classes to be pre-loaded when the queue manager is started.

- *Bridges.*

  Information to be passed to the bridges object on creation, typically the load bridge rule.

- *Permissions*

  Permission statements to be actioned when the local queue manager is started.

The tabs that are available at any time depend upon many factors, including:

- Whether creating or modifying a queue manager
- Whether MQe_Explorer created the queue manager.
- Whether MQe_Explorer is running as a slave or a master.
- Whether the queue manager is local or remote.
- Whether on-line or off-line administration is selected.
- The type of the queue manager.
- The nature of the registry.

© Copyright IBM Corp. 2000, 2001

The most complete information is available for queue managers created and subsequently modified through MQe_Explorer. In those cases it should not be necessary to manually edit the associated *.ini* file.

The *General* tab has the appearance:

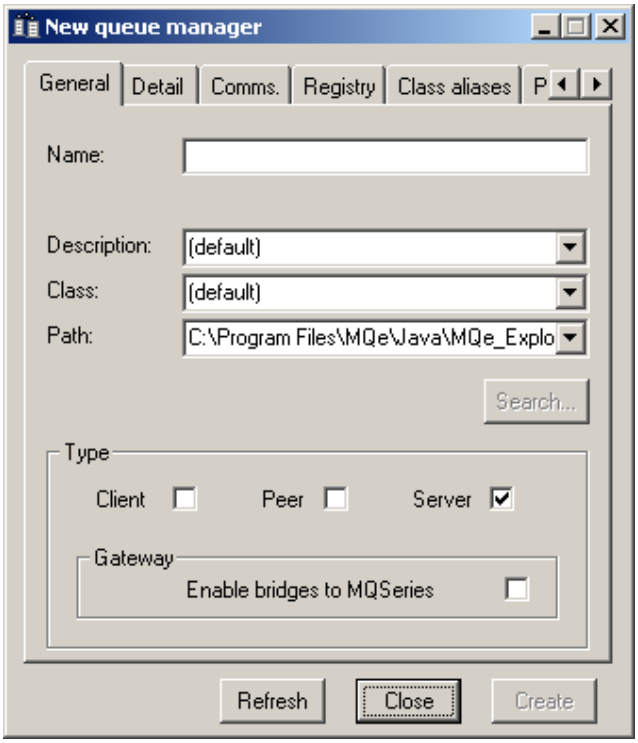


Figure 4-17: New queue manager dialog – General tab

The following input fields may be present:

- *General* tab:
  - *Name* – the name of the queue manager.
  - *Description* – any UNICODE string.
  - *Class (or alias)* – the class of the queue manager.
    - *(default)* is mapped to "com.ibm.mqe.MQeQueueManager".
  - *Path* – the location for the initialization file. The path can include the file name and type. If the type is omitted it will be set to *.ini*; if the file name is omitted a default will be used. The *Search* button can be used to locate a path or a file – after a queue manager name is present.
  - *Client* – checking this box will create a basic queue manager with only client capabilities (as defined above). As clients have restricted communications capabilities, do not choose this option unless a client is known to be required.
  - *Peer* – checking this box will create a queue manager with peer capabilities (as defined above). As peers have restricted communications capabilities, do not choose this option unless a peer is known to be required.

© Copyright IBM Corp. 2000, 2001, 2002

- *Server* – checking this box will create a queue manager with server capabilities (as defined above). This is the default type – another should be chosen only if the relevant capabilities are known to be required.
- *Enable bridges to MQSeries* – checking this box will create a gateway queue manager, i.e. a server queue manager that is also capable of bridging to MQSeries. This option is only available if the *Server* box has been checked.

The *Summary* tab is only displayed for an existing queue manager; it has the following appearance:

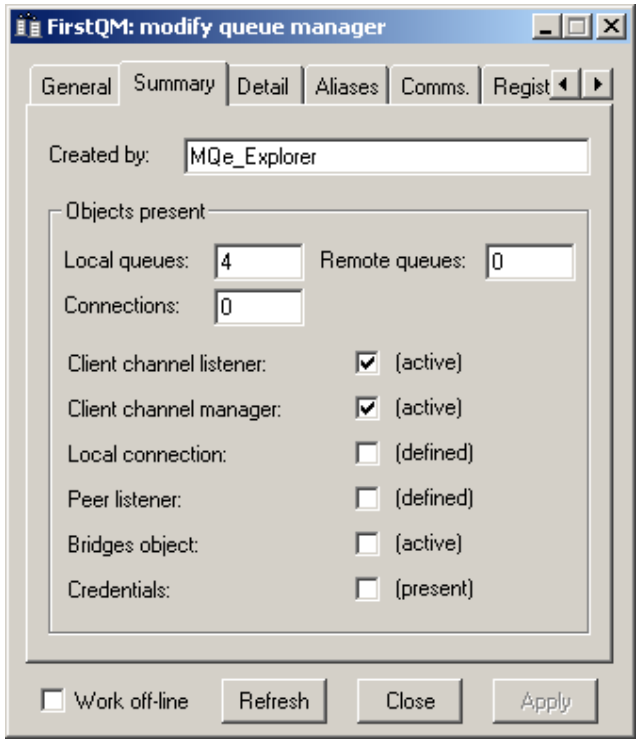


Figure 4-18: Modify queue manager dialog – Summary tab

© Copyright IBM Corp. 2000, 2001

- *Summary* tab:
    - *Created by* – the name of the application that originally created the queue manager.
    - *Local queues* – the number of local queue definitions present.
    - *Remote queues* – the number of remote queue definitions present.
    - *Connections* – the number of connection definitions present, including any local connection definition
    - *Client channel listener* – if checked, an active client/server channel listener is present.
    - *Client channel manager* – if checked, an active client/server channel manager is present.
    - *Local connection* – if checked, a local connection has been defined. This may be present to define a peer channel listener and/or queue manager aliases.
    - *Peer listener* – if checked, a peer channel listener has been defined.
    - *Bridges object* – if checked, an active bridges object is present. This object allows gateway capabilities.
    - *Credentials* – if checked, the registry contains queue manager mini-certificate credentials.

The *Detail* tab has the following appearance:

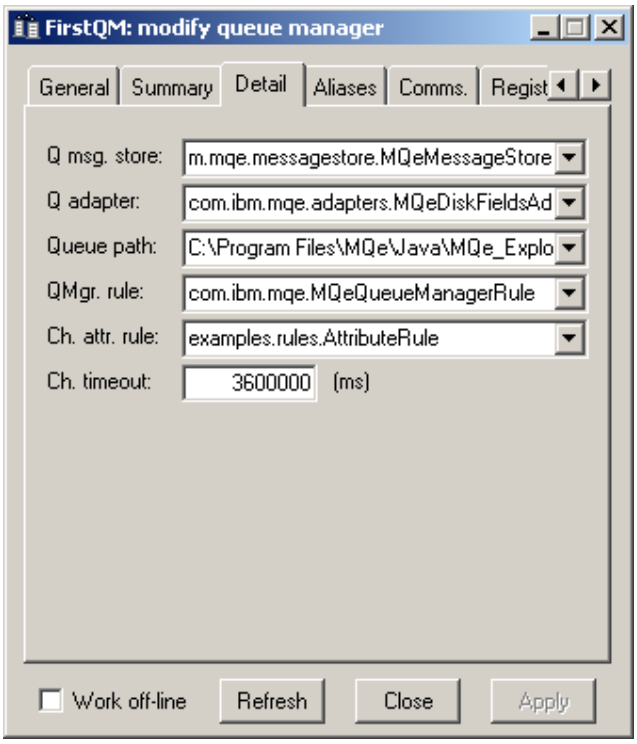


Figure 4-19: New queue manager dialog – Detail tab

© Copyright IBM Corp. 2000, 2001, 2002

- *Detail* tab:
    - *Queue message store (or alias)* – the default message store class that determines the way that the queue adapter stores messages.
        - *(default)* is mapped to "com.ibm.mqe.messagestore.MQeMessageStore".
    - *Queue adapter (or alias)* – the default queue adapter class.
        - *(default)* is mapped to "com.ibm.mqe.adapters.MQeDiskFieldsAdapter".
    - *Queue path* – the default path locating the physical storage of queues – passed to the queue adapter.
        - *(default)* is mapped to <*initialization file path*>\ <*qMgr name*>\Queues\.
    - *Queue manager rule class (or alias)* – the rule class to be used with the queue manager.
        - *(default)* is mapped to "com.ibm.mqe.MQeQueueManagerRule".
        - *(null)* is mapped to a null value (i.e. no rule).
    - *Channel attribute rule class (or alias)* – the rule class to be associated with the channel attribute.
        - *(default)* is mapped to "examples.rules.AttributeRule".

© Copyright IBM Corp. 2000, 2001

- *Channel timeout* – the time in milliseconds after which an idle channel will time out. This value must be zero or a positive integer; the default value is 3,600,000

The *Aliases* tab is only displayed for an existing queue manager; it has the following appearance:

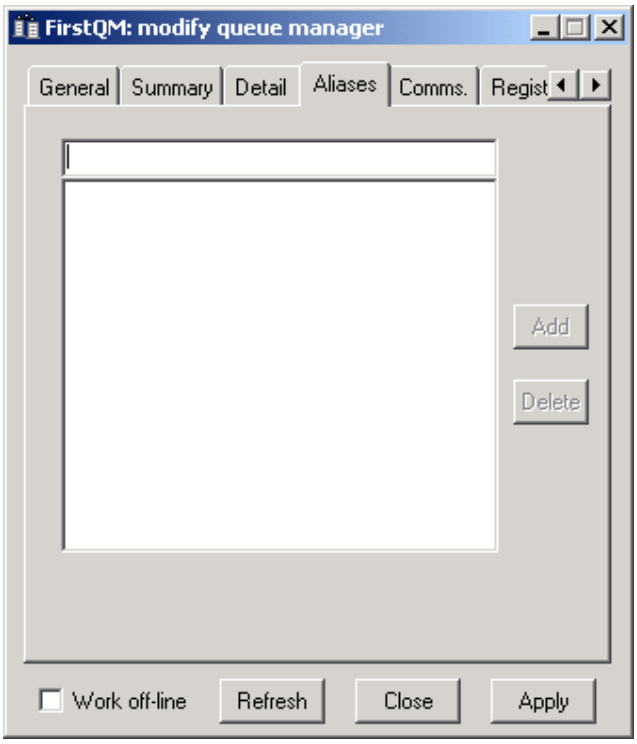


Figure 4-20: New queue manager dialog – Aliases tab

- *Aliases* tab:
  - *Alias names* – alias names for this queue manager. The *Add* button adds additional alias names entered in the input field; the *Delete* button deletes selected alias names.

© Copyright IBM Corp. 2000, 2001, 2002

The *Comms.* tab has the following appearance:

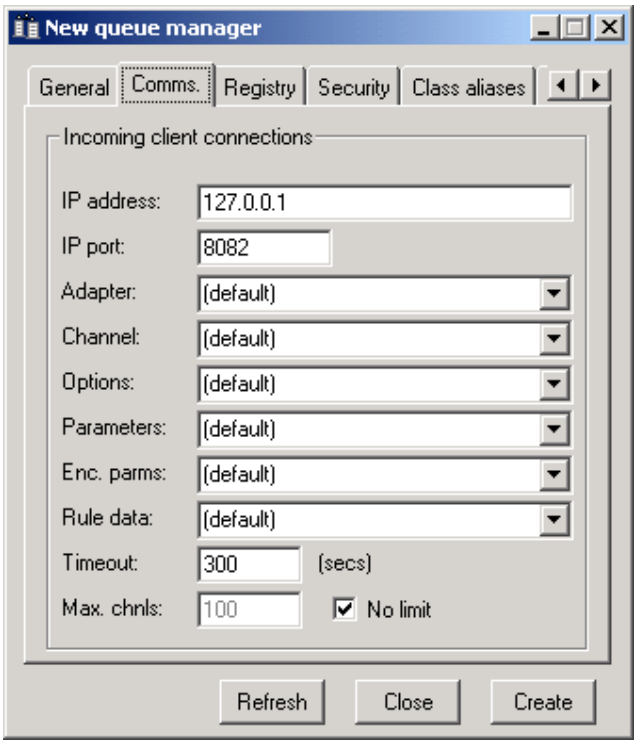


Figure 4-21: New queue manager dialog – Comms tab

- *Comms*. tab. For a peer queue manager, this information relates to an incoming peer channel connection; for a server or gateway queue manager, it relates to an incoming client connection:
    - *IP address* – the name or numeric IP address of the local machine hosting MQe_Explorer (a value is pre-loaded but must be checked because it may not always be appropriate). This value is only used when configuring remote queue managers to communicate with this queue manager.
    - *IP port number* – the port number used for incoming client channel or peer channel connection requests. A default value is pre-loaded but this may not always be appropriate (and is certainly incorrect if multiple queue managers are configured on the same machine).
    - *Communications adapter (or alias)* – the class of the communications protocol adapter for incoming connection requests.
        - *(default)* is mapped to "com.ibm.mqe.adapters.MQeTcpipHistoryAdapter".
    - *Channel class (or alias)* – the class that handles data transfers from a remote queue manager.
        - For a peer: *(default)* is mapped to "com.ibm.mqe.MQePeerChannel".
        - For a server or gateway: *(default)* is mapped to "com.ibm.mqe.MQeChannel".

© Copyright IBM Corp. 2000, 2001

- *Options* – a series of ASCII options to be passed to the communications protocol adapter. The format for specifying options is to enclose each option value within angled brackets.
    - *(default)* is mapped to "<PERSIST><HISTORY>" when the "com.ibm.adapters.MQeTcpipHistoryAdapter" adapter class or the alias "FastNetwork" is specified; otherwise *(default)* is mapped to "".
    - *(none)* is mapped to "".
- *Parameters* – an ASCII parameter string to be passed to the primary adapter, for example for an HTTP connection, the name of the servlet.
    - *(default)* and *(none*) are both mapped to "".
- *Encoded parameters* – a byte array to be passed to the primary adapter – the format used is two characters use to represent each byte, for example, "00" results in a byte with bits set 00000000;" FF" results in a byte with bits set 11111111. If the characters string has an odd length it is left padded with a "0". Encoded parameters are not yet supported.
- *Rule data* – an ASCII parameter string associated with the adapter that is passed to the connection manager as rules data – not yet supported.
- *Socket timeout* – the timeout in seconds to be used for incoming client/server connection requests.
- *Max channels* – the maximum number of client/server channels that are to be instantiated at any one time. Checking *No limit* removes any numeric restriction.

© Copyright IBM Corp. 2000, 2001, 2002

The *Registry* tab has the following appearance:

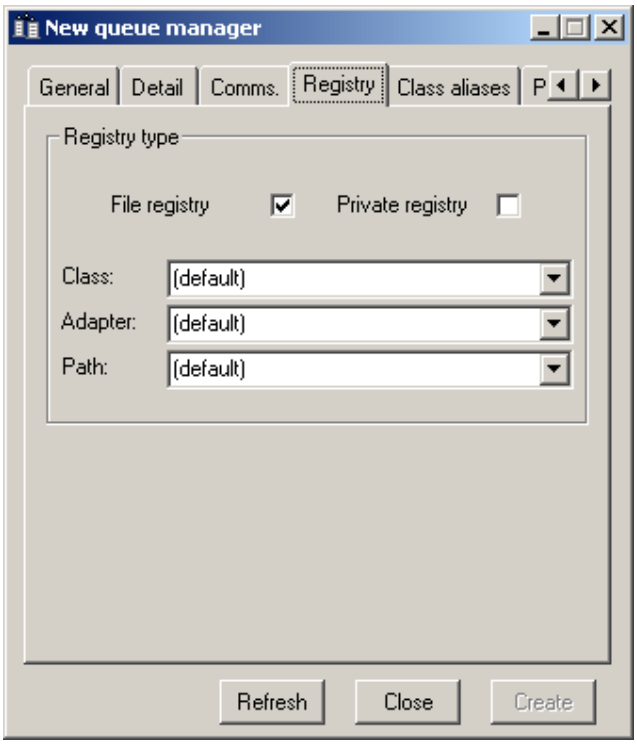


Figure 4-22: New queue manager dialog – Registry tab

- *Registry* tab:
  - *File registry* – checking *File registry* creates an unprotected registry (for more details see below).
  - *Private registry* – checking *Private registry* creates a protected registry – the details are set on the *Security* tab (for more details see below).
  - *Class (or alias)* – the registry class.
    - *(default)* is mapped to "com.ibm.mqe.registry.MQeFileSession" for file registries, and to "com.ibm.mqe.registry.MQePrivateSession" for private registries.
  - *Registry adapter (or alias)* – the class of the adapter used to map the registry into storage.
    - *(default)* is mapped to "com.ibm.mqe.adapters.MQeDiskFieldsAdapter".
  - *Path* – the registry location.
    - *(default)* is mapped to <*initialization file path*>\ <*qMgr name*>\Registry\.

© Copyright IBM Corp. 2000, 2001

A *file registry* is unprotected; a *private registry* has PIN-controlled access. Note that the use of a private registry is a prerequisite to using any form of mini-certificate based authentication based on the *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* class. The private registry by itself does not protect the queues (and their messages) in any way; queues and messages are protected through the use of either queue-based security (using cryptors and authenticators) or message-based security.

The *Security* tab has the following appearance:

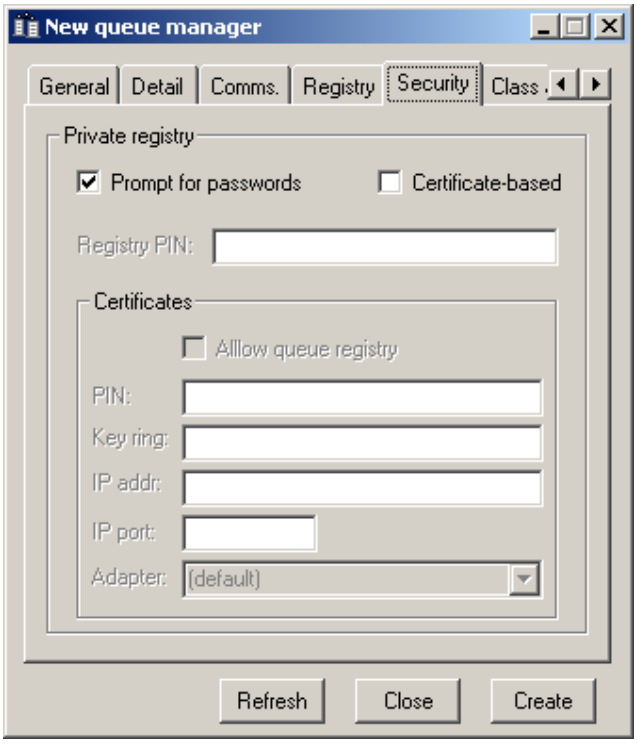


Figure 4-23: New queue manager dialog – Security tab

© Copyright IBM Corp. 2000, 2001, 2002

- *Security* tab:
    - *Prompt for passwords* – specifies that PINs and passwords should not be stored in the initialization file but requested from the user when required.
    - *Certificate-based*[9] – specifies that mini-certificate based authentication be enabled. This will cause auto-registration with the MQe mini-certificate server when the queue manager is created (or re-started). The mini-certificate server requires prior set-up such that it is allowed to issue a mini-certificate to the named queue manager. Certificates are stored in the private registry and have a pre-determined lifetime, set when issued. They can be renewed through the *Actions→Renew Credentials* menu item.
    - *Registry PIN* – the PIN for private registry access (for more details see below).
    - *Certificates panel: Allow queue registry* – if checked, allows queues to have their own credentials; if unchecked, queues must use the credentials of the queue manager (for more details see below).
    - *Certificates panel: PIN* – the certificate request PIN for the mini-certificate server (for more details see below).
    - *Certificates panel: Key ring* – the password used to protect the registry's private key (for more details see below).
    - *Certificates panel: IP address* – the IP address of the mini-certificate server.
    - *Certificates panel: IP port* – the IP port number of the mini-certificate server.
    - *Certificates panel: Adapter* – the communications adapter class (or alias) to be used to communicate with the mini-certificate server.
        - *(default)* is mapped to "com.ibm.mqe.adapters.MQeTcpipHttpAdapter".

  The *Security* tab is only enabled for queue managers with a private registry. The *registry PIN* controls access to the private registry; the *key ring password* controls access to the private key within the registry. Checking the *certificates-based* security box ensures that mini-certificates will be used for authentication and is a prerequisite to any use of authentication based on the *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* class. The *certificate PIN* is used in the request to the mini-certificate server when a certificate is required. If the *allow queue registry* box is unchecked (the default setting), queues are not allowed their own credentials, but must use those of the queue manager. If *prompt for passwords* is checked (the default setting) then PINs and passwords are never stored but always requested when required. The effect of these options is as follows:

  1. No use of certificates:

     If the prompt for passwords box is checked then the registry PIN must be provided every time the queue manager is restarted; otherwise the PIN will be stored (insecurely) in the *.ini* file. The *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* cannot be used for any form of authentication (but other authenticators are available).

[9] For more details of MQe security and mini-certificates, see the *MQSeries Everyplace Configuration Guide*.

© Copyright IBM Corp. 2000, 2001

2. Certificates in use:

    a. No prompt for passwords:

        i. Queue registry enabled: the three PINs/passwords are stored in the *.ini* file. The certificate PIN will be used for all auto-registration requests for certificates (queue manager and queue). If a different auto-registration certificate PIN is required for any reason, then the file must be edited and the queue manager restarted. If an authenticatable entity has its certificate renewed (*Action→Renew Credentials*) then the PIN associated with that renewal will be prompted for, and that PIN will replace the *.ini* file certificate PIN the *next time* the queue manager is re-started.

        ii. Queue registry not enabled: the three PINs/passwords are stored in the *.ini* file. The certificate PIN will be used for the auto-registration request for the queue manager certificate and will not be used again, though will remain in the *.ini* file. If the queue manager has its certificate renewed (*Action→Renew Credentials*) the PIN associated with that renewal will be prompted for, and that PIN will be placed in the *.ini* file (though will not be used again).

    b. Prompt for passwords:

        i. Queue registry enabled: the three PINs/passwords will be requested each time the queue manager is started. The certificate PIN input at start-up will be used for all auto-registration requests for certificates (queue manager and queue). If a different auto-registration certificate PIN is required for any reason, then the queue manager must be re-started. If an authenticatable entity has its certificate renewed (*Action→Renew Credentials*) then the PIN associated with that renewal will be prompted for, but this not affect the PIN used for auto-registration requests.

        ii. Queue registry not enabled: the first time the queue manager is started the three PINs/passwords will be requested. After a successful auto-registration of the queue manager, the certificate PIN will no longer be requested at start up. If the queue manager has its certificate renewed (*Action→Renew Credentials*) the PIN associated with that renewal will be prompted for.

© Copyright IBM Corp. 2000, 2001, 2002

The *Certificates* tab has the following appearance:

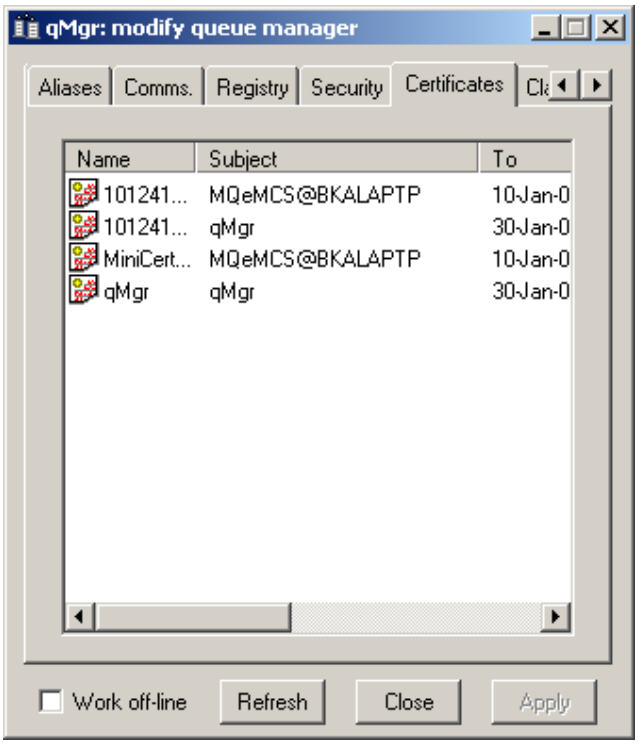


Figure 4-24: Modify queue manager dialog – Certificates tab

- *Certificates* tab:
  - *Name* – the name of the certificate[10]. Names that start with a number are old certificates that have been renamed.
  - *Subject* – the name of the authenticatable entity that owns the certificate.
  - *To* – the date after which the certificate is invalid.
  - *From* – the date from which the certificate is valid.
  - *Issuer* – the name of the entity issuing the certificate.

Only certificates for queue manager credentials are displayed; queue-based credentials are shown in the *queue properties* panel for the relevant queue.

[10] MQe_Explorer removes the suffix "_MiniCertificate".

© Copyright IBM Corp. 2000, 2001

The *Class aliases* tab has the following appearance:

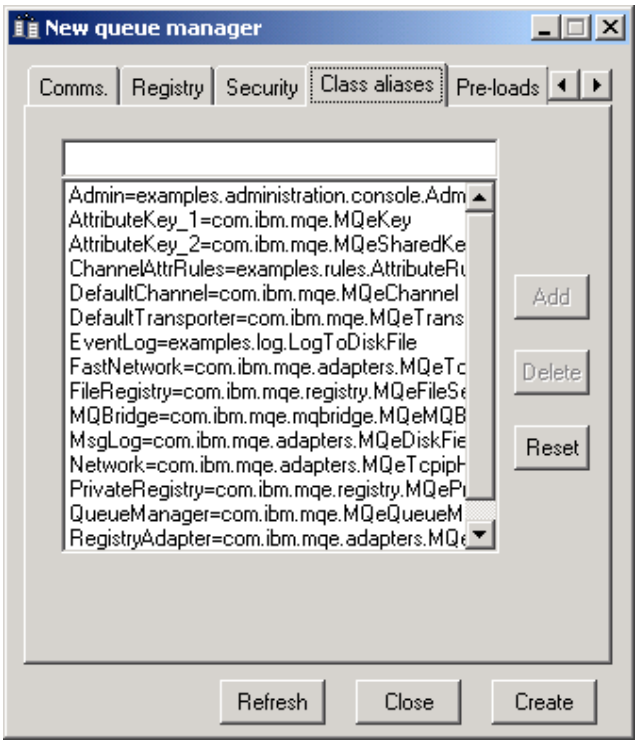


Figure 4-25: New queue manager dialog – Class aliases tab

- *Class aliases* tab:

  Alias definitions define alternative names that can be used locally to reference Java classes (usually stored in the *[Alias]* section of the initialization file). The *Add* button adds additional alias names entered in the input field; the *Delete* button deletes selected alias names; the *Reset* button resets the values to the defaults defined in the *Tools→Options→Classes* dialog.

  Alias definitions have the format:

  <Alias name>=<Class name>

  Caution should be exercised before changing the mapping of a class alias name. If an MQe property takes a class as its value then, if an alias name is used, MQe will store that alias name and not the immediate resolution of that name. If the class alias is later re-defined, that re-definition will affect all properties for which that alias name has been used. This may give rise to unexpected behavior, and may even prevent a queue manager or an MQe object from access or use. For this reason, wherever possible, MQe_Explorer itself does not use alias names unless specifically requested to do so.

© Copyright IBM Corp. 2000, 2001, 2002

The *Pre-loads* tab has the following appearance:

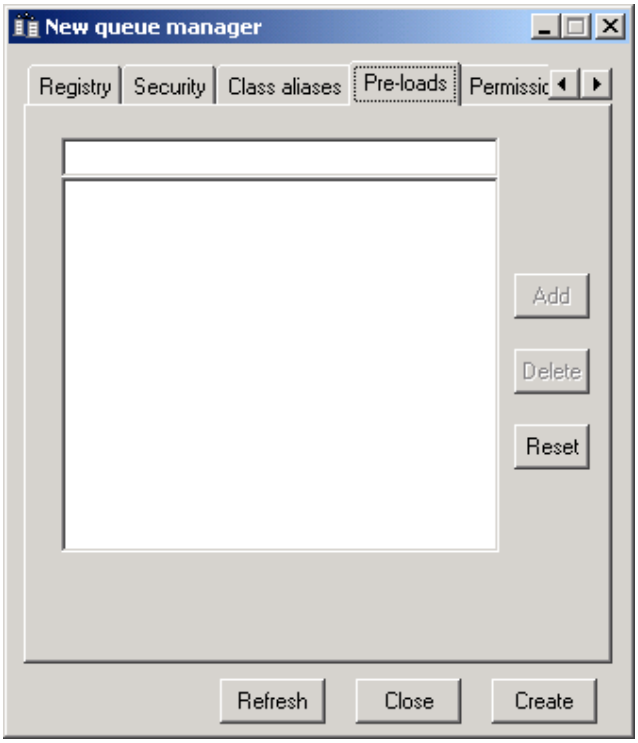


Figure 4-26: New queue manager dialog – Pre-loads tab

- *Pre-loads* tab:

  Classes (or their alias names) to be pre-loaded when the queue manager is loaded (usually stored in the *[PreLoad]* section of the initialization file). The *Add* button adds additional pre-load class names entered in the input field; the *Delete* button deletes selected names; the *Reset* button resets the values to the defaults defined by the *Tools→Options→Classes* dialog.

© Copyright IBM Corp. 2000, 2001

The *Bridges* tab has the following appearance:

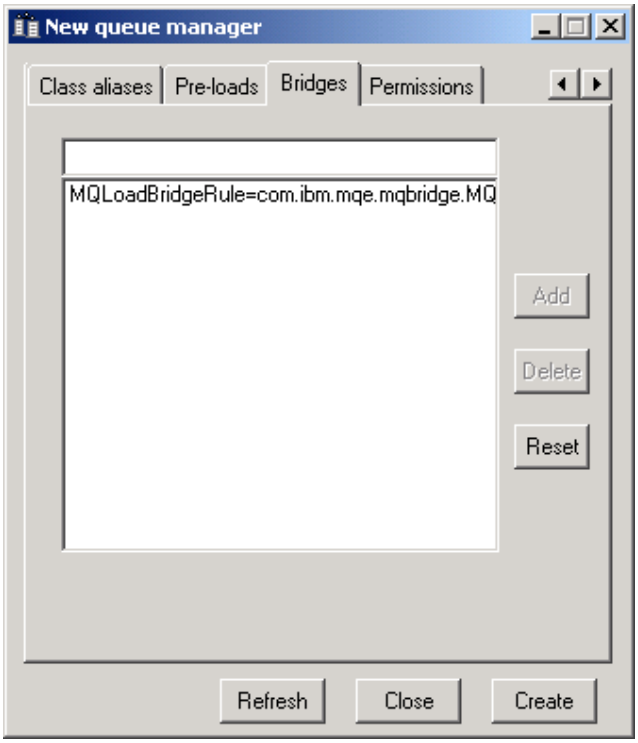


Figure 4-27: New queue manager dialog – Bridges tab

- *Bridges* tab:

  Definitions used to configure the Bridges object, i.e. the parent of any MQ bridges (usually stored in the *[MQBridge]* section of the initialization file). The *Add* button adds additional definitions entered in the input field; the *Delete* button deletes selected definitions; the *Reset* button resets the values to the defaults defined by the *Tools→Options→Classes* dialog.

  Bridges definitions have the format:

  <Bridges parameter>=<Class name>

© Copyright IBM Corp. 2000, 2001, 2002

The *Permissions* tab has the following appearance:

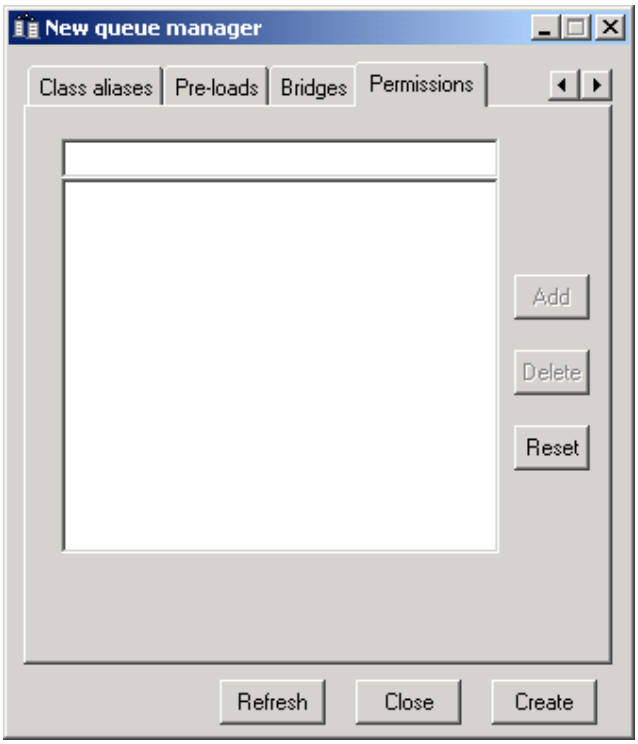


Figure 4-28: New queue manager dialog – Permissions tab

- *Permissions* tab:

  Statements defining allowed channel commands, file descriptor mappings and adapter calls (usually stored in the *[Permission]* section of the initialization file). The *Add* button adds additional statements entered in the input field; the *Delete* button deletes selected statements; the *Reset* button resets the values to the defaults defined by the *Tools→Options→Classes* dialog.

  Permission statements have the format:

  <Object name>=<Object value>

The *Create* button creates the new queue manager; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

Open

This command is used to load a queue manager and displays the *Open file* dialog to select the associated initialization file. If the file belongs to a configured queue manager then that queue manager is started. However, if the queue manager is un-configured, MQe_Explorer will configure it, given user confirmation. The menu item *Tools→Options→Initialization* controls the sections of the file processed during loading (for further details on loading and configuration see *Initialization* on page 116).

Files with extensions other than *.ini* may be opened – but are assumed to be in MQe initialization file format. *File→New* is not available after a queue manager has been loaded.

© Copyright IBM Corp. 2000, 2001

Close

The *Close* command closes the current queue manager and all associated windows. The command is not available if MQe_Explorer is running as a slave.

Properties

This command loads the properties pages for the selected object, i.e. one of:

- *Connection*.
- *MQ bridge*.
- *MQ client connection*.
- *MQ listener*.
- *MQ queue manager proxy*.
- *Queue*.
- *Queue manager*.

Property pages are used to view or modify object properties.

Connection properties

A typical *Connection property page* is:

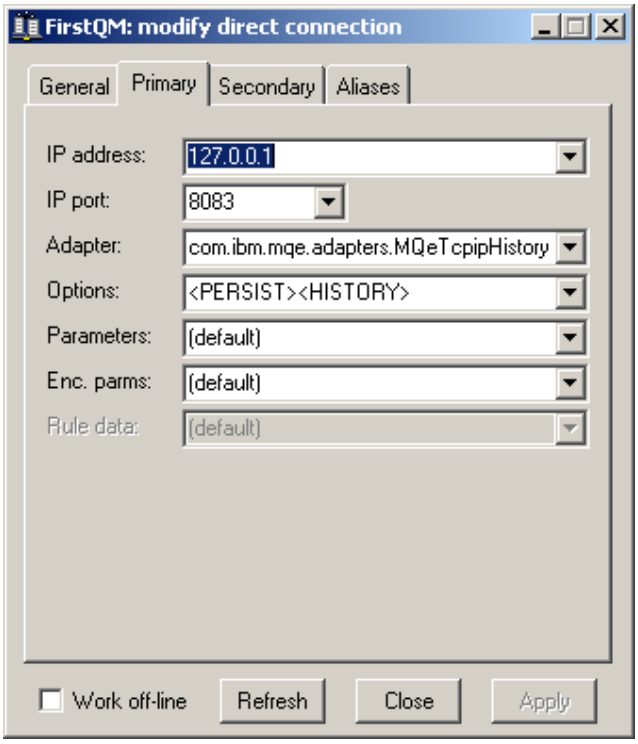


Figure 4-29: Typical connection property page

The properties are similar to those for creating a new connection definition in *File→New→Connection*. Some properties cannot be changed after the object has been created, e.g. connection name, local queue manager name.

© Copyright IBM Corp. 2000, 2001, 2002

The tabs and properties shown will vary according to the connection type and individual property values. If any adapter property is changed then all the related properties for that adapter are also reset (according to the values shown in the property page). It is not possible to change just one property and leave the others with their existing value. For example, if the adapter class, address or port is changed, then the adapter options are also changed.

The *Apply* button updates the existing properties, the local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window. *Work off-line* allows connection definitions to be modified on remote queue managers that are not currently available on the network.

MQ bridge properties

A typical *MQ bridge property page* is:

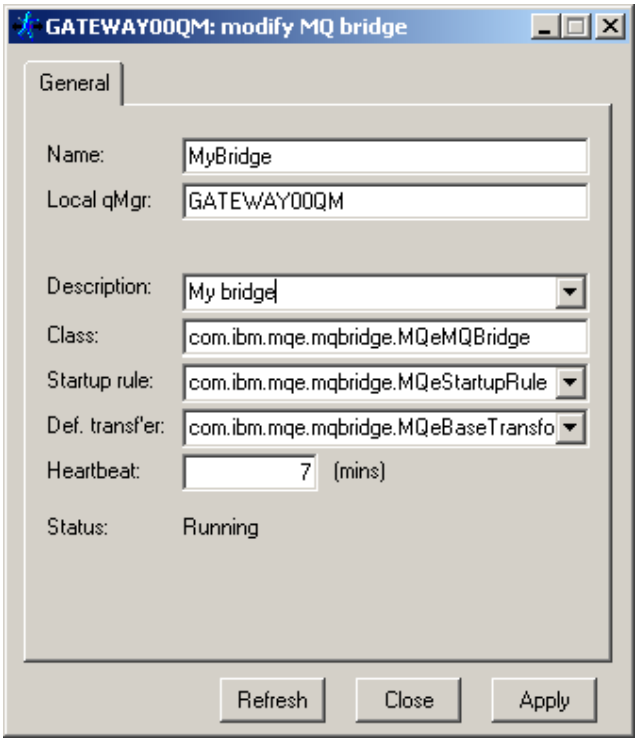


Figure 4-30: Typical MQ bridge property page

The properties are similar to those for creating a new MQ bridge definition in *File→New→MQ Bridge*. Some of these cannot be changed after the object has been created, e.g. bridge name, local queue manager name, class.

The *Apply* button updates the existing properties, local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

© Copyright IBM Corp. 2000, 2001

MQ client connection properties

A typical *MQ client connection property page* is:

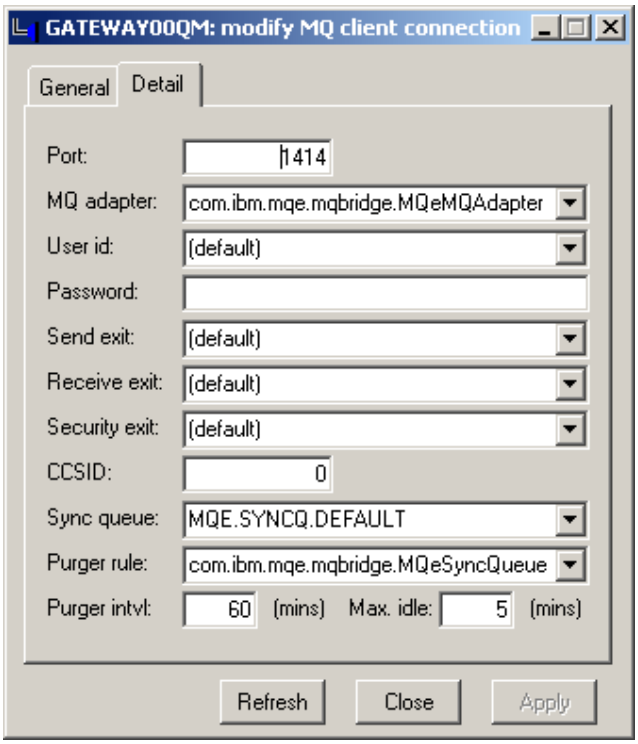


Figure 4-31: Typical MQ client connection property page

The properties are similar to those for creating a new MQ client connection definition in *File→New→MQ Client Conn*. Some properties cannot be changed after the object has been created, e.g. client connection name, proxy name, bridge name, local queue manager name, class.

The *Apply* button updates the existing properties, local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

© Copyright IBM Corp. 2000, 2001, 2002

MQ listener properties

A typical *MQ listener property page* is:

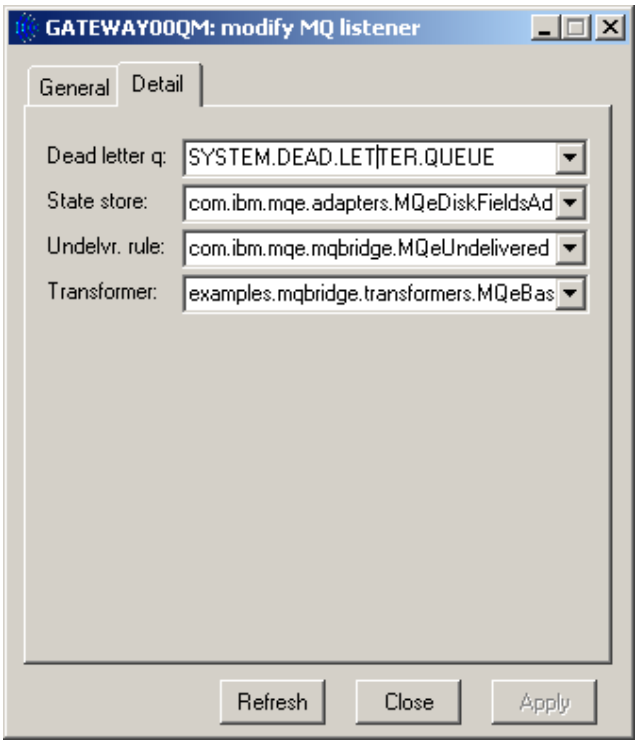


Figure 4-32: Typical MQ listener property page

The properties are similar to those for creating a new MQ listener definition in *File→New→MQ Listener*. Some of these cannot be changed after the object has been created, e.g. listener name, client connection name, proxy name, bridge name, local queue manager name, class.

The *Apply* button updates the existing properties, local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

© Copyright IBM Corp. 2000, 2001

MQ queue manager proxy properties

A typical *MQ queue manager proxy property page* is:

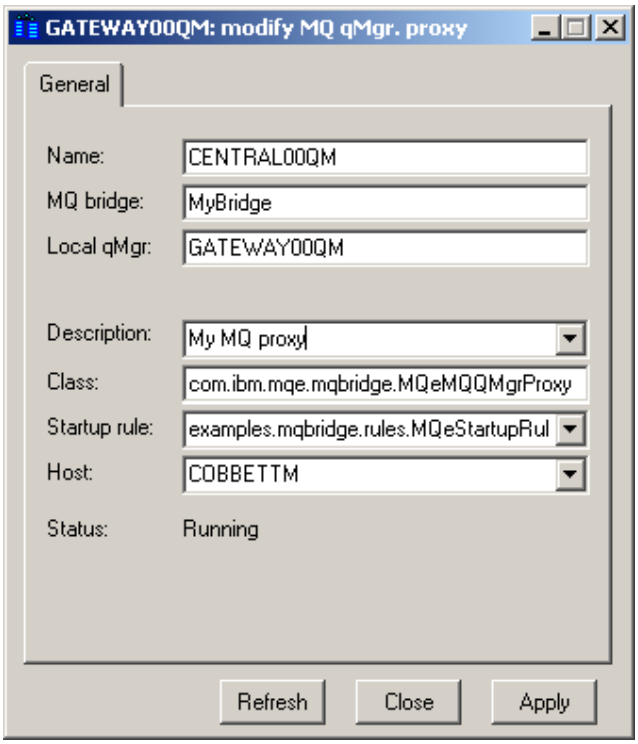


Figure 4-33: Typical MQ queue manager proxy property page

The properties are similar to those for creating a new MQ queue manager proxy definition in *File→New→MQ QMgr. proxy*. Some of these cannot be changed after the object has been created, e.g. proxy name, bridge name, local queue manager name, class.

The *Apply* button updates the existing properties, local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

© Copyright IBM Corp. 2000, 2001, 2002

Queue manager properties

A typical *Queue manager property page* is:

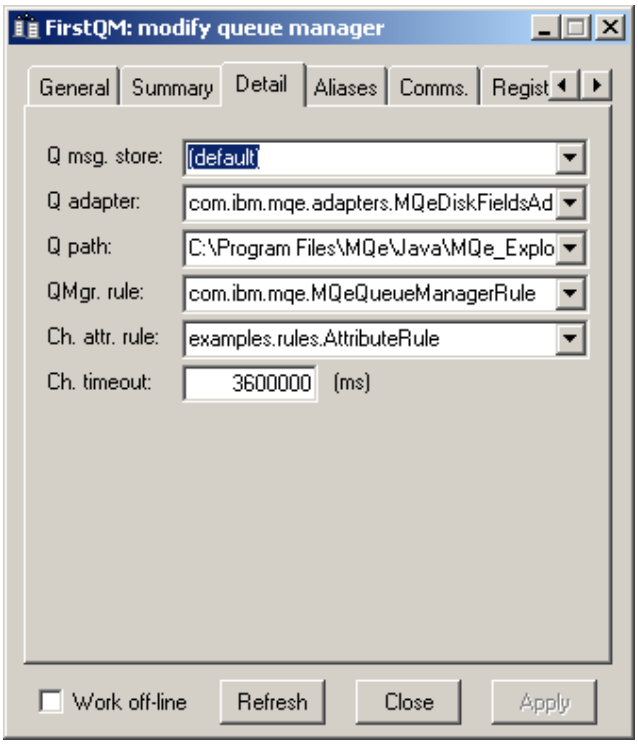


Figure 4-34: Typical queue manager property page

The properties are similar to those for creating a new queue manager definition in *File→New→Queue Manager*. Some of these cannot be changed after the object has been created, e.g. name, registry information, security, class.

The available tabs and properties vary, being dependent upon whether the queue manager is local or remote, whether MQe_Explorer originally created it, and upon many other factors.

The *Apply* button updates the existing properties, local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window. *Work off-line* allows queue manager definitions to be modified on remote queue managers that are not currently available on the network.

© Copyright IBM Corp. 2000, 2001

Queue properties

A typical *Queue property page* is:

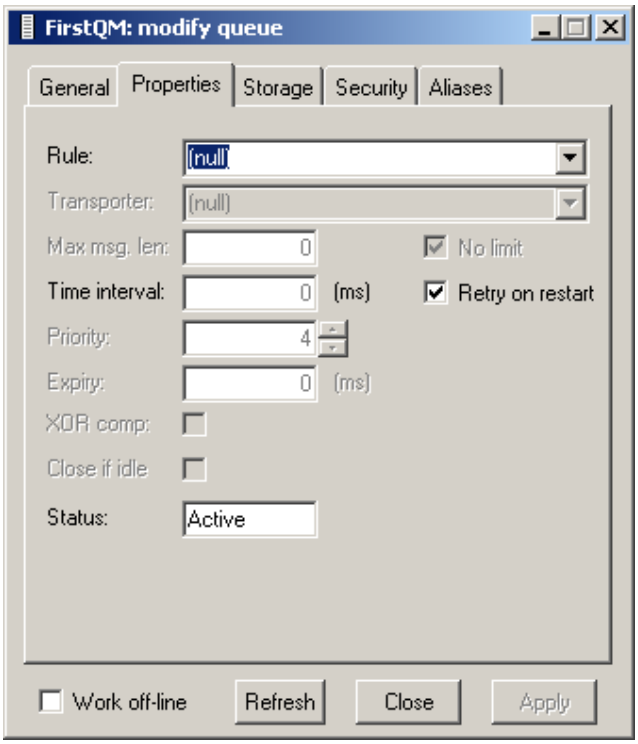


Figure 4-35: Typical queue property page

The properties are similar to those for creating a new queue definition in *File→New→Queue*. Some of these cannot be changed after the object has been created, e.g. queue name, local queue manager name, queue queue manager name, class, date created.

The tabs and properties shown will vary according to the queue type and individual property values.

The *Apply* button updates the existing properties, local cache and display; *Refresh* refreshes the available values in the dialog; *Close* removes the window. *Work off-line* allows queue definitions to be modified on remote queue managers that are not currently available on the network.

Exit

If MQe_Explorer is running as a master, this command closes the local queue manager and then terminates MQe_Explorer, before finally closing the JVM. If classes have been loaded from MQe_Explorer and set to run on non-daemon threads, MQe_Explorer will not terminate until those classes have completed their execution.

If MQe_Explorer is running as a slave, the queue manager is not closed, nor is the JVM.

© Copyright IBM Corp. 2000, 2001, 2002

4.2 Edit

Clear

Clear deletes the all the messages on the selected queue in the tree view or list view panes. This does not guarantee that the queue will then be empty, since messages may subsequently arrive on the queue. The local cache and display are refreshed after the operation.

Cut

Cut copies one (or more) messages selected in the list view pane to the clipboard; then deletes the selected messages (if still present on the source queue). Note that the copies are made from the cached messages held by MQe_Explorer – and at the time that they are cut it may be true (but immaterial) that the source queue no longer holds these messages.

The local cache and the display of messages are updated to reflect the cut operation – but not refreshed since this would change the message numbering in force.

Copy

Copy copies to the clipboard one or more messages selected in the list view pane. Note that the copies are made from the cached copies held by MQe_Explorer – and at the time that they are copied it may be true (but immaterial) that the source queue no longer holds these messages.

Delete

Delete deletes the selected object(s) in the tree view or list view panes, provided that they are of one of the following object types: *connection*, *MQ bridge*, *MQ client connection*, *MQ listener*, *MQ queue manager proxy*, *queue* or *message*. Excepting messages, only one object can be deleted at a time.

For all objects except messages, the local cache and display are refreshed after the delete operation. For messages, the local cache and display are updated but not refreshed, since refreshing would change the message numbering.

Paste

Paste copies the message(s) on the clipboard, resets their UIDs (to avoid duplicates), and puts them on the target queue. The clipboard contents are not cleared after a paste operation – thus messages may be re-pasted. The target queue cache and display are then refreshed (note that this unavoidably changes the message numbering in force).

Select all

Select All selects all the messages in the list view pane, provided that one or more messages are already selected.

4.3 View

Console

Console provides a means to view and manage applications running against the local queue manager. Typically such applications (or classes) will have been loaded through the use of the *Tools→Load* menu item. Further details of the console capabilities are given in the chapter *Console* on page 106.

© Copyright IBM Corp. 2000, 2001

Details

This view option affects the list view pane – all objects are displayed as a row of column values. The first column holds the name together with an associated icon. The remaining columns display object properties. The columns available and their default widths are set through the *Tools→Options* menu; to change these values use the tab appropriate to the object type.

Columns with no data can be set to zero width via the *View→Hide* Empty menu; additional rows per object alias name can be enabled through the *View→Use Multiple* menu.

Hide empty

This view option affects the list view pane and causes columns with no data to be displayed as columns of zero width. The effect is to reduce the need for scrolling in the list view pane.

List

This view option affects the list view pane – all objects are displayed as the object name together with an associated icon, with a single object per row. No object properties are available.

Off-line admin

This menu item displays the *Off-line administration window*, used for viewing and managing off-line administration requests and responses. See *Off-line administration* on page 100.

Refresh

Refresh refreshes the cached and displayed information for the selected object and its children. Use of *Refresh* is necessary to see the current state of an object where it may have been externally updated, e.g. to see recent messages on a queue, refresh the queue object.

In most cases where properties are changed through MQe_Explorer, an automatic refresh and display update takes place. Notable exceptions are message *Cut* and *Paste* operations, where MQe_Explorer does not automatically refresh in order to preserve the current message numbering scheme.

Small icons

This view option affects the list view pane – all objects are displayed as the object name together with an associated icon, with multiple objects per row. No object properties are available.

Trace

This menu item displays the *Trace window*, used for debugging MQe, MQe_Explorer or MQe applications. See *Trace* on page 109.

Translate admin

This view option affects the list view pane only and translates certain message field names:

- Admin parameter names in messages are translated into meaningful names, e.g. *admact* becomes ‘Action’.
- Reserved field names are translated into meaningful names, e.g. **Msg_Style* becomes ‘Style’.

© Copyright IBM Corp. 2000, 2001, 2002

The translated names are shown in single quotes.

Use multiple

This view option affects the display of objects with alias names.

Normally in the detailed list view mode (*View→Details*), a single row (the base row) is used to display an object. With the option, enabled an additional (alias) row is shown for each alias name. For both base and alias rows, the *Base* column shows the name of the base object and the *Aliases* column the list of alias names. For base rows the *Name* column (always the first) shows the base name along with an object icon; for alias rows it shows the alias name along with a shortcut object icon. For example, for a queue manager with an alias name, the original row will have the queue manager icon  and an alias row will be have the shortcut queue manager icon  .

In the small icons view mode (*View→Details)*, or the list view mode (*View→Details)*, additional entries are present for each alias name.

Normal object operations can be applied to alias rows or entries, for example, a right hand context menu can be displayed, a double click display the child objects, etc. In the details list view both base and alias rows participate in sort and re-order operations.

© Copyright IBM Corp. 2000, 2001

4.4 Action

Get new credentials

Queue managers created with a private registry and using certificate-based security, have a WTLS certificate in their registry that authenticates the identity of the queue manager, and another that authenticates the certificate supplier. These certificates were obtained through auto-registration with the MQe mini-certificate server and occurred at the first queue manager re-start after certificate-based security was enabled. If such queue managers also use *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* class authenticators on queues, and have the target registry property set to 'Queue', then they will have additional certificates present that authenticate queue identities.

Mini-certificates issued by the MQe mini-certificate server expire at a pre-determined time months after issue; normally they are renewed through the *Renew Credentials* action – this method retains the existing public and private keys. In cases where the server that issued the original certificates is not available, the credentials cannot be renewed, but must be replaced. The *Get New Credentials* action causes a request to be issued to the mini-certificate server to replace the existing mini-certificates, using new public and private keys. The old certificates are renamed[11] and remain in the private registry. The mini-certificate server must have been prior authorized to issued the new credentials; associated with the issue is a certificate-request PIN. This PIN must be supplied during the renewal process. For more details see the *Security* tab of the *Queue manager* on page 39, and also the *MQSeries Everyplace Configuration Guide*.

The *Get New Credentials* action applies to the selected object in the tree or list view pane. Only the queue manager hosting MQe_Explorer and its child queue objects may have their credentials updated in this way.

Ping

Ping is used to determine if the selected queue manager is alive. A message is sent synchronously[12] to the admin queue of the queue manager and a reply is requested. If the message cannot be sent, an error message will be displayed. When the response is received a message is then displayed, with the identity of the remote queue manager and with the overall elapsed time between the ping being requested and the response being received.

Note that ping uses the network connectivity defined for the remote queue manager on the hosting queue manager. If a ping request cannot be sent, or if a response is not subsequently received, this may indicate either a network problem between the hosting queue manager and the target queue manager, or alternatively the target queue manager may be unresponsive. If queue-level confirmation is required then a test message can be sent using *File→New→Message*.

---

[11] The rename adds a time prefix to the existing name, as determined by the algorithm: *Long.toString(new Date().getTime()) + "_"*

[12] This assumes that the remote admin queue is named according to the convention and that the local queue manager does not have a relevant asynchronous remote queue defined.

© Copyright IBM Corp. 2000, 2001, 2002

Renew credentials

Queue managers created with a private registry and using certificate-based security, have a WTLS certificate in their registry that authenticates the identity of the queue manager, and another that authenticates the certificate supplier. The certificates were obtained through auto-registration with the MQe mini-certificate server and occurred at the first queue manager re-start after certificate-based security was enabled. If such queue managers also use *com.ibm.mqe.attributes.MQeWTLSCertAuthenticator* class authenticators on queues, and have the target registry property set to 'Queue', then they will have additional certificates present that authenticate queue identities.

Mini-certificates issued by the MQe mini-certificate server expire after a pre-determined time after issue. The *Renew Credentials* menu item causes a request to be issued to the mini-certificate server to renew the mini-certificates, keeping the same public and private keys. The old certificates are renamed[13] and remain in the private registry. The mini-certificate server must have been prior authorized to issued the renewed credentials; associated with the renewal is a certificate-request PIN. This PIN must be supplied during the renewal process. For more details see the *Security* tab of the *Queue manager* on page 39, and also the *MQSeries Everyplace Configuration Guide*.

The *Renew Credentials* action applies to the selected object in the tree or list view pane. Only the queue manager hosting MQe_Explorer and its child queue objects may have their credentials renewed in this way.

Start

The *Start* menu item issues a start command to the selected object that affects both that object and any children. Only the bridges object (represented by the *Bridges folder*), MQ bridge, MQ queue manager proxy, MQ client connection and MQ listener objects may be started.

Stop

The *Stop* menu item issues a stop command to the selected object that affects both that object and any children. Only the bridges object (represented by the *Bridges folder*), MQ bridge, MQ queue manager proxy, MQ client connection and MQ listener objects may be started.

Trigger local

Trigger Local causes the immediate sending of any messages awaiting transmission on the local queue manager. Normally the current queue manager rule initiates message triggering; with the default rule in force, this happens on queue manager start-up or when a new message arrives on the queue. The *Trigger* command provides a convenient way of re-trying message transmission without sending additional messages or re-cycling the queue manager.

Note that *Trigger Local* only affects the queue manager used to host MQe_Explorer.

---

[13] The rename adds a time prefix to the existing name, as determined by the algorithm: *Long.toString(new Date().getTime()) + "_"*

© Copyright IBM Corp. 2000, 2001

Send File

Action→Send File enables the *Send file* dialog used to send copies of file to a target queue. The currently selected queue in the list or tree view panes is assumed to be the queue to receive the file, but this can be overridden. By default, it will use the message class *com.ibm.mqe.mqe_explorer.MQeFileTransferMsg*[14]; optionally it will use the *com.ibm.mqe.mqemqmessage.MQeMQMsgObject* class instead.

The *MQeFileTransferMsg* option supports the segmentation of files across multiple messages, integrity checking of the data upon receipt to detect corruption, and the transport of additional associated information, such as a description and the source location. These facilities are not available with *MQeMQMsgObject*; however such messages will flow across the MQe gateway to Websphere MQ queue managers without the need for custom transformation rules.

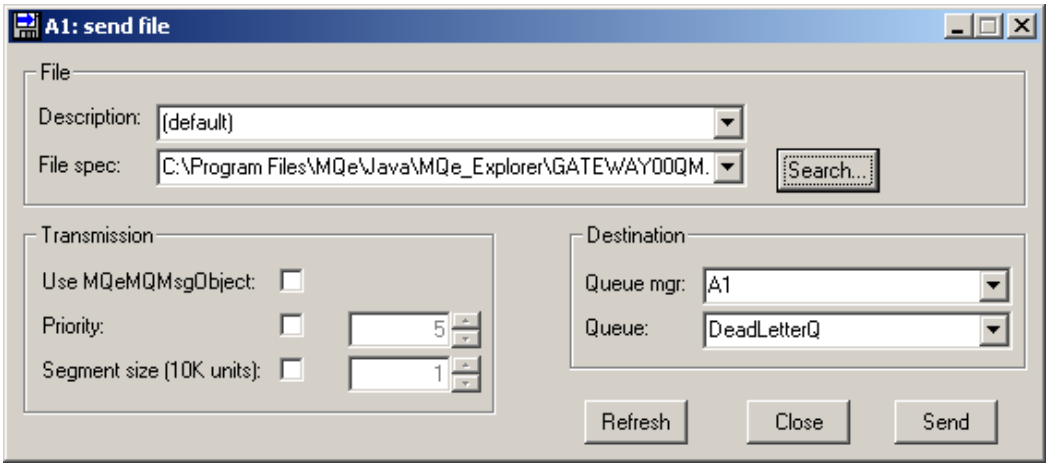


Figure 4-36: Send file dialog

The following input fields are present:

- *File* group:
  - *Description* – a description of the file.
    - *(default)* generates a description that automatically includes the name of the sending queue manager, the date and time.
  - *File specification* – an ASCII string specifying the location of the file.
    - The *Search* button brings up a file dialog and can be used to load this field.
- *Destination* group:
  - *Queue manager* – the name of the target queue manager. This may be entered or selected from the drop-down list. Any name must identify a queue manager to which addressability exists from the queue manager hosting MQe_Explorer, i.e. it must be one of the following: the name of the hosting queue manager (or an alias); a connection name (or an alias); a destination name owned by a *store and forward queue* (or an alias).

[14] Further details on this class and its use for file transfer are given in *Appendix C: MQeFileTransferMsg class* on page 202.

© Copyright IBM Corp. 2000, 2001, 2002

- *Queue* – the name of the target queue. This may be entered or selected from the drop-down list. Any name entered must either be known to the queue manager hosting MQe_Explorer (in conjunction with the target queue manager name) as a queue name (or alias), or otherwise will be assumed to identify a synchronous remote queue.

- *Transmission* group:
  - *Use MQeMQMsgObject* – the message object class used is *com.ibm.mqe.mqemqmessage.MQeMQMsgObject* (this class may be useful when sending to MQ queue managers as it is supported by the default bridge transformer).
  - *Priority* – if the box is unchecked the priority will default to 5; if checked the selected priority (0 – 9) is added to the message.
  - *Segment size* – if the box is unchecked the file will be transmitted in a single message; if checked the segment size (number of bytes per message) can be set in multiples of 10240 bytes.

The *Send* button sends a copy of the file; *Refresh* refreshes the available values in the dialog; *Close* removes the window.

Receive File

Action→Receive File enables the *Receive file* dialog used to receive files from messages (and typically sent through the *Action→Send File* dialog or an equivalent). A message containing the file (or a segment thereof) must have been previously selected in either the tree or list view panes.

The function available depends upon whether the file was sent via message(s) of the *com.ibm.mqe.mqe_explorer.MQeFileTransferMsg* or the *com.ibm.mqe.mqemqmessage.MQeMQMsgObject* class (for more details see *Send File* on page 70).

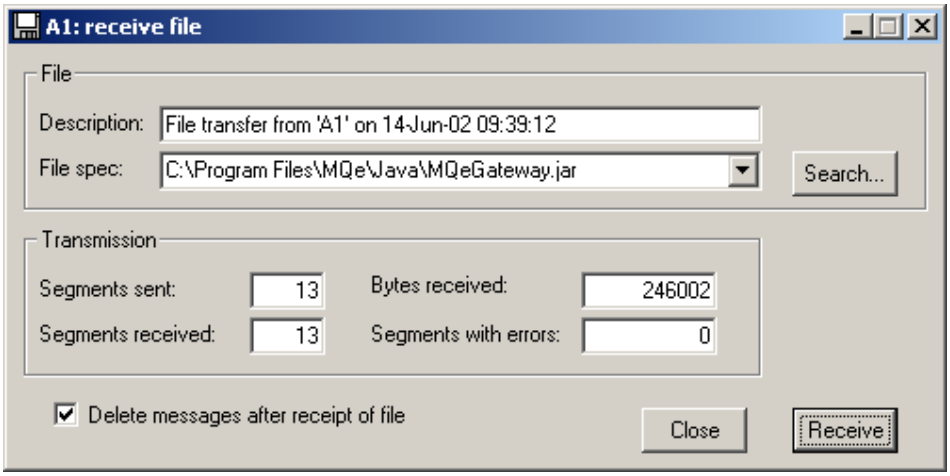


Figure 4-37: Receive file dialog

© Copyright IBM Corp. 2000, 2001

The following fields are present:

- *File* group:
  - *Description* – a description of the file.
  - *File specification* – an ASCII string initially loaded (where possible) with the source location of the file; this value must be edited into the target file location.
    - The *Search* button brings up a file dialog and can be used to load this field.
- *Transmission* group:
  - *Segments sent* – the number of segments (messages) sent.
  - *Segments received* – the number of segments (messages) received.
  - *Bytes received* – the total number of bytes received.
  - *Segments with error* – the number of segments (messages) detected as being corrupt.
- *Delete messages after receipt of file* – if checked, the messages are deleted after the file has been successfully retrieved.

The *Retrieve* button saves the file (and optionally deletes the messages); *Close* removes the window.

© Copyright IBM Corp. 2000, 2001, 2002

4.5 Tools

Configure remote

Configure Remote automatically configures a peer, server or gateway remote queue manager for on-line and/or off-line administration by MQe_Explorer; client queue managers must be manually configured. The queue manager to be configured should first be selected in either the tree view of the list view pane[15]. The following panel appears, with two tabs:

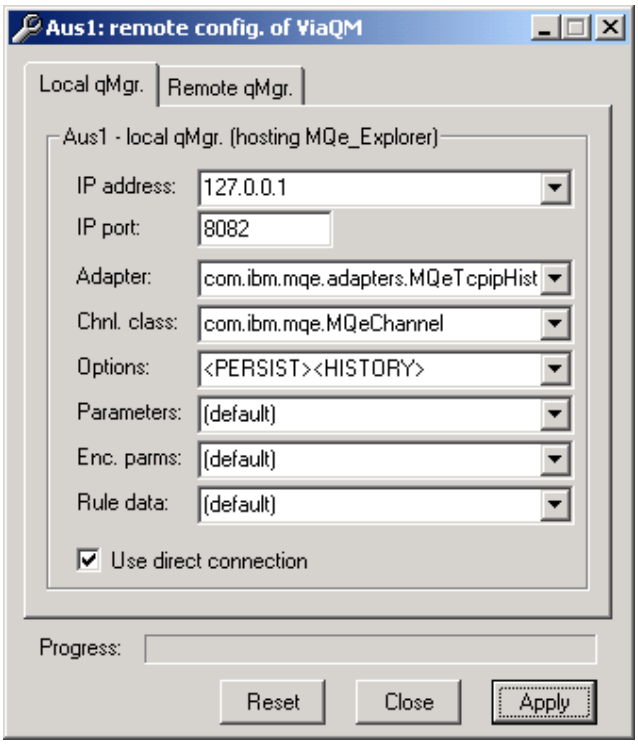


Figure 4-38: Remote configuration dialog – Local qMgr tab

[15] For a queue manager to so appear, a connection definition to it from the local queue manager must have been created. For more details see *Managing remote queue managers* on page 123.

© Copyright IBM Corp. 2000, 2001

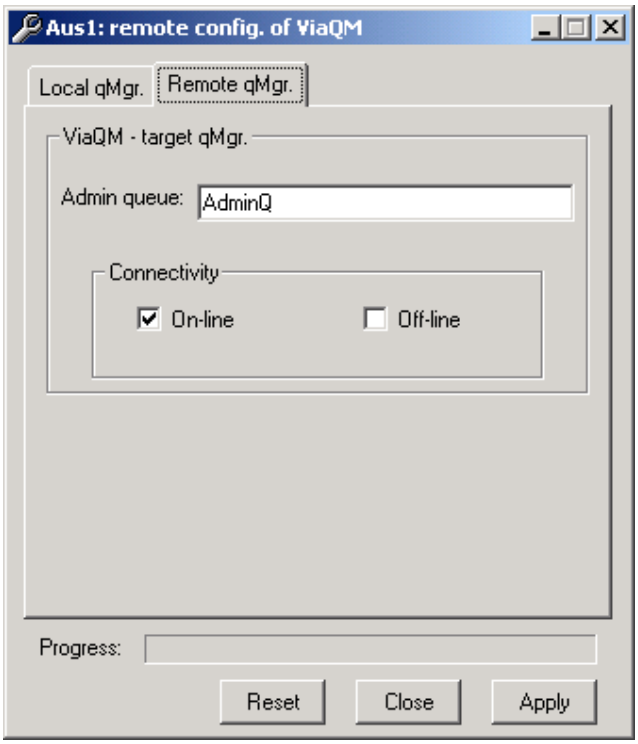


Figure 4-39: Remote configuration dialog – Remote qMgr tab

The local queue manager tab describes the local queue manager, i.e. the one that is currently hosting MQe_Explorer and consequently the one that will administer the remote queue manager. The information displayed should be carefully checked; it is used to create a connection definition on the remote queue manager, enabling it to communicate with the local queue manager.

- *Local queue manager* tab:
  - *IP address* – the numeric or string IP address of the machine hosting the remote queue manager.
  - *IP port* – the port number used by the remote queue manager to service incoming connection requests.
  - *Adapter* – the class of the communications protocol adapter.
    - *(default*) is mapped to "com.ibm.adapters.MQeTcpipHistoryAdapter".
  - *Channel class (or alias)* – the class that handles data transfers to the remote queue manager.
    - *(default*) is mapped to "com.ibm.mqe.MQeChannel".
  - *Options* – a series of ASCII options to be passed to the communications protocol adapter. The format for specifying options is to enclose each option value within angled brackets.
    - *(default)* is mapped to "<PERSIST><HISTORY>" when the "com.ibm.adapters.MQeTcpipHistoryAdapter" adapter class or the alias "FastNetwork" is specified; otherwise *(default)* is mapped to "".
    - *(none)* is mapped to "".

© Copyright IBM Corp. 2000, 2001, 2002

- *Parameters* – an ASCII parameter string to be passed to the primary adapter, for example for an HTTP connection, the name of the servlet.
  - *(default)* and *(none*) are both mapped to "".
- *Encoded parameters* – (*this property not yet supported by MQe)*.
- *Rule data* – (*this property not yet supported by MQe)*.
- *Use direct connection* – check this box if a direct connection is required (default); uncheck the box if a connection via another (intermediate) queue manager is to be used.
- *Via queue manager* – for an indirect connection, enter the name of the intermediate (via) queue manager.

The remote queue manager tab specifies the admin characteristics required of the remote queue manager.

- *Remote queue manager* tab:
  - *Admin queue* – the name of the queue used to process administrative messages.
  - *On-line mode* – indicates whether the remote on-line administration is required.
  - *Off-line mode* – indicates whether the remote off-line administration is required.

The *Apply* button configures the remote queue manager; *Reset* resets the available values in the dialog; *Close* removes the window. The *Progress* bar indicates the progress of remote configuration. Noticeable pauses in progress will be visible whilst MQe_Explorer waits for timeouts to expire. Reducing the timeouts through the *Tools→Options→Advanced-2* panel will hasten remote configuration, though care should be taken to allow for responses to be received over a high latency network.

For further details of remote configuration see *Managing remote queue managers* on page 123; also see *Automatic remote configuration* on page 125.

Demo mode

Demo Mode is provided as a way of automatically generating messages on a queue – such that MQe_Explorer can then be used to view them. It avoids the need to have application programs running when learning about MQe messaging or when demonstrating MQe (or MQe_Explorer).

Demo Mode, where possible, makes a copy of all messages sent or received by MQe_Explorer. The copies are placed on the *AdminReplyQ* of the hosting queue manager.

© Copyright IBM Corp. 2000, 2001

Copies are constructed such that:

- The message class of the copy matches that of the original.
- The message *UID* is reset.
- The original *UID* is copied into a new field called **Original_UID*.
- The value of the original **Msg_CorrelId* field is set to an array of zero bytes.
- The original **Msg_CorrelId* field value is copied into a new field called **Original_CID*.
- A new UNICODE **Copy_Info* field is added with the date/time when the copy was taken.
- All other fields are copied unchanged.

The changes to the *UID* are made because MQe requires that all *UIDs* be unique. The changes to **Msg_CorrelID* are to avoid the copy of the message being interpreted by MQe_Explorer as a valid admin message.

Load

The *Load* option enables Java classes to be loaded into the current JVM and started, thus providing for one or more MQe applications to be run concurrently with MQe_Explorer against the local queue manager. The window below is used to load the classes:

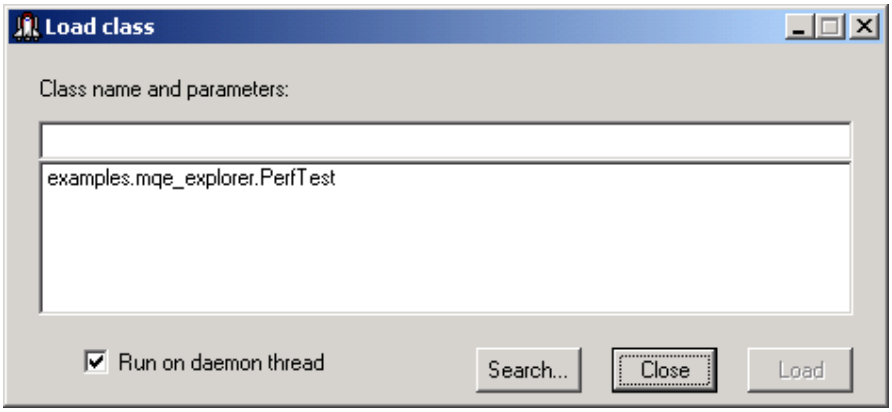


Figure 4-40: Load Java class dialog

The list box contains the default list of classes (changeable via *Tools→Options→Classes*). The name of a class to be loaded can be selected from the list box or entered on the input line (with or without the *.class* suffix); alternatively the file system can be searched using the *Search* button. Search will load a modified form of the full path of the selected file into the input area (the modifications change the "\" characters in the path to "." and remove the drive prefix). This modified string must then be edited (if necessary) into a proper Java class name that is consistent with the *classpath* system variable[16] – such that the Java class loader can locate it. Loaded classes (or load attempts) are added to the list box beneath the input area; a class can be re-loaded by selecting its name in the list box and clicking *Load*.

MQe_Explorer will start a new execution thread for the class and then invoke its static main method. It is not responsible for subsequently closing that application, which should occur before MQe_Explorer itself is closed. If you do close MQe_Explorer first, the local queue

[16] The current value of *classpath* is displayed in *System Information*, accessible from the *Help→About MQe_Explorer* menu item.

© Copyright IBM Corp. 2000, 2001, 2002

manager will be shut down whilst the loaded application is still running – which is likely to result in application errors. MQe_Explorer will not finally exit until all of the application's execution threads have been destroyed. It is not recommended that you exit from the Java Virtual Machine from within an application – this prevents a graceful shutdown of MQe.

By default, the application will be run on a daemon thread; if the *Run on daemon thread* check box is unchecked, then a user thread will be used. The status of running classes can be viewed in the MQe_Explorer console, accessed via the *View→Console* menu item. The *View→Trace* command may be used to trace application execution errors and exceptions.

The sample performance tool supplied with MQe_Explorer is an MQe application that can be loaded in this way; the class name to be used is: *examples.mqe_explorer.PerfTest*. This tool enables simple performance measurements; more details are provided in *Appendix A: Performance tool* on page 171.

Further information on writing applications that can be loaded by MQe_Explorer is given in *Programming MQe_Explorer* applications on page 168.

Options

The *Options* menu changes MQe_Explorer settings, with the results being stored in the associated configuration file. See *Configuration files* on page 115 for more information. The *Tools→Options* displays a tabbed control, each being for a related group of settings. Changes only take effect after the *Apply* button is clicked. The *Close* button removes the window and any pending changes not yet applied are lost. The *Reset* button resets all current and all saved options to their default values.

Advanced-1

The *New configuration – create queues* group, controls the queues that MQe_Explorer will create when a new queue manager is to be configured. See *Default configuration* on page 121.

The *Admin request queues used* group, controls the names of the queues to which MQe_Explorer will send admin messages. See *Queues* on page 123.

Advanced-2

The *Object access* group and the *Polling* group control the way in which MQe_Explorer asynchronously accesses remote objects. See *Object cache* on page 126.

Classes

This tab controls the contents of most of the list boxes presented by MQe_Explorer. These typically each hold a list of class names, but other lists are also supported, for example, lists of adapter options. By default, all of the classes supplied with MQe are enabled for selection; however additional classes/values can be added or existing ones removed. At least one value must be present for each class/value type. Coded values, i.e. those shown in brackets such as "*(default)*", can neither be added nor deleted.

Details of the class types and the available classes are given in *MQe classes* on page 186.

The following lists of classes/values are supported:

© Copyright IBM Corp. 2000, 2001

Admin queue classes – used in the *Class* property of queues (*General* tab).

Admin queue rule classes – used in the *Rule class* property of queues (*Properties* tab).

Alias name definitions – used in the configuration of queue managers (*Class aliases* tab).

Attribute rule classes – used in the *Attribute rule class* property of queues (*Security* tab) and the *Channel attribute rule class* property of queue managers (*Detail* tab).

Authenticator (with registry) classes – used to enable the *Target registry* property of queues (*Security* tab).

Authenticator classes – used in the *Authenticator class* property of queues (*Security* tab).

Bridge queue classes – used in the *Class* property of queues (*General* tab).

Bridge queue rule classes – used in the *Rule class* property of queues (*Properties* tab).

Bridges initialization – used in the configuration of queue managers (*Bridges* tab).

Channel classes – used in the *Channel class* property of connections (*General* tab), queue managers (*Comms*. tab) and the configuration of remote queue managers (*Local qMgr*. tab).

Comm. adapter classes – used in the *Adapter class* property of connections (*Primary* and *Secondary* tabs), queue managers (*Comms*. tab) and the configuration of remote queue managers (*Local qMgr.* tab).

Compressor classes – used in the *Compressor class* property of queues (*Security* tab).

Connection manager classes – used in the *Connection manager class* property of connections (*General* tab).

Connection manager rule classes – used in the *Rule class* property of connections (*General* tab).

Cryptor classes – used in the *Cryptor class* property of queues (*Security* tab).

Encoded parameters (adapters) – used in the *Encoded parameters* property of connections (*Primary* and *Secondary* tabs), queue manager (*Comms*. tab) and in the configuration of remote queue managers (*Local qMgr*. tab).

File registry classes – used in the *Class* property of queue managers (*Registry* tab).

Home server queue classes – used in the *Class* property of queues (*General* tab).

Listener state storage specifications – used in the *State store* property of MQ listeners (*Detail* tab).

Load classes – used in the *Load class* dialog (*Tools→Load*).

Local queue classes – used in the *Class* property of queues (*General* tab).

Local queue rule classes – used in the *Rule class* property of queues (*Properties* tab).

Message store classes – used in the *Message store* property of queues (*Storage* tab) and queue managers (*General* tab).

© Copyright IBM Corp. 2000, 2001, 2002

MQ adapter classes – used in the *MQ adapter* property of MQ client connections (*Detail* tab).

MQ bridge classes – used in the *Class* property of the MQ bridges (*General* tab).

MQ client connection classes – used in the *Class* property of the MQ bridges (*General* tab).

MQ listener classes – used in the *Class* property of the MQ listeners (*General* tab).

MQ proxy classes – used in the *Class* property of the MQ proxy queue managers (*General* tab).

Options (adapters) – used in the *Options* property of connections (*Primary and Secondary* tabs), queue manager (*Comms*. tab) and in the configuration of remote queue managers (*Local qMgr*. tab).

Parameters (adapters) – used in the *Parameters* property of connections (*Primary and Secondary* tabs), queue manager (*Comms*. tab) and in the configuration of remote queue managers (*Local qMgr*. tab).

Peer channel classes – used in connection definitions to identify a peer channel.

Permission statements – used in the configuration of queue managers (*Permissions* tab).

Private registry classes – used in the *Class* property of queue managers (*Registry* tab).

Pre-load classes – used in the configuration of queue managers (*Preloads* tab).

Queue adapter classes – used in the *Queue adapter class* property of queue managers (*General* tab) and the *Adapter class* property of Queues (*Storage* tab).

Queue manager classes – used in the *Class* property of queue managers (*General* tab).

Queue manager rule classes – used in the *Queue manager rule class* property of queue managers (*Detail* tab).

Remote queue classes – used in the *Class* property of queues (*General* tab).

Remote queue rule classes – used in the *Rule class* property of queues (*Properties* tab).

Rules data (adapters) – used in the *Rules data* property of connections (*Primary and Secondary* tabs), queue manager (*Comms*. tab) and in the configuration of remote queue managers (*Local qMgr*. tab).

Start-up rule classes – used in the *Start-up rule* property of MQ bridges (*General* tab), MQ queue manager proxies (*General* tab), MQ client connections (*General* tab) and MQ listeners (*General* tab).

Store and forward queue classes – used in the *Class* property of queues (*General* tab).

Store and forward queue rule classes – used in the *Rule class* property of queues (*Properties* tab).

Symmetric cryptor classes – used to enable the *Attribute rule class* property of queues (*Security* tab) and to determine the *Cryptor*

© Copyright IBM Corp. 2000, 2001

classes available (*Security* tab) when the queue manager does not have credentials.

Sync. queue purge rule classes – used in the *Purger rule class* property of MQ client connections (*Detail* tab).

Target registry identifications – used in the *Target registry* property of queues (*Security* tab).

Transformer classes – used in the *Transformer class* property of queues (*MQ bridge* tab) and MQ listeners (*Detail* tab), and the *Default transformer class* property of MQ bridges (*General* tab).

Transporter classes – used in the *Transporter class* property of queues (*Properties* tab).

Undelivered message rule classes – used in the *Undelivered message rule class* property of MQ listeners (*Detail* tab).

A sample panel is:

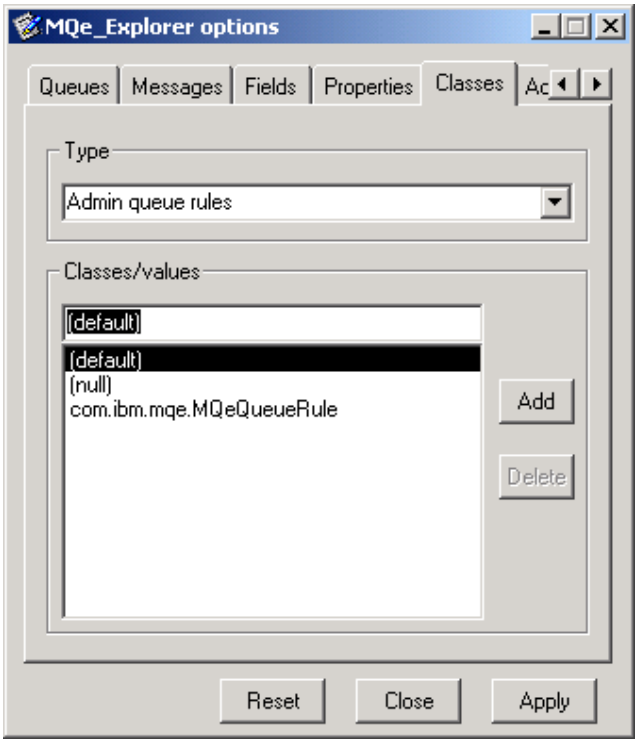


Figure 4-41: Options dialog – Classes tab

The *Add* button adds additional values entered in the input field; the *Delete* button deletes selected values.

© Copyright IBM Corp. 2000, 2001, 2002

Connections

This tab controls the detailed list view display of connection properties (*View→Details*), i.e. when the *Connections* folder node is selected in the tree view pane. The panel is:

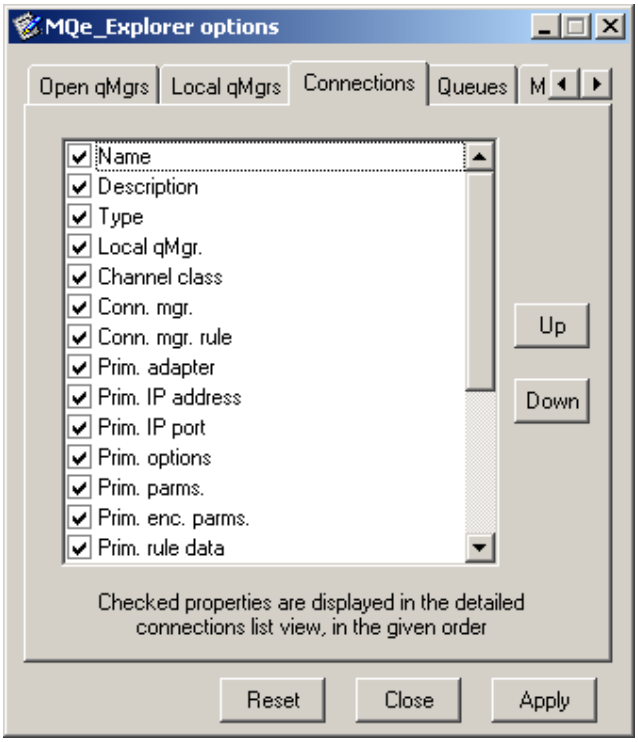


Figure 4-42: Options dialog – Connections tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001

Fields

This tab controls the detailed list view display of field properties (*View→Details*), i.e. when a message or a fields node is selected in the tree view pane. The panel is:

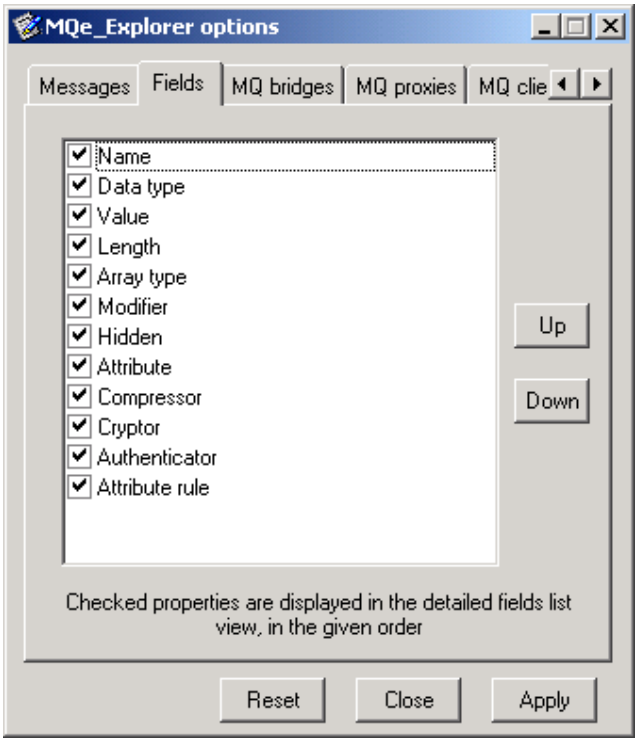


Figure 4-43: Options dialog – Fields tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001, 2002

Local qMgrs

This tab controls the detailed list view display of local queue manager properties (*View→Details*), i.e. when the *MQe root* node is selected. The panel is:

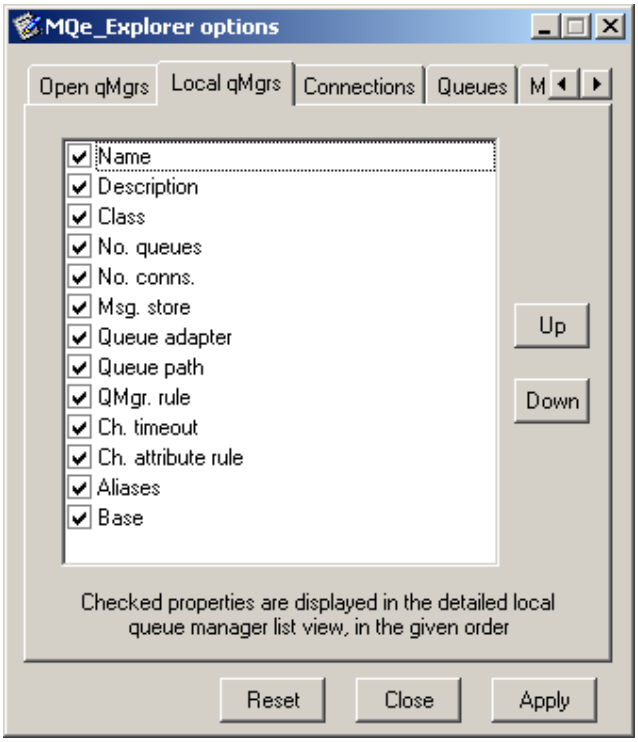


Figure 4-44: Options dialog – Local qMgrs tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001

Messages

This tab controls the detailed list view display of message properties (*View→Details*), i.e. when a local or remote queue node is selected in the tree view pane. The panel is:

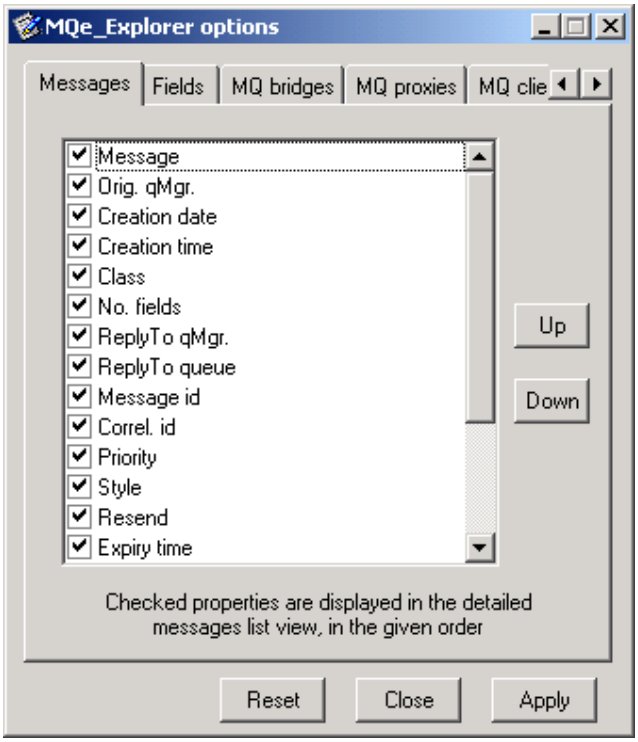


Figure 4-45: Options dialog – Messages tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001, 2002

MQ bridges

This tab controls the detailed list view display of MQ bridge properties (*View→Details*), i.e. when the Bridges object is selected in the tree view pane. The panel is:

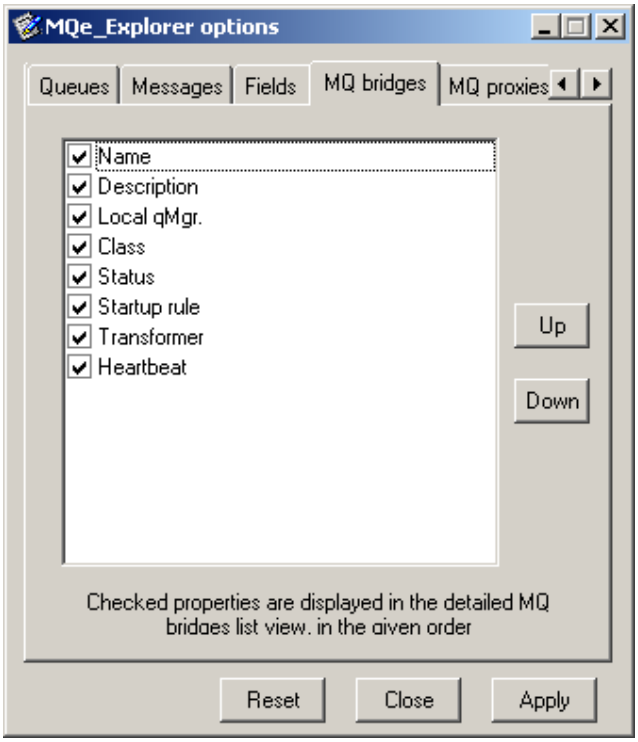


Figure 4-46: Options dialog – MQ bridges tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001

MQ client conns.

This tab controls the detailed list view display of field properties (*View→Details*), i.e. when an MQ proxy queue manager node is selected in the tree view pane. The panel is:

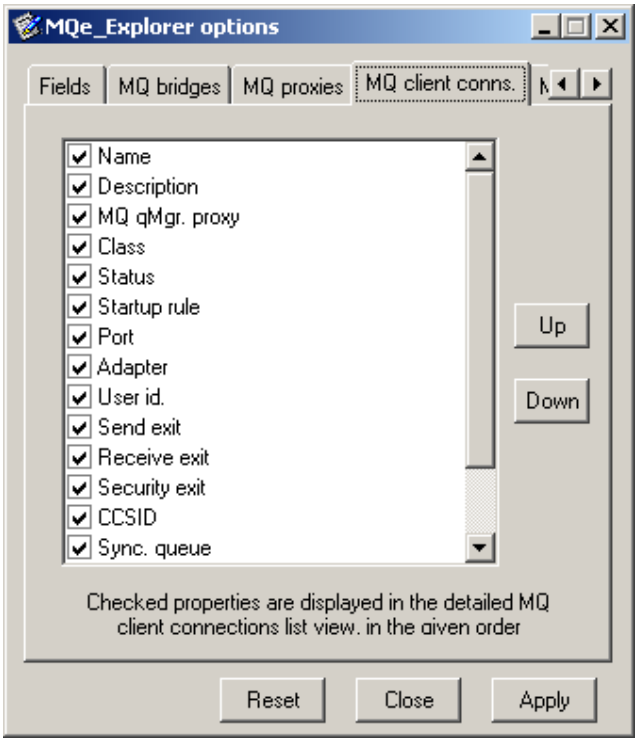


Figure 4-47: Options dialog – MQ client conns tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001, 2002

MQ listeners

This tab controls the detailed list view display of field properties (*View→Details*), i.e. when an MQ client connection node is selected in the tree view pane. The panel is:

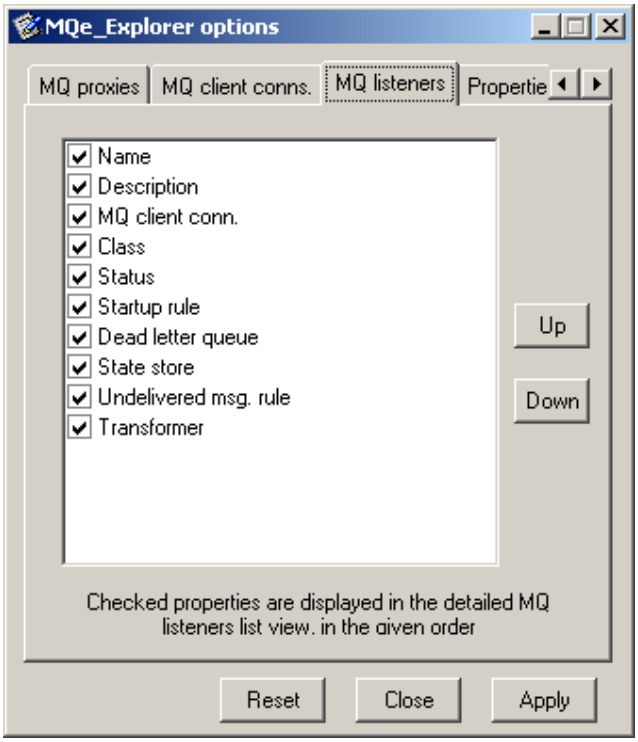


Figure 4-48: Options dialog – MQ listeners tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001

MQ proxies

This tab controls the detailed list view display of field properties (*View→Details*), i.e. when an MQ bridge node is selected in the tree view pane. The panel is:

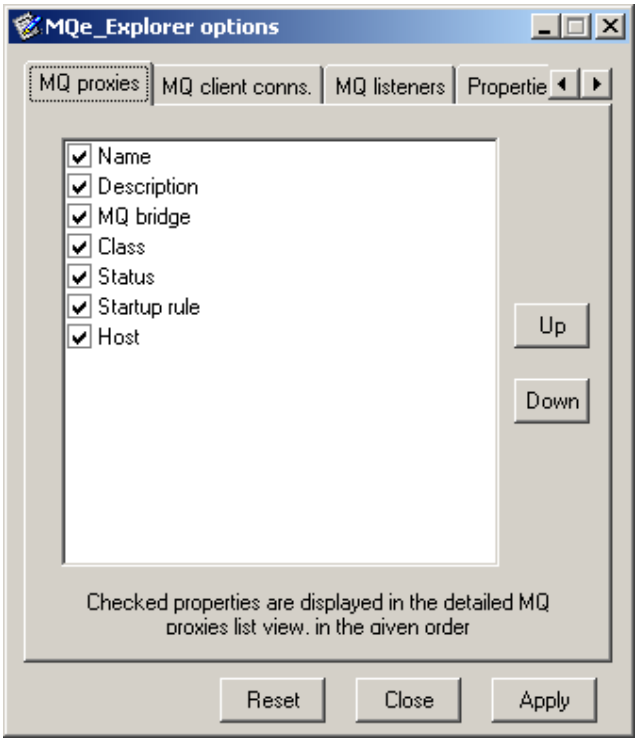


Figure 4-49: Options dialog – MQ proxies tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

Open qMgrs

This tab determines which sections of the MQe initialization file are processed when MQe_Explorer loads a queue manager. See *Initialization files* on page 116 for more information.

© Copyright IBM Corp. 2000, 2001, 2002

Queues

This tab controls the detailed list view display of local queue properties (*View→Details*), i.e. when the *Local queues* folder node is selected in the tree view pane. The panel is:

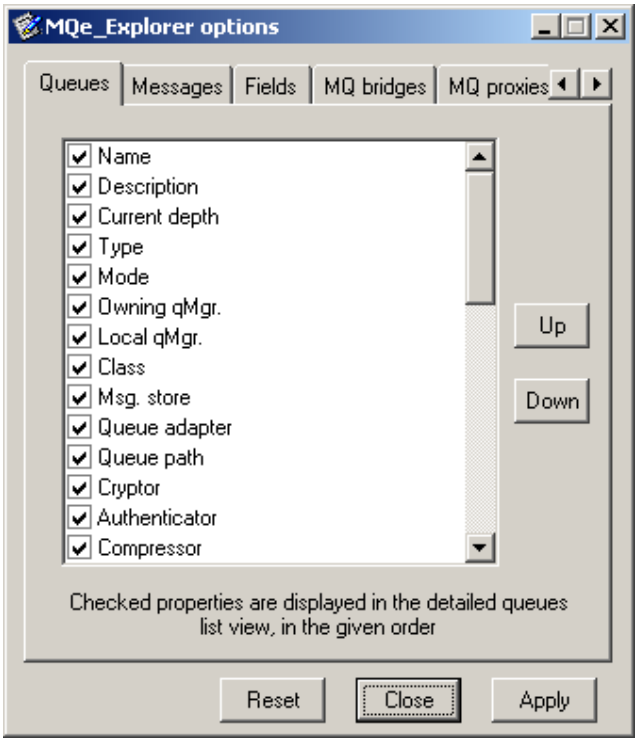


Figure 4-50: Options dialog – Queues tab

The order of the columns shown in the list view pane matches the order of the column names shown in the tab list. The first column cannot be re-ordered; others can be re-ordered by being selected and then moved up or down by clicking either the *Up* or *Down* buttons.

Columns can be deleted from the display by un-checking their associated check box – in this case their order in the list is irrelevant.

© Copyright IBM Corp. 2000, 2001

Properties

This tab controls the column width for a property when displayed in the detailed view format (*View→Details*) of the list view pane. To change a property either select it in the list box, or type the first few characters of its name in the input field (above the list box) and then select it. The selected property name will be displayed below the list box along width its current width (in characters), as illustrated below:

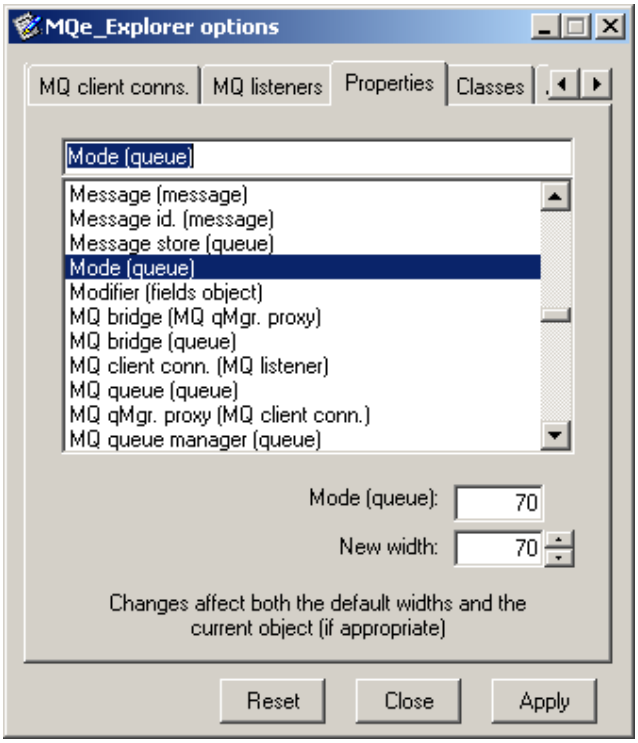


Figure 4-51: Options dialog – Properties tab

A new width can be set in the *New width* field. Changes only take effect when the *Apply* button is clicked. The default widths for an object property are controlled by this tab, as are the widths for a currently displayed object. Any other object that is not currently displayed and that has had a property column width explicitly re-sized (that is, by moving the separators between the columns) is not affected.

4.6 Window

Close all

Close All closes all open MQe_Explorer windows, excepting the main window, the console window (if present), the trace window (if present) and the options window (if present).

Minimize all

Minimize All minimizes all open MQe_Explorer windows, excepting the main window.

Restore all

Restore All restores all minimized MQe_Explorer windows.

© Copyright IBM Corp. 2000, 2001, 2002

4.7 Help

About MQe_Explorer

This menu item lists the program name, product identifier and version. The window is closed via the system close icon in the top right hand corner. The *System Info* button provides information on the local environment – IBM service staff may require this. The information available is as below:

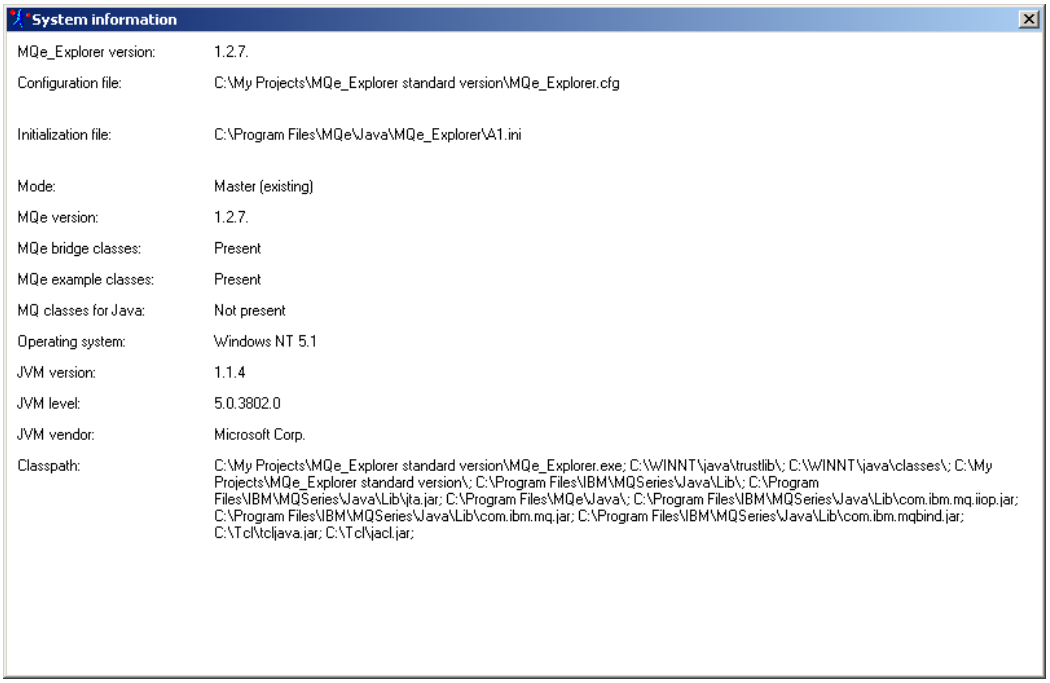


Figure 4-52: System information panel

In the formatting of the *classpath* variable value, MQe_Explorer adds a space after every semi-colon – this is to enable the string to word spill across multiple rows, if necessary. If any values (such as file paths) are not completely visible, because they are long strings without embedded blanks, then the value may be clicked and the cursor moved to the right – the value will then scroll in the field. All values can be selected and copied to the system clipboard.

Supervisor mode

MQe_Explorer can be put into supervisor mode by closing the *About MQe_Explorer* window with the control key held down. The mode is indicated by the *MQe+* environment value in the status bar. Supervisor mode should only be used when requested by IBM service staff.

© Copyright IBM Corp. 2000, 2001

5 The list view pane

The list view pane displays the children of the object currently selected in the tree view pane. If the detailed view mode has been set, then a tabular view of their characteristics is also shown.

The list view pane offers a number of functions:

- Empty columns are hidden by enabling the *View→Hide Empty* menu item (or equivalent toolbar button).
- Additional alias rows are shown by enabling the *View→Use Multiple* menu item (or equivalent toolbar button).
- Admin field names are translated to meaningful strings by enabling the *View→Translate Admin* menu item (or equivalent toolbar button).
- Rows are sorted (in increasing and decreasing column order) by clicking (or re-clicking) the column header.
- Column widths are changed by dragging the column boundaries (the settings are then remembered on an object-by-object basis – this memory does not survive a refresh, session change or an explicit reset for that object).
- Default column widths for properties are permanently set in *Tools→Options→Properties.*
- Columns are temporarily re-ordered by dragging the column titles.
- Columns are permanently re-ordered (or hidden) using *Tools→Options→…* (depending upon the type of property).

For most objects only one row may be selected at a time; however for message rows, multiple select is enabled.

Drag and drop operations (i.e. a move with either no key or with the *Shift* key held down; a copy with the *Control* key held down) are supported for message(s) from the list view pane to certain queue nodes in the tree view pane. The precise details of message move and copy parallel those for the equivalent operations undertaken via the clipboard (see the section on the *Edit* menu operation on page 65). The conditions for a tree view node to be acceptable as a message drop target are:

- It must be a queue of type:
  - Local.
  - Remote.
  - Bridge.
- It must not already own the messages.

The status bar indicates the progress and completion status of drag and drop operations.

© Copyright IBM Corp. 2000, 2001, 2002

5.1 Connections

Information on connections is displayed in the list view pane when the *Connections folder* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→Connections*; their default width is defined in *Tools→Options→Properties)*. In default order:

Name: the name of the connection (and therefore the name or an alias of the target remote queue manager). If the *View→Use Multiple* option is enabled then *Name* may also be an alias name for the connection (accompanied by a shortcut connection icon).

Description: a locally held description of the connection.

Type: the connection type (direct, indirect, etc).

Local qMgr: the name of the local queue manager owning the connection definition.

Channel class: the channel class (or alias) to be used in the connection.

Conn. mgr: (*this property not yet supported by MQe)*.

Conn. mgr. rule: (*this property not yet supported by MQe)*.

Prim. adapter: the class (or alias) of the primary communications protocol adapter.

Prim. IP address: the numeric or string IP address of the machine hosting the remote queue manager (associated with the primary adapter). If the connection is indirect then this field shows the name of the queue manager to be used.

Prim. IP port: the IP port number used by the remote queue manager to service incoming connection requests (associated with the primary adapter).

Prim. options: options to be passed to the primary adapter.

Prim. parms: ASCII parameters to be passed to the primary adapter.

Prim. enc. parms: binary options to be passed to the primary adapter.

Prim. rule data: the rule data to be used for the primary adapter.

Sec. adapter: (*this property not yet supported by MQe)*.

Sec. IP address: (*this property not yet supported by MQe)*.

Sec. IP port: (*this property not yet supported by MQe)*.

Sec. options: (*this property not yet supported by MQe)*.

Sec. parms: (*this property not yet supported by MQe)*.

Sec. enc. parms: (*this property not yet supported by MQe)*.

Sec. rule data: (*this property not yet supported by MQe)*.

Aliases: a list of alias names for the connection.

Base: the name of the connection.

© Copyright IBM Corp. 2000, 2001

5.2 Fields

Information on fields is displayed in the list view pane when either a message or a fields field is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→Fields*; their default width is defined in *Tools→Options→Properties*). In default order:

Name:	the name of the field.
Data type:	the data type of the field.
Value:	value of the field.
Length:	the length of the array or string.
Array type:	indicates whether an array is static or dynamic.
Modifier:	the field modifier.
Hidden:	indicates whether the field is hidden.
Attribute:	the attribute object associated with the fields object.
Compressor:	the compressor class (or alias) associated with the attribute object.
Cryptor:	the cryptor class (or alias) associated with the attribute object.
Authenticator:	the authenticator class (or alias) associated with the attribute object.
Attribute rule:	the rule class (or alias) associated with the attribute object.

5.3 Local queue managers

Information on local queue managers is displayed in the list view pane when the *MQe root node* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→Local queue managers*; their default width is defined in *Tools→Options→Properties)*. In default order:

Name:	the name of the local queue manager. If the *View→Use Multiple* option is enabled then *Name* may also be an alias name for the queue manager (accompanied by a shortcut queue manager icon).
Description:	a locally held description of the queue manager.
Class:	the class (or alias) of the queue manager.
No. queues:	the number of queues (both local and remote) owned by the queue manager.
No. conns:	the number of connections (possibly including a local connection) owned by the queue manager.
Message store:	the default message store class (or alias) for a queue.
Queue adapter:	the default adapter class (or alias) for a queue.
Queue path:	the default path locating the physical storage of queues – passed to the queue adapter.
Rule:	the rule class (or alias) to be used by the queue manager.
Ch. timeout:	the time in milliseconds after which the channel should time out.
Ch. attribute rule:	the rule class (or alias) to be associated with the channel attribute.
Aliases:	a list of alias names for the queue manager.
Base:	the name of the queue manager.

© Copyright IBM Corp. 2000, 2001, 2002

5.4 Messages

Information on messages is displayed in the list view pane when either a *local queue* or a *remote queue* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→Messages*; their default width is defined in *Tools→Options→Properties)*. These columns show not just the message object properties but also anticipate that certain common fields may be also present. The full set of fields for a particular message is available by selecting the message itself in the tree view pane. In default order:

Message:	a number assigned by MQe_Explorer to a message, reflecting its priority and position in the queue. Thus message number 1 is the first available message – true at the time that the cached snapshot was taken.
Orig. qMgr:	the queue manager that created the message.
Creation date:	the date the message was created.
Creation time:	the time the message was created.
Class:	the class of the message.
No. of fields:	the number of fields in the messages.
ReplyTo qMgr:	the name of the queue manager to which a reply should be sent.
ReplyTo queue:	the name of the queue to which a reply should be sent.
Message id:	the UID of the message.
Correl. Id:	the correlation id.
Priority:	the value of the priority field.
Style:	the style of the message (for example, *Reply*, *Request* etc).
Resend:	indicates whether the message is a resend of a previous message.
Expiry time:	the relative time in milliseconds when the message will expire, or the date when it will expire.
Lock id:	the lock id.
Attribute:	the attribute object associated with the message object.
Compressor:	the compressor class (or alias) associated with the attribute object.
Cryptor:	the cryptor class (or alias) associated with the attribute object.
Authenticator:	the authenticator class (or alias) associated with the attribute object.
Attribute rule:	the rule class (or alias) associated with the attribute object.

© Copyright IBM Corp. 2000, 2001

5.5 MQ bridges

Information on MQ bridges is displayed in the list view pane when the *Bridges folder* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→MQ bridges*; their default width is defined in *Tools→Options→Properties)*. In default order:

Name:	the name of the MQ bridge.
Description:	a locally held description of the MQ bridge.
Local qMgr:	the name of the queue manager owning the definition.
Class:	the object class (or alias).
Status:	object stopped or started.
Start-up rule:	the rule class (or alias) that determines whether the object and its children are started at object creation time or when the queue manager is opened.
Transformer:	the default transformer class (or alias) used for message conversion.
Heartbeat:	the name of the connection (that is, the *Name* property).

5.6 MQ client connections

Information on client connections is displayed in the list view pane when an *MQ queue manager proxy* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→MQ proxies*; their default width is defined in *Tools→Options→Properties)*. In default order:

Name:	the name of the MQ client connection.
Description:	a locally held description of the MQ client connection.
MQ qMgr. proxy:	the name of the MQ queue manager proxy owning the definition.
Class:	the object class (or alias).
Status:	object stopped or started.
Start-up rule:	the rule class (or alias) that determines whether the object and its children are started at object creation time or when the queue manager is opened.
Port:	the IP port number used by the target MQSeries queue manager.
Adapter:	the MQ adapter class (or alias) used to move messages from MQe to the target MQSeries queue manager.
User id:	the user id. used by MQSeries.
Send exit:	the send exit specified at the remote end of the MQSeries client channel.
Receive exit:	the receive exit specified at the remote end of the MQSeries client channel.
Security exit:	the security exit specified at the remote end of the MQSeries client channel.
CCSID:	the CCSID property used by MQSeries.
Sync. queue:	the name of the synchronization queue on the MQSeries queue manager used by the MQ bridge.

© Copyright IBM Corp. 2000, 2001, 2002

Purger rule: the rule class (or alias) that is used when a message on the sync. queue indicates a failure of MQe to confirm a message.

Max. idle: the time after which an idle connection to MQSeries is discarded and the resources returned to the pool.

Purger interval: the time between successive purges of the synchronization queue.

5.7 MQ listeners

Information on MQ listeners is displayed in the list view pane when an *MQ client connection* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→MQ client conns*; their default width is defined in *Tools→Options→Properties)*. In default order:

Name: the name of the MQ listener.

Description: a locally held description of the MQ listener.

MQ client connection: the name of the MQ client connection owning the definition.

Class: the object class (or alias).

Status: object stopped or started.

Start-up rule: the rule class (or alias) that determines whether the object and its children are started at object creation time or when the queue manager is opened.

Dead letter queue: the MQSeries queue used to hold messages that cannot be delivered from MQSeries to MQe.

State store: the specification of permanent storage used to hold state information as messages are moved from MQSeries to MQe.

Undelivered message rule: the class (or alias) determining the action to be taken when a message cannot be delivered from MQSeries to MQe.

Transformer: the class (or alias) used for message conversion from MQSeries to MQe.

5.8 MQ queue manager proxies

Information on queue manager proxies is displayed in the list view pane when an *MQ bridge* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→MQ bridges*; their default width is defined in *Tools→Options→Properties)*. In default order:

Name: the name of the MQ queue manager proxy.

Description: a locally held description of the MQ queue manager proxy.

MQ bridge: the name of the MQ bridge owning the definition.

Class: the object class (or alias).

Status: object stopped or started.

Start-up rule: the rule class (or alias) that determines whether the object and its children are started at object creation time or when the queue manager is opened.

Host: the IP address of the target MQSeries queue manager.

© Copyright IBM Corp. 2000, 2001

5.9 Queues

Information on local queues is displayed in the list view pane when the *Local queues folder* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available (the ones shown and their order is defined in *Tools→Options→Queues*; their default width is defined in *Tools→Options→Properties).* Note that the columns used are also common to those for remote queues; not all columns are relevant to all queue types. In default order:

Name: the name of the queue. If the *View→Use Multiple* option is enabled then *Name* may also be an alias name for the queue (accompanied by a shortcut queue icon).

Description: a locally held description of the queue.

Current depth: the number of messages on the queue.

Type: indicates the type of the queue, for example: *local, admin*, *remote* etc.

Mode: indicates whether the queue is accessed *synchronously* or *asynchronously*.

Owning qMgr: the name of the queue manager that owns the physical queue.

Local qMgr: the name of the queue manager that owns the definition of the queue.

Class: for a local queue, the queue class (or alias) implementing the queue; for a remote queue, the class (or alias) implementing the remote definition.

Message store: the message store class (or alias).

Queue adapter: the queue adapter class (or alias).

Queue path: the path locating the physical storage of queues – passed to the queue adapter.

Cryptor: the cryptor class (or alias) associated with the queue.

Authenticator: the authenticator class (or alias) associated with the queue.

Compressor: the compressor class (or alias) associated with the queue.

Attribute rule: the attribute rule class (or alias) associated with the queue.

Max. msg. length: the maximum length of messages on the queue.

Max. queue depth: the maximum number of messages on the queue.

Priority: the default priority to be associated with a message.

Status: indicates whether the queue is *active* or *inactive*.

Creation date: the date/time the queue (or local definition) was created.

Rule: the rule class (or alias) to be used by the queue.

Target registry: the registry to be used by the authenticator.

Expiry: the time in milliseconds after which messages expire.

Time interval: the time in milliseconds between attempting to retrieve messages (home server queue) or re-attempting the processing of messages (admin queue).

Transporter: the transporter class (or alias) to be used (this class handles the getting and putting of messages to/from remote queues).

© Copyright IBM Corp. 2000, 2001, 2002

Destinations:	the queue managers destinations for which a *store* (*and forward*) queue will hold messages.
Target qMgr:	this applies to *store and forward queues* and is the name of the next queue manager that will receive the messages.
XOR compress:	indicates whether the transporter should use XOR compression.
Close idle:	indicates whether the transporter should be closed once all the messages have been transmitted.
MQ bridge:	the MQ bridge object that handles the target MQ queue.
Transformer:	the transformer class (or alias) converting the message from MQe to MQ format.
Max. idle time:	the maximum time (in seconds) that the MQ bridge queue can hang on to an idle connection before it is returned to the connection pool.
Location:	indicates whether the queue is *local* or *remote*.
Aliases:	a list of alias names for the queue.
Base:	the name of the queue.

5.10 Queue manager folders

The queue manager folders are displayed when a *queue manager* is selected in the tree view pane. In the detailed view (*View→Details*) the following columns are available; in default order:

Name:	the folder name.	
Number:	the number of objects associated with the folder*:*	
	Bridges:	the number of MQ bridges present. Note that the Bridges folder represents the MQe bridges object and therefore is only displayed when the queue manager is of the gateway type.
	Connections:	the number of connections (including a local connection).
	Local queues:	the number of local queues.
Status:	*stopped* or *running* (only applicable to the bridges object).	

5.11 Remote queues

Information on remote queues is displayed in the list view pane when a *Connection* is selected in the tree view pane. In the detailed view (*View→Details*) the columns available are as for the *Local queues* list view pane (see page 94). The properties shown and their order is defined in *Tools→Options→Queues*; default widths are defined in *Tools→Options→Properties)*. In default order:

© Copyright IBM Corp. 2000, 2001

6 Off-line administration

MQe_Explorer supports the off-line administration of MQe objects[17] by asynchronous sending admin messages to queue managers that are not currently on-line. In order to use off-line administration both the local queue manager hosting MQe_Explorer and the remote queue manager must have been appropriately configured (see *Managing remote queue managers* on page 123). With the notable exception of object deletion, off-line administration uses property pages – just as for on-line administration. The most obvious difference however is that that MQe_Explorer can no longer pre-load these pages with the current property values.

Off-line working is activated in various ways. To change the properties of an unavailable queue manager (identified by the icon), right click on the queue manager icon (or use the menus) and select the *Properties* menu item. To manipulate other objects belonging to the unavailable queue manager, expand the tree to the first folder level (*Bridges*, *Connections*, *Local queues*) in the tree or list view. Then, right clicking on the *Connections* folder (or via the menus) enables connection properties to be created, deleted or changed. Similarly, the *Local queues* folder enables both local and remote queues to be created deleted or changed. Note that these menu items are only enabled for unavailable queue managers. Off-line administration is not supported for bridges and related objects.

An alternative technique is to bring up the appropriate properties page and check the *Off-line working* box in the bottom left.

Where possible, the properties are pre-loaded with the *(no change)* option[18], as seen below:

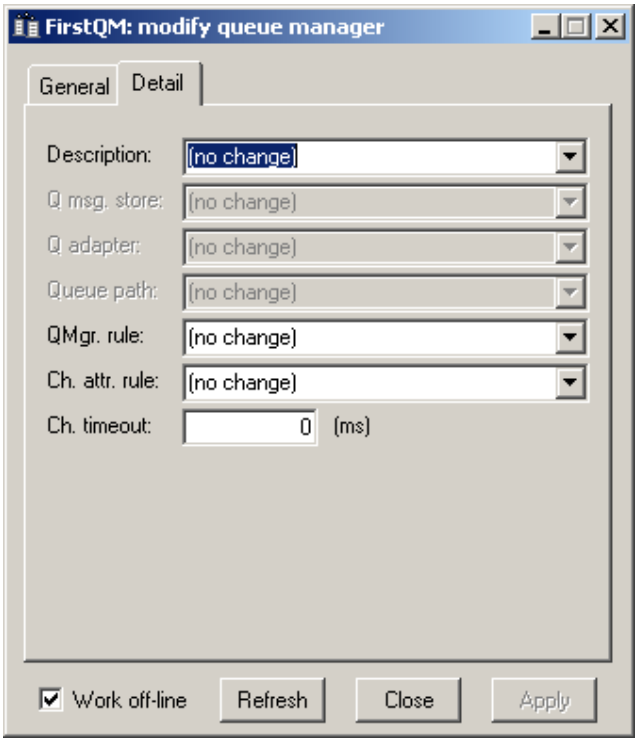


Figure 6-1: Off-line administration – modify queue manager

[17] Off-line administration of MQe bridge-related objects is not supported.
[18] Numeric values cannot be pre-loaded in this way. However if the operator does not change the displayed value, the resulting admin message will not set that parameter.

© Copyright IBM Corp. 2000, 2001, 2002

In some cases collections of parameters are modified in their entirety. For example, when changing a connection, the IP address alone cannot be changed (such that other adapter-related parameters are left with their existing values). Unless otherwise specified, all adapter parameters will be set to the defaults shown in the property pages.

Through off-line working it is not possible to change aliases, nor to change the destinations associated with a store and forward queue.

When changes are applied the messages are sent asynchronously to the remote admin queue using the queue name set up in the *Tools→Options→Advanced-1→Admin request queue used* group. For the configuration aspects of this see *Queues* on page 123.

A message box appears assigning an admin request number to the request – this number increases non-sequentially and wraps. It is not settable by the operator but it is significant in tracking the progress of an off-line admin request.

Figure 6-2: Off-line administration – admin request assignment

The off-line administration window shows the progress of off-line admin requests. This window is accessed through the *View→Off-line* admin menu item. A row appears for each request, and for each response[19]; the request number correlates the rows. The particular message types shown are determined by the checked *View* menu items (or their associated toolbar buttons).

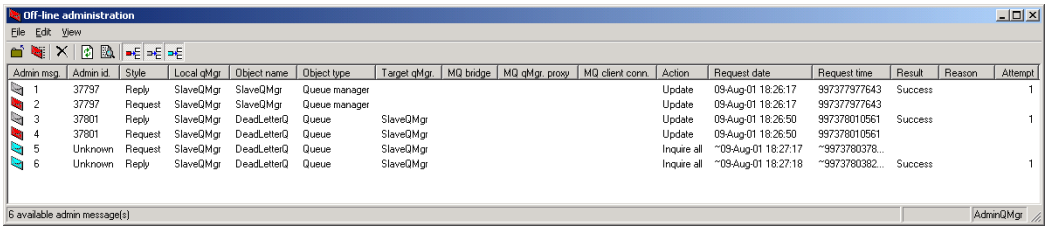


Figure 6-3: The off-line administration window

The example[20] shows two requests, 37797 and 37801, identified by the icon ; replies have been received. Successful replies are associated with the icon ; unsuccessful replies with . Two 'unknown' messages are also shown, indicated by the – in this case these are copies that have been requested of messages used by MQe_Explorer internally (see later). The following columns are displayed:

Admin message: a sequential numbering of the admin requests and responses, with an associate status icon (operation successful, partially unsuccessful or failed).

Admin id: an admin identifier for the request/response.

---

[19] In principle multiple responses can be received to an admin request; in MQe version 1.0 – 1.27 only a single response may be expected.

[20] For simplicity, a number of columns have been set to zero width and others re-ordered.

© Copyright IBM Corp. 2000, 2001

Style: request or response.

Local queue manager: the name of the queue manager holding the managed object.

Object name: the name of the managed object (for the bridges object the name of the local queue manager is displayed).

Object type: the type of the managed object (queue manager etc).

Target queue manager:
for a remote application queue - the queue queue manager;
for a store & forward queue – the target queue manager;
for a connection – the destination of the connection;
for an MQ queue manager proxy – the MQSeries queue manager;
for an MQ client connection – the MQSeries queue manager;
for an MQ listener – the MQSeries queue manager

MQ bridge: the name of the (associated) MQ bridge.

MQ queue manager proxy: the name of the (associated) MQ queue manager proxy.

MQ client connection: the name of the (associated) MQ client connection.

Action: the action to be performed (update, delete, inquire, etc).

Request date: the time the admin request was made, formatted as date/time. Where the admin id is unknown, the time is approximated to the request object creation time – and prefixed with "~".

Request time: the time the admin request was made, formatted as an integer. Where the admin id is unknown, the time is approximated to the request object creation time – and prefixed with "~".

Result: one of: "Success", "Mixed", "Fail".

Reason: an elaboration of *Result*.

Attempt: the admin operation attempt number for which this is the response.

Further details on requests and responses are shown if the relevant row is double clicked (or activated through the *View→Detail* menu item, or button). The properties to be managed are displayed – together with details of any errors that occurred, for example:

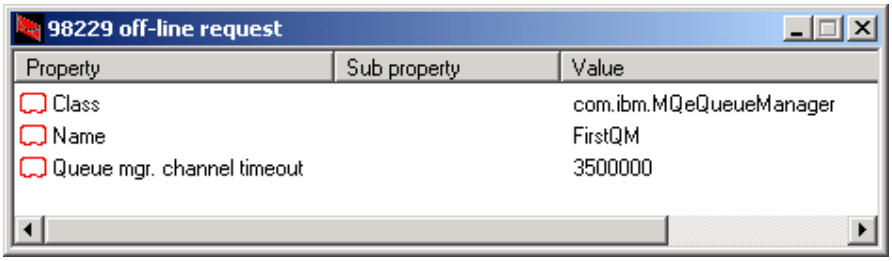


Figure 6-4: Off-line request detail

The off-line administration has many features in common with the main MQe_Explorer window. Columns can be re-ordered by dragging a column heading across other headings; clicking a column header sorts (and re-sorts) the data; one or more rows can be selected as required. Rows can be deleted or the display cleared. *File→Resend* or the button retries the admin operation; *View→Refresh* or the button refreshes the display. Context menus are supported.

The status bar at the bottom of the display has three panels; the left panel shows the number of status messages, the middle panel identifies the number of rows selected, and the right panel the hosting queue manager name (shown in [] brackets when MQe_Explorer is running as a slave).

© Copyright IBM Corp. 2000, 2001, 2002

6.1 Message properties

The off-line admin window displays all messages on the *AdminReplyQ* that have the following properties:

- The message class is *MQeAdminMsg* (or a subclass thereof).
- The message does not contain the field **Async_Load* unless it is a copy made by MQe_Explorer (i.e. unless it also contains the field **Original_CID*).

These conditions mean that all off-line requests and replies will be shown, as will all copies made of MQe_Explorer admin messages (i.e. those copies made by setting *Tools→Demo Mode*). Original inquiry messages issued by MQe_Explorer in the background will not appear.

Messages are classified as follows:

- messages with a non-zero **Msg_CorrelID* and a **Msg_Style* of 'request' are *Requests*.
- messages with a non-zero **Msg_CorrelID* and a **Msg_Style* of 'reply' are *Replies*.
- all other messages are *Others*.

© Copyright IBM Corp. 2000, 2001

7 Off-line administration menus

The off-line admin window shares many of the properties of the main list view pane. Thus columns may be sorted (or re-sorted) by clicking, columns may be re-ordered by dragging their headers, and rows may be double-clicked for additional details. Certain operations support multiple selected rows.

7.1 Edit

Clear

Clear deletes all the messages displayed in the off-line admin window; other off-line messages may still exist however, depending upon the current value of the view options (i.e. *Requests*, *Replies*, *Others*).

Delete

Delete deletes all the selected messages in the off-line admin window.

Select All

Select All selects all the messages in the off-line admin window.

7.2 File

Exit

Exit removes the off-line admin window.

Resend

Resend re-sends a previously sent admin request. The request to be resent must be selected in the off-line admin window.

7.3 View

Detail

Detail provides additional information on both admin request and replies.

Refresh

Refresh updates the admin off-line replies window with current messages.

Requests

If checked, then off-line request messages are displayed.

Replies

If checked, then off-line reply messages are displayed.

© Copyright IBM Corp. 2000, 2001, 2002

Others

If checked, then off-line other messages are displayed (i.e. messages that are not identifiable as either off-line admin requests or replies).

© Copyright IBM Corp. 2000, 2001

8 Console

The MQe_Explorer console provides a means of viewing and managing local applications. Since all applications are classes that have been loaded and assigned to threads, the console presents the information as a status list of JVM threads. The JVM allows MQe_Explorer to have knowledge only of threads that it owns, for example, the main thread used for its execution, or threads that either it has created, or that descend from its own threads. This means that the console cannot always show all threads present in a JVM. However, if MQe_Explorer was used to instantiate the JVM (and indeed the queue manager), then the console will present a complete picture. An example of a console window is shown below:

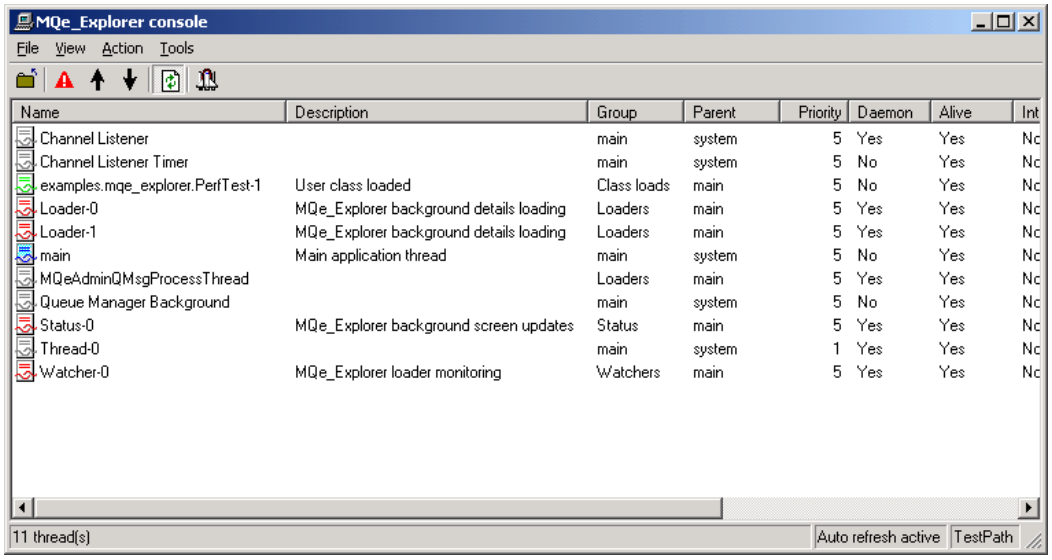


Figure 8-1: The console window

Each row represents a thread with the icons distinguishing various kinds. indicates a thread running a class loaded by the user; a thread used by MQe_Explorer; highlights the main thread; and is used for any other threads present (including those used by MQe). The columns give information about the thread:

Name: the name for threads created by MQe_Explorer. The name normally has a suffix which is a hexadecimal number increased sequentially as each thread of that type is created (starting at 0). Threads running user-loaded classes are named with the class name.

Description: further information on those threads known to MQe_Explorer.

Group: the name of the group to which the thread belongs.

Parent: the name of the parent thread.

Priority: thread priority (typically limited to the range of 0 – 10).

Daemon: indicates if the thread is a daemon, as opposed to a user thread.

Alive: indicates if the thread is alive.

Interrupted: indicates if the thread is interrupted.

Class: the thread class (threads running user-loaded classes and launched by MQe_Explorer always have the class name *java.lang.Thread*).

© Copyright IBM Corp. 2000, 2001, 2002

The status bar has three sub-panels; at the left is a status area used for messages and information; next is the auto-refresh status (see below); the rightmost sub-panel shows the name of the local queue manager (enclosed in [] if MQe_Explorer is running as a slave).

By default the panel is auto-refreshed, i.e. updated in the background without the need for user-interaction (and indeed the thread doing this – and other updates – is visible in the panel under the name *Status-x*).

Columns can be re-arranged by dragging the column headers; they can be sorted (and re-sorted) by clicking the same headers – however auto-refresh must be disabled before sorting is permitted.

Threads can be selected one at a time by clicking anywhere on the row. When selected, various operations are available:

Increase/decrease priority: increases/decreases the likelihood of the thread being scheduled.

Interrupt: interrupts the thread. The effect of this is entirely dependent upon the class being executed; interrupt behavior must be programmed into the class. In the case of the *Performance tool* shipped with MQe_Explorer, interrupting the thread will cause the tool to close immediately (if running a test) or to abort as soon as a new test is started.

Operations can be initiated from the menus, the toolbar, or by right clicking and bringing up the context menu. Certain operations are not allowed on some threads in order to protect the integrity of the system. If auto-refresh is enabled, then it is automatically suspended whilst thread operations are pending.

The Console window allows user classes to be launched through the *Tools→Load* menu item (or via the toolbar) – this is an alternative route to the class load function available from the main MQe_Explorer window.

© Copyright IBM Corp. 2000, 2001

9 Console menus

The console window shares many of the properties of the main list view pane. Thus columns may be sorted (or re-sorted) by clicking; likewise columns may be re-ordered by dragging their headers. Only one console window may be present.

9.1 Action

Increase priority

Increase Priority increases the selected thread priority by one, then updates the display.

Decrease priority

Decrease Priority decreases the selected thread priority by one, then updates the display.

Interrupt

Interrupt interrupts the selected thread, then updates the display.

9.2 File

Exit

Exit removes the console window.

9.3 Tools

Load

Load is identical to *Tools→Load* available in the main MQe_Explorer window.

9.4 View

Refresh

When checked auto-refresh is enabled (i.e. the thread display is updated automatically); when unchecked, no updates take place.

© Copyright IBM Corp. 2000, 2001, 2002

10 Trace

The trace window provides a means of debugging MQe applications (and MQe_Explorer and MQe). It traps the two output streams *System.out* and *System.err* and allows them to be re-directed to the display; additionally *System.err* may be logged to a file. The facility is implemented as a user interface on top of the MQe-provided class examples.trace.MQeTrace. The text area used to display output supports the rich text format (*.rtf*) and data from this area may be cut, copied or pasted to/from other applications, for example Microsoft Notepad, Wordpad or Word. Text may also be highlighted, colored, annotated, cleared, searched etc. MQe_Explorer supports a subset of text editing capabilities, including word select, deletion, backspacing, tab insertions, etc.

The nature of the *System.err* data collected is controlled through the *Action* and *View* menus. The *Action* menu determines the items that are traced – these are documented in the MQe publications, but in summary, MQe trace message types are traced as follows:

_: displayed if *System* is checked.

i: displayed if *System* & *Information* are checked.

w: displayed if *System* & *Warning* are checked.

e: displayed if *System* & *Error* are checked.

s: displayed if *Security* & *Error* are checked.

d: displayed if *System* & *Debug* are checked.

I: displayed if *Information* is checked.

W: displayed if *Warning* is checked.

E: displayed if *Error* is checked.

S: displayed if *Secuity* is checked.

D: displayed if *Debug* is checked.

Calls to debug: displayed if *Calls to debug* is checked.

The *View* menu options *Thread names*, *Timestamp* and *Object names* determine whether those characteristics are included in the traced data.

A continuous trace of the *System.err* data can be spooled to a disk file in plain text format (*.txt*); this is enabled via the *File→Log System.err* menu item. The same data can also be displayed in the text area under the *System.err* tab. However, in the latter case the display can be stopped and started via the *View→System.err* menu item – note that stopping and starting has no effect on disk file logging. The contents of the *System.err* text area can be saved, either as an *.rtf* or *.txt* file, through the *File→Save As* menu item (the *System.err* contents must be uppermost in the window).

In a similar manner *System.out* is shown in the text area under the *System.out* tab; the contents can be saved via the *File→Save As* menu item when the *System.out* contents are uppermost. There is no filtering of *System.out* – and no disk logging, however the editing commands are fully supported.

© Copyright IBM Corp. 2000, 2001

An example of the trace window is shown below:

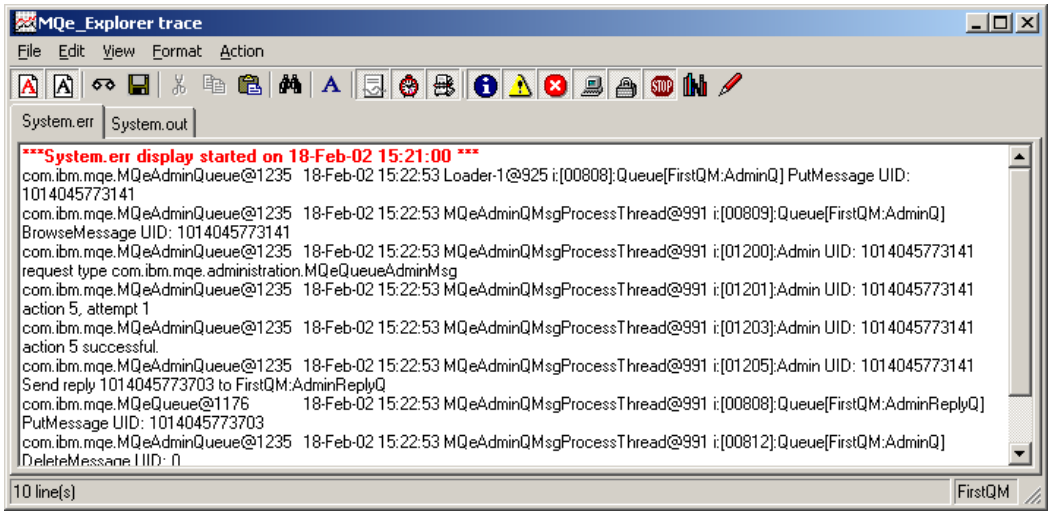


Figure 10-1: The trace window

© Copyright IBM Corp. 2000, 2001, 2002

11 Trace menus

The console window shares many of the properties of the main list view pane. Thus columns may be sorted (or re-sorted) by clicking; likewise columns may be re-ordered by dragging their headers. Only one console window may be present.

11.1 Action

Call to debug

Call to Debug causes call to debug trace messages to be captured on *System.err*. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

Debug

Debug causes debug trace messages to be captured on *System.err*. *Debug* by itself captures application debug messages; *Debug* with *System* also captures system debug messages. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

Error

Error causes error trace messages to be captured on *System.err*. *Error* by itself captures application error messages; *Error* with *System* also captures system error messages; *Error* with *Security* captures system security messages. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

Exception

Exception causes exception trace messages to be captured on *System.err*. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

Information

Information causes information trace messages to be captured on *System.err*. *Information* by itself captures application information messages; *Information* with *System* also captures system information messages. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

Security

Security causes security trace messages to be captured on *System.err*. *Security* by itself captures application security messages; *Security* with *Error* also captures system security messages. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled..

© Copyright IBM Corp. 2000, 2001

System

System causes system trace messages to be captured on *System.err*. It can also be used in conjunction with *Information*, *Warning* and *Error* to capture system information, warning and error messages. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

Warning

Warning causes warning trace messages to be captured on *System.err*. *Warning* by itself captures application warning messages; *Warning* with *System* also captures system warning messages. If logging is enabled, they will be written to a log file, along with any other captured messages. They are also displayed in the text area under the *System.err* tab, provided that viewing of *System.err* is enabled.

11.2 Edit

Clear

Clear deletes the content of the uppermost displayed text area (i.e. the output from *System.err* or *System.out)*.

Copy

Copy copies the currently selected text in the uppermost displayed text area (i.e. the output from *System.err* or *System.out*) to the system clipboard. The text is stored in both rich text and plain text formats.

Cut

Cut cuts the currently selected text in the uppermost displayed text area (i.e. the output from *System.err* or *System.out*) to the system clipboard. The text is stored in both rich text and plain text formats.

Delete

Delete deletes the currently selected text in the uppermost displayed text area (i.e. the output from *System.err* or *System.out*).

Find

Find locates a text string in the uppermost displayed text area (i.e. the output from *System.err* or *System.out*) and, if found, leaves the result selected. If *Find* is invoked with text selected, then that selected string is primed as the search string. All search strings used in an invocation of *Find* are available for re-use in the drop down list box. Repeat identical searches begin where the previous search finished, until all the target text area has been searched. Options are provided for case matched and whole word searches.

Paste

Paste pastes text from the system clipboard to the uppermost displayed text area (i.e. the output from *System.err* or *System.out*). If text at the destination is selected, it is replaced; otherwise the clipboard contents are inserted at the current text cursor position.

© Copyright IBM Corp. 2000, 2001, 2002

Select all

Select All selects the entire contents of the uppermost displayed text area (i.e. the output from *System.err* or *System.out*).

11.3 File

Exit

Exit stops tracing and removes the trace window.

Log system.err

Log System.err causes the output of *System.err* to be written to a log file in plain text format (*.txt*). The contents logged are controlled by the settings chosen in the *Action* and *View* menus. The logging is unaffected by stopping and starting the display in the text area, nor does it reflect any changes made to the text area contents.

Save as

Save As saves the entire contents of the uppermost displayed text area (i.e. the output from *System.err* or *System.out*) as either a plain text file (*.txt*) or as a rich text file (*.rft*).

11.4 Format

Font

Font formats the currently selected text in the uppermost displayed text area (i.e. the output from *System.err* or *System.out*). The font, style, size, color and effects may be set.

11.5 View

Object names

Object Names determines whether object names are included in the messages displayed or logged.

System.err

System.err stops/starts the display of the *System.err* stream in the text area. A timed message is added, indicating when the stream was stopped/started. Note that this message does not appear in any logged output.

System.out

System.out stops/starts the display of the *System.out* stream in the text area. A timed message is added, indicating when the stream was stopped/started.

Thread names

Thread Names determines whether thread names are included in the messages displayed or logged.

© Copyright IBM Corp. 2000, 2001

Timestamp

Timestamp determines whether timestamps are included in the messages displayed or logged.

© Copyright IBM Corp. 2000, 2001, 2002

12 Advanced topics

12.1 Configuration files

All settings that have been created through the *Tools→Options* menu are stored in a configuration file, which is identifiable by its *.cfg* extension[21]. By default this file is named *MQe_Explorer.cfg* and by default is expected to be located in the same directory as the executable *MQe_Explorer.exe*. If the configuration file cannot be found then a message box will be displayed requesting permission to create a new one – when created, it will be initialized with the default values for the MQe_Explorer options.

If multiple instances of MQe_Explorer are run from a single *.exe* then, by default, they will share a single configuration file. This is permitted; note however that the file is only loaded when an MQe_Explorer instance is initialized and it is saved whenever the *Apply* button is clicked after one (or more) options have been changed. The stored file will therefore reflect the state of the last MQe_Explorer to change an option, but any such changes will not affect other already running instances.

If individual instances are to use their own configuration file, then the file path must be passed as a run time parameter, see *Invocation* on page 122 for more details. The configuration file in use is listed in the *System Information*, accessible from the *Help→About MQe_Explorer* menu item.

[21] Configuration files (*.cfg*) have the internal format of an initialization file (*.ini*) and the advanced user can edit them manually, for example with the Windows *Notepad* editor (but changes must not be made that would not be permitted by MQe_Explorer – for example, bracketed values for *Classes* must not be modified in any way). These files can also be read and written through MQe, for example with *com.ibm.mqe.MQeQueueManagerUtils.loadConfigFile()*.

© Copyright IBM Corp. 2000, 2001

12.2 Initialization

Initialization files

MQe_Explorer supports the use of standard MQe initialization files (typically identified by a *.ini* extension) to establish the queue manager environment and to identify the queue manager to be loaded. The following sections are supported:

[Alias]	*defines alias names for MQe classes.*
[ChannelManager]	*sets channel manager parameters, such as the maximum number of incoming concurrent channels permitted.*
[Listener]	*sets listener properties, such as the TCP/IP port number and the communications adapter.*
[MQBridge]	*sets parameters for the bridges object (e.g. the load bridge rule).*
[MQe_Explorer]	*holds information required on communication parameters – used to configure other queue managers. It also captures information on the required type of the queue manager (client, peer, server or gateway)*[22]*.*
[Permission]	*sets allowed channel commands, file descriptor mappings and adapter calls.*
[PreLoad]	*identifies classes to be pre-loaded when the queue manager is activated.*
[QueueManager]	*identifies the queue manager by name.*
[Registry]	*locates the registry and sets parameters (including security).*

The *[Alias]*, *[QueueManager]* and *[Registry]* sections must be present; others are optional – note however that many aliases are in common use in MQe installations and therefore an *[Alias]* section containing these well-known defaults is strongly recommended.

[22] The contents of the [MQe_Explorer] section are described in *Initialization file* section on page 180.

© Copyright IBM Corp. 2000, 2001, 2002

The *FirstQM.ini* file, created in *Configuring a first queue manager* on page 7 has the content:

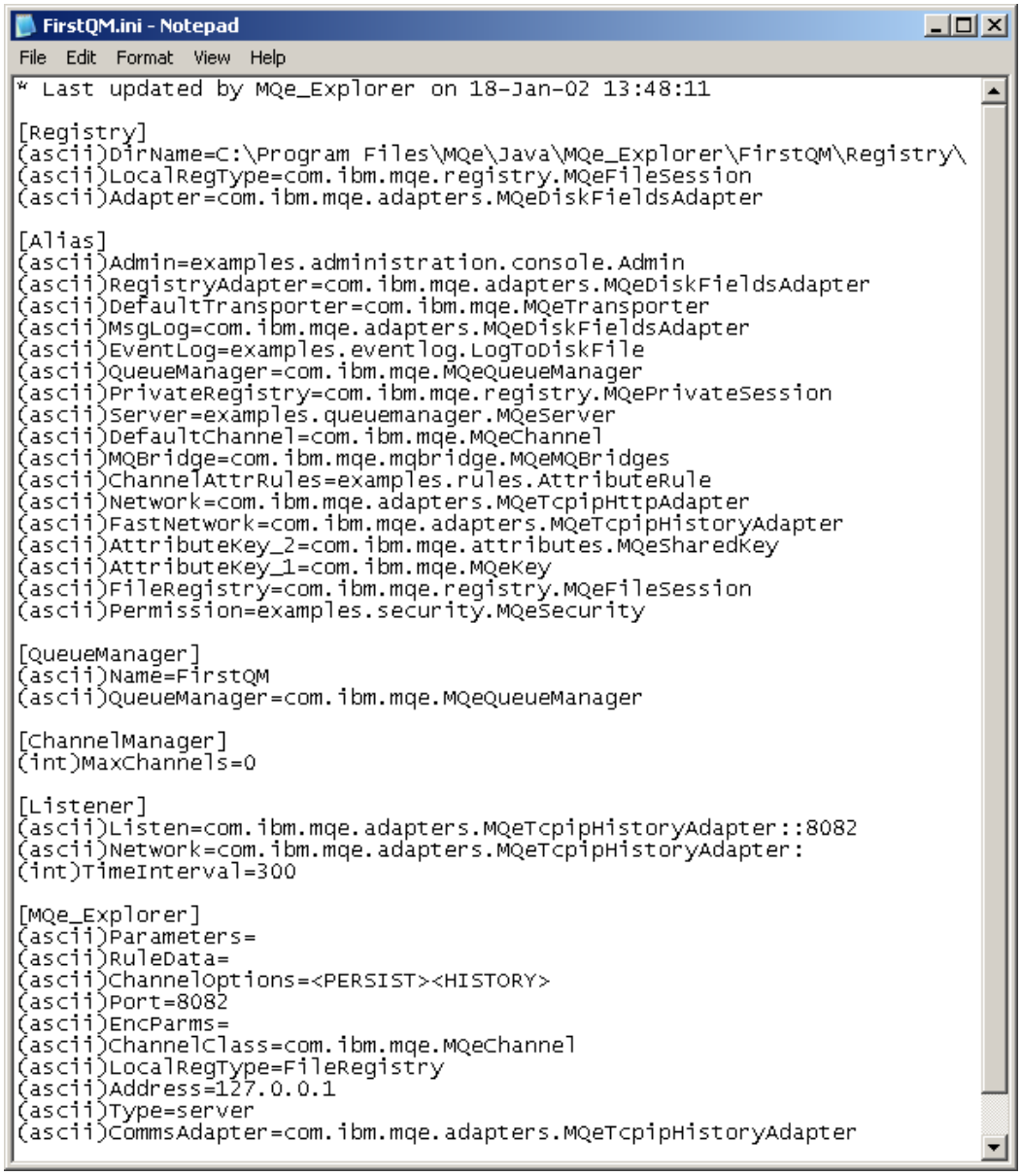


Figure 12-1: The *FirstQM* initialization file

Initialization files can be edited with the Windows *Notepad* editor. However, for local queue managers on Windows platforms it should not be necessary to edit such files – MQe_Explorer provides higher level functions to manage their content.

Within an initialization file, paths can be specified for certain parameters, for example, the location of the directory containing the queue manager registry. If a relative path is used (.\) it will be taken as being relative to the location of the *.ini* file itself[23].

[23] Note that this differs from the default MQe behavior for initialization file processing; note also that it differs from the way in which relative paths in run time parameters are interpreted. MQe_Explorer will issue a warning message if relative paths are in use.

© Copyright IBM Corp. 2000, 2001

Passwords can be placed in MQe initialization files. To avoid the adverse security implications of this technique, MQe_Explorer implements a special value for such passwords. If the value is set to '*(prompt*)' then MQe_Explorer will prompt for the actual password during queue manager loading. Passwords must meet these criteria:

- *be at least 6 characters in length.*
- *contain at least 4 unique characters.*
- *not match various excluded values.*

Sections of the initialization file that are to be processed by MQe_Explorer during queue manager loading are specified in the *Tools→Options→Open qMgrs* dialog:

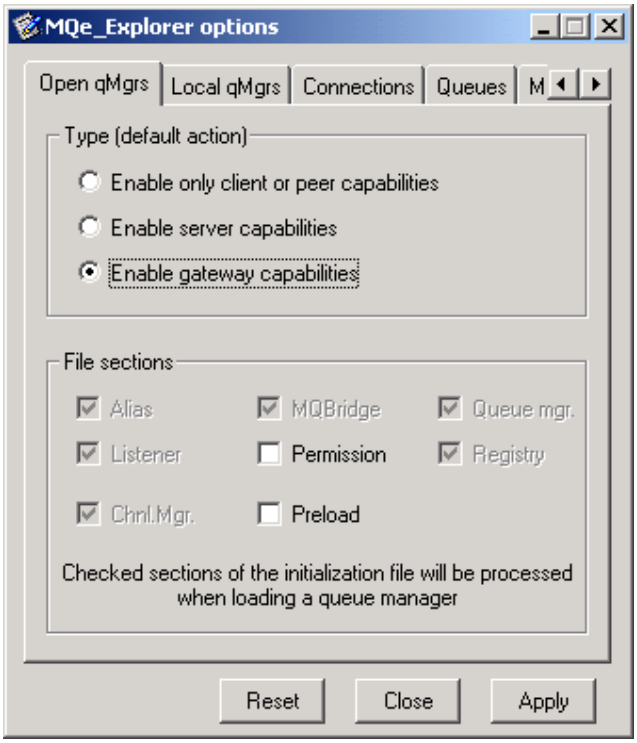


Figure 12-2: The options dialog – Open qMgrs tab

If the initialization file has an *[MQe_Explorer]* section (i.e. it has been created by MQe_Explorer via the *File→New* menu item), then the queue manager type requirements will take priority over any such option specification (e.g. a server or gateway queue manager will always have the *[Listener]* and [*ChannelManager]* sections processed; a gateway queue manager will have the *[MQBridge]* section processed, etc).

The initialization file in use is listed in the *System Information*, accessible from the *Help→About MQe_Explorer* menu item.

Connection style

The *Type* group options determines the queue manager capabilities that are to be activated on loading - effective only for those queue managers not created by MQe_Explorer:

- *Enable only client or peer capabilities:* this option disables the processing of the *[Listener]* and the *[ChannelManager]* sections of the initialization file – the MQe channel listener and channel manager objects are not created. Server operation is therefore unavailable.

© Copyright IBM Corp. 2000, 2001, 2002

- *Enable server capabilities:* this option enables the processing of the *[Listener]* and the *[ChannelManager]* sections of the initialization file – the MQe channel listener and channel manager objects are created (if the sections contain the required information.
- *Enable gateway capabilities:* this option enables server capabilities (as above); additionally it allows the bridges object to be created for gateway operation, provided that the initialization file contains a *[MQBridge]* section.

The *File sections* group determines which other sections are to be processed.

Note that initialization is concerned with creating the queue manager environment and loading the queue manager. It does not include defining the connections that are required before queue managers can communicate with each other. These connection requirements are:

For *peer-to-peer* communications:

- If the queue manager is to be capable of initiating *peer-to-peer* communications (i.e. to be a *master*) it must have connection definitions (using a *peer-to-peer channel*) for each peer to be contacted (that is, for each *slave*); these definitions include the network address of the slave.
- If the queue manager is to be capable of responding to a single *peer-to-peer* connection request at a time (i.e. to be a *slave*) it must have a *local connection definition*[24] (using a *peer-to-peer channel*); this definition must be that of a listener[25]. If a queue manager is defined with a type setting of peer, then the necessary listener will be automatically created by MQe_Explorer. If a server or gateway queue manager is also to have peer listener capabilities then the necessary definition can be added manually; alternatively a peer-type queue manager queue manager can be upgraded to a server-type or gateway-type queue manager by MQe_Explorer, in which case the peer listener will be retained.

For a *client-server* communications[26]:

- If the queue manager is to be capable of initiating *client-serv*er communications (i.e. to be a *client*) it must have connection definitions (using a *standard channel*) for each server to be contacted; the definitions include the network address of the server.
- If the queue manager is to be capable of responding to clients (i.e. to be a *server*) it must have the *listener* and *channel manager* components present. This is most easily achieved by having MQe_Explorer create (or upgrade) the queue manager; alternatively the initialization file can be appropriately modified.

By default, MQe_Explorer loads queue managers that it has created with the necessary components to match their chosen type; other queue managers are loaded with the components necessary for *client*, *server* and *peer* communications capabilities, subject to the initialization file containing the relevant sections.

If remote administration is required, then MQe_Explorer itself imposes communications requirements on the queue managers. It initiates communications with the queue managers that it manages; likewise it receives multiple concurrent responses from those managed

---

[24] A local connection definition is a connection definition to *itself*, i.e. the name of the connection is the name of the local queue manager.
[25] This definition specifies an adapter and a port number – there is no address present.
[26] The essence of client-server is that the server listens on a single address and on that address can process multiple incoming connection requests.

© Copyright IBM Corp. 2000, 2001

queue managers – which may appear as incoming connection requests. Consequently, MQe_Explorer needs to run on a queue manager that has one or both of the following capabilities (depending upon the nature of the queue managers being managed):

- *A master in a peer-to-peer configuration* – if the target is a peer.
- *A server in a client-server configuration* – if the target is a client, server or a gateway.

The managed queue manager needs to be able to receive messages from MQe_Explorer and to be able to initiate the sending of reply messages. Therefore:

- If the managed queue manager is a peer:
  - *MQe_Explorer must have peer-to-peer capabilities*
- If the managed queue manager is a server or gateway
  - *MQe_Explorer must have server capabilities*
    (note that a server/gateway includes the components necessary for a client)
- If the managed queue manager is a client
  - *MQe_Explorer must have server capabilities*
  - *The client must have a home server queue defined to pull its admin messages from MQe_Explorer*[27]

[27] It is not possible for MQe_Explorer (or any other tool using admin messages) to manage a client that does not have a home server queue defined. This also requires that a store queue (a store & forward queue without a target queue manager) has been established in the network, with any other definitions needed to ensure that admin messages go via this route. For on-line administration, the home server queue must pull messages every few seconds.

© Copyright IBM Corp. 2000, 2001, 2002

Default configuration

If a new queue manager is created through the *File→New→Queue Manager* menu item, it must be configured, i.e. its registry needs to be loaded with initial objects, values, etc. Similarly, if an initialization file for an un-configured queue manager is opened, MQe_Explorer will configure it (subject to operator confirmation). Some of the configuration details are controlled by sections in the initialization file[28] – such as *[Registry]* and *[QueueManager]*; others by the settings in the *Tools→Options→Advanced-1* tab, *New configuration* group:

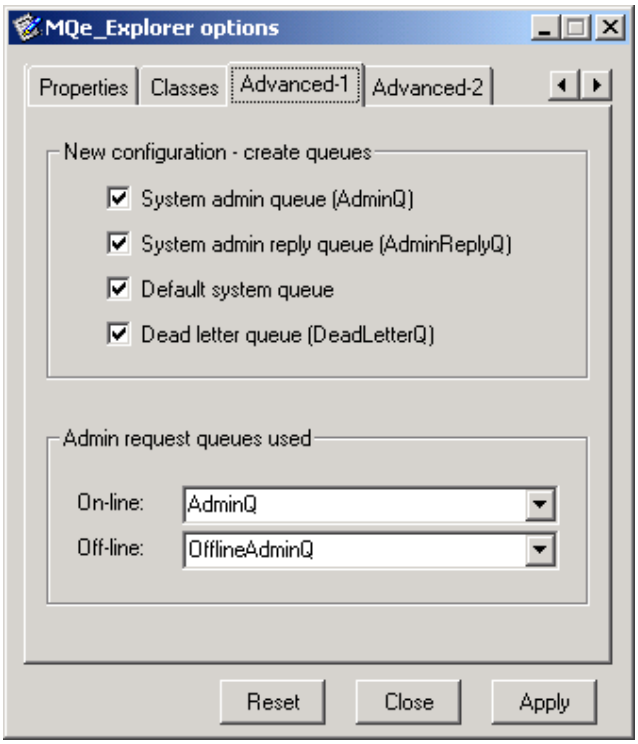


Figure 12-3: The options dialog – Advanced-1 tab

The *New configuration* parameters determine the queues to be initially created; others can be created later using the standard MQe_Explorer functions:

> The *AdminQ* and the *AdminReplyQ* queues are always present because they are needed for MQe_Explorer operation. The MQe *Default system queue* (SYSTEM.DEFAULT.LOCAL.QUEUE) may optionally be created, as may the *DeadLetterQ*; neither of these is directly used by MQe_Explorer.

[28] Note that the sections to be used are controlled by the *Tools→Options→Open qMgrs* panel.

© Copyright IBM Corp. 2000, 2001

12.3 Invocation

Invocation options

MQe_Explorer can be invoked in three different ways:

1. *By executing MQe_Explorer.exe*

   Only all MQe_Explorer classes required are contained within the executable. The *classpath* is required to identify the source of the MQe classes and any additional classes that may be specified.

2. *By calling the static main method of the MQe_Explorer.class*

   All MQe and MQe_Explorer classes are located via the *classpath*. The main method is declared as:

   public static void main(String args[ ])

   This technique allows MQe_Explorer to run inside an already created JVM. If an existing queue manager is present then MQe_Explorer will use it. In this mode the queue manager name shown in the status bar is enclosed in [ ] brackets to indicate that MQe_Explorer is running as a slave application. As a slave, certain operations are grayed out, for example:

   File→Open, File→New, File→Close

   When invoking MQe_Explorer in this way, the calling application should create a new thread to be used for the MQe_Explorer invocation.

Invocation parameters

Irrespective of the way in which MQe_Explorer is invoked, it may take up to three run time parameters:

1. *Configuration file specification: /c <file path> (or: -c <file path>)*

   Where *<file path>* is the configuration file path specification. The file type must be *.cfg* – if this extension is not present in the string then it will be added. If a relative path is used (for example: .\) this will be taken to be relative to the location of the current directory (typically the location of the *.exe* file). If the named file cannot be found, a default file will be created with that specification.

2. *Initialization file specification: /i <file path> (or: -i <file path>)*

   Where *<file path>* is the initialization file path specification. The file type may be any type but *.ini* would be conventional. If a relative path is used (for example: .\) this will be taken to be relative to the location of the current directory (typically the location of the *.exe* file).

3. *Message suppression specification: /s <suppression keyword> (or: -s <suppression keyword>)*

   Where valid values of *<suppression keyword> are:*

   suppressIO – suppress all background loading messages that report I/O errors.

Thus the following is acceptable:

```
MQe_Explorer.exe    /i  .\MyMQe\FirstQM.ini   /c  C:\Options\MQe_Explorer.cfg
```

© Copyright IBM Corp. 2000, 2001, 2002

Additional control of MQe_Explorer operation is available; these interfaces are documented in MQe_Explorer external variables on page 182 .

12.4 Managing remote queue managers

Connections

The local queue manager hosting MQe_Explorer needs connectivity to each remote queue manager to be managed; these definitions must be entered manually and can be constructed from either connections definitions and/or destinations listed in store (and forward) queues, plus whatever aliases are also required. The target queue manager also needs to be appropriately configured. This can be done either manually (see later) or, if the target is not a client, MQe_Explorer can do the configuration remotely via te *Tools→Configure Remote* menu, after a connection definition has been created (also see later).

Once the target queue manager has been configured for administration, full remote management is possible. However it may be the case that the target queue manager has its own connectivity definitions that relate to queue managers unknown to the local queue manager hosting MQe_Explorer. This will prevent certain operations; for example it will not be possible to browse queues on those distant queue managers, nor to send them messages.

To overcome this, add additional connection definitions to the local queue manager hosting MQe_Explorer. These can be either direct or indirect – whatever is appropriate. If these additional queue managers are not MQe queue managers but instead belong to MQSeries, then add indirect connection definitions that specify that communications are to be routed via the appropriate MQe gateway.

Queues

MQe_Explorer sends admin messages to target queue managers; it receives replies back on the admin reply queue of its local queue manager. The admin queue to which messages are addressed depends upon the settings in the *Tools→Options→Advanced-1* tab, *Admin request queues used* group. By default, the names are:

On-line working*: AdminQ*

Off-line working*: OfflineAdminQ*

These names cannot be varied on an individual target queue manager basis. If such unique names are required then an appropriate queue aliases should be added, corresponding to the name(s) that MQe_Explorer will use.

The admin reply queue of the local queue manager used to receive admin replies is called *AdminReplyQ*; in order to avoid potential problems the properties of this queue cannot be modified through MQe_Explorer.

If MQe_Explorer is used to create (or configure) a queue manager it will always create the *AdminReplyQ* and an admin queue called *AdminQ*. If MQe_Explorer is run on a pre-configured queue manager that does not have these queues, then it will add them. It does not explicitly create *OfflineAdminQ* because MQe_Explorer has no use for it on the local queue manager.

If off-line working and on-line working are both required, the remote queue manager to be managed must have an alias name set for the *AdminQ* (with this name being *OfflineAdminQ* in the default case). Additionally, in all cases where off-line working is required, the local queue manager hosting MQe_Explorer must have an asynchronous remote queue definition for that admin queue – with this definition using the chosen alias queue name to refer to the remote admin queue.

© Copyright IBM Corp. 2000, 2001

Connectivity with remote clients

If the remote queue manager to be managed is a *client*, then the local queue manager hosting MQe_Explorer must have *server* capabilities; for more information on this topic see *Queue manager* on page 39. For information to flow from the client to the server at the instigation of the client, the client must have a connection definition to the server (which may be direct or indirect); for more information see *Connection* on page 15. By definition, the server is not allowed to instigate data transfer to the client – hence messages from MQe_Explorer to the client must be pulled by the client from the server. This requires that a store queue (a store and forward queue without a forwarding target) be defined on the server, with the name of client added as a destination for which it will collect messages. To complement this, a home server queue must be defined on the client, pointing at this store queue on the server. The time interval property on the home server queue controls how often the client polls for messages. For more information on store and forward queues and home server queues see *Queue* on page 27.

Connectivity with remote peers

If the remote queue manager to be managed is a *peer*, then the local queue manager hosting MQe_Explorer must have *peer* capabilities (for more information on this topic see *Queue manager* on page 39). Since MQe_Explorer instigates messaging activity, the local queue manager must be configured as a master and the remote queue manager as a slave. This means that the local queue manager has a direct connection definition to the remote queue manager, using peer channels. The remote queue manager has a peer listener connection definition. For more information see *Connection* on page 15.

The remote queue manager replies to received admin requests using the channel created by the incoming peer connection request. The channel timeouts should be set such that the channel is not dropped before the remote queue manager has had a chance to reply. If this eventually is to be completely avoided then additional configuration should be undertaken such that the remote queue manager can also act as a master and the local queue manager as a slave (i.e. a direct connection definition using peer channels added to the remote queue manager, and a peer listener definition added to the local queue manager).

Connectivity with remote servers and gateways

If the remote queue manager to be managed is a *server* (or *gateway*), then the local queue manager hosting MQe_Explorer must have *server* capabilities (for more information on this topic see *Queue manager* on page 39). This is true because a server also includes *client* capabilities. For MQe_Explorer to send a message to the remote queue manager, the local queue manager must have a direct (or indirect) connection definition to the remote queue manager, using client/server channels. Likewise, for the remote queue manager to send a message to the local queue manager, it must have a direct (or indirect) connection definition to the local queue manager, also using client/server channels. For more information see *Connection* on page 15.

© Copyright IBM Corp. 2000, 2001, 2002

Automatic remote configuration

MQe_Explorer can configure remote queue managers (excepting clients) for on-line and/or off-line administration through the *Tools→Configure Remote* menu. In this case it creates:

- *On the local queue manager*
  - *Synchronous and/or asynchronous remote queue definition(s) to target admin queue.*
- *On the target queue manager*
  - *Alias(es) to the admin queue (if required, such that the alias name(s) match the name(s) used locally in the remote queue definitions).*
  - *A remote queue definition to the local queue manager AdminReplyQ (synchronous if only on-line working required – otherwise asynchronous).*
  - *A connection definition to the local queue manager.*

Automatic remote configuration requires that the remote queue managers have the necessary listeners already defined, such that they can receive admin messages from MQe_Explorer. For peer queue managers, remote configuration supports both local and remote queue managers being either masters or slaves.

12.5 MQe_Explorer messages

MQe_Explorer uniquely identifies the messages it uses to distinguish them from any others in the network, and to avoid confusion on shared queues.

- All messages include a Boolean field with a coded name[29] that translates to **MQe_Explorer* and has a value of 'true'.
- Messages that inquire on object properties include a boolean field with a coded name that translates to **Async_Load* and has a value of 'true'.
- Messages that inquire on, change object properties or ping queue managers, include a byte array field with a coded name that translates to **Msg_ CorrelID* and has a non-zero value.
- Messages that ping queue managers include a long field with a coded name that translates to *Ping_Time.
- Messages that are copied (as a result of *Tools→Demo Mode* being checked) have the **Msg_CorrelID* field set to a zero value and the UID re-set. Additionally they have the following fields added:
  - A UNICODE field with a coded name that translates to **Copy_Info* and a value detailing when the copy was made
  - A byte array field with a coded name that translates to **Original_CID* with a value of the original **Msg_CorrelID* field
  - A fields field with a coded name that translates to **Original_UID* with a value of the original message UID.

[29] This is a reserved UNICODE string value that is short, but unprintable. MQe_Explorer translates it when displaying field names – all such translated names begin with the * character.

© Copyright IBM Corp. 2000, 2001

12.6 Operation

MQe_Explorer caches MQe object details in order to improve responsiveness. This technique also allows a static (but refreshable) view of elements of the network to be explored and this has important usability benefits. Since messages are constantly arriving at and being deleted from queues, a truly dynamic presentation of the object status would inhibit effective user interaction.

The principal caches used are:

- *Object cache* – contains an object for each of the following inquiries:
  - *Connection details* (both remote and local connections).
  - *MQ bridge details*
  - *MQ client connection details*
  - *MQ listener details*
  - *MQ queue manager proxy details*
  - *Queue details* (both local and remote queues).
  - *Queue manager details.*
- *Message cache* – contains an object for each unique combination of message and associated path.

The object cache is loaded asynchronously and its operation may be controlled through the *Tools→Options→Advanced-2* settings. The message cache is loaded synchronously and is not configurable.

Object cache

The status of the cache is displayed in the status bar (in the third panel from the left at the bottom of the main window). The first number indicates the number of object request details outstanding; the second shows the total number of objects in the cache. Further details on the cache contents can be obtained by double clicking the status bar – the following panel is displayed:

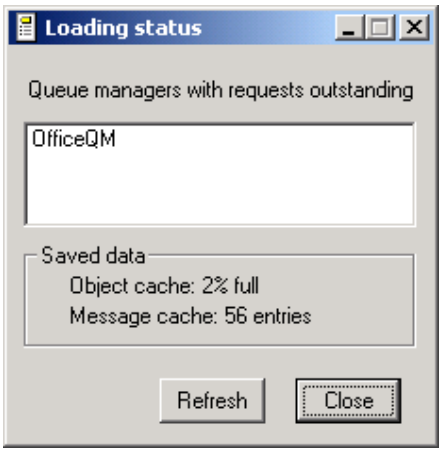


Figure 12-4: Loading status

This shows those queue managers that have failed to respond – the missing response may be for queue manager details, or for any related object owned by that queue manager. An entry here will typically indicate that either the queue manager is down or that the network is not functional.

© Copyright IBM Corp. 2000, 2001, 2002

If the cache is full, the oldest entry will be deleted to make room for a new request (subject to the protection of key entries). This may mean that information is refreshed when not expected – or that delays occur. Where possible the cache size should be set sufficiently large to avoid this eventuality.

Control of the object cache is through *Tools→Options→Advanced-2*:

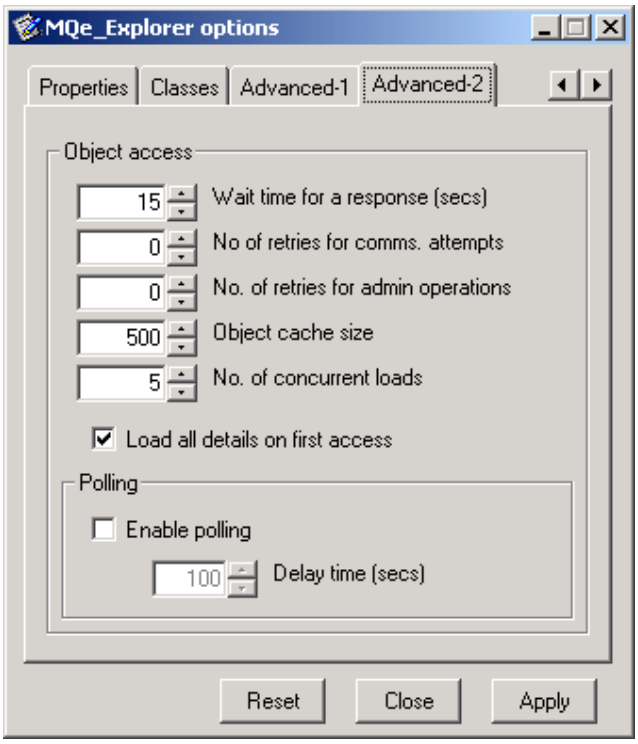


Figure 12-5: The options dialog – Advanced-2 tab

The significance of the parameters is:

Wait time for a response[30]*:* if no response to an admin request is received after the specified number of seconds, a further attempt (if permitted) is scheduled.

Number of retries for communication attempts: if an admin request results in no response, successive attempts are scheduled up to this limit.

Number of retries for admin operations[31]*:* the number of retries attempted on the target queue manager before the request is deemed to have failed.

Object cache size: the number of objects held in the cache.

Number of concurrent loads: the maximum number of background threads used to load objects into the cache. The current number of background threads can be seen by displaying the Console window (*View→Console*); all such threads belonging to the *Loaders* group.

Load all details on first access: if unchecked, details are requested just ahead of being required (for example, when the *MQe root* node is selected in the tree view pane the queue manager details of all known queue managers are requested; when a *Local queues* folder is selected, the relevant

[30] This parameter is also used to set the wait time for property update requests.
[31] This parameter is also used to set the admin operation retry count for property update requests.

© Copyright IBM Corp. 2000, 2001

queue details are requested). If checked, when a local queue manager is selected in the tree view pane, then details on all of its associated objects are requested at that time.

Enable polling: if checked, then if the initial attempt(s) fail, periodic successive requests are made for the object details.

Polling delay time: the time in seconds between polling requests.

© Copyright IBM Corp. 2000, 2001, 2002

13 Sample scripts

The following scripts[32] are supplied to introduce the features of MQSeries Everyplace and to gain familiarity with the MQe_Explorer:

1. *Concepts and objects*
    - An overview of MQSeries Everyplace messaging including the basic objects and their properties.
    - A general introduction to the use of MQe_Explorer.
2. *Basic messaging*
    - Illustrates direct and indirect connections between queue managers.
    - Synchronous and asynchronous messaging.
    - Use of the home server and store & forward queues.
3. *Advanced messaging*
    - Illustrates the use of the home server, store & forward queues and intermediate network queuing.
4. *Gateway configuration and usage*
    - Illustrates the use of MQe_Explorer to configure a gateway queue manager and how an MQe gateway can be used to exchange messages with MQSeries.

Note: In the following scripts blue text is used to highlight actions to be taken.

[32] The scripts (arbitrarily) use MQe messages flowing over a raw TCP/IP protocol to *local host*; with minor configuration changes they can be modified to exploit HTTP. When using a physical network to connect machines with different IP addresses, ensure that the influence of any SOCKS stacks and/or HTTP proxies set-up is fully taken into account. Whilst all these configurations can be supported, the presence of firewalls, SOCKS and HTTP proxies can require explicit MQe configuration changes.

© Copyright IBM Corp. 2000, 2001

13.1 Concepts and objects

Overview

This script introduces the key components of messaging and the object structure of MQSeries Everyplace (MQe) and is also useful as an introduction to the MQe_Explorer. It alludes to the many advanced facilities of MQe but makes no attempt to demonstrate all of the features. It is written such that it can be run on a standalone machine, for example an IBM ThinkPad, by using multiple queue managers listening on different ports of the local host.

Script

Introduction

It is assumed that the computer has no queue managers defined – and therefore no queues, no messages etc. The machine is in the state expected following an initial install of the MQe product, followed by MQe_Explorer. MQe_Explorer is used to create everything needed. Firstly a queue manager must be created[33].

Creating a queue manager

Double click the MQe_Explorer desktop icon. The MQe_Explorer window appears[34].

This window looks like the *Windows File Explorer* with menus, a toolbar and left and right hand panes. To create new queue manager:

Click the 🗋 on the toolbar (the *New folder* button).

The new queue manager creation dialog is displayed. As a minimum, the new queue manager name needs to be entered.

Enter the new queue manager name "FirstQM". Check that the path is *C:\Program Files\MQe\Java\MQe_Explorer* – or edit the path to this value[35]. Leave the *Server* check box checked. Leave the other fields with their default values.

On the *Comms.* tab, check that the IP address is correct – if you are connected to a network it will show your current IP address; if you are not connected, the address will be 127.0.0.1. If you have a dynamic IP address or you will want to work in disconnected mode in the future (as well as work on-line) edit the IP address to 127.0.0.1[36]. The port number can normally be left as 8082. Click the *Create* button.

A message is issued giving the location of the new queue manager's initialization file. This is the file that must be opened if you subsequently wish to restart this queue manager.

---

[33] If you followed *Configuring a first queue manager* on page 7 you already have the first queue manager *FirstQM* defined.
[34] If there is no configuration file available (used by MQe_Explorer to store option settings) a message will appear. Click *OK*.
[35] This is the recommended path name but you may change it – the significance of this parameter is that it identifies where your queue manager and associated files will be stored.
[36] This address is not actually used by this queue manager – it is made available to others – but can be edited again at that time.

© Copyright IBM Corp. 2000, 2001, 2002

Click OK on the initialisation file message.

Now many things have happened. A configured and running queue manager has been created. A message appears in the status area at the bottom left "1 queue manager(s)" – meaning that MQe_Explorer only knows about one queue manager. On the same status bar at the far right is a panel containing the string "FirstQM" – this gives the name of the queue manager used by this instance of MQe_Explorer for its admin operations.

Hover the cursor over the various status bar elements (starting at the left and moving to the right).

The leftmost panel is already understood, the next panel shows the nature of the local environment, the next the number of objects selected in a multi-select operation, the next the cache object request status "0:7" that is the number of requests outstanding is 0, versus a total number of requests of 7. MQe_Explorer asynchronously requests information on various objects (such as queue managers, queues, connections etc.); thus so far it has received information back on everything requested and its cache contains six such results. The last panel shows the name of the local queue manager.

In the left main panel (the tree pane) is displayed the root of a tree – a map of the MQe network. So far there is just a root (called *MQe root*) and a ⊞ sign that indicates there is more information to be seen.

In the right hand panel (the list view pane) a row shows details of the *FirstQM* queue manager just created.

Expand the application window by pulling the bottom left hand corner down and to the right.

Now the full description of the queue manager is displayed – confirming that MQe_Explorer created it a few seconds ago. Other properties are apparent.

Move the scroll bar in the list view pane to the right.

More properties are visible; the queue manager has a class (which is truncated). The full class name can be seen if the column width is increased.

Drag the right hand side of the *Class* column header to the right. The column width increases. Then move the scroll bar so that the next set of columns is clearly visible (number of queues, connections, etc).

The new queue manager has three queues – these are queues that MQe_Explorer has created by default because they are generally useful – there is no MQe requirement that such queues must be present. No connections are defined (unsurprisingly) – a connection definition describes how one queue manager communicates with another. There is a queue path – which determines where the queue content is to be stored by default; similarly the queue adapter determines the default mechanism (the local file system).

Bring up the *Windows Explorer* and navigate to the directory *C:\Program Files\MQe\Java\MQe_Explorer* and display its contents.

A new folder has been created called *FirstQM* – holding the new queue manager.

© Copyright IBM Corp. 2000, 2001

In the *Windows Explorer* open the folder *C:\Program Files\MQe\Java\MQe_Explorer\FirstQM* in the left hand pane and expand all the sub-directories so that the full folder structure is visible.

A collection of files and folders has been created – divisible into queue folders and registry folders. The registry holds the detailed configuration information – the queues hold the messages. The registry is a sophisticated component of MQe that can be configured to provide secure storage of MQe information.

Leave the *Windows Explorer* and return to the *MQe_Explorer* window.

In the list view pane there are other queue manager properties, some of which are blank (that is, not set) such as *Aliases*. Others are set – for example *Channel timeout*.

Properties of a simple queue manager

The details of the *FirstQM* queue manager can be investigated.

In the tree pane click on the ⊞ sign next to *MQe root*.

A sub node appears for the queue manager *FirstQM*.

In the tree pane click on the *FirstQM* node.

A ⊞ sign appears alongside indicating there are more elements below in the tree. The detail in the right hand pane gives more information.

Select *FirstQM* in the tree pane.

The right hand pane shows the detail associated with the selected element in the tree. Thus a queue manager has associated connections and queues. If a gateway queue manager were being managed then a *Bridges* folder would also be displayed.

Double click on *Local queues* in the list view pane.

The left hand pane shows the expanded tree with the *Local queues* node selected. The right hand pane shows that there are four local queues (*AdminQ*, *AdminReplyQ, DeadLetterQ and SYTEM.DEFAULT.LOCAL.QUEUE*[37]). Three queues have standard application queue icons like this:  . The icon for the *AdminQ* is:  , colored to indicate that this is a rather special queue because of its admin implications and to ensure that it is not unintentionally modified. Other types of queues will be seen later with different associated icons. The *AdminQ* is the one to which admin messages must be sent. MQe_Explorer will send to this queue when it wants to query the *FirstQM* queue manager or to change the queue manager configuration. *AdminReplyQ* is used to receive the results of admin requests; *DeadLetterQ* is a queue typically used to hold messages that cannot be otherwise delivered.

Scroll the list view pane to see the properties of queues.

Queues have many properties – a *Description*, *Current depth*, *Type*, *Mode* etc. *Type* - indicates the nature of the queue, *Mode* – how it is accessed, etc. These properties are shown as snapshots, taken at the time when MQe_Explorer first accessed them – to get a current view the display must be refreshed (either implicitly or explicitly) – this will be done later after changes have been made.

---

[37] This strangely named queue is provided for compatibility with MQSeries queue managers. By default, all queue managers (i.e. both MQe and MQSeries) will have such a queue available. If not required it can be deleted, alternatively it need not be created by default (see *Tools→Options→Advanced-1*.

© Copyright IBM Corp. 2000, 2001, 2002

Many of the properties are not set and it is convenient to inhibit the display of such empty columns.

Click on the button (*Hide Empty columns*) on the toolbar and then scroll the right hand pane.

Now only columns that contain useful information are displayed. Columns can also be re-ordered.

Drag the column heading of *Type* left into the middle of *Description*.

The columns have been re-ordered. Column re-ordering is only a temporary operation. Column width changes however are permanent for a given object. Full control of column display, both order and width, is available through the *Tools* menu.

Expand the width of the *Type* column so that all the data can be seen. Click once on the *Type* column header, and then click again.

Clicking on a column header sorts the data in that column; clicking again sorts in the reverse order.

Repeat the operation on the *Name* column.

In the right hand pane right click on the *AdminReplyQ* row. In the sub-menu select the *Properties* item.

In the property window for the *AdminReplyQ*, the title indicates that this shows the local details of a queue called *AdminReplyQ on the FirstQM* queue manager. The various tabs display its properties, grouped logically into *General*, *Properties*, *Storage*, *Security* and *Aliases*.

Click the various tabs, ending up back on the *General* tab.

Many properties are grayed out meaning that they are not relevant to a queue of this type – thus, for example, a local queue does not have a queue queue manager property (such a property is valid for a remote queue). Other properties are read-only indicating that they may not be changed – in this case, for example, no properties may be changed on the *AdminReplyQ* queue whilst it is being used by MQe_Explorer to receive admin replies – normally local queues are not so restricted.

Other queues have different properties, for example the *DeadLetterQ*.

Click the *Close* button. In the right hand pane right click on the *DeadLetterQ* row. In the sub-menu select the *Properties* item. Explore all the tabs and finish back on the *General* tab.

This shows a typical local queue – parameters such as *Maximum message length* and *Maximum queue depth* can be changed (to enter a number the *No limit* box must be unchecked). For example, to change the *Description*:

Click on the *Description* field and edit the value to "Default Dead Letter Queue". Click the *Apply* button.

After a brief pause the cursor changes back to the default style to indicate that the operation is complete. Looking in the list view pane of the main MQe_Explorer window, the description of the *DeadLetterQ* has also changed there – a refresh of the display has occurred. Such refreshes are implicit when an object property is changed.

Many objects in MQe can have alias (alternative) names in addition to their base name – again, the property pages are used to create aliases, for example, for the *DeadLetterQ*.

© Copyright IBM Corp. 2000, 2001

Click on the *Aliases* tab, type the string "DLQ" in the typing area, and click the *Add* button. Then click the *Apply* button. In the list view pane scroll so that the *Aliases* column is visible.

The alias name has been added to the queue properties.

There is another way of displaying information in the right hand pane of the main window that makes alias names more conspicuous. Instead of a row per object, it is possible to have a row per object-name pair.

In the tool bar click the button (*Use extra rows*). Click the *Close* button on queue properties dialog to remove the window.

An extra row now appears in the list view pane – with a variant of the queue icon . This new icon indicates a shortcut to that queue, that is, an alias. As before, the list can be sorted by name.

Click the *Name* column heading a few times to demonstrate sorting by name.

Queues normally contain messages. Drilling down a level enables messages to be seen.

Double click the *AdminReplyQ* node in the list view pane.

The list view pane is now empty and the status bar indicates that there are no messages. This is not too surprising since no messages have been explicitly created (although MQe_Explorer has been using messages itself). However an option exists for displaying the messages being used by MQe_Explorer.

Click on *Tools* in the menu bar, and then click on *Demo Mode*.

The effect of demo mode is that MQe_Explorer will make a copy of all admin messages sent and received and then put these copies in the *AdminReplyQ*. In order to see the effect of this we must cause admin messages to be issued (and then copied); manually refreshing a queue will have this effect. However for other reasons, in this case we want to do something that changes a queue property.

In the tree pane right click on the *DeadLetterQ* node. In the sub-menu select *Properties* and on the second tab increase the *Priority* by one. Click *Apply* then *Close*. Then in the list view pane right click on the *AdminReplyQ* and select *Refresh*. Double click the *AdminReplyQ* row to show the messages.

Six messages appear in the right hand pane. The first is the priority change request, the third an "Inquire all" on the *DeadLetterQ*, the fifth an "Inquire all" on the *AdminReplyQ*. Messages two, four and six are the respective replies.

Scroll the right hand pane to see the properties.

There are many properties, such as the *Origin queue manager* (with a value "FirstQM" as we might expect). There is a *Creation date* and *Creation time*, *Class*, *Number of fields* (in this case 19), *Correl id* etc. Although the display claims the message has 19 fields there are not 19 displayed – the *Hide Empty columns* option is causing some columns to be suppressed.

Move the cursor into the tree pane.

The hover help in the tree pane shows the class of the queue; a similar display occurs with messages. When a message (or fields object within a message) is selected in the tree pane, the hover help shows the class of the associated object.

© Copyright IBM Corp. 2000, 2001, 2002

In the toolbar re-click on the *Hide Empty columns* button (so that it is no longer depressed).

More columns appear – mainly to do with various security settings for a message. What is being shown at this level are various properties of the message itself (such as *Creation date*, *Class* etc.) and a number of common fields that messages might be expected to contain (such as *Priority*, *Correl Id* etc). Of course messages are not obliged to have any fields at all, apart from the unique identifier that MQe adds. That unique identifier is actually displayed as the two columns *Originating queue manager* and *Creation time*. In order to make the time display more useful a third column is also generated called *Creation date* – but this is only a reformatting of the value in the *Creation time*.

The column *Message* contains an icon and a number. That number is not present in the message – it is a number assigned by MQe_Explorer itself to the message to make it memorable. Strictly it relates to the copy MQe_Explorer has of the message – a snapshot of the queue that has been taken. In an active system, where multiple applications are running, the original message may already have been removed from the queue – however if the display kept up with reality management would be impossible. MQe_Explorer therefore keeps copies of messages so that they can be displayed or manipulated.

There is another way of looking at messages on the *AdminReplyQ* that only applies to this particular queue. The view is part of the off-line administration support. When objects are off-line at the time of administration the results will come back at some future time – the off-line administration support provides a means to track these actions. Although in this case we are not dealing with off-line administration, the provision allows for the inspection of any admin messages on the *AdminReplyQ* (excepting those that are only property enquiries or their replies). Select the menu item *View→Off-line Admin*.

A new window appears with no messages present. Maximise to full size. On its toolbar click the button; "other" (i.e. non off-line) admin messages present on the *AdminReplyQ* will then be added to the display.

Two messages are shown. The first is an "Update" request on the *DeadLetterQ*; the second is the resultant successful reply. Double clicking the request shows more details of the request – the priority change can be seen; double clicking the reply shows details of any errors that occurred – in this case no errors are present. The column *Request time* shows the time the original off-line admin request was made – in these cases there is no actual off-line request time to display and so the message creation time is displayed instead – hence the times shown are approximate and the "~" prefix is used to indicate this. However the times correlate exactly with the *Creation time* field in the corresponding message view that can be seen in the main display window.

Double click the two rows in turn to see the detailed information – then close the detailed windows. Then close the off-line display window and return to the main MQe_Explorer window.

One very powerful feature of the MQe_Explorer arises from the fact that MQe messages are objects with an internal structure. Drilling into a message can show this structure.

Double click the fourth message in the list view pane.

Now the first level of message content is displayed. There are 19 rows corresponding to the 19 fields.

© Copyright IBM Corp. 2000, 2001

Scroll horizontally (and vertically if necessary) to see all the data presented.

There is a lot of information presented. Firstly the *Field* names – some are prefixed with an * such as **Msg_CorrelI*. This notation indicates that the actual name is in the message is an (unprintable) coded value that corresponds to a defined constant; MQe_Explorer displays the constant's name prefixed with an *. Encoded names are typically used to save space in messages and standard codes exist for all common field names.

Many field names can be (optionally) translated to more meaningful strings. For the names that occur in admin messages MQe_Explorer is able to provide a more meaningful interpretation – click the button  on the toolbar. The name *admact* now appears as 'Action', in quotes to indicate that a translation has taken place; likewise **Msg_CorrelId* becomes 'Correl. Id'.

The message displayed is a copy of an admin message – the reply to an "Inquire all" request used to get the details of the that we saw earlier on the *AdminReplyQ*. The value of the *Action* field tells admin that this is a query (as opposed to a delete, update, etc.), *Max attempts* says how many times the operation should be attempted (here just once), *Target queue manager* is the name of the queue manager to which the request is directed, and so on.

Extend the application window vertically so that no scrolling is required to see all 18 rows.

There are two interesting rows – 'Original correl. Id' and 'Original UID'. MQe_Explorer created the copy from the original message but the original UID could not be duplicated because MQe does not allow duplicate UIDs – so MQe_Explorer saved the original value in a new field so that its value was not lost. The copy preserved the class of the message. The 'Correl. Id' was changed because this is frequently used to identify particular messages (indeed MQe_Explorer uses it in this way) and leaving the original value would cause application confusion. However the original values are there if needed, albeit under a new field name. The 'Copy info' field is added to record the details of the copy operation.

Click the  button from the left on the toolbar (*Up one level*) and then go back to the message details by double clicking the message.

Then click the *Data type* column heading.

Various data types are visible, *ASCII* fields, *Byte*, *Integer*, *Fields* fields and *UNICODE*. They all have a value – except *Fields* fields (which will be investigated later), a length and some other properties. Some are arrays – such as the 'Correl. Id' – the array of bytes data type is chosen to maintain compatibility with other MQSeries products. 'Correl. Id' is a static array that is it has fixed bounds; other arrays can be dynamic will flexible bounds. Other properties exist – the more interesting of which are the security attributes (such as *Compressor*, *Authenticator* etc. that can be applied to a *Fields* field and used to protect component elements of a message).

A basic message is itself a *Fields* object. The *Fields* field within a message represents a nested element. Double clicking can expand it.

Double click the 'Parameters' row.

This fields object is used to hold the detailed parameters of an admin message. Now all the properties of the *AdminReplyQ* queue are visible – thus the field *Queue message priority* gives the priority, the field *Queue/forward queue manager name* the queue manager name that owns the queue – and so on.

© Copyright IBM Corp. 2000, 2001, 2002

Move the spacer bar between the left and right hand panes to the right so that the full tree structure in the left hand pane is visible. Click on the 'Original UID' row.

The right hand pane now displays the UID of the original message before a copy was made. The UID has been formatted for display as an originating queue manager name and a time.

At this point it is appropriate to exit demo mode - otherwise too many messages will be collected.

Click the *Tools* menu and disable *Demo Mode*.

Creating a new queue

A new queue can easily be created.

Right hand click the *Local queues* node in the tree view. Select the *New queue* menu item. A new queue definition window appears. Rearrange the various MQe_Explorer windows on the screen so that they are all fully visible.

MQe_Explorer is not modal. Although we are in the in the middle of creating a new queue – other objects can still be explored, as previously.

Click around the original window to show it is still active. Return to the new window.

Queues have many properties – and these depend upon the nature of the queue. Initially we will create a local application queue.

Type the string "MyQ" into the *Name* field and "A new queue" into the *Description*. Explore the drop-downs on all the tabs. Finish with the *General* tab showing.

Note that mode and many other properties are not available – local queues do not office a choice of mode. Simple default properties can be changed as follows.

On the *Storage* tab uncheck the *No limit* against *Max depth* and set the *Max depth* value to 10. On the *Properties* tab use the spin button to increase the *Priority* property to 6. Click the *Create* button.

Note that the window stays in case another queue is to be created. In the main MQe_Explorer window the new queue is evident – both in the left and the right hand panes. In the right pane it is apparent that it has the custom properties set. A test message can be sent to the new queue to confirm that it is operational.

Right click the *MyQ* row in the right hand pane and select the *New message* menu item. A new window appears.

Everything is already set up for a default message - clicking *Send* would send the message. However there are useful options to explore.

Explore the drop-downs etc.

A custom message could be sent, with a user-selected field and /or encoding, to a different queue manager target (if there were any), and to different queues etc. Note that queue manager and queue names can be entered – in addition to those available from the drop-down list (this allows for the use of alias names, as well as for dynamic discovery of queue details in synchronous messaging). In our case however, a default message can be sent.

© Copyright IBM Corp. 2000, 2001

Click the *Send* button. A dialog appears with information on the message sent – click *OK*.

To see if the message arrived:

Right click the *MyQ* row in the list view pane and select *Refresh*. The message appears.

As expected the message is there. To see its contents double click it.

Double click the message row. The contents appear – explore them.

The contents are as expected. More messages can be sent.

Click the *Send* button on the *Send test message* dialog. Then click *OK* on the resulting message box. Repeat the set of operations once more.

In the main window, tree view pane, nothing new is visible.

On the *MyQ* node in the tree pane, right click, and then select the *Refresh* sub-menu item. Both panes are updated – and three messages exist.

The difference in behavior here is that the after the first message was sent MQe_Explorer had not taken a snapshot of the queue contents. When it needed the contents of *MyQ* it took a snapshot at that time – and hence it found the test message. Subsequently, it did not have a reason to look again (which might have been the desired behavior) – however when a refresh was explicitly requested, a new snapshot was obtained and the three messages were discovered.

Messages can be deleted, copied and moved.

Select the first message in the right hand pane and then hit the delete key. Click *Yes* on the confirm dialog – the message is deleted. Select the message 2 and then with shift held down select message 3. Both messages are selected. Do a drag copy operation (the Control key down with a mouse select) and drag the messages from the right hand pane into the tree pane – and drop on the *DeadLetterQ*. Whilst over other nodes the drop will have been disabled – but is permitted over valid drop targets such as the *DeadLetterQ*. Click the *DeadLetterQ* in the tree view pane. The copies of the messages are visible in the list view pane.

These copied messages are not identical to the original messages because they do not have the same UID. Messages can be moved in a similar way, and can also be cut and pasted via the clipboard.

Using the menu item *Window→Close All*, close all windows except the main MQe_Explorer window.

Creating a connection

It is more interesting to experiment when there are multiple queue managers. A second queue manager called *SecondQM* will be created (on the same machine) and the two queue managers will then communicate with each other.

Firstly *FirstQM* needs information on *SecondQM*; this is provided by defining a new connection (sometimes known as a remote queue manager definition).

© Copyright IBM Corp. 2000, 2001, 2002

In the tree pane right click the *Connections* node. Select the *New Connection* menu item. A window appears. Explore the tabs finishing back on *General*.

The name of this connection is *SecondQM* since that is the name of the remote queue manager. The description can be any ASCII string. For all other properties on this tab the defaults will be used. *SecondQM* will be defined as a second queue manager running on the same machine as *FirstQM*.

Enter "SecondQM" into the *Name* field and "Second queue manager" into the *Description*. Leave the *Type* field with the default value of "Direct connection". Move to the *Primary* tab.

The *Primary* tab describes the first of the connection adapters to be used[38]. Many defaults can be taken, but the basic protocol and addressing information must be specified.

In the *IP address* field enter the local host address of *SecondQM* (either the numeric address 127.0.0.1 or the name "localhost" can be used). In the *IP port number* field enter the port number 8083 which will be listened on by *SecondQM*. Leave all other fields with their default values. Click the *Create* button. Changes appear in both panes of the main window.

In the tree pane a new queue manager node has appeared – *SecondQM* – equivalent in the hierarchy to *FirstQM*; however it has a modified icon alongside that indicates that MQe_Explorer has not yet been able to contact it. The MQe network now comprises two queue managers. Moreover the *MQe root\FirstQM\Connections* node now has a ⊞ sign alongside to indicate more data is available at a lower level. The list view pane shows the details of the new connection.

Double click the row in the right hand pane.

The pane is empty and the status bar indicates that there are zero queues. No remote queues on *SecondQM* have been defined yet on *FirstQM* and so this is correct. More interesting is the *SecondQM* node in the tree pane directly under root.

Click the *SecondQM* node (the last node in the tree in the left hand pane). A ⊞ sign appears alongside and the right hand pane shows the next level of structure. Click again and the tree expands one level. Then click on any of the resultant folders.

The status bar displays "No information" – an indication (again) that *SecondQM* cannot be contacted. This does not mean that it cannot be managed – off-line administration capability permits MQe_Explorer to asynchronously send messages to remote queue managers that are unavailable at the time.

Right click the *Local queues* folder under the *SecondQM*.

The context menu has a number of commands for dealing with queues on *SecondQM*; similarly the *Connections* folder would have offered connection-related options. In fact neither *FirstQM* nor *SecondQM* is fully set-up at this stage for off-line administration because an asynchronous remote admin queue has not been defined, nor has a connection back from *SecondQM* to convey the results of any operations. At this point it is appropriate to create the *SecondQM* queue manager.

---

[38] In a future release of MQe, multiple adapters will be able to be specified for use in a single connection definition – for example, RAS services along with TCP/IP. Currently only the primary adapter is permitted.

© Copyright IBM Corp. 2000, 2001

Using the *Windows→Close All* command close all windows except the main MQe_Explorer window.

Bringing up a second queue manager

Double click the MQe_Explorer desktop icon. A second MQe_Explorer window appears. Click the button – the create queue manager dialog appears. Using the same steps as before, create a new queue manager *SecondQM* (as before leave the *Server or client/server* check box checked). Ensure the following properties are set (all others should be defaulted):

***QMgr. name*: SecondQM**
***Path*: C:\Program Files\MQe\Java\MQe_Explorer\ (i.e. as for *FirstQM*)**
***IP address*: 127.0.0.1 (i.e. as for *FirstQM*)**
***IP port number*: 8083**

and click *Create*.

Before the queue manager is created a message will be issued giving the name and location of an initialization file. This time it is useful to examine the contents of such files.

Bring up *Notepad* (*Start menu→Programs→Accessories→Notepad*). Open the file *C:\Program Files\MQe\Java\MQe_Explorer\SecondQM.ini* (you will need to change the file types displayed to see the file in the directory).

The contents of this *.ini* file are very similar to the *FirstQM.ini* file that was created earlier. Much of the content is due to the defaults that have been taken. Notice that it contains information on where the registry is located and the name of the queue manager. This information is used whenever the queue manager is restarted. The other content is described in the MQe product publications; MQe_Explorer itself uses the *[MQe_Explorer]* section when configuring remote queue managers.

Close *Notepad*. Arrange the two MQe_Explorer windows side by side so that the contents of each are equally visible.

Now two queue managers running on a single machine, each in their own JVM. Likewise there are two MQe_Explorers running as well – only one of which is needed for admin (either could be chosen). Other applications besides MQe_Explorer could be launched into these JVMs – the menu command *Tools→Load* will load classes but that goes beyond the scope of this script. The two MQe_Explorers can be distinguished by the name of their local queue manager, shown in the bottom right panel of the status bar, and also by the string shown in the associated button in the Windows task bar. Although *FirstQM* has all the information it needs to talk to *SecondQM*, the converse is not yet true. We could use MQe_Explorer on *SecondQM* to configure that queue manager with a connection definition for *FirstQM*, in the same way that *FirstQM* was configured. An alternative way is to let *FirstQM* remotely configure *SecondQM*. Not only will we set up a connection definition but we will set up various queue definitions and aliases as well.

On *FirstQM* ensure that the queue manager *SecondQM* is selected in the tree. Then click the menu item *Tools→Configure Remote*. A warning message appears.

You can ignore this warning. You are configuring a remote server.

Click *Yes*. A window appears with two tabs available.

All the information needed for default remote configuration is already loaded in the input fields. In this case we will configure *SecondQM* such that it can be managed in both on-line and off-line mode.

© Copyright IBM Corp. 2000, 2001, 2002

Select the *Remote qMgr.* tab (where information relating to *SecondQM* is displayed) and check the *Off-line* box; then click the *Apply* button. The progress bar tracks the changes until a completion message is issued. There will be a delay of a few seconds about a third the way through the process – this is a timeout parameter checking that the remote queue manager is not already communicating. When the completion message appears click *OK*. Then *Close* the remote configuration window

There are many changes. The *MQe root\FirstQM\Connections\SecondQM* node is selected and two remote queues are displayed in the list view pane. *AdminQ* is a synchronous remote queue; *Off-lineAdminQ* is an asynchronous remote queue. Both of these queues map to the *AdminQ* of *SecondQM*. Moreover, the icon for *MQe root\SecondQM* will have changed from  to  showing that *FirstQM* can now talk to *SecondQM*.[39]

On the *FirstQM* queue manager expand the node *MQe root\SecondQM* and then continue expanding the tree until the node *MQe root\SecondQM\Local queues\AdminQ* is selected. Right click this queue and select the *Properties* menu item. In the window that appears, select the *Aliases* tab.

The *AdminQ* on *SecondQM* now has an alias name *Off-lineAdminQ*. Other changes are apparent on *SecondQM* and they can be seen either directly (through its own MQe_Explorer) or remotely. We will continue to manage *SecondQM* remotely.

On FirstQM expand the node MQe root\SecondQM\Connections\FirstQM.

SecondQM now has a connection definition to *FirstQM* and an associated remote queue definition to the *AdminReplyQ*.

On FirstQM select MQe root\SecondQM\Connections\FirstQM.

A remote queue is seen in the list view pane. Scrolling the pane reveals that this has an asynchronous mode. It is this queue definition that is used (and is being used) to deliver responses back from *SecondQM* to *FirstQM*. It is asynchronous because we requested both a synchronous and an asynchronous admin capability from *FirstQM* to *SecondQM* – if we had just wanted synchronous, the MQe_Explorer would have not configured the *OfflineAdminQ* on either queue manager, nor would it have set up the alias, and this queue would have been synchronous.

To demonstrate that the queue managers can exchange messages, a test message can be sent from *FirstQM* to *SecondQM*.

On *FirstQM*, right click *MQe root\SecondQM\Local queues\DeadLetterQ* and choose *New message*. Taking all the defaults, click the *Send* button. A confirmation message appears; click *OK*.

On SecondQM, expand the tree and select the node MQe root\SecondQM\Local queues\DeadLetterQ.

The list view pane shows the message just sent from *FirstQM*. It can be drilled into as demonstrated previously. The message could also have been seen from *FirstQM* by selecting *MQe root\FirstQM\Connections\SecondQM\DeadLetterQ* on that queue manager (a *Refresh* may be needed – depending upon whether it has previously been accessed).

The tree structure on *SecondQM* does not feature *FirstQM* – that is because the tree has not been refreshed after the remote configuration.

---

[39] In the event of the icon not changing automatically, just refresh *MQe root\SecondQM*.

© Copyright IBM Corp. 2000, 2001

Select *MQe root* and click the refresh button on the tool bar.

FirstQM now appears in the tree, initially with an unavailable queue manager icon – but this will change automatically to an available queue manager icon after *FirstQM* and *SecondQM* have exchanged information.

Advanced queue properties

Now that a remote queue manager exists, queues can be re-visited and more advanced queue types explored.

On *FirstQM*, in the tree pane right click on *MQe root\SecondQM\Local queues* and select the *New Queue* item.

The new queue dialog is displayed, primed to (remotely) create a local queue definition on *SecondQM*. A *store & forward queue* can be defined. Imagine that two additional remote queue managers *ThirdQM* and *FourthQM* exist. Messages are to reach them by being sent to *FirstQM*. A *Store & forward queue* can be configured to handle this situation.

Select *Store & forward queue* in the *Type* field, enter "StoreForwardQ" in the *Name* field and "Store & forward queue" in the *Description* field. In the *Target qMgr* field enter (or select from the drop-down list) "FirstQM". On the *Destinations* tab type "ThirdQM" and click *Add*. Then type "FourthQM" and click *Add*. Then click *Create*.

In the tree pane there is a new node under the connection to *FirstQM*, i.e. *MQe root\SecondQM\Connections\FirstQM\StoreForwardQ* with a new icon  . This indicates a local[40] queue that gathers messages for the target queue managers and passes them on to *FirstQM*. It is assumed that *FirstQM* has connection definitions for these two new queue managers; alternatively it could have another store & forward queue that passed them on elsewhere.

Similarly a store queue can be defined that gathers messages but does not forward them. Typically such a queue would hold them ready for them to be pulled by a *Home server queue*.

Re-using the new queue dialog, ensure *Store & forward queue* is selected in *Type* field, edit the *Name* field to "StoreQ" and the *Description* field to "Store queue". In the *Target qMgr* field enter (or select from the drop-down list) "SecondQM". On the *Destinations* tab type "FifthQM" and click *Add*. Then click *Create* and then click *Close*.

The new queue appears as a local queue with a new icon  . A matching *Home server queue* could now be created on the queue manager *FifthQM* – if such a queue manager had been defined. Instead, we will pretend that this store queue also holds messages for *FirstQM*. Then on *FirstQM* the *Home server queue* would be defined as follows.

On *FirstQM* select the node *MQe root\FirstQM\Connections\SecondQM* and right click to define a new queue. In the *Name* field type "StoreQ", the *Description* is "Home server queue" and the *Type* is set to "Home server queue". Click *Create* followed by *Close*.

The new home server queue is visible along with other remote queues associated with that remote queue manager – and the associated icon is  . However, since this queue is not appropriate since it will never actually get any messages for *FirstQM*, it can be deleted.

---

[40] Although it is a local queue, it appears in the tree as a remote queue, indicating where it sends its messages.

© Copyright IBM Corp. 2000, 2001, 2002

Select the queue in either the tree or the list view pane and delete it by hitting the delete key. Click *OK* on the confirmation messages.

Although some interesting queue types have been configured they have not been used for moving messages. Likewise only synchronous messaging has been demonstrated. Further investigation is deferred to later scripts[41].

Queues have many useful properties – and all can be exploited by rules. The security attribute can be easily illustrated[42]. A queue will be created that is protected by an authenticator – in this case the sample NT authenticator that is supplied with MQe.

On *FirstQM* select the node *MQe root\FirstQM\Local queues* and bring up the *New queue definition* dialog. In *Name* type "ProtectedQ", in *Description* type "NT authenticator protected queue". On the *Security* tab select the *Authenticator* "examples.attributes.NTAuthenticator". Click *Create* followed by *Close*. The new queue is seen in the right hand pane. Double click the *ProtectedQ* row in that pane. A panel appears requiring a valid NT user id and password before access is permitted[43].

The queue is now protected through the example NT authenticator. In practice a custom authenticator would be used with an appropriate user interface – or some other way of ensuring the identity of the user.

The *Expiry* property is another useful attribute, ideal for preventing transient data filling queues with out of date information.

Using the *New queue definition* dialog: in *Name* type "TransientQ", in *Description* type "Transient queue". On the *Properties* tab put a value of 10000 (i.e. 10 secs.) in the *Expiry* property. Click *Create* followed by *Close*. The new queue is seen in the tree. Select the new queue and send it a test message using the right context menu. Refresh the queue content display using the  button in the toolbar – the new message will be seen. Refresh again a few seconds later – the message will have disappeared.

Messages only persist on the *TransientQ* queue for 10 seconds as specified – then they are deleted.

---

[41] Script *Basic messaging* on page 145 illustrates direct & indirect connections and synchronous and asynchronous messaging. Script *Advanced messaging* on page 153 illustrates pushing and pulling messages, use of intermediate queues and the details of home server and store & forward queues.
[42] This section cannot be run on Windows 98 systems – the sample MQe NT authenticator is inappropriate; nor can it be run unless MQe has been installed.
[43] Enter valid details here (*Domain name*, *User id* and *Password*) and, if set-up has been correct, access will be granted – note that if you are not part of a Windows 2000 domain (for example, running standalone) you must leave the *Domain* field with its default value intact. If you have failed to set-up Windows security correctly and you attempt to validate a user id/password you will get an Exception message – to recover click *Continue* and then click elsewhere in the left hand pane tree. If you know that you have not set-up Windows security then click *Cancel* in the authentication dialog – this will avoid the exception.

© Copyright IBM Corp. 2000, 2001

On the *Tools* menu select *Options* and explore the *Options* dialog.

The *Open qMgrs* tab provides control of the environment components to be present when MQe_Explorer loads a queue manager. The next 5 tabs allow control of the various list view panes – the columns seen and their respective order. The *Properties* tab is used to set the default widths that are used by default in the right hand list pane – subsequently the displayed widths can be dragged and are remembered on an object-by-object basis. The *Classes* tab controls the contents of list boxes and other items and hence allows MQe_Explorer to be usefully customized. The two *Advanced* tabs control the queues that are generated and/or used, and the number of threads allowed, timeouts etc.

Not shown in this script is off-line working nor the loading of other applications into a queue manager. Nor any exploitation of rules and many of the other advanced features of MQe, such as security.

Resetting the script

To reset the script to its initial state delete the following files and directories (and their associated contents and sub-directories):

C:\Program Files\MQe\Java\MQe_Explorer\FirstQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\FirstQM

C:\Program Files\MQe\Java\MQe_Explorer\SecondQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\SecondQM

© Copyright IBM Corp. 2000, 2001, 2002

13.2 Basic messaging

Overview

This script illustrates the simpler messaging network configuration and delivery options that are available with MQe. Familiarity with the operations detailed in the *Concepts and objects* script is assumed.

The following features are shown:

- Synchronous messaging (with both direct routing and indirect routing)
- Asynchronous messaging (with both direct and indirect routing) using:
  - *per queue* queuing (remote queue definitions).
  - *shared queue* queuing (store & forward queues).

Scenario (direct & indirect routing)

The script sets up the following configuration:

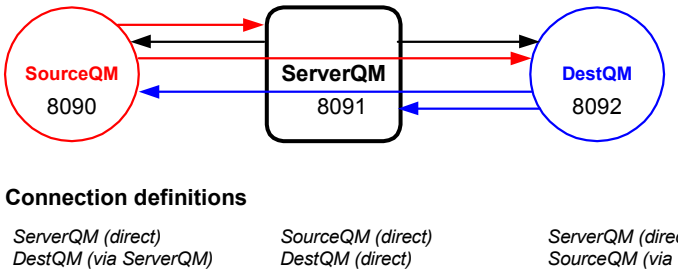


Figure 13-1: Basic messaging scenario

Three queue managers are used (*SourceQM*, *ServerQM* and *DestQM*) each of which can send messages to the others. The script generates direct connections as follows:

SourceQM to *ServerQM*

DestQM to *ServerQM*

ServerQM to *SourceQM*

ServerQM to *DestQM*

and indirect connections:

SourceQM to *DestQM* (via *ServerQM*)

DestQM to *SourceQM* (via *ServerQM*)

© Copyright IBM Corp. 2000, 2001

Script

Introduction

MQe supports both *direct* connections and *indirect* connections. A *direct* connection exists when one queue manager has a single transport link to another queue manager. An *indirect* connection means that no direct transport exists, but instead one or more intermediate queue managers must be traversed between the source and target queue managers. Indirect connections permit transport protocol changes to occur between the source and the target queue managers.

Creating the queue managers

Three queue managers are required: *SourceQM*, *ServerQM* and *DestQM*.

Click three times on the MQe_Explorer desktop icon and then create three new queue managers with the following characteristics.

1. ***QMgr. name*: SourceQM**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8090**
   ***All other boxes*: default values**
2. ***QMgr. name*: ServerQM**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8091**
   ***All other boxes*: default values**
3. ***QMgr. name*: DestQM**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8092**
   ***All other boxes*: default values**

Re-arrange the windows such that they are organised leftwards and stacked vertically in the order *SourceQM* (top), *ServerQM* (middle) and *DestQM* (bottom).

The default admin time-outs should be reduced to minimize delays in waiting for a response (note that this script frequently attempts to talk to queue managers that are unavailable).

© Copyright IBM Corp. 2000, 2001, 2002

On each of the queue managers, from the *Tools*/*Options* menu, on the *Options* property pages, display the *Advanced-2* tab. In *Object access* reduce the *Wait time for a response* to 3 seconds. Also enable polling, to ensure that objects that have not yet responded are re-contacted on a regular basis. In *Polling* tick *Enable polling* and set *Delay time* to 5 seconds. Click *Apply*, then *Close*[44,45].

Creating the connections

The direct connections should be created first. Unlike the first script we will not use remote configuration but will individually configure each queue manager. Using the respective instances of MQe_Explorer, create the following connections:

1. ***On queue manager* SourceQM:**

   ***Name*: ServerQM**
   ***Description*: Server queue manager**
   ***Type*: Direct connection**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8091**

2. ***On queue manager* ServerQM:**

   ***Name*: SourceQM**
   ***Description*: Source queue manager**
   ***Type*: Direct connection**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8090**

3. ***On queue manager* ServerQM:**

   ***Name*: DestQM**
   ***Description*: Destination queue manager**
   ***Type*: Direct connection**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8092**

4. ***On queue manager* DestQM:**

   ***Name*: ServerQM**
   ***Description*: Server queue manager**
   ***Type*: Direct connection**
   ***IP address*: 127.0.0.1**
   ***IP port number*: 8091**

Then create the indirect connections:

1. ***On queue manager* SourceQM:**

   ***Name*: DestQM**
   ***Description*: Destination queue manager**
   ***Type*: Indirect connection**
   ***Via qMgr*: ServerQM**

---

[44] This script (unusually) uses a shared configuration file across the three queue managers. Changes to the settings for the first queue manager changed the saved (but not loaded) values for the others – clicking *Apply* for all three will ensure that they each use the changed values. This shared usage is not good practice but is convenient here.

[45] These settings are suitable for this script; it is not intended to suggest that they are suitable for operational use.

© Copyright IBM Corp. 2000, 2001

2. ***On queue manager* DestQM:**

   ***Name*: SourceQM**
   ***Description*: Source queue manager**
   ***Type*: Indirect connection**
   ***Via qMgr*: ServerQM**

The three queue managers have been created and connections configured – and are now able to talk to each other. Available queue manager icons have replaced all the unavailable queue manager icons in the tree. In fact they have already exchanged information.

Expand the tree in the *SourceQM* instance of MQe_Explorer.

A detailed view of the network is available – and a similar view will be available from the other queue managers. A number of remote queue definitions have already been created automatically by MQe – these are a consequence of queue discovery occurring during MQe_Explorer's use of admin operations. Exactly what is visible depends upon what had been created at the time the tree was expanded – to get the current situation *Refresh* should be used. For example:

On SourceQM refresh nodes MQe root\SourceQM\Connections\ServerQM and MQe root\SourceQM\Connections\DestQM.

Two remote queue definitions (*AdminQ* and *AdminReplyQ*) are seen to have been created for each connection.

From this point, *SourceQM* will arbitrarily be used for the admin operations.

Minimize ServerQM and DestQM. On SourceQM refresh the tree from MQe root\DestQM, MQe root\ServerQM and MQe root\SourceQM.

A custom target queue can now be created and used to investigate the sending of messages, both directly or indirectly, and synchronously or asynchronously.

Creating the queues

Using MQe_Explorer on *SourceQM*, remotely create a local queue on *DestQM* with the following properties:

1. ***Name*: DestQ**
   ***Description*: Destination queue**
   ***Type*: Local queue**
   ***Aliases*: SyncDestQ, AsyncDestQ**

Using the toolbar, or *View* menu, set *Use multiple rows for aliases*. Three extra rows are now visible in the right hand pane (*DestQ*, *SyncDestQ* and *AsyncDestQ*).

DestQ is a standard application queue, its only slightly unusual aspect is it has two alias names[46] - these will be exploited in remote queue definitions on *SourceQM* to enable synchronous and asynchronous access to the same physical queue.

[46] The script could be re-worked to use just one alias name – but using two makes the steps more symmetrical and hence clearer.

© Copyright IBM Corp. 2000, 2001, 2002

Still using MQe_Explorer on *SourceQM*, create the following remote queue definitions (that is, right click on *MQe root\SourceQM\Connections\DestQM*).

1. ***Name*:** **SyncDestQ**
   ***Description*:** **Synchronous access to DestQ**
   ***Type*:** **Remote queue**
   ***Local qMgr*:** **SourceQM**
   ***Queue qMgr*:** **DestQM**
   ***Mode*:** **Synchronous**
2. ***Name*:** **AsyncDestQ**
   ***Description*:** **Asynchronous access to DestQ**
   ***Type*:** **Remote queue**
   ***Local qMgr*:** **SourceQM**
   ***Queue qMgr*:** **DestQM**
   ***Mode*:** **Asynchronous**

The new queues can be seen in the list view pane.

The queue managers are configured so that messages can be sent either synchronously and asynchronously from *SourceQM* to *DestQM*. They will go indirectly (that is, via *ServerQM* because that is the connectivity defined between *SourceQM* and *DestQM)*.

Direct and indirect, synchronous and asynchronous messaging

Using SourceQM, by right clicking on MQe root\SourceQM\Connections\DestQM\AsyncDestQ and MQe root\SourceQM\Connections\DestQM\SyncDestQ[47] send the following two test messages:

1. ***Contents*:** **A message sent asynchronously from SourceQM**
   ***Destination queue mgr*:** **DestQM**
   ***Destination queue*:** **AsyncDestQ**
2. ***Contents*:** **A message sent synchronously from SourceQM**
   ***Destination queue mgr*:** **DestQM**
   ***Destination queue*:** **SyncDestQ**

Both messages are successfully sent. To confirm that they arrived they can be inspected on their destination queue.

Collapse the MQe_Explorer tree on *SourceQM* and then expand *MQe root\DestQM\Local queues*. Refresh *MQe root\DestQM\Local queues\DestQ*. Display its contents in the right hand pane.

Two messages appear; the *Original* queue manager property shows that they were sent from *SourceQM*; the *Creation date* values correspond to the times they were sent.

It is interesting to note that the asynchronous message was sent immediately – that happened because the rules associated with the *SourceQM* queue manager and the definition of the queue *DestQ* on *SourceQM* specified this particular behavior. Alternate rules could have been in place that enforced different behavior, for example that asynchronous messages should only be transmitted immediately if they had a high priority set, that less important messages were transmitted off-peak, or when at least 10 were pending, … etc.

[47] Another way of synchronously sending a message to *DestQ* is to right click *MQe root\DestQM\Local queues\DestQ* – this will use the remote queue definition for *DestQ* on *SourceQM*.

© Copyright IBM Corp. 2000, 2001

Explore the contents of each message – the value row of the *Name* property identifies which was sent synchronously and which asynchronously.

Messages have now been sent both synchronously and asynchronously, in both cases going indirectly via *ServerQM*. At the same time all admin operations have been through the MQe_Explorer running on *SourceQM*; it has been communicating synchronously and directly to the relevant queue managers.

Asynchronous messaging has behaved identically to synchronous messaging in the script so far because all the queue managers (and hence all the connections) have been available. The power of asynchronous messaging can be illustrated by disabling the intermediate queue manager *ServerQM*.

Restore the window for *ServerQM* and close the queue manager; minimize the window. Now send a further two messages through the *SourceQM* MQe_Explorer. Right click on *MQe root\SourceQM\Connections\DestQM\AsyncDestQ* and *MQe root\SourceQM\Connections\DestQM\SyncDestQ* and send the following:

1. ***Contents*:** **Second asynch send from SourceQM**
   ***Destination queue manager*:** **DestQM**
   ***Destination queue*:** **AsyncDestQ**
2. ***Contents*:** **Second sync send from SourceQM**
   ***Destination queue manager*:** **DestQM**
   ***Destination queue*:** **SyncDestQ**

The asynchronously transmitted message is sent successfully; the synchronously transmitted message cannot be sent – because a channel cannot be established to the destination queue *DestQ*. It is interesting to locate the message that was sent.

On SourceQM attempt to refresh the properties of MQe root\SourceQM\Connections\DestQM\AsyncDestQ.

The queue's contents cannot be accessed because this queue intentionally forbids browse (amongst other operations). However properties of the remote queue definition can be inspected.

On SourceQM display the properties of MQe root\SourceQM\Connections\DestQM\AsyncDestQ.

The queue depth is now shown as one; this is the message waiting to be sent and currently held in the backing store.

Using the *Storage* tab, click in the *Path* value field and then move the cursor to the right to display the entire value (that is, *C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues*). Then (without closing the *Properties* page), using *Windows Explorer* show the contents of *C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues\DestQM\AsyncDestQ*.

A single file of type *.MQeMsg* represents the message.

The *ServerQM* queue manager can be re-activated so that communication between *SourceQM* and *DestQM* can be restarted.

© Copyright IBM Corp. 2000, 2001, 2002

In the MQe_Explorer instance that was previously used for *ServerQM*, restart the queue manager by opening the file *C:\Program Files\MQe\Java\MQe_Explorer \ServerQM.ini* and then click on *MQe root* in the tree. Minimize this window and then go back to MQe_Explorer on *SourceQM*; refresh *MQe root\DestQM\Local queues\DestQ*. Display its contents in the right hand pane.

No new message has arrived – in fact, if the property pages of the remote queue definition on *SourceQM* are refreshed, the queue depth on the *Storage* tab is still shown as 1[48].

Click the *Refresh* button on the *C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues\DestQM\AsyncDestQ* properties page and examine the queue depth.

The message appears to be stuck; in fact MQe is enforcing the queue manager and queue rules on *SourceQM* – the default rules state that messages are triggered for transmission when the queue manager starts up, or when a message arrives on a queue. In this case the queue manager had already started – and the message was already there. If another had been sent – then both would have been transmitted – this behavior is also determined by the rules.

An easy solution in this case is to trigger the transmission manually using the trigger button  on the toolbar.

Click the *Trigger* button on the toolbar – then refresh the properties page again – the queue depth is now zero. Click *Close*.

An alternative approach is to changing the queue manager rule. A new rule class *examples.mqe_explorer.QmRule*[49, 50] triggers message transmission every 5 seconds.

On the *SourceQM* queue manager change the queue manager rule (via the queue manager properties) to *examples.mqe_explorer.QmRule*.

The new rule is now in effect but the change in triggering behavior only happens when the queue manager is re-activated[51].

On *SourceQM* display the contents of *MQe root\DestQM\Local queues\DestQ* contents in the right hand pane (three messages).

The script *Advanced messaging* on page 153 can now be used with this same configuration to illustrate some of the more advanced network messaging options.

---

[48] Be aware that the depth is only displayed when the queue is in the *active* state. For demonstration purposes if you want to see what messages are queued at any time then look in the file system.
[49] If you are using *MQe_Explorer.exe* ensure that you have installed the new rule class shipped with MQe_Explorer before proceeding (install via *MQe_ExplorerC.exe*).
[50] This is a variation on the *examples.rules.ExampleQueueManagerRules* class provided with MQe.
[51] This queue manager rule only takes effect when the queue manager is activated – it would have been possible to write it in such a way that a queue manager restart was not required. In general, a rule change does not require a queue manager restart.

© Copyright IBM Corp. 2000, 2001

Resetting the script

Full reset

If the *Advanced messaging* script is not going to be used, then to reset this script to its initial state delete the following files and directories (and their associated contents and sub-directories):

C:\Program Files\MQe\Java\MQe_Explorer\SourceQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\SourceQM

C:\Program Files\MQe\Java\MQe_Explorer\ServerQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\ServerQM

C:\Program Files\MQe\Java\MQe_Explorer\ DestQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\DestQM

To restore the MQe_Explorer configuration file to its initial state click *Tools→Options→Reset*.

© Copyright IBM Corp. 2000, 2001, 2002

13.3 Advanced messaging

Overview

This script illustrates more advanced of the messaging network configurations available with MQSeries Everyplace (MQe). It is designed to follow on immediately from the previous script.

The following features are illustrated:

- Asynchronous messaging:
  - *message push* (remote queue definitions and store & forward queues)
  - *message pull* (home server queues)
  - *shared intermediate* queues
    - locally
    - within the network

Scenario

The configuration is almost identical to that of the previous script. As before, *SourceQM* is sending messages to *DestQM* via *ServerQM*. In this case however the indirect connections between *SourceQM* and *DestQM* are no longer needed for application messaging, however they are needed if network administration is conducted from either *SourceQM* or *DestQM*[52]:

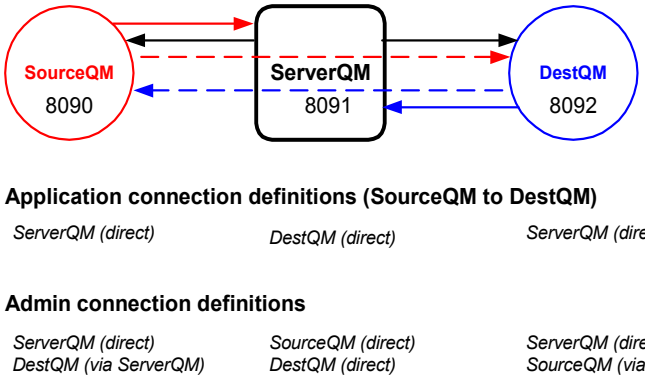


Figure 13-2: Advanced messaging scenario

Note: it is assumed that that the queue managers *SourceQM*, *ServerQM* and *DestQM* exist, along with their associated properties and queues.

[52] For simplicity the connection definitions are left unchanged from script 2. If the script is to be run without the indirect connections then it must be modified so that all admin is either conducted from *ServerQM* or each queue manager is administered directly from its local instance of MQe_Explorer.

© Copyright IBM Corp. 2000, 2001

Script

Introduction

MQe supports the use of intermediate queues (store & forward queues and asynchronous remote queues) so that messages can be queued within the network.

Store & forward queues hold messages bound for one or more queue managers – they are insensitive to the identity of the target queue on a particular queue manager. The first task in this script is to set up a store & forward queue on *SourceQM* that will hold all messages destined for *DestQ* on *DestQM* – previously they were held in the *SourceQM*'s remote queue definition for *AsyncDestQ* on *DestQM*.

Preparation

The queue managers require minor configuration changes. On *DestQM* all the messages sent previously to the *DestQ* on *DestQM* must be deleted.

Using *DestQM*, select *MQe root\DestQM\Local queues\DestQ* in the tree. In the right hand pane select a message, followed by *Control A*, and followed by the *Delete* key. Accept the deletion of all messages.

No messages must exist elsewhere – for example, on the remote queue definition for *AsyncDestQ* on *SourceQM*.

Using *Windows Explorer* show that the following directory is empty (and if not – delete any messages present):

C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues\DestQM\AsyncDestQ

In order for messages to move when possible the queue manager rules for *DestQM* and *ServerQM* require modification[53] – those for *SourceQM* have already been changed in the script *Basic messaging* on page 145.

Using *DestQM*, change its queue manager rule to *examples.mqe_explorer.QmRule* (the class is available on the drop-down if the default *Tools→Options* are in force – or use the *Reset* button to reset to the defaults). Close the queue manager. Do the same rule change to *ServerQM* and close the queue manager.

The basic configuration changes have been made. The queue managers can now be started.

Bring up *SourceQM* and *DestQM* followed by *ServerQM*. Arrange the windows on the screen so that they are stacked vertically, *SourceQM* at the top, followed by *ServerQM* and then *DestQM*.

Using a store & forward queue on the source queue manager

A store & forward queue on *SourceQM* can be defined that will collect messages bound for *DestQM* and send them to *ServerQM*. Store & forward queues can collect messages for more than one destination but in this case only one is needed. They do not need to forward them anywhere (this will be seen later) but here they are required to go to *ServerQM* – and not sent directly to the destination queue manager.

---

[53] If you are using *MQe_Explorer.exe* ensure that you have installed the new rule class shipped with MQe_Explorer before proceeding (install via *MQe_ExplorerC.exe*).

© Copyright IBM Corp. 2000, 2001, 2002

Define a queue on *SourceQM* with the following properties (right click on *MQe root\SourceQM\Connections\ServerQM*):

1. ***Name*:** **SFQ**
   ***Description*:** **Store & forward queue for DestQM**
   ***Type*:** **Store and forward queue**
   ***Local qMgr*:** **SourceQM**
   ***Target qMgr*:** **ServerQM**
   ***Destinations*:** **DestQM**

Now a message can be sent from *SourceQM* to the target queue *DestQ* at *DestQM*. Repeating previous practice the alias name for this queue will be used, that is *AsyncDestQ*.

Using MQe_Explorer on *SourceQM*, send a test message to *AsyncDestQ* at *DestQM* from *SourceQM* (right click on *MQe root\SourceQM\Connections\DestQM\AsyncDestQ*). Then refresh and display the contents of *MQe_root\DestQM\Local queues\DestQ*.

The message arrives as expected. However in order to see how it was routed *DestQM* needs to be taken off-line.

Close the *DestQM* queue manager (leave the window available) – then resend the message as previously. Using *Windows Explorer* inspect the contents of:

C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues\DestQM\AsyncDestQ

The new store & forward queue has had no effect – the message is waiting in the remote queue definition for *AsyncDestQ*. This is because behavior determined by a remote queue definition takes precedence over that arising from a store & forward queue definition. If the *AsyncDestQ* definition on *SourceQM* is deleted, the use of the store & forward queue will become apparent. However the *AsyncDestQ* queue cannot be deleted until it has delivered its message.

Using MQe_Explorer on *SourceQM*, attempt to delete the remote queue definition on *SourceQM* for *AsyncDestQ* at *DestQM* – an error results. In order to do this, bring up *DestQM* and then if necessary, recycle *ServerQM*. Then repeat the deletion attempt (the queue is successfully deleted). Take down *DestQM*.

Send a test message to *AsyncDestQ* at *DestQM* from *SourceQM* – to do this right click on any queue in the path *MQe root\DestQM\Local queues*, or in the path *MQe root\SourceQM\Connections\DestQM*. Then select *New*, *Message* and in the *Destination queue* box type in the queue name "AsyncDestQ"; click *Send*. Using *Windows Explorer*, display the contents of *C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues\ServerQM\SFQ*.

The message is now queued for transmission in the store & forward queue at the source queue manager. Another consequence of this change is that the properties of *SFQ* on *SourceQM* can now be used to determine the security attributes associated with transmission – in this case however none have been set.

© Copyright IBM Corp. 2000, 2001

Using a store & forward queue in the network

The second task in this script is to arrange for messages to be stored in a store & forward queue in the network (that is, on *ServerQM*) when *DestQM* is unavailable. The first step is therefore to define the new queue and then experiment sending messages.

Using MQe_Explorer on *SourceQM*, display the remote queue definitions on *ServerQM* at *DestQM*, that is, select *MQe root\ServerQM\Connections\DestQM*. If *AsyncDestQ* exists, delete it (note that it will be defined as a synchronous connection). Then define a queue on *ServerQM* with the following properties:

1. ***Name*:** **SFQ**
   ***Description*:** **Store & forward queue for DestQM**
   ***Type*:** **Store and forward queue**
   ***Local qMgr*:** **ServerQM**
   ***Target qMgr*:** **DestQM**
   ***Destinations*:** **DestQM**

Send a test message to AsyncDestQ at DestQM from SourceQM (as previously). Using the Windows Explorer, display the contents of C:\Program Files\MQe\Java\MQe_Explorer\SourceQM\Queues\ServerQM\SFQ (no messages) and C:\Program Files\MQe\Java\MQe_Explorer\ServerQM\Queues\DestQM\SFQ (two messages).

The message is now queued on the server – as is the message sent previously (now moved closer to delivery). If the server had not been available then the message would have been queued at the source, in the *SFQ* queue there. As previously, this only works because there is no remote queue definition for *AsyncDestQ* on *ServerQM* – if such a definition was present then the *SFQ* on *ServerQM* would not be used for any messages destined for *AsyncDestQ* on *DestQM*.

If *DestQM* is now brought up, it will now receive all the messages that are queued for it – this behavior occurs because queue manager rules were changed from the defaults to *examples.mqe_explorer.QmRule* – which requires that the queue manager attempt message delivery every 5 secs.

Re-load *DestQM*. In the right hand pane display the contents of *MQe root\DestQM\Local queues\DestQ*; then delete all the messages.

Using a home server queue on the target queue manager

Home server queues are often used in conjunction with store & forward queues – they provide an ability to pull messages from a server. This can be illustrated as follows – first the *SFQ* queue on *ServerQM* must be replaced with a new queue *StoreQ* that stores message for *DestQM* but does not forward them.

Using the MQe_Explorer on *SourceQM*, delete the *MQe root\ServerQM\Connections\DestQM\SFQ*. Create a new local queue on *ServerQM* with the following properties (right click on *MQe root\ServerQM\Local queues*):

1. ***Name*:** **StoreQ**
   ***Description*:** **Store queue for DestQM**
   ***Type*:** **Store and forward queue**
   ***Local qMgr*:** **ServerQM**
   ***Target qMgr*:** **(none)**
   ***Destinations*:** **DestQM**

Now a message can be sent from *SourceQM* to *AsyncDestQ* at *DestQM* – note that all three queue managers: *SourceQM*, *ServerQM* and *DestQM* are now active.

© Copyright IBM Corp. 2000, 2001, 2002

Send a message from SourceQM to AsyncDestQ at DestQM. Using Windows Explorer, show the contents of C:\Program Files\MQe\Java\MQe_Explorer\ServerQM\Queues\ServerQM\StoreQ (one message).

The message has reached the store queue – however it has not been forwarded to *DestQM* even though that queue manager is available. However if a home server queue on *DestQM* is defined it can be configured to pull messages from the store queue on *ServerQM*.

Using the MQe_Explorer on *SourceQM*, create a new local queue on *DestQM* with the following properties (right click on *MQe root\DestQM\Connections\ServerQM*):

1. ***Name*:** **StoreQ**
   ***Description*:** **Home server queue pulling from ServerQM**
   ***Type*:** **Home server queue**
   ***Local qMgr*:** **DestQM**
   ***Queue qMgr*:** **ServerQM**

Display the contents of MQe root\DestQM\Local queues\DestQ.

The message has been successfully pulled from the server.

Using remote queue definitions in the network

One advantage of using store & forward queues is that are queue manager-based, that is, they route based on queue manager name, rather than on the combination of queue & queue manager name. It is possible for messages to be routed and stored at intermediate queue managers in the network without involving store & forward queues through the use of remote queue definitions.

Below only remote queue definitions are used to exploit *ServerQM* as a message server for the target queue *DestQ* on *DestQM*. Firstly, delete all store & forward queues and the home server queue.

Using MQe_Explorer on *SourceQM*, refresh the views of *ServerQM* and *DestQM*. Then delete the following queues:

MQe root\SourceQM\Connections\ServerQM\SFQ

MQe root\ServerQM\Local queues\StoreQ

MQe root\DestQM\Connections\ServerQM\StoreQ

Create the following queues (using MQe root\SourceQM\Connections\DestQM and MQe root\ServerQM\Connections\DestQM:

1. ***Name*:** **AsyncDestQ**
   ***Description*:** **Asynchronous access to DestQ**
   ***Type*:** **Remote queue**
   ***Local qMgr*:** **SourceQM**
   ***Queue qMgr*:** **DestQM**
   ***Mode*:** **Asynchronous**

© Copyright IBM Corp. 2000, 2001

2. ***Name*:** **AsyncDestQ**
   ***Description*:** **Asynchronous access to DestQ**
   ***Target type*:** **Remote queue**
   ***Local qMgr*:** **ServerQM**
   ***Queue qMgr*:** **DestQM**
   ***Mode*:** **Asynchronous**

Using MQe_Explorer on *DestQM,* close the queue manager so that it is no longer able to receive messages.

Using MQe_Explorer on SourceQM, send a message from SourceQM to AsyncDestQ by clicking on MQe root\SourceQM\Connections\DestQM\AsyncDestQ.

Using Windows Explorer, show the contents of C:\Program Files\MQe\Java\MQe_Explorer\ServerQM\Queues\DestQM\AsyncDestQ (one message).

The message just sent is now queued on the server. This technique allows individual target queue-based security to be applied, but requires queue level definitions on the server – thus it is potentially more expensive in terms of network definitions. If *DestQM* is brought back up, the message will be delivered.

Reload DestQM. Show the contents of MQe root\DestQM\Local queues\DestQ. Using Windows Explorer, show the contents of C:\Program Files\MQe\Java\MQe_Explorer\ServerQM\Queues\DestQM\AsyncDestQ (no messages).

The script has demonstrated how MQe can queue messages at the source or within the network, how messages can be pushed, or alternatively pulled. It has also illustrated the implementation of queue manager- and queue-based routing.

Resetting the script

To reset the script to the state existing before the script *Basic messaging*, delete the following files and directories (and their associated contents and sub-directories):

C:\Program Files\MQe\Java\MQe_Explorer\DestQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\DestQM

C:\Program Files\MQe\Java\MQe_Explorer\ServerQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\ ServerQM

C:\Program Files\MQe\Java\MQe_Explorer\ SourceQM.ini

C:\Program Files\MQe\Java\MQe_Explorer\SourceQM

© Copyright IBM Corp. 2000, 2001, 2002

13.4 Gateway configuration and usage

Overview

This script demonstrate how to use the MQSeries Explorer management tool on an MQSeries queue manager, together with the MQe_Explorer on MQe, to exchange test messages between an MQSeries queue manager, an MQe gateway and an MQe server.

Scenario

Test messages are sent between all queue managers in a simple three queue manager network. The topology is a simple one, very similar to that described in the *MQe Programming Guide*, chapter 6, in the section *Administration from the command line*. The network comprises three queue managers:

BRANCH000QM – an MQe server.

GATEWAY00QM – an MQe gateway.

CENTRAL00QM – an MQSeries queue manager.

Message traffic is pushed from *BRANCH000QM* through the *GATEWAY00QM* to the *CENTRAL00QM*; message traffic is pulled from *CENTRAL00QM* by the *GATEWAY00QM*, and pushed on to the *BRANCH000QM*.

Note: The script uses upper-case names for the queue managers.

Script

Step 1: Create and configure the MQSeries queue manager

Use the *Start→MQSeries→MQSeries Explorer* to create an MQSeries queue manager called "CENTRAL00QM". Set the queue manager up so it listens on the default port 1414 (tick the *Create listener for TCP/IP* checkbox).

Step 2: Create the sync queue on the MQSeries queue manager

The *GATEWAY00QM* MQe queue manager (to be created later in this script) requires some MQSeries objects configured, in order to allow the MQe queue manager to talk to the MQSeries *CENTRAL00QM* queue manager. One such object is a *sync queue*. This is a local queue that must be persistent.

Using the MQSeries Explorer, create a local queue, called *SYNC.Q.FOR.GATEWAY00QM*. Allow both *put* and *get* of messages. Make the *Default Persistence* have the value "Persist", normal usage.

The sync queue is used by the MQe bridge to hold the persistent state information generated whilst moving messages between the MQe and MQSeries systems (this information is required for assured delivery).

Step 3: Create a server connection channel on the MQSeries queue manager

The bridge to MQSeries also requires a *server connection channel* object, as the MQe bridge may be configured to connect to the MQSeries system using the MQSeries client channel.

© Copyright IBM Corp. 2000, 2001

Use the MQSeries Explorer to create a server connection channel on the *CENTRAL00QM* queue manager:

***Channel Name*:**	**FOR.GATEWAY00QM**
***Transmission Protocol*:**	**TCP/IP**

Step 4: Create a transmission queue to MQe

In order to route a message from the MQSeries system to the *GATEWAY00QM* MQe queue manager, we also need to create a transmission queue. This transmission queue will not have an associated MQSeries channel; it must however be persistent. You could create more than one such transmission queue, if you wanted to partition the traffic heading to MQe using different routes, but one is enough in this example topology.

Messages bound for the *GATEWAY00QM* or *BRANCH000QM* will be directed to this transmission queue. Later, we will define an MQe bridge object, a *transmission queue listener*, which has the job of "pulling" the messages from this transmission queue and putting the messages onto the MQe network.

Using the MQSeries Explorer, create the local queue on the *CENTRAL00QM* queue manager as follows:

***Queue Name*:**	**TO.GATEWAY00QM**
***Put Messages*:**	**Allowed**
***Get Messages*:**	**Allowed**
***Default Persistence*:**	**Persist**
***Usage*:**	**Transmission**

Step 5: Create queue manager aliases to refer to the MQe queue managers

Now set up a remote queue manager alias such that the *BRANCH000QM* and *GATEWAY00QM* can both be sent messages, and that such messages are routed to the *TO.GATEWAY00QM* transmission queue.

Using the MQSeries Explorer on the *CENTRAL00QM* queue manager, create the following remote queue definition:

***Queue Name*:**	**GATEWAY00QM**
***Put Messages*:**	**Allowed**
***Default Persistence*:**	**Persist**
***Remote Queue Name*:**	***Leave this blank.***
***Remote Queue Manager Name*:**	**GATEWAY00QM**
***Transmission Queue Name*:**	**TO.GATEWAY00QM**

This special style of remote queue definition is known as a *remote queue manager alias*. It tells MQSeries to put any messages that are destined for the *GATEWAY00QM* queue manager to the *TO.GATEWAY00QM* transmission queue.

© Copyright IBM Corp. 2000, 2001, 2002

For the *BRANCH000QM* MQe queue manager, using the MQSeries Explorer on the *CENRAL00QM* queue manager, create the remote queue definition:

***Queue Name*:**	**BRANCH000QM**
***Put Messages*:**	**Allowed**
***Default Persistence*:**	**Persist**
***Remote Queue Name*:**	***Leave this blank***
***Remote Queue Manager Name*:**	**BRANCH000QM**
***Transmission Queue Name*:**	**TO.GATEWAY00QM**

Repeat for any other MQe queue managers on the MQe network to which you wish to send messages, for example, *BRANCH001QM*, *BRANCH002QM*, … etc.

Using remote queue manager aliases in this way requires only one remote queue definition per MQe queue manager, even if your application puts to six queues on the MQe queue manager. You could alternatively define exact remote queue definitions for every queue on the remote MQe queue manager if you wish, which would need six queue definitions per MQe queue manager. In this script, we use the remote queue manager alias method in order to keep things as simple as possible.

Step 6: Create BRANCH.SALES.QUEUE local queue on CENTRAL00QM

On The *CENTRAL00QM* queue manager, create a normal local queue called "BRANCH.SALES.QUEUE":

***Queue Name*:**	**BRANCH.SALES.QUEUE**
***Put Messages*:**	**Allowed**
***Get Messages*:**	**Allowed**
***Default Persistence*:**	**Persistent**

Later in this script we will later access this queue from MQe and put test messages to it.

Step 7: Milestone

You should now have an MQSeries queue manager ready for connection to your MQe queue manager. This completes the configuration of the *CENTRAL00QM* MQSeries queue manager.

© Copyright IBM Corp. 2000, 2001

MQSeries Explorer should show the following information:

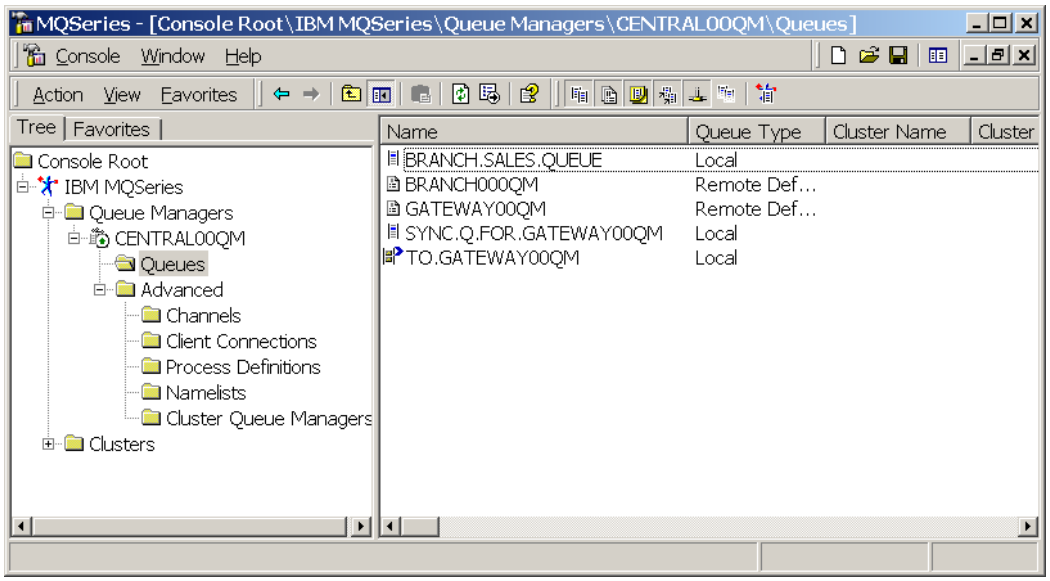


Figure 13-3: MQSeries Explorer view

Step 8: Create the *GATEWAY00QM* MQe queue manager

MQe_Explorer can create new local queue managers, which can be of type *client*, *peer*, *server* or *gateway*. Normally when creating such queue managers, the defaults can be used for most properties, however it is important to ensure that the IP address information is correctly set for incoming communications.

First however, check that the defaults are correctly set such that the necessary local queues will be created[54]. This script requires that the admin-related queues are present, as well as *SYSTEM.DEFAULT.LOCAL.QUEUE*.

Start MQe_Explorer and select the menu item *Tools→Options*. Using the scroll buttons, scroll to the Advanced-1 tab and select it. In the *New configuration – create queues* group box make sure that all four checked boxes are checked. If you have made changes, click the *Apply* button, then in all cases, click *Close*.

The gateway queue manager can now be created.

Using the MQe_Explorer, click on the menu item *File→New→Queue Manager* (or the equivalent toolbar button) and create a new MQe queue manager:

***General tab*:**
- ***QMgr name*:** **GATEWAY00QM**
- ***Description*:** **Gateway 00 queue manager**
- ***Type*:**
  - ***Server*:** **true**
  - ***Enable bridges…*:** **true**

***Comms tab*:**
- ***IP address*:** ***The IP address of your machine***
- ***IP Port*:** **8082**

---

[54] This check is only necessary if you have changed the defaults from the shipped values, or if you have used version 1.25 or earlier.

© Copyright IBM Corp. 2000, 2001, 2002

Step 9: Create the *BRANCH000QM* MQe queue manager

In a separate instance of MQe_Explorer, create your *BRANCH000QM* MQe queue manager:

***General tab*:**
- ***QMgr name*:** **BRANCH000QM**
- ***Description*:** **Branch 000 server queue manager**
- ***Type*:**
  - ***Server*** **true**
  - ***Enable bridges…*:** **false**

***Comms tab*:**
- ***IP address*:** ***The IP address of your machine***
- ***IP port*:** **8083**

Step 10: Tell the gateway queue manager about the MQSeries *CENTRAL00QM* queue manager

Using the *Gateway00QM* instance – create a new connection (right click on *MQe root\GATEWAY00QM\Connections* and select *New Connection*):

***General tab*:**
- ***Name*:** **CENTRAL00QM**
- ***Local qMgr*:** **GATEWAY00QM**
- ***Type*:** **Alias-only/MQ connection**

Step 11: Tell the gateway queue manager about the branch queue manager

Using the *Gateway00QM* instance – create a new connection (right click on *MQe root\GATEWAY00QM\Connections* and select *New Connection*):

***General tab*:**
- ***Name*:** **BRANCH000QM**
- ***Local qMgr*:** **GATEWAY00QM**
- ***Type*:** **Direct connection**

***Primary tab*:**
- ***IP address*:** ***The IP address of your machine***
- ***IP port*:** **8083**

© Copyright IBM Corp. 2000, 2001

Step 12: Tell the branch queue manager about the gateway queue manager

Using the *BRANCH000QM* instance – create a new connection (right click on *MQe root\BRANCH000QM\Connections* and select *New Connection*):

***General tab*:**

***Name*:**	**GATEWAY00QM**
***Local qMgr*:**	**BRANCH000QM**
***Type*:**	**Direct connection**

***Primary tab*:**

***IP address*:**	***The IP address of your machine***
***IP port*:**	**8082**

Step 13: Tell the branch that it can get to *CENTRAL00QM* via *GATEWAY00QM*

Using the *BRANCH000QM* instance – create a new connection (right click on *MQe root\BRANCH000QM\Connections* and select *New Connection*):

***General tab*:**

***Name*:**	**CENTRAL00QM**
***Local qMgr*:**	**BRANCH000QM**
***Type*:**	**Indirect connection**

***Primary tab*:**

***Via qMgr*:**	**GATEWAY00QM**

Step 14: Create an MQ bridge object on the gateway MQe queue manager

Using the *Gateway00QM* instance – right click on the *Bridges* node in the tree for the *GATEWAY00QM* queue manager (i.e. *MQe root\GATEWAY00QM\Bridges*) and select *New MQ Bridge*. Using the *New MQ Bridge* panel, create the bridge:

***Name*:**	**MyBridge**
***Local qMgr*:**	**GATEWAY00QM**

Step 15: Create an MQ queue manager proxy for the *CENTRAL00QM* queue manager

Note that the name of the proxy object matches the name of the MQSeries queue manager to which it refers.

Using the *Gateway00QM* instance – right click on the *MyBridge* node in the tree for the *GATEWAY00QM* queue manager (i.e. *MQe root\GATEWAY00QM\Bridges\MyBridge*) and select *New MQ QMgr. Proxy*. Using the *New MQ QMgr. Proxy* panel, create the proxy:

***Name*:**	**CENTRAL00QM**
***MQ Bridge*:**	**MyBridge**
***Local qMgr*:**	**GATEWAY00QM**

Step 16: Create an MQ client connection

Note that the name of the MQ client connection matches the name of the server connection channel we created earlier on the MQSeries queue manager.

© Copyright IBM Corp. 2000, 2001, 2002

Using the *Gateway00QM* instance – right click on the *CENTRAL00QM* proxy node in the tree for the *GATEWAY00QM* queue manager (i.e. *MQe root\GATEWAY00QM\Bridges\MyBridge\CENTRAL00QM*) and select *New MQ Client Conn*. Using the *New MQ Client Connection* panel, create the client connection:

***General tab*:**

***Name*:**	**FOR.GATEWAY00QM**
***Proxy*:**	**CENTRAL00QM**
***MQ bridge*:**	**MyBridge**
***Local qMgr*:**	**GATEWAY00QM**

***Detail tab*:**

***Port*:**	**1414 (to match that listened on by MQSeries).**
***Sync queue*:**	**SYNC.Q.FOR.GATEWAY00QM**

Step 17: Create a listener

This will connect to the transmission queue on MQSeries, pulling messages continuously from that queue and posting them onto the MQe network. Note that you require one such object for each MQSeries transmission queue that is conveying messages to the MQe network. Note also that the name of the listener is the same as the transmission queue name on which it acts.

Using the *Gateway00QM* instance – right click on the *FOR.GATEWAY00QM* client connection node in the tree for the *GATEWAY00QM* queue manager (i.e. *MQe root\GATEWAY00QM\Bridges\MyBridge\CENTRAL00QM\FOR.GATEWAY00QM*) and select *New MQ Listener*. Using the *New MQ Listener* panel, create the listener:

***General tab*:**

***Name*:**	**TO.GATEWAY00QM**
***Client conn*:**	**FOR.GATEWAY00QM**
***Proxy*:**	**CENTRAL00QM**
***MQ bridge*:**	**MyBridge**
***Local qMgr*:**	**GATEWAY00QM**

Step 18: Start the bridge, so that message transfer is activated

Using the *Gateway00QM* instance – right click on the *MyBridge* node in the tree for the *GATEWAY00QM* queue manager (i.e. *MQe root\GATEWAY00QM\Bridges\MyBridge*) and select the *Start* action. The state of the bridge will change to "Running"; this can be seen by selecting the Bridges node in the tree (*MQe root\GATEWAY00QM\Bridges*); in the right hand pane will be seen the properties of *MyBridge*.

From this time, any messages sent from MQSeries to either of the MQe queue managers should be pulled-down by the running listener.

© Copyright IBM Corp. 2000, 2001

Step 19: Create a bridge queue that refers to the *BRANCH.SALES.QUEUE* on the MQSeries queue manager

Using the *Gateway00QM* instance – under the *CENTRAL00QM* connection node of the *GATEWAY00QM* queue manager, create a bridge queue, i.e. right click on *MQe root\GATEWAY00QM\Connections\CENTRAL00QM* and select *New Queue*:

***General tab*:**

***Name*:**	**BRANCH.SALES.QUEUE**
***Local qMgr*:**	**GATEWAY00QM**
***Queue qMgr*:**	**CENTRAL00QM**
***Type*:**	**Bridge queue**

***MQ Bridge tab*:**

***MQ bridge*:**	**MyBridge**
***Proxy*:**	**CENTRAL00QM**
***Client conn*:**	**FOR.GATEWAY00QM**

Step 20: Put a test message from the *GATEWAY00QM* MQe queue manager to the *BRANCH.SALES.QUEUE*

Using the *Gateway00QM* instance – right click on the newly created bridge queue and use the *New Message* menu item:

***Queue mgr*:**	**CENTRAL00QM**
***Queue*:**	**BRANCH.SALES.QUEUE**

You should get the message: "Message sent from 'GATEWAY00QM' … etc." If you get the error: "Unable to send the message – generic MQ error" then check that the properties of the client connection you created – it must have a sync queue which matches that defined on your MQSeries queue manager – in this case *SYNC.QUEUE.FOR.GATEWAY00QM*. Note that if you change this property, you need to stop and then re-start the bridge to make the change take effect.

Another common cause of problems is the *classpath* environment variable, as set in the *Start→Control Panel→>System→Advanced→Environment Variables*. It should reference the following jar files:

C:\Program Files\IBM\MQSeries\java\lib\com.ibm.mq.jar;
C:\Program Files\IBM\MQSeries\java\lib\com.ibm.mqbind.jar;
C:\Program Files\IBM\MQSeries\java\lib\com.ibm.mq.iiop.jar;
(Note: The above is mq.iiop.jar and not mqiiop.jar)

The current value of the *classpath* variable is shown by MQe_Explorer via *Help→About MQe_Explorer→System Info*.

When you have successfully put a message, look at the message by using the MQSeries Explorer and browsing the *BRANCH.SALES.QUEUE*. Now view the same queue from MQe_Explorer by refreshing the *BRANCH.SALES.QUEUE*. Delete the message using the MQSeries Explorer if you wish.

© Copyright IBM Corp. 2000, 2001, 2002

Step 21: Tell the BRANCH000QM about the BRANCH.SALES.QUEUE on CENTRAL00QM

Using the *BRANCH000QM* instance – add a remote queue definition to the *BRANCH001QM* queue manager to refer to the *BRANCH.SALES.QUEUE* on the *CENTRAL00QM* queue manager. You can right click on *MQe root\BRANCH000QM\Connections\CENTRAL00QM* and select *New Queue*.

***Name*:**	**BRANCH.SALES.QUEUE**
***Local qMgr*:**	**BRANCH000QM**
***Queue qMgr*:**	**CENTRAL00QM**
***Type*:**	**Remote queue**
***Mode*:**	**Synchronous**

Step 22: Put a test message to the *BRANCH.SALES.QUEUE* from the *BRANCH000QM* queue manager

Using the *BRANCH000QM* instance – right click on the remote queue definition for *BRANCH.SALES.QUEUE* on the *CENTRAL00QM* queue manager (i.e. *MQe root\BRANCH000QM\Connections\CENTRAL00QM\BRANCH.SALES.QUEUE*) and select *New Message*. Use the defaults to send the message.

You should now be able to view the message, which is sitting on the MQSeries queue, using MQe's synchronous queue browse operation, from *BRANCH000QM's* MQe_Explorer.

You can also view the test message from the MQSeries Explorer, as before.

Step 23: Put a message from MQSeries to the gateway queue manager

Put a number of test messages from the MQSeries queue manager to the MQe branch queue manager. An example program for this purpose (*PutFromMQ*) is shipped with MQe; in a default installation it is available in the *Program Files\MQe\Java\examples\mqbridge\application* directory. Invoke this program from a DOS prompt using the string:

```
jview examples.mqbridge.application.PutFromMQ CENTRAL00QM
    SYSTEM.DEFAULT.LOCAL.QUEUE GATEWAY00QM 1 1 hello
```

The *PutFromMQ* program connects to the MQSeries queue manager and sends a message to the *SYSTEM.DEFAULT.LOCAL.QUEUE* queue on *GATEWAY00QM*. The remote queue manager alias on *CENTRAL00QM* directs the message to the *FOR.GATEWAY00QM* transmission queue. The MQ transmission queue listener on the *GATEWAY00QM's* bridge configuration (called *FOR.GATEWAY00QM)* pulls the message down from the transmission queue, and posts it to the MQe network. The *GATEWAY00QM* queue manager directs the message to its local *SYSTEM.DEFAULT.LOCAL.QUEUE* queue.

Use the MQe_Explorer to view the test messages.

Step 24: Send test messages from MQSeries to the *BRANCH000QM* queue manager

Use the *PutFromMQ* program as above, this time with the command string:

```
jview examples.mqbridge.application.PutFromMQ CENTRAL00QM
    SYSTEM.DEFAULT.LOCAL.QUEUE BRANCH000QM 1 1 hello
```

The route taken by these messages is the same as that used by the previous messages, except that the *GATEWAY00QM* queue manager pushes the messages to the *BRANCH000QM* queue manager, which then deposits them on its local *SYSTEM.DEFAULT.LOCAL.QUEUE* queue.

© Copyright IBM Corp. 2000, 2001

14 Programming MQe_Explorer applications

MQe_Explorer can be used to launch applications against the local queue manager. In this scenario typically MQe_Explorer is used to create and run the queue manager, then additional applications (i.e. classes) are loaded into the same JVM. The MQe_Explorer and the applications share the same queue manager. These applications can be stopped and started through their own graphical user interfaces; meanwhile the queue manager and MQe_Explorer continue to execute. The MQe_Explorer can be used to manage the various MQe objects and display their status, whilst its trace and console capabilities can be used to debug the applications during development.

Launching classes in this way is described in the section Load on page 108. The application *examples.mqe_explorer.PerfTest* can be run as a sample to explore these facilities. In this chapter more information is provided on writing such applications.

14.1 Windows application shell

Applications that require a Windows user interface, implemented through the Windows Foundation Classes, can be created using the Microsoft Windows J++ development environment. Such applications however will only be able to execute on a Windows platform and will require a Microsoft JVM.

The basic shell of such an application is shown in *Figure 14-1: Basic Windows application shell* on page 169. The import statements include the MQe and Windows classes needed for simple applications. The application *MyApp* is shown as being a subclass of *com.ms.wfc.ui.Form* and therefore presents a window on the screen – typically this form will have buttons, menus and other graphical interface elements defined (not shown in the example). The application is initiated by a call to the static *main()* method – and parameters can be passed as strings in that call. An instance of the *MyApp* class is created and this is passed to the *run()* method of the windows *com.ms.wfc.app.Application* class; starting the application in this way initiates the messaging environment needed by Windows such that notification of interesting events, such as mouse clicks etc. will be received.

The constructor has an obligatory *initForm()* call needed by Windows to initialize the form; following that it may be useful to get addressability to the MQe queue manager. From this point, the application logic can be programmed.

The *close()* method shows the minimum processing needed to close the application. The queue manager itself should not be closed since MQe_Explorer will still be using it, likewise the JVM should not be destroyed.

© Copyright IBM Corp. 2000, 2001, 2002

```
import com.ms.wfc.app.*;
import com.ms.wfc.core.*;
import com.ms.wfc.ui.*;
import com.ibm.mqe.*;
import com.ibm.mqe.administration.*;

public class MyApp extends Form
{
  MQeQueueManager          qMgr;    //local queue manager

  public MyApp()
  {
            // Required for Visual J++ Form Designer support
            initForm();
            //get addressability to the local queue manager
            qMgr = MQeQueueManager.getReference(null);
            //application continues ……
  }

  //close application
  private void close()
  {
            //dispose of the form and close the application
            this.dispose();
            Application.exitThread();
  }

  //dispose of all resources
  public void dispose()
  {
            super.dispose();
            components.dispose();
            return;
  }

  //main entry point for the application.
  //  - args: array of parameters passed to the application via the command line.
  public static void main(String args[])
  {
            try                              //errors may occur if exit before the form is loaded
            {
                     Application.run(new MyApp());
            }
            catch (Throwable x) {}
  }

}
```

Figure 14-1: Basic Windows application shell

14.2 Non-windows application shell

Applications without a user-interface, or applications that use the cross-platform Java classes can also be loaded and run by MQe_Explorer. The basic shell is similar and *Figure 14-2: Standard application shell* shows the code needed in the simplest case.

© Copyright IBM Corp. 2000, 2001

```
import com.ibm.mqe.*;
import com.ibm.mqe.administration.*;

public class MyApp
{
  MQeQueueManager         qMgr;    //local queue manager

  public MyApp()
  {
            //get addressability to the local queue manager
            qMgr = MQeQueueManager.getReference(null);
            //application continues ……

            // …… application finishes
  }

  //main entry point for the application.
  //  - args: array of parameters passed to the application via the command line.
  public static void main(String args[])
  {
            MyApp myApp = new MyApp();
  }

}
```

Figure 14-2: Standard application shell

14.3 Handling console interrupts

The MQe_Explorer console allows threads to be interrupted; this is provided so that applications can be terminated from the console. However, this supports requires the application to be written such that these termination are possible. Very little code is required, as is shown in *Figure 14-3: Console interrupt programming*.

Periodic checks are required to test if the application thread has been interrupted; if so, the application must be closed, (as in the *close()* method in Figure 14-1: Basic Windows application shell) before the thread itself terminates.

```
//application work
while (running)
{
            //application work …
            //……..

            if (Thread.currentThread().isInterrupted())
            {
                        this.close();
                        return;
            }
}
```

Figure 14-3: Console interrupt programming

© Copyright IBM Corp. 2000, 2001, 2002

15 Appendix A: Performance tool

A performance tool is included with MQe_Explorer, not just as an example of a program that can be loaded concurrently into the same JVM and able to share the same local queue manager, but also to support basic performance measurements of MQe functions.

The following operations are supported:

- Put – in a series of put/get pairs (both one- and two-phase operation)
- Put – in a series of consecutive puts
- Get – in a series of put/get pairs (both one- and two-phase operation)
- Put/get pairs (both one- and two-phase operation)
- Browse (retrieve full message or just UID; without locking or lock/unlock pairs)

The target queue may optionally cleared before the test and/or be pre-loaded with messages and directed at be chosen from any local or remote queue manager. Messages are either of the message class *com.ibm.mqe.MQeMsgObject* with a single field containing an array of bytes, of the designated payload size. Alternatively the message class can be *com.ibm.mqe.mqemqmessage.MQeMQMsgObject*. All bytes are randomly generated and all messages are unique. Intentionally, no data compression is possible.

Multiple instances of the performance tool can be run simultaneously; however when using the same target queue care must be taken that simultaneously tests do not conflict. For example, simultaneous use of the clear queue option may destroy a message that another instance is expecting to retrieve.

© Copyright IBM Corp. 2000, 2001

15.1 Menus and operations

The main window is as follows, showing the *General* tab:

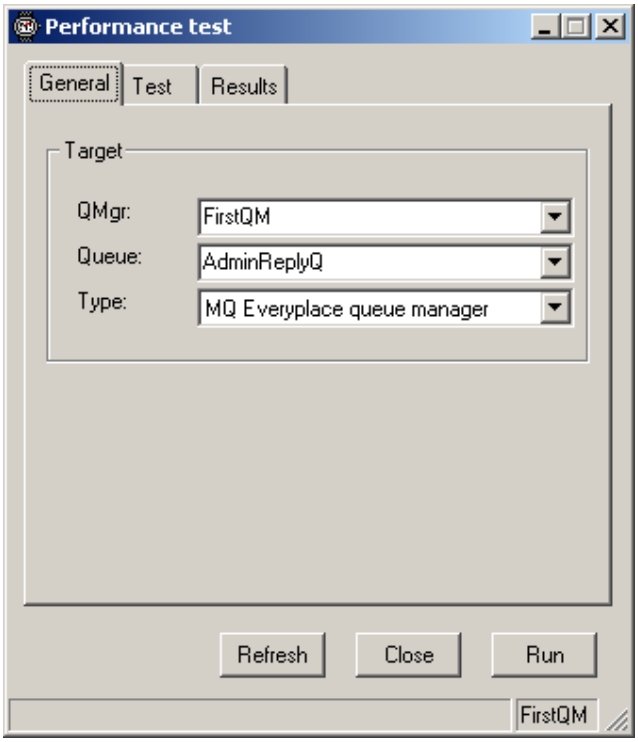


Figure 15-1: The performance tool – General tab

The status bar at the bottom of the window is used to display information messages. On the right hand side is shown the name of the local queue manager.

© Copyright IBM Corp. 2000, 2001, 2002

The following input fields are present on the *General* tab:

- *General* tab:
  - *Queue manager* – the name of the queue manager hosting the target queue to be used.
  - *Queue* – the name of the queue to be used.
  - *Description* – any UNICODE string.
  - *Type* – the type of queue manager hosting the target queue; one of:
    - MQ Everyplace queue manager
    - MQSeries queue manager

    If MQSeries queue manager is selected then the performance tool will not attempt to clear the target queue before a test is run; nor can the number of initial messages be set.

The *Test* tab has the following appearance:

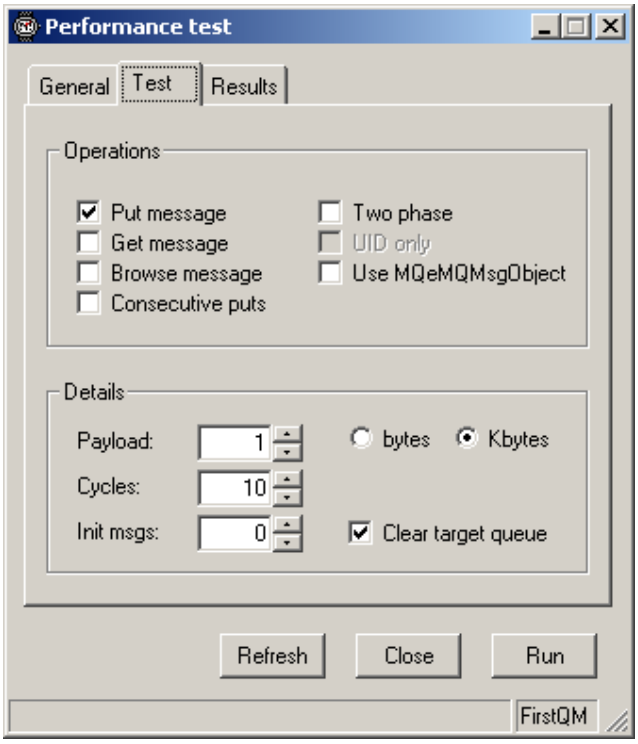


Figure 15-2: The performance tool – Test tab

© Copyright IBM Corp. 2000, 2001

The following input fields are present on the *Test* tab:

- *Test* tab:
  - *Put message* – checking this box causes the put time in a put/get pair to be measured.
  - *Get message* – checking this box causes the get time in a put/get pair to be measured. No filter is present in the get operation.
  - *Browse message* – checking this box causes the browse time to be measured. No filter is present in the browse operation – all available messages on the target queue will be browsed; no messages are removed.
  - *Consecutive puts* – checking this box causes the put time in a series of consecutive puts to be measured.
  - *Two phase* – checking this box causes two-phase operations to be used. For all puts/gets a two-step put and/or get is used, with non-zero confirm ids; otherwise a one-step operation is used with zero confirm ids. For browse a browse & lock operation is used, followed by an unlock; otherwise a single browse without lock is used.
  - *UID only* – only applicable to browse; if checked only the UIDs of messages are retrieved in the browse.
  - *Use MQeMQMsgObject* – if checked, the test messages are of class *com.ibm.mqe.mqemqmessage.MQeMQMsgObject*; otherwise they are of class *com.ibm.mqe.MQeMsgObject*.
  - *Payload* – the size of the message payload in either bytes of Kbytes.
  - *Cycles* – the number of times the test is run; the results are averaged across the cycles.
  - *Initial messages* – the number of messages to be placed on the target queue before the test is run – these messages have identical characteristics to the test messages themselves. In this way tests can be run against loaded queues. This parameter is especially important for browse since it controls the number of messages browsed.
  - *Clear target queue* – if checked, the target queue is cleared before the queue is loaded with the initial number of messages. If unchecked, the target queue is not cleared and no initial messages are loaded.

© Copyright IBM Corp. 2000, 2001, 2002

The *Results* tab has the following appearance:

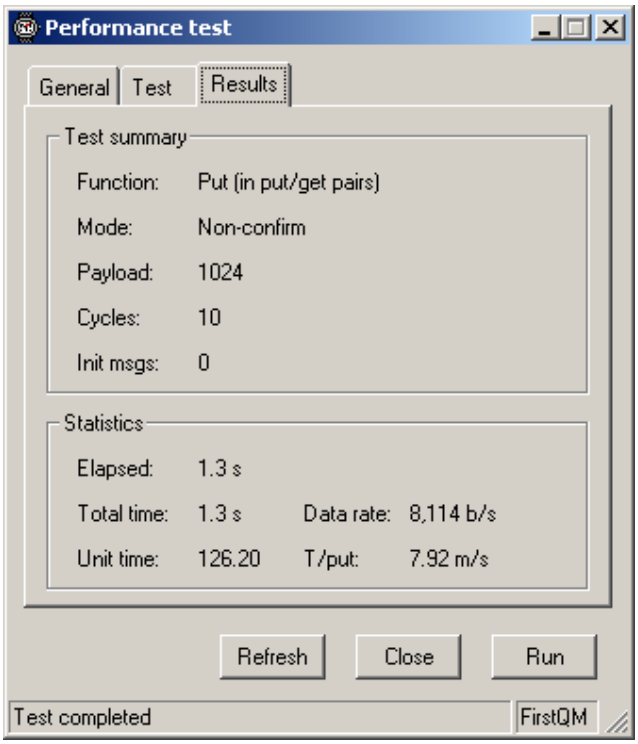


Figure 15-3: The performance tool – Results tab

The following output fields are present on the *Results* tab after the test has been run; changing the input data clears the results:

- *Results* tab:
  - *Function* – the operation tested.
  - *Mode* – details of the way the operation was implemented.
  - *Payload* – the size of the message payload in bytes.
  - *Cycles* – the number of times the test was repeated.
  - *Initial messages* – the number of messages initially loaded on the target queue.
  - *Elapsed* – the elapsed time of the test in seconds (including operations that were not part of the test itself).
  - *Total time* – the total time in seconds for all cycles of the operation to be tested.
  - *Data rate* – the bytes per second measured for the test operation.
  - *Unit time* – the average time in milliseconds for one cycle of the test operation.
  - *Throughput* – the number of messages per second for the test operation.

The *Close* button terminates the performance test application. The *Refresh* button reloads the list boxes with the currently available queue manager and queue names. The *Run* button starts the test. A running test can be aborted by interrupting the class (see *Console* on page 106 for more details); in this case the instance of the tool will also be closed.

© Copyright IBM Corp. 2000, 2001

16 Appendix B: Reference section

16.1 Icons and buttons

MQe_Explorer uses the following icons and buttons (where two icons are shown the first represents the unselected icon, the second the selected icon).

MQe object icons

Bridge-related icons

	Bridges folder (represents Bridges object)
	MQ bridge
	MQ queue manager proxy
	MQ client connection
	MQ listener

Connections

	Alias name of a remote connection to an MQ queue manager, or an alias of an alias-only connection
	Alias name of a remote connection to an MQe queue manager (direct or indirect)
	Remote connection to an MQ queue manager, or an alias-only connection
	Remote connection to an MQe queue manager (direct or indirect)
	Local connection or peer channel listener

Fields

	Fields array field
	ASCII field
	Boolean field
	Byte field
	Double field
	Fields field
	Float field
	Integer field
	Long field
	Short field
	UNICODE field
	Untyped field

© Copyright IBM Corp. 2000, 2001, 2002

Miscellaneous

Admin message (with unknown admin id)
Admin request message
Admin request detail
Admin response message (without errors)
Admin response message (with errors)
Admin response detail
WTLS mini-certificate
Channel listener
Channel manager
Folder
Message
MQe root

Queue managers

Queue manager (responding)
Queue manager (not responding)

Queues

Admin queue
Alias name of an admin queue
Alias name of a local application queue
Alias name of a queue manager
Alias name of a remote queue
Bridge queue
Forward queue
Home server queue
Local application queue
Remote queue

Threads

Class loaded by user
Main thread
MQe_Explorer thread
Other thread

© Copyright IBM Corp. 2000, 2001

Buttons

Toolbar – console window

Auto refresh (≡ *View→Refresh*

Close file (≡ *File→Close*)

Decrease priority (≡ *Action→Decrease Priority*)

Increase priority (≡ *Action→Increase Priority*)

Interrupt (≡ *Action→Interrupt*)

Load a class (≡ *Tools→Load*)

Toolbar – main window

Close current queue manager (≡ *File→Close)*

Copy (≡ *Edit→Copy)*

Create a new queue manager (≡ *File→New Queue Mgr.)*

Cut (≡ *Edit→Cut)*

Hide empty columns (≡ *View→Hide Empty)*

Open an existing queue manager (≡ *File→Open)*

Paste (≡ *Edit→Paste)*

Ping (≡ *Action→Ping)*

Refresh selected object (≡ *View→Refresh)*

Start (≡ *Action→Start)*

Stop (≡ *Action→Stop)*

Translate coded admin field names (≡ *View→Translate Admin)*

Trigger sending of messages (≡ *Action→Trigger Local)*

Select parent of the currently selected object

Use multiple rows for aliases (≡ *View→Use Multiple)*

© Copyright IBM Corp. 2000, 2001, 2002

Toolbar – off-line admin window


Close file (≡ *File→Close*)


Delete object (≡ *Edit→Delete*)


Refresh selected object (≡ *View→Refresh*)


Resend admin message (≡ *File→Resend*)


Show detail (≡ *View→Detail*)


Show off-line request messages (≡ *View→Requests*)


Show off-line reply messages (≡ *View→Replies*)


Show other messages (≡ *View→Others*)

Toolbar – trace window

Copy System.err to a log file (≡ *File→Log System.err)*


Copy selected tab contents to a file (≡ *File→Save As…)*


Cut selected text to the clipboard (≡ *Edit→Cut)*


Copy selected text to the clipboard (≡ *Edit→Copy)*


Paste text from the clipboard (≡ *Edit→Paste)*


Find text string (≡ *Edit→Find)*


Change font formatting of selected text (≡ *Format→Font)*


Include time stamps (≡ *View→Timestamps)*


Include object names (≡ *View→Object Names)*


Include thread names (≡ *View→Thread Names)*


Show calls to debug messages (≡*Action →Calls to Debug)*


Show debug messages (≡ *Action→Debug)*


Show error messages (≡ *Action→Error)*


Show exception messages (≡ *Action→Exception)*


Show information messages (≡ *Action→Information)*


Show security messages (≡*Action →Security)*


Show system messages (≡ *Action→System)*


Show the contents of System.err (≡ *View→System.err)*


Show the contents of System.out (≡ *View→System.out)*


Show warning messages (≡*Action →Warning)*

© Copyright IBM Corp. 2000, 2001

16.2 Initialization file sections

MQe_Explorer uses additional information in initialization (*.ini*) files beyond that described in the MQe product publications.

[MQe_Explorer section]

The presence of an [MQe_Explorer] section in an initialization file indicates that MQe_Explorer created the queue manager. The section can contain the following entries:

Entry name	Description
(ascii)Address	IP address of the queue manager.
(ascii)ChannelClass	Remote channel class to be used for an incoming connection.
(ascii)ChannelOptions	Options to be set on a remote communications adapter used for an incoming connection.
(ascii)CommsAdapter	The class (or alias) of a remote communications adapter to be used for an incoming connection.
(ascii)EncParms	Encoded parameters to be passed to a remote communications adapter when defining an incoming connection.
(ascii)LocalRegType	Local registry type; one of the following values: *FileRegistry*, *PrivateRegistry*.
(ascii)Parameters	ASCII parameters to be passed to a remote communications adapter when defining an incoming connection.
(ascii)Port	IP port number of the queue manager
(ascii)RuleData	Data to be passed to a remote communications adapter when defining an incoming connection.
(ascii)Type	The type of queue manager; one of the following values: *client*, *peer*, *server*, *gateway*.

Figure 16-1: [MQe_Explorer] section entries

The entries present depend upon the type of the queue manager:

Entry name	Client	Peer	Server	Gateway
Address		✓	✓	✓
ChannelClass		✓	✓	✓
ChannelOptions		✓	✓	✓
CommsAdapter		✓	✓	✓
EncParms		✓	✓	✓
LocalRegType	✓	✓	✓	✓
Parameters		✓	✓	✓
Port		✓	✓	✓
RuleData		✓	✓	✓
Type	✓	✓	✓	✓

Figure 16-2: [MQe_Explorer] entry presence

© Copyright IBM Corp. 2000, 2001, 2002

[QueueManager] section

The following entry will be present in an initialization file created by MQe_Explorer:

(ascii)QueueManager=<*queue manager class*>

The <*queue manager class*> identifies the queue manager class that MQe_Explorer will instantiate on starting the queue manager. If the entry is not present, a class that is mapped to the alias "QueueManager" will be used.

The following entries will be actioned if present in the configuration file of an unconfigured queue manager. After configuration, any such entry(ies) will be deleted.

(ascii)ChAttrRule=<*channel attribute rule*>

The <*channel attribute rule*> value will be used to set the queue manager channel attribute rule property. If the entry is not present, a default value will be used.

(long)ChTimeout=<*channel timeout*>

The <*channel timeout*> value will be used to set the queue manager channel timeout property. If the entry is not present, a default value will be used.

(unicode)Description=<*description*>

The <*description*> value will be used to set the queue manager description property. If the entry is not present, a default value will be used.

(ascii)MessageStore=<*message store*>

The <*message store*> value will be used to set the queue manager default message store property. If the entry is not present, a default value will be used.

(ascii)QMgrRule=<*queue manager rule*>

The <*queue manager rule*> value will be used to set the queue manager rule property. If the entry is not present, a default value will be used.

(ascii)QueueAdapter=<*queue adapter*>

The <*queue adapter*> value will be used to set the queue manager default queue adapter property. If the entry is not present, a default value will be used.

(ascii)QueuePath=<*queue path*>

The <*queue path*> value will be used to set the queue manager default queue path property. If the entry is not present, a default value will be used.

[Registry] section

The following entry will be present in an initialization file created by MQe_Explorer:

(boolean)QueueRegistry=<*enabled*>

The parameter <*enabled*> has the value 'true' or 'false' depending on whether queue registries are to be allowed. If the entry is not present, the value 'true' is assumed.

© Copyright IBM Corp. 2000, 2001

16.3 MQe_Explorer external variables

The following variables give additional control over MQe_Explorer operation:

Class: MQeOptions

The following variables determine whether error messages will be shown when MQe_Explorer detects errors whilst background loading object details:

```
public static boolean   showLinkageErrors;          //show linkage errors
public static boolean   showClassNotFoundErrors;    //show class not found errors
public static boolean   showIllegalAccessErrors;    //show illegal access errors
public static boolean   showIOExceptionErrors;      //show I/O exception errors
public static boolean   showExceptionErrors;        //show exception errors
```

© Copyright IBM Corp. 2000, 2001, 2002

16.4 Accessibility features

The following general keystroke aids to navigation are supported:

Keystroke(s)	Place	Function
↑	Main window: list pane Console window Off-line admin window	Go up one list view item
↑	All windows: menu	Go up one menu item
↑	Main window: tree pane	Go up one node in the tree
↓	Main window: list pane Console window Off-line admin window	Go down one list view item If nothing is selected, select the first item
↓	All windows: menu	Go down one menu item
↓	Main window: tree pane	Go down one node in the tree
←	Main window: list pane Console window Off-line admin window	Scroll left
←	All windows: menu	Go left one menu item
←	Main window: tree pane	Shrink current node in the tree
→	Main window: list pane Console window Off-line admin window	Scroll right
→	All windows: menu	Go right one menu item
→	Main window: tree pane	Expand the current node in the tree
F6	Main window	Cycle through panes
F10	All windows	Move focus to the menu bar *(then press underlined key to activate menu item)*
Alt	All windows	Move focus to the menu bar *(then press underlined key to activate menu item)*
Alt+space	All windows	Control box
Alt+tab	All	Cycle through windows
AltGr+tab	Trace Property pages	Cycle through tabs
Ctrl +tab	Trace Property pages	Cycle through tabs
Tab	Property pages	Cycle forwards through input fields & buttons
Enter	Property pages: buttons	Click selected button
Enter	All windows: menus	Click selected menu item
Shift+Tab	Property pages	Cycle backwards through input fields & buttons

Figure 16-3: General keystroke aids to navigation

© Copyright IBM Corp. 2000, 2001

The following keystroke aids to navigation are supported only in the Trace window *System.err* and *System.out* text areas:

Keystroke(s)	Function
↑	Move insertion point up a line
↓	Move insertion point down a line
←	Move insertion point one character to the left
→	Move insertion point one character to the right
Ctrl+↑	Move insertion point one line up
Ctrl+↓	Move insertion point one line down
Ctrl+←	Move insertion point one character to the left
Ctrl+→	Move insertion point one character to the right
Ctrl+Shift+↑	Extend selection to the beginning of the paragraph
Ctrl+Shift+↓	Extend selection to the end of the paragraph
Ctrl+Shift+←	Extend selection to the beginning of the word
Ctrl+Shift+→	Extend selection to the end of the word
Ctrl+A	Extend selection to include all text
Ctrl+End	Move insertion point to the end
Ctrl+Home	Move insertion point to the beginning
End	Move insertion point to the end of the line
Home	Move insertion point to the beginning of the line
Shift+↑	Extend selection one line up
Shift+↓	Extend selection one line down
Shift+←	Extend selection one character to the left
Shift+→	Extend selection one character to the right
Ctrl+Shift+Home	Extend selection to the beginning
Ctrl+Shift+End	Extend selection to the end
Shift+End	Extend selection to the end of the line
Shift+Home	Extend selection to the beginning of the line
Shift+PgDn	Extend selection to end
Shift+PgUp	Extend selection to beginning
Tab	If no insertion point, display insertion point at end If insertion point present, insert a tab

Figure 16-4: Restricted keystroke aids to navigation

© Copyright IBM Corp. 2000, 2001, 2002

Shortcuts

The following keystroke shortcuts are supported:

Keystroke(s)	Window	Name	Menu equivalent	Function
Ctrl+A	Main Off-line Admin Trace	Select All	Edit→Select All	Select all
Ctrl+C	Main Trace	Copy	Edit→Copy	Copy to clipboard
Ctrl+D	Console	Decrease priority	Action→Decrease priority	Decrease thread priority
Ctrl+D	Main	Demo mode	Tools→Demo Mode	Demo Mode
Ctrl+Delete	Main Off-line Admin Trace	Clear	Edit→Clear	Clear
Ctrl+F	Main	Off-line admin	View→Off-line Admin	View off-line admin window
Ctrl+F	Trace	Find	Edit→Find	Find string/word
Ctrl+G	Main	Ping	Action→Ping	Ping queue manager
Ctrl+I	Console	Increase priority	Action→Increase priority	Increase thread priority
Ctrl+L	Main Console	Load	Tools→Load	Load a class
Ctrl+N	Main	Options	Tools→Options	View options window
Ctrl+O	Main	Open	File→Open	Open an initialization file
Ctrl+P	Main	Stop	Action→Stop	Stop selected object
Ctrl+R	Main	Renew cert.	Action→Renew Credentials	Renew qMgr mini-cert.
Ctrl+R	Off-line admin	Resend	File→Resend	Resend message
Ctrl+S	Main	Start	Action→Start	Start selected object
Ctrl+T	Console	Interrupt	Action→Interrupt	Interrupt selected thread
Ctrl+T	Main	Trigger	Action→Trigger Local	Trigger local queue manager
Ctrl+X	Main Trace	Cut	Edit→Cut	Cut to clipboard
Ctrl+V	Main Trace	Paste	Edit→Paste	Paste from clipboard
Ctrl+Z	Main	Console	View→Console	View console window
Delete	Main Off-line Admin Trace	Delete	Edit→Delete	Delete selected
F4	Main	Properties	File→Properties	Display object properties
F5	Console Main Off-line Admin	Refresh	View→Refresh	Refresh selected object

Figure 16-5: Keystroke shortcuts

© Copyright IBM Corp. 2000, 2001

16.5 MQe classes

MQe_Explorer allows MQe classes to be assigned as the value of object properties. This section briefly summarizes their characteristics, from an administrative perspective.

Class types

MQe_Explorer supports the following MQe class types:

Admin console: a class that supports a graphical user interface allowing the management of MQe objects.

Admin queue: a class that implements a queue object that accepts admin messages and processes them. Typically admin messages are used to configure MQe objects at the target queue manager; replies may be sent after processing.

Attribute key: a security key that is used by an attribute object to protect data. An attribute object contains the mechanisms necessary to perform authentication, encryption and compression.

Attribute rule: a rule that determines whether the authenticator, cryptor or compressor in an attribute object can be replaced.

Authenticator: a mechanism used to determine whether supplied credentials allow access to the underlying data.

Bridge queue: a class that implements a queue object acting as a proxy for an MQ queue on an MQ queue manager. It uses an MQ client connection and a transformer object to send messages to, or to browse, that queue.

Channel: a class that handles the movement of data from one queue manager to another, exploiting the capabilities of an underlying communications adapter.

Communications adapter: a class that handles the sending and receipt of data over a network. Each communications adapter class is specific to a particular network protocol (e.g. HTTP, TCP/IP, UDP). More than one communications adapter may be available for any one protocol – they may differ in size and/or performance i.e. speed and wire footprint.

Compressor: a mechanism used to encode underlying data in order to reduce its size. Individual compressors are normally optimized to specific types of data.

Connection: an MQe object that supplies all the information necessary for one MQe queue manager to communicate with another MQe queue manager (e.g. address data, channel adapter, communications adapter, miscellaneous parameters etc).

Connection manager: a class that handles the case where multiple communications adapters being used to realize a connection to a remote queue manager.

Connection manager rule: a class that is invoked whenever there is a change of state in the connection manager.

Cryptor: a mechanism used to encode underlying data in order to hide its meaning.

Home server queue: a class that implements a queue object acting as an intermediary in queue manager-based routing. Home server queues pull messages destined for their local queue manager from a remote store and forward queue. Messages pulled are then delivered to the target queues on the local queue manager. Home server queues do not store messages.

© Copyright IBM Corp. 2000, 2001, 2002

Load bridge rule: the rule class invoked when an MQe bridges object is initiated and which determines whether or not each bridge defined in the registry should be loaded.

Message storage adapter: a class that determines the way in which an underlying storage adapter is used to hold messages.

Message transformer: a class that transforms an MQe message to an MQ message, and vice versa.

MQ bridge: an object that represents a set of routings between an MQe queue manager and one or more MQ queue managers.

MQ client connection: an object that represents a single routing between an MQe queue manager and one or more MQ queue managers.

MQ listener: an object that connects to a single MQ transmission queue and moves messages from that queue to MQe.

MQ proxy queue manager: an object that represents an MQ queue manager to an MQe queue manager.

MQ adapter: an object that moves messages from an MQe queue manager to an MQ queue manager.

Queue manager: a class that implements the functions of an MQe queue manager, i.e. managing queues, connections, channels and other related objects.

Queue manager rule: a class that is invoked whenever there is a change of state in the queue manager.

Queue: a class that accepts or handles messages.

Queue rule: a class that is invoked whenever there is a change of state in the queue.

Registry service provider: a class that provides access to persistent storage for the MQe registry.

Remote queue: a class that implements a queue object acting as a proxy for a queue elsewhere in the MQe network. Remote queues transmit messages to the destination queue. If unable to transmit them and, if configured for an asynchronous delivery, then they store the messages until such time as they are able to deliver.

Start-up rule: a class that determines if, when an object is loaded, it should be started, or left in the stopped state; likewise with that object's parents.

Storage adapter: a class that provides support for the reading and writing of data to physical storage. Each storage adapter class is specific to a particular medium (e.g. file system, memory, database etc). More than one storage adapter may be available for any one medium – they may differ in performance and the level of assurance supplied.

Store and forward queue: a class that implements a queue object acting as an intermediary in queue manager-based routing. Store and forward queues collect messages destined for one or more remote destination queue managers. Optionally they may forward such messages to a named target queue manager. If unable to transmit them, they store the messages until such time as they are able to deliver. Applications are not able to access messages on store and forward queues.

© Copyright IBM Corp. 2000, 2001

Sync queue purger rule: a class that determines what action should be taken with records that left on the sync queue. The sync queue contains one record per MQe message that is in the state of having been sent to MQ, but still locked ready for deletion at the MQe queue manager. In the event of a crash, these locked messages may be left on the sync queue and can build up over time.

Transporter: a class that handles the movement of data to a target queue on a target queue manager, exploiting the capabilities of an underlying channel.

Undelivered message rule: a class that determines what action an MQ listener object should take if a message cannot be delivered from MQ to MQe. Examples of possible actions are: retrying after a wait, moving the message to the MQ queue manager's dead letter queue, or stopping the listener.

Class details

com.ibm.mqe.adapters.MQeDiskFieldsAdapter

Type

Storage adapter

Overview

Provides a persistent store for data, using the file system. Typically this is the default adapter for queues and the registry, since it offers the greatest assurance that data has not been lost. It does not rely on the operating system to complete lazy writes.

---

com.ibm.mqe.adapters.MQeMemoryFieldsAdapter

Type

Storage adapter

Overview

Provides a non-persistent, temporary store for data, using memory. Typically this adapter is used to store queues where fast access is required and where the messages need not survive the queue manager, nor survive system failure. In the current release, a registry service provider cannot use this adapter.

---

com.ibm.mqe.adapters.MQeReducedDiskFieldsAdapter

Type

Storage adapter

Overview

Provides a persistent store for data, using the file system. This adapter is a higher-speed alternative to the *com.ibm.mqe.adapters.MQeDiskFieldsAdapter*. However, it does introduce a dependency on the operating system staying up long enough to complete lazy writes.

© Copyright IBM Corp. 2000, 2001, 2002

com.ibm.mqe.adapters.MQeTcpipHistoryAdapter

Type

Communications adapter

Overview

Implements an efficient protocol over TCP/IP. By default, this is the preferred adapter for TCP/IP, offering better performance than either *com.ibm.mqe.adapters.MQeTcpipLengthAdapter* or *com.ibm.mqe.adapters.MQeTcpipHTTPAdapter*. This adapter takes options that control its operation:

<HISTORY> ensures that recently used data is cached to avoid re-transmission.

<NOHISTORY> ensures that recently used data is not cached.

<NOPERSIST> ensures that the connection does not beyond a single data transfers.

<PERSIST> ensures that the connection survives data transfers and is re-used.

<HISTORY><PERSIST> are the defaults if no options are specified.

---

com.ibm.mqe.adapters.MQeTcpipHttpAdapter

Type

Communications adapter

Overview

Implements communications over HTTP using the HTTP 1.0 protocol. Since HTTP adds additional overhead, where there is a choice the *com.ibm.mqe.adapters.*MQeTcpipHistoryAdapter should be used in preference.

---

com.ibm.mqe.adapters.MQeTcpipLengthAdapter

Type

Communications adapter

Overview

Implements a simple, byte efficient protocol over TCP/IP. The connection is not maintained between data transfers. For almost all purposes where TCP/IP is required, excepting where minimal storage is an important consideration, the *com.ibm.mqe.adapters.MQeTcpipHistoryAdapter* should be used in preference.

---

com.ibm.mqe.adapters.MQeUdpipAdapter

Type

Communications adapter

Overview

Provides support for assured data transfer using UDP/IP datagrams.

© Copyright IBM Corp. 2000, 2001

com.ibm.mqe.adapters.MQeWESAuthenticationAdapter

Type

Communications adapter

Overview

Provides support for tunneling HTTP requests through WebSphere Everyplace Authentication and transparent proxies.

---

com.ibm.mqe.attributes.MQe3DESCryptor

Type

Cryptor

Overview

Provides an *MQeCryptor* object that uses triple DES CBC to encrypt/decrypt a byte array.

---

com.ibm.mqe.attributes.MQeDESCryptor

Type

Cryptor

Overview

Provides an *MQeCryptor* object that uses DES CBC to encrypt/decrypt a byte array.

---

com.ibm.mqe.attributes.MQeGZIPCompressor

Type

Compressor

Overview

Compresses and decompresses a byte array using the GZIP algorithm. This likely to be effective where the data has frequently repeating words or byte patterns; the algorithm uses a reference back to the first occurrence of a pattern.

---

com.ibm.mqe.attributes.MQeLZWCompressor

Type

Compressor

Overview

Compresses and decompresses a byte array using the LZW algorithm. The algorithm uses a dictionary structure and is likely to be effective where the data has frequently repeating words or byte patterns.

© Copyright IBM Corp. 2000, 2001, 2002

com.ibm.mqe.attributes.MQeMARSCryptor

Type

Cryptor

Overview

Provides an *MQeCryptor* object that uses MARS to encrypt/decrypt a byte array.

---

com.ibm.mqe.attributes.MQeRC4Cryptor

Type

Cryptor

Overview

Provides an *MQeCryptor* object that uses RC4 to encrypt/decrypt a byte array.

---

com.ibm.mqe.attributes.MQeRC6Cryptor

Type

Cryptor

Overview

Provides an *MQeCryptor* object that uses RC6 to encrypt/decrypt a byte array.

---

com.ibm.mqe.attributes.MQeRleCompressor

Type

Compressor

Overview

Compresses and decompresses a byte array using a simple run length encoding algorithm. This is effective where the data to be compressed contains strings of repeated bytes.

---

com.ibm.mqe.attributes.MQeSharedKey

Type

Attribute key

Overview

Manages Diffie Hellman key exchange and the establishment of shared master-secret key needed by symmetric cryptors.

---

com.ibm.mqe.attributes.MQeWTLSCertAuthenticator

Type

Authenticator

Overview

Manages mutual authentication using WTLS size-optimized certificates.

© Copyright IBM Corp. 2000, 2001

com.ibm.mqe.attributes.MQeXorCryptor

Type

Cryptor

Overview

Provides an *MQeCryptor* object that uses a simple exclusive OR algorithm to disguise the data. This cryptor does not provide a high level of data protection but obscures the data sufficiently to prevent visual recognition.

---

com.ibm.mqe.messagestore.MQeMessageStore

Type

Message store adapter

Overview

Implements the standard MQe message store and uses a file for each message with a name in a 16.6 format. Unless special considerations apply, this message store adapter should be selected.

---

com.ibm.mqe.messagestore.MQeShortFilenameMessageStore

Type

Message store adapter

Overview

Implements an MQe message store that uses a file for each message with a name in an 8.3 format. This adapter is provided for those situations where either only an 8.3 file system exists or where the 8.3 file system has essential performance characteristics. Be aware that the number of uniquely available names may cause problems. In almost all cases the *com.ibm.mqe.messagestore.MQeMessageStore* adapter should be used in preference.

---

com.ibm.mqe.mqbridge.MQeBaseTransformer

Type

Message transformer

Overview

A basic transformer for transforming MQe messages to MQ format and vice versa.

---

com.ibm.mqe.mqbridge.MQeClientConnection

Type

MQ client connection

Overview

Provides the default implementation of the MQe MQ client connection object.

© Copyright IBM Corp. 2000, 2001, 2002

com.ibm.mqe.mqbridge.MQeListener

Type

MQ listener

Overview

Provides the default implementation of the MQe MQ listener object.

---

com.ibm.mqe.mqbridge.MQeLoadBridgeRule

Type

Load bridge rule

Overview

Provides the default implementation of a load bridge rule. This implementation allows all bridges to load.

---

com.ibm.mqe.mqbridge.MQeMQAdapter

Type

MQ adapter

Overview

Provides the default implementation of the MQe MQ adapter object.

---

com.ibm.mqe.mqbridge.MQeMQBridge

Type

MQ bridge

Overview

Provides the default implementation of the MQe MQ bridge object.

---

com.ibm.mqe.mqbridge.MQeMQBridgeQueue

Type

Bridge queue

Overview

Provides the default implementation of the MQe bridge queue object.

---

com.ibm.mqe.mqbridge.MQeMQBridges

Type

Bridges object

Overview

Provides the default implementation of the MQe bridges object.

© Copyright IBM Corp. 2000, 2001

com.ibm.mqe.mqbridge.MQeMQQMgrProxy

Type

MQ queue manager proxy

Overview

Provides the default implementation of the MQe MQ queue manager proxy object.

---

com.ibm.mqe.mqbridge.MQeStartupRule

Type

Start-up rule

Overview

Provides the default implementation of a start-up rule. The object itself and its parents are started.

---

com.ibm.mqe.mqbridge.MQeSyncQueuePurgerRule

Type

Sync queue purger rule

Overview

Provides the default implementation of a sync queue purger rule. Warnings are generated for messages over ten minutes old but no messages are purged.

---

com.ibm.mqe.mqbridge.MQeUndeliveredMessageRule

Type

Undelivered message rule

Overview

Provides the default implementation of an undelivered message rule. It retries three times with increasing timeouts, after which time the listener stops.

---

com.ibm.mqe.MQeAdminQueue

Type

Admin queue

Overview

Provides the default implementation of the MQe admin queue object.

© Copyright IBM Corp. 2000, 2001, 2002

com.ibm.mqe.MQeChannel

Type

Channel

Overview

Provides the default implementation of an MQe channel object providing client/server communication. Client/server communication means that only the client can initiate data transfers between the parties; the data itself can flow in either direction.

---

com.ibm.mqe.MQeConnectionManager

Type

Connection manager

Overview

Provides the default implementation of an MQe connection manager.

---

com.ibm.mqe.MQeHomeServerQueue

Type

Home server queue

Overview

Provides the default implementation of the MQe home server queue object.

---

com.ibm.mqe.MQeKey

Type

Attribute key

Overview

Creates a base *MQeKey* object that can be attached to and used by an *attribute* object.

---

com.ibm.mqe.MQePeerChannel

Type

Channel

Overview

Provides the default implementation of an MQe channel object providing peer-to-peer communication. Peer-to-peer communication means that either party can initiate data transfers over the channel.

© Copyright IBM Corp. 2000, 2001

com.ibm.mqe.MQeQueue

Type

Queue

Overview

Provides the default implementation of the MQe local queue object.

---

com.ibm.mqe.MQeQueueManager

Type

Queue manager

Overview

Provides the default implementation of the MQe queue manager object.

---

com.ibm.mqe.MQeQueueManagerRule

Type

Queue manager rule

Overview

Provides the default implementation of the MQe queue manager rule.

---

com.ibm.mqe.MQeQueueRule

Type

Queue rule

Overview

Provides the default implementation of the MQe queue rule.

---

com.ibm.mqe.MQeRemoteQueue

Type

Remote queue

Overview

Provides the default implementation of the MQe remote queue object.

---

com.ibm.mqe.MQeStoreAndForwardQueue

Type

Store and forward queue

Overview

Provides the default implementation of the MQe store and forward queue object.

© Copyright IBM Corp. 2000, 2001, 2002

com.ibm.mqe.MQeTransporter

Type

Transporter

Overview

Provides the default implementation of the MQe transporter object.

---

com.ibm.mqe.registry.MQeFileSession

Type

Registry service provider

Overview

Provides a registry service provider that implements an unsecured registry in the file system. It uses a storage adapter to access persistent storage.

---

com.ibm.mqe.registry.MQePrivateSession

Type

Registry service provider

Overview

Provides a registry service provider that implements a secured registry in the file system. It uses a storage adapter to access persistent storage.

---

examples.administration.console.Admin

Type

Admin console

Overview

An example of a simple GUI, using Java Swing classes, that allows the management of MQe objects, excluding those related to the bridge. The source code is provided.

---

examples.attributes.NTAuthenticator

Type

Authenticator

Overview

A sample authenticator, specific to Windows NT/2000/XP systems, that presents a simple GUI interface for the local user to supply his/her Windows user credentials (user id/password/domain). The source code is provided.

© Copyright IBM Corp. 2000, 2001

examples.attributes.TableCryptor

Type

Cryptor

Overview

Provides an example of an *MQeCryptor* object that uses a simple data substitution table to encrypt/decrypt a byte array. The source code is provided.

---

examples.attributes.UnixAuthenticator

Type

Authenticator

Overview

Provides an example of an authenticator, specific to Unix systems, that presents a simple GUI interface for the local user to supply his/her user credentials (user id/password). The source code is provided.

---

examples.attributes.UseridAuthenticator

Type

Authenticator

Overview

Provides an example of an authenticator using a user id and password file that presents a simple GUI interface for the local user to supply his/her user credentials. The source code is provided.

---

examples.mqbridge.administration.console.AdminGateway

Type

Admin console

Overview

An example of a simple GUI, using Java Swing classes, that allows the management of MQe objects, including those related to the bridge. The source code is provided.

---

examples.mqbridge.rules.DestructiveMQSyncQueuePurgerRule

Type

Sync. queue purger rule

Overview

Provides an example of a sync queue purger rule. Messages over ten minutes old are purged. The source code is provided.

© Copyright IBM Corp. 2000, 2001, 2002

examples.mqbridge.rules.MQeStartupRule

Type

Start-up rule

Overview

Provides the default implementation of a start-up rule. The object itself and its parents are started. This has the same behavior as *com.ibm.mqe.mqbridge.MQeStartupRule*. The source code is provided.

---

examples.mqbridge.rules.MQeSyncQueuePurgerRule

Type

Sync. queue purger rule

Overview

Provides the default implementation of a sync queue purger rule. Warnings are generated for messages over ten minutes old but no messages are purged. This has the same behavior as *com.ibm.mqe.mqbridge.MQeSyncQueuePurgerRule*. The source code is provided.

---

examples.mqbridge.rules.MQeUndeliveredMessageRule

Type

Undelivered message rule

Overview

Provides the default implementation of an undelivered message rule. It retries three times with increasing timeouts, after which time the listener stops. This has the same behavior as *com.ibm.mqe.mqbridge.MQeUndeliveredMessageRule*. The source code is provided.

---

examples.mqbridge.rules.UndeliveredMQMessageToDLQRule

Type

Undelivered message rule

Overview

Provides an implementation of an undelivered message rule. This class either returns any undelivered message to the MQ dead letter queue or, if the message has MQRO_DISCARD set, discards it. The source code is provided.

---

examples.mqbridge.transformers.MQeBaseTransformer

Type

Message transformer

Overview

Provides the default implementation of a basic transformer for transforming MQe messages to MQ format and vice versa. This has the same behavior as *com.ibm.mqe.mqbridge.MQeBaseTransformer*. The source code is provided.

© Copyright IBM Corp. 2000, 2001

examples.mqe_explorer.PerfTest

Type

Performance test

Overview

An application for investigating the performance of basic MQe operations, such as *get*, *put* and *browses*. It allows message size to be controlled, parameters to be set, and queues to be selected. It is supplied as an example of a class that can be loaded by MQe_Explorer and run in parallel against the same queue manager. More details are provided in Appendix A: Performance tool on page 171.

---

examples.mqe_explorer.QmRule

Type

Queue manager rule

Overview

A variant of the *com.ibm.mqe.MQeQueueManagerRule* class in which the trigger transmission method is invoked every 5 secs.

---

examples.rules.AttributeRule

Type

Attribute rule

Overview

Provides a sample implementation of an attribute rule. The source code is provided.

---

examples.rules.ExampleQueueManagerRules

Type

Queue manager rule

Overview

Provides a default implementation of the MQe queue manager rule. This has the same behavior as *com.ibm.mqe.MQeQueueManagerRule*. The source code is provided.

---

examples.rules.QueueLoggerRule

Type

Queue rule

Overview

Provides a sample implementation of a queue rule. The activation and closure of queues is logged. The source code is provided.

© Copyright IBM Corp. 2000, 2001, 2002

examples.xml.adapters.MQeXMLFieldsAdapter

Type

Storage adapter

Overview

Provides a sample implementation of a storage adapter in which the data is written out in XML format. The source code is provided.

---

examples.xml.bridge.MQeXMLTransformer

Type

Message transformer

Overview

Provides a sample message transformer that transforms MQe messages to MQ messages where the content is rendered into XML (and vice versa). The source code is provided.

© Copyright IBM Corp. 2000, 2001

17 Appendix C: MQeFileTransferMsg class

The *MQeFileTransferMsg* class is contained in the *com.ibm.mqe.mqe_explorer* package and is used to represent a file (or a segment of a file) being transferred as a message. The class is a subclass of *MQeMsgObject* and adds the following function:

The ability to carry ancillary data associated with the file, such as description, source file specification, segment identification etc.

A validation check on the integrity of the data. When data is stored in the message object a digest is calculated and carried with the message; when retrieval of the data is attempted a new digest is calculated – data is released only if the digests match.

Every instance of an *MQeFileTransferMsg* object represents a segment of a file; an associated file may be mapped into one (or more) such segments. The collection of instances representing one such file are uniquely identified by the combination of:

A unique id (supplied by the programmer).

The originating queue manager name, i.e. that queue manager that instantiated the *MQeFileTransferMsg* objects (and available through the method call *MQeMsgObject.getOriginQMgr*).

Each segment instance carries all the common information associated with the file, such as description, source file specification, etc. Each segment instance uniquely carries its segment identity and the data associated with that segment.

When data is stored in a segment via the *MQeFileTransferMsg.setData* method, a 20 byte SHA digest is generated and carried in the object. When data retrieval is attempted via the *MQeFileTransfer.getData* method, a new digest is calculated; if the digest matches the original, the data is returned, otherwise *null*.

This class has no dependencies on any other classes in the *com.ibm.mqe.mqe_explorer* package.

17.1 Constructor

Syntax

public MQeFileTransferMsg() throws Exception

Description

Creates a new MQeFileTransferMsg object with default values set as follows:

Code:	0
Data:	new byte[0]
Description:	""
File path:	""
Priority:	5
Segment:	0
Segments:	1
Unique id:	0

For details of these properties see the corresponding *get* methods.

Parameters

None.

Return values

None.

Exceptions

Passed from the superclass.

© Copyright IBM Corp. 2000, 2001, 2002

17.2 Methods

MQeFileTransferMsg.getCode

Syntax

public int getCode() throws Exception

Description

Returns the *code* property. The code values are undefined in release 1.27 but may be used to reflect the encoding of the data (e.g. code page) associated with the file.

Parameters

None.

Return values

code – code property.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.getData

Syntax

public byte[] getData() throws Exception

Description

Returns the data associated with the segment. If no data has been set then a new byte[0] is returned; if the data has been corrupted *null* is returned.

Parameters

None.

Return values

data – the data associated with the segment.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.getDescription

Syntax

public String getDescription() throws Exception

Description

Returns a description associated with the file. If no description has been set then a zero length String is returned.

Parameters

None.

Return values

description – the description associated with the file.

Exceptions

Passed from the superclass.

© Copyright IBM Corp. 2000, 2001

MQeFileTransferMsg.getFilePath

Syntax

public String getFilePath() throws Exception

Description

Returns a file specification associated with the file – this is expected to be the source location of the file, but can be used to indicate the target location.

Parameters

None.

Return values

path – the file specification associated with the file.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.getPriority

Syntax

public byte getPriority() throws Exception

Description

Returns the priority associated with the message; this priority is used by MQe to sequence transmission and retrieval order in queues. It is expected that all segments relating to the file will be given the same priority.

Parameters

None.

Return values

priority – MQe message priority.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.getSegment

Syntax

public int getSegment() throws Exception

Description

Returns the segment identity. Segments associated with the same file must be numbered sequentially, starting from zero.

Parameters

None.

Return values

segment – the segment identity.

Exceptions

Passed from the superclass.

© Copyright IBM Corp. 2000, 2001, 2002

MQeFileTransferMsg.getSegments

Syntax

public int getSegments() throws Exception

Description

Returns the number of segments associated with the file.

Parameters

None.

Return values

segments – the number of segments associated with the file.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.getUniqueId

Syntax

public long getUniqueId() throws Exception

Description

Returns the unique id associated with the file, chosen by the programmer. An easy choice would be to use the current time, from *System.currentTimeMillis*. The combination of this with the originating queue manager name, which is held within the object itself by virtue of the superclass, will ensure that segments associated with a single file share a key that is unique across an MQe network.

Parameters

None.

Return values

id – unique id associated with the file.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.setCode

Syntax

public void setCode(int *code*) throws Exception

Description

Sets the *code* property. There is no value checking.

Parameters

code – code property.

Return values

None.

Exceptions

Passed from the superclass.

© Copyright IBM Corp. 2000, 2001

MQeFileTransferMsg.setData

Syntax

public void setData(byte[] *data*) throws Exception

Description

Sets the data associated with the segment. If *null* is passed the data is set to new byte[0].

Parameters

data – the data associated with the segment.

Return values

None.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.setDescription

Syntax

public void setDescription(String *descr*) throws Exception

Description

Sets the description associated with the file. If null is passed the description is set to a zero length String. The full range of UNICODE characters is permitted.

Parameters

descr – the description associated with the file.

Return values

None.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.setFilePath

Syntax

public void setFilePath(String *path*) throws Exception

Description

Sets the file specification associated with the file. If null is passed the file specification is set to a zero length String. The characters in the specification are limited to the ASCII subset.

Parameters

path – the file specification associated with the file.

Return values

None.

Exceptions

Passed from the superclass.

© Copyright IBM Corp. 2000, 2001, 2002

MQeFileTransferMsg.setPriority

Syntax

public void setPriority(byte *priority*) throws Exception

Description

Sets the priority associated with the message. The priority must be in the range 0 – 9; values outside this will be set to the max. or min. as appropriate.

Parameters

priority – the MQe priority associated with the message.

Return values

None.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.setSegment

Syntax

public void setSetSegment(int *segment*) throws Exception

Description

Sets the segment identity. The identity must be >= 0; values outside this range will be set to zero.

Parameters

segment – the segment identity.

Return values

None.

Exceptions

Passed from the superclass.

MQeFileTransferMsg.setSegments

Syntax

public void setSegments(int *segments*) throws Exception

Description

Sets the number of segments associated with the file. The number of segments must be >= 1; values outside this range will be set to 1.

Parameters

Return values

None.

Exceptions

Passed from the superclass.

© Copyright IBM Corp. 2000, 2001

MQeFileTransferMsg.setUniqueId

Syntax

public void setUniqueId(long *id*) throws Exception

Description

Sets the unique id associated with the file.

Parameters

id – unique id associated with the file.

Return values

None.

Exceptions

Passed from the superclass.

17.3 Usage notes

Sending files

Files can be sent either in a single message or segmented across multiple messages. If segmented, a segment size is typically used to partition the bytes across messages, with equal numbers of bytes in the first segments and the remaining bytes in the last segment. All the segments are expected to hold all the common file data, such as description etc.; segments associated with a single file should differ only in their segment identifier and data content. The unique id property must be set to a common value for all segments associated with a single file.

In many cases it will be appropriate for file transfer messages to be sent to a dedicated destination queue, but this is not required. If a dedicated queue is not being used, and message class is not to be used to distinguish the messages on arrival, then it may be necessary to add an addition field into all file transfer messages to identify them as such – and of a type that can be used as an MQe filter in *browse* or *getMessage* operations.

If compression and security is required, this can be achieved by using MQe security attributes on that target queue.

If the messages are to be sent to a Websphere MQ queue then the contents must flow across the bridge in an MQe gateway; this will require appropriate message transformation rules to be set at the gateway.

Receiving files

There are many ways in which files can be retrieved. If a dedicated arrival queue is used then it is known that all messages are expected to contain files. If not, then the messages can be distinguished on message class or through an additional unique identifier. When one (arbitrary) message in a segmented file transfer is received, the other associated messages can be received through the unique id contained in each associated message. It may be necessary to wait for all such messages to arrive. Missing messages are obvious. The integrity of the data is established by reconstructing the original byte array representing the file; any message returning null data is corrupt.

© Copyright IBM Corp. 2000, 2001, 2002

18 Appendix D: Notices

This information was developed for products and services offered in the U.S.A. IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other 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 NON-INFRINGEMENT, 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.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire
England
SO21 2JN

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee. The licensed program described in this information and all licensed material

available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright IBM Corp. 2000, 2001

Trademarks

The following terms are trademarks of International Business machines Corporation in the United States, or other countries, or both.

AIX

IBM

MQSeries

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

UNIX is a registered trademark of X/Open in the United States and other countries.

Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

© Copyright IBM Corp. 2000, 2001, 2002

19 Appendix E: International Program License Agreement

Part 1 - General Terms

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THE PROGRAM. IBM WILL LICENSE THE PROGRAM TO YOU ONLY IF YOU FIRST ACCEPT THE TERMS OF THIS AGREEMENT. BY USING THE PROGRAM YOU AGREE TO THESE TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE UNUSED PROGRAM TO THE PARTY (EITHER IBM OR ITS RESELLER) FROM WHOM YOU ACQUIRED IT TO RECEIVE A REFUND OF THE AMOUNT YOU PAID.

The Program is owned by International Business Machines Corporation or one of its subsidiaries (IBM) or an IBM supplier, and is copyrighted and licensed, not sold.

The term "Program" means the original program and all whole or partial copies of it. A Program consists of machine-readable instructions, its components, data, audio-visual content (such as images, text, recordings, or pictures), and related licensed materials.

This Agreement includes Part 1 - General Terms, Part 2 - Country-unique Terms, and "License Information" and is the complete agreement regarding the use of this Program, and replaces any prior oral or written communications between you and IBM. The terms of Part 2 and License Information may replace or modify those of Part 1.

1. License

Use of the Program

IBM grants you a nonexclusive license to use the Program.

You may 1) use the Program to the extent of authorizations you have acquired and 2) make and install copies to support the level of use authorized, providing you reproduce the copyright notice and any other legends of ownership on each copy, or partial copy, of the Program.

If you acquire this Program as a program upgrade, your authorization to use the Program from which you upgraded is terminated.

You will ensure that anyone who uses the Program does so only in compliance with the terms of this Agreement.

You may not 1) use, copy, modify, or distribute the Program except as provided in this Agreement; 2) reverse assemble, reverse compile, or otherwise translate the Program except as specifically permitted by law without the possibility of contractual waiver; or 3) sublicense, rent, or lease the Program.

Transfer of Rights and Obligations

You may transfer all your license rights and obligations under a Proof of Entitlement for the Program to another party by transferring the Proof of Entitlement and a copy of this Agreement and all documentation. The transfer of your license rights and obligations terminates your authorization to use the Program under the Proof of Entitlement.

2. Proof of Entitlement

The Proof of Entitlement for this Program is evidence of your authorization to use this Program and of your eligibility for warranty services, future upgrade program prices (if announced), and potential special or promotional opportunities.

3. Charges and Taxes

IBM defines use for the Program for charging purposes and specifies it in the Proof of Entitlement. Charges are based on extent of use authorized. If you wish to increase the extent of use, notify IBM or its reseller and pay any applicable charges. IBM does not give refunds or credits for charges already due or paid.

If any authority imposes a duty, tax, levy or fee, excluding those based on IBM's net income, upon the Program supplied by IBM under this Agreement, then you agree to pay that amount as IBM specifies or supply exemption documentation.

4. Limited Warranty

© Copyright IBM Corp. 2000, 2001

IBM warrants that when the Program is used in the specified operating environment it will conform to its specifications. IBM does not warrant uninterrupted or error-free operation of the Program or that we will correct all Program defects. You are responsible for the results obtained from the use of the Program. The warranty period for the Program expires when its Program services are no longer available. The License Information specifies the duration of Program services.

During the warranty period warranty service is provided without charge for the unmodified portion of the Program through defect-related Program services. Program services are available for at least one year following the Program's general availability. Therefore, the duration of warranty service depends on when you obtain your license. If the Program does not function as warranted during the first year after you obtain your license and IBM is unable to resolve the problem by providing a correction, restriction, or bypass, you may return the Program to the party (either IBM or its reseller) from whom you acquired it and receive a refund in the amount you paid for it. To be eligible, you must have acquired the Program while Program services (regardless of the remaining duration) were available for it.

THESE WARRANTIES ARE YOUR EXCLUSIVE WARRANTIES AND REPLACE ALL OTHER WARRANTIES OR CONDITIONS, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

These warranties give you specific legal rights, and you may also have other rights which vary from jurisdiction to jurisdiction. Some jurisdictions do not allow the exclusion or limitation of implied warranties, so the above exclusion or limitation may not apply to you. In that event such warranties are limited in duration to the warranty period. No warranties apply after that period.

5. Limitation of Liability

Circumstances may arise where, because of a default on IBM's part or other liability, you are entitled to recover damages from IBM. In each such instance, regardless of the basis on which you may be entitled to claim damages from IBM, (including fundamental breach, negligence, misrepresentation, or other contract or tort claim), IBM is liable for no more than 1) damages for bodily injury (including death) and damage to real property and tangible personal property and 2) the amount of any other actual direct damages up to the greater of U.S. $100,000 (or equivalent in your local currency) or the charges for the Program that is the subject of the claim.

IBM WILL NOT BE LIABLE FOR ANY SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS), EVEN IF IBM, OR ITS RESELLER, HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

IBM will not be liable for 1) loss of, or damage to, your records or data or 2) any damages claimed by you based on any third party claim.

This limitation of liability also applies to any developer of a Program supplied to IBM. It is the maximum for which IBM and its suppliers are collectively responsible.

6. General

Nothing in this Agreement affects any statutory rights of consumers that cannot be waived or limited by contract.

IBM may terminate your license if you fail to comply with the terms of this Agreement. If IBM does so, your authorization to use the Program is also terminated.

You agree to comply with applicable export laws and regulations.

Neither you nor IBM will bring a legal action under this Agreement more than two years after the cause of action arose unless otherwise provided by local law without the possibility of contractual waiver or limitation.

Neither you nor IBM is responsible for failure to fulfill any obligations due to causes beyond its control.

The laws of the country in which you acquire the Program govern this Agreement, except 1) in Australia, the laws of the State or Territory in which the transaction is performed govern this Agreement; 2) in Albania, Armenia, Belarus, Bosnia/Herzegovina, Bulgaria, Croatia, Czech Republic, Georgia, Hungary, Kazakhstan, Kirghizia, Former Yugoslav Republic of Macedonia (FYROM), Moldova, Poland, Romania, Russia, Slovak Republic, Slovenia, Ukraine, and Federal Republic of Yugoslavia, the laws of Austria govern this Agreement; 3) in the United Kingdom, all disputes relating to this Agreement will be governed by English Law and will be submitted to the exclusive jurisdiction of the English courts; 4) in Canada, the laws in the Province of Ontario govern this Agreement; and 5) in the United States and Puerto Rico, and People's Republic of China, the laws of the State of New York govern this Agreement.

© Copyright IBM Corp. 2000, 2001, 2002

Part 2 - Country-unique Terms

AUSTRALIA:

Limited Warranty (Section 4):

The following paragraph is added to this Section:

The warranties specified in this Section are in addition to any rights you may have under the Trade Practices Act 1974 or other legislation and are only limited to the extent permitted by the applicable legislation.

Limitation of Liability (Section 5):

The following paragraph is added to this Section:

Where IBM is in breach of a condition or warranty implied by the Trade Practices Act 1974, IBM's liability is limited to the repair or replacement of the goods, or the supply of equivalent goods. Where that condition or warranty relates to right to sell, quiet possession or clear title, or the goods are of a kind ordinarily acquired for personal, domestic or household use or consumption, then none of the limitations in this paragraph apply.

EGYPT:

Limitation of Liability (Section 5):

The following replaces item 2 in the first paragraph of this Section:

2) as to any other actual direct damages, IBM's liability will be limited to the total amount you paid for the Program that is the subject of the claim.

FRANCE :

Limitation of Liability (Section 5):

The following replaces the second sentence in the first paragraph of this Section:

In such instances, regardless of the basis on which you are entitled to claim damages from IBM, IBM is liable for no more than 1) damages for bodily injury (including death) and damage to real property and tangible personal property; and 2) the amount of any other actual direct damages up to the greater of a) U.S. $100,000 (or equivalent in local currency) or b) the charges for the Program which is the subject of the claim.

GERMANY:

Limited Warranty (Section 4):

The following paragraphs are added to this Section:

The minimum warranty period for Programs is six months.

In case a Program is delivered without Specifications, we will only warrant that the Program information correctly describes the Program and that the Program can be used according to the Program information. You have to check the usability according to the Program information within the "money-back guaranty" period.

The following replaces the first sentence of the first paragraph of this Section:

The warranty for an IBM Program covers the functionality of the Program for its normal use and the Program's conformity to its Specifications.

Limitation of Liability (Section 5):

The following paragraph is added to the Section:

The limitations and exclusions specified in the Agreement will not apply to damages caused by IBM with fraud or gross negligence, and for express warranty.

In item 2, replace "U.S. $100,000" with "DEM 1.000.000".

The following sentence is added to the end of item 2 of the first paragraph:

© Copyright IBM Corp. 2000, 2001

IBM's liability under this item is limited to the violation of essential contractual terms in cases of ordinary negligence.

INDIA:

Limitation of Liability (Section 5):

The following replaces items 1 and 2 in the first paragraph:

1) liability for bodily injury (including death) or damage to real property and tangible personal property will be limited to that caused by IBM's negligence; and 2) as to any other actual damage arising in any situation involving nonperformance by IBM pursuant to, or in any way related to the subject of this Agreement, IBM's liability will be limited to the charge paid by you for the individual Program that is the subject of the claim.

General (Section 6):

The following replaces the fourth paragraph of this Section:

If no suit or other legal action is brought, within two years after the cause of action arose, in respect of any claim that either party may have against the other, the rights of the concerned party in respect of such claim will be forfeited and the other party will stand released from its obligations in respect of such claim.

IRELAND:

Limited Warranty (Section 4):

The following paragraph is added to this Section:

Except as expressly provided in these terms and conditions, all statutory conditions, including all warranties implied, but without prejudice to the generality of the foregoing, all warranties implied by the Sale of Goods Act 1893 or the Sale of Goods and Supply of Services Act 1980 are hereby excluded.

Limitation of Liability (Section 5):

The following replaces items 1 and 2 in the first paragraph of this Section:

1) death or personal injury or physical damage to your real property solely caused by IBM's negligence; and 2) the amount of any other actual direct damages, up to the greater of Irish Pounds 75,000 in respect of Programs or 125 percent of the charges for the Program that is the subject of the claim or which otherwise gives rise to the claim.

The following paragraph is added at the end of this Section:

IBM's entire liability and your sole remedy, whether in contract or in tort, in respect of any default will be limited to damages.

ITALY:

Limitation of Liability (Section 5):

The following replaces the second sentence in the first paragraph:

In each such instance unless otherwise provided by mandatory law, IBM is liable for no more than damages for bodily injury (including death) and damage to real property and tangible personal property and 2) as to any other actual damage arising in all situations involving non-performance by IBM pursuant to, or in any way related to the subject matter of this Agreement, IBM's liability, will be limited to the total amount you paid for the Program that is the subject of the claim.

NEW ZEALAND:

Limited Warranty (Section 4):

The following paragraph is added to this Section:

The warranties specified in this Section are in addition to any rights you may have under the Consumer Guarantees Act 1993 or other legislation which cannot be excluded or limited. The Consumer Guarantees Act 1993 will not apply in respect of any goods or services which IBM provides, if you require the goods or services for the purposes of a business as defined in that Act.

Limitation of Liability (Section 5):

© Copyright IBM Corp. 2000, 2001, 2002

The following paragraph is added to this Section:

Where Programs are not acquired for the purposes of a business as defined in the Consumer Guarantees Act 1993, the limitations in this Section are subject to the limitations in that Act.

PEOPLE'S REPUBLIC OF CHINA:

Charges (Section 3):

The following paragraph is added to the Section:

All banking charges incurred in the People's Republic of China will be borne by you and those incurred outside the People's Republic of China will be borne by IBM.

UNITED KINGDOM:

Limitation of Liability (Section 5):

The following replaces items 1 and 2 in the first paragraph of this Section:

1) death or personal injury or physical damage to your real property solely caused by IBM's negligence; 2) the amount of any other actual direct damages, up to the greater of Pounds Sterling 75,000 in respect of Programs or 125 percent of the charges for the Program that is the subject of the claim or which otherwise gives rise to the claim.

The following item is added:

3) breach of IBM's obligations implied by Section 12 of the Sale of Goods Act 1979 or Section 2 of the Supply of Goods and Services Act 1982.

The following paragraph is added at the end of this Section:

IBM's entire liability and your sole remedy, whether in contract or in tort, in respect of any default will be limited to damages.

Z125-3301-10 (10/97)

End of Document

© Copyright IBM Corp. 2000, 2001