

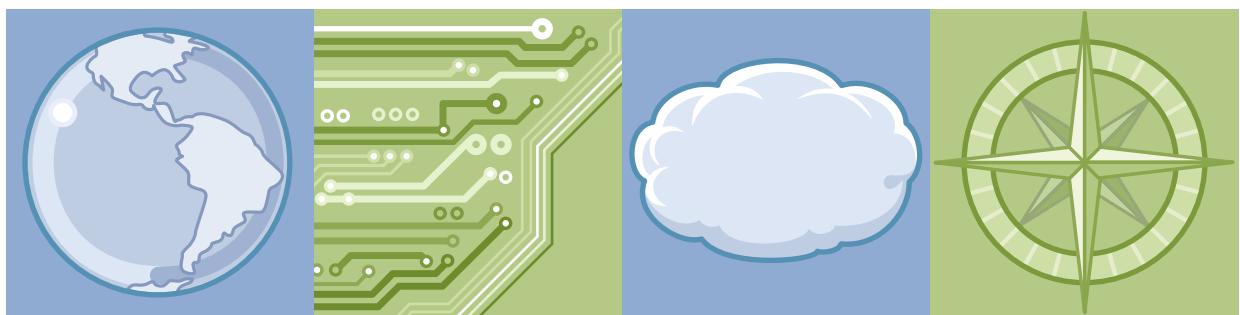


# IBM Training

Student Notebook

## IBM MQ V8 System Administration (using Linux for labs)

Course code WM209 ERC 1.0



WebSphere Education

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	CICS®	FFST™
First Failure Support Technology™	HACMP™	iSeries®
PartnerWorld®	QMFT™	System i®
System z®	Tivoli®	WebSphere®
z/OS®		

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

### June 2014 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

© Copyright International Business Machines Corporation 2014.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Trademarks .....</b>	<b>xiii</b>
<b>Course description.....</b>	<b>xvii</b>
<b>Agenda .....</b>	<b>xix</b>
<b>Unit 1. IBM MQ review .....</b>	<b>1-1</b>
Unit objectives .....	1-2
IBM MQ functional overview .....	1-3
Messages .....	1-4
Queues .....	1-5
Queue manager .....	1-6
Requesting and responding applications .....	1-7
Transmission queues and channels .....	1-8
Parallel processing .....	1-9
Asynchronous processing .....	1-10
IBM MQ client .....	1-11
Advantages of messaging with IBM MQ .....	1-12
Publish/subscribe integration .....	1-13
IBM MQ publications .....	1-14
Unit summary .....	1-15
Checkpoint questions .....	1-16
Checkpoint answers .....	1-17
<b>Unit 2. Installing IBM MQ.....</b>	<b>2-1</b>
Unit objectives .....	2-2
Before the installation .....	2-3
Supported server and client operating systems (1 of 2) .....	2-4
Supported server and client operating systems (2 of 2) .....	2-5
Multiple version installation .....	2-6
Multiple version installation commands .....	2-8
Multiple version installation commands examples .....	2-9
Choosing an installation name .....	2-10
Choosing an installation location .....	2-12
Choosing what to install .....	2-13
Preparing the system .....	2-14
MQ installation .....	2-15
Finding more information .....	2-16
Installing MQ on Windows (1 of 4) .....	2-17
Installing MQ on Windows (2 of 4) .....	2-18
Installing MQ on Windows (3 of 4) .....	2-19
Installing MQ on Windows (4 of 4) .....	2-20
Installing multiple instances of MQ .....	2-21
WebSphere MQ SupportPacs .....	2-22
SupportPac example: MSOP .....	2-23
IBM MQ product documentation .....	2-24
Uninstalling MQ .....	2-25
Unit summary .....	2-26

Checkpoint questions .....	2-27
Checkpoint answers .....	2-28
<b>Unit 3. Creating a basic configuration .....</b>	<b>3-1</b>
Unit objectives .....	3-2
Administration interfaces .....	3-3
IBM MQ command modes .....	3-5
Creating queue managers .....	3-7
Starting and deleting a queue manager .....	3-9
Naming IBM MQ objects .....	3-10
End queue manager .....	3-11
Using MQSC to configure IBM MQ objects .....	3-12
Starting the MQSC command interface .....	3-13
Stopping the MQSC command interface .....	3-14
MQSC command rules .....	3-15
MQSC script files .....	3-16
Running MQSC script files .....	3-18
Defining a local queue .....	3-19
Defining a local queue examples .....	3-20
Defining other queue types .....	3-21
Clearing and deleting queues .....	3-23
Displaying attributes .....	3-24
Altering attributes .....	3-25
Defining special queues .....	3-26
Defining a dead-letter queue .....	3-28
Unit summary .....	3-29
Checkpoint questions (1 of 2) .....	3-30
Checkpoint questions (2 of 2) .....	3-31
Checkpoint answers .....	3-32
Exercise 1 .....	3-33
Exercise objectives .....	3-34
<b>Unit 4. Introduction to IBM MQ Explorer .....</b>	<b>4-1</b>
Unit objectives .....	4-2
IBM MQ Explorer .....	4-3
Starting IBM MQ Explorer .....	4-4
Welcome page .....	4-5
IBM MQ Explorer default views .....	4-6
Creating a local queue manager (1 of 5) .....	4-7
Creating a local queue manager (2 of 5) .....	4-8
Creating a local queue manager (3 of 5) .....	4-9
Creating a local queue manager (4 of 5) .....	4-10
Creating a local queue manager (5 of 5) .....	4-11
Queue manager content view .....	4-12
Grouping queue managers .....	4-13
Creating a queue manager set (1 of 7) .....	4-14
Creating a queue manager set (2 of 7) .....	4-15
Creating a queue manager set (3 of 7) .....	4-16
Creating a queue manager set (4 of 7) .....	4-17
Creating a queue manager set (5 of 7) .....	4-18
Creating a queue manager set (6 of 7) .....	4-19

Creating a queue manager set (7 of 7) . . . . .	4-20
Creating a local queue (1 of 3) . . . . .	4-21
Creating a local queue (2 of 3) . . . . .	4-22
Creating a local queue (3 of 3) . . . . .	4-23
Shortcut icons . . . . .	4-24
Queue content view . . . . .	4-25
Queue status information . . . . .	4-26
Viewing queue filters . . . . .	4-27
Managing queue filters . . . . .	4-28
Adding queue filters . . . . .	4-29
Queue filter attributes . . . . .	4-30
Comparing queues (1 of 2) . . . . .	4-31
Comparing queues (2 of 3) . . . . .	4-32
IBM MQ object definition tests . . . . .	4-33
IBM MQ General tests . . . . .	4-34
Running the IBM MQ tests . . . . .	4-35
IBM MQ test results example . . . . .	4-36
IBM MQ Explorer preferences . . . . .	4-37
Context-sensitive help . . . . .	4-38
Unit summary . . . . .	4-39
Checkpoint questions . . . . .	4-40
Checkpoint answers . . . . .	4-41
Exercise 2 . . . . .	4-42
Exercise objectives . . . . .	4-43
<b>Unit 5. Introduction to the Message Queue Interface (MQI)</b> . . . . .	<b>5-1</b>
Unit objectives . . . . .	5-2
IBM MQ Message Queue Interface (MQI) . . . . .	5-3
Basic MQI calls . . . . .	5-4
Basic MQI calls in use . . . . .	5-6
MQI notation examples . . . . .	5-7
Order of retrieving messages . . . . .	5-8
Browsing messages in IBM MQ Explorer (1 of 2) . . . . .	5-10
Browsing messages in IBM MQ Explorer (2 of 2) . . . . .	5-11
Message properties . . . . .	5-12
General message properties . . . . .	5-13
Report message properties . . . . .	5-14
Context message properties . . . . .	5-15
Identifiers message properties . . . . .	5-16
Segmentation message properties . . . . .	5-17
Data message properties . . . . .	5-18
Message property preferences in IBM MQ Explorer . . . . .	5-19
Message property preferences: Examples . . . . .	5-20
Distribution lists . . . . .	5-21
Asynchronous message reception: Callback . . . . .	5-23
Testing IBM MQ configuration . . . . .	5-24
Testing with IBM MQ sample programs . . . . .	5-25
Sample program example . . . . .	5-27
Testing with IBM MQ Explorer . . . . .	5-28
Testing with RFHUtil (SupportPac IH03) . . . . .	5-29
Unit summary . . . . .	5-30

Checkpoint questions .....	5-31
Checkpoint answers .....	5-32
Exercise 3 .....	5-33
Exercise objectives .....	5-34
 <b>Unit 6. Implementing trigger messages and monitors .....</b>	 <b>6-1</b>
Unit objectives .....	6-2
What is triggering? .....	6-3
Triggering .....	6-4
Components of triggering .....	6-5
Queue attributes that control triggering .....	6-6
Defining triggering properties in IBM MQ Explorer .....	6-7
Process attributes .....	6-8
Defining a process in IBM MQ Explorer .....	6-9
Triggering conditions .....	6-10
Fields in the trigger message .....	6-12
Running a trigger monitor .....	6-14
Defining a trigger monitor in IBM MQ Explorer .....	6-15
Trigger monitor service properties .....	6-16
Starting the trigger monitor in IBM MQ Explorer .....	6-17
Testing a trigger monitor with IBM MQ samples .....	6-18
Trigger monitor messages .....	6-19
Triggering program problems .....	6-20
Unit summary .....	6-21
Checkpoint questions .....	6-22
Checkpoint answers .....	6-23
Exercise 4 .....	6-24
Exercise objectives .....	6-25
 <b>Unit 7. Message persistence and recovery .....</b>	 <b>7-1</b>
Unit objectives .....	7-2
Message persistence .....	7-3
Defining message persistence on queues .....	7-4
Types of logs .....	7-6
Circular logging .....	7-7
Linear logging .....	7-8
Estimating the size of the log file .....	7-10
Defining default log settings in IBM MQ Explorer .....	7-11
Recovering persistent messages .....	7-12
Dumping the log .....	7-13
Damaged objects and media recovery .....	7-14
Writing an image to a log .....	7-15
Re-creating an object from an image .....	7-16
Valid object types for recording and re-creating images .....	7-17
Multiple, asynchronous units of work .....	7-18
Sync point control .....	7-20
Compensating transactions .....	7-21
Coordinating units of work .....	7-22
Database coordination .....	7-23
External coordination of global units of work .....	7-24
Unit summary .....	7-25

Checkpoint questions .....	7-26
Checkpoint answers .....	7-27
Exercise 5 .....	7-28
Exercise objectives .....	7-29
<b>Unit 8. Identifying the cause of problems .....</b>	<b>8-1</b>
Unit objectives .....	8-2
Where is the message? .....	8-3
Finding out why a message is on the dead-letter queue .....	8-4
Missing message checklist .....	8-5
Step 1: Checking MQ reason codes .....	8-6
Step 2: Checking for MQ network problems .....	8-7
Step 3: Checking message persistence .....	8-9
Step 4: Checking message expiration .....	8-10
Step 5: Uncommitted messages .....	8-11
Displaying the queue status .....	8-12
First Failure Support Technology (FFST) .....	8-13
FFST files .....	8-14
FFST report example .....	8-15
IBM MQ error logs .....	8-16
Example error log contents .....	8-17
IBM MQ AMQ messages .....	8-18
AMQ message example .....	8-20
Application activity trace .....	8-21
mqat.ini (1 of 2) .....	8-23
mqat.ini (2 of 2) .....	8-24
Extract from an activity trace .....	8-25
Tracing IBM MQ components .....	8-26
Trace commands .....	8-27
Enabling trace in IBM MQ Explorer .....	8-28
Example trace on IBM MQ for Solaris .....	8-29
Configuration files and problem determination .....	8-30
Stopping a queue manager manually .....	8-31
Removing a queue manager manually .....	8-32
Channel problem determination .....	8-33
Remote administration failures .....	8-34
Channel triggering problems .....	8-35
Unit summary .....	8-36
Checkpoint questions .....	8-37
Checkpoint answers .....	8-38
Exercise 6 .....	8-39
Exercise objectives .....	8-40
<b>Unit 9. Implementing distributed queuing .....</b>	<b>9-1</b>
Unit objectives .....	9-2
Message channel .....	9-3
Distributed component overview .....	9-5
Distributed queuing overview .....	9-7
Distributed queuing components .....	9-8
Basic channel types .....	9-10
Sender-receiver channels .....	9-11

Requester-server channels . . . . .	9-12
Requester-sender channels . . . . .	9-13
Choosing a transmission queue . . . . .	9-14
Example of using a default transmission queue . . . . .	9-15
Transmission queue header . . . . .	9-16
Channel initiators and listeners . . . . .	9-17
Channel-exit programs . . . . .	9-18
Configuration file stanzas for distributed queuing . . . . .	9-20
Example queue manager configuration file stanzas . . . . .	9-21
Minimum channel definitions for remote communication . . . . .	9-22
Define channel command . . . . .	9-23
Defining channels example . . . . .	9-25
Defining channels in IBM MQ Explorer (1 of 2) . . . . .	9-26
Defining channels in IBM MQ Explorer (2 of 2) . . . . .	9-27
Controlling the listener process . . . . .	9-28
Listener object . . . . .	9-29
Starting a message channel . . . . .	9-30
Reasons why a channel might fail to start . . . . .	9-32
Channel initiator . . . . .	9-33
Channel control parameters . . . . .	9-34
Channel initiator triggering . . . . .	9-36
Channel states: DISPLAY CHSTATUS command (1 of 2) . . . . .	9-37
Channel states: DISPLAY CHSTATUS command (2 of 2) . . . . .	9-38
Queue definitions for distributed queuing . . . . .	9-39
Define remote queue (QREMOTE) . . . . .	9-40
Using IBM MQ Explorer to define a remote queue . . . . .	9-41
Managing a remote queue manager with IBM MQ Explorer . . . . .	9-42
Methods for accessing a remote queue manager . . . . .	9-43
Multi-hopping . . . . .	9-44
Channel sharing . . . . .	9-45
Using different channels . . . . .	9-46
Using a queue manager alias . . . . .	9-47
Separating message flows . . . . .	9-48
When a message arrives at a queue manager . . . . .	9-50
Dead-letter queue (1 of 2) . . . . .	9-51
Dead-letter queue (2 of 2) . . . . .	9-52
Using dead-letter queues . . . . .	9-53
Dead-letter header (MQDLH) . . . . .	9-54
Data conversion . . . . .	9-55
Data representation . . . . .	9-56
Data representation fields in the MQMD . . . . .	9-57
Requesting application data conversion options . . . . .	9-58
What application data conversion can be done . . . . .	9-59
Unit summary . . . . .	9-60
Checkpoint questions . . . . .	9-61
Checkpoint answers . . . . .	9-62
Exercise 7 . . . . .	9-63
Exercise objectives . . . . .	9-64
 <b>Unit 10. Implementing basic security . . . . .</b>	 <b>10-1</b>
Unit objectives . . . . .	10-2

Security mechanisms .....	10-3
IBM MQ security implementations .....	10-4
Planning for security .....	10-5
IBM MQ access control overview .....	10-6
OAM installable service .....	10-7
Principals and groups .....	10-8
Access control with the OAM .....	10-9
OAM access control lists .....	10-10
Set or reset authorization .....	10-12
Display authorization .....	10-13
Create authorization report .....	10-14
Managing authorization in IBM MQ Explorer .....	10-15
Queue manager authorization in IBM MQ Explorer .....	10-16
Queue authorization in IBM MQ Explorer .....	10-17
Channel authorization in IBM MQ Explorer .....	10-18
Access control for IBM MQ control programs .....	10-19
Security and distributed queuing .....	10-20
Security authorization for remote queues .....	10-21
Authorization checking in the MQI .....	10-22
Channel authentication control .....	10-24
Channel authentication commands .....	10-25
Channel authentication example .....	10-26
IBM MQ security exits .....	10-27
Connection authentication .....	10-28
Enabling connection authentication on a queue manager .....	10-29
Enabling connection authentication examples .....	10-30
Secure sockets layer (SSL) .....	10-32
IBM MQ SSL support .....	10-33
Enabling SSL on the queue manager .....	10-34
Enabling SSL by using IBM MQ Explorer .....	10-35
Channel attributes for SSL .....	10-36
Default security configuration in IBM MQ Explorer .....	10-37
Storing IBM MQ Explorer passwords .....	10-38
Unit summary .....	10-39
Checkpoint questions .....	10-40
Checkpoint answers .....	10-41
Exercise 8 .....	10-42
Exercise objectives .....	10-43
<b>Unit 11. IBM MQ clients .....</b>	<b>11-1</b>
Unit objectives .....	11-2
IBM MQ client .....	11-3
IBM MQ clients explained .....	11-4
Sync point control on a base client .....	11-5
Extended transactional client .....	11-6
Client configuration methods .....	11-7
Method 1: Using MQSERVER .....	11-8
Defining an MQI channel for a client connection .....	11-9
Method 2: Client configuration file .....	11-10
Method 3: Using a client channel definition table .....	11-11
Accessing a client channel definition table .....	11-13

Weighted selection on CLNTCONN channels .....	11-14
Channel instances .....	11-16
Instance limits on SVRCONN channels .....	11-17
Auto-definition of channels .....	11-18
IBM MQ client limitations .....	11-19
IBM MQ client security .....	11-20
Access control for an IBM MQ client .....	11-21
IBM MQ client security checks .....	11-22
Unit summary .....	11-23
Checkpoint questions .....	11-24
Checkpoint answers .....	11-25
Exercise 9 .....	11-26
Exercise objectives .....	11-27
 <b>Unit 12. Backing up and restoring IBM MQ object definitions</b> .....	 12-1
Unit objectives .....	12-2
Overview of the IBM MQ file system structure .....	12-3
The need to back up .....	12-4
When to back up .....	12-5
How to back up .....	12-6
Overview of IBM MQ objects .....	12-7
Why back up IBM MQ object definitions .....	12-8
Overview of facilities available .....	12-9
Backing up resource definitions with runmqsc .....	12-10
Save IBM MQ configuration: dmpmqcfg .....	12-11
Backing up queue manager configuration .....	12-12
Using dmpmqcfg: Syntax examples .....	12-13
Using dmpmqcfg: Output MQSC .....	12-14
Using dmpmqcfg: Output setmqaut .....	12-15
Overview of security definitions .....	12-16
Need to back up security definitions .....	12-17
Backing up security definitions .....	12-18
dmpmqaut command .....	12-19
Backing up security definitions in IBM MQ Explorer .....	12-20
Unit summary .....	12-21
Checkpoint questions .....	12-22
Checkpoint answers .....	12-23
Exercise 10 .....	12-24
Exercise objectives .....	12-25
 <b>Unit 13. Monitoring and configuring IBM MQ for performance</b> .....	 13-1
Unit objectives .....	13-2
Collecting statistics and accounting data .....	13-3
MQI statistics .....	13-4
Queue statistics .....	13-5
Channel statistics .....	13-6
Flush statistics .....	13-8
IBM MQ accounting data collection .....	13-9
IBM MQ queue accounting data .....	13-11
Queue manager accounting monitoring .....	13-12
Queue manager statistics monitoring .....	13-13

Queue and channel statistics configuration .....	13-14
Displaying statistics and accounting data .....	13-15
Examples of accounting output .....	13-16
Display accounting and statistics .....	13-17
MSOP accessing statistics and accounting data (1 of 2) .....	13-18
MSOP accessing statistics and accounting data (2 of 2) .....	13-19
MSOP: Example statistics data .....	13-20
MSOP: Example accounting data .....	13-21
MSOP: Example queue accounting data .....	13-22
MSOP command-line interface .....	13-23
Online monitoring .....	13-24
Configuring and tuning IBM MQ for performance .....	13-25
Tuning options for message types .....	13-26
Queue manager log .....	13-27
Default log settings in MQ Explorer .....	13-28
Log file location .....	13-29
Log write integrity level .....	13-30
Type of logging .....	13-31
Log file pages .....	13-32
Log buffer pages .....	13-33
Number of log files .....	13-34
Queue manager channels performance tuning .....	13-35
Queue manager listeners performance tuning .....	13-37
Queue buffer sizes .....	13-38
Changing queue buffer sizes .....	13-39
Unit summary .....	13-40
Checkpoint questions .....	13-41
Checkpoint answers .....	13-42
Exercise 11 .....	13-43
Exercise objectives .....	13-44
<b>Unit 14. JMS administration overview .....</b>	<b>14-1</b>
Unit objectives .....	14-2
What is JMS? .....	14-3
JMS architecture .....	14-4
JMS support in IBM MQ .....	14-6
IBM MQ classes for JMS or JMS .....	14-7
IBM JMS extensions .....	14-8
IBM MQ JMS administration utility .....	14-9
MQ Explorer administration for JMS .....	14-10
Create JMS and MQ queues simultaneously .....	14-11
JMS queue properties .....	14-12
Unit summary .....	14-13
Checkpoint questions .....	14-14
Checkpoint answers .....	14-15
<b>Unit 15. Course summary .....</b>	<b>15-1</b>
Unit objectives .....	15-2
Course learning objectives .....	15-3
To learn more on the subject .....	15-4
Unit summary .....	15-5

<b>Appendix A. List of abbreviations.....</b>	<b>A-1</b>
<b>Appendix B. Resource guide.....</b>	<b>B-1</b>

# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	CICS®	FFST™
First Failure Support Technology™	HACMP™	iSeries®
PartnerWorld®	QMFT™	System i®
System z®	Tivoli®	WebSphere®
z/OS®		

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.







# Course description

## IBM MQ V8 System Administration (using Linux for labs)

**Duration:** 4 days

### Purpose

This 4-day course is designed to provide technical professionals with the skills needed to administer IBM MQ queue managers on distributed platforms such as Windows 2008 Server and Linux.

In this course, you learn how to install, customize, operate, administer, and monitor IBM MQ on distributed platforms. The course covers topics such as configuration, day-to-day administration, problem recovery, and managing security and performance.

In addition to the instructor-led lectures, you participate in hands-on lab exercises that are designed to reinforce lecture content. The lab exercises use IBM MQ V8.0, giving you practical experience with tasks such as handling queue recovery, implementing security, and problem determination.

### Audience

This course is designed for technical professionals who require the skills to administer any of the IBM MQ queue managers except IBM MQ for z/OS or IBM MQ for iSeries.

### Prerequisites

Before taking this course, you should:

- Have a basic knowledge of IBM MQ V8 concepts, either through successful completion of course WM102, *Technical Introduction to IBM MQ*, or through equivalent practical experience
- Be familiar with and able to use standard functions within the operating system that is used in the lab exercises
- Have a basic knowledge of TCP/IP

### Objectives

After completing this course, you should be able to:

- Plan the implementation of IBM MQ
- Install IBM MQ
- Perform basic customization and administration tasks

- Enable a queue manager to exchange messages with another queue manager
- Use a trigger message and a trigger monitor to start an application to process messages
- Implement basic queue manager restart and recovery procedures
- Identify the cause of a problem
- Plan for and implement basic IBM MQ security features
- Use accounting and statistics messages to monitor the activities in an IBM MQ system
- Use IBM MQ sample programs to test queue manager, queue, channel, and client configuration

## **Curriculum relationship**

This course replaces course WM204, *IBM WebSphere MQ V7.5 System Administration (using Windows for labs)*.

Course WM102, *A Technical Introduction to IBM MQ V8.0* is a prerequisite for this course.

This course is a prerequisite for:

- WM212, *IBM MQ V8 Advanced System Administration*
- WM252, *Designing and Architecting Clustering Solutions for IBM MQ V8*

# Agenda

## Day 1

- Course introduction
- Unit 1 - IBM MQ review
- Unit 2 - Installing IBM MQ
- Unit 3 - Creating a basic configuration
- Exercise 1 - Using commands to create queue managers and queues
- Unit 4 - Introduction to IBM MQ Explorer
- Exercise 2 - Using IBM MQ Explorer to create queue managers and queues

## Day 2

- Unit 5 - Introduction to the Message Queue Interface (MQI)
- Exercise 3 - Using IBM MQ sample programs to test the configuration
- Unit 6 - Implementing trigger messages and monitors
- Exercise 4 - Implementing a trigger monitor
- Unit 7 - Message persistence and recovery
- Exercise 5 - Using a media image to restore a queue
- Unit 8 - Identifying the cause of problems

## Day 3

- Exercise 6 - Running an IBM MQ trace
- Unit 9 - Implementing distributed queuing
- Exercise 7 - Connecting queue managers
- Unit 10 - Implementing basic security
- Exercise 8 - Controlling access to IBM MQ
- Unit 11 - IBM MQ clients

## Day 4

- Exercise 9 - Connecting an IBM MQ client
- Unit 12 - Backing up and restoring IBM MQ object definitions
- Exercise 10 - Backing up and restoring IBM MQ object definitions
- Unit 13 - Monitoring and configuring IBM MQ for performance
- Exercise 11 - Monitoring and configuring IBM MQ for performance
- Unit 14 - JMS administration overview
- Unit 15 - Course summary



# Unit 1. IBM MQ review

## What this unit is about

This unit reviews IBM MQ terminology and basic concepts.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the features and benefits of IBM MQ
- Identify IBM MQ components and their functions

## How you will check your progress

- Checkpoint questions

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Describe the features and benefits of IBM MQ
- Identify IBM MQ components and their functions

© Copyright IBM Corporation 2014

---

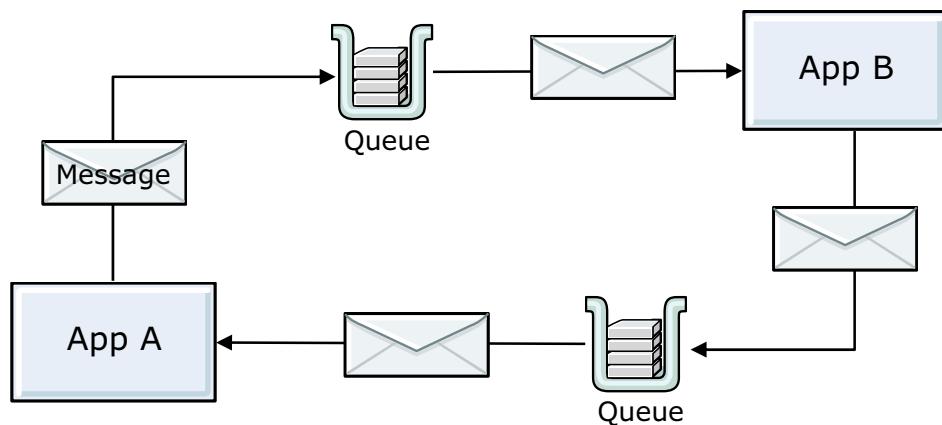
Figure 1-1. Unit objectives

WM2091.0

### Notes:

## IBM MQ functional overview

- Common application programming interface
- Assured message delivery
- Time-independent processing
- Application parallelism
- Faster application development



© Copyright IBM Corporation 2014

Figure 1-2. IBM MQ functional overview

WM2091.0

### Notes:

IBM MQ uses messages and queues to provide program-to-program communication. The communicating applications can be on the same system, or distributed across a network of IBM and non-IBM systems.

The major benefits of IBM MQ are:

- There is a common application programming interface, the Message Queue Interface (MQI) that is consistent across all the supported operating systems.
- IBM MQ can transfer data with assured delivery; messages do not get lost, even in the event of a system failure. There is no duplicate delivery.
- Communicating applications do not have to be active at the same time.
- An application is divided into discrete functional modules that communicate with each other with messages. In this way, the modules can run on different systems, be scheduled at different times, or they can act in parallel.
- Application development is faster because the developer is shielded from the complexities of the network.

## Messages

- Contain the application data or payload
- Are placed on queues, which allow programs to run independently of each other, at different speeds and times, in different locations, and without a logical connection between them
- Contain MQ message descriptor (MQMD) with control information
- Can contain optional message properties that are created by the application



Message descriptor (MQMD)	Message properties (Optional)	Application data
------------------------------	----------------------------------	------------------

© Copyright IBM Corporation 2014

Figure 1-3. Messages

WM2091.0

### Notes:

A message is any information that one application wants to communicate to another. A message can convey a request for a service, or it can be a reply to such a request. It might also report on the progress of another message; to confirm its arrival or report on an error, for example. A message can also carry information for which no reply is expected.

A message consists of two parts:

- Message descriptor
- Application data

The message descriptor contains information about the message. The sending application supplies both the message descriptor and the application data when it puts a message on a queue. Both the message descriptor and the application data are returned to the application that gets the message from the queue. The application that puts the messages on the queue sets some of the fields in the message descriptor; the queue manager sets others on behalf of the application.

## Queues

- A defined end-point destination for messages
  - *Local queue* is owned by the queue manager to which the program is connected
  - *Remote queue* is owned by a different queue manager to the one to which the program is connected
  - *Alias queue* is a pointer to a local queue or a locally owned remote queue
  - *Model queue* is a template to create a dynamic local queue



Only queues that are defined as local queues (QLOCAL) hold messages

© Copyright IBM Corporation 2014

Figure 1-4. Queues

WM2091.0

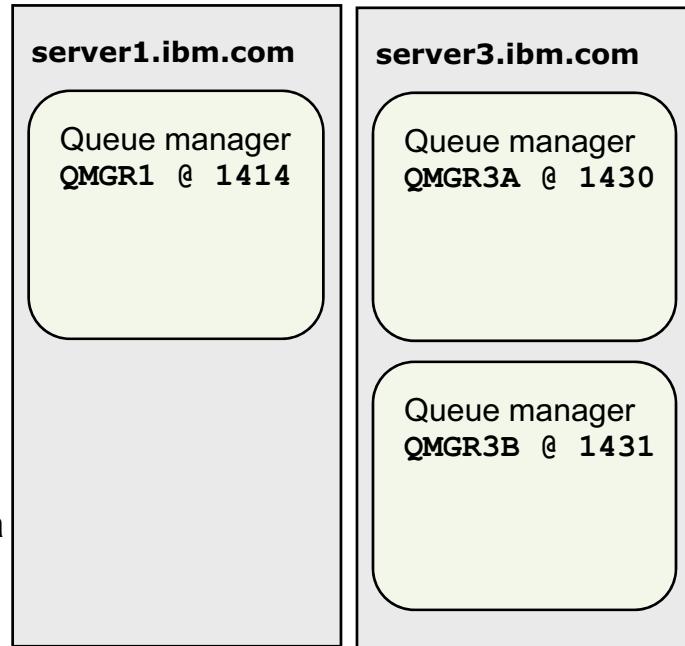
### Notes:

A *queue* is a place to store messages until they can be processed. This figure describes the type of queues the MQ supports.

The time that a message waits to be retrieved and processed can be short. Alternatively, it can be a long time if it must wait for the receiving application to start. Either way, the ability to store a message safely is an important characteristic of a queue.

## Queue manager

- Provides messaging services to applications
- Ensures that messages are sent to the correct queue or are routed to another queue manager
- Hosts the queues and the channels that connect queue managers
- A server can host more than one queue manager
- Queue managers that share a server must use different TCP/IP ports



© Copyright IBM Corporation 2014

Figure 1-5. Queue manager

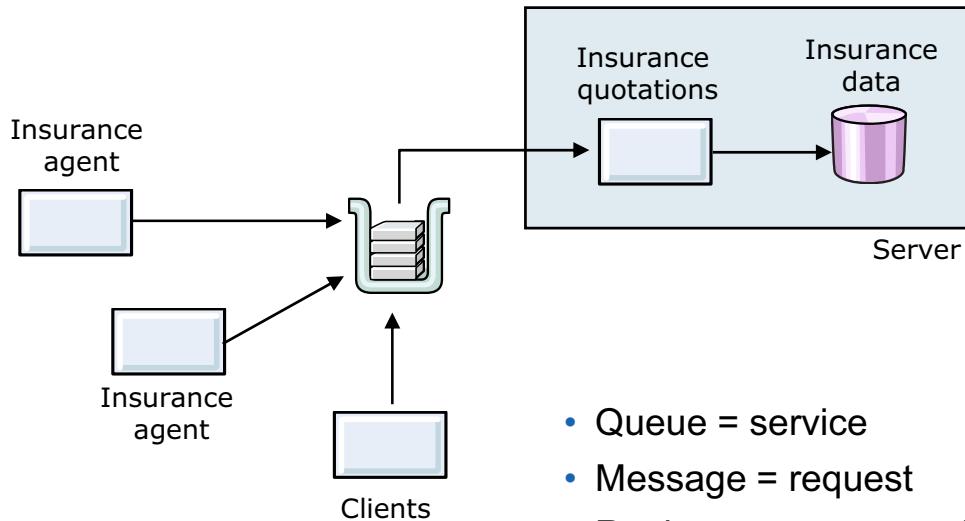
WM2091.0

### Notes:

The queue managers are the main components in an MQ messaging network. The queue managers host the other objects in the network, such as the queues and the channels that connect the queue managers together.

In a distributed environment, each queue manager is assigned a unique TCP/IP port.

## Requesting and responding applications



- Queue = service
- Message = request
- Reply-to queue name in message descriptor
- Multiple instances of server possible

© Copyright IBM Corporation 2014

Figure 1-6. Requesting and responding applications

WM2091.0

### Notes:

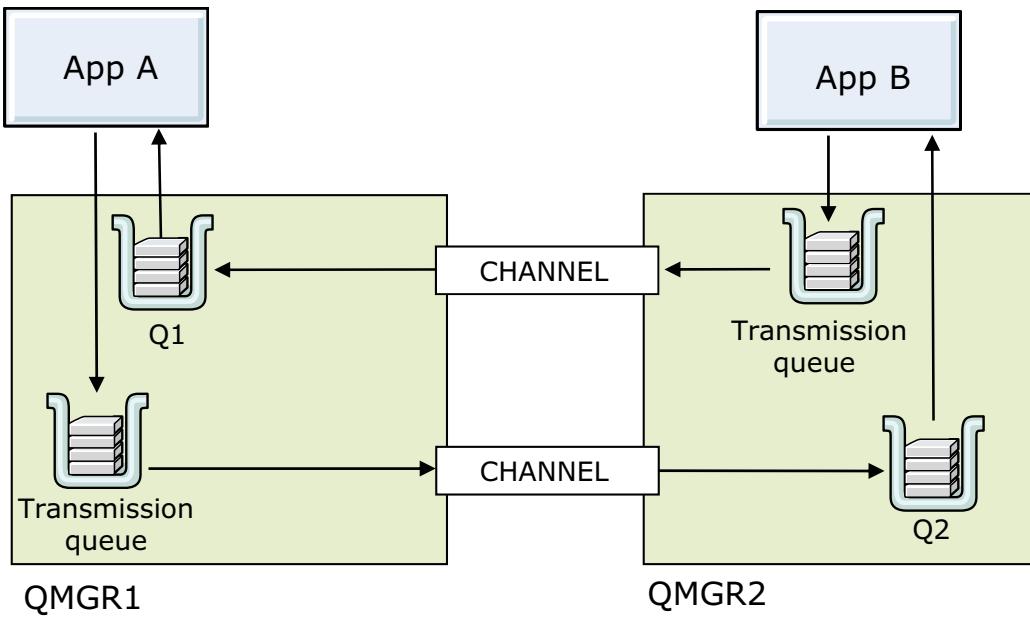
In general, multiple applications can put messages on the same queue to request the same service. The application that serves the queue gets each message and responds to it.

In this example, the client applications can put messages on the same queue. The server application that is named **Insurance quotations** gets the messages from the queue.

In this example, the message descriptor provides the name of the reply-to queue. The reply-to queue informs the server application where to put the reply message. In this way, each client application can receive its replies separately from other client applications.

The message descriptor also has a field to hold an identifier for a message. Furthermore, the message descriptor of a reply message can contain the identifier of the request message to which it relates. In this way, a client application can correlate a reply with a request it sent previously.

## Transmission queues and channels



- Transmission queue stores message first
- Application is not stopped if the link is inactive

© Copyright IBM Corporation 2014

Figure 1-7. Transmission queues and channels

WM2091.0

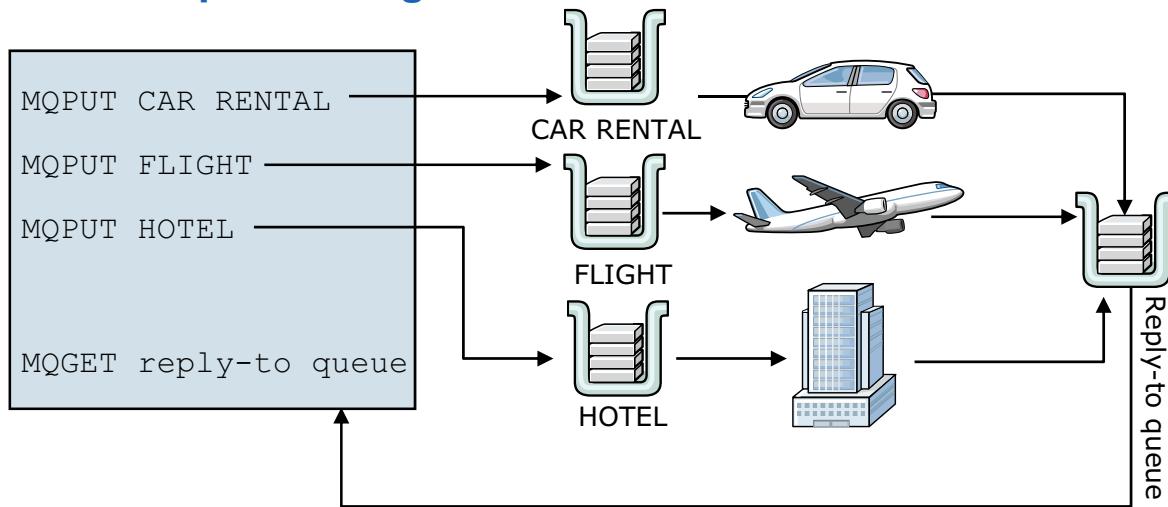
### Notes:

In many cases, communicating applications are installed on different systems. These applications still communicate with each other by using the MQI; they do not need to know that they are remote from each other.

Assured delivery is another benefit of IBM MQ. It is the result of the protocol that is used when one queue manager transmits a message to another queue manager. As far as the applications are concerned, this process of transmitting messages is asynchronous and not apparent. Furthermore, the protocol ensures that no message is lost or delivered to its destination queue more than one time.

In the example in the figure, Application A puts a message on queue Q1. The queue manager to which Application A is connected, QMGR1, knows that Q1 is a remote queue and puts the message on a transmission queue. Asynchronously, queue manager QMGR1 transmits the message to queue manager QMGR2, which puts it on queue Q2. The message then becomes available for Application B to get. When Application B puts a reply message on queue Q2, the message on the transmission queue stages the delivery of the message.

## Parallel processing



- Requests are not serialized
- Replies are consolidated
- Transactions have shorter elapsed time

© Copyright IBM Corporation 2014

Figure 1-8. Parallel processing

WM2091.0

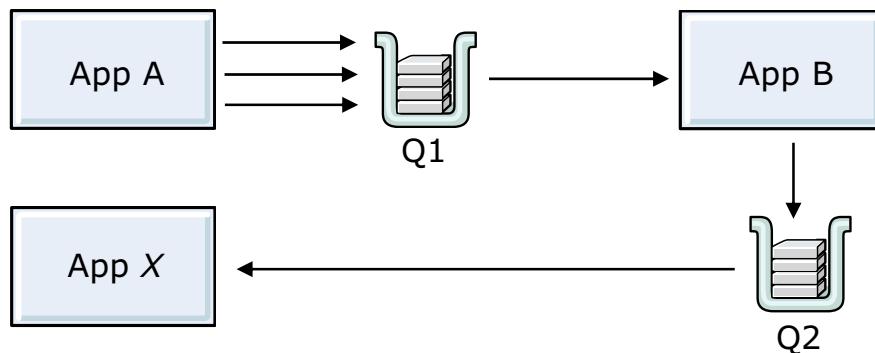
### Notes:

Parallel processing allows an application to send several requests without waiting for a reply to one request before sending the next. All the requests can then be processed in parallel.

The application can process the replies when they are all received, and produce a consolidated answer. The program logic might also specify what to do when only a partial set of replies is received within a specified period.

A more complex application might involve sending a number of requests to different servers. In the example in the figure, a travel agent uses an application to arrange a trip for a customer. The application must make a number of requests, such as making a rental car, airplane, or hotel reservation. By using queues, these steps do not have to be completed serially. The requests can be put on different queues, and processed in parallel.

## Asynchronous processing



- Separate process for replies
- No need for communicating programs to be active at the same time
- Time independence

© Copyright IBM Corporation 2014

Figure 1-9. Asynchronous processing

WM2091.0

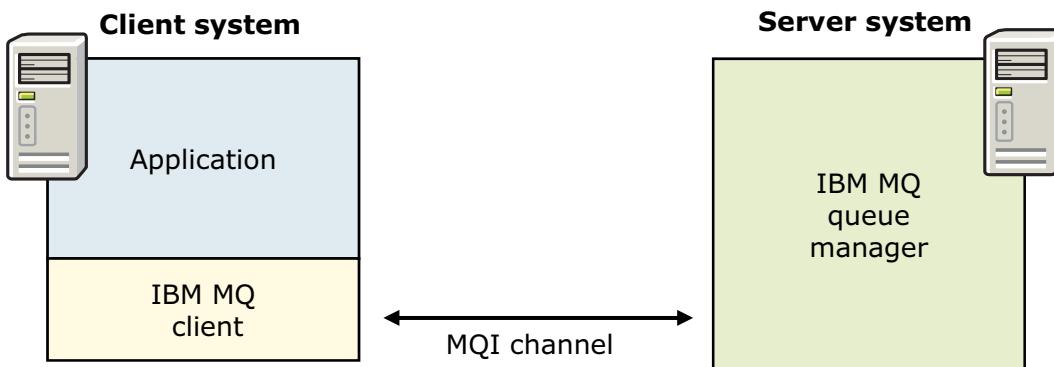
### Notes:

In the asynchronous model, instead of waiting for a reply to its first message, Application A continues to send further requests to Application B. It is a separate process, Program X, which receives the replies when they arrive.

In this model, Application A is not dependent on Application B to be running when the requests are sent. It can continue to do work even when Application B is stopped.

The application does expect Application X to receive the replies at some time, but not necessarily at the same time that Application A or Application B. This scenario illustrates another of the major benefits of IBM MQ, *time independence*.

## IBM MQ client



- IBM MQ component on a system without a queue manager
  - Communicates with the queue manager by using an *MQI channel*
  - Allows an application that runs on the client system to connect to a queue manager that is running on another system

© Copyright IBM Corporation 2014

Figure 1-10. IBM MQ client

WM2091.0

### Notes:

An IBM MQ client is a component of IBM MQ that allows an application that is running on one system to send MQI calls to a queue manager that is running on another system.

The *client connection* receives the input parameters of an MQI call from the application and sends them over a communications connection to the *server connection* on the same system as the queue manager. The server connection then sends the MQI call to the queue manager on behalf of the application. After the queue manager completes the MQI call, the server connection sends the output parameters of the callback to the client connection, which then passes them onto the application.

The combination of a client connection, a server connection, and a communications connection between them (for example, an SNA LU6.2 conversation or a TCP connection) is called an *MQI channel*.

The full range of MQI calls and options are available to a client application.



## Advantages of messaging with IBM MQ

- IBM MQ clients enable an application to connect remotely or locally to an MQ queue manager
- Publish/subscribe support increases messaging capability from point-to-point messaging to a less coupled style of messaging
- IBM MQ clusters allow multiple instances of the same service to be hosted through multiple queue managers to enable load-balancing, and fail-over and simplify administration
- Secure Sockets Layer (SSL) protocol can be used to secure communication between queue managers or IBM MQ client
- IBM MQ supports a wide range of hardware and operating systems

© Copyright IBM Corporation 2014

Figure 1-11. Advantages of messaging with IBM MQ

WM2091.0

### Notes:

Listed in the figure are some of the IBM MQ features that make communication and configuration easier.

## Publish/subscribe integration

- Publish/subscribe engine is fully integrated into the queue manager and is automatically enabled
- Publish directly to topics
- Distributed publish/subscribe allows applications that are connected to separate queue managers to use publish/subscribe messaging
  - *Publish/subscribe cluster* uses IBM MQ cluster technology that connects queue managers by using cluster channels
  - *Hierarchical publish/subscribe* topology is built on queue managers connected by using standard distributed message channels or cluster channels
- Use MQ Explorer to simplify configuration, development, and deployment of a publish/subscribe environment

© Copyright IBM Corporation 2014

Figure 1-12. Publish/subscribe integration

WM2091.0

### Notes:

IBM MQ fully supports a publish/subscribe topology. No additional software is required to implement publish/subscribe.

In the IBM MQ publish/subscribe model, the topic that the publisher associates with the information connect publishing and subscribing applications. Publishers and subscribers need to agree on the topic only to connect to one another. Each different piece of information has its own topic that is associated with it. Subscribers nominate which types of information they want to receive by subscribing to specific topics.

The performance of a publish/subscribe network can be improved by arranging the queue managers in a publish/subscribe cluster. A publish/subscribe cluster consists of a set of queue managers that are connected together, with direct channel links between all members, to form all or part of a publish/subscribe network.

Queue managers can also be grouped in a hierarchy, where the hierarchy contains one or more queue managers that are directly connected. Queue managers are connected together by using a connection-time parent and child relationship.



## IBM MQ publications

- IBM MQ product site:

<http://www.ibm.com/software/products/en/ibm-mq>

- Latest news
- References
- Beta code
- Software downloads
- SupportPacs

© Copyright IBM Corporation 2014

Figure 1-13. IBM MQ publications

WM2091.0

### Notes:

For more information about IBM MQ, see the IBM MQ product site.

Some publications describe functions that relate to two or more queue managers, the so called cross-platform publications. Other publications are platform-specific.

Discover IBM MQ on the World Wide Web. The web address for the IBM MQ home page is

<http://www.ibm.com/software/integration/wmq>

## Unit summary

Having completed this unit, you should be able to:

- Describe the features and benefits of IBM MQ
- Identify IBM MQ components and their functions

© Copyright IBM Corporation 2014

Figure 1-14. Unit summary

WM2091.0

### Notes:



## Checkpoint questions

1. True or False: IBM MQ supports asynchronous messaging only.
2. IBM MQ assured delivery means that:
  - a. A report of delivery can always be sent back.
  - b. Unless the entire system goes down, no messages are lost.
  - c. Messages can be duplicated but never lost.
  - d. Messages are delivered with no loss or duplication.
3. Applications place messages on queues by using the IBM MQ:
  - a. program-to-program interface.
  - b. message queue interface.
  - c. command processor.

© Copyright IBM Corporation 2014

---

Figure 1-15. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.

## Checkpoint answers

1. **False.** IBM MQ supports both synchronous and asynchronous messaging.
2. **d**
3. **b**

© Copyright IBM Corporation 2014

Figure 1-16. Checkpoint answers

WM2091.0

### Notes:



# Unit 2. Installing IBM MQ

## What this unit is about

This unit describes the IBM MQ installation process, which includes installation planning.

## What you should be able to do

After completing this unit, you should be able to:

- Identify the steps that are required to install IBM MQ on Windows or UNIX
- Find the hardware and software prerequisites for IBM MQ installation
- Identify the installation directory structure on Windows and UNIX
- Identify the configuration files and where to locate them on Windows and UNIX
- Locate and install IBM MQ Fix Packs
- Use IBM SupportPacs to complement the IBM MQ family of products

## How you will check your progress

- Checkpoint

## References

IBM MQ V8 product documentation

## Unit objectives

After completing this unit, you should be able to:

- Identify the steps that are required to install IBM MQ on Windows or UNIX
- Find the hardware and software prerequisites for IBM MQ installation
- Identify the installation directory structure on Windows and UNIX
- Identify the configuration files and where to locate them on Windows and UNIX
- Locate and install IBM MQ Fix Packs
- Use IBM SupportPacs to complement the IBM MQ family of products

© Copyright IBM Corporation 2014

Figure 2-1. Unit objectives

WM2091.0

### Notes:



## Before the installation

- Decide on the installation name and path
  - New installation with no previous IBM MQ installation on the computer
  - Upgrade from previous version of WebSphere MQ
- Get base code and any maintenance level updates (Fix Packs)
- Evaluate SupportPacs
- Prepare the environment
- Have a backup and fallback plan
- Consider application migration issues
- Have a migration plan and strategy
- Perform the installation

© Copyright IBM Corporation 2014

Figure 2-2. Before the installation

WM2091.0

### Notes:

The installation process for IBM MQ for Windows detects whether it is a new installation or an update from a previous level of this product. If you upgrade from a previous level of IBM MQ, back up your system before you install a new version.

## Supported server and client operating systems (1 of 2)

Platforms	Server	Client
Windows: • 8.1 Enterprise, Professional, and Standard • 7 Enterprise, Professional, and Ultimate • 8 Enterprise, Professional, and Standard • Server 2008 R2 Datacenter, Enterprise, and Standards Editions • Server 2012 R2 Datacenter and Standard Editions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Linux: • Asianux Linux 3.0 • Red Hat Enterprise Linux Server 6 • SUSE Linux Enterprise Server 11 • Ubuntu 12.04 LTS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

See the IBM MQ product website for the most current and detailed information about supported platforms

© Copyright IBM Corporation 2014

Figure 2-3. Supported server and client operating systems (1 of 2)

WM2091.0

### Notes:

The table lists the supported operating systems for IBM MQ server and client installation on Windows and Linux.

For details of the specific operating system and hardware requirements, see the IBM MQ product documentation.

## Supported server and client operating systems (2 of 2)

Platforms	Server	Client
AIX <ul style="list-style-type: none"> <li>• AIX V6.1</li> <li>• AIX V7.1</li> </ul>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Solaris 10 and 11 on SPARC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
HP-UX Itanium 11i v3 on IA64	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
IBM z/OS <ul style="list-style-type: none"> <li>• z/OS 1.13</li> <li>• z/OS 2.1 z</li> </ul>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
IBM i 7.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

See the IBM MQ product website for the most current and detailed information about supported platforms

© Copyright IBM Corporation 2014

Figure 2-4. Supported server and client operating systems (2 of 2)

WM2091.0

### Notes:

The table in the figure lists the supported IBM System z, IBM System i, AIX, Oracle Solaris, and HP-UX operating systems.

Always check the IBM MQ product documentation and the IBM MQ product website for the most current and detailed information about supported operating systems.

## Multiple version installation

- Install multiple instances and versions of IBM MQ and WebSphere MQ in the same environment on UNIX and Windows
  - Improves hardware usage
  - Aids staged migrations
- Identify *primary installation* to define which IBM MQ installation is used in system-wide locations
  - During installation
  - After installation with a command
- A specific installation owns the queue managers
  - Governs the level of function available when the queue manager is running
  - Ownership can be changed to a newer installation for migration

**Note:** You can install IBM MQ V8.0 on the same system with WebSphere MQ V7.0.1.6 and higher

© Copyright IBM Corporation 2014

Figure 2-5. Multiple version installation

WM2091.0

### Notes:

You can install more than one copy of IBM MQ on UNIX and Windows operating systems. An installation is the collection of binary libraries that make up IBM MQ.

Multiple version installation simplifies migration strategies because you can continue to use one version of IBM MQ and gradually upgrade applications to a new version, without needing parallel hardware. In addition, vendor-acquired applications can embed IBM MQ in a private directory, and can choose which versions of IBM MQ are supported, without worrying about the “visible” version of IBM MQ that a user might be using.

The installation details vary by operating system, but conceptually you first define an installation, optionally giving it a name and description. You then run the installation program, which copies the code from media onto the disk in your chosen directories.

Generally, no “default” paths are created to an installation. However, a primary installation does insert itself into default locations.

You can have a single primary installation only on a system. On Windows, one installation is always identified as the primary installation because some operating system functions, such as how Windows .dll files are registered, require a single location.

An installation owns each queue manager, which governs the level of function available when running a queue manager. When you install a newer level of code, the queue manager can be moved to that newer installation, and new functions can be used.

For easier migration, you can have an existing copy of WebSphere MQ V7.0.1.6 (or higher) on your systems. It is not necessary to upgrade before using the multiple installation capabilities. If you already have WebSphere MQ V7.0.1 installed on your system, it is always the primary installation on all operating systems.



## Multiple version installation commands

- Show installation details such as which queue managers exist and owning installations
  - **dspmq** and **dspmqinst**: Display installation information
  - **dspmqver**: Display installation version information

**Note:** Commands are not in the default PATH for UNIX

© Copyright IBM Corporation 2014

Figure 2-6. Multiple version installation commands

WM2091.0

### Notes:

IBM MQ commands can be used to examine and verify the installation information. In addition, installation details are maintained in the `/etc/opt/mqm/mqinst.ini` file on UNIX and in the registry on Windows.

The commands are not in the default PATH for UNIX. If you are not working with the primary installation, all the control commands must include an explicit path. You can also use the `setmqenv` command to automatically set up the environment for use with a different installation of MQ. The `setmqenv` command and multiple version installation is described in detail in the IBM MQ product documentation.

## Multiple version installation commands examples

```
$ /usr/mqm/bin/dspmqver -i
Name: IBM MQ
Version: 8.0.0.0
Level: p000-L110915
BuildType: IKAP - (Production)
Platform: IBM MQ for AIX
Mode: 64-bit
O/S: AIX 6.1
InstName: Installation1
InstPath: /usr/mqm
InstDesc: My default installation
DataPath: /var/mqm
Primary: Yes
MaxCmdLevel: 710

Name: IBM MQ
Version: 8.0.0.0
InstName: Installation2
InstPath: /usr/mqm2/usr/mqm
InstDesc: A second installation
Primary: No
```

**\$ dspmq -o installation**

QMNAME (V80A)	INSTNAME (Installation1)
	INSTPATH (/usr/mqm)
	INSTVER (8.0.0.0)
QMNAME (V80B)	INSTNAME (Installation1)
	INSTPATH (/usr/mqm)
	INSTVER (8.0.0.0)
QMNAME (INST2QM)	INSTNAME (Installation2)
	INSTPATH (/usr/mqm2/usr/mqm)
	INSTVER (8.0.0.0)

**\$ /usr/mqm/bin/endmqm INST2QM**

AMQ5691: Queue manager 'INST2QM' is associated with a different installation.

© Copyright IBM Corporation 2014

Figure 2-7. Multiple version installation commands examples

WM2091.0

### Notes:

The figure shows some examples of the commands that assist with installation management.

The **dspmq** and **dspmqver** commands show all of the installations, their directories, and the associated queue managers.

The queue manager control commands must be run from the correct directory; they return an error if you try to run them against a queue manager that is not associated with that installation.

Optionally, you can also use the **setmqenv** command to automatically set up the environment for a different installation of MQ.

## Choosing an installation name

- Each installation of IBM MQ on UNIX, Linux, and Windows, has a unique *installation name* that associates queue managers and configuration files with an installation
  - On Windows, you can choose the installation name during the installation process
  - On UNIX and Linux, the first IBM MQ installation is named `Installation1`; use the `crtmqinst` command for subsequent installation names before installing the product
  - Installation name cannot be changed after the product is installed
- Installation name can be up to 16 bytes and must be a combination of alphabetic and numeric characters in the ranges a-z, A-Z, and 0-9
- Use the `dspmqinst` command to find the installation name that is assigned to an installation in a particular location

© Copyright IBM Corporation 2014

Figure 2-8. Choosing an installation name

WM2091.0

### Notes:

Each installation of IBM MQ on UNIX, Linux, and Windows, has a unique identifier that is known as an *installation name*. The installation name associates objects such as queue managers and configuration files with an installation.

You can choose an installation name that is meaningful to you. For example, you might call a test system **testMQ**. An installation name can be up to 16 bytes and must be a combination of alphabetic and numeric characters in the ranges a-z, A-Z, and 0-9. You cannot use blank characters. The installation name must be unique, regardless of whether uppercase or lowercase characters are used. For example, the names **INSTALLATIONNAME** and **InstallationName** are not unique.

If you do not specify an installation name when the product is installed, a default installation name is automatically assigned. For the first installation, this name is **Installation1**. For the second installation, the name is **Installation2**. The installation name **Installation0** is reserved for an installation of WebSphere MQ version 7.0.1. The installation name cannot be changed after the product is installed.

On UNIX and Linux systems, the first IBM MQ installation is automatically given an installation name of **Installation1**. For subsequent installations, you can use the `crtmqinst` command to set the installation name before installing the product.

On Windows systems, you can choose the installation name during the installation process.

You can find out what installation name is assigned to an installation in a particular location by using the `dspmqinst` command.



## Choosing an installation location

- You can install IBM MQ to a custom location during the installation process
  - Path that is specified must either be an empty directory, the root of an unused file system, or a path that does not exist
  - Path length is limited to 256 bytes
- Default installation location for the IBM MQ on Windows
  - Software: C:\Program Files\IBM\WebSphere MQ
  - Data files and logs: C:\ProgramData\IBM\MQ
- Default installation location for IBM MQ on Linux
  - Software: /opt/mqm
  - Data: /var/mqm

© Copyright IBM Corporation 2014

Figure 2-9. Choosing an installation location

WM2091.0

### Notes:



## Choosing what to install

- Select the components or features that you require when you install IBM MQ:
  - IBM MQ server
  - IBM MQ client
  - IBM MQ Explorer
  - WebSphere Managed File Transfer components
  - Support files for Java, .NET messaging, and web services
  - Development Toolkit
  - IBM MQ Telemetry server and basic or advanced clients
  - IBM MQ Advanced Message Security
- Options vary by operating system

© Copyright IBM Corporation 2014

Figure 2-10. Choosing what to install

WM2091.0

### Notes:

## Preparing the system

- Might be necessary to complete several tasks before you install MQ depending on the installation operating system.
- On UNIX and Linux:
  - Create user ID of `mqm` with a primary group for this user of `mqm`
  - Create the file systems
  - After installation, run `mqconfig` script to compare the system kernel settings against the IBM default limits
- On Windows:
  - Ensure that the machine name does not contain any spaces
  - For MQ authorizations, ensure that the names of user IDs and groups are 20 characters or less (spaces are not allowed)
  - Default user ID of `MUSR_MQADMIN` that is used to run the MQ Windows service is created during the installation process

© Copyright IBM Corporation 2014

Figure 2-11. Preparing the system

WM2091.0

### Notes:

On UNIX and Linux, the `mqconfig` script compares the system kernel settings against the IBM default limits, which should be sufficient for many systems. However, these settings might be insufficient if you have a busy system with high production volumes, or if your system is also hosting other products like databases or application servers. Run the `mqconfig` script when your system is processing its peak volume to ensure that the actual resource usage, which is shown as a percentage, is not excessively high.

The `mqconfig` script is described in the IBM Technote “How to configure UNIX and Linux systems for IBM MQ” at <http://www.ibm.com/support/docview.wss?uid=swg21271236>

## MQ installation

Similar to installing other software on the same platform

- IBM AIX
  - SMIT
  - Easy installation
- HP-UX
  - swinstall
- Linux
  - Red Hat Package Manager
- Oracle Solaris
  - Run the supplied installation script from the CD-ROM
- Windows
  - Automatic execution of setup from CD-ROM

© Copyright IBM Corporation 2014

Figure 2-12. MQ installation

WM2091.0

### Notes:

The general rules for installation are as follows:

- Installing IBM MQ is like installing any other software on the same operating system.
- Always follow the instructions in the appropriate System Management Guide for each of the remaining queue managers.
- IBM MQ has a typical, compact, and a custom installation option. To ensure that the installation completes, select **Custom** so that you can select all the features. You can easily miss new features if you just chose a **Typical** installation.

Pay particular attention to instructions about what to do **before** installation. For IBM MQ on UNIX systems for example, you must create a user ID with the name **mqm** whose primary group is **mqm**.

You can use SMIT to install IBM MQ for AIX, or you can choose the easy installation. The easy installation places a minimal, typical set of components on your system. It excludes, for example, the online documentation and the application development support.



## Finding more information

- Latest details of hardware and software requirements on all supported operating systems on the MQ requirements website:  
<http://www.ibm.com/support/docview.wss?uid=swg27006467>
- Latest product support information on the MQ support website:  
[http://www.ibm.com/support/entry/portal/product/websphere/websphere\\_mq?productContext=24824631](http://www.ibm.com/support/entry/portal/product/websphere/websphere_mq?productContext=24824631)
- Product readme file (**readme.html**) for information about last-minute changes and known problems and workarounds

© Copyright IBM Corporation 2014

Figure 2-13. Finding more information

WM2091.0

### Notes:



## Installing MQ on Windows (1 of 4)

[Welcome](#)

[Software Requirements](#)
  
[Network Configuration](#)
  
[WebSphere MQ Installation](#)
  
  
[Installation Guide](#)
  
  
[Release Notes](#)

### Welcome to the WebSphere MQ Launchpad

To install IBM WebSphere MQ successfully you must make sure you have satisfied all the prerequisites. This launchpad guides you through this process.

Click on the left-hand buttons, described below, to prepare your system :-

Software Requirements	Identify the software you need, install it from the IBM WebSphere MQ CDROM, the Internet, or your local computer.
Network Configuration	Check the suitability of your user account and authorization.
WebSphere MQ Installation	Start the installation program.
Installation Guide	Installation instructions.
Release Notes	Up-to-date information about the product.

[Exit Launchpad](#)

© Copyright IBM Corporation 2014

Figure 2-14. Installing MQ on Windows (1 of 4)

WM2091.0

### Notes:

On Windows, you can use the IBM MQ Launchpad to check the system for prerequisites and install IBM MQ:

- **Software Requirements:** Check the operating system and level
- **Network Configuration:** Check user account and authorization for installation
- **WebSphere MQ Installation:** Install the server, client, IBM MQ Explorer, and other components
- **Release Notes:** List unresolved issues and workarounds

**WebSphere MQ Installation**

Pre-Installation Status

Software Requirements:

All prerequisite software has been installed and is at the required level.

Network Configuration Steps:

WebSphere MQ requires this information in order to function correctly. Click on 'More Information' button in the Network Configuration panel to find out what to ask your domain administrator.

Select installation language. English

Launch IBM WebSphere MQ Installer

Exit Launchpad

© Copyright IBM Corporation 2014

Figure 2-15. Installing MQ on Windows (2 of 4)

WM2091.0

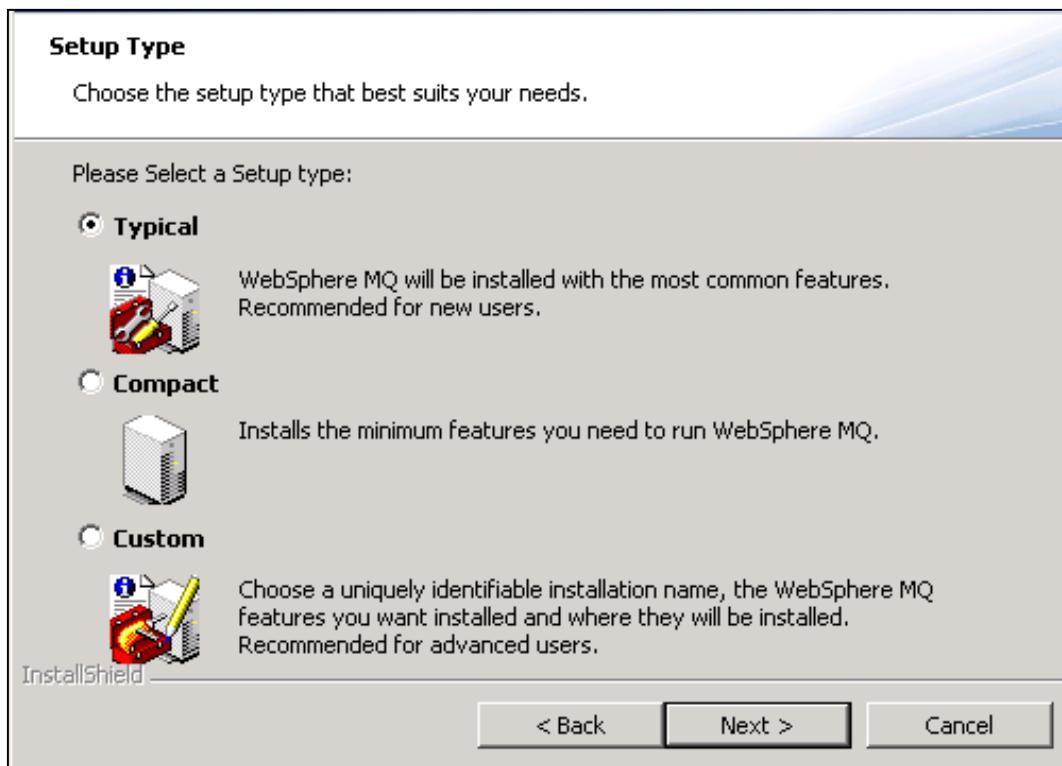
**Notes:**

To start the installation:

1. Click **WebSphere MQ Installation**.
2. Click **Launch IBM WebSphere MQ Installer**.



## Installing MQ on Windows (3 of 4)



© Copyright IBM Corporation 2014

Figure 2-16. Installing MQ on Windows (3 of 4)

WM2091.0

### Notes:

In a **Typical** installation, software is installed to a default location. The software packages that are installed are:

- IBM MQ server
- IBM MQ Explorer
- Development Toolkit
- Java messaging and SOAP transport

**Compact** installation installs the IBM MQ server only to the default location.

**Custom** installation contains the base software and an option to install the following components:

- Server: Telemetry service
- IBM MQ Explorer
- Windows Client: Client extended transaction support and Telemetry clients
- Java and .Net Messaging and web services: Java extended transaction support and .NET extended transaction support
- Development toolkit and sample programs

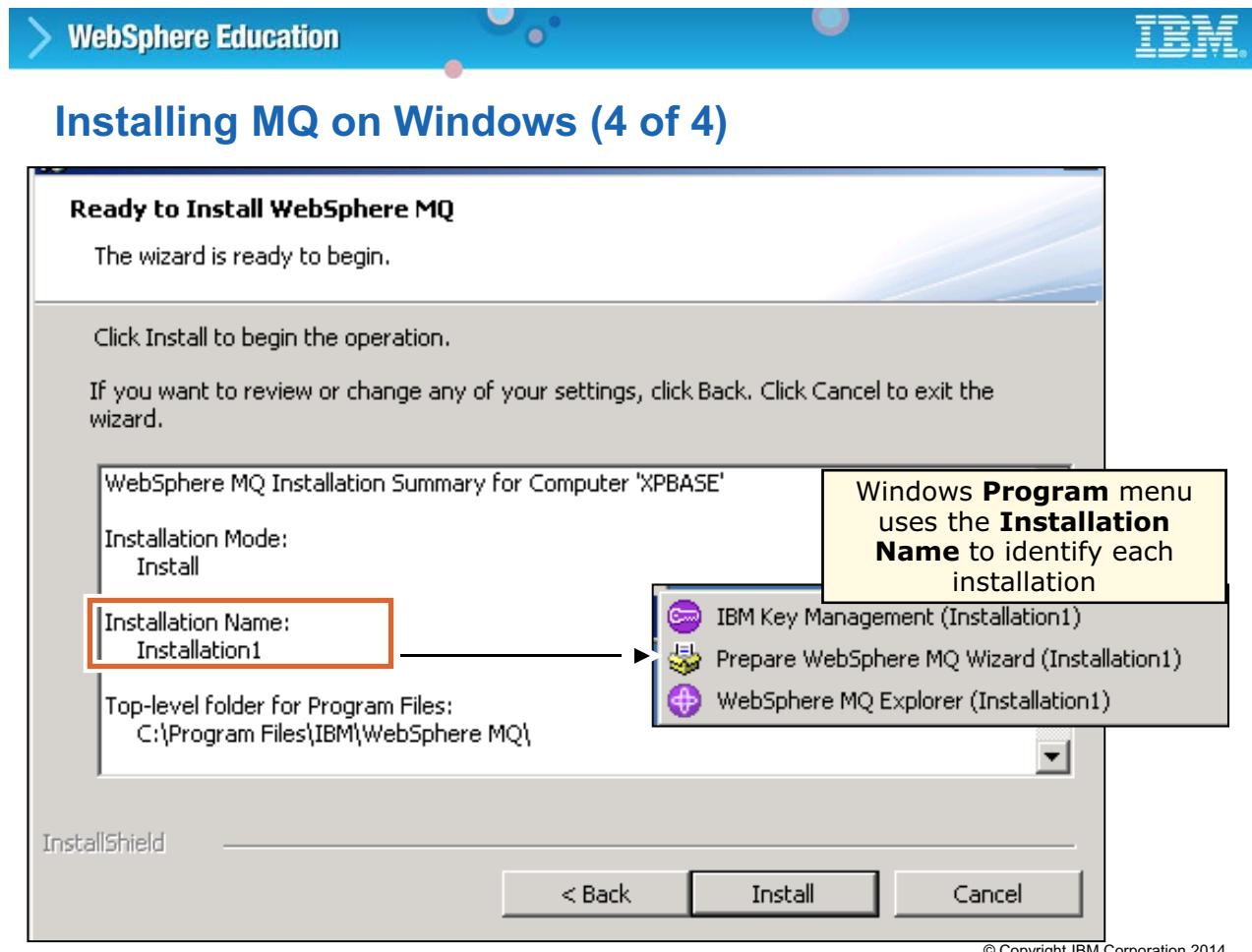


Figure 2-17. Installing MQ on Windows (4 of 4)

WM2091.0

## Notes:

As shown in the figure, the IBM MQ installation wizard lists a summary of the installation options. The options include the installation name and location.

The **Installation Name** setting identifies each installation of IBM MQ in support of multiple instance installations. As shown in the figure, the **Installation Name** setting identifies the installation instance on the Windows **Program** menu. For example, if the **Installation Name** is 'Installation1', each IBM MQ Program menu option includes 'Installation1'.

At this step in the installation, you can cancel or change your installation settings. Click **Install** to continue the installation.



## Installing multiple instances of MQ

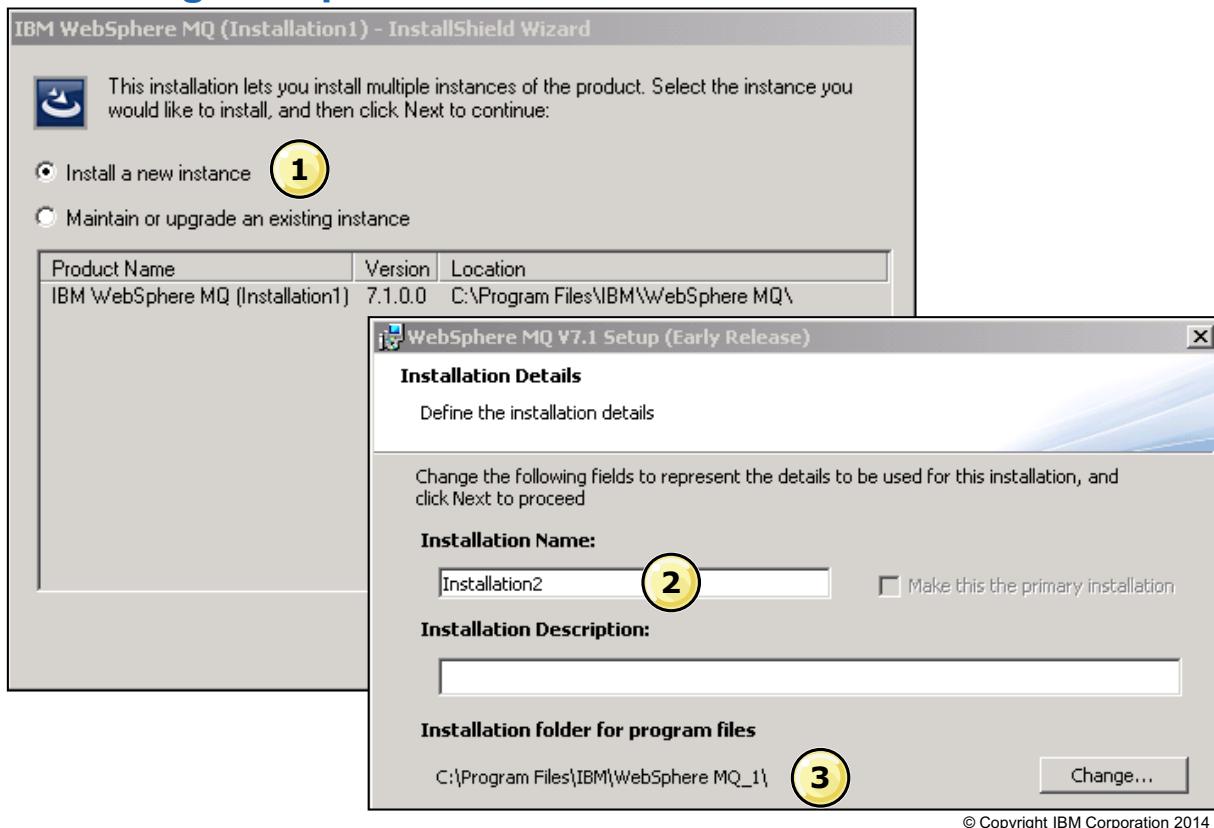


Figure 2-18. Installing multiple instances of MQ

WM2091.0

### Notes:

The figure shows the MQ Launchpad wizard screens that you see when you attempt to install another instance of IBM MQ.

To install a new instance on Windows by using the Launchpad:

1. Select **Install a new instance**. Click **Next**.
2. Enter a unique name to identify the installation.
3. Each installation requires a unique path for program files. Click **Next** to accept the default path or click **Change** to select a custom path.

From this point, the installation process is the same as the installation of the first instance of IBM MQ.

The screenshot shows the IBM WebSphere Education SupportPacs page. At the top, there's a blue header bar with the IBM logo and the text "WebSphere Education". Below the header, the main title is "WebSphere MQ SupportPacs". On the left side, there's a sidebar with links like "IBM Support Portal", "Support & downloads" (which is highlighted in blue), "Bookmark this page", "View my bookmarks", and "Feedback". Under "Support & downloads", there are sections for "Tags" (with a "Add a tag" button) and "My tags | All tags". A "View as cloud | list" link is also present. The main content area has a section titled "WebSphere MQ - SupportPacs by Category" with a sub-section "Product documentation". Below this, there's an "Abstract" section stating that tables summarize SupportPac categories. A "Content" section lists categories: "Fee based services SupportPacs", "As-is SupportPacs", "Product Extensions SupportPacs", and "Third Party Contributions SupportPacs". The "Fee based services SupportPacs" item is highlighted with a red box. There's also a "Related Links - SupportPac Categories" section. To the right, there's a "Document information" sidebar with details: "WebSphere MQ", "Software version: 5.1, 5.3, 6.0, 7.0", "Operating system(s): AIX, HP-UX, Linux, OS/2, Solaris, Windows, z/OS", and "Reference #: WM2091.0". At the bottom right of the content area, it says "© Copyright IBM Corporation 2014".

Figure 2-19. WebSphere MQ SupportPacs

WM2091.0

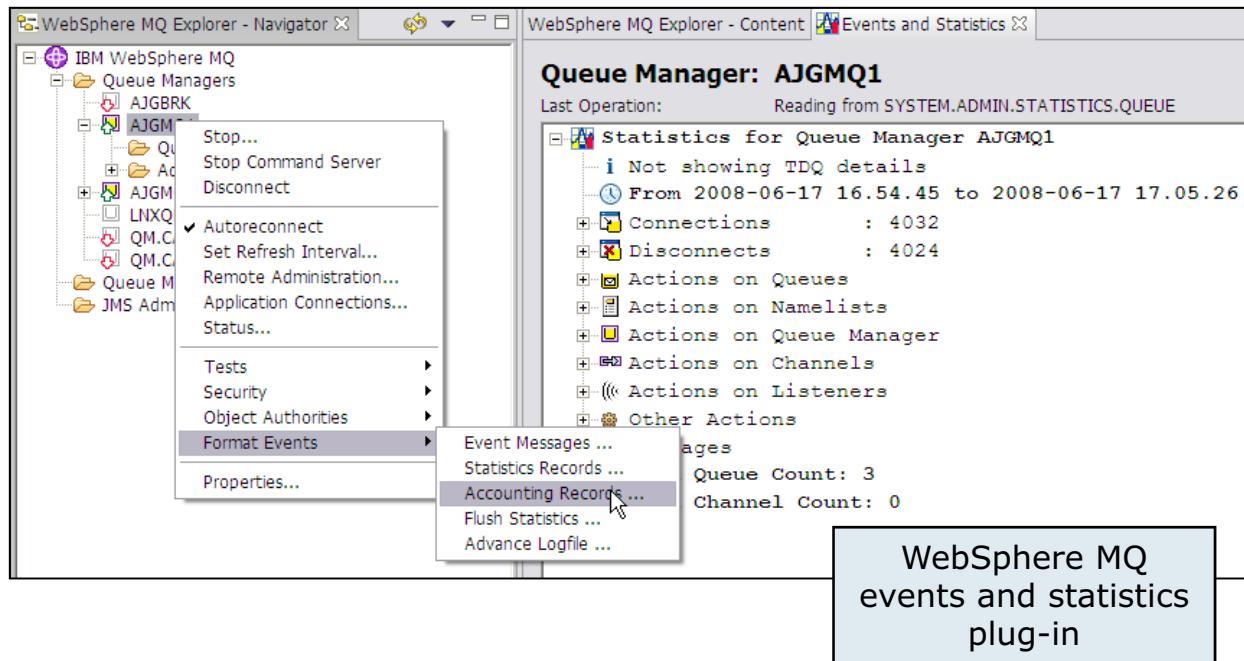
## Notes:

The WebSphere MQ family SupportPacs provide you with a wide range of downloadable code and documentation that complements the IBM MQ family of products.

WebSphere MQ SupportPacs are available in four categories:

- Fee-based services (not available for download)
- Freeware that is provided “as is” with no warranty or defect correction
- Fully supported product extensions that are developed by IBM
- Vendor contributions that are provided “as is” with no warranty or defect correction

## SupportPac example: MSOP



© Copyright IBM Corporation 2014

Figure 2-20. SupportPac example: MSOP

WM2091.0

### Notes:

The figure shows an example of the WebSphere MQ MSOP SupportPac. This SupportPac contains a set of plug-ins for the IBM MQ Explorer, extending capabilities for queue manager configuration, and displaying more information about them.

 **Note**

At the time of printing this manual, the current support level for this SupportPac was WebSphere MQ V7.1



## IBM MQ product documentation

- Detailed instructions on how to complete the tasks that you need to perform to create and maintain your MQ environment
- **Learning** topics help you learn about the product
- **Tasks** topics help you with basic and advanced tasks
- **Community and Support** contain links to communities where you can discover and share information with other product users

© Copyright IBM Corporation 2014

Figure 2-21. IBM MQ product documentation

WM2091.0

### Notes:

The IBM MQ product documentation contains documentation for the IBM MQ components that you installed.

In addition to reference documentation, the IBM MQ product documentation also contains links to tutorials and product tours.

The IBM MQ product documentation can be installed locally or accessed online at  
[http://www.ibm.com/support/knowledgecenter/SSFKSJ\\_8.0.0/welcome/WelcomePagev8r0.html](http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/welcome/WelcomePagev8r0.html).



## Uninstalling MQ

- Uninstall IBM MQ Explorer before uninstalling IBM MQ
- Ensure that there are no running IBM MQ programs or processes
- If running IBM MQ with Microsoft Cluster Service (MSCS), remove queue managers from the MSCS control before uninstalling IBM MQ
- Methods for uninstalling IBM MQ on Windows:
  - Start the installation process and then select the appropriate option
  - Use the **Add or Remove Programs** option in the Windows control panel

© Copyright IBM Corporation 2014

Figure 2-22. Uninstalling MQ

WM2091.0

### Notes:

If necessary, you can uninstall IBM MQ. Ensure that the applications and services are stopped before attempting to uninstall IBM MQ.

On IBM MQ in a Windows environment, you can use the **Add or Remove Programs** application in the Windows **Control Panel** to uninstall IBM MQ components and services.

You can also uninstall IBM MQ from a command if it was installed in the default location.

For example, on Linux, you would complete the following steps to uninstall IBM MQ:

1. Type `installp -u mqm`
2. Delete the contents of the IBM MQ data directory `/var/mqm`
3. Delete the file `/etc/opt/mqm/mqinst.ini`



## Unit summary

Having completed this unit, you should be able to:

- Identify the steps that are required to install IBM MQ on Windows or UNIX
- Find the hardware and software prerequisites for IBM MQ installation
- Identify the installation directory structure on Windows and UNIX
- Identify the configuration files and where to locate them on Windows and UNIX
- Locate and install IBM MQ Fix Packs
- Use IBM SupportPacs to complement the IBM MQ family of products

© Copyright IBM Corporation 2014

Figure 2-23. Unit summary

WM2091.0

### Notes:



## Checkpoint questions

1. True or False: A queue manager can be owned by only one MQ installation at a time.
2. True or False: On Windows systems, the installation name is automatically assigned during the installation process.

© Copyright IBM Corporation 2014

Figure 2-24. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

1.

2.



## Checkpoint answers

1. **True.**
2. **False.** On Windows systems, you can choose the installation name during the installation process.

© Copyright IBM Corporation 2014

Figure 2-25. Checkpoint answers

WM2091.0

### Notes:

# Unit 3. Creating a basic configuration

## What this unit is about

This unit describes the steps for using the IBM MQ commands and command scripts to verify an installation by creating a queue manager and local queues.

## What you should be able to do

After completing this unit, you should be able to:

- Use IBM MQ commands to create a queue manager and local queues
- Use IBM MQ commands to start and stop a queue manager
- Create IBM MQ command scripts

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM MQ product documentation



## Unit objectives

After completing this unit, you should be able to:

- Use IBM MQ commands to create a queue manager and local queues
- Use IBM MQ commands to start and stop a queue manager
- Create IBM MQ command scripts

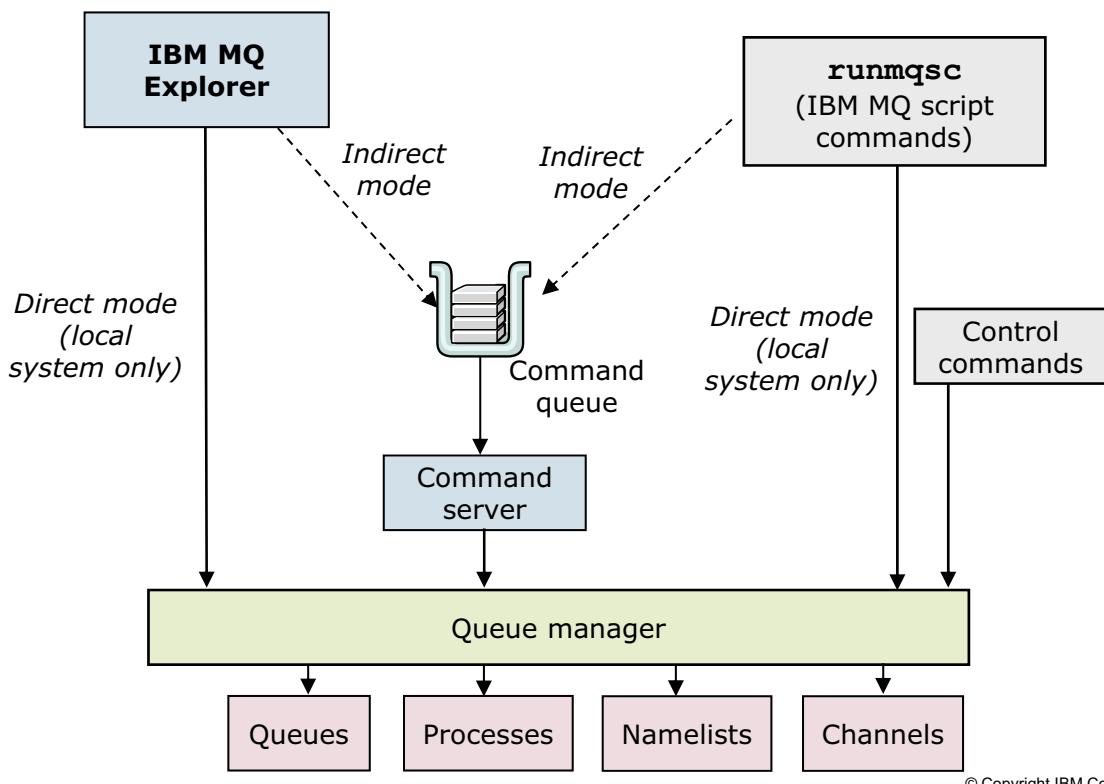
© Copyright IBM Corporation 2014

Figure 3-1. Unit objectives

WM2091.0

### Notes:

## Administration interfaces



© Copyright IBM Corporation 2014

Figure 3-2. Administration interfaces

WM2091.0

### Notes:

The primary IBM MQ administration interfaces are:

- MQ Explorer
- MQ script commands
- MQ control commands

MQ Explorer is a graphical user interface that runs on Linux and Windows. It can connect to, control, and configure MQ on all operating systems. MQ Explorer is described in detail later in this course.

MQ script commands and MQ control commands are described in more detail in this unit and throughout this course.

Other administration interfaces include the following components:

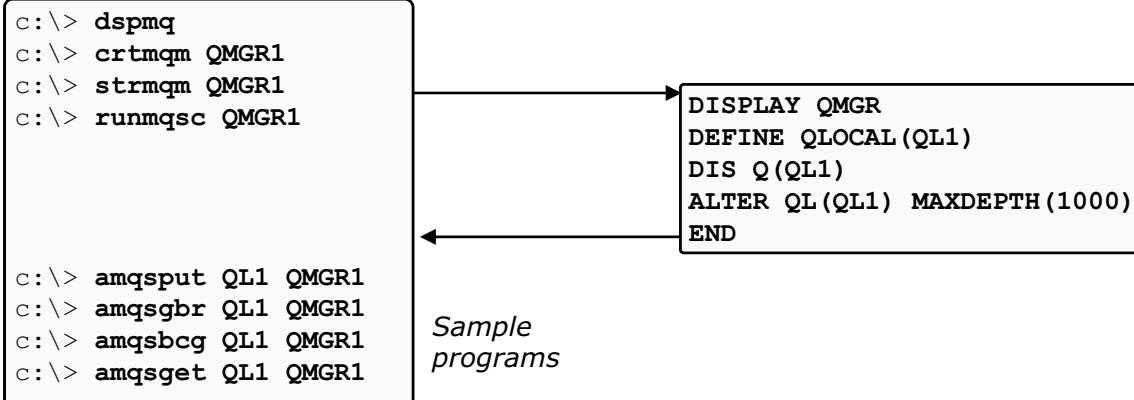
- Programmable command format (PCF) commands: PCF commands have a highly structured format and contain binary information and character information. The structured format makes it easier for an application to generate the components of a PCF command dynamically.

PCF commands are documented in the *IBM MQ Programmable Command Formats and Administration Interfaces* reference guide.

- IBM MQ Administration Interface (MQAI): The IBM MQAI is a programming interface to IBM MQ. It uses C language and Visual Basic for Windows. It does administrative tasks on an IBM MQ queue manager by using data bags. Data bags allow you to handle properties (or parameters) of objects in a way that is easier than using the other administrative interface, PCFs.
- Event messages: When a queue manager detects an instrumentation event during its operation, it puts an event message on an event queue. The event message contains information about the event that occurred.

## IBM MQ command modes

- Control commands
  - Entered at an operating system command prompt
  - Described in IBM MQ System Administration Guide
  - Lowercase when referenced in this course
  - Require authorization
- IBM MQ script commands (MQSC)
  - Entered in MQSC mode (no command prompt)
  - Described in IBM MQ Script (MQSC) Command Reference
  - Uppercase when referenced in course



© Copyright IBM Corporation 2014

Figure 3-3. IBM MQ command modes

WM2091.0

### Notes:

As shown in the figure, MQ supports two command modes: control command mode and MQ script (MQSC) mode.

Control commands are entered at the operating system command prompt. They can also be included in an operating system command file such as a shell script on a UNIX system or a batch file in Windows. The sample programs that are included with MQ are run with control commands.

You need the following authority to use a control command.

- On UNIX, your user ID must belong to the group “mqm”, but not necessarily as its primary group.
- On Windows, your user ID must belong either to the “mqm” local group or to the “Administrators” local group. It is possible for your user ID to belong to a global group, which, in turn, belongs to either of these local groups.

MQSC mode is started by entering the `runmqsc` control command. MQSC commands can be entered interactively, by typing an MQ command at the keyboard and waiting for the result. Alternatively, you can use a text editor to create a file that contains a sequence of MQSC commands and then run the file.

The MQSC commands are described in the *IBM MQ Script (MQSC) Command Reference*.

The figure shows examples of control commands and MQSC commands. In the example, the control command, `runmqsc QMGR1` starts MQSC mode for the queue manager that is named QMGR1. The commands in the MQSC mode are sent to QMGR1 until the `END` command is sent and the mode returns to control command mode.



**Important**

The operating command prompt is not available when the system is in the MQSC mode.

MQSC commands can be entered in uppercase or lowercase. In this course, however, MQSC commands are always shown in uppercase to help distinguish them from IBM MQ control commands.



## Creating queue managers

- Use `crtmqm` command or IBM MQ Explorer
- Can have multiple queue managers per system
- Specify a unique queue manager name
- Optionally, can specify:
  - One queue manager as the default queue manager (`-q`)
  - Dead-letter queue (`-u`)
  - Default transmission queue (`-d`)
  - Logging options
  - On UNIX and Linux systems, whether user authorization is to be used

Example: Create a default queue manager that is named QMR01 with linear logging and a dead letter queue that is named QMR01.DLQ

```
crtmqm -q -l1 -u QMR01.DLQ QMR01
```

© Copyright IBM Corporation 2014

Figure 3-4. Creating queue managers

WM2091.0

### Notes:

After installation, the first IBM MQ object to create is a queue manager. Typically, you must create only one queue manager per system, but you can create multiple queue managers that are based on needs, such as security or separation of business units, testing, or development purposes.

Every queue manager has a name, which should be unique within a network of queue managers that exchange messages with each other. When a queue manager generates a unique message identifier for a message, it uses the first 12 characters of its name as part of the identifier.

Queue manager names are case-sensitive. Always use uppercase characters to avoid problems that can be caused by using the incorrect case.

Plan conventions for queue manager names. For example, do not use host names as queue manager names when possible. Using host names as queue names can lead to problems over time if the host names change or frequent server reassignments occur. Some large organizations use host names that are combined with Domain Name System (DNS) servers, which can decrease the configuration work that is required when queue managers move to different dedicated IBM MQ servers. Most organizations successfully use geographic area, the application that the MQ queue manager is serving, or a combination of the two.

You can define a default transmission queue and dead-letter queue when you create the queue manager. You can also identify one queue manager on the system as the default queue manager.

If you are creating a queue manager for some simple tests, accept the default values for all the parameters unless there is a reason to use different ones.

For the complete description of `crtmqm` control command, see the IBM MQ product documentation. For some of the parameters, you can accept the default values and change them later by using the `ALTER QMGR` MQSC command.

## Starting and deleting a queue manager

- Start queue manager

```
strmqm QMgrName
```

- Display names and details about queue managers

```
dspmq  
dspmq -m QMgrName
```

- Delete queue manager

```
dltmqm QMgrName
```

**Note:** You must stop the queue manager first.

© Copyright IBM Corporation 2014

Figure 3-5. Starting and deleting a queue manager

WM2091.0

### Notes:

When a queue manager is first created, it is created in a *stopped* state. To start the queue manager, use the **strmqm** command followed by the name of the queue manager.

To display the status and queue manager configuration information, use the **dspmq** command.

To delete a queue manager and its associated objects, use the **dltmqm** command. You must stop the queue manager before you can delete it.

The create queue manager command, **crtmqm**, and the commands that are listed in this figure are control commands. The name of the queue manager is a required parameter on some of these commands. It is a good practice to always include the queue manager name in the command to ensure that the command is run against the correct queue manager.

The control commands are described in *IBM MQ System Administration Guide* and in the IBM MQ product documentation.

## Naming IBM MQ objects

- Allowable character set A - Z, a - z, 0 - 9, and the characters . / % \_
- Maximum of 48 characters for the names of:
  - Queue managers
  - Queues
  - Process definitions
  - Namelists
  - Clusters
  - Listeners
  - Services
  - Authentication information objects
  - Topics and subscriptions
- Maximum of 20 characters for the names of channels
- No implied structure in a name
- Object names that begin with SYSTEM are reserved
- Names in IBM MQ are case-sensitive

© Copyright IBM Corporation 2014

Figure 3-6. Naming IBM MQ objects

WM2091.0

### Notes:

The figure lists some of the rules for naming MQ objects such as queue managers and queues.

In general, MQ object names are limited to 48 characters. Channel names are limited to 20 characters but otherwise follow the standard rules for naming MQ objects.

Agree on all names before you start, and remember that names in MQ are case-sensitive. Be consistent in using uppercase and lowercase characters. Many organizations use uppercase characters for names, especially if z/OS queue managers are used within the MQ environment.

Period, forward slash, percent sign, and underscore are special characters that are allowed in MQ object names.

National language characters are not allowed.

Names can be enclosed in double quotation marks. If you use the forward slash or percent sign characters in a name, the name must be enclosed in double quotation marks.

Leading or embedded blanks are not allowed in MQ object names.

## End queue manager

- **Controlled (quiesced):** Allows connected applications to end, but no new ones

```
endmqm -c QMgrName
```

- **Controlled (wait):** Same as quiesced, except that `endmqm` reports queue manager shutdown status periodically

```
endmqm -w QMgrName
```

- **Immediate:** Completes all current MQI calls, but no new ones

```
endmqm -i QManagerName
```

- **Preemptive:** Stops without waiting for applications to disconnect or for MQI calls to complete

```
endmqm -p QManagerName
```

© Copyright IBM Corporation 2014

Figure 3-7. End queue manager

WM2091.0

### Notes:

The control command to stop the queue manager is `endmqm`. There are four options for controlling how the queue manager shuts down.

- **Controlled (or quiesced) shutdown:** The queue manager stops after all applications are disconnected. All new requests to connect to the queue manager fail. Controlled shutdown is the default mode.
- **Controlled shut down with wait:** End programs in the same manner as the controlled option. However, the command prompt does not return until the queue manager stops.
- **Immediate shut down:** The queue manager stops after it completes all the in-process MQI calls. Any MQI calls that are sent after this command is entered fail. Any incomplete units of work are rolled back when the queue manager is next started.
- **Preemptive shut down:** The queue manager stops without waiting for applications to disconnect or for MQI calls to complete. Use of this mode can lead to unpredictable results and should be used as the last option.



## Using MQSC to configure IBM MQ objects

- Command interface that is started by entering `runmqsc` control command
- Used to run IBM MQ script commands and script files
- Used to create, delete, modify, and display IBM MQ queue manager objects such as queues, channels, topics, and subscriptions

© Copyright IBM Corporation 2014

Figure 3-8. Using MQSC to configure IBM MQ objects

WM2091.0

### Notes:

The control command that starts MQSC mode is `runmqsc`. This command is entered at a control in a Windows command prompt or UNIX Shell prompt. The queue manager must be running to issue the `runmqsc` command. The queue manager can be local or remote.

The MQSC commands are known as script commands because they are frequently batched together to create a repeatable set of MQ resource definitions in a script file.

MQSC commands are used to create, modify, delete, and display MQ objects. Some of these object types are listed here.

The MQSC commands are also used to display the runtime status of objects like channels.

When entering MQSC commands interactively, you can get help by typing the name of the command and a question mark character: `command?`

## Starting the MQSC command interface

- For a local queue manager: `runmqsc QMgrName`
- For remote queue manager: `runmqsc -w WaitTime -m LocalQMgrName`
  - w WaitTime**  
The time, in seconds, that MQSC waits for replies from the remote queue manager. Assumes that the required channel and transmission queues are configured for MQSC on the remote queue manager.
  - m LocalQMgrName**  
Local queue manager to use to submit commands to the remote queue manager. If this parameter is omitted, the local default queue manager is used to submit commands to the remote queue manager.
- To verify a command: `runmqsc -e -v`
  - e** Do not copy source text to output report.
  - v** Perform syntax check only. This mode is only available locally.

© Copyright IBM Corporation 2014

Figure 3-9. Starting the MQSC command interface

WM2091.0

### Notes:

On all queue managers, the administration interface for entering MQSC (Script) commands is the control command `runmqsc`.

The input to `runmqsc` is zero, one, or more MQ script commands and the output is the result of running those commands, including operator and error messages. Alternatively, you can create a file that contains a sequence of MQSC commands and then have them run with the results directed to a file.

All parameters are optional in the `runmqsc` command. With no parameters, `runmqsc` attempts to connect the local default queue manager.

MQSC has three modes of operation:

- Verify:** Input commands are read and checked but not run.
- Direct:** Connect to a local queue manager and do not use any intermediate queues or channels.
- Indirect:** Connect to a remote queue manager.

## Stopping the MQSC command interface

- Use the **END**, **EXIT**, or **QUIT** commands to return to operating system command prompt

```
c:\> runmqsc QMGR1
DISPLAY QMGR
DEFINE QLOCAL(QL1)
DISPLAY QUEUE(QL1)
ALTER QLOCAL(QL1) MAXDEPTH(1000)
END
c:\> amqsput QL1 QMGR1
```

© Copyright IBM Corporation 2014

Figure 3-10. Stopping the MQSC command interface

WM2091.0

### Notes:

To exit MQSC mode and return to control command mode and the operating system prompt, enter **END**, **EXIT**, or **QUIT**. Optionally, you can enter the end-of-file (EOF) character for the system that you are using.

- Windows:** Ctrl+Z, and press **Enter**
- UNIX systems:** CTRL+D

The example in the figure shows the control command that starts MQSC mode for the queue manager that is named QMGR1: **runmqsc QMGR1**

The next four statements are MQSC commands that:

- Display information about the queue manager
- Define a local queue that is named **QL1**
- Display information about the local queue that is named **QL1**
- Change the **MAXDEPTH** property for the local queue that is named **QL1**

The **END** statement stops MQSC mode and returns control to the operating system.

## MQSC command rules

- In most cases, the script command format is:

verb object-type object-name parameter<sub>1</sub> .. parameter<sub>n</sub>

- Parameters are either keyword or keyword with value

Example: TRIGGER TRIGTYPE (FIRST)

- Keywords are not case-sensitive but string values are

- Parameter order is not significant although some exceptions do exist

- Some verbs can be abbreviated

Example: DEFINE as DEF

- Repeat parameters are not allowed

© Copyright IBM Corporation 2014

Figure 3-11. MQSC command rules

WM2091.0

### Notes:

The figure lists some of the rules for the specification of MQSC commands.

The command verb must always come first and is followed by, in most cases, is an MQ object, and then optional keyword and keyword/value parameters.

Script commands are not case-sensitive; names and parameter values are case-sensitive.

In most cases, parameter order is not important, and some verbs can be abbreviated. An exception to parameter order is that the parameter CHLTYPE must be the first parameter that is specified in `DEFINE CHANNEL` and `ALTER CHANNEL` commands except on z/OS.

Abbreviations are fixed and are described in the MQ product documentation under their relevant MQSC command entries as synonyms. For example, the synonym for `REFRESH CLUSTER` is `REF CLUSTER`. The synonym for `DEFINE QLOCAL()` is `DEF QL()`.

Parameters cannot be repeated within the same script command. For example, `ALTER QLOCAL(ANDREW) TRIGGER TRIGGER` is not valid. Repeating the negation of the same parameter, as in, `ALTER QLOCAL(ANDREW) TRIGGER NOTRIGGER` is not valid.

## MQSC script files

- Each command starts on new line
- Commands and keywords are not case-sensitive
- Blank line and lines that are prefixed with an asterisk (\*) are ignored
- Restrict maximum line length to 72 characters for portability
- Last non-blank character determines continuation
  - Minus sign (-) continues command from start of next line
  - Plus sign (+) continues command from first non-blank character in the next line

Example:

```
DEFINE QLOCAL(MY_DEAD_LETTER_Q) +
REPLACE

ALTER QMGR DEADQ(MY_DEAD_LETTER_Q)

DEF QL(ANOTHER) REPLACE +
DESCR('This is a test queue')

* This is a comment
```

© Copyright IBM Corporation 2014

Figure 3-12. MQSC script files

WM2091.0

### Notes:

MQSC commands can be included in a text file and then run against a queue manager as a script. This figure lists some of the rules for writing MQSC script files. More syntax rules for writing MQ commands are:

- You can optionally use a semicolon (;) to end a command.
- An MQSC command can contain a string of characters. A string that contains blanks, lowercase characters, or special characters other than characters valid in the name of an MQ object, must be enclosed in single quotation marks. Lowercase characters that are not enclosed in single quotation marks are internally converted to uppercase.

A string that has no characters is not valid.

The figure shows three examples of MQ commands.

- The first command creates a local queue with the name MY\_DEAD\_LETTER\_Q.
- The second command alters the queue manager by declaring MY\_DEAD\_LETTER\_Q as the dead-letter queue of the queue manager.

- The third command creates another local queue. A keyword, such as DESCRIPTOR, can be in lowercase or uppercase. In this course, all MQSC commands and keywords are shown in uppercase to distinguish them from control commands.

The single quotation marks that enclose the string This is a test queue are required because the string contains blanks.

The MQ commands are described in the *IBM MQ Script (MQSC) Command Reference* and in the IBM Knowledge Center. On UNIX systems, **man** pages are provided for all the IBM MQ commands.



## Running MQSC script files

- Redirect the input from a file to run a sequence of frequently used commands that are contained in the file

Example: `runmqsc QMgrName < mqsc.in`

- Redirect the input from a file and redirect the output report to a file

Example: `runmqsc QMgrName < mqsc.in > mqsc.out`

© Copyright IBM Corporation 2014

Figure 3-13. Running MQSC script files

WM2091.0

### Notes:

As shown in the figure, the `runmqsc` command reads from the *standard input device* and writes to the *standard output device*. Typically, the standard input device is the keyboard and the standard output device is the display. However, by using redirection operators, input can be taken from a file and output can be directed to a file.

In the first example, the file that is named `mqsc.in` is read and processed by the queue manager that is identified with the `runmqsc` command.

In the second example, the file that is named `mqsc.in` is read and processed by the queue manager that is identified with the `runmqsc` command and a report that shows the processing of each command is created in the `mqsc.out` file.

It is useful to maintain MQSC files, for the following reasons:

- To replicate a queue manager configuration on multiple systems.
- To recover a queue manager configuration.
- When going through a number of iterations in testing a queue manager configuration.



## Defining a local queue

- Use **DEFINE QLOCAL** to define a local queue
  - **DEFINE** can be abbreviated to **DEF**
  - **QLOCAL** can be abbreviated to **QL**
  - **DEFINE** with **REPLACE** sets all the parameters with unspecified parameters taken from the default definition (SYSTEM.DEFAULT.LOCAL.QUEUE) or from queue that is named on the **LIKE** parameter
- Queue names are case-sensitive
- Useful naming conventions
  - Name of a queue to describe its function, not its type, or location
  - Common prefix for names of related queues simplifies administration

© Copyright IBM Corporation 2014

Figure 3-14. Defining a local queue

WM2091.0

### Notes:

The MQSC command **DEFINE QLOCAL** defines a queue that is local to the queue manager.

Keywords and their values specify the values of attributes of the local queue that is created. The values of the attributes that are not explicitly defined are taken from the values of the corresponding attributes of the default local queue, **SYSTEM.DEFAULT.LOCAL.QUEUE**. The **SYSTEM.DEFAULT.LOCAL.QUEUE** is created during MQ installation.

Queue names in MQ are case-sensitive. Use a standard convention for case such as all uppercase or all lowercase.

Other queue name guidelines include the following rules:

- Do not use the type or location of the queue in the queue name. If a queue is changed from a local to a remote queue for example, the same name can still be used for the queue; it is not necessary to change the applications that reference the queue.

Instead, name the queue after its function.

- Use a common prefix for the names of related queues to aid administration. For example, name all queues that are related to the same application with the same prefix.

## Defining a local queue examples

```
* Define a local queue that is named QL.APPINPUT
DEFINE QLOCAL(QL.APPINPUT)

* Define a local queue that is named QL.TESTQ and replace
* some of the default definition parameters

DEF QL(QL.TESTQ) REPLACE +
DESCR('This is a local test queue') +
PUT(ENABLED) GET(ENABLED) +
DEFPSIST(YES) +
MAXDEPTH(1000) MAXMSGL(2000)

* Define a local queue that is named QL.TESTQ2 by using
* QL.TESTQ as the model

DEF QL(QL.TESTQ2) LIKE(QL.TESTQ)
```

© Copyright IBM Corporation 2014

Figure 3-15. Defining a local queue examples

WM2091.0

### Notes:

The figure shows three examples of the MQSC command `DEFINE QLOCAL`. The second and third examples use the synonym `DEF QL`.

The first example creates a local queue that is named QL.APPINPUT and uses the default attributes for a local queue.

The second example creates a local queue that is named QL.TESTQ and replaces some of the default attributes. The `REPLACE` keyword indicates that if the queue exists, replace its definition with the new one as defined in this command. Any messages on an existing queue are retained.

The third example defines a local queue that is named QL.TEST2. The `LIKE` keyword means that the named queue (QL.TESTQ in this example) is used for the default values of attributes rather than the default local queue SYSTEM.DEFAULT.LOCAL.QUEUE.



## Defining other queue types

- Alias queue
  - Provides a level of indirection to another queue or a topic object
  - Uses SYSTEM.DEFAULT.ALIAS.QUEUE for default definition

**DEFINE QALIAS (AAA) TARGET (BBB)**
  
- Local definition of a remote queue (remote queue definition)
  - A local definition of a remote queue, a queue manager alias, or a reply-to queue alias
  - Uses SYSTEM.DEFAULT.REMOTE.QUEUE for default definition

**DEFINE QREMOTE (AAA) RNAME (BBB) RQMNAME (CCC)**
  
- Model queue
  - Queue template for creating dynamic queues
  - Uses SYSTEM.DEFAULT.MODEL.QUEUE for default definition

**DEFINE QMODEL (AAA)**

© Copyright IBM Corporation 2014

Figure 3-16. Defining other queue types

WM2091.0

### Notes:

An *alias queue* is an IBM MQ object that refers indirectly to another queue. The MQSC command to create an alias queue is **DEFINE QALIAS** or **DEF QA**. The **TARGET** keyword specifies the name of the queue to which the alias queue resolves. The target queue can be a local queue or a local definition of a remote queue.

A *local definition of a remote queue*, or a *remote queue definition*, is an IBM MQ object that is owned by one queue manager that refers to a queue owned by another queue manager. The MQSCQ command to create a local definition of a remote queue is **DEFINE QREMOTE** or **DEF QR**. The keyword **RNAME** is the name of the queue as it is known on the remote queue manager. The keyword **RQMNAME** is the name of the remote queue manager.

A *model queue* is an IBM MQ object whose attributes are used as a template for creating a dynamic queue. The MQSC command to create a model queue is **DEFINE QMODEL**. When an application opens a model queue, the queue manager creates a dynamic queue. The **DEFTYPE** keyword specifies whether a dynamic queue created from the model queue is:

- A *temporary* dynamic queue, which is deleted when it is closed and does not survive a queue manager restart, or

- A *permanent* dynamic queue, whose deletion on the MQCLOSE call is optional and, which does survive a queue manager restart.

Of the two types of dynamic queues, only a permanent dynamic queue can store persistent messages. A typical use of a dynamic queue is as a reply-to queue for a client program that is sending requests to a server.

## Clearing and deleting queues

- Delete all messages on a local queue

```
CLEAR QLOCAL (XXX)
```

- Delete a queue

```
DELETE QLOCAL (XXX)
DELETE QREMOTE (XXX)
DELETE QALIAS (XXX)
```



Commands fail if the queue is in use.

© Copyright IBM Corporation 2014

Figure 3-17. Clearing and deleting queues

WM2091.0

### Notes:

This figure shows the MQSC commands that can be used to clear all messages on a queue and delete a queue.

The commands `DELETE QLOCAL`, `DELETE QALIAS`, `DELETE QREMOTE`, and `CLEAR QLOCAL` fail if the queue:

- Contains uncommitted messages that are put on the queue under sync point
- Is open by an application (with any open options)
- Is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open



## Displaying attributes

- Display all the attributes of a specific local queue

```
DISPLAY QLOCAL (XXX)
```

- Display all or selected attributes of one or more queues of any type.

```
DISPLAY QUEUE (XXX) DESCRIPTOR GET PUT
DISPLAY QUEUE (YYY) MAXDEPTH CURDEPTH
```

- Wildcards are allowed to display attributes for multiple queue managers and queues

```
DIS Q(SYSTEM*)
```

- Display all attributes of the queue manager object

```
DISPLAY QMGR
```

© Copyright IBM Corporation 2014

Figure 3-18. Displaying attributes

WM2091.0

### Notes:

The MQSC command to display all the attributes of a specific queue is **DISPLAY** followed by the queue type keyword (**QLOCAL**, **QALIAS**, **QREMOTE**, or **QMODEL**).

To display selected attributes for one or more queues, use the **DISPLAY QUEUE** command. This command applies to all types of queue: local, alias, remote, and model.

The **DISPLAY QUEUE** command can specify a generic queue name by using a wildcard trailing asterisk (\*). An asterisk without any other characters specifies “all queues”.

The default behavior of the **DISPLAY QUEUE** command is to display all queue attributes. You can request the display of selected attributes by using the appropriate keywords, as shown in the examples in the figure.

The MQSC command to display the attributes of the queue manager object is **DISPLAY QMGR**. Do not enter the name of the queue manager in the command.

The default action of the **DISPLAY QMGR** command is to display all the attributes. You can choose to display different sets of queue manager parameters, such as **CHINIT**, **CLUSTER**, **EVENT**, **SYSTEM**, and **PUBSUB**.

## Altering attributes

- Alter selected attributes on a queue manager

```
ALTER QMGR DESCRIPTOR('New description')
```

- Alter selected attributes on a queue

```
ALTER QLOCAL(XXX) PUT(DISABLED)  
ALTER QALIAS(AAA) TARGET(YYY)
```

© Copyright IBM Corporation 2014

Figure 3-19. Altering attributes

WM2091.0

### Notes:

The MQSC command **ALTER QMGR** command is used to modify queue manager attributes.

The MQSC commands **ALTER QLOCAL**, **ALTER QALIAS**, **ALTER QREMOTE**, and **ALTER QMODEL**, change the specified attributes of the queue to which the command is applied. The remaining attributes of the queue are left unchanged.



## Defining special queues

- Dead-letter queue for messages that cannot be delivered
  - One per queue manager
  - Always identify a dead-letter queue when defining the queue manager
- Initiation queues for triggering
- Transmission queues
- Command queue
- Event queues
- Default queues

© Copyright IBM Corporation 2014

Figure 3-20. Defining special queues

WM2091.0

### Notes:

Some local queues have special purposes in IBM MQ.

- A *dead-letter queue* is a designated queue where a queue manager puts messages that cannot otherwise be delivered. It is not mandatory for a queue manager to have a dead-letter queue, but it is a good practice and critical to the health of the queue manager. The SYSTEM queue SYSTEM.DEAD.LETTER.QUEUE is not automatically assigned as the queue manager's dead-letter queue.
- An *initiation queue* is a queue that is used to implement triggering. Triggering is taught in more detail later in the course.
- *Transmission queues* are queues that temporarily store messages that are destined for remote queue managers.
- The *command queue*, SYSTEM.ADMIN.COMMAND.QUEUE, is a local queue to which suitably authorized applications can send MQSC commands for processing. The IBM MQ command server retrieves these commands. The command server validates the commands, passes the valid ones on for processing by the queue manager, and returns any responses to the appropriate reply-to queue.

- When a queue manager detects an instrumentation event, which can either be a channel, queue manager, performance, or logger event, the queue manager puts a message that describes that event on an *event queue*. A system management application can monitor an event queue, get the messages put on these queues, and then take appropriate action.
- The purpose of the *default queues* is to identify the default values of the attributes of any new queue that you create. There is one default queue for each of the four types of queues: local, alias, remote, and model. You must include in the definition of a queue those attributes only whose values are different from the default values when creating a queue. You can change the default value of an attribute by redefining the appropriate default queue.



## Defining a dead-letter queue

### Option 1:

1. Identify a dead-letter queue when you create the queue manager with the **-u** option on **crtmqm** command

Example: **crtmqm -u QMGR01.DLQ QMGR01**

2. Create the dead-letter queue as local queue in MQSC

Example: **DEF QL(QMGR01.DLQ)**

### Option 2:

1. Create the queue manager and do not identify a dead-letter queue
2. Create a local queue for the dead-letter queue
3. Alter the queue manager to use the new local queue for the dead-letter queue

Example: **DEF QL(APP.DLQ)**

**ALTER QMGR DEADQ(APP.DLQ)**

© Copyright IBM Corporation 2014

Figure 3-21. Defining a dead-letter queue

WM2091.0

### Notes:

There are two options for creating and assigning a dead-letter queue to a queue manager.

The first option is to identify the dead-letter queue when you create the queue manager with the **-u** option. In the example, the command that creates the queue manager that is named **QMGR01** identifies the dead-letter queue that is named **QMGR01.DLQ** as its dead-letter queue.

The second option is to assign the dead-letter queue after the queue manager is created by using the QMSC **ALTER QMGR** command. In this second option, step 2 creates a local queue that is named **APP.DLQ**. Step 3 defines the new local queue as the queue manager dead-letter queue.



## Unit summary

Having completed this unit, you should be able to:

- Use IBM MQ commands to create a queue manager and local queues
- Use IBM MQ commands to start and stop a queue manager
- Create IBM MQ command scripts

© Copyright IBM Corporation 2014

Figure 3-22. Unit summary

WM2091.0

### Notes:

## Checkpoint questions (1 of 2)

1. True or False: Queue manager names can contain any printable ASCII character.
2. Which of the following queue types can be the target of an alias queue?
  - a. Another alias queue
  - b. Local queues
  - c. Model queues
3. Any local queue can be a dead-letter queue if it:
  - a. Is identified as the dead-letter queue to the queue manager.
  - b. Has PUT enabled.
  - c. Is not in use by any other application.
4. True or False: You can alter queue attributes while the queue manager is running and the changes take effect immediately.

© Copyright IBM Corporation 2014

Figure 3-23. Checkpoint questions (1 of 2)

WM2091.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Checkpoint questions (2 of 2)

5. What does the command `crtmqm -q -u DLQ CHIWINSVR` create?
  - a. A queue manager that is named DLQ
  - b. A default queue manager that is named CHIWINSVR with queue DLQ assigned to the queue manager as the dead-letter queue
  - c. A queue manager that is named CHIWINSVR
  - d. A queue manager that is named CHIWINSVR with a default transmission queue DLQ assigned to the queue manager

© Copyright IBM Corporation 2014

Figure 3-24. Checkpoint questions (2 of 2)

WM2091.0

### Notes:

Write your answer here:

5.



## Checkpoint answers

1. **False.** Queue manager names do not support the entire set of printable ASCII characters.
2. b
3. a
4. **True**
5. b

© Copyright IBM Corporation 2014

Figure 3-25. Checkpoint answers

WM2091.0

### Notes:

## Exercise 1



Using commands to create queue  
managers and queues

© Copyright IBM Corporation 2014  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

8.0

Figure 3-26. Exercise 1

WM2091.0

### Notes:

In this exercise, you create a queue manager, start it, and then create queues.



## Exercise objectives

After completing this exercise, you should be able to:

- Use IBM MQ commands to create a local queue manager, local queues, and alias queues
- Use IBM MQ commands to display and alter queue manager and queue attributes
- Create and run an IBM MQ command file

© Copyright IBM Corporation 2014

---

Figure 3-27. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for the exercise description and detailed instructions.

# Unit 4. Introduction to IBM MQ Explorer

## What this unit is about

IBM MQ Explorer is a graphical Interface for administering IBM MQ. This unit describes the use of the IBM MQ Explorer application to administer IBM MQ objects.

## What you should be able to do

After completing this unit, you should be able to:

- Use IBM MQ Explorer to create a local queue manager and queues
- Use IBM MQ Explorer to create and manage queue manager sets
- Use IBM MQ Explorer to run tests to verify IBM MQ object definitions

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM MQ product documentation



## Unit objectives

After completing this unit, you should be able to:

- Use IBM MQ Explorer to create a local queue manager and queues
- Use IBM MQ Explorer to create and manage queue manager sets
- Use IBM MQ Explorer to run tests to verify IBM MQ object definitions

© Copyright IBM Corporation 2014

---

Figure 4-1. Unit objectives

WM2091.0

### Notes:



**Queues**

Queue name	Queue type	Definition type	Open input count	Open output count
Canberra2	Local	Predefined	0	0
SYSTEM.ADMIN.ACCTING.QUEUE	Local	Predefined	0	0
SYSTEM.ADMIN.ACTIVITY.QUEUE	Local	Predefined	0	0
SYSTEM.ADMIN.CHANNEL.EVENT	Local	Predefined	0	0
SYSTEM.ADMIN.COMMAND.QUEUE	Local	Predefined	1	1
SYSTEM.ADMIN.LOGGER.EVENT	Local	Predefined	0	0
SYSTEM.ADMIN.PERF.MEVENT	Local	Predefined	0	0
SYSTEM.ADMIN.QMGR.EVENT	Local	Predefined	0	0
SYSTEM.ADMIN.STATISTICS.QUEUE	Local	Predefined	0	0
SYSTEM.ADMIN TRACE.ROUTE.QUEUE	Local	Predefined	0	0
SYSTEM.AUTH.DATA.QUEUE	Local	Predefined	1	1
SYSTEM.BROKER.ADMIN.STREAM	Local	Predefined	1	0
SYSTEM.BROKER.CONTROL.QUEUE	Local	Predefined	3	1

- Graphical tool for configuring and controlling IBM MQ from a single console
- Configure all IBM MQ objects and resources, including JMS, and publish/subscribe
- Available on Windows and Linux (x86) only

© Copyright IBM Corporation 2014

Figure 4-2. IBM MQ Explorer

WM2091.0

## Notes:

IBM MQ Explorer is the graphical user interface for administering and monitoring IBM MQ objects, whether your local computer or a remote system hosts them.

MQ Explorer also supports:

- Advanced filtering
- Management of application connections
- Grouping of queue managers to simplify management
- Publish/subscribe administration

MQ Explorer is built as a Rich Client Platform (RCP) application on Eclipse. It runs on Windows and Linux (x86) operating systems but can be used to remotely manage IBM MQ objects on other operating systems.



## Starting IBM MQ Explorer

- On Windows, can be started from:

- Windows **Start** menu
  - **strmqcfg.cmd**

- On Linux, can be started from:

- System menu
  - <INSTALL\_DIR>/bin/MQExplorer.cmd
  - <INSTALL\_DIR>/bin/strmqcfg.cmd

- **strmqcfg** optional parameters:

- c Delete any cached data
  - i Discard configuration information that IBM MQ uses
  - x Write debug messages to the console

© Copyright IBM Corporation 2014

Figure 4-3. Starting IBM MQ Explorer

WM2091.0

### Notes:

IBM MQ Explorer can be started from a command or from the system start menu.

On both Windows and Linux, the command to start IBM MQ Explorer is **strmqcfg**. The **strmqcfg** command has three optional parameters to clear the cache and generate debug messages to the console.



## Welcome page

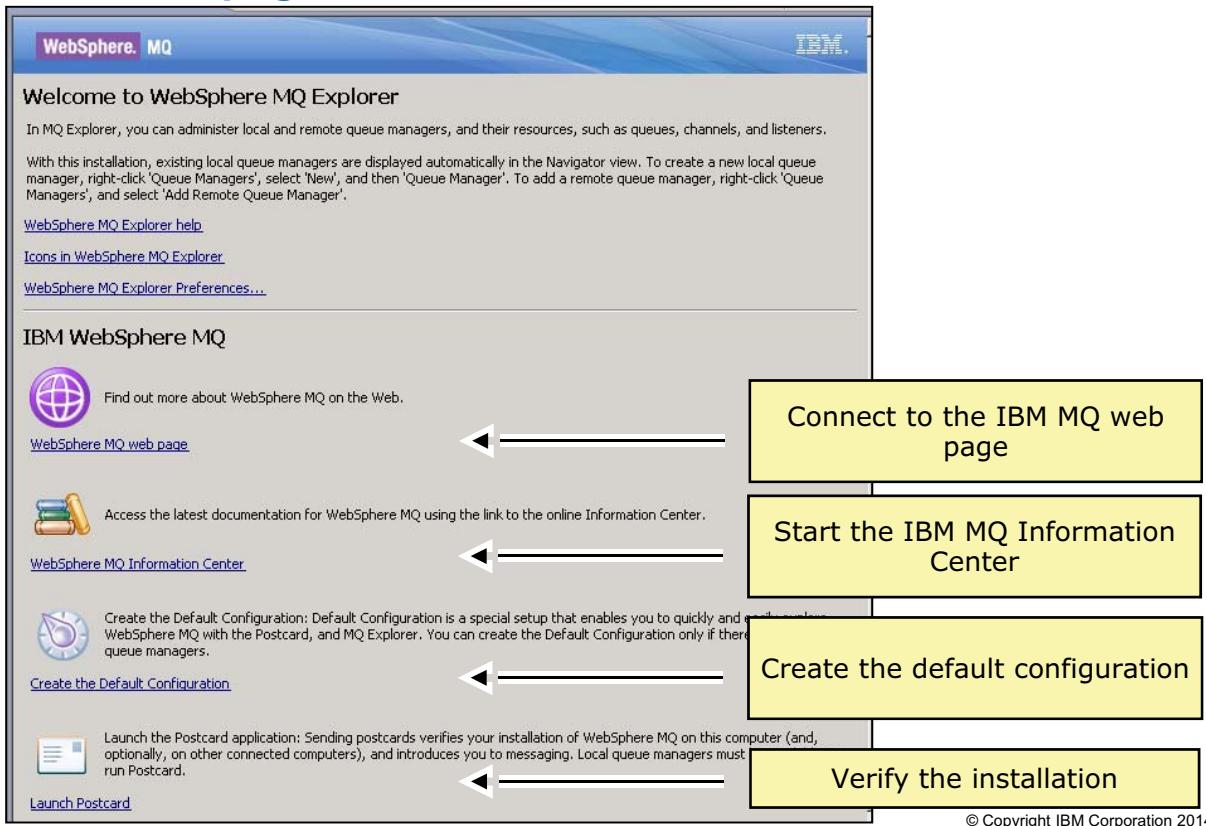


Figure 4-4. Welcome page

WM2091.0

### Notes:

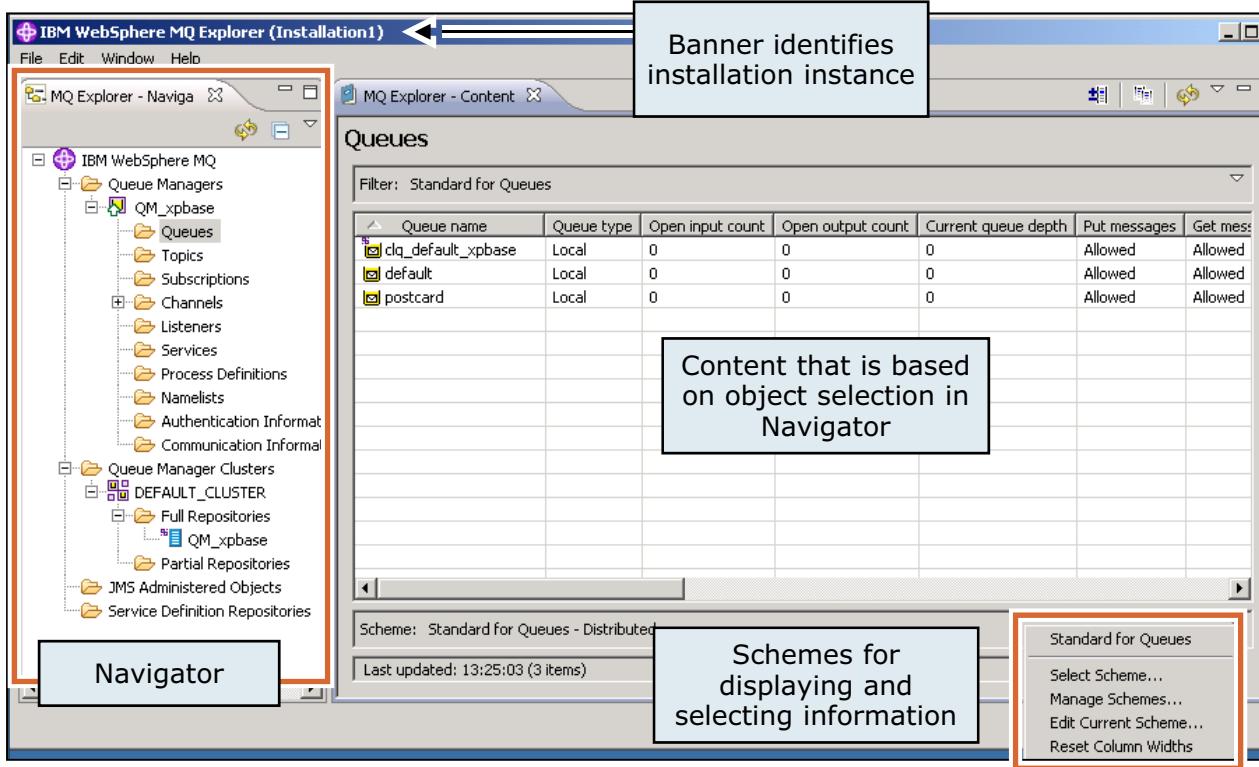
The IBM MQ Explorer Welcome page is displayed the first time that IBM MQ Explorer starts.

As shown in the figure, the Welcome page contains links for information about IBM MQ, for creating the default configuration, and verifying the installation.

The Welcome page can be accessed at any time by clicking **Help > Welcome**.



## IBM MQ Explorer default views



© Copyright IBM Corporation 2014

Figure 4-5. IBM MQ Explorer default views

WM2091.0

### Notes:

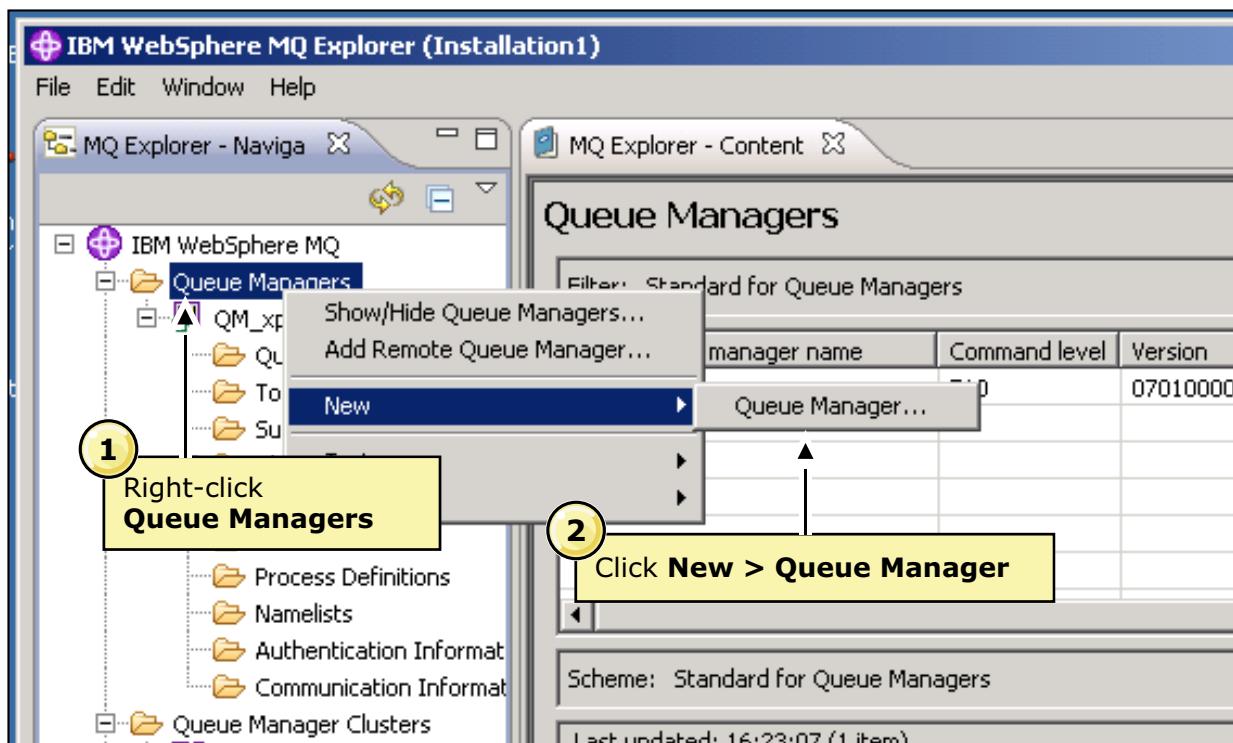
The figure shows the general layout of the IBM MQ Explorer user interface.

The Navigator view on the left contains a list of all the objects that IBM MQ Explorer manages. Status icons next to some of the objects, such as queue managers, indicate the status of the object.

The Content views contain more information about the object that is selected in the Navigator. For example, if the **Queues** folder under a queue manager is selected in the Navigator view, the content view contains information about the queues on that queue manager.

The **Scheme** icon (bottom portion of the **Queue** content view) allows you to edit the current scheme and select the information to display in the queue property columns. You can also change the order of the columns.

## Creating a local queue manager (1 of 5)



© Copyright IBM Corporation 2014

Figure 4-6. Creating a local queue manager (1 of 5)

WM2091.0

### Notes:

A queue manager can be created from a command or by using MQ Explorer.

MQ Explorer automatically shows all local queue managers regardless of the method that is used to create them.

The figure shows the first two steps for adding a queue manager to the server by using MQ Explorer.

1. Right-click **Queue Managers** in the Navigator.
2. Click **New > Queue Manager**.



## Creating a local queue manager (2 of 5)

**Create Queue Manager**

**Queue Manager**  
Enter basic values (Step 1)

Queue manager name:   
 Make this the default queue manager

Default transmission queue:

Dead letter queue:

Max handle limit:

Trigger interval:

Max uncommitted messages:

**3** Enter the **Queue manager name** and the basic values

< Back    Next >    Finish    Cancel

© Copyright IBM Corporation 2014

Figure 4-7. Creating a local queue manager (2 of 5)

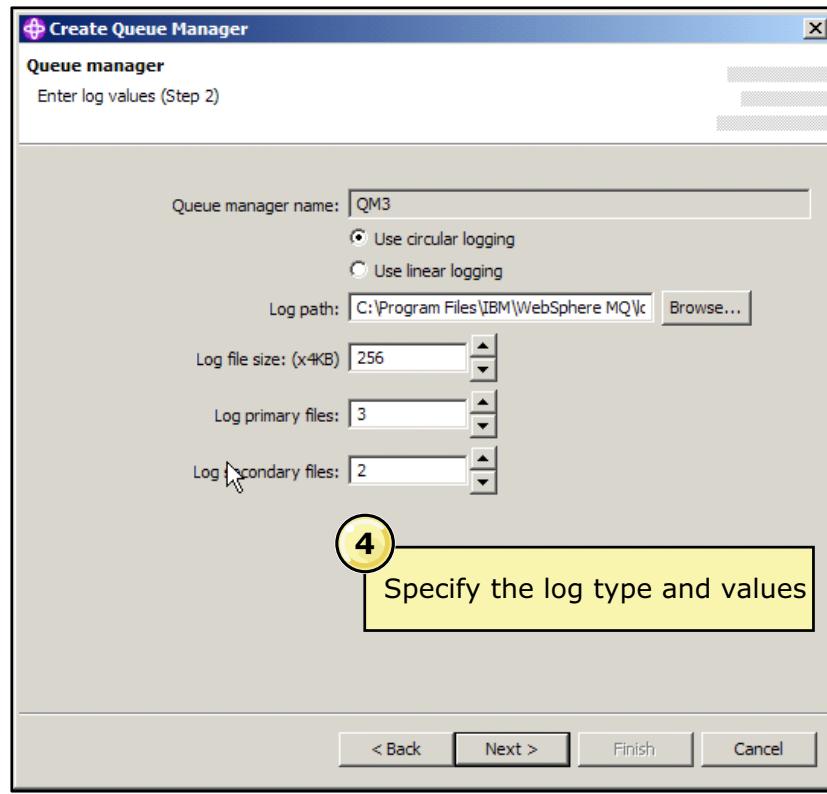
WM2091.0

### Notes:

As shown in the figure, the next step for creating a queue manager in MQ Explorer is to specify a queue manager name and other basic configuration. Basic configuration includes specifying a default transmission queue, dead-letter queue, and trigger interval.



## Creating a local queue manager (3 of 5)



© Copyright IBM Corporation 2014

Figure 4-8. Creating a local queue manager (3 of 5)

WM2091.0

### Notes:

The next step for creating a queue manager is to specify the queue manager log values.

The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

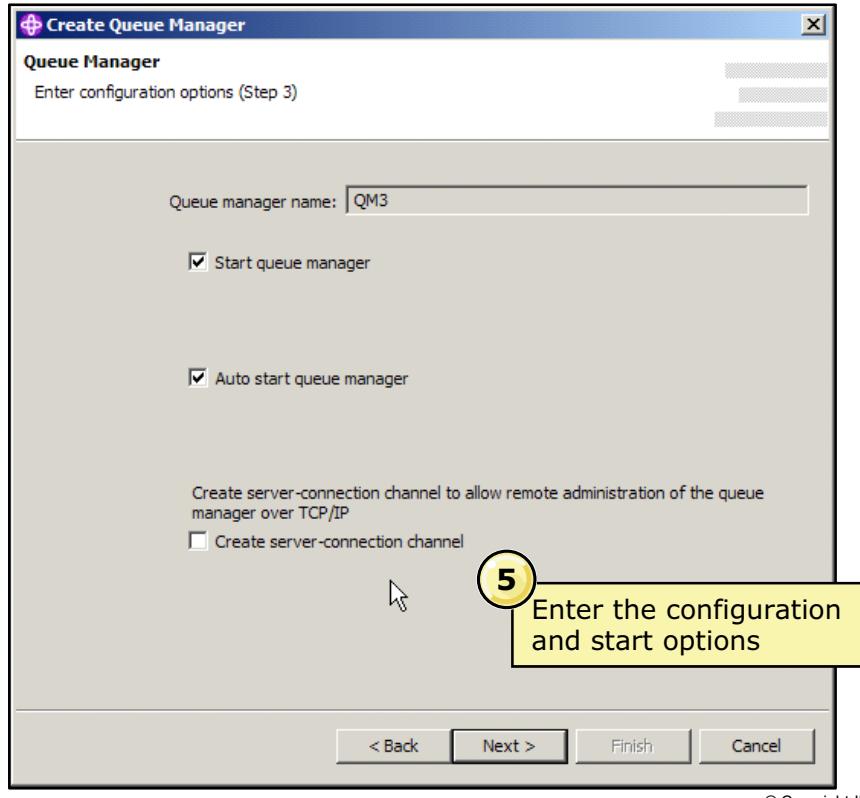
In IBM MQ for UNIX systems, the default number of log file pages is 1024, giving a log file size of 4 MB. The minimum number of log file pages is 64 and the maximum is 65535.

In IBM MQ for Windows, the default number of log file pages is 256, giving a log file size of 1 MB. The minimum number of log file pages is 32 and the maximum is 65535.

Queue manager log files are described in more detail throughout this course.



## Creating a local queue manager (4 of 5)



© Copyright IBM Corporation 2014

Figure 4-9. Creating a local queue manager (4 of 5)

WM2091.0

### Notes:

The next step for creating a queue manager is to enter the queue manager configuration options.

- Select **Start queue manager** if you want the queue manager to start immediately after it is created.
- Select **Auto start queue manager** if you want the queue manager to automatically start after a system restart.
- Select **Create server-connection channel** to allow another administrator on another computer to manage this queue manager on this computer. Enable remote administration for single-point administration.



## Creating a local queue manager (5 of 5)

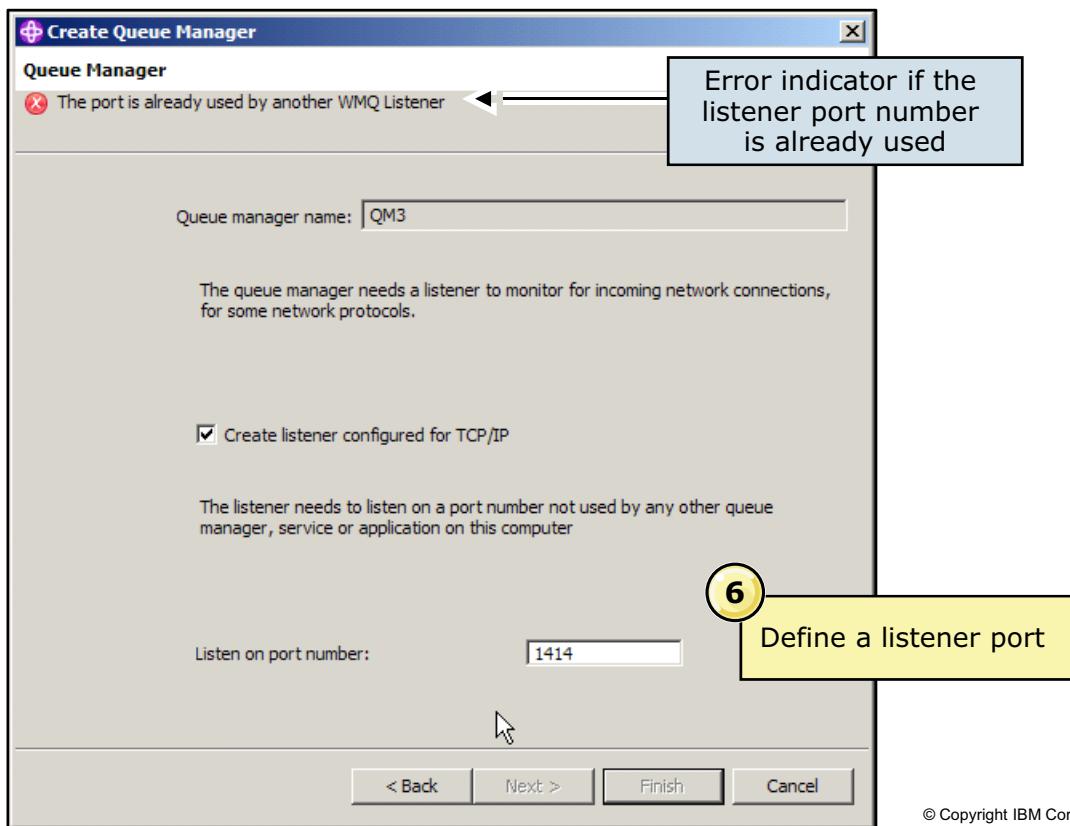


Figure 4-10. Creating a local queue manager (5 of 5)

WM2091.0

### Notes:

The final step for creating a queue manager is to create a listener for TCP/IP and identify the listener port value.

A unique port number is required for each listener on the same host. If MQ detects that another listener is using TCP/IP port number, an error message is displayed at the top of the page, as shown in the figure.



## Queue manager content view

MQ Explorer - Content

**Queue Manager QM1**

Connection QuickView:

Connection status	Connected
Connection type	Local
Connection name	
Channel name	
Channel definition table	
Refresh interval	15

Last updated: 13:44:55

Status QuickView:

Queue manager status	Running
Command server status	Running
Channel initiator status	Running
Connection count	21
Standby	Not permitted
Start date	Apr 25, 2014
Start time	1:33:17 PM
Installation name	Installation1

Last updated: 13:44:55

Properties QuickView:

Queue manager name	QM1
Description	
Platform	Windows
Command level	750
Version	07050001
Default transmission queue	

Connection QuickView

Status QuickView

Properties QuickView

© Copyright IBM Corporation 2014

Figure 4-11. Queue manager content view

WM2091.0

### Notes:

The Queue Manager content view is divided into three QuickView panes.

- The **Connection QuickView** shows the queue manager connection status information, such as connection status, connection type, and connection name.
- The **Status QuickView** shows the status of the queue manager. It includes the command server status, channel initiator status, and the installation name.
- The **Properties QuickView** shows some of the configurable properties of the queue manager such as the operating system, command level (version of MQ), and the default transmission queue.

## Grouping queue managers

- Queue managers can be grouped into *sets*
- Actions can be performed on a set of queue managers:
  - Show or hide all
  - Connect or disconnect all
  - Start or stop all local
  - Run default or custom tests
- Grouping can be done
  - Manually
  - Automatically
- Automatic grouping by filtering on:
  - Command level
  - Operating system
  - Any queue manager attribute by creating a custom filter
- Queue managers can be members of none, one, or many sets
- Sets cannot contain other sets

© Copyright IBM Corporation 2014

Figure 4-12. Grouping queue managers

WM2091.0

### Notes:

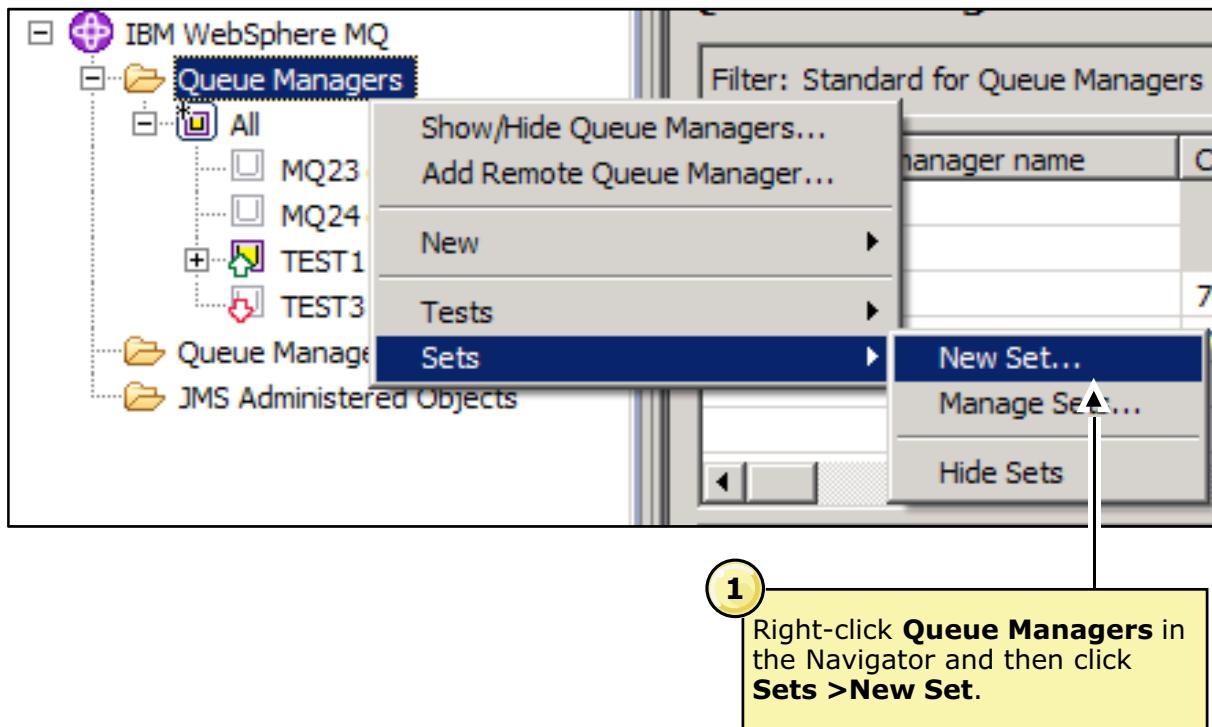
You can group queue managers into sets of MQ applications services for use in service-oriented architectures.

Grouping queue managers makes it more efficient to complete operations on a select set of queue managers at the same time rather than one at a time. For example, you can stop and start all queue managers in a group at the same time.

You can group queue managers manually by selecting the queue managers that are members of the group. You can also group queue managers automatically by defining filter conditions. For example, you can define a group that includes running queue managers, or queue managers that use the same dead-letter queue.



## Creating a queue manager set (1 of 7)



© Copyright IBM Corporation 2014

Figure 4-13. Creating a queue manager set (1 of 7)

WM2091.0

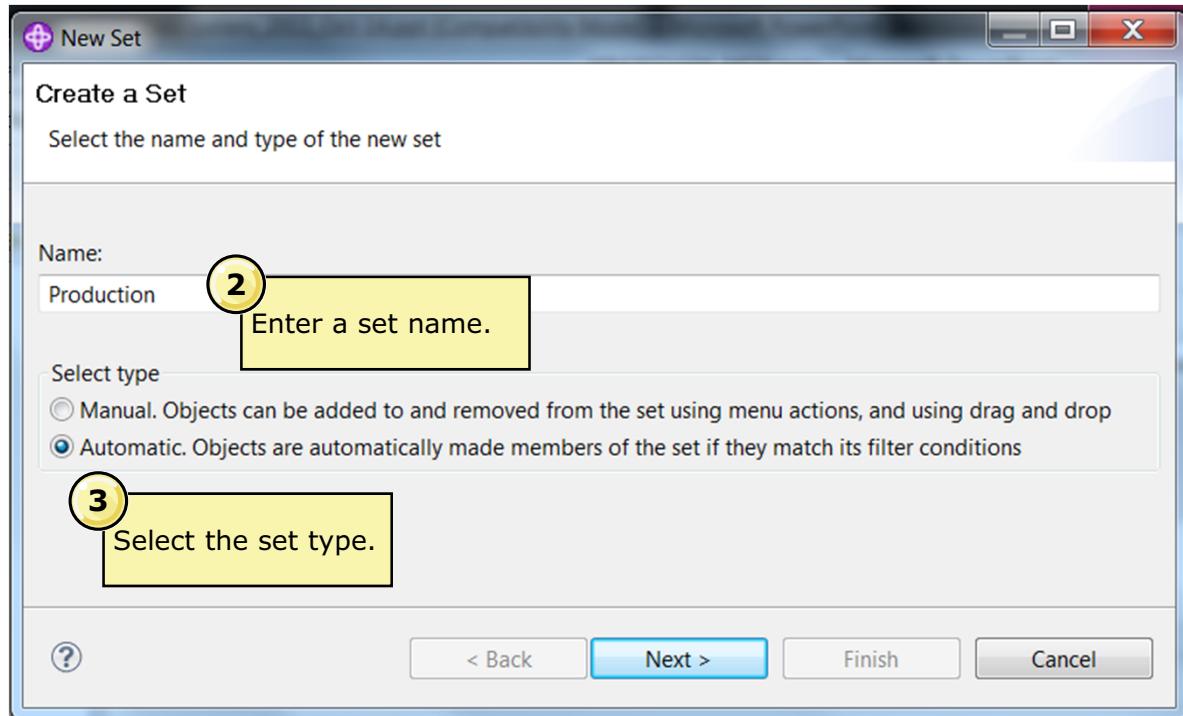
### Notes:

The first step for creating a queue manager set is:

1. Right-click **Queue Managers** in the MQ Explorer Navigator and then click **Sets > New Set**.



## Creating a queue manager set (2 of 7)



© Copyright IBM Corporation 2014

Figure 4-14. Creating a queue manager set (2 of 7)

WM2091.0

### Notes:

The next steps for creating a queue manager set are:

2. Enter a descriptive set name.

For example, create a set that is named **Production** that includes all queue managers that are used in a production environment.

3. Select **Manual**, if you want to select the queue managers to add to the set by name. Select **Automatic** if you want to use a filter to determine the queue managers that are added to the set.

## Creating a queue manager set (3 of 7)

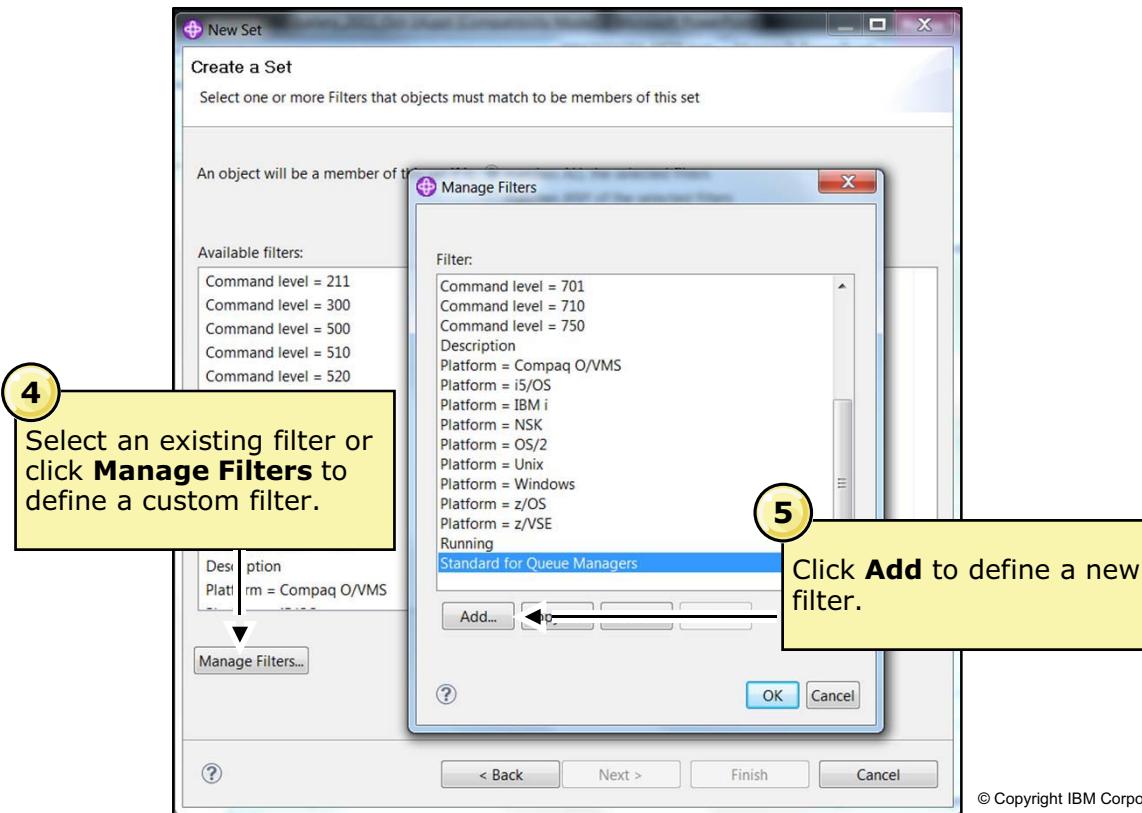


Figure 4-15. Creating a queue manager set (3 of 7)

WM2091.0

### Notes:

If you selected **Manual** as the set type, the final step for creating a queue manager set is to select the queue managers that are members of the set.

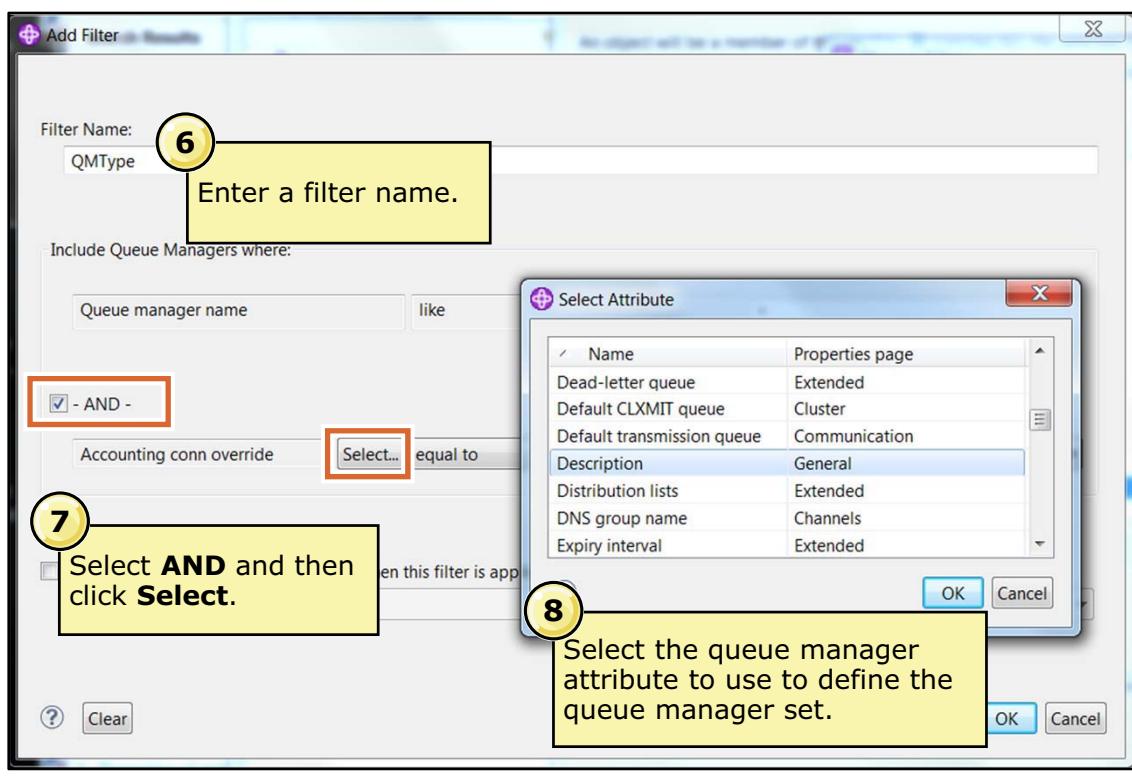
If you selected **Automatic** for the queue manager set type, the next steps are to:

4. Select the filter that automatically adds queue managers. For example, you can create a queue manager set that is named "Windows Queue Managers". You can then select the filter condition of **Platform = Windows** so that any queue managers that are running on Windows are automatically added to the set.

As an option, click **Manage Filters** to define a custom filter. The example in this figure and the next set of figures show the steps for defining a custom filter.

5. Click **Add** on the **Manage Filters** window to define a new filter.

## Creating a queue manager set (4 of 7)



© Copyright IBM Corporation 2014

Figure 4-16. Creating a queue manager set (4 of 7)

WM2091.0

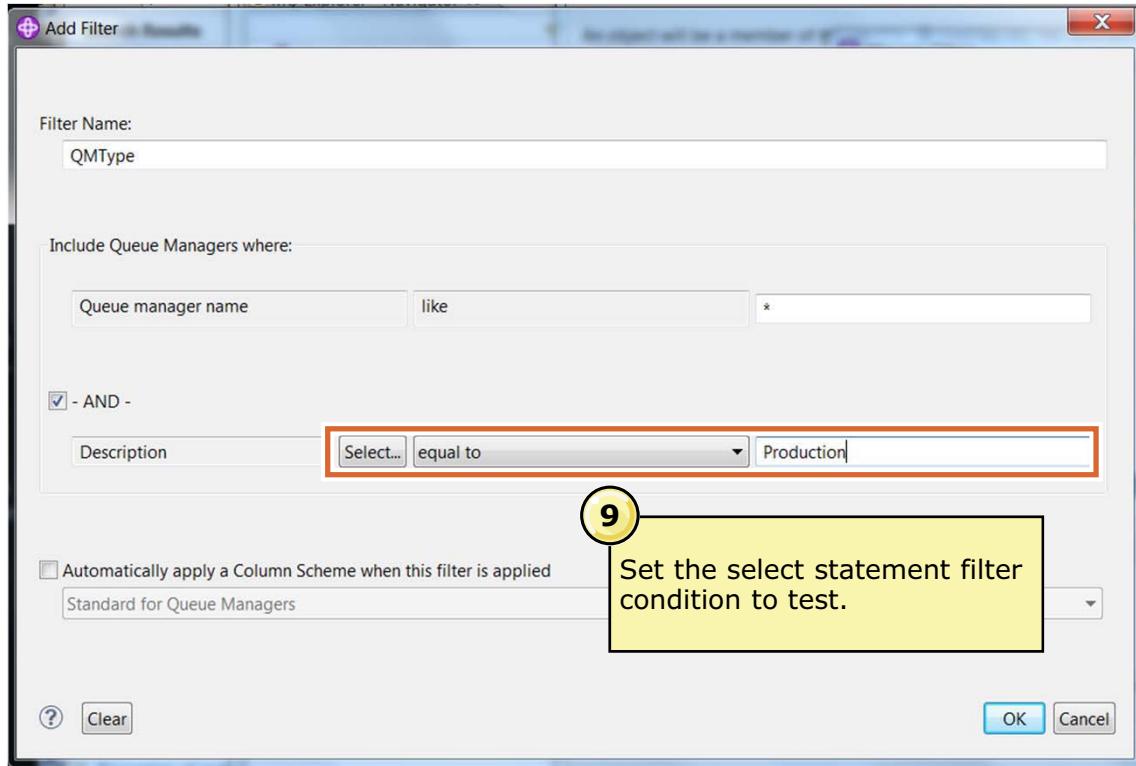
### Notes:

This figure shows the next steps for creating a custom queue manager set filter for all queue managers.

6. Enter a unique descriptive filter name.
7. Select the **AND** check box and then click **Select** to continue to define the custom filter expression.
8. Select the queue manager attribute in the **Select Attribute** window and then click **OK**.



## Creating a queue manager set (5 of 7)



© Copyright IBM Corporation 2014

Figure 4-17. Creating a queue manager set (5 of 7)

WM2091.0

### Notes:

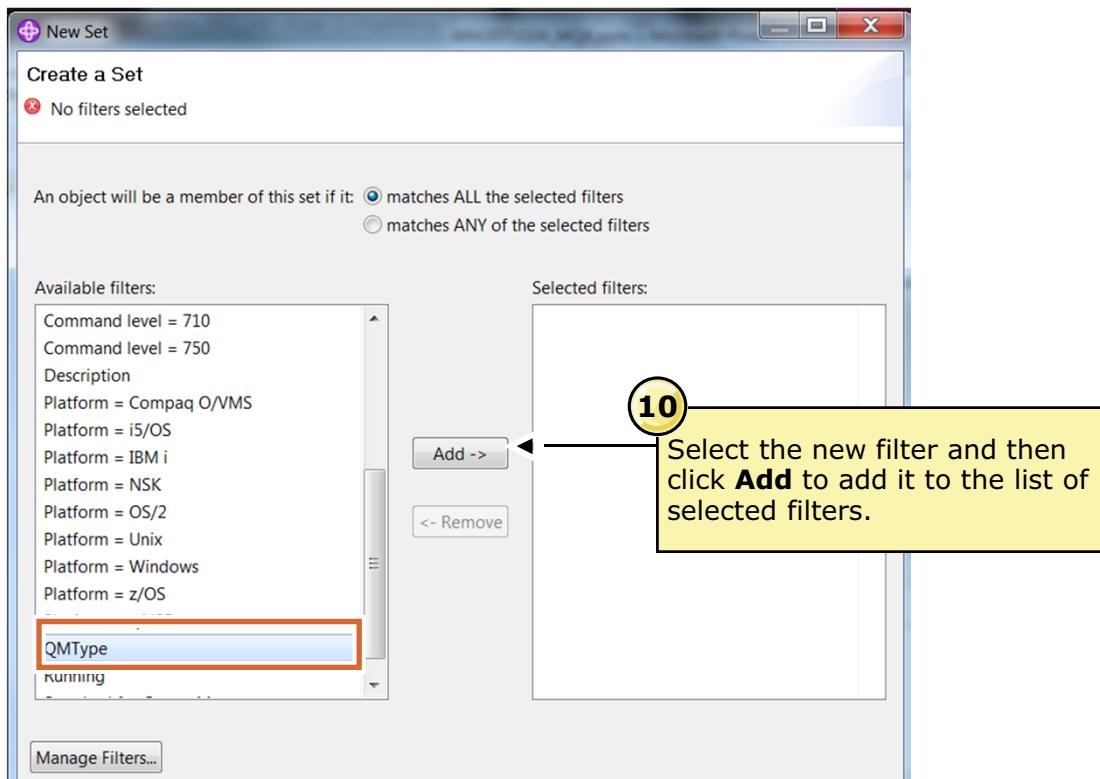
This figure shows the next step for defining a custom filter expression.

9. Select the expression condition such as **equal to** or **like** and then enter the string to evaluate.

In this example, the queue manager is automatically added to the group if the queue manager **Description** attribute is set to **Production**.



## Creating a queue manager set (6 of 7)



© Copyright IBM Corporation 2014

Figure 4-18. Creating a queue manager set (6 of 7)

WM2091.0

### Notes:

This figure shows the final step for defining an automatic queue manager set.

10. Select the filter in the **Available filters** pane and then click **Add** to move it to the **Selected filters** pane.

If you add more than one filter, be sure to verify the options for whether an object must match ALL selected filters or ANY of the selected filters.



## Creating a queue manager set (7 of 7)

IBM WebSphere MQ Explorer (Installation1)

File Edit Window Help

MQ Explorer - Navigator

- IBM WebSphere MQ
  - Queue Managers
    - All
      - DEVQMGR
      - IB9QMGR
      - QM1
      - QMC02
    - Production**
      - QMC02**
  - JMS Administered Objects
  - Managed File Transfer
  - Service Definition Repositories
  - Integration Nodes
  - Bar Files

MQ Explorer - Content

Connection QuickView:

Connection status	Connected
Connection type	Local
Connection name	

Last updated: 09:06:49

Status QuickView:

Queue manager status	
Command server status	
Channel initiator status	
Connection count	
Standby	Not permitted

Last updated: 09:06:49

Properties QuickView:

Description	Production
-------------	------------

Last updated: 09:06:49

The queue manager set folder is added to the **Queue Managers** folder and contains all queue managers that match the filter criteria.

© Copyright IBM Corporation 2014

Figure 4-19. Creating a queue manager set (7 of 7)

WM2091.0

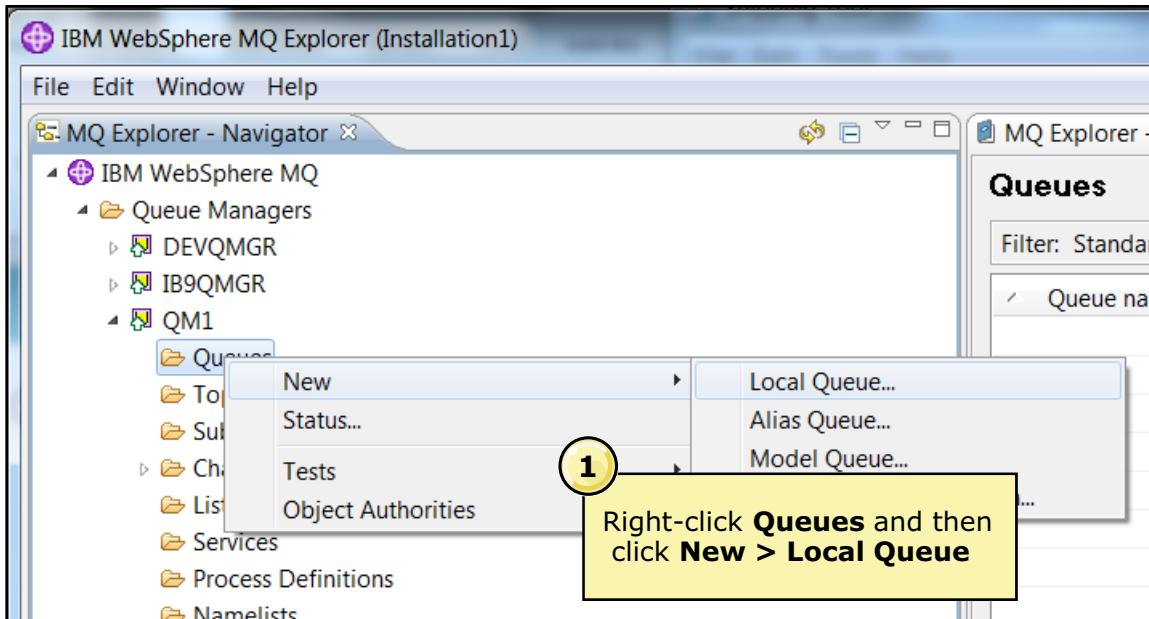
### Notes:

After you define a queue manager group, the MQ Explorer Navigator shows the new queue manager set and another queue manager set that is named **All**. The **All** set contains all queue managers that MQ Explorer can manage.

In this example, the queue manager that is named QMC02 is automatically added to the **Production** set because the **Description** attribute is set to **Production**. You can create another queue manager set that is named **Development** and define another custom filter where **Description** is equal to **Development** to further separate and manage the queue managers by use.



## Creating a local queue (1 of 3)



© Copyright IBM Corporation 2014

Figure 4-20. Creating a local queue (1 of 3)

WM2091.0

### Notes:

It is possible to define queues for any local queue manager that is connected to MQ Explorer.

To create a new local queue:

1. Right-click the **Queues** folder under the queue manager in the Navigator and then click **New > Local Queue**.



Figure 4-21. Creating a local queue (2 of 3)

WM2091.0

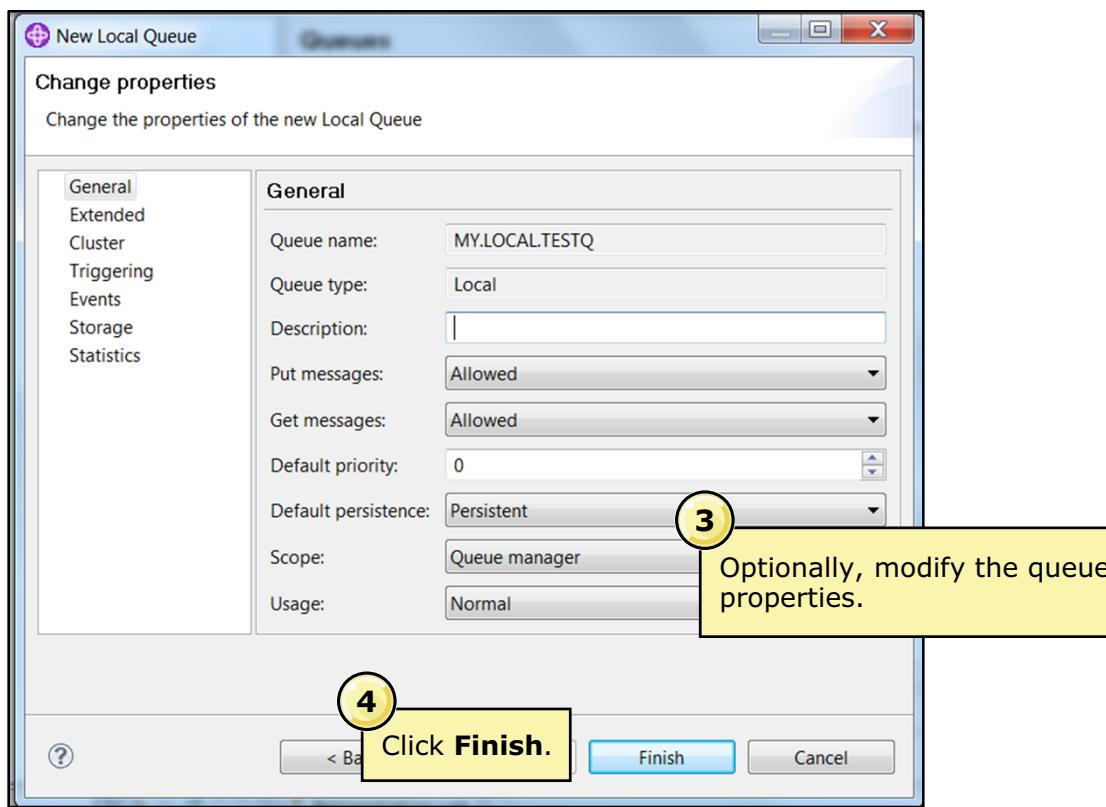
## Notes:

2. The next step for creating a new queue is to enter a queue name.

By default, MQ uses a default SYSTEM queue as the model for the new queue. You can specify a different queue to use as the model for this queue if you want.

There is also an option to automatically start another wizard to create a matching JMS queue. JMS is introduced later in this course.

## Creating a local queue (3 of 3)



© Copyright IBM Corporation 2014

Figure 4-22. Creating a local queue (3 of 3)

WM2091.0

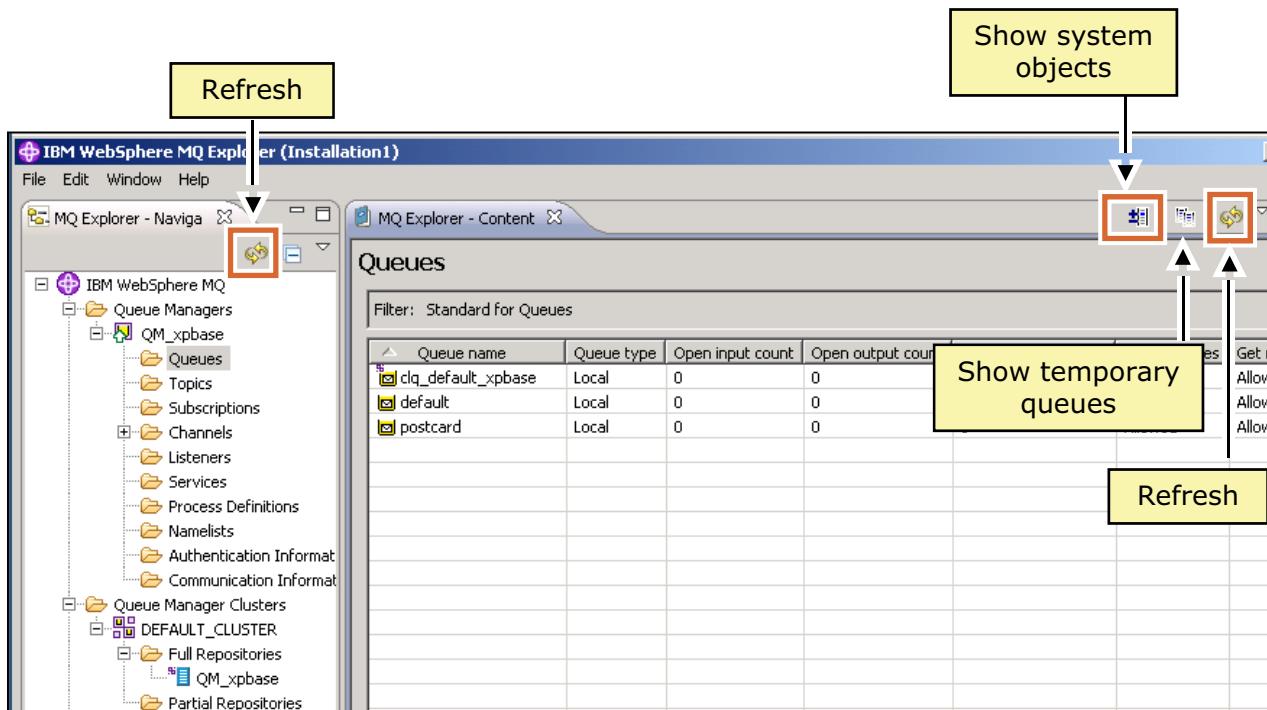
### Notes:

This figure shows the next steps for creating a local queue.

3. Modify the queue properties, if necessary. The queue properties are categorized by function:
  - General
  - Extended
  - Cluster
  - Triggering
  - Events
  - Storage
  - Statistics
4. Click **Finish**.



## Shortcut icons



© Copyright IBM Corporation 2014

Figure 4-23. Shortcut icons

WM2091.0

### Notes:

As shown in the figure, MQ Explorer has shortcut icons for refreshing lists and views and showing system objects.

More shortcut icons are available based on the selected content. For example, when the content area contains the list of queues, a shortcut icon is available for showing temporary queues and SYSTEM queues.



## Queue content view

**Double-click a queue to open the **Properties** view, or right-click a queue for more actions.**

Queue name	Queue type	Open input count	Open output count	Current queue depth	Put messages	Get messages
clq_default_xpbase	Local	0	0	0	Allowed	Allowed
default	Local	0	0	0	Allowed	Allowed
postca		0	0	0	Allowed	Allowed

Scheme: Standard for Queues - Distributed  
Last updated: 13:33:05 (3 items)

© Copyright IBM Corporation 2014

Figure 4-24. Queue content view

WM2091.0

### Notes:

The **Queue** content view is used to view, clear, put messages, and obtain status information for the selected queue. It is also used to define object authorities on the queue that identify groups and users that can put and get messages.

You can modify queue properties by double-clicking a queue or right-clicking a queue and clicking **Properties**.

Right-click the queue for actions such as putting test messages, getting messages, and browsing messages.

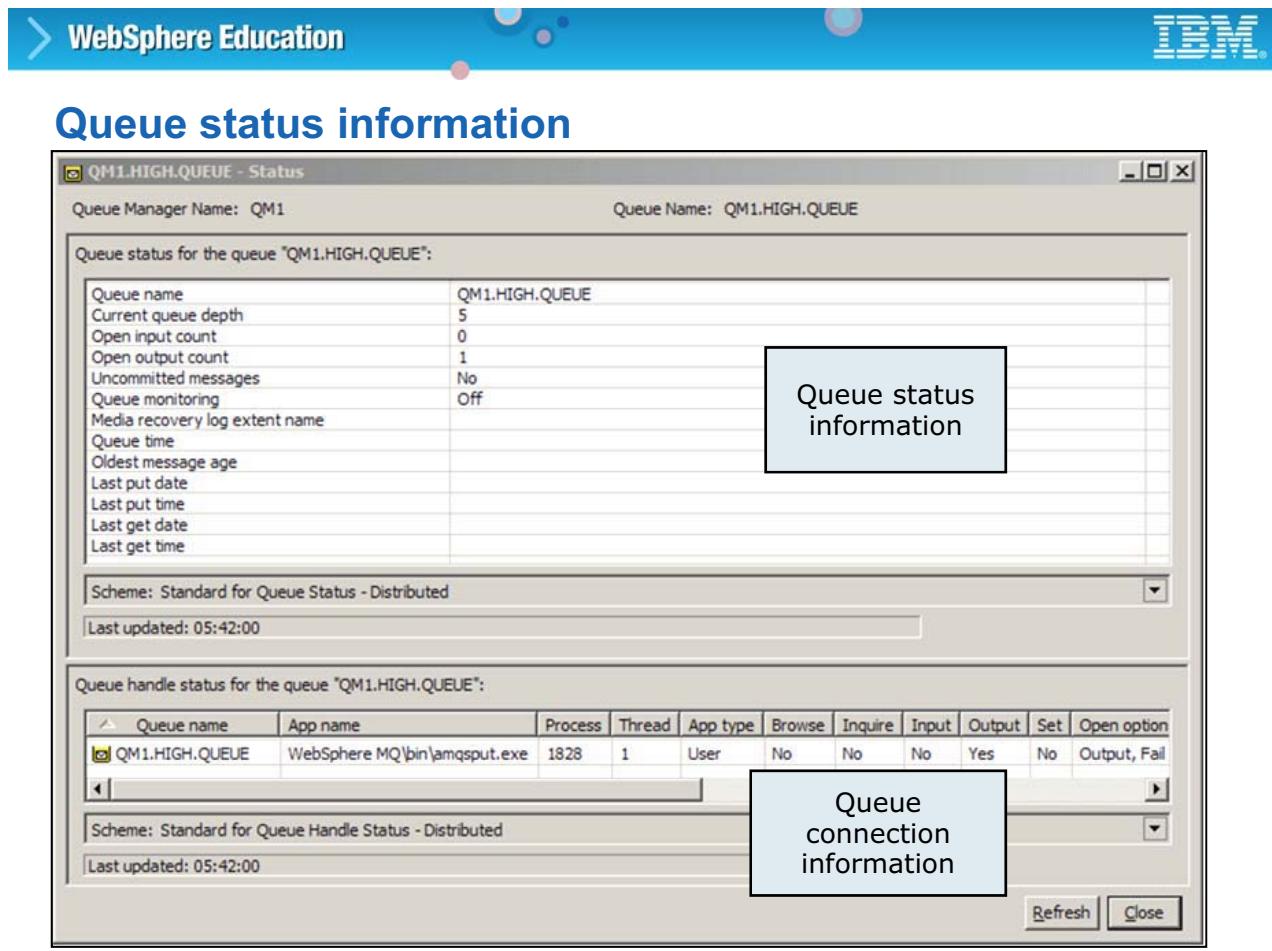


Figure 4-25. Queue status information

WM2091.0

## Notes:

The queue status information view can help to determine real-time queue usage information. This information can assist application programmers with diagnostic information.

Obtain status information by right-clicking the queue and then selecting **Status**.

Status information describes the following queue attributes:

- Number of uncommitted messages
- Last put and get times
- Queue connection information
- Application usage information

The queue connection view, which is shown in the figure, contains the active applications that are using the queue. It contains the process ID, number of threads, and open options (**MQGET** or **MQPUT**). It also shows what open options are requested and the user ID that the process is running under. The queue connection information can be instrumental in assisting application testing.

The queue connection example in the figure shows an **amqspput** application with process number 1828 by using one thread.

## Viewing queue filters

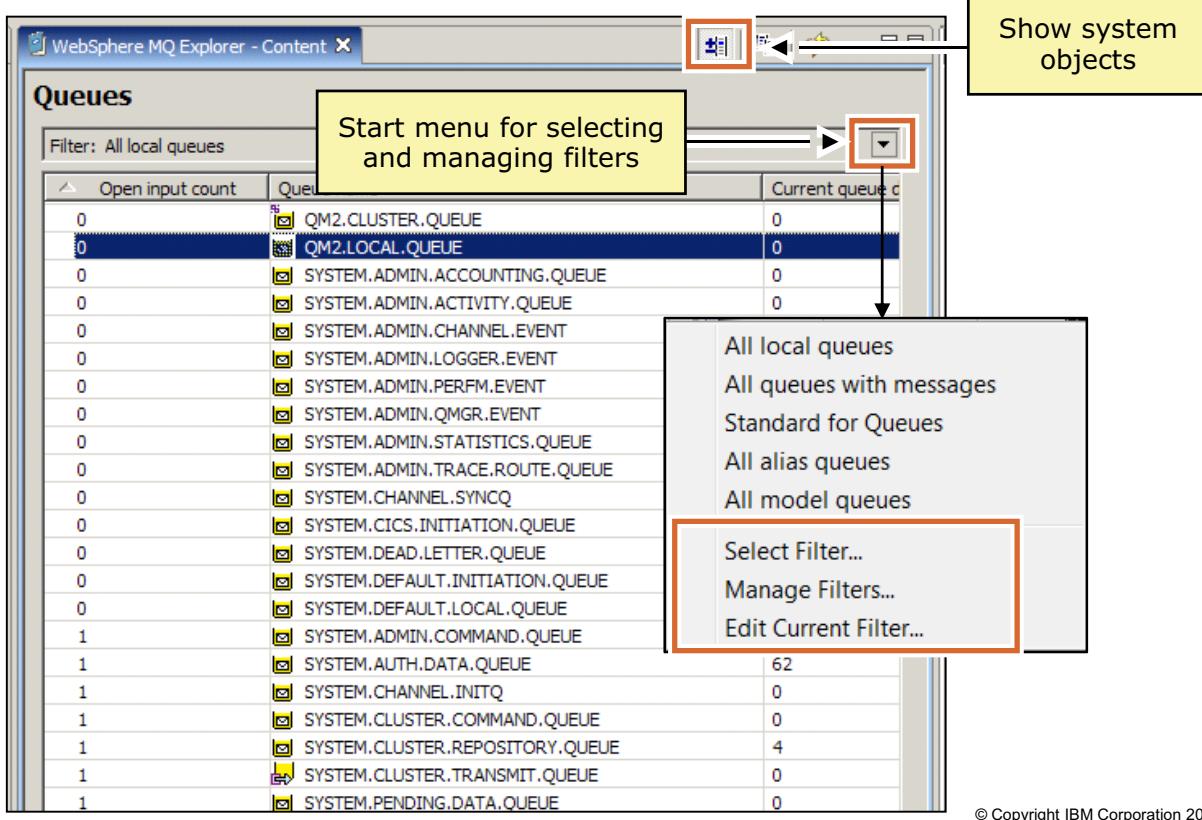


Figure 4-26. Viewing queue filters

WM2091.0

### Notes:

Filters provide a way to selectively view queues that meet certain criteria such as queues that contain messages. A filter allows the display of queues that meet the filter criteria and hides queues that do not conform to the selection criteria.

Click the **Filter** menu icon to access the options for selecting and managing the filters.

The **Show system objects** icon is a toggle button. When active, it shows the system queues for the queue manager.

# WebSphere Education

## Managing queue filters

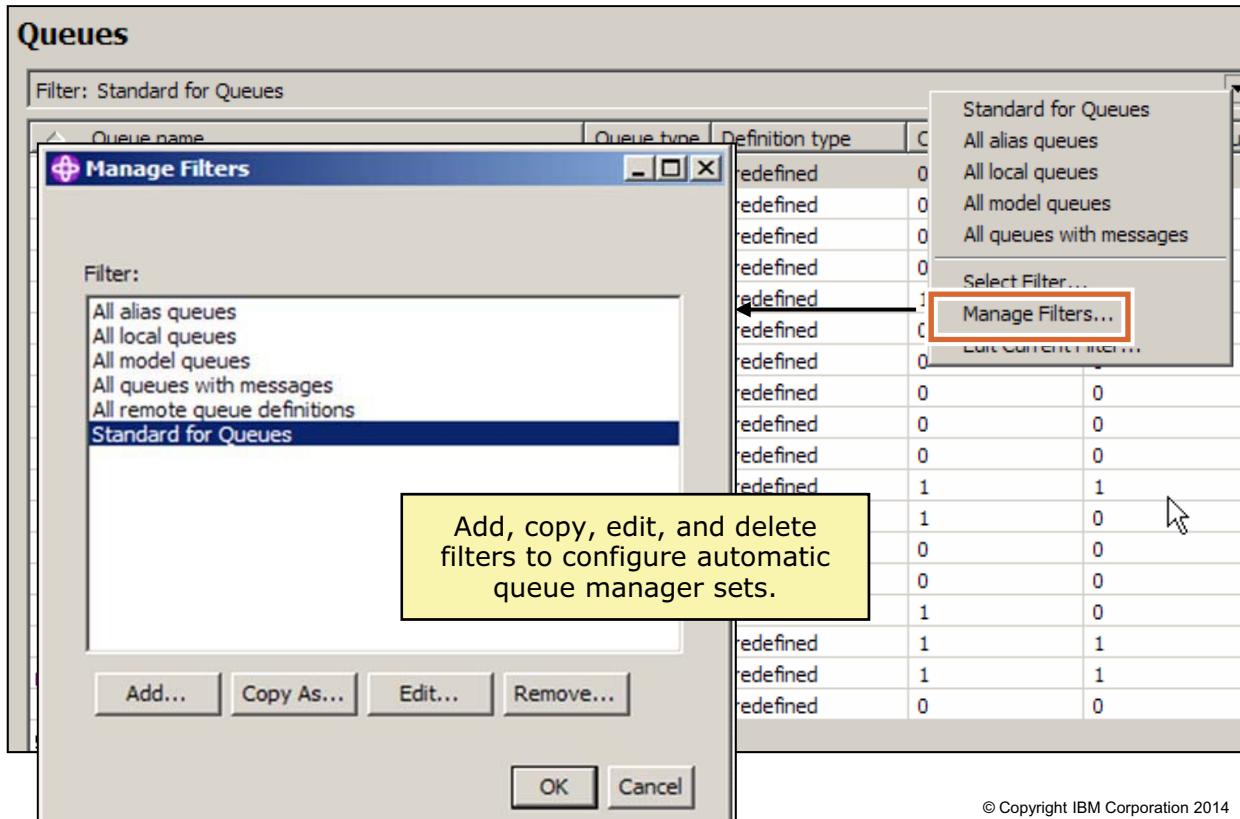


Figure 4-27. Managing queue filters

WM2091.0

### Notes:

The **Manage Filters** option allows you to create, modify, and delete queue display filters.

In the figure, the list of filters includes a filter to show all alias queues, all local queues, all model queues, all queues with messages, and all remote queues.

In the **Manage Filters** window, you can:

- Click **Add** to create a filter.
- Click **Copy As** to copy a filter.
- Click **Edit** to edit filters to add, remove, or change the criteria that are set for the filter.
- Click **Remove** to delete a filter.

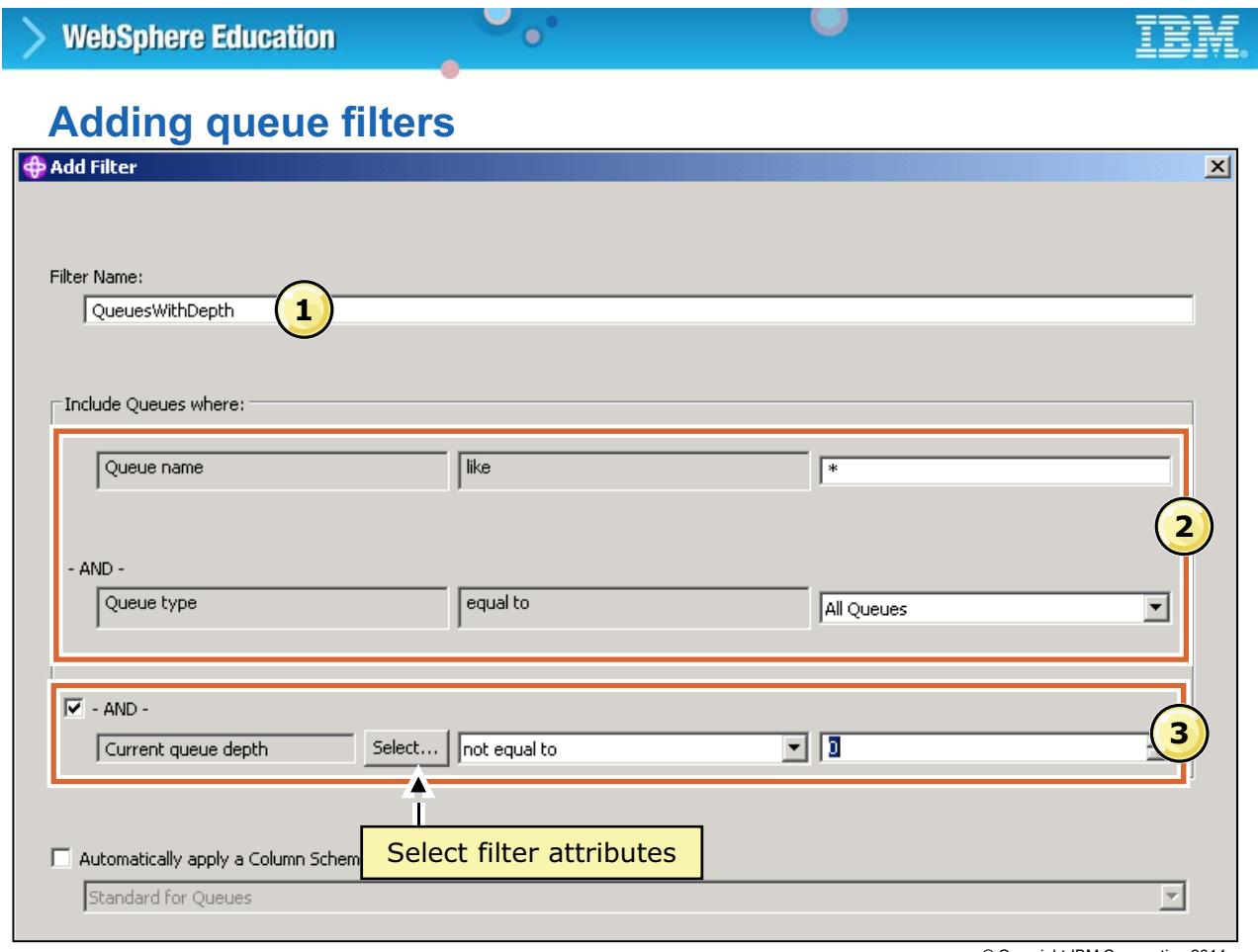


Figure 4-28. Adding queue filters

WM2091.0

### Notes:

To add a queue display filter in MQ Explorer, click **Add** on the **Manage filters** window. The figure shows the next steps for defining a custom filter.

1. Enter a descriptive filter name.
2. Select the filter conditions.
3. Optionally, add a condition by selecting **AND** and the filter attributes. Click **Select** to select the filter attribute from a list of valid attributes.

In the example, a filter that is named **QueuesWithDepth** filters on all queues. The filter expression displays the queues where the current queue depth is not equal to zero.

## Queue filter attributes

The screenshot shows a 'Select Attribute' dialog box. On the left, there is a list of attributes with their types: Name (Properties page), Archive, Backout requeue queue, Backout threshold, Base object, Base type, Cluster name, Cluster name list, CLWL queue priority, CLWL queue rank, CLWL use queue, Coupling facility name, Current queue depth, Default bind type, Default input open option, Default persistence, Default priority, Default put response type, Default read ahead, Definition type, Description, Distribution lists, and Get messages. The 'Backout requeue queue' attribute is selected. On the right, a detailed list of filter attributes is shown in a grid:

Name	Properties page	
Archive	Storage	Storage
Backout requeue queue	Storage	Storage
Backout threshold	Storage	Storage
Base object	General	General
Base type	General	General
Cluster name	Cluster	Cluster
Cluster name list	Cluster	Cluster
CLWL queue priority	Cluster	Cluster
CLWL queue rank	Cluster	Cluster
CLWL use queue	Cluster	Cluster
Coupling facility name	Storage	Storage
Current queue depth	Statistics	Statistics
Default bind type	Cluster	Cluster
Default input open option	Extended	Extended
Default persistence	General	General
Default priority	General	General
Default put response type	Extended	Extended
Default read ahead	Extended	Extended
Definition type	Extended	Extended
Description	General	General
Distribution lists	Extended	Extended
Get messages	General	General
Harden get backout		Storage
Index type		Extended
Initiation queue		Triggering
Max message length		Extended
Max queue depth		Extended
Message delivery sequence		Extended
NPM class		Storage
Open input count		Statistics
Open output count		Statistics
Page set ID		Extended
Pipe name		Extended
Process name		Triggering
Property control		Extended
Put messages		General
Queue accounting		Statistics
Queue depth high events		Events
Queue depth high limit		Events
Queue depth low events		Events
Queue depth low limit		Events
Queue depth max events		Events
Queue monitoring		Statistics
Queue service interval		Events
Queue service interval events		Events
Queue statistics		Statistics

© Copyright IBM Corporation 2014

Figure 4-29. Queue filter attributes

WM2091.0

### Notes:

The figure shows the filter attributes that are available when specifying an extra **AND** condition on the queue filter expression. Examples of filter attributes are cluster name, current queue depth, maximum queue depth, and process name.

The filter attributes are displayed when you click **Select** in the **Add filter** view.

## Comparing queues (1 of 2)

- Can compare two like resources
- Can compare resources across queue managers

Queue name	Queue type	Definition type	Open input count	Open output count
QM2.LOCAL	Local	Predefined	0	0
SYSTEM.ADMIN.QUEUE	Local	Predefined	0	0
SYSTEM.ADMIN.STATUS	Local	Predefined	0	0
SYSTEM.ADMIN.DELETE	Local	Predefined	0	0
SYSTEM.ADMIN.CLEAR	Local	Predefined	1	1
SYSTEM.ADMIN.PUT	Local	Predefined	0	0
SYSTEM.ADMIN.BROWSE	Local	Predefined	0	0
SYSTEM.ADMIN.PROPERTIES	Local	Predefined	0	0
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	Local	Predefined	0	0
SYSTEM.AUTH.DATA.QUEUE	Local	Predefined	1	1
SYSTEM.CHANNEL.INITQ	Local	Predefined	1	0
SYSTEM.CHANNEL.SYNCQ	Local	Predefined	0	0
SYSTEM.CICS.INITIATION.QUEUE	Local	Predefined	0	0
SYSTEM.CLUSTER.COMMAND.QUEUE	Local	Predefined	1	0
SYSTEM.CLUSTER.REPOSITORY.QUEUE	Local	Predefined	1	1

© Copyright IBM Corporation 2014

Figure 4-30. Comparing queues (1 of 2)

WM2091.0

### Notes:

Using MQ Explorer, you can compare queues. The queue compare functions can be used as a migration aid to verify queue attributes after they are moved into production, for example.

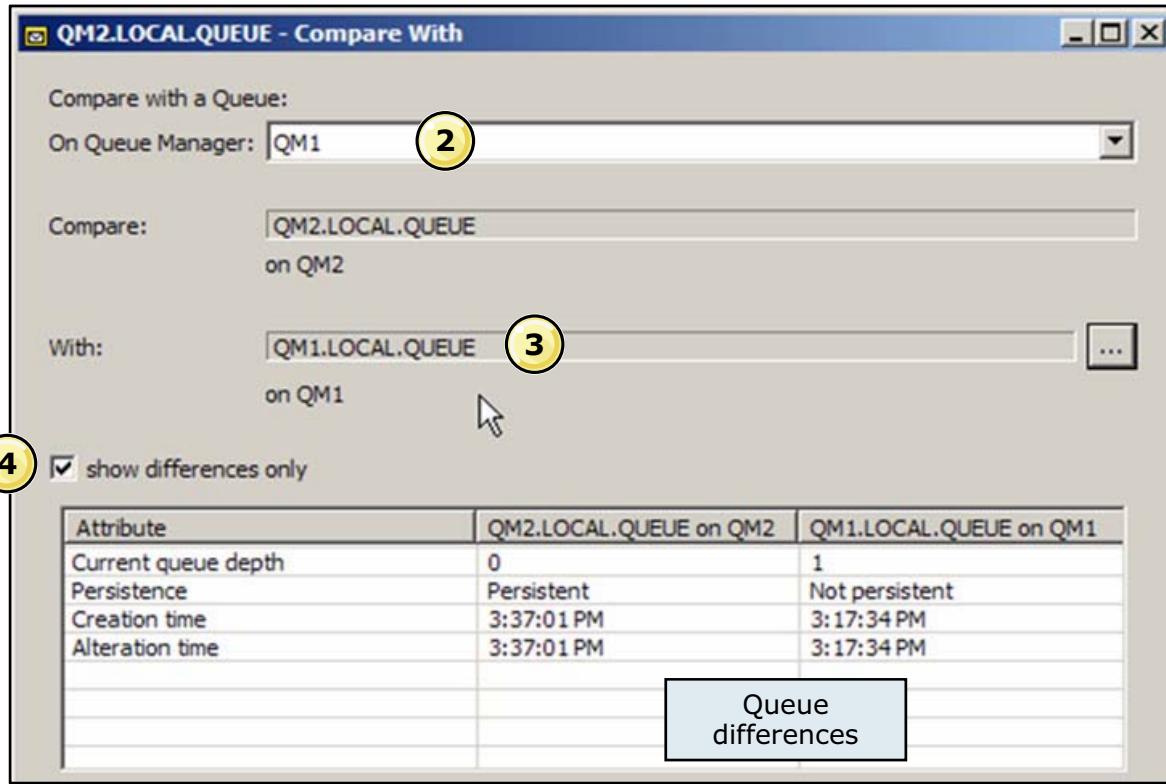
To compare two queues:

1. Right-click the first queue and then click **Compare with...**.

WebSphere Education

IBM

## Comparing queues (2 of 3)



© Copyright IBM Corporation 2014

Figure 4-31. Comparing queues (2 of 3)

WM2091.0

### Notes:

As shown in the figure, the next steps for comparing two queues are:

2. Select the queue manager for the second queue.
3. Select the queue in the **With** field.
4. Optionally, select **show differences only** to display the queue properties that are different. If you do not select **show differences only**, the list shows all the properties for both queues.

The example in the figure compares two queues from two different queue managers, QM1 and QM2.



## IBM MQ object definition tests

- General tests
  - Queue manager names
  - Dead-letter queue definitions
  - FFST error log
  - Stopped queue managers
  - Verify default transmission queue
- Queue tests
  - Identify full queues
  - Verify alias queue definitions
  - Verify queue names
  - Verify that queues are get-enabled
  - Verify that queues are put-enabled
  - Verify remote queue definitions
  - Verify use of transmission queue in queues

© Copyright IBM Corporation 2014

Figure 4-32. IBM MQ object definition tests

WM2091.0

### Notes:

MQ Explorer can run a group of tests that can be used to test the general functions of MQ and test queues and queue connectivity.

This figure summarizes the MQ Explorer default tests.

## IBM MQ General tests

- Queue manager names
  - Looks for similar names
  - Displays warnings for queue managers that are hosted on different machines but with identical names
- Dead-letter queue definitions
  - Warning for any queue manager that does not have a dead-letter queue
  - Error for any queue manager that has dead-letter queue attributes that are not valid;
- FFST error log: Displays an error if any FFST logs were written on the local system
- Stopped queue managers: Displays a warning for each queue manager that is stopped
- Verify default transmission queues: Displays errors for any invalid uses of the **Default Transmission Queue** attribute

© Copyright IBM Corporation 2014

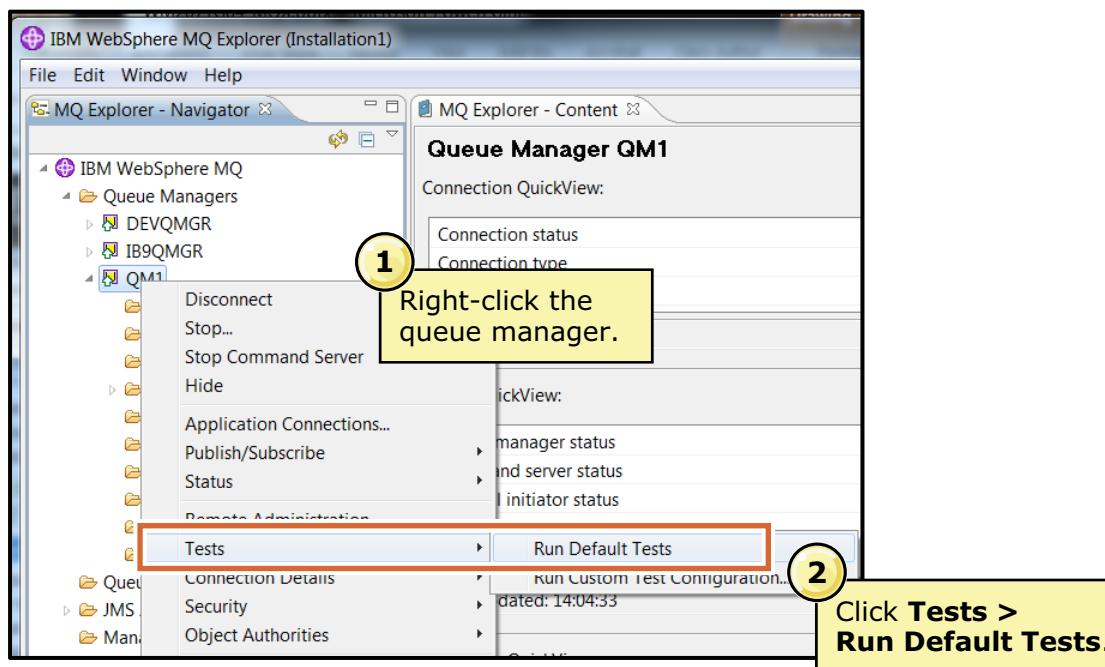
Figure 4-33. IBM MQ General tests

WM2091.0

### Notes:

The MQ Explorer tests provide some basic information about queue manager names, dead-letter queue definitions, the Failure First Support Technology (FFST) error log, stopped queue managers, and default transmission queues.

## Running the IBM MQ tests



© Copyright IBM Corporation 2014

Figure 4-34. Running the IBM MQ tests

WM2091.0

### Notes:

To run the MQ tests:

1. Right-click a queue manager in the Navigator.
2. Click **Tests > Run Default Tests**.



## IBM MQ test results example

MQ Explorer - Test Results X  
1 error, 3 warnings, 40 infos

Description	Object name	Category
✖ 1 errors have been written to the FFST (first-failure support technology) directory		Queue Manager / General
⚠ SSL key repository file cannot be found	QM1	Queue Manager / SSL
⚠ Stash file for SSL key repository cannot be found	QM1	Queue Manager / SSL
⚠ No dead-letter queue is defined for QM1	QM1	Queue Manager / General
ⓘ Test completed: 'Verify process names'	QM1	Queue Manager / Triggering
ⓘ Test completed: 'Verify initiation queue definitions'	QM1	Queue Manager / Triggering
ⓘ Test completed: 'Verify use of triggered queues'	QM1	Queue Manager / Triggering
ⓘ Test completed: 'Verify process definitions of queues'	QM1	Queue Manager / Triggering
ⓘ Test completed: 'Verify trigger data in queue definitions'	QM1	Queue Manager / Triggering
ⓘ Test completed: 'Verify process definitions'	QM1	Queue Manager / Triggering
ⓘ Test completed: 'Discover topic inheritance'	QM1	Queue Manager / Topics
ⓘ Test completed: 'Verify the basic topic setup'	QM1	Queue Manager / Topics
ⓘ Test completed: 'Verify topic names'	QM1	Queue Manager / Topics
ⓘ Test completed: 'Verify subscription names'	QM1	Queue Manager / Subscriptions
ⓘ Test completed: 'Verify the basic subscription setup'	QM1	Queue Manager / Subscriptions
ⓘ Test completed: 'Verify that channels have been restarted'	QM1	Queue Manager / SSL
ⓘ Test completed: 'Verify SSL key repository files'	QM1	Queue Manager / SSL
ⓘ Test completed: 'Verify SSL channel authentication'	QM1	Queue Manager / SSL
ⓘ Test completed: 'Verify SSL client authentication'	QM1	Queue Manager / SSL
ⓘ Test completed: 'Verify SSL peer values'	QM1	Queue Manager / SSL
ⓘ Test completed: 'Verify alias queue definitions'	QM1	Queue Manager / Queues
ⓘ Test completed: 'Verify that queues are put-enabled'	QM1	Queue Manager / Queues
ⓘ Test completed: 'Verify remote queue definitions'	QM1	Queue Manager / Queues
ⓘ Test completed: 'Identify full queues'	QM1	Queue Manager / Queues

© Copyright IBM Corporation 2014

Figure 4-35. IBM MQ test results example

WM2091.0

### Notes:

This figure is an example of the MQ test results. An entry is provided for each test, which is identified by a **Category**. The test results are listed in order with all tests that generated errors listed first, followed by all tests that produced warnings, and lastly all tests that completed successfully.



Figure 4-36. IBM MQ Explorer preferences

WM2091.0

### Notes:

MQ Explorer preferences can be used to change some of the default behaviors and display options. Preferences include default refresh intervals and enabled plug-ins.

To display the MQ Explorer preferences, click **Windows > Preferences**.

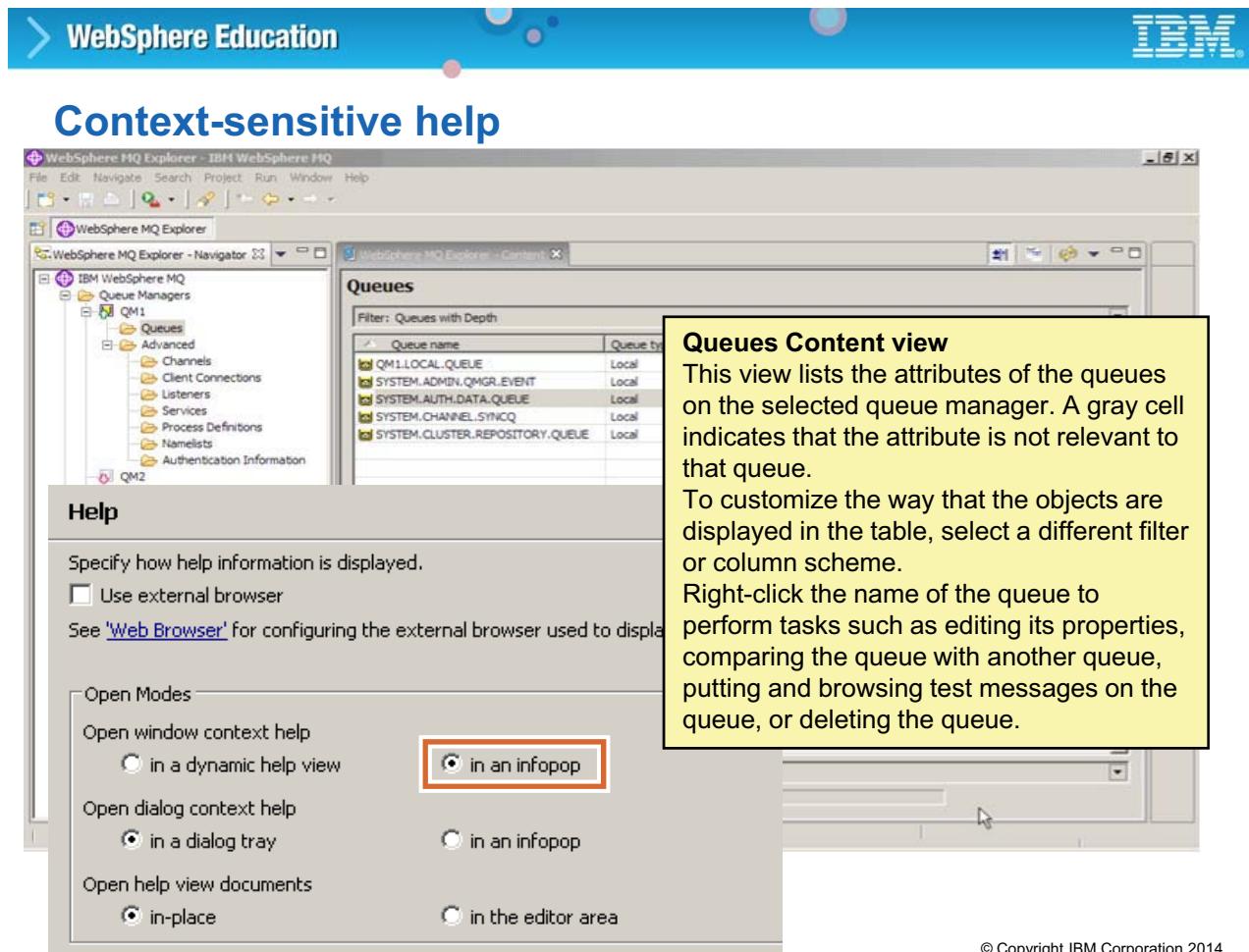


Figure 4-37. Context-sensitive help

WM2091.0

## Notes:

Quick access to reference documentation is supported in MQ Explorer by using context-sensitive help.

Context-sensitive help can be started by pressing **F1** on your keyboard while your cursor is over the required field.

The MQ Explorer Help preferences control the help information display mode. By default, the **Help** view is an extra view, but it can be changed to a window as shown in the figure, if you prefer. Help preferences are configured by clicking **Windows > Preferences > Help**.



## Unit summary

Having completed this unit, you should be able to:

- Use IBM MQ Explorer to create a local queue manager and queues
- Use IBM MQ Explorer to create and manage queue manager sets
- Use IBM MQ Explorer to run tests to verify IBM MQ object definitions

© Copyright IBM Corporation 2014

Figure 4-38. Unit summary

WM2091.0

### Notes:



## Checkpoint questions

1. Displaying the status of a selected queue shows:
  - a. Current queue depth
  - b. Name of the applications that are currently using the queue
  - c. The time of the last PUT operation on the queue
  - d. Whether messages are persistent
  - e. User ID of the process that is using the queue
  
2. True or False: Queue managers can be grouped into sets and the IBM MQ Explorer allows actions to be performed on all the queue managers that are defined to the set.

© Copyright IBM Corporation 2014

Figure 4-39. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

1.

2.



## Checkpoint answers

1. a, b, c, and e

2. True

© Copyright IBM Corporation 2014

---

Figure 4-40. Checkpoint answers

WM2091.0

### Notes:

## Exercise 2



Using IBM MQ Explorer to create queue managers and queues

© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.O

Figure 4-41. Exercise 2

WM2091.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Use IBM MQ Explorer to create a local queue manager, local queues, and alias queues
- Use IBM MQ Explorer commands to display and modify queue manager and queue properties
- Use IBM MQ Explorer to create a queue manager set

© Copyright IBM Corporation 2014

Figure 4-42. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercise Guide* for detailed instructions.



# Unit 5. Introduction to the Message Queue Interface (MQI)

## What this unit is about

This unit provides a brief introduction to the Message Queue Interface (MQI) calls that application programs use. This unit also describes the IBM MQ sample programs that you can use to test IBM MQ applications.

## What you should be able to do

After completing this unit, you should be able to:

- Recognize MQI calls in a program
- Explain the purpose of the fields in the IBM MQ message descriptor
- Use IBM MQ sample programs to put, get, and browse messages
- Use IBM MQ Explorer to put, get, and browse messages

## How you will check your progress

- Checkpoint
- Hands-on exercises

## References

IBM MQ product documentation



## Unit objectives

After completing this unit, you should be able to:

- Recognize MQI calls in a program
- Explain the purpose of the fields in the IBM MQ message descriptor
- Use IBM MQ sample programs to put, get, and browse messages
- Use IBM MQ Explorer to put, get, and browse messages

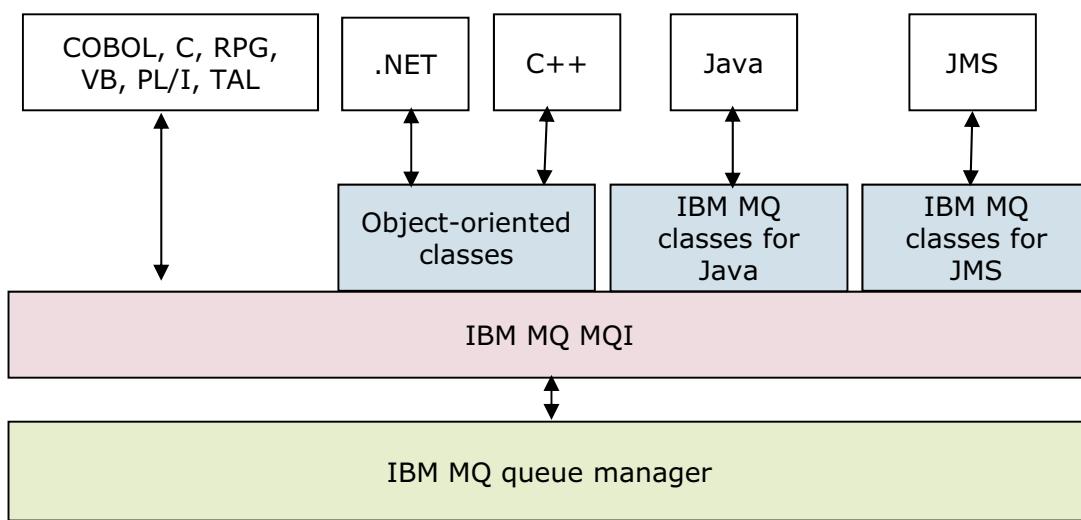
© Copyright IBM Corporation 2014

Figure 5-1. Unit objectives

WM2091.0

### Notes:

## IBM MQ Message Queue Interface (MQI)



- MQI enables external client applications to access queues and messages

© Copyright IBM Corporation 2014

Figure 5-2. IBM MQ Message Queue Interface (MQI)

WM2091.0

### Notes:

The IBM MQ Message Queue Interface (MQI) allows an application to access its queues and the messages they contain. The MQI is a simple application programming interface that is consistent across all operating systems that IBM MQ supports. The MQI effectively protects applications from needing to know how a queue manager physically manages messages and queues. The MQI allows full access to MQ messaging support.

You can develop client applications by using the MQ classes for .NET, MQ classes for Java, or MQ classes for Java Message Service (JMS). You can use Java and JMS clients on IBM i, UNIX, Linux, and Windows.

JMS is a specification of a portable application programming interface (API) for asynchronous messaging. JMS is an object-oriented Java API with a set of generic messaging objects for programmers to write event-based messaging applications.

MQ classes for .NET allow a program that is written in the .NET programming framework to connect to MQ as an MQ MQI client or to connect directly to an MQ server.

## Basic MQI calls

- Send messages
  - MQPUT: Put message
  - MQPUT1: Put one message
- Receive messages
  - MQGET: Get message
- Housekeeping calls
  - MQCONN: Connect queue manager
  - MQDISC: Disconnect queue manager
  - MQOPEN: Open object
  - MQCLOSE: Close object
- View or change properties
  - MQINQ: Inquire object attributes
  - MQSET: Set object attributes
- Perform transactions
  - MQBEGIN: Begin unit of work
  - MQCMIT: Commit changes
  - MQBACK: Backout changes

© Copyright IBM Corporation 2014

Figure 5-3. Basic MQI calls

WM2091.0

### Notes:

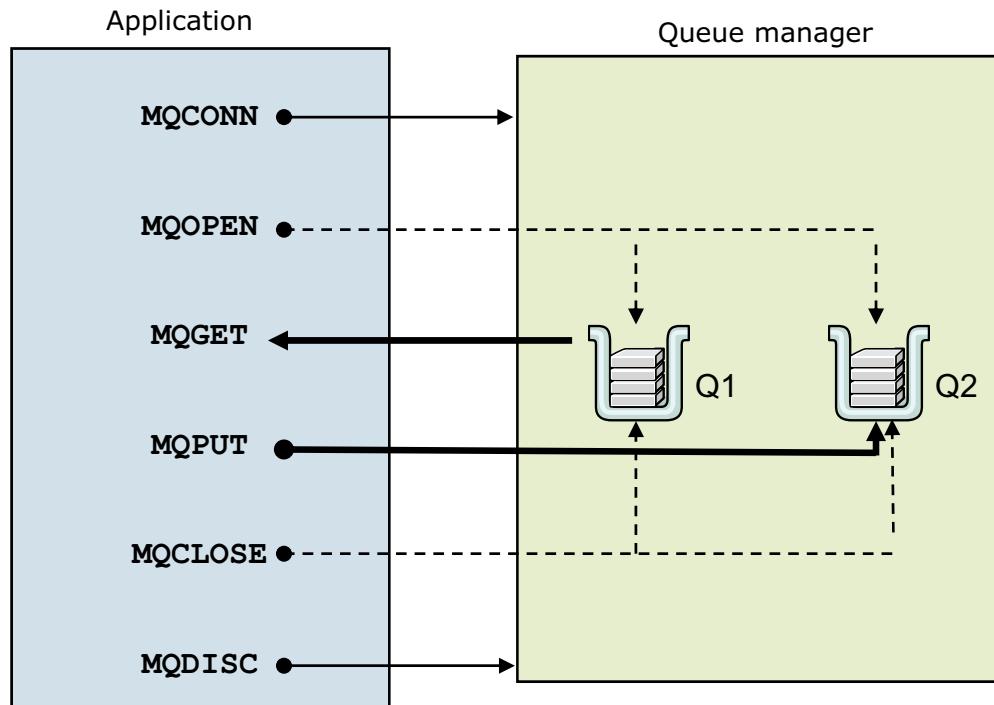
The most basic MQI calls allow an application to put a message on a queue and get a message from a queue.

- **MQPUT** and **MQPUT1** put a message on a named queue. Generally, a message is added to the end of a queue.
- **MQGET** gets a message from a named queue. Generally, a message is removed from the front of a queue.
- **MQCONN**, **MQCONNX**, and **MQDISC** allow an application to connect to a queue manager and disconnect from a queue manager. An application must connect to a queue manager before it can send any more MQI calls.
- **MQOPEN** and **MQCLOSE** allow an application to open a queue for specified operations and close the queue when access to it is no longer required. An application must open a queue before it can access it in any way; to put messages on it, or get messages from it.
- **MQINQ** and **MQSET** inquire and set the attributes of an object. All MQ objects, such as a queue, a process, and the queue manager object, have a set of attributes.

- **MQBEGIN**, **MQCMIT**, and **MQBACK** allow an application to put and get messages as part of a unit of work.

The MQI calls are described in detail in the *IBM MQ Application Programming Reference* guide.

## Basic MQI calls in use



© Copyright IBM Corporation 2014

Figure 5-4. Basic MQI calls in use

WM2091.0

### Notes:

The MQI is a simple call interface. The figure shows the most common calls that are used to connect and disconnect to a queue manager, and to put and get messages from a queue.

In a typical sequence:

- MQCONN connects to the queue manager.
- MQOPEN establishes access to an MQ object such as a queue.
- MQGET retrieves a message from a queue.
- MQPUT places a message on a queue or publishes to a new topic.
- MQCLOSE releases access to an MQ object.
- MQDISC disconnects from the queue manager.

## MQI notation examples

- IBM MQ application programming reference

```
MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength,
        Buffer, CompCode, Reason)
```

- Equivalent in C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
        BufferLength, Buffer, &CompCode, &Reason);
```

- Equivalent in COBOL

```
CALL "MQPUT1" USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
      BUFFERLENGTH, BUFFER, COMPCODE, REASON
```

© Copyright IBM Corporation 2014

Figure 5-5. MQI notation examples

WM2091.0

### Notes:

The IBM MQ Application Programming Reference defines the MQI. The figure shows an example of the notation that is used in that publication and others.

MQPUT1 is an example of an MQI call. The items in parentheses after MQPUT1 are its parameters.

The figure includes examples of an MQPUT1 call in both C and COBOL.

## Order of retrieving messages

- Application can retrieve messages on a queue in the same order as another application puts them on the queue, provided:
  - The messages all have the same priority
  - The messages are all put within the same unit of work, or all put outside of a unit of work
  - No other application is getting messages from the queue
  - The queue is local to the putting application
  - Messages might be interspersed with messages put by other applications
- If the queue is not local to the putting application, the order of retrieval is still preserved provided:
  - The first three conditions that are listed in the preceding list still apply
  - Only a single path is configured for the messages
  - No message is put on a dead-letter queue
  - No nonpersistent messages are transmitted over a fast message channel

© Copyright IBM Corporation 2014

Figure 5-6. Order of retrieving messages

WM2091.0

### Notes:

On a fast message channel, non-persistent messages overtake persistent messages on the same channel and so arrive out of sequence. This behavior is because the receiving MCA commits a non-persistent message on its destination queue as soon as it receives it instead of waiting for the end of the batch.

Most applications do not have a strict need to process messages in the same order as they were created. But some applications might have such a requirement, even a legal obligation to do so.

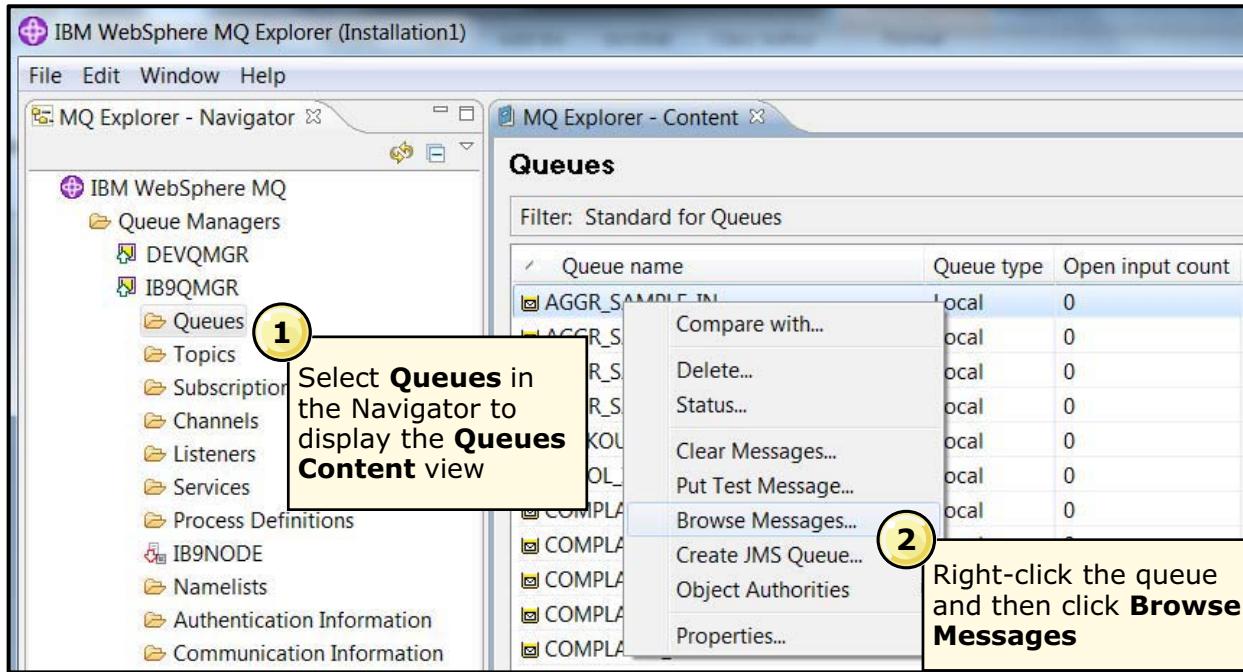
MQ documents the conditions under which an application can retrieve a sequence of messages from a queue in the same order that another application puts them. The conditions assume that the application that retrieves the sequence of messages is not using selection criteria.

The first set of conditions in the figure assumes that the application that puts the messages and the application that gets the messages are both connected to the same queue manager; no remote queuing is involved. These conditions are described more fully in the *IBM MQ Application Programming Guide*.

The second set of conditions that is shown in the figure is the case that each application is connected to a different queue manager and so remote queuing is involved. These conditions are described more fully in *IBM MQ Intercommunication*.

If the conditions are not met, applications must either include sequencing information in the application data of a message or establish a means of acknowledging receipt of a message before the next one is sent.

## Browsing messages in IBM MQ Explorer (1 of 2)



© Copyright IBM Corporation 2014

Figure 5-7. Browsing messages in IBM MQ Explorer (1 of 2)

WM2091.0

### Notes:

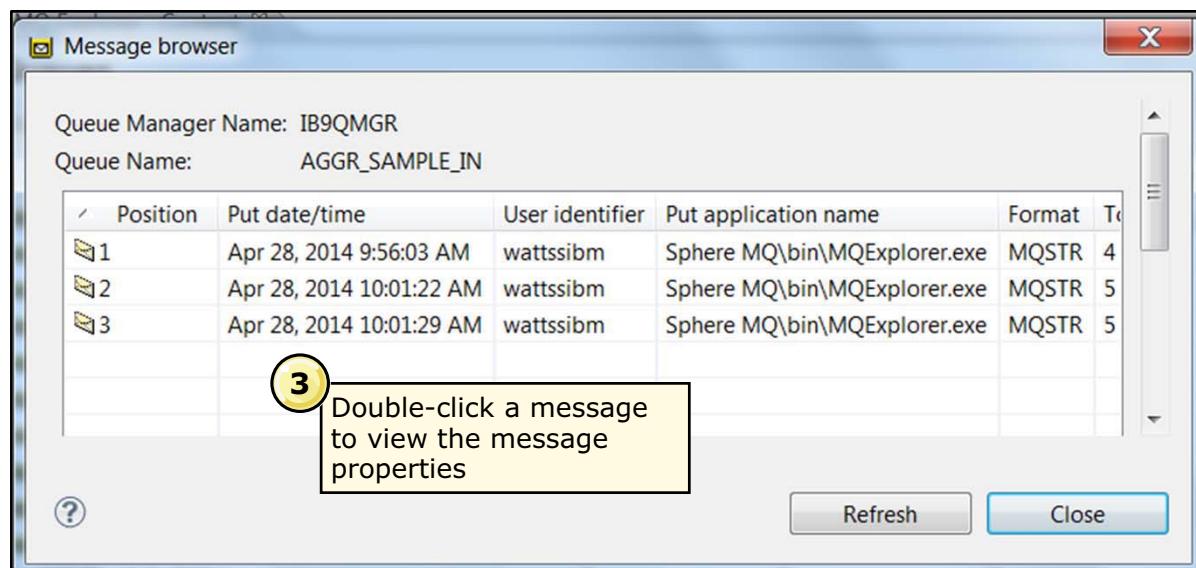
You can browse the queue at any time to see the messages that are on a queue.

This figure and the next show the steps for browsing the contents of a queue in MQ Explorer.

1. Select **Queues** in the Navigator to display the **Queues Content** view.
2. Right-click the queue and then click **Browse Messages**.



## Browsing messages in IBM MQ Explorer (2 of 2)



© Copyright IBM Corporation 2014

Figure 5-8. Browsing messages in IBM MQ Explorer (2 of 2)

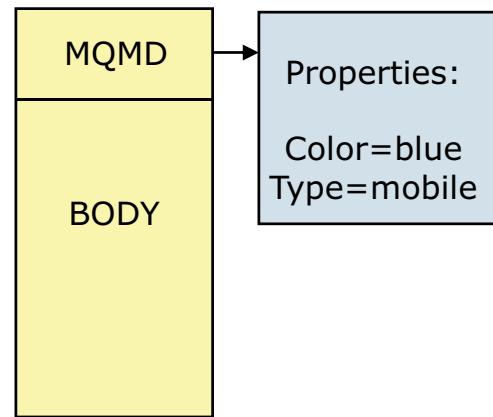
WM2091.0

### Notes:

3. The **Message browser** provides a summary of each message that is on the queue.  
For more information about a specific message, double-click the message in the **Message browser**.

## Message properties

- Attributes of IBM MQ messages that are associated with the message but not part of the body
  - General
  - Report
  - Context
  - Identifiers
  - Segmentation
  - Named properties
  - MQRFH2 properties
  - Data
  - Dead-letter header
- Properties can be integers, strings, Booleans, and so on
- Allows explicit statement of relationships between messages
- Can be viewed in IBM MQ Explorer



© Copyright IBM Corporation 2014

Figure 5-9. Message properties

WM2091.0

### Notes:

Message selectors use message properties to filter publications to topics or to selectively get messages from queues. Message properties can be used to include business data or state information without storing it in the message data.

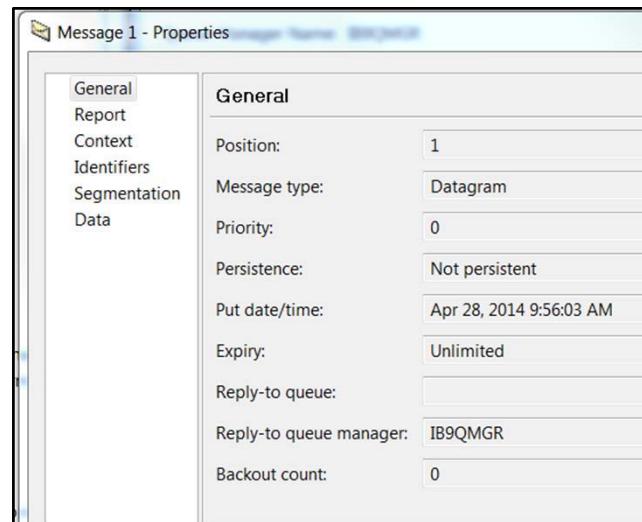
Message properties are arbitrary name-value pairs that can be assigned to a message. Properties are assigned by using two MQI verbs: MQSETPROP to set the properties and MQINQPROP to inquire on the message property value.

Message properties are primarily used for establishing explicit relationships between messages over and above the message ID and correlation ID. For example, message properties can specify that *message X* is a reply to **message Y**.

You can view the message properties in MQ Explorer or by using the WebSphere MQ SupportPac RFHUtil. In MQ Explorer, message properties group into the nine categories that are listed in the figure.

## General message properties

- **Position:** Position in the queue
- **Message type:** Type of the message
- **Priority:** Priority of the message
- **Persistence:** Indicates whether the message is persistent or not persistent
- **Put date/time:** Date and time the message was put on the queue
- **Expiry:** Time after which the message is eligible to be discarded
- **Reply-to queue:** Name of message queue to which the application that gets the message should send reply messages
- **Reply-to queue manager:** Name of queue manager on which the reply-to queue is defined
- **Backout count:** Number of times the message was returned by the MQGET call as part of a unit of work, and then backed out.



© Copyright IBM Corporation 2014

Figure 5-10. General message properties

WM2091.0

### Notes:

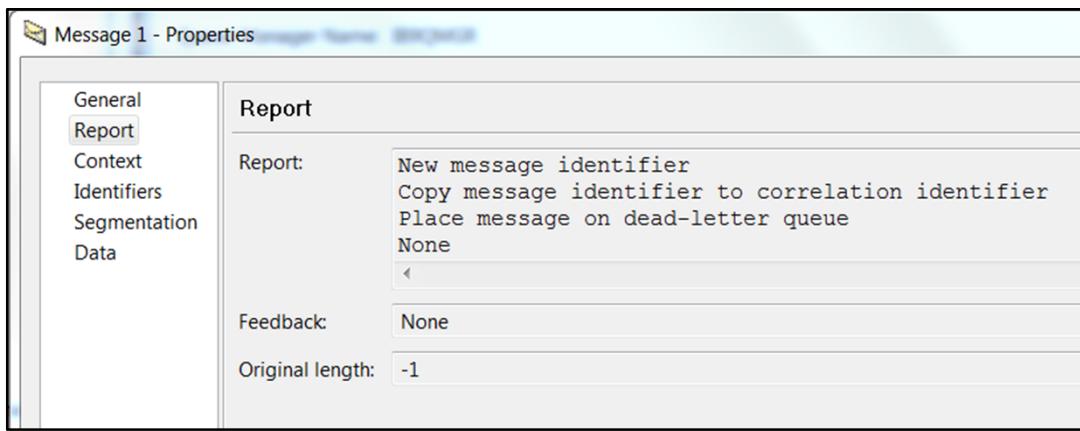
This figure provides an overview of the **General** message properties in the MQ Explorer message browser.

Some message properties, such as message type, priority, and persistence are set by the application in the MQMD.



## Report message properties

- **Report:** Sender application specifies whether report messages are required, whether the application data is to be included in the report messages, and how the message and correlation identifiers in the report or reply message set
- **Feedback:** Nature of the report
- **Original length:** Length of the original message to which the report relates



© Copyright IBM Corporation 2014

Figure 5-11. Report message properties

WM2091.0

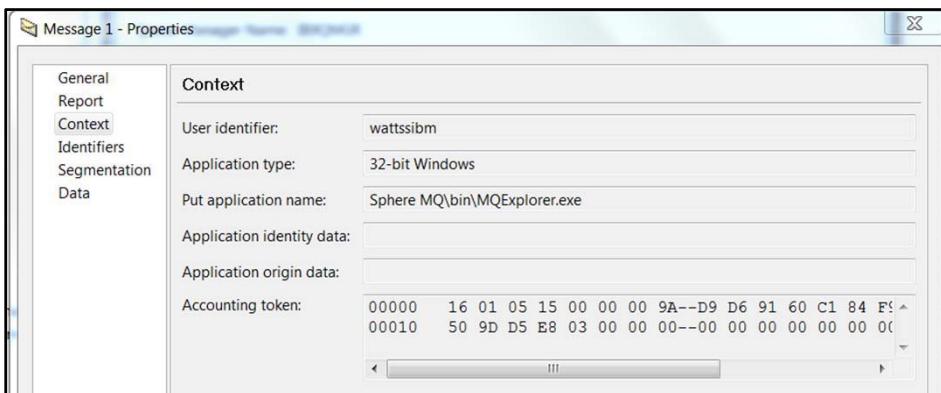
### Notes:

A report is a message about another message. It is used to inform the application about expected or unexpected events that relate to the original message.

The **Report** page displays the attributes that are related to report messages. This figure provides an overview of the **Report** message properties in the MQ Explorer message browser.

## Context message properties

- **User identifier:** User identifier of the application that originated the message
- **Application type:** Type of application that put the message
- **Put application name:** Name of the application that put the message
- **Application identity data:** Information that is defined by the application to provide information about the message or its originator
- **Application origin data:** Information that is defined by the application suite to provide additional information about the origin of the message
- **Accounting token:** Information that allows the application to appropriately charge work that is done as a result of the message



© Copyright IBM Corporation 2014

Figure 5-12. Context message properties

WM2091.0

### Notes:

The **Context** page displays information from the sender application about the message.

This figure provides an overview of the message properties that are contained on the **Context** page.

## Identifiers message properties

- **Message identifier:** Distinguishes one message from another
- **Message identifier bytes:** Message identifier in byte form
- **Correlation identifier:** Identifier that the application can use to relate one message to another, or to relate the message to other work that the application is performing
- **Correlation identifier bytes:** Correlation identifier in byte form
- **Group identifier:** Identifies the particular message group or logical message to which the physical message belongs
- **Group identifier bytes:** Group identifier in byte form

© Copyright IBM Corporation 2014

Figure 5-13. Identifiers message properties

WM2091.0

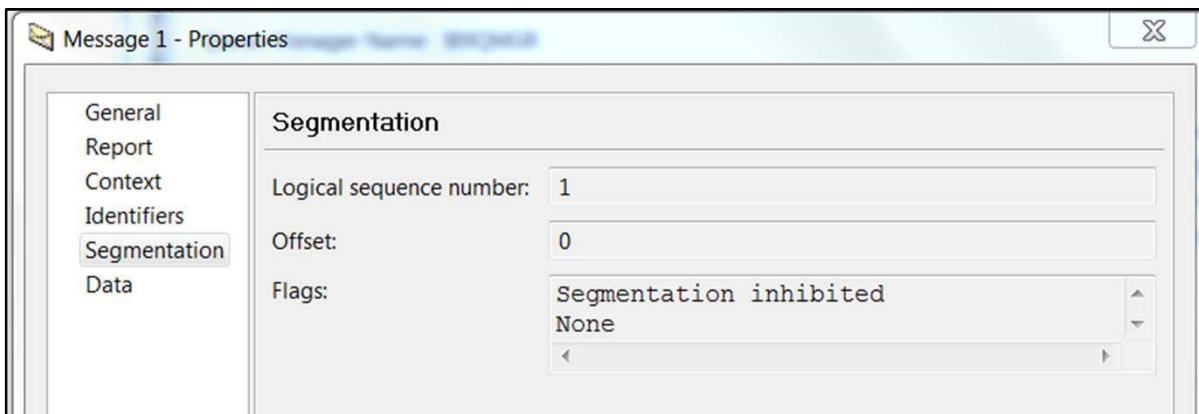
### Notes:

The **Identifiers** page displays identification information that is associated with the message.

This figure provides an overview of the message properties that are contained on the **Identifiers** page.

## Segmentation message properties

- **Logical sequence number:** Sequence number of the logical message within the group
- **Offset:** Offset of data in the physical message from the start of the logical message
- **Flags:** Message flags that specify attributes of the message, or control its processing



© Copyright IBM Corporation 2014

Figure 5-14. Segmentation message properties

WM2091.0

### Notes:

The **Segmentation** page displays the attributes that are related to segmenting large messages.

This figure provides an overview of the message properties that are contained on the **Segmentation** page.

## Data message properties

- **Data length:** Length of the original message.
- **Format:** Name that the sender of the message that is used to indicate to the receiver the nature of the data in the message.
- **Coded character set identifier:** Identifier of the character data in the application message data.
- **Encoding:** Numeric encoding of numeric data in the message. This value does not apply to numeric data in the MQMD structure itself.
- **Message data:** Message data in human readable ASCII text.
- **Message data bytes:** Message data in hexadecimal format.

© Copyright IBM Corporation 2014

Figure 5-15. Data message properties

WM2091.0

### Notes:

The **Data** page displays the message data itself and information about the data format.

This figure provides an overview of the message properties on the **Data** page.



## Message property preferences in IBM MQ Explorer

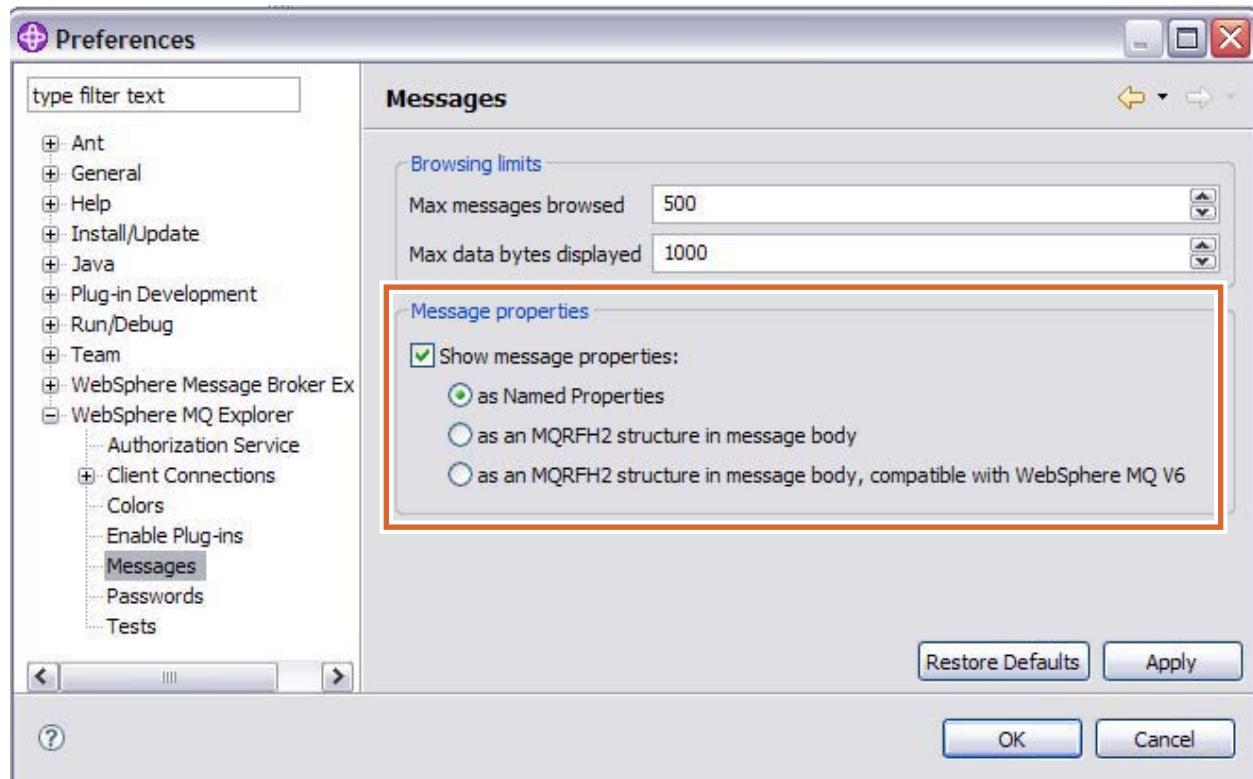


Figure 5-16. Message property preferences in IBM MQ Explorer

WM2091.0

### Notes:

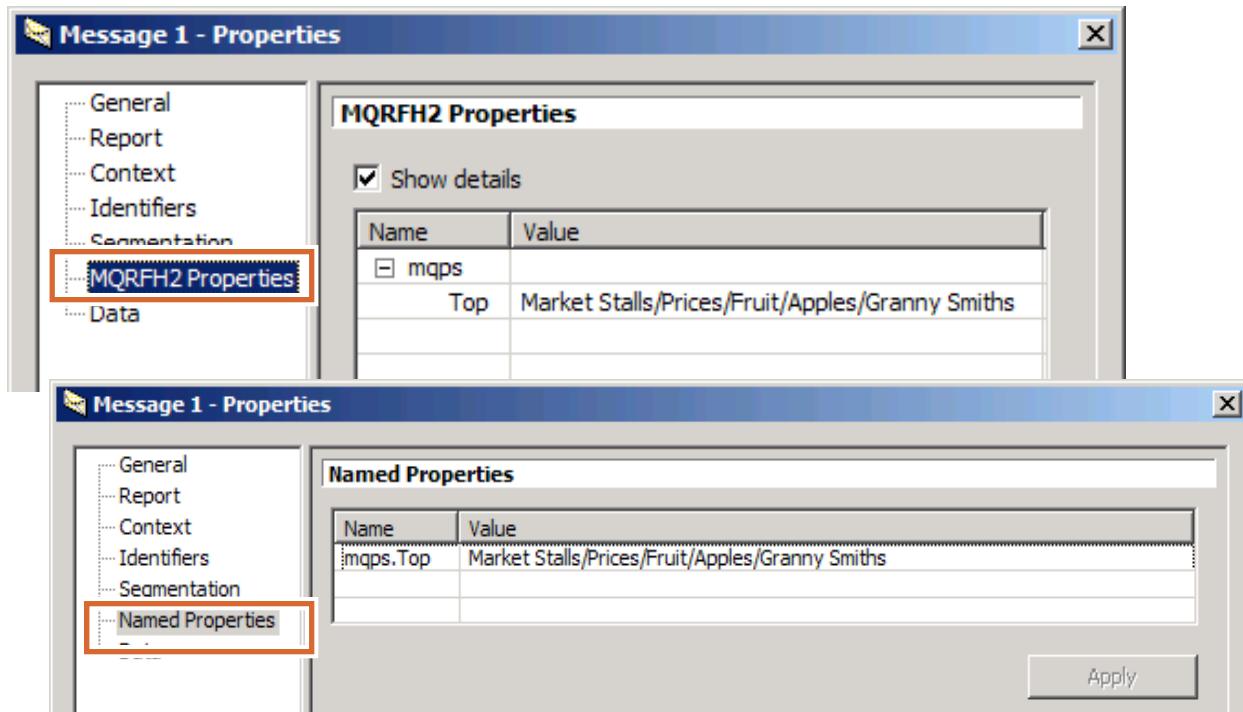
MQ Explorer can be configured to read properties in messages on the queue dependent on the message properties.

The way message properties are shown in MQ depends on the configuration of MQ Explorer and the definition of the queue.

Depending on the **Preference** property setting, you see properties either as **Named Properties** or **MQRFH2 Properties** when looking at the properties of a browsed message.



## Message property preferences: Examples



© Copyright IBM Corporation 2014

Figure 5-17. Message property preferences: Examples

WM2091.0

### Notes:

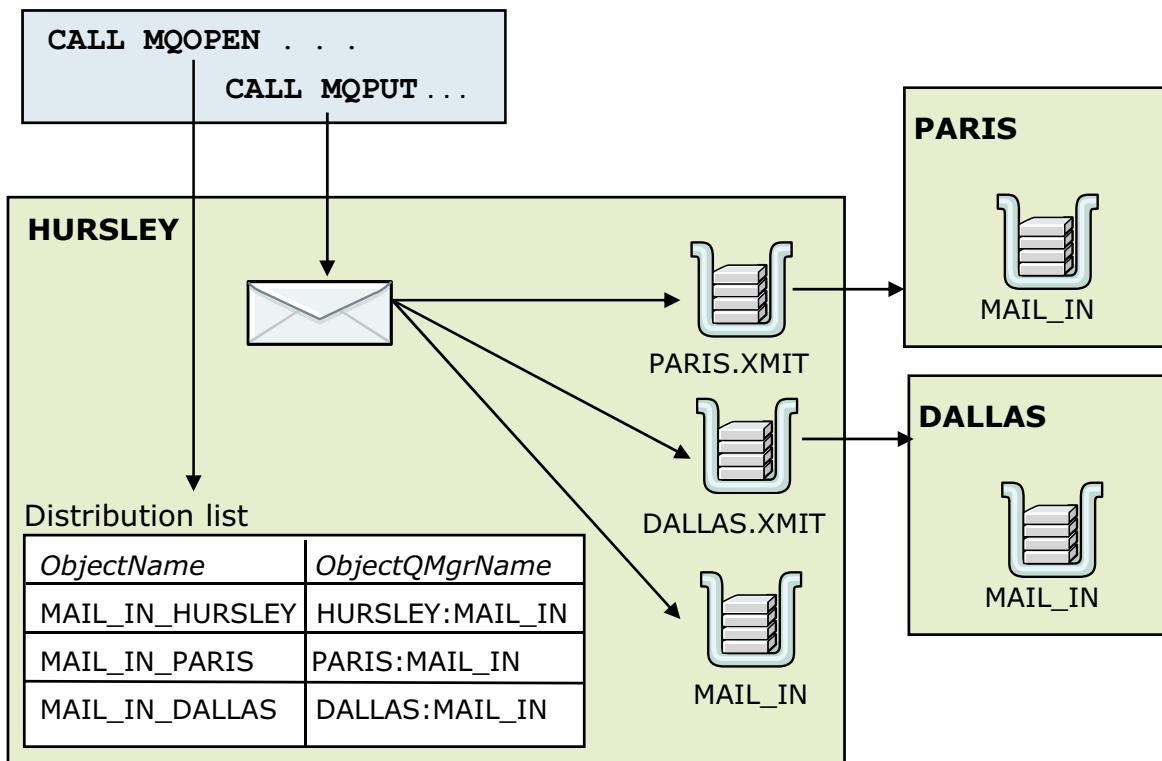
If you selected one of the MQRFH2 message property settings in the **MQ Preferences**, you can display the MQRH2 properties when you browse the message.

This figure shows the two different ways that you can display message properties.

The table contains two columns:

- **Name** is the name of the message property.
- **Value** is the actual value of the named property.

## Distribution lists



© Copyright IBM Corporation 2014

Figure 5-18. Distribution lists

WM2091.0

### Notes:

Distribution lists allow you to put a message to multiple destinations in a single MQPUT or MQPUT1 call. A single MQOPEN call can open multiple queues and a single MQPUT call can then put a message to each of those queues.

A distribution list might contain the name of an alias queue or the name of a local definition of a remote queue. In the example of a distribution list that is shown in the figure, the following are assumed.

- MAIL\_IN\_HURSLEY is an alias queue that resolves to the local queue MAIL\_IN on queue manager HURSLEY.
- MAIL\_IN\_PARIS is a remote queue definition that resolves to the queue MAIL\_IN on queue manager PARIS by using transmission queue PARIS.XMIT.
- MAIL\_IN\_DALLAS is a remote queue definition that resolves to the queue MAIL\_IN on queue manager DALLAS by using transmission queue DALLAS.XMIT.

After a distribution list is opened, the application can call MQPUT to put a message on the queues in the list. As a response to the call, the queue manager puts **one** copy of the message on each

local queue, including the transmission queues. Thus, only one copy of the message is put on the transmission queue DALLAS even though the message is ultimately destined for two queues.

A distribution header and a transmission queue header prefix the application data on the transmission queue. The message that is transmitted between HURSLEY and DALLAS effectively contains a distribution list for the two destinations.

An important property of the implementation is that a message destined for multiple queues is only replicated at the last possible point at each stage of its journey. In this way, network traffic is minimized.

## Asynchronous message reception: Callback

- Allows identification of a function to be started when a message arrives on a queue
  - Not the same as triggering or MQGET with wait
  - MQGET is not used at all
- MQCB call to register and unregister; suspend and resume consumers
- MQCTL call to start and stop; suspend and resume

© Copyright IBM Corporation 2014

Figure 5-19. Asynchronous message reception: Callback

WM2091.0

### Notes:

Asynchronous message reception, or callback, allows messages to be used from multiple queues or topics. This option facilitates the implementation of a message-driven processing model. The alternative would be to use message triggering or message dispatchers that are based on MQGET with wait or to have programs that are unaware of the message size that is received.

Callback can simplify the design and implementation of new applications. Message processing functions are registered with the MQCB MQI call. The queue manager starts the function when there are messages available that match the selection criteria. The control function (MQCTL) indicates to the queue manager when to start dispatching the callback functions to pass messages to them. The callback functions receive and process messages in asynchronous mode.

## Testing IBM MQ configuration

- IBM MQ sample programs
  - Available as C source, COBOL source, and C runtime
  - Runtime files on Linux: `/opt/mqm/samp/bin`
  - Runtime files on Windows:  
`C:\Program Files\IBM\WebSphere MQ\Tools\c\Samples\Bin64`
- MQ Explorer
  - Put test messages
  - Browse messages
- WebSphere MQ SupportPac IH03 RFHutil
  - Put, get, and browse test messages
  - View message content in multiple formats
  - Modify message headers
  - Test functions for JMS and publish/subscribe

© Copyright IBM Corporation 2014

Figure 5-20. Testing IBM MQ configuration

WM2091.0

### Notes:

MQ provides many ways to test your MQ configuration.

MQ sample programs are available as C source, COBOL source, and C run time. They can be run from any location on Windows. On Linux and UNIX, you must run the sample program from its home directory or set the PATH to the sample program directory.

MQ Explorer also has some sample programs for putting and browsing messages.

Many MQ administrators use the WebSphere SupportPac IH03 RFHUtil for testing queues. It allows test messages to be captured and stored in files. Messages can also be read and displayed in various formats. At the time of publication of this course, WebSphere SupportPac IH03 was not yet updated to support MQ V8.

## Testing with IBM MQ sample programs

- **amqspput QName QMgrName**  
Reads text from the standard input device and put messages
- **amqsget QName QMgrName**  
Gets messages and write to the standard output device
- **amqsbcg QName QMgrName**  
Browses messages showing both application data and message descriptor
- **amqsgbr QName QMgrName**  
Browse messages showing application data only
- **amqsreq QName QMgrName ReplyToQName**  
Reads lines of text from the standard input device, converts them to request messages, and MQPUTs the messages on the named queue

**Note:** The queue manager name is optional when referencing the default queue manager

© Copyright IBM Corporation 2014

Figure 5-21. Testing with IBM MQ sample programs

WM2091.0

### Notes:

This figure describes the sample programs that are provided with MQ.

The Put sample program, **amqspput**, connects to the queue manager and opens the queue. It reads lines of text from the standard input device, generates a message from each line of text, and puts the messages on the named queue. After reading a null line or the EOF character from the standard input device, it closes the queue, and disconnects from the queue manager.

The Get sample program, **amqsget**, connects to the queue manager, opens the queue for input, and gets all the messages from the queue. It writes the text within the message to the standard output device, waits 15 seconds (60 seconds if there is no message at the start) in case any more messages are put on the queue. The program then closes the queue and disconnects from the queue manager.

There are two browser sample programs:

- The **amqsbcg** program connects to the queue manager and opens the queue for browsing. It browses all the messages on the queue and writes their contents, in both hexadecimal and character format, to the standard output device. It also shows, in a readable format, the fields in

the message descriptor for each message. It then closes the queue and disconnects from the queue manager.

- The **amqsbr** program browses messages on a queue by using the MQGET call.

The Request sample program, **amqsreq**, demonstrates client/server processing. It puts a series of request messages on the target server queue by using the MQPUT call. These messages specify the local queue, SYSTEM.SAMPLE.REPLY as the reply-to queue, which can be a local or remote queue.

## Sample program example

```
C:\Users\WMQ_ADMIN>amqspput QL.TEST QMGR1
Sample AMQSPUTO start
Target queue is QL.TEST
This is my test message
This is another test message
[Press Enter]                                ← A blank line
                                                indicates the end of
                                                message input
Sample AMQSPUTO end

C:\User\WMQ_ADMIN>amqsget QL.TEST QMGR1
Sample AMQSGETO start
message <This is my test message>
message <This is another test message>
C:\Users\WMQ_ADMIN>
```

© Copyright IBM Corporation 2014

Figure 5-22. Sample program example

WM2091.0

### Notes:

This figure shows an example of the **amqspput** and **amqsget** sample programs.

The **amqspput** and **amqsget** sample programs require two parameters: the queue name and the queue manager name.

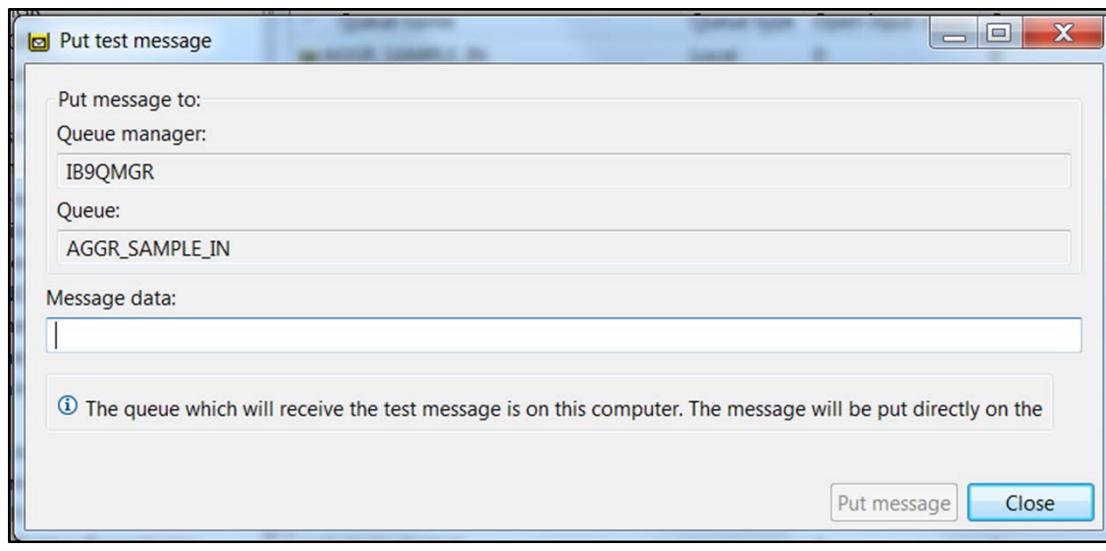
To use the **amqspput** sample program, enter one or more test messages after the program starts. When you are finished entering test messages, press Enter on a blank line to end the sample program.

As shown in the example, the **amqsget** program gets all the messages that are in the queue, which empties the queue. When you run the **amqsget** sample program, the program waits for some time in case any more messages are written to the queue; it might take a minute for the program to complete.



## Testing with IBM MQ Explorer

- Browse messages on queue
- Clear messages on queue
- Put messages on queue
- View message contents and properties



© Copyright IBM Corporation 2014

Figure 5-23. Testing with IBM MQ Explorer

WM2091.0

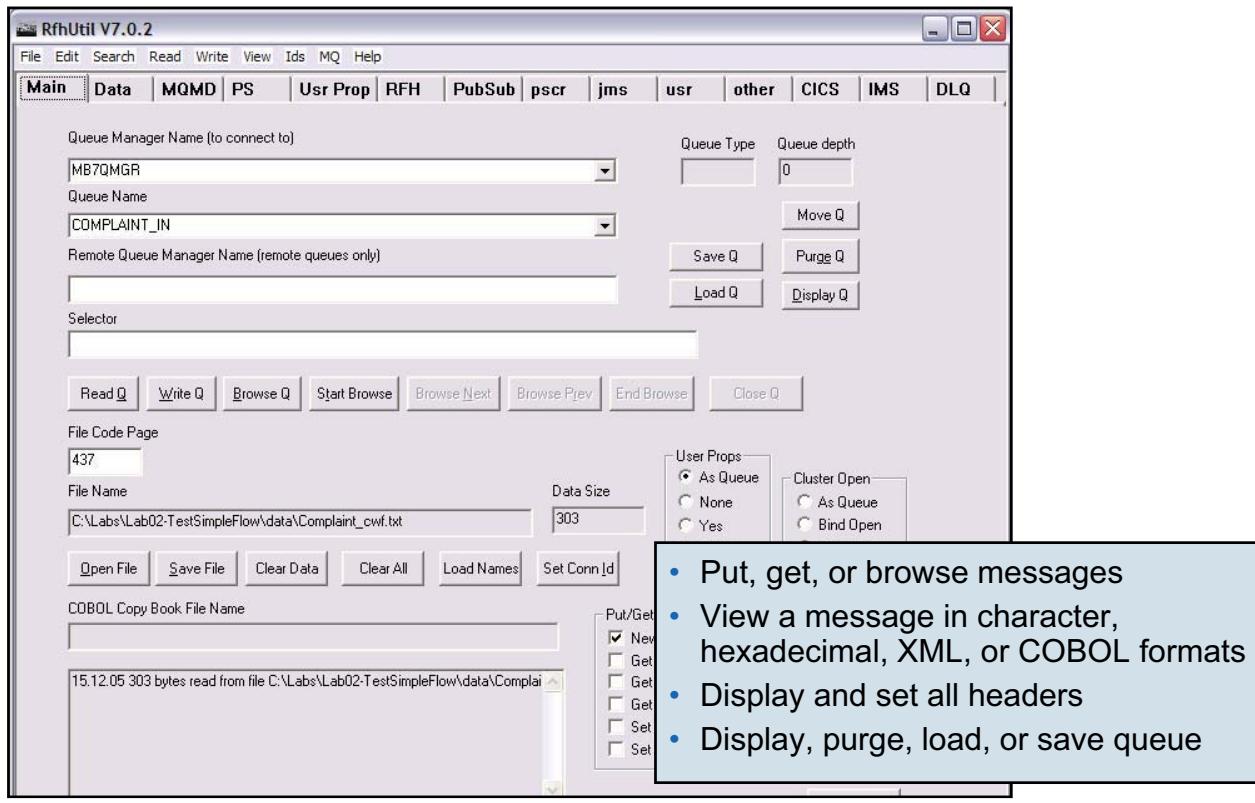
### Notes:

With MQ Explorer, you can put messages, browse messages, clear (get) messages, and view message contents and properties.

To use the MQ Explorer test options, right-click the queue name in the **Queue** content view and then click **Test**.



## Testing with RFHUtil (SupportPac IH03)



© Copyright IBM Corporation 2014

Figure 5-24. Testing with RFHUtil (SupportPac IH03)

WM2091.0

### Notes:

The RFHUtil utility program is a WebSphere MQ SupportPac. It reads data from files, queues, or both. It also writes data to files, queues, or both, and displays data in various formats.

The user data portion of the message can be displayed in various formats, but it cannot be changed. Another program must be used to create or change the user data.

With RFHUtil, you can add rules and formatting headers (RFH) to messages or files. It writes and formats these headers when they are found in messages or files that it reads. The headers can include publish/subscribe commands.

You can download RFHUtil from the WebSphere MQ SupportPac website at:

<http://www.ibm.com/support>



## Unit summary

Having completed this unit, you should be able to:

- Recognize MQI calls in a program
- Explain the purpose of the fields in the IBM MQ message descriptor
- Use IBM MQ sample programs to put, get, and browse messages
- Use IBM MQ Explorer to put, get, and browse messages

© Copyright IBM Corporation 2014

Figure 5-25. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. True or False: The correlation identifier is normally used to provide an application with a means of matching a reply or report message with the original message.
  
2. An application can retrieve a message from a queue that is based on:
  - a. Message ID
  - b. Correlation ID
  - c. Next available message that is based on the MsgDeliverySequence value of the queue
  - d. An SQL-92 selector
  - e. All of the above

© Copyright IBM Corporation 2014

Figure 5-26. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
  
- 2.



## Checkpoint answers

1. True.
2. e.

© Copyright IBM Corporation 2014

Figure 5-27. Checkpoint answers

WM2091.0

### Notes:

## Exercise 3



Using IBM MQ sample programs to test the configuration

© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.O

Figure 5-28. Exercise 3

WM2091.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Use IBM MQ sample programs to put messages onto a queue, browse messages on a queue, and get messages from a queue
- Use IBM MQ Explorer and IBM MQ commands to display queue contents

© Copyright IBM Corporation 2014

Figure 5-29. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for detailed instructions.

# Unit 6. Implementing trigger messages and monitors

## What this unit is about

This unit describes how queues can be configured to trigger an application when certain conditions are true.

## What you should be able to do

After completing this unit, you should be able to:

- Configure IBM MQ to enable a trigger monitor
- Run the trigger monitors on UNIX and Windows systems
- Determine the cause of a triggering failure

## How you will check your progress

- Checkpoint questions
- Hands-on lab exercise

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Configure IBM MQ to enable a trigger monitor
- Run the trigger monitors on UNIX and Windows systems
- Determine the cause of a triggering failure

© Copyright IBM Corporation 2014

Figure 6-1. Unit objectives

WM2091.0

### Notes:

One of the most common functions that is used in MQ is triggering. Triggering provides message-driven, or event-driven processing. In this unit, the requirements for triggering are described so that you can gain a clear understanding of the advantages, and some of the more common problems that you might encounter with triggering.

## What is triggering?

- Queue manager defines certain conditions as constituting trigger events
- If triggering is enabled for a queue and a trigger event occurs, the queue manager sends a trigger message to a queue called an initiation queue
- Trigger messages are not persistent

© Copyright IBM Corporation 2014

Figure 6-2. What is triggering?

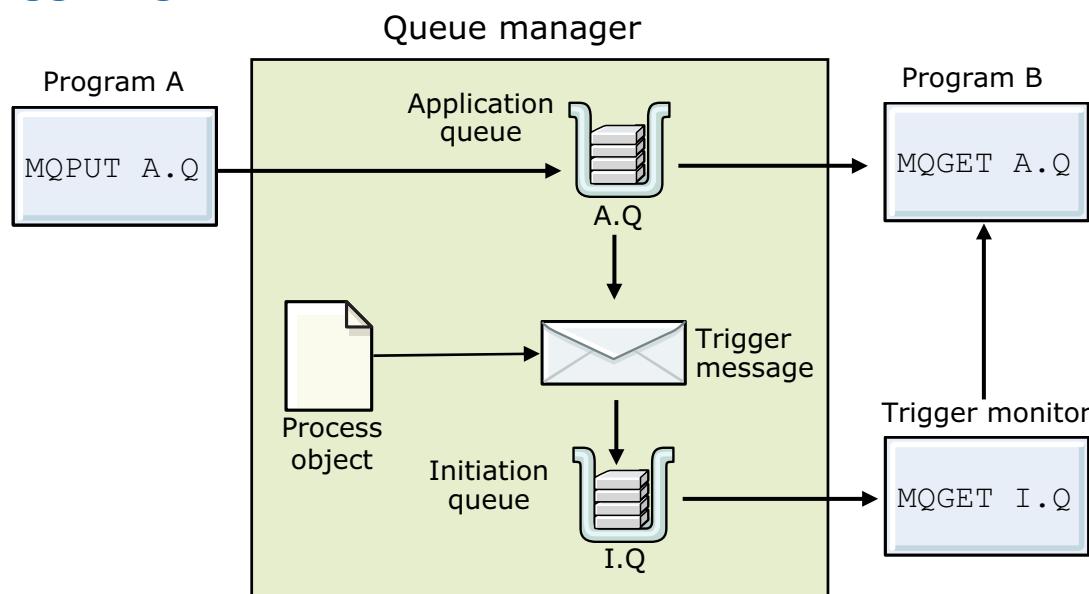
WM2091.0

### Notes:

The queue manager defines certain conditions as constituting trigger events. If triggering is enabled for a queue and a trigger event occurs, the queue manager sends a trigger message to a queue called an *initiation queue*. The presence of the trigger message on the initiation queue indicates that a trigger event occurred.

Trigger messages that the queue manager generates are not persistent. Non-persistent messages reduce logging which improves performance, and minimizes duplicates during restart, so improving restart time.

## Triggering



- Triggering allows
  - Instantiation as required
  - Conservation of system resources
  - Automation of flow

© Copyright IBM Corporation 2014

Figure 6-3. Triggering

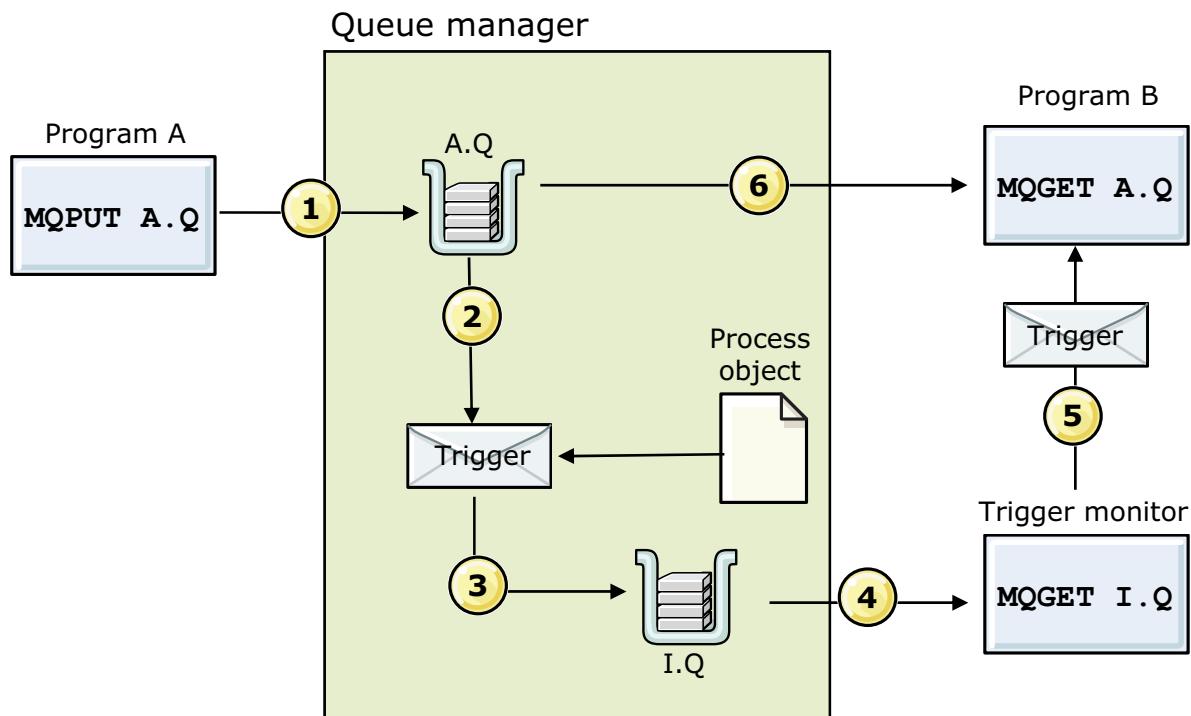
WM2091.0

### Notes:

MQ triggering provides an enhancement to the implementation of time-independent processing. The arrival of a message on an application queue might indicate that it is an appropriate time for another application to start to process the messages on the application queue. When the queue manager detects the right conditions, the triggering facility starts the application to service the application queue.

In the example in the figure, Program A puts a message to an application queue. When triggering is enabled, a special trigger message is put on an initiation queue. The trigger monitor “watches” the queue. When a message is put on the initiation queue, the trigger monitor sends a message to Program B to start.

## Components of triggering



© Copyright IBM Corporation 2014

Figure 6-4. Components of triggering

WM2091.0

### Notes:

The figure shows the triggering sequence actions.

1. Program A puts a message on an application queue that is enabled for triggering.
2. If the conditions for triggering are met, a *trigger event* occurs, and the queue manager examines the process object that the application queue references. The process object identifies the application to start, Program B.
3. The queue manager creates a *trigger message* whose fields contain information that is copied from certain attributes of the process object and the application queue. The queue manager puts the trigger message on an *initiation queue*.
4. A long running program that is called a *trigger monitor* gets the trigger message and examines its contents.
5. The trigger monitor starts Program B, passing the entire trigger message as a parameter.
6. Program B opens the application queue and gets messages from it.

## Queue attributes that control triggering

```
DEFINE QLOCAL(MY_SERVER) TRIGGER +
PROCESS(ECHO) +
INITQ(SYSTEM.DEFAULT.INITIATION.QUEUE)
```

- **TRIGGER or NOTRIGGER**

Specifies whether trigger messages are written to the initiation queue

- **TRIGMPRI (*integer*)**

Priority of messages that queue manager uses when deciding whether to generate a trigger event

- **TRIGTYPE (FIRST) or TRIGTYPE (DEPTH) or TRIGTYPE (EVERY) or TRIGTYPE (NONE)**

Specifies whether and under what conditions a trigger message is written to the initiation queue

- **TRIGDEPTH (*integer*)**

Number of messages that are required to generate a trigger event

- **TRIGDATA (*string*)**

Data that is inserted in the trigger message

© Copyright IBM Corporation 2014

Figure 6-5. Queue attributes that control triggering

WM2091.0

### Notes:

To define an application queue for triggering, the MQSC command **DEFINE QLOCAL** must contain the following parameters:

- **PROCESS(*string*)**: The name of the process object that identifies the application that can service the application queue.
- **INITQ(*string*)**: The name of the initiation queue.

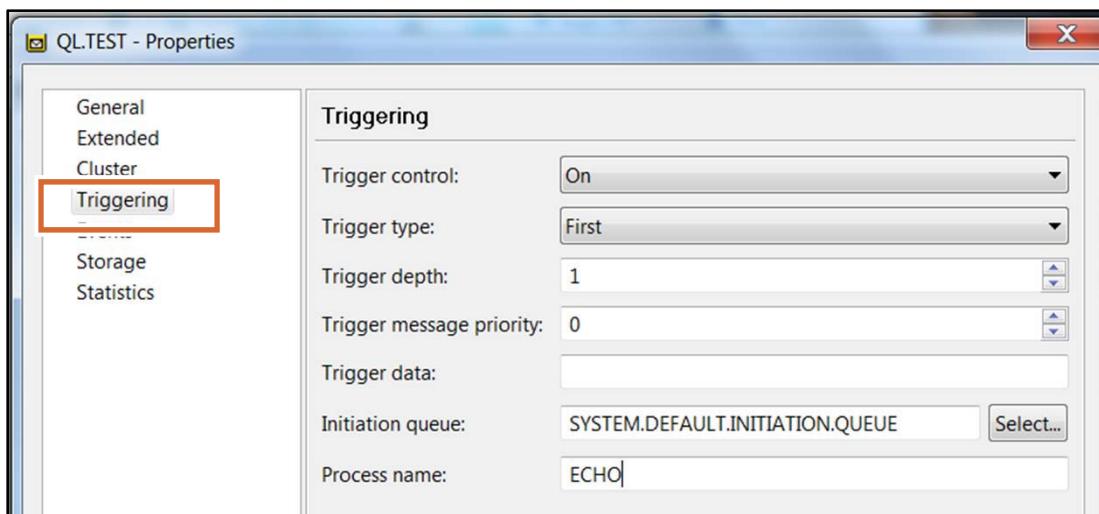
The figure lists the parameters of the MQSC command **DEFINE QLOCAL** that control triggering.

When **TRIGTYPE** is set to **FIRST**, a trigger event occurs when the queue changes from empty to having one message.

When **TRIGTYPE** is set to **DEPTH**, a trigger event occurs when the number of messages on the queue reaches the **TRIGDEPTH** parameter the value. When triggering by depth, the queue manager disables triggering by setting the application queue to **NOTRIGGER** after it creates a trigger message. It is the responsibility of the application to reenable triggering by using the MQSET call.

When **TRIGTYPE** is set to **EVERY**, a trigger event occurs for every message that is put on the application queue.

## Defining triggering properties in IBM MQ Explorer



1. Right-click the queue in the **Queue** Contents view and then click **Properties**.
2. Click **Triggering**.
3. Enter the triggering properties for the queue and then click **Apply**.

© Copyright IBM Corporation 2014

Figure 6-6. Defining triggering properties in IBM MQ Explorer

WM2091.0

### Notes:

You can also enable triggering and define the trigger properties for a queue in MQ Explorer, as shown in the figure.

## Process attributes

```
DEFINE PROCESS(ECHO) +
APPLICID('/opt/mqm/samp/bin/amqsech')
```

<b>APPLICID (<i>string</i>)</b>	Name of the application to start
<b>APPLTYPE (WINDOWS)</b>	Application type to start
<b>APPLTYPE (UNIX)</b>	
<b>ENVRDATA (<i>string</i>)</b>	Environment information that is passed to the application by the trigger monitor
<b>USERDATA (<i>string</i>)</b>	User information that is passed as part of a parameter list by the trigger monitor

© Copyright IBM Corporation 2014

Figure 6-7. Process attributes

WM2091.0

### Notes:

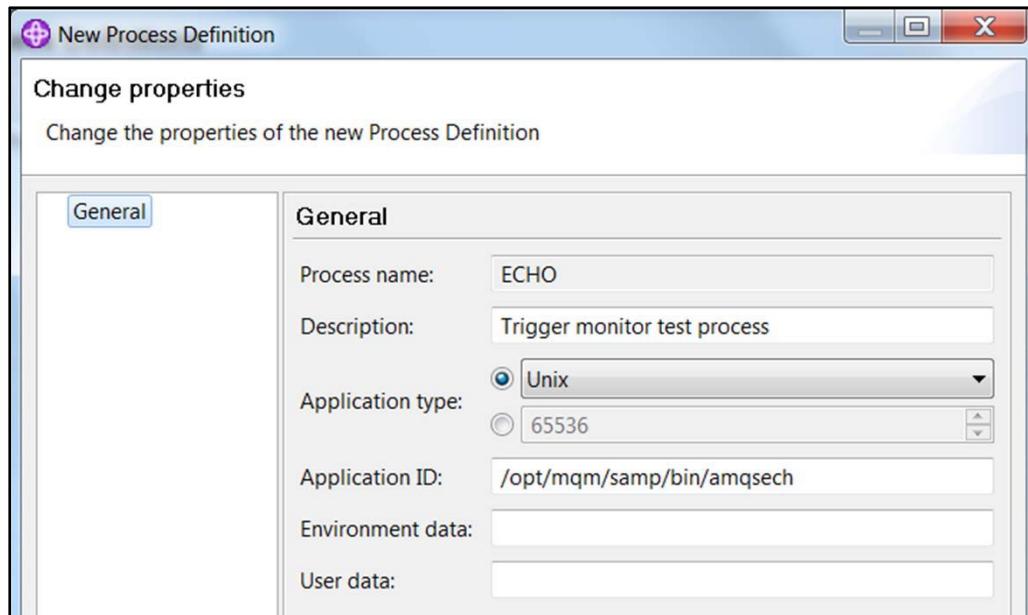
The trigger definition requires the name of the process object that identifies the application that can service the application queue. This figure shows the MQSC command and options that defines a process object.

A process and a queue are allowed to have the same name. The name of an object must be unique within an object type.

The **APPLICID** value depends on the **APPLTYPE** parameter. For example, if the application type is UNIX, the **APPLICID** is a fully qualified path of a runnable object, as shown in the example in the figure.

If you want to trigger the start of a channel, it is not necessary to define a process definition object. The transmission queue definition can determine the channel to trigger.

## Defining a process in IBM MQ Explorer



1. Right-click **Process Definitions** under queue manager and then click **New > Process Definition**.
2. Enter a process name and then click **Next**.
3. Enter process definition properties and then click **Finish**.

© Copyright IBM Corporation 2014

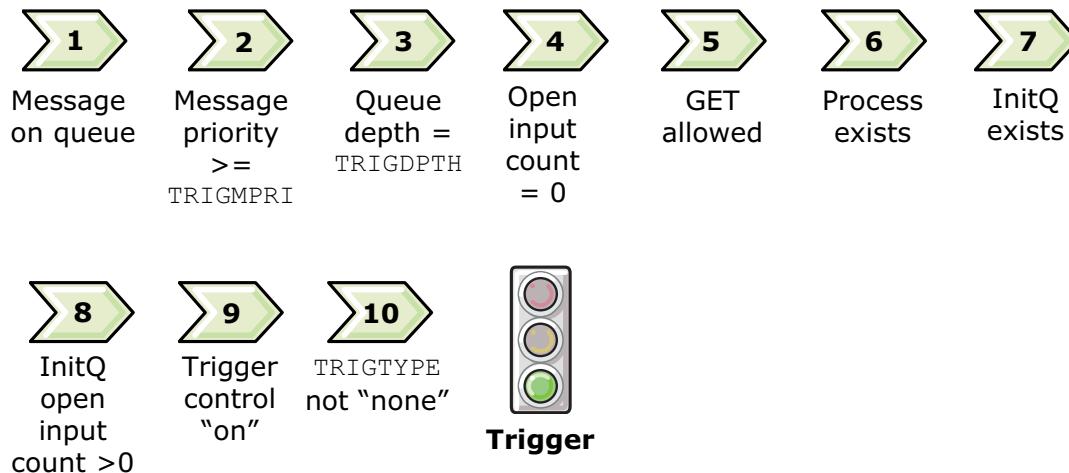
Figure 6-8. Defining a process in IBM MQ Explorer

WM2091.0

### Notes:

As shown in the figure, you can also define a process in MQ Explorer.

## Triggering conditions



© Copyright IBM Corporation 2014

Figure 6-9. Triggering conditions

WM2091.0

### Notes:

This figure shows all of the conditions that must be satisfied for a trigger event to occur.

1. A message is put on a queue.
2. The priority of the message is greater than or equal to the value of the trigger message priority attribute of the queue.
3. The number of messages on the queue with a priority greater than or equal the trigger message priority attribute was previously:
  - No messages, for trigger type FIRST
  - Any number of messages, for trigger type EVERY
  - Trigger depth minus one message, for trigger type DEPTH
4. For triggering of type FIRST or DEPTH, no program has the application queue open for removing messages (the Open Input Count of the local queue attribute is zero).
5. The queue is enabled for GET requests.
6. The process object that the queue **Process Name** attribute identifies exists.

7. The initiation queue that is identified by the **Initiation Queue** queue attribute exists, and is enabled for PUT and GET requests.
8. A trigger monitor currently has the initiation queue open for removing messages (the OpenInputCount of the local queue attribute is greater than zero).
9. Trigger control for the application queue is enabled; that is, the attribute for trigger control is set to ON.
10. The value of the **Trigger Type** attribute of the application queue is not NONE.

If a trigger event does not occur when it is expected, check this list.

A full statement of all of these conditions can be found in the *IBM MQ Application Programming Guide*.

## Fields in the trigger message

- Copied from the corresponding attributes of the application queue
  - QName
  - ProcessName
  - TriggerData
- Copied from the corresponding attributes of the process object
  - ApplType
  - ApplId
  - EnvData
  - UserData
- Queue manager name

© Copyright IBM Corporation 2014

Figure 6-10. Fields in the trigger message

WM2091.0

### Notes:

The *IBM MQ trigger message 2 character format* (MQTMC2) defines the trigger message structure. Some fields in the trigger message are derived from certain attributes of the application queue and some are derived from certain attributes of the process object that the definition of the queue references.

The figure lists the fields that are contained in the trigger message.

- **QName** is the name of the application queue.
- **ProcessName** is the name of the process object that identifies the application that can service the application queue.
- **TriggerData** is the trigger data that the started application uses. This field can be used to specify the name of the channel to trigger.
- **ApplType** is the application operating system, such as CICS, UNIX, and WINDOWS.
- **ApplId** is the application identifier of the application to start.
- **EnvData** is the environment data that the trigger monitor uses.

- **UserData** is the user data for use by the trigger monitor or by the started application.
- **QMgrName** is the name of the queue manager that owns the application queue.

## Running a trigger monitor

- To start a trigger monitor

```
runmqtrm -m QMgrName -q InitQName
```

- Default initiation queue is SYSTEM.DEFAULT.INITIATION.QUEUE
- Command string issued by trigger monitor to start the application

```
ApplId "MQTMC2" EnvData
```

Where:

- ApplId is the name of the program to run, as it is entered on the command
- MQTMC2 is a trigger message, which is enclosed in quotation marks, from the initiation queue and formatted in IBM MQ trigger message character format
- EnvrData is the environment data from the relevant process definition

© Copyright IBM Corporation 2014

Figure 6-11. Running a trigger monitor

WM2091.0

### Notes:

A trigger monitor is a long running program that gets a trigger message from an initiation queue and starts an application to service the application queue.

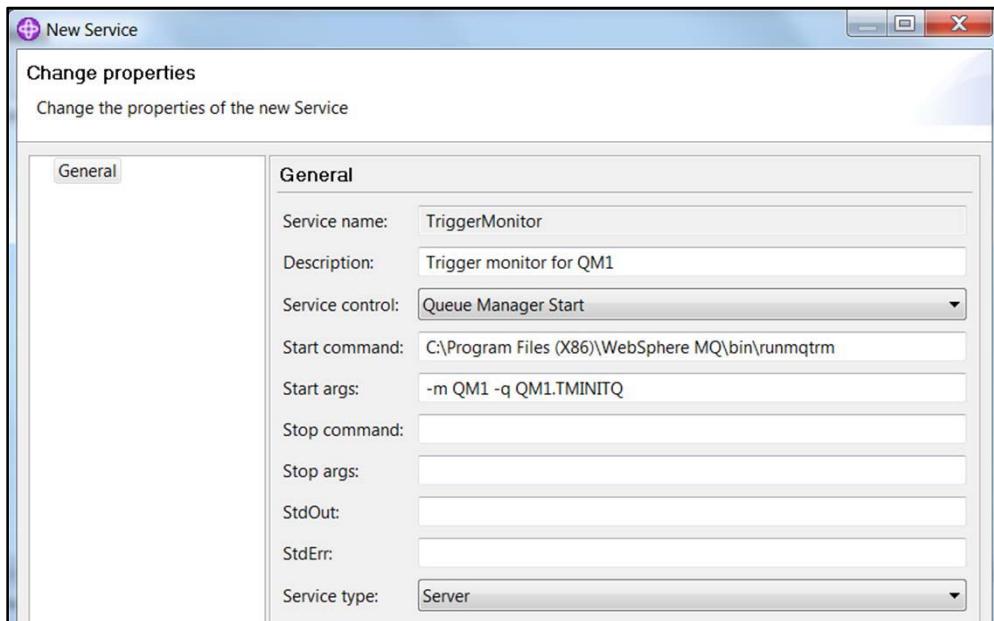
To start a trigger monitor on distributed systems, enter `runmqtrm` control command with the queue manager name, and the initiation queue name. If the name of an initiation queue is not explicitly specified on the `runmqtrm` command, the queue SYSTEM.DEFAULT.INITIATION.QUEUE is used by default.

The trigger monitor sends a command string, by using the MQTMC2 structure, as a parameter. This structure supplies the started application with the name of the queue that caused the trigger event and the name of the queue manager that owns it. The started application can connect to the queue manager and open the queue.

On UNIX and Windows, a separate control command, `runmqtmc`, is provided for MQ clients. It links with the MQ client libraries.



## Defining a trigger monitor in IBM MQ Explorer



1. Right-click **Services** under queue manager and then click **New > Service**.
2. Enter a service name and then click **Next**.
3. Enter the service properties and then click **Finish**.

© Copyright IBM Corporation 2014

Figure 6-12. Defining a trigger monitor in IBM MQ Explorer

WM2091.0

### Notes:

This figure shows the steps for defining a trigger monitor as a service in MQ Explorer.

## Trigger monitor service properties

- **Service control** specifies how the service starts and stops:
  - **Queue Manager** starts and stop the service when the queue manager starts and stops
  - **Queue Manager Start** starts the service when the queue manager starts but does not stop when the queue manager stops
  - **Manual** requires a manual start and stop
- **Start command** is the full path to the `runmqtrm` command
- **Start args** specifies the `runmqtrm` command arguments
  - If the queue manager is not the default queue manager, type `-m QMgrName`
  - To use an initiation queue other than SYSTEM.DEFAULT.INITIATION.QUEUE, type `-q InitQName`
- **Service type** selects the type of service to run
  - **Command** allows multiple instances of the service but cannot view the status of the services in IBM MQ Explorer
  - **Server** allows one instance of the service but can view the status of the service in IBM MQ Explorer

© Copyright IBM Corporation 2014

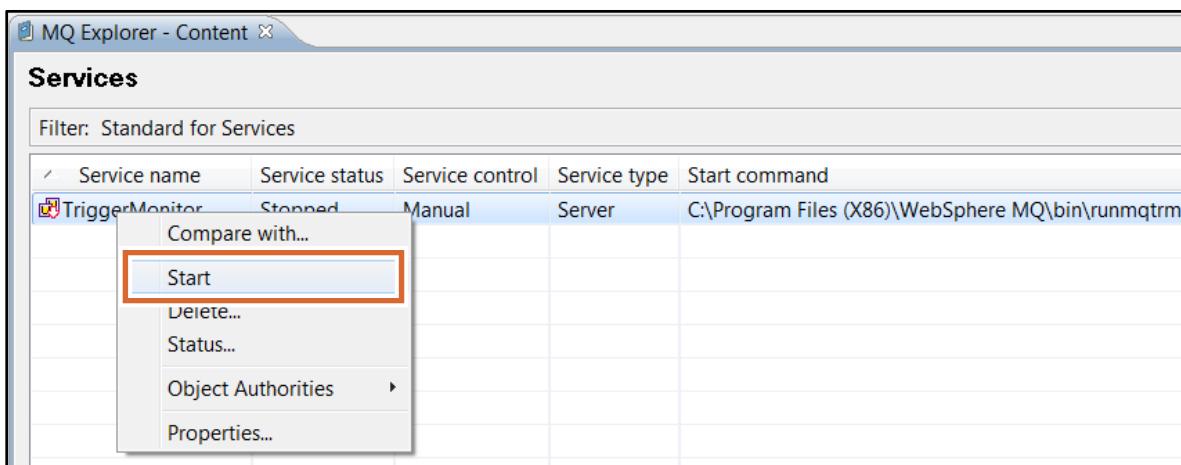
Figure 6-13. Trigger monitor service properties

WM2091.0

### Notes:

This figure describes the properties on the **New Service** window and how to configure them for a trigger monitor.

## Starting the trigger monitor in IBM MQ Explorer



1. In the Navigator view, click **Services** to display the services in the Content view.
2. In the **Services** Content view, right-click the service and then click **Start**. The icon next to the service changes to show whether the service is running.

© Copyright IBM Corporation 2014

Figure 6-14. Starting the trigger monitor in IBM MQ Explorer

WM2091.0

### Notes:

After you define the trigger monitor service in the **New Service** window, it is listed in the Services view.

You must start the trigger monitor from the MQ Explorer by right-clicking the service and then clicking **Start**, as shown in this figure.

## Testing a trigger monitor with IBM MQ samples

- **amqsreq ReqQ QMgrName ReplyToQ**
  - Reads text from the standard input device, converts them to request messages, and puts the messages on the request queue
  - When a blank line is entered, the program gets the messages from the reply-to queue and displays the messages on the standard output device
- **amqsech**
  - Started by a trigger monitor
  - Gets a message from the request queue
  - Creates a message with the same application data as the original message
  - Puts the new message on the reply-to queue
- **amqsinq**
  - Started by a trigger monitor
  - Gets a message from the request queue
  - Constructs a reply message with the attribute values
  - Puts the message on the reply-to queue

© Copyright IBM Corporation 2014

Figure 6-15. Testing a trigger monitor with IBM MQ samples

WM2091.0

### Notes:

You can use some MQ sample programs to verify that your trigger monitor is working.

The **amqsreq** sample program is started from a command prompt. The first parameter is the name of the request queue. The second parameter is the queue manager name. The third parameter is the reply-to queue name. The program reads lines of text from the standard input device, converts them to request messages, and puts the messages on the named queue. When a blank line is entered, the program begins to get the messages from the reply-to queue and displays those messages on the standard output device.

A trigger monitor starts the **amqsech** sample program. The program connects to the queue manager that is named in the structure that is passed to it by the trigger monitor and opens the request queue that is also named in the structure.

A trigger monitor starts the **amqsinq** sample program. The program connects to the queue manager, opens the request queue, gets a message from the queue, and interprets the application data as the name of a queue. The **amqsinq** program constructs a reply message and puts the message on the reply-to queue. The program then gets each of the remaining messages on the request queue in turn and generates a reply in the same way. When the request queue is empty, the program closes the queue and disconnects from the queue manager.

## Trigger monitor messages

- Message can be written to:
  - Standard output device
  - Error log
  
- Normal activity messages
 

Examples:

  - Trigger monitor that is started
  - Waiting for a trigger message
  - Trigger monitor ended
  
- Abnormal condition messages
 

Examples:

  - Initiation queue cannot be opened
  - Use of trigger monitor not authorized
  - Error starting triggered application

© Copyright IBM Corporation 2014

Figure 6-16. Trigger monitor messages

WM2091.0

### Notes:

Trigger operation messages are produced for two reasons.

- There are messages to report on normal activities such as when a trigger monitor starts and when it ends. These messages do not normally require any user action.
- There are messages to report on abnormal conditions such as when a trigger monitor fails to open the initiation queue or when it fails to start the specified application. These messages normally indicate that user action is required to correct the condition.

Examples of both types of messages are provided in the figure. Trigger monitor messages can be written to a standard output device to an error log.

If the trigger monitor program detects an error, it can put a trigger message on the dead-letter queue. The trigger monitor adds a dead-letter header structure (MQDLH) to the message. It uses a feedback code in the **Reason** field of the structure to explain why it put the message on the dead-letter queue.

## Triggering program problems

- Make sure that the trigger monitor is running.
- Check that the trigger monitor is monitoring the initiation queue, not the trigger queue.
- Verify that your applications are putting messages to the trigger queue, not the initiation queue.
- Use the string that is specified in the `ApplId` property of the process definition to try to start the trigger program manually.
- Verify that the user ID that is used to start the trigger monitor has the authority to access the entire path to the runnable file.

© Copyright IBM Corporation 2014

Figure 6-17. Triggering program problems

WM2091.0

### Notes:

The figure lists diagnostic tips if you are having problems with triggering.

First, make sure that the trigger monitor is running.

- On Windows, use the Task Manager and look for the `runmqtrm` program.
- On UNIX, type: `ps -ef | grep runmqtrm`

It is common to confuse the trigger queue with the initiation queue. Verify that the trigger monitor is monitoring the initiation queue and putting messages on the trigger queue.

You must also ensure that the user ID that is used to start the trigger monitor can access the `runmqtrm` program.



## Unit summary

Having completed this unit, you should be able to:

- Configure IBM MQ to enable a trigger monitor
- Run the trigger monitors on UNIX and Windows systems
- Determine the cause of a triggering failure

© Copyright IBM Corporation 2014

Figure 6-18. Unit summary

WM2091.0

### Notes:

Use of message-driven triggering starts applications as they are needed, alleviating the requirement to manually start them.



## Checkpoint questions

1. Which of the following messages from the trigger monitor are reporting on normal activities?
  - a. Waiting for a trigger message
  - b. Initiation queue cannot be opened
  - c. Use of trigger monitor not authorized
  - d. Trigger monitor ended
2. True or False: A trigger monitor requires a process object.

© Copyright IBM Corporation 2014

Figure 6-19. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

1.

2.

## Checkpoint answers

1. **a** and **d**. **b** and **c** report abnormal conditions.
2. True.

© Copyright IBM Corporation 2014

Figure 6-20. Checkpoint answers

WM2091.0

### Notes:

## Exercise 4



### Implementing a trigger monitor

© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.0

Figure 6-21. Exercise 4

WM2091.0

### Notes:

In this exercise, you modify queue parameters to implement triggering. You define a process object that identifies an MQ utility (`amqsech`) that is triggered when a message arrives on that queue. Students also use the `amqsreq` utility to send messages to the queue and retrieve the responses.



## Exercise objectives

After completing this exercise, you should be able to:

- Apply triggering parameters to queues
- Start a trigger monitor
- Test triggering by using IBM MQ sample programs

© Copyright IBM Corporation 2014

Figure 6-22. Exercise objectives

WM2091.0

### Notes:

See the Student Exercises Guide for detailed instructions.



# Unit 7. Message persistence and recovery

## What this unit is about

This unit describes the various ways that IBM MQ maintains messages. The unit also explains the differences between circular and linear logging, and the implications of using persistence. Transaction management and the methods for capturing and restoring an object image are also described.

## What you should be able to do

After completing this unit, you should be able to:

- Describe how IBM MQ uses logging to record significant changes to the data controlled by the queue manager
- Describe the difference between circular and linear logging
- Evaluate situations where message persistence is required
- Use a media image to recover objects that become damaged

## How you will check your progress

- Checkpoint
- Hands-on exercise

## Unit objectives

After completing this unit, you should be able to:

- Describe how IBM MQ uses logging to record significant changes to the data controlled by the queue manager
- Describe the difference between circular and linear logging
- Evaluate situations where message persistence is required
- Use a media image to recover objects that become damaged

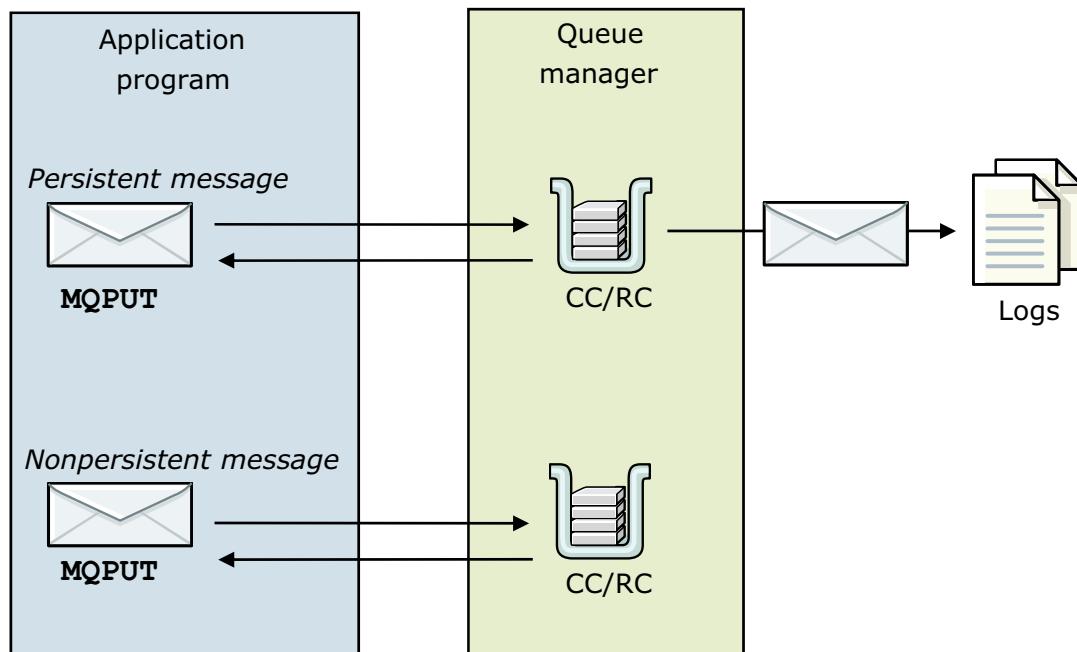
© Copyright IBM Corporation 2014

Figure 7-1. Unit objectives

WM2091.0

### Notes:

## Message persistence



**CC** = Completion code

**RC** = Reason code

© Copyright IBM Corporation 2014

Figure 7-2. Message persistence

WM2091.0

### Notes:

A message survives a queue manager restart if that message is defined as persistent. Persistence applies even when an operator stops the queue manager or because of a system failure. A queue manager restart succeeds because the persistent messages are written to a log when the messages are put to the queue.

A message that is defined as non-persistent does not survive a queue manager restart.

Non-persistence applies even if a queue manager finds an intact non-persistent message on disk during restart. Therefore, the queue manager discards it.

Both persistent and non-persistent messages can be stored on the same queue, except for temporary dynamic queues.

For each MQI call, such as an MQPUT, the queue manager returns a completion code (CC), and a reason code (RC) to indicate the success or failure of the call. The MQ completion codes are:

- 0: Successful completion (MQCC\_OK)
- 1: Warning (partial completion) (MQCC\_WARNING)
- 2: Call failed (MQCC\_FAILED)

## Defining message persistence on queues

- When defining queues by using an MQSC command:
  - **DEFPSIST (YES)** specifies that messages on this queue survive a restart of the queue manager when application specifies `MQPER_PERSISTENCE_AS_Q_DEF`
  - **NPMCLASS (HIGH)** on the local queue or model queue to allows non-persistent messages to survive a restart of a queue manager after that queue manager shuts down
- When defining queues by using MQ Explorer, queue **Default persistence** property to **Persistent**
- Assumes adequate storage is allocated for messages
- Can be overridden by the sending application



Message persistence should be set by the application. Set persistence at the queue level in special cases only.

© Copyright IBM Corporation 2014

Figure 7-3. Defining message persistence on queues

WM2091.0

### Notes:

It is possible to define the handling of persistent and non-persistent messages when you create queues.



#### Important

The sending application typically defines the persistence of the message and overrides the persistence set at the queue level.

The **DEFPSIST** attribute specifies the message persistence to be used when applications specify the `MQPER_PERSISTENCE_AS_Q_DEF` option.

- **NO**: Messages on this queue are lost across a restart of the queue manager.
- **YES**: Messages on this queue survive a restart of the queue manager.

The **NPMCLASS** attribute defines the level of reliability to be assigned to non-persistent messages that are put to the queue.

- **NORMAL:** Non-persistent messages are lost after a failure, or queue manager shutdown. These messages are discarded on a queue manager restart.
- **HIGH:** The queue manager attempts to retain non-persistent messages on this queue over a queue manager restart or switch over.

You cannot set the **NPMCLASS** attribute on z/OS.

You can also set the **Default persistence** attribute in MQ Explorer.

## Types of logs

Circular	Linear
Log files are viewed as a closed loop	Log files are viewed as a sequence
Amount of disk space that is required for the log does not increase with time	Log file is never deleted but becomes inactive when it contains no entries that are required to restart the queue manager
	Can be archived when it becomes inactive
	Required for media recovery

© Copyright IBM Corporation 2014

Figure 7-4. Types of logs

WM2091.0

### Notes:

MQ records all significant changes to the data controlled by the queue manager in a recovery log. Changes include creating and deleting objects, persistent message updates, transaction states, changes to object attributes, and channel activities.

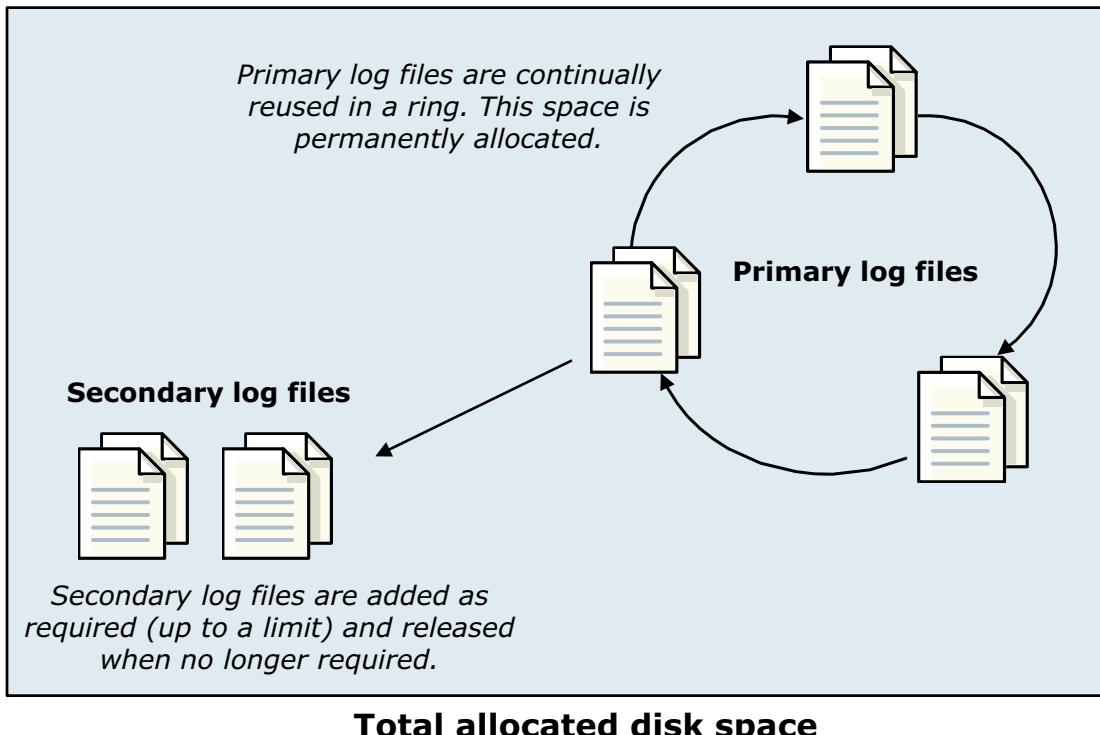
The log contains the information that is required for recovering all updates to message queues by:

- Keeping records of queue manager changes
- Keeping records of queue updates for use by the restart process
- Allowing you to restore data after a hardware or software failure

The type of logging is selected when a queue manager is created. Unless you request linear logging when you create a queue manager, circular logging is provided by default.

Periodically, the queue manager completes a log checkpoint. Information about the last checkpoint, including its location in the log, is held in the checkpoint file, `amqalchk.fil`.

## Circular logging



© Copyright IBM Corporation 2014

Figure 7-5. Circular logging

WM2091.0

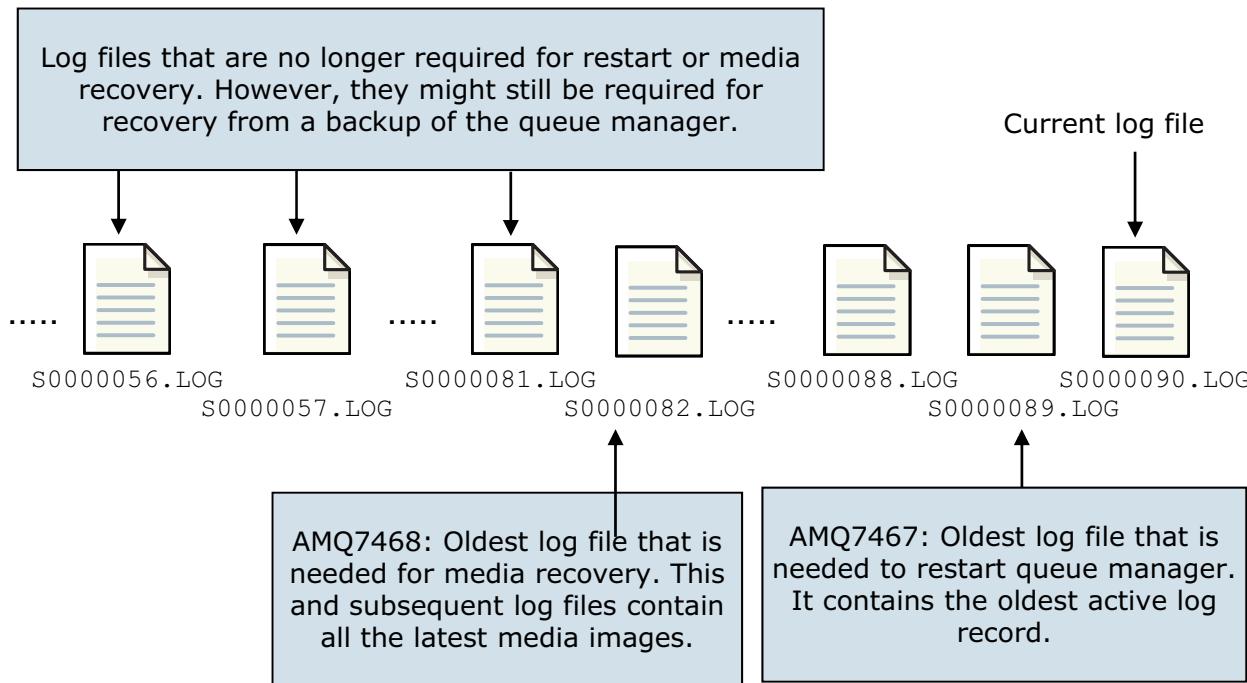
### Notes:

With circular logging, the log files are viewed as a closed ring. A log file becomes available for reuse when it contains no active log records. An active log record is one that is still required to restart the queue manager.

With circular logging, MQ can recover messages that follow a system failure but is unable to recover messages that follow a media failure. It has the advantage that the amount of disk space that is required for the log does not increase with time.

When you specify the type of logging, you can also specify the size and location of the log files if you do not want to accept the default values as specified in the MQ configuration file.

## Linear logging



**Safe implementation of linear logs requires regular oversight and management of the log files.**

© Copyright IBM Corporation 2014

Figure 7-6. Linear logging

WM2091.0

### Notes:

With linear logs, the log files are viewed as a sequence. A log file is never deleted but it becomes inactive when it contains no active log records. New log files are added to the sequence as required. Space is not reused. A linear log can recover from a media failure but it requires the regular archival of inactive log files.



#### Important

A normally operating linear logged queue manager can fail if the logs are not actively managed. Depending on message traffic load, message persistence, excessive MAXDEPTH, and MAXMSGL settings and disk space allocations, the logs will use all available space and the queue manager stops unless you take steps to prevent it.

IBM provides several SupportPacs that manage linear log files. Typically, these tools use a scripting language to identify inactive log extents and dispose them. The queue manager provides

commands to inquire on which extents are active, as an option for administrators that want to provide their own instrumentation for this process.

## Estimating the size of the log file

- Determined by configuration parameters
  - **LogFilePages**: Size of each primary and secondary log file in units of 4 K pages
  - **LogPrimaryFiles**: Number of pre-allocated primary log files
  - **LogSecondaryFiles**: Number of secondary log files that can be created when the primary log files are full
- Depends on operation and number and size of persistent messages

Examples:

Get message	260 bytes
Put persistent message	750 bytes + message length

© Copyright IBM Corporation 2014

Figure 7-7. Estimating the size of the log file

WM2091.0

### Notes:

After deciding whether the queue manager uses circular or linear logging, you must estimate the size of the log that the queue manager needs. The log configuration parameters that are listed in the figure determine the size of the log.

You can change the number of primary and secondary log files each time the queue manager starts.

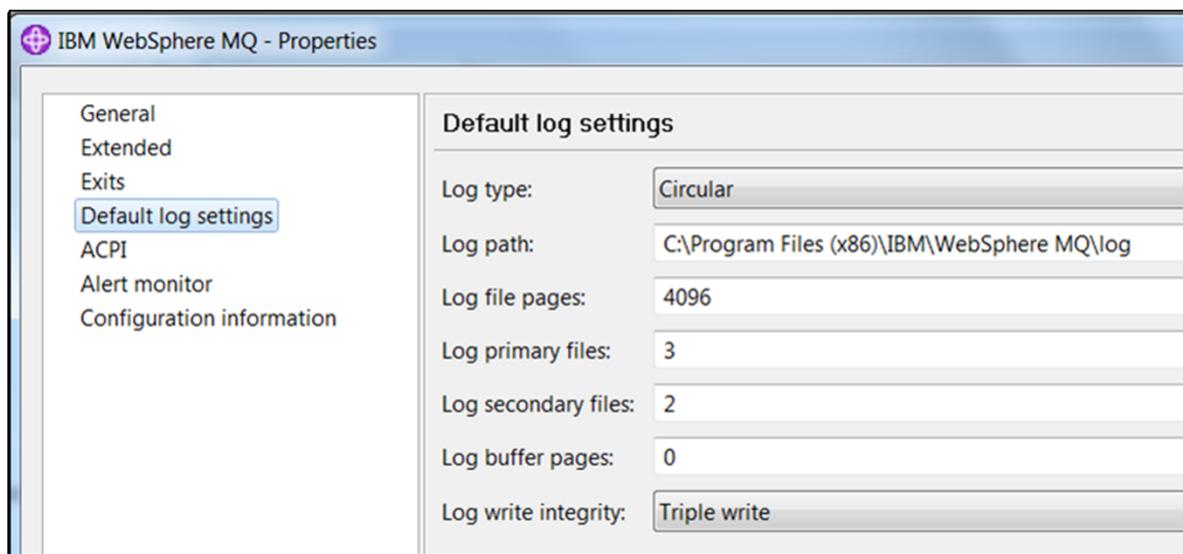
You cannot change the log file size; you must determine the file size before creating the queue manager.

The number of primary log files and the log file size determine the amount of log space that is preallocated when the queue manager is created.

The total number of primary and secondary log files cannot exceed 511 on UNIX systems, or 255 on Windows. In the presence of long-running transactions, the number of log files limits the maximum amount of log space available to the queue manager for restart recovery. The amount of log space that the queue manager might require for media recovery does not share this limit.



## Defining default log settings in IBM MQ Explorer



1. Right-click **IBM MQ** in Navigator and then click **Properties**.
2. Click **Default log settings**.
3. Specify log settings and then click **Apply**.

© Copyright IBM Corporation 2014

Figure 7-8. Defining default log settings in IBM MQ Explorer

WM2091.0

### Notes:

This figure shows how to configure the default log settings in the MQ **Properties**.

The default log settings include:

- Log type
- Log path
- Log file pages
- Log primary files
- Log secondary files
- Log buffer pages
- Log write integrity

## Recovering persistent messages

- To restart, a queue manager requires:
  - Log records that are written since the last checkpoint
  - Log records that are written by transactions that were still active at the time the queue manager stopped
- Persistent messages are recovered automatically when the queue manager is restarted
- A damaged local queue can be detected only later
  - Reported as "object damaged"
  - Normally needs to be recovered manually

© Copyright IBM Corporation 2014

Figure 7-9. Recovering persistent messages

WM2091.0

### Notes:

The queue manager recovers any damaged object that would prevent it from starting, but it would not normally include a local queue that is damaged. Such a queue can be detected later when an attempt is made to access it.

To restart, a queue manager requires:

- Log records that are written since the last checkpoint.
- Log records that are written by transactions that were still active when the queue manager stopped. Uncommitted persistent messages, put, or got inside these transactions, are rolled back during restart.

## Dumping the log

- Use `dmpmqlog` to dump a formatted version of the log
- Queue manager must be stopped
- By default, the dump starts from the head of the log. Optionally, the dump can start from:
  - Base of the log
  - Log record that a specified *log sequence number* (LSN) identifies
  - Log file that a specified *extent number* (linear logs only) identifies
- Log records include:
  - Put and get of persistent messages
  - Transaction events
  - Creation, alteration, and deletion of IBM MQ objects

© Copyright IBM Corporation 2014

Figure 7-10. Dumping the log

WM2091.0

### Notes:

The `dmpmqlog` control command can be used to dump a formatted version of the log. It can be used only when the queue manager is not running.

The head of the log is the checkpoint that starts the active portion of the log. Normally, the head of the log would be the most recent checkpoint. The head of the log might be positioned at an earlier checkpoint if transactions were active when the queue manager stopped and there are uncommitted persistent messages inside these transactions before the most recent checkpoint.

- The base of the log is the first log record in the log file that contains the head of the log.
- A unique log sequence number (LSN) identifies each log record.
- Each log file has a file name of the form `snnnnnnnn.LOG`, where `nnnnnnnn` is the extent number.

Full details, including a sample log with notes, can be found in the *IBM MQ System Administration Guide*.

## Damaged objects and media recovery

- IBM MQ objects can be marked as damaged
  - Corrupted data in the queue file
  - Missing queue file
  - Disk failure
- Damaged objects can be deleted
- A damaged object can be re-created from a **linear log**
  - Known as *media recovery*
  - Queue manager records media images automatically at certain times
  - Use the control command `rcdmqimg` to record the media image of a local queue regularly
- Media recovery
  - Automatic if a damaged object is detected during restart
  - For a local queue, it is normally done by using the control command `rcrmqobj`

© Copyright IBM Corporation 2014

Figure 7-11. Damaged objects and media recovery

WM2091.0

### Notes:

A damaged local queue is normally only detected when an attempt is made to access it. When this type of error occurs, the local queue can be re-created from a linear log by using the `rcrmqobj` control command. Record a media image of a local queue regularly so that the time needed to re-create it, if it becomes necessary to do so, does not become too long.

## Writing an image to a log

- Use the `rcdmqimg` command to write an image of an object, or group of objects, to the log for use in media recovery
  - Requires linear logging
  - Moves the log sequence number forward and frees up old log files for archival or deletion
  - Can take a long time to run if the queues on the system contain many messages

**Example:** Record an image of the queue manager object QMGR1 in the log.

```
rcdmqimg -t qmgr -m QMGR1
```

- t Identifies the types of object to record
- m Identifies the queue manager for which to record images

© Copyright IBM Corporation 2014

Figure 7-12. Writing an image to a log

WM2091.0

### Notes:

The control command to record a media image is `rcdmqimg`.

For example, the following command records a media image of a local queue:

```
rcdmqimg -m QMgrName -t qlocal QName
```

The object name can have a trailing asterisk to record any objects with names that match the portion of the name before the asterisk.

## Re-creating an object from an image

- Use the `rcrmqobj` command to re-create an object, or group of objects, from their images that are contained in the log
  - Can be used with using linear logging only
  - Use on a running queue manager

### Examples

- Re-create all local queues for the default queue manager:  
`rcrmqobj -t ql *`
- Re-creates all remote queues that are associated with queue manager store:  
`rcrmqobj -m store -t qr *`

© Copyright IBM Corporation 2014

Figure 7-13. Re-creating an object from an image

WM2091.0

### Notes:

The `rcrmqobj` control command can be used to re-create a damaged object.

For example, the following command re-creates a local queue:

```
rcrmqobj -m QMgrName -t qlocal QName
```

## Valid object types for recording and re-creating images

<b>* or all</b>	All object types
<b>authinfo</b>	Authentication information object for SSL channel security
<b>channel or chl</b>	Channels
<b>clntconn or clcn</b>	Client connection channels
<b>clchltab</b>	Client channel table
<b>listener or lstr</b>	Listener
<b>namelist or nl</b>	Namelists
<b>process or prcs</b>	Processes
<b>queue or q</b>	All types of queue
<b>qalias or qa</b>	Alias queues
<b>qlocal or ql</b>	Local queues
<b>qmmodel or qm</b>	Model queues
<b>qremote or qr</b>	Remote queues
<b>service or srvc</b>	Service
<b>syncfile</b>	Channel synchronization file
<b>topic or top</b>	Topics

© Copyright IBM Corporation 2014

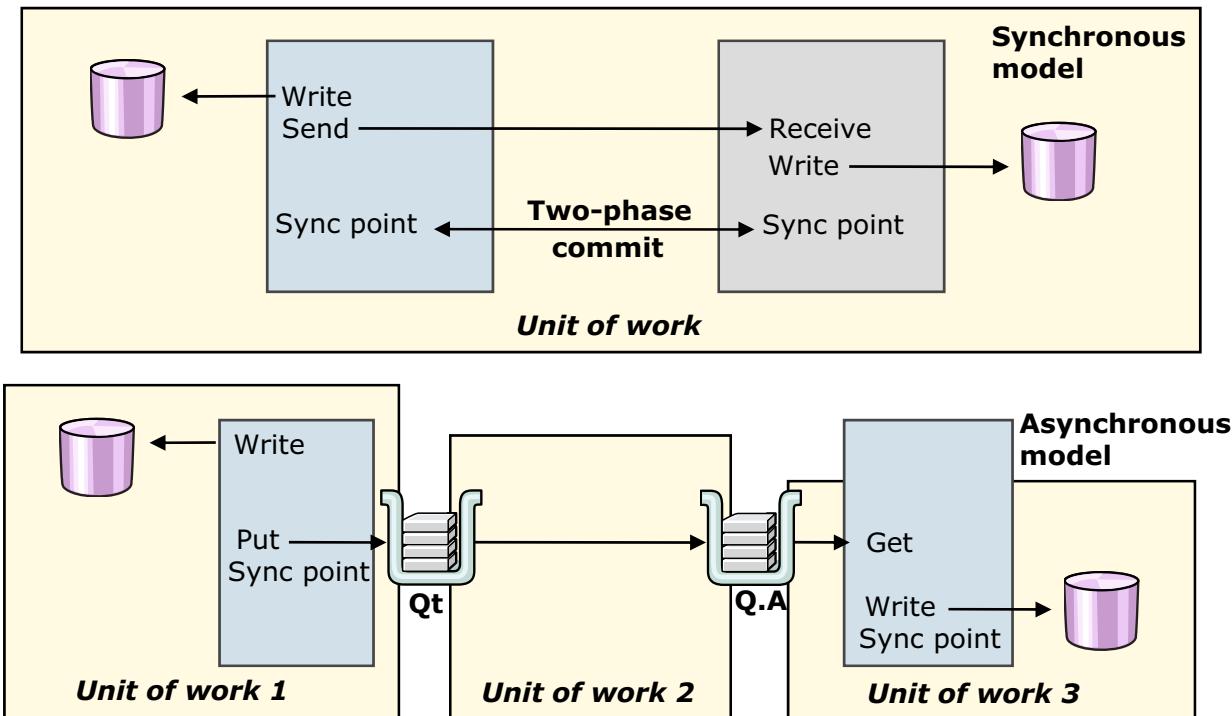
Figure 7-14. Valid object types for recording and re-creating images

WM2091.0

### Notes:

When you use a control command to record a media image and re-create objects, you can specify the object type to record and re-create. This figure lists the object types and syntax.

## Multiple, asynchronous units of work



© Copyright IBM Corporation 2014

Figure 7-15. Multiple, asynchronous units of work

WM2091.0

### Notes:

A complex business transaction might be implemented as many separate asynchronous processes where each process constitutes a unit of work.

The importance of the assured, one time delivery property of MQ in this design can clearly be seen. This property, with the ability to change MQ resources as part of a unit work, means that the separate processes can operate safely and independently without holding complex locks across a network.

Implementing a business transaction as a single distributed unit of work requires a two-phase commit protocol. The MQ implementation uses multiple units of work that act asynchronously, as shown in the lower flow in the figure.

In the first unit of work, the application writes to a database, puts a message on a queue, and then sends a sync point to commit the changes to the two resources. Because the queue is a remote queue, the message gets no further than the transmission queue within this unit of work. When the unit of work is committed, the message becomes available for retrieval by the sending MCA.

In the second unit of work, the sending MCA gets the message from the transmission queue and sends it to the receiving MCA on the system that contains the second database. The receiving MCA

puts the message on the destination queue. When this unit of work is committed, the message becomes available for retrieval by the second application.

In the third unit of work, the second application uses the data that is contained in the message to get the message from the destination queue and updates the database.

## Sync point control

```

MQGET (customer order)
Update DB
MQPUT (dispatch request)
MQPUT (delivery confirmation)
Commit

```

- Option on MQPUT and MQGET calls
  - If **NO\_SYNCPOINT**, message is added or removed immediately
  - If **SYNCPOINT**, result of an MQPUT or an MQGET call becomes visible only when the unit of work is committed
  - **SYNCPOINT\_IF\_PERSISTENT** gets message within sync point control only if it is persistent (on MQGET call only)
- Default depends on the operating system

© Copyright IBM Corporation 2014

Figure 7-16. Sync point control

WM2091.0

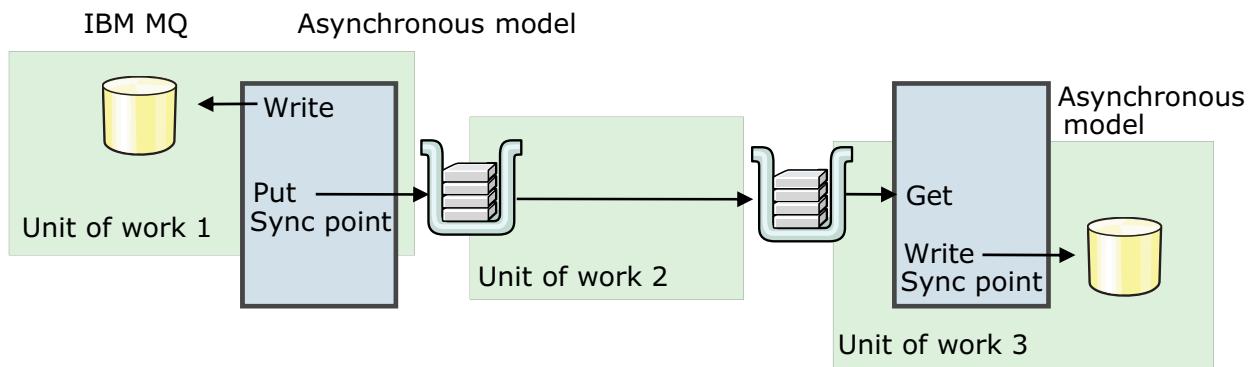
### Notes:

A message that is put on a queue within sync point control is not available for another application to get until the unit of work is committed. Similarly, a message got from a queue within sync point control does not cause the message to be removed from the queue until the unit of work is committed.

An application can specify whether an MQPUT call or an MQGET call is within, or outside of, sync point control or leave it as the default. The default option depends on the operating system.

- Using the **NO\_SYNCPOINT** option, a message is added to a queue or removed from a queue immediately.
- Using the **SYNCPOINT** option, the result of an MQPUT or MQGET call takes effect when the unit of work is committed.
- The **SYNCPOINT\_IF\_PERSISTENT** option on an MQGET call causes the request to operate within the normal unit-of-work protocols, if the message retrieved is persistent.

## Compensating transactions



**Local** Sync point participation = committed changes

Examples:

- |  |   |
|--|---|
| 1. Debit account,<br>Send credit message               | 1. Account not known,<br>Send debit reversal              |
| 2. Update travel itinerary,<br>Send flight reservation | 2. Fully booked,<br>Send nearest alternative              |
| 3. Confirm order,<br>Send shipping request             | 3. Out of stock so reorder,<br>Send revised delivery date |

© Copyright IBM Corporation 2014

Figure 7-17. Compensating transactions

WM2091.0

### Notes:

Local sync point coordinates messages with database updates on that system. But the actual processing of the message might take place on a different system, or later.

When a message is processed later, the application might find an error, in the database, that prevents the message from being processed. It is too late to roll back the original unit of work.

The answer is to include *compensating transactions*, which send messages to reverse the original request. The figure lists some examples of compensating transactions.



## Coordinating units of work

- **Local** unit of work is one in which the only resources that are updated are the resources of the queue manager
- **Global** unit of work is one in which the resources of other resource managers are also being updated

© Copyright IBM Corporation 2014

Figure 7-18. Coordinating units of work

WM2091.0

### Notes:

A *local* unit of work is one in which the resources that are updated are those resources of the queue manager to which the application is connected.

A *global* unit of work is one in which the resources of other resource managers and the resources of a queue manager are updated. The coordination of global units of work can be internal or external to the queue manager.



## Database coordination

- Supported database managers
  - DB2
  - Informix
  - Oracle
  - Sybase
- Restrictions
  - An IBM MQ client cannot participate in a global unit of work, unless it is an IBM MQ extended transactional client
  - Only one queue manager can participate in a global unit of work
  - Database server can be installed on a different server from the queue manager server, while the database client is installed on the same server as the queue manager, and it supports this function

© Copyright IBM Corporation 2014

Figure 7-19. Database coordination

WM2091.0

### Notes:

The figure lists the XA-compliant database managers that MQ queue managers support and can participate in a global unit of work that MQ coordinates.

There are some restrictions for the internal coordination of global units of work.

- An MQ client application cannot participate in a global unit of work unless it is an MQ extended transactional client.

An MQ extended transactional client allows a client application, within the same unit of work to:

- Put messages to, and get messages from the connected queue manager's queues
- Update the resources of a resource manager other than an MQ queue manager
- You cannot configure two or more queue managers as participants in a global unit of work even if a queue manager is XA-compliant. An application can connect to only one queue manager at a time.
- Normally, updates to MQ resources and to resources of a database manager must be made on the same system. MQ cannot coordinate a distributed unit of work. However, a database manager can be installed on a different system to the queue manager provided it can supply an XA-compliant client feature that exists on the same system as the queue manager.



## External coordination of global units of work

- AIX, Solaris, HP-UX, and Linux
  - Oracle Tuxedo
  - TXSeries for Multiplatforms
  - WebSphere Application Server
- Windows
  - Oracle Tuxedo
  - TXSeries for Multiplatforms
  - WebSphere Application Server
  - Microsoft MTS/COM

© Copyright IBM Corporation 2014

Figure 7-20. External coordination of global units of work

WM2091.0

### Notes:

The external sync point coordinators that are listed in the figure use the X/Open XA interface for coordinating changes to MQ resources and to other resource managers.

An MQ client application can participate in a global unit of work by using the MQ extended transactional client and an external sync point coordinator. The extended transactional client is supported on AIX, Solaris, HP-UX, Linux, and Windows.

As this list occasionally changes, check the IBM website for the most recent information:  
<http://www.ibm.com/software/integration/wmq/support/>

## Unit summary

Having completed this unit, you should be able to:

- Describe how IBM MQ uses logging to record significant changes to the data controlled by the queue manager
- Describe the difference between circular and linear logging
- Evaluate situations where message persistence is required
- Use a media image to recover objects that become damaged

© Copyright IBM Corporation 2014

Figure 7-21. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. Linear logging should be considered for:
  - a. Ease of administration
  - b. Requirement that persistent queues be recoverable
  - c. Requirement that queues be rolled back to a point particular time
  - d. Production environment
2. True or False: Use the `dmpmqlog` command to back up IBM MQ logs.
3. True or False: IBM MQ can be both an XA transaction coordinator and a transaction manager.

© Copyright IBM Corporation 2014

Figure 7-22. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.



## Checkpoint answers

1. **b and d.**
2. **False.** Logs can be backed-up only if the queue manager is stopped, then use a suitable file backup utility. The `dmpmqlog` program is a diagnostic tool only.
3. **True.**

© Copyright IBM Corporation 2014

Figure 7-23. Checkpoint answers

WM2091.0

### Notes:

## Exercise 5



© Copyright IBM Corporation 2014  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.0

Figure 7-24. Exercise 5

WM2091.0

### Notes:

In this exercise, you capture a media image of a queue, deliberately damage the queue, and then recover it.



## Exercise objectives

After completing this exercise, you should be able to:

- Capture an object media image
- Re-create an IBM MQ object from an object media image

© Copyright IBM Corporation 2014

Figure 7-25. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for detailed instructions.



# Unit 8. Identifying the cause of problems

## What this unit is about

This unit describes the IBM MQ trace mechanism, explains the contents of the **AMQERR01.LOG** file, and describes the First Failure Support Technology. Problem determination hints and tips are also provided for some of the more common types of problems.

## What you should be able to do

After completing this unit, you should be able to:

- Determine the possible causes and locations of a missing message
- Analyze the error logs that IBM MQ generates
- Locate First Failure Support Technology (FFST) files on a system
- Use an IBM MQ trace to collect detailed information about IBM MQ operation
- Describe some common problem types and how to approach initial problem determination
- Stop and remove a queue manager manually

## How you will check your progress

- Checkpoint questions
- Hands-on exercise

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Determine the possible causes and locations of a missing message
- Analyze the error logs that IBM MQ generates
- Locate First Failure Support Technology (FFST) files on a system
- Use an IBM MQ trace to collect detailed information about IBM MQ operation
- Describe some common problem types and how to approach initial problem determination
- Stop and remove a queue manager manually

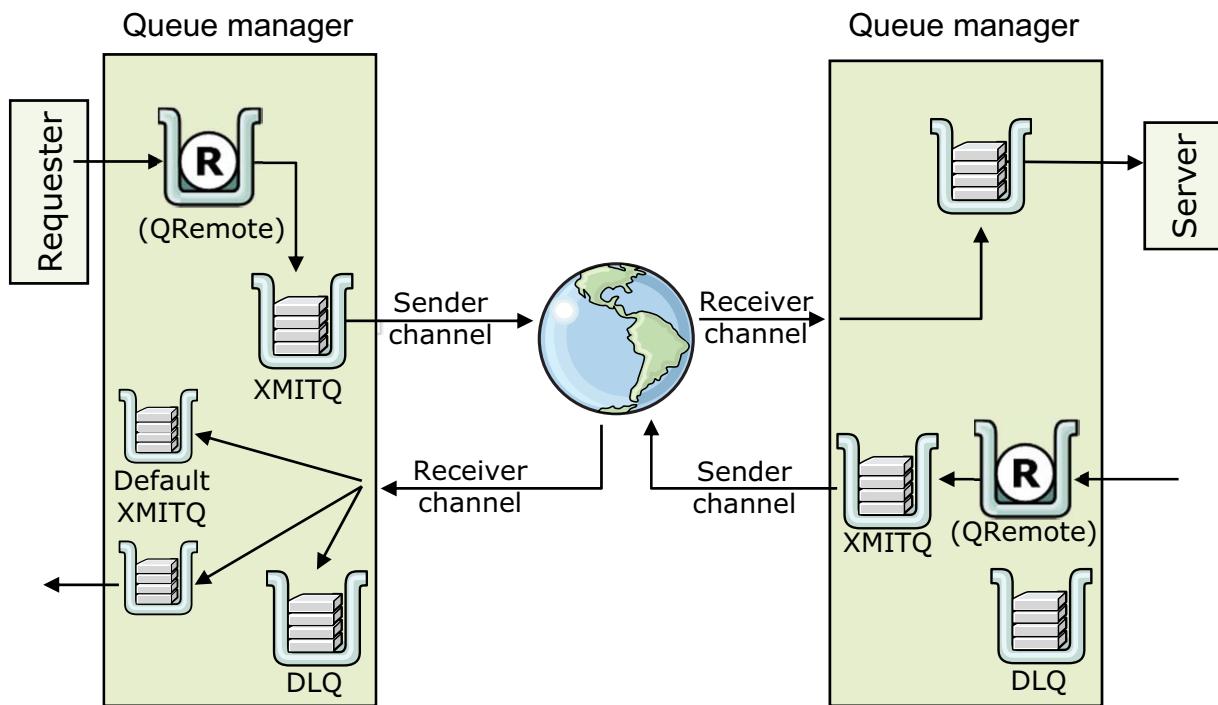
© Copyright IBM Corporation 2014

Figure 8-1. Unit objectives

WM2091.0

### Notes:

## Where is the message?



© Copyright IBM Corporation 2014

Figure 8-2. Where is the message?

WM2091.0

### Notes:

The figure shows a typical MQ application with two queue managers and queues. There are many places a message might end up after a successful MQPUT, especially when it must cross systems.

MQ does not lose messages by design, but messages can be lost for various reasons.

## Finding out why a message is on the dead-letter queue

1. Browse the message on the dead-letter queue.
2. Locate the dead-letter header (DLH).

Example:

```
444C 4820 0000 0001 0000 0108 4D59 2E51 'DLH .....MY.Q'
```

3. Find the dead-letter reason (third word in dead-letter header).

Example: Dead-letter reason is: x'0000 0108'

4. Convert the reason from hex to decimal.

Example: Hex '0000 0108' is decimal 264.

5. Use the dead-letter reason code to determine why the message was placed on the dead-letter queue.

Example: MQFB\_CHANNEL\_FAIL 264 x'00000108'

© Copyright IBM Corporation 2014

Figure 8-3. Finding out why a message is on the dead-letter queue

WM2091.0

### Notes:

One of the first places to look for a message is the queue manager dead-letter queue. The figure lists the steps for determining why a message was put on the dead-letter queue.

The first steps are to browse the message on the dead-letter queue and find the dead-letter reason code in the dead-letter header.

The dead-letter reason provides a code that identifies the reason that the message was put on the dead-letter queue. It reveals a reason code (MQRC\_\*) or a feedback code (MQFB\_\*).

After you determine the dead-letter reason code, you can use it to determine the reason why the message was placed on the dead-letter queue.

## Missing message checklist

1. Is the MQ completion code “OK”?
2. Are there problems within the MQ network?
3. Is the message persistent or not persistent?
4. Did the message expire?
5. Was the message committed?

© Copyright IBM Corporation 2014

Figure 8-4. Missing message checklist

WM2091.0

### Notes:

What if the message is not on the dead-letter queue and the message seems to be lost? What can you do next?

This figure lists the items to check when investigating the possible loss of a message. Each of these items is described in more detail in this unit.

## Step 1: Checking MQ reason codes

- Queue manager or exit program returns a completion code and reason code to indicate the success or failure of the call
- Use the `mqrc` command to display information about return codes if completion code is `MQCC_WARNING` or `MQCC_FAILED`

Example:

```
C:\> mqrc 2085
2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME
```

- Developers should include logic in applications to ensure that the MQI call was successful

Example:

```
IF REASON IS NOT EQUAL TO MQRC-NONE
  IF REASON IS EQUAL TO MQRC_Q_FULL
    DISPLAY 'Queue contains max number of messages'
  ELSE
    DISPLAY 'MQPUT ended with reason code ' REASON
  END-IF
END-IF
```

© Copyright IBM Corporation 2014

Figure 8-5. Step 1: Checking MQ reason codes

WM2091.0

### Notes:

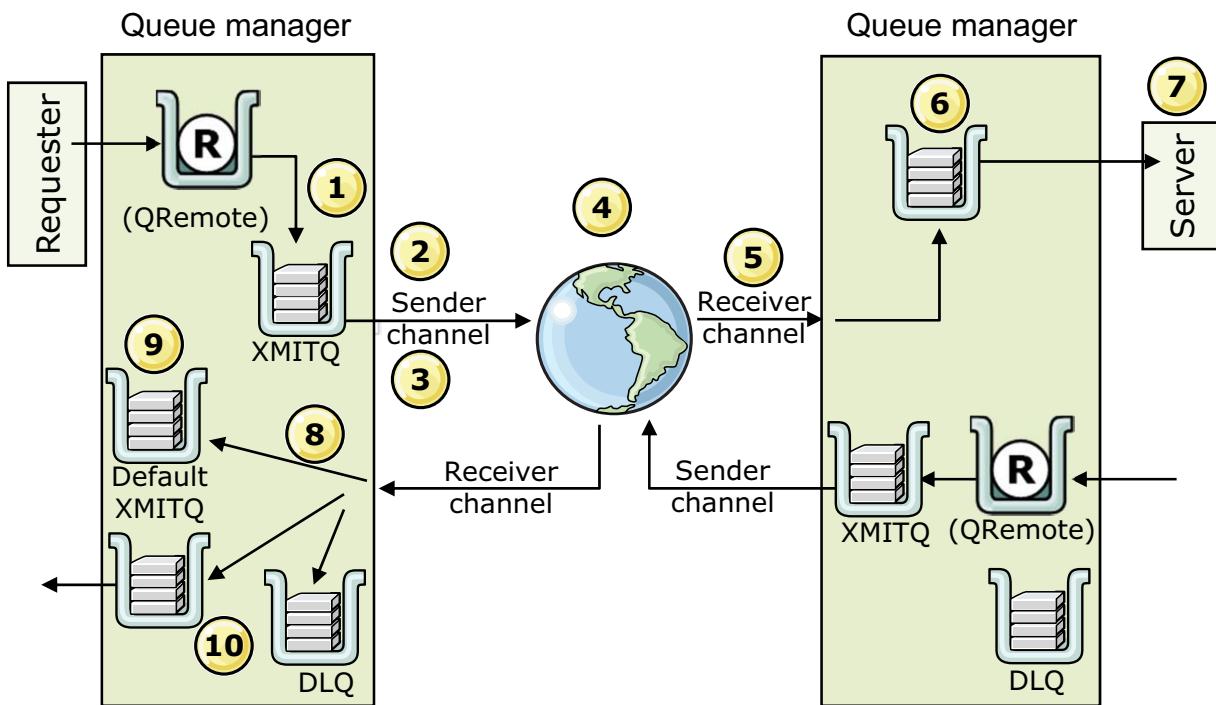
The queue manager or exit program returns a completion code and a reason code for each MQI and MQAI call to indicate the success or failure of the call.

Always check the completion code of a call and ensure that the application behaves appropriately if an error is encountered. If the completion code is not `MQCC_OK`, it is possible that the message never made it to the target queue or topic.

If the completion code is `MQCC_WARNING` or `MQCC_FAILED`, use the `mqrc` control command to check the reason code for more information about the problem.

An application that is retrieving a message should always check the completion code or the reason code. The figure shows sample logic that tests the reason code. If the reason code is not equal to `MQRC-NONE`, the sample logic displays an appropriate error message.

## Step 2: Checking for MQ network problems



© Copyright IBM Corporation 2014

Figure 8-6. Step 2: Checking for MQ network problems

WM2091.0

### Notes:

The next step for determining the reason for a missing message is to check the MQ network for problems.

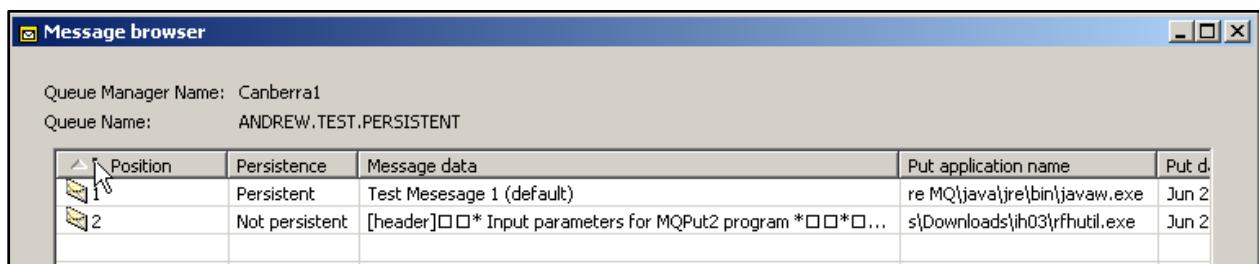
This figure shows a typical application with two queue managers and multiple queues. Potential failure points are highlighted in the figure:

1. Incoming messages were not put to the transmission queue because it is “put” or “get” disabled, or not defined as a transmission queue.
2. The sender channel is not running or was not triggered to start, or cannot start because the transmit queue is “get” disabled or at the maximum channel (MAXCHL) limit.
3. The channel initiator is not running.
4. The channel failed to start because of a network problem.
5. The receiver channel did not start because listener is not running, or the receiver failed to put to a target queue because the target queue is “put” disabled or full. Maybe the channel initiator or event of the queue manager is not running.
6. The server program did not get the message because the target queue has “get” disabled.

7. The server program is not started or failed to be triggered.
8. The receiver channel received a message for another remote queue manager, but no transmission queue is defined for it, so the receiver channel put the message to the default transmit queue.
9. No channel is defined to serve the default transmission queue.
10. The receiver channel received an inbound message for an unknown local target queue, or the local target queue was full, so it was put on the dead-letter queue.

The same failure points are present in both directions; this list is not complete.

## Step 3: Checking message persistence



- If the queue manager is restarted while a non-persistent message is on the queue, the message is lost
  - Use MQ Explorer or the `DISPLAY QUEUE` command to check the **Default persistence** property (`DEFPSIST`) on the queue
  - Use the MQ Explorer message browser to check the message persistence that the PUT application defines

© Copyright IBM Corporation 2014

Figure 8-7. Step 3: Checking message persistence

WM2091.0

### Notes:

If the queue manager is restarted while a non-persistent message is on the queue, the message is discarded. This behavior is standard for non-persistent messages. They have much lower memory and storage requirements because they are not hardened to a log.

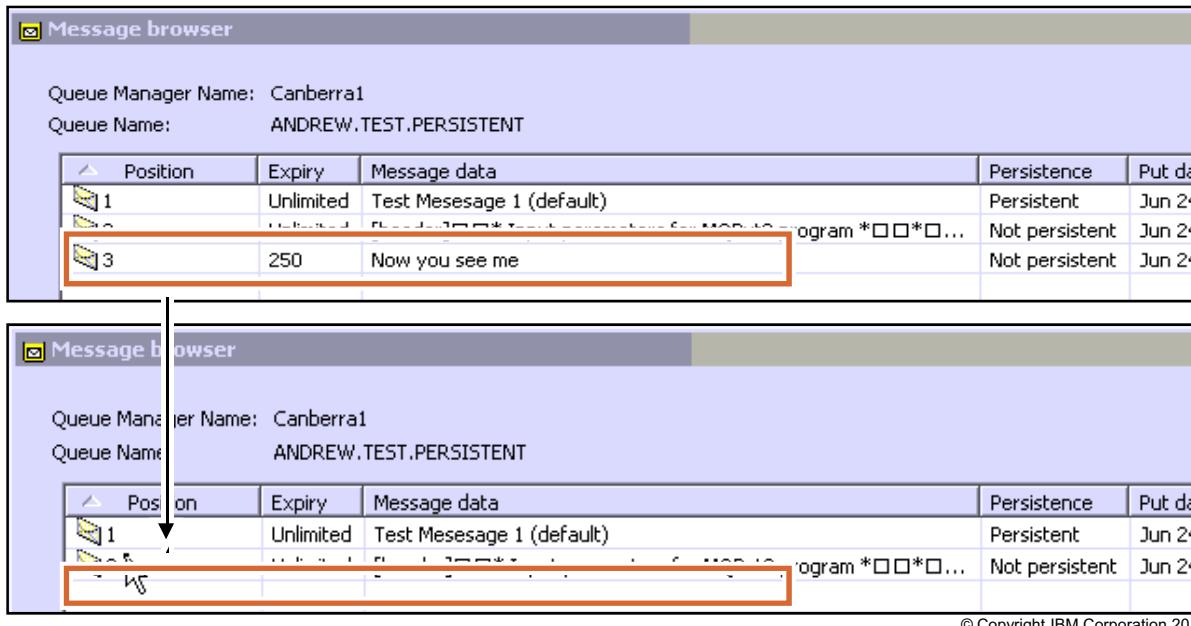
Restarting a queue manager with non-persistent messages on the queues is one of the most common reasons that messages are lost.

Queues that have the default persistence property (`DEFPSIST`) set to "yes" can have non-persistent messages on them. The application that puts the message on the queue can override the queue persistence property in the `MQMD` and force a non-persistent message on a queue.

One way to see the persistence of a message on a queue is to use the message browser in MQ Explorer.

## Step 4: Checking message expiration

- Application that puts the message on the queue sets the expiration
- Delays in processing can cause messages to expire before they were intended
- Check the **Expiry** property on the message by using the MQ Explorer message browser



© Copyright IBM Corporation 2014

Figure 8-8. Step 4: Checking message expiration

WM2091.0

### Notes:

Messages can expire, but not by default. Applications can explicitly set a life span of a message.

Setting a life span is typically the case with time-sensitive data such as quotations.

The message expiry time is in 1/10ths of second and represents the amount of time that remains (time to live). In the figure, the top image shows that the third message on this queue was put with an expiration time of 30 seconds (300 tenths of a second).

The bottom image shows the contents of the queue after the message expired; it is no longer on the queue.

Sometimes delays in message processing can result in messages that expire before they were intended. This case is an application design issue.

## Step 5: Uncommitted messages

- Messages that are put to a queue under transactional control, are not available to other application until the transaction is either committed or rolled back
- Use the **DISPLAY QSTATUS** command to see information about the queue and processes (handles) that are attached to the queue
  - Current queue depth property (**CURDEPTH**) includes the uncommitted messages, even though they are not available
  - Uncommitted messages (**UNCOM**) property indicates whether some messages are not committed
  - Open processes (**OPPROCS**) property identifies the number of processes that are attached to the queue

© Copyright IBM Corporation 2014

Figure 8-9. Step 5: Uncommitted messages

WM2091.0

### Notes:

Messages that are put to a queue under transactional control, are not available to another application until the transaction is either committed or rolled back. However, the display of the current depth of a target queue includes the uncommitted messages, even though they are not available. In this scenario, it can seem that messages are lost if the number of messages that an application processes do not match the current depth count that is seen on a previous display. More often, the problem seems to be that messages are irretrievable.

Enter the MQSC **DIS QSTATUS** command to discover information about the processes that are attached to the queue.

Example: **DIS QSTATUS(TESTQ1) TYPE(HANDLE) ALL**

Then, check the uncommitted messages properties to see whether some messages are not committed.

## Displaying the queue status

- Use the MQSC command **DISPLAY QSTATUS** to display the status of one or more queues and the handles that are accessing the queues

```
DIS QSTATUS (TESTQ1) TYPE (QUEUE)
AMQ8450: Display queue status details.
    QUEUE (TESTQ1)                               TYPE (QUEUE)
    CURDEPTH (4)                                IPPROCS (0)
    :
    OPPROCS (1)                                QTIME( , )
    UNCOM (YES)
```

```
DIS QSTATUS (TESTQ1) TYPE (HANDLE)
AMQ8450: Display queue status details
    QUEUE (TESTQ1)                               TYPE (HANDLE)
    APPLTAG (c:\WMQ\bin\amqspput.exe)           APPLTYPE (USER)
    :
    OUTPUT (YES)                                PID (10880)
    QMURID (0.0)                                SET (NO)
    TID (1)
    URID (XA_FORMATID[ ] XA_GTRID[ ] XA_BQUAL[ ])
    URTYPE (QMGR)                               USERID (slw@localhost)
```

© Copyright IBM Corporation 2014

Figure 8-10. Displaying the queue status

WM2091.0

### Notes:

In the first example in this figure, the MQSC **DIS QSTATUS** command is run against the queue with the **TYPE(HANDLE)** option. The results of the command show that the queue has a current depth (**CURDEPTH**) of 4, but that 1 or more of these messages are uncommitted as **UNCOM(YES)** indicates. However, you cannot tell how many of these four messages are unavailable.

In the second example, the **DIS QSTATUS** is run against the queue with the **TYPE(HANDLE)** option. The results of this command show that the application **amqspput** is attached to the queue.

## First Failure Support Technology (FFST)

- “Unexpected” errors
  - Internal queue manager failure but not automatically a software problem
  - Associated storage dumps might also exist
  - Data to help analyze software events
- If an error occurs:
  - Note a description of the problem
  - Look for any related error log entries
  - Identify any FFST reports

© Copyright IBM Corporation 2014

Figure 8-11. First Failure Support Technology (FFST)

WM2091.0

### Notes:

FFST is used to record an internal problem at the point at which it is discovered.

MQ uses FFST to:

- Detect and report software events, for example, internal queue manager failures
- Collect information about software events, for example, dumps of storage
- Generate data to help analyze software events, for example, probe IDs

Each FFST report contains various items of useful information.

- A probe ID that identifies where in the code the error was detected
- The date and time the error occurred
- Any associated error message
- A variable number of memory dumps that include the function stack and trace history.

If you experience frequent FFST reports that are related to shared memory on a UNIX system, it might mean that you need to change the values of certain kernel parameters to support MQ.

## FFST files

- Contain information about a configuration problem with the system or an MQ internal error
- Files are named **AMQnnnn.mm.FDC**
  - nnnn** is the ID of the process that reports the error.
  - mm** starts at 0. If the full file name exists, this value increments by one until a unique FFST file name is found.
- Location of FDC files:
  - Windows: **C:\ProgramData\IBM\MQ\errors**
  - UNIX: **/var/mqm/errors**

© Copyright IBM Corporation 2014

Figure 8-12. FFST files

WM2091.0

### Notes:

FFST files contain information about a configuration problem with the system or an MQ internal error.

FFST files are named **AMQnnnnn.mm.FDC**, where **nnnnn** is the process ID reporting the error and **mm** is a sequence number, normally 0.

An instance of a process writes all FFST information to the same FFST file. If multiple errors occur while a process runs, an FFST file can contain many records. These records indicate either a configuration problem with the system or an MQ internal error.

On Windows, when a process writes an FFST record it also sends a record to the Event Log. The record contains the name of the FFST file to help automatic problem tracking. The Event Log entry is made at the application level.

## FFST report example

```

Date/Time      :- Thu May 22 2014 07:19:02 Pacific Daylight Time
UTC Time       :- 1400768342.384000
UTC Time Offset :- 60 (Pacific Daylight Time)
Host Name      :- MyHost
Operating System :- Windows Server 2008 R2 Server Standard Edition
PIDS           :- 5724H7251
LVLS           :- 8.0.0.0
Product Long Name :- WebSphere MQ for Windows (x64 platform)
Vendor          :- IBM
O/S Registered   :- 1
Data Path        :- C:\ProgramData\IBM\MQ
Installation Path :- C:\Program Files\IBM\WebSphere MQ
Installation Name :- Installation1 (1)
License Type     :-
→ Probe Id      :- AD004020
Application Name :- MQM
→ Component    :- adhOpen
SCCS Info       :- F:\build\slot1\p000_P\src\lib\lqm\amqadho0.c,
. . .
UserID          :- MUSR_MQADMIN
→ Process Name  :- C:\Program Files\IBM\WebSphere MQ\bin64\amqzlaa0.
. . .
QueueManager     :- QML01
UserApp          :- FALSE
. . .
→ Major Errorcode :- arce_OBJECT_MISSING
Minor Errorcode  :- OK
Probe Type       :- INCORROUT
Probe Severity   :- 2
→ Probe Description :- AMQ6125: An internal WebSphere MQ error has occurred.

```

© Copyright IBM Corporation 2014

Figure 8-13. FFST report example

WM2091.0

### Notes:

This figure shows an FFST report (an FDC file) that reports a “log file full” condition on a Windows MQ system.

Some of the fields in the report provide useful information.

- **Probe Id:** A unique error code that identifies where in the code the error is detected
- **Component:** The function that failed
- **Process Name:** The name of the process that is running at the time of the failure
- **Major Errorcode:** The named reason for the failure
- **Probe Description:** The externalized message ID and meaning, which is written to the error log

## IBM MQ error logs

- Captures messages that concern:
  - Operation of MQ
  - Any queue managers that you start
  - Error data that comes from the channels that are in use
- Location depends on whether queue manager name is known and whether error is associated with a client
  - If queue manager name is known and queue manager is available:
 

Windows: C:\ProgramData\IBM\MQ\Qmgrs\QMgrName\errors\AMQERR01.LOG  
 UNIX: /var/mqm/qmgrs/QMgrName/errors/AMQERR01.LOG
  - If an error occurs with a client application:
 

Windows: C:\ProgramData\IBM\errors\AMQERR01.LOG  
 UNIX: /var/mqm/errors/AMQERR01.LOG
- Errors subdirectory can contain up to three error log files:  
**AMQERR01.LOG**, **AMQERR02.LOG**, **AMQERR03.LOG**

© Copyright IBM Corporation 2014

Figure 8-14. IBM MQ error logs

WM2091.0

### Notes:

Error messages are written to an error log file that is called **AMQERR01.LOG**. There is a separate error log file with this name in each of error directories that are as described in the figure.

Messages identify normal errors, typically caused by users, such as the use of a non-valid parameter on a control command.

When the error log file **AMQERR01.LOG** is full, its contents are copied to **AMQERR02.LOG** and **AMQERR01.LOG** is then reused. Before the copy, **AMQERR02.LOG** is copied to **AMQERR03.LOG**. The previous contents of **AMQERR03.LOG** if any, are discarded. In this way, **AMQERR02.LOG** and **AMQERR03.LOG** maintain a history of error messages.

On Windows, also examine the Windows application Event Log for relevant messages.

## Example error log contents

```
1/20/2014 09:01:40 - Process(13492.1) User(userlibm) Program(endmqsvc.exe)
                      Host(IBM-12345) Installation(Installation1)
                      VRMF(8.0.0.1)
AMQ7291: The MQ service for installation 'Installation1' failed to end with
error 1051.
```

**EXPLANATION:**

The attempt to end the MQ service (amqsvc.exe) for installation 'Installation1' failed, the error from the operating system was 1051.

The formatted message text for error 1051 is 'A stop control has been sent to a service that other running services are dependent on.' (if blank this indicates that no message text was available).

**ACTION:**

Check that the service named 'IBM WebSphere MQ (Installation1)' has been properly configured and is enabled, then re-issue the command.

----- endmqsvc.c : 419 -----

© Copyright IBM Corporation 2014

Figure 8-15. Example error log contents

WM2091.0

### Notes:

This figure shows an example of the contents of an error log. Each log entry is identified with a date and time stamp, the host, and MQ installation. It also includes the error message and the action to take.



## IBM MQ AMQ messages

- IBM MQ diagnostic messages that are grouped according to the part of IBM MQ from which they originate
  - AMQ4xxx: User interface messages (Windows and Linux systems)
  - AMQ5xxx: Installable services
  - AMQ6xxx: Common services
  - AMQ7xxx: IBM MQ product
  - AMQ8xxx: Administration
  - AMQ9xxx: Remote
- Use the `mqrc` command to display information about AMQ messages

© Copyright IBM Corporation 2014

Figure 8-16. IBM MQ AMQ messages

WM2091.0

### Notes:

MQ AMQ messages are diagnostic messages that are grouped according to the part of MQ from which they originate.

Each message contains:

- Four-digit decimal code.
- Summary of the message.
- Message severity:
  - 0: Information
  - 10: Warning
  - 20: Error
  - 30: Severe error
  - 40: Stop error
  - 50: System Error
- An explanation of the message with further information.

- The response that is required from the user. In some cases, particularly for information messages, the response might be "none".

## AMQ message example

```
C:\>mqrc AMQ5005
536901397 0x20005005 zrcX_PLUG_UNEXCEPTED
MESSAGE:
Unexpected error
EXPLANATION:
An unexpected error occurred in an internal function of the
product.
ACTION:
Save any generated output files and use either the MQ
Support site:
//www.ibm.com/software/integration/wmq/support/ or IBM
Support Assistant (ISA):
http://www.ibm.com/software/support/isa/, to see whether a
solution is already available. If you are unable to find a
match, contact your IBM support center
C:>
```

© Copyright IBM Corporation 2014

Figure 8-17. AMQ message example

WM2091.0

### Notes:

This figure shows the example of an AMQ message that is displayed by running the `mqrc` control command.

The message includes the explanation of the message and an action to take.



## Application activity trace

- Can be used to:
  - Track messages
  - Analyze or model application behavior and outage impact
  - Audit options that applications use
- Generation of detailed MQI activity
  - If configured, each MQI operation produces an activity record
  - Multiple records can be bundled into an activity trace message that is based on time or record count thresholds
  - Activity records are grouped PCF format
- Detail of activity records can be adjusted
  - ACTVTRC and ACTVCONO queue manager properties
  - Detailed settings in `mqat.ini`
  - IBM MQ Explorer queue manager **Online monitoring** properties
- Sample program (**amqsact**) provided to format PCF activity trace data into human readable form

© Copyright IBM Corporation 2014

Figure 8-18. Application activity trace

WM2091.0

### Notes:

An application activity trace reports on all the MQI operations from an application. The advantage of an activity trace is that applications and their relationships and resources can be analyzed without inspecting the application source code.

The application activity trace messages are sent to the `SYSTEM.ADMIN TRACE.ACTIVITY.QUEUE` queue. Like other events, it is possible to redefine the event queue to be a topic alias so that multiple consumers can work with these messages.

Activity trace runs “inside” the queue manager so it has access to more than just the MQI parameters that the application passes. Other information that is reported includes the queues that the application uses.

The output of the activity trace is similar to many other reports. It is a set of PCF events, where each event holds details about multiple MQI calls.

You can specify whether activity trace information is collected by setting the queue manager property `ACTVTRC` to `ON`. The `ACTVCONO` queue manager property specifies whether applications can override the settings of the `ACTVTRC` queue manager property by using the MQI connection options.

You can also set the activity trace properties and enable an activity trace in MQ Explorer and in the **mqat.ini** file.

## mqat.ini (1 of 2)

```

#*****
#* Module Name: mqat.ini
#* Type       : WebSphere MQ queue manager configuration file
#* Function   : Define the configuration of application activity
#*               trace for a single queue manager.
#*
#*****
AllActivityTrace:                      # Global settings stanza
    ActivityInterval=1                  # Time interval between trace messages
        # Values: 0-99999999 (0=off)
        # Default: 1
    ActivityCount=1000                 # Number of operations between trace msgs
        # Values: 0-99999999 (0=off)
        # Default: 1000
    TraceLevel=MEDIUM                  # Amount of data traced for each operation
        # Values: LOW | MEDIUM | HIGH
        # Default: MEDIUM
    TraceMessageData=0                 # Amount of message data traced
        # Values: 0-104857600
        # Default: 0
    StopOnGetTraceMsg=ON              # Stop trace on get of activity trace message
        # Values: ON | OFF
        # Default: ON
#*****

```

Created in the queue manager data directory when queue manager created with default contents

© Copyright IBM Corporation 2014

Figure 8-19. mqat.ini (1 of 2)

WM2091.0

### Notes:

An initialization file (**mqat.ini**) defines the required granularity for the application activity report. The file is created in the queue manager data directory when the queue manager is created.

You can have reports of all message data for all applications, but that might be excessive.

Changes to the initialization file can be made dynamically without restarting the queue manager; change the queue manager attribute that determines whether to collect these reports.

You can also allow applications to disable their own data collection.

The figure shows the first part of the **mqat.ini** file that contains the configuration information for an activity trace for a single queue manager.

**mqat.ini (2 of 2)**

```
#####
# Prevent the activity trace program from generating data #
#####

ApplicationTrace:          # Application specific settings stanza
ApplClass=ALL               # Application type
#   Values: (USER | MCA | ALL) Default: USER
ApplName=*                  # Application name (may be wildcarded)
#   (matched to app name without path)
#   Default: *
ApplFunction=AMQSACT*       # Application function (may be wildcarded)
#   (matched to app function) Default: *
Trace=OFF                   # Activity trace switch for application
#   Values: ( ON | OFF ) Default: OFF
ActivityInterval=0           # Time interval between trace messages
#   Values: 0-99999999 (0=off) Default: 0
ActivityCount=0              # Number of operations between trace msgs
#   Values: 0-99999999 (0=off) Default: 0
TraceLevel=MEDIUM            # Amount of data traced for each operation
#   Values: LOW | MEDIUM | HIGH
#   Default: MEDIUM
TraceMessageData=0            # Amount of message data traced
#   Values: 0-104857600
#   Default: 0
```

© Copyright IBM Corporation 2014

Figure 8-20. mqat.ini (2 of 2)

WM2091.0

**Notes:**

This figure shows the second part of the application activity report initialization file (**mqat.ini**).

The default values in the application activity report initialization file are set to prevent the activity trace program from generating data.

## Extract from an activity trace

```

MonitoringType: MQI Activity Trace
QueueManager: 'QM1'
Host Name: 'host1.ibm.com'
CommandLevel: 800
ApplicationName: 'WebSphere MQ Client for Java'
ApplicationPid: 18612354
UserId: 'mquser'
ConnName: '9.20.95.106'
Channel Type: MQCHT_SVRCONN
Platform: MQPL_UNIX
=====
Time      Operation  CompCode  MQRC   HObj (ObjName)
10:04:09  MQXF_INQ   MQCC_OK   0000   2
10:04:09  MQXF_CLOSE MQCC_OK   0000   2
10:04:09  MQXF_OPEN  MQCC_OK   0000   4  ()
10:04:09  MQXF_INQ   MQCC_OK   0000   4
10:04:09  MQXF_CLOSE MQCC_OK   0000   4
10:04:09  MQXF_OPEN  MQCC_OK   0000   4  (SYSTEM.DEFAULT.LOCAL.QUEUE)
10:04:09  MQXF_INQ   MQCC_OK   0000   4

```

© Copyright IBM Corporation 2014

Figure 8-21. Extract from an activity trace

WM2091.0

### Notes:

This figure shows part of an application activity trace. It contains the queue manager name, application name, process ID, connection name, operating system, and operations.

The sample program (`amqsact`) formats the events into a usable format. You can use the sample program results as input to more sophisticated tools.

## Tracing IBM MQ components

- Extra information might be needed to find a problem
  - Files can be large
  - Time or component might cause limits
- Can also trace the MQI, which is a useful aid for application debugging
- Trace commands:
  - Start trace: **strmqtrc**
  - Stop trace: **endmqtrc**
  - Format trace: **dspmqtrc**

© Copyright IBM Corporation 2014

Figure 8-22. Tracing IBM MQ components

WM2091.0

### Notes:

IBM Service personnel might ask you to re-create a problem with an MQ component enabled. The files that the component trace produces can be large so it is important to limit a trace by time or trace only specified components.

The control command to start a trace is **strmqtrc**

The control command to end a trace is **endmqtrc**

The control command to format the trace results is **dspmqtrc**

## Trace commands

- Trace files are written to:
  - C:\Program Files\IBM\WebSphere MQ\trace on Windows
  - /var/mqm/trace on UNIX
  - Delete or relocate old trace files before beginning a new trace
  
- Start trace
  - For every IBM MQ process: `strmqtrc -e`
  - For one queue manager: `strmqtrc -m MY.QMGR`
  - High detail trace for one queue manager:  
`strmqtrc -t all -t detail -m MY.QMGR`
  - High detail trace that limits the file size to ~5 MB:  
`strmqtrc -l 5 -t all -t detail -m MY.QMGR`
  
- End all tracing: `endmqtrc -a`
- Format the trace files on UNIX: `dspmqtrc *.TRC`
- Format wrapping trace files: `dspmqtrc *.TRC *.TRS`

© Copyright IBM Corporation 2014

Figure 8-23. Trace commands

WM2091.0

### Notes:

The trace files are written to specific locations on the file system.

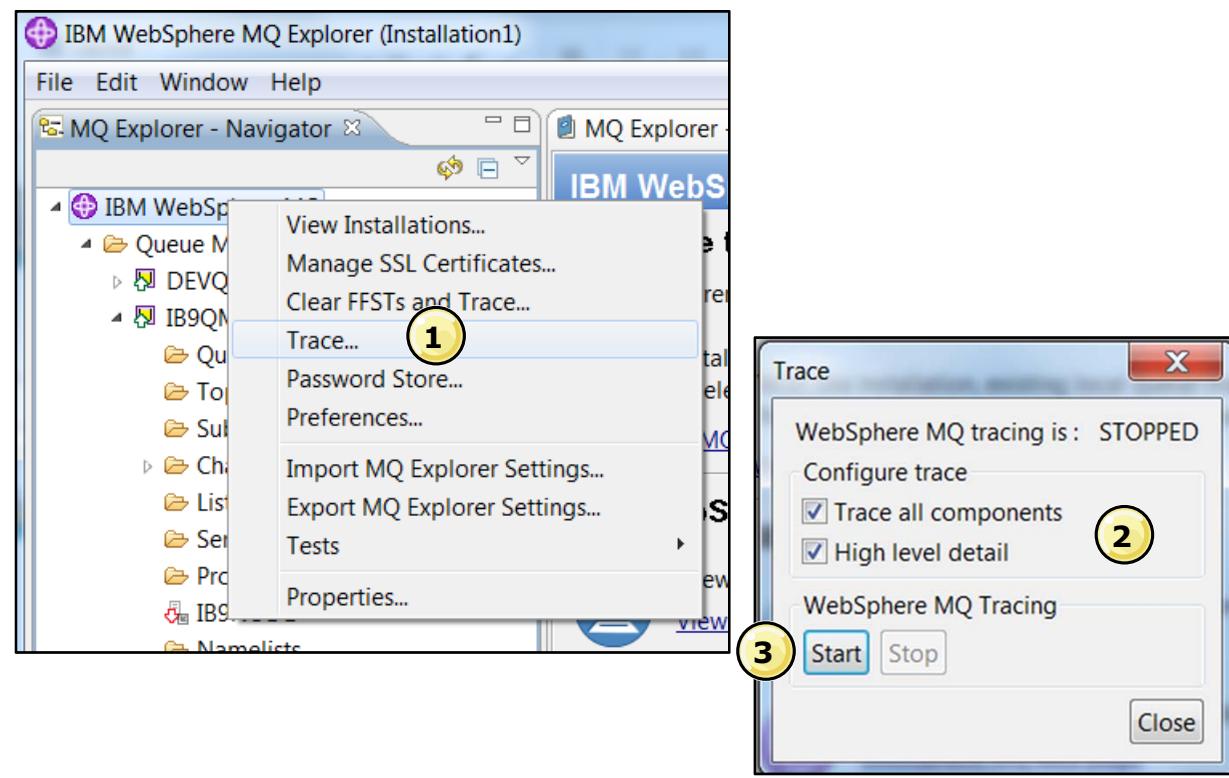
The figure shows some examples of starting an MQ component trace for every process, and for one queue manager. It also shows examples of how you can modify the commands to control the trace detail and limit the size of the trace file.

The trace formatter program, `dspmqtrc`, converts binary files that are named `AMQppppp.TRC` into readable files that are named `AMQppppp.FMT` where `ppppp` is the process identifier that created the file. The trace files are readable without formatting; they are still named `AMQppppp.TRC`.

If you specify a wrapping trace, then each time a trace (.TRC) file reaches the specified size limit, MQ renames it to use a .TRS extension and starts a new trace file. The trace formatter can convert both files to a single formatted file, but only if you format the .TRC and .TRS files at the same time, as shown in the last example in the figure.

 WebSphere Education

## Enabling trace in IBM MQ Explorer



© Copyright IBM Corporation 2014

Figure 8-24. Enabling trace in IBM MQ Explorer

WM2091.0

**Notes:**

You can also enable a component trace in MQ Explorer but the options for the trace are limited to:

- Trace all components
- High-level detail

## Example trace on IBM MQ for Solaris

Timestamp	Process	Thread	Trace Data
15:00:04.324190	12277.1		Version : 6.0.0.0 Level : p000-L050203
15:00:04.325045	12277.1		Date : 07/02/05 Time : 15:00:04
15:00:04.325375	12277.1		PID : 12277 Process : strmqm
15:00:04.325403	12277.1		QueueManager : QM1
15:00:04.325419	12277.1		-----
15:00:04.325446	12277.1		Trace Control Memory:
15:00:04.325471	12277.1		StrucId:
15:00:04.325490	12277.1		EarlyTraceOptions: 0
15:00:04.325507	12277.1		EarlyTraceMaxFileSize: 0
15:00:04.325527	12277.1		ActiveEntries: 0
15:00:04.325544	12277.1		Options MaxFileSize FileCount SubPoolName
15:00:04.325566	12277.1		74fffff 0 0 elk
15:00:04.325587	12277.1		0 0 0
15:00:04.325609	12277.1		0 0 0
15:00:04.325632	12277.1		0 0 0
15:00:04.325654	12277.1		0 0 0
15:00:04.325677	12277.1		0 0 0
15:00:04.325698	12277.1		0 0 0
15:00:04.325774	12277.1		0 0 0
15:00:04.325798	12277.1		0 0 0
15:00:04.325891	12277.1		Thread stack
15:00:04.325971	12277.1		-> zslWaitEC
15:00:04.326078	12277.1		-> zslCheckIfRunning
15:00:04.326098	12277.1		-> xcsInitialize
15:00:04.326147	12277.1		-> xcsGetEnvironmentString
15:00:04.326186	12277.1		---{ xcsGetEnvironmentString
15:00:04.326241	12277.1		xcsGetEnvironmentString[AMQ_SERVICE_MODULE] = NULL

© Copyright IBM Corporation 2014

Figure 8-25. Example trace on IBM MQ for Solaris

WM2091.0

### Notes:

The figure shows an example of a trace on Solaris.

Trace reports are unformatted on UNIX systems. Use the `dspmqtrc` control command to format the trace report before viewing.

For example, to format all trace files in the current directory enter the following command:

```
dspmqtrc *.TRC
```

## Configuration files and problem determination

- Errors can stop a queue manager from being found

Example: **QUEUE MANAGER UNAVAILABLE**

- What to check

- Configuration files exist
- Configuration files have the appropriate permissions
- IBM MQ configuration file (or Windows registry) references the queue manager with the correct information for locating its files

© Copyright IBM Corporation 2014

Figure 8-26. Configuration files and problem determination

WM2091.0

### Notes:

If you receive an error message that indicates that the queue manager is unavailable, the cause might be something simple. For example, the queue manager is not started or an application specified an incorrect queue manager name.

If the cause is not obvious:

- Ensure that the MQconfiguration files exist.
- Ensure that the MQconfiguration files have the appropriate permissions. For example, on a UNIX system, the MQ configuration file should have the following permissions.  
`-rwxrwxr-x 1 mqm mqm 1371 Sep 17 14:32 /var/mqm/mqs.ini`
- Ensure that the MQ configuration file references the queue manager and has the correct information for locating the files that are associated with it.

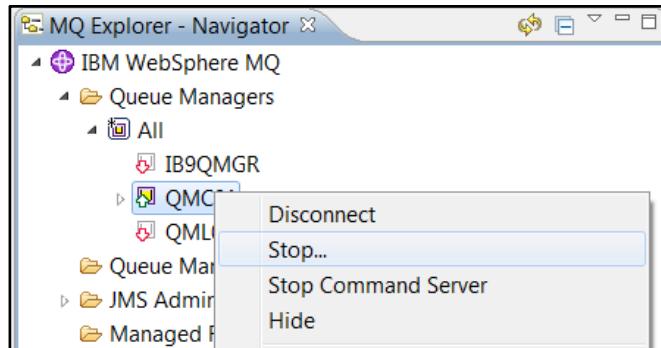
Errors in a configuration file can typically prevent a queue manager from being found.



## Stopping a queue manager manually

- Use `endqm` control command or IBM MQ Explorer

- Might take time
- Give it a chance



- Only if no other option, find and stop the processes that are still running
  - On Windows, use Windows Task Manager
  - On UNIX, enter the following command to locate all processes:

```
ps -ef | grep amq
```

© Copyright IBM Corporation 2014

Figure 8-27. Stopping a queue manager manually

WM2091.0

### Notes:

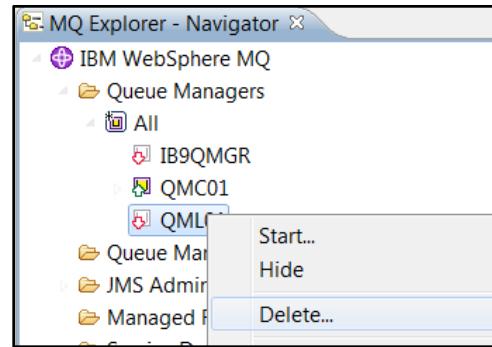
During normal system administration, it might become necessary to stop the queue manager. You can stop the queue manager by using the `endqm` control command or in the MQ Explorer.

If, for some reason, the queue manager does not stop, it might be necessary to force the queue manager to shut down by stopping the queue manager process.

- On Windows, you can use the Task Manager to find and then shutdown the process.
- On UNIX, use the `ps -ef` command with `grep` to get a list all the MQ processes and then shutdown the process with the UNIX `kill` command.

## Removing a queue manager manually

- Use `dltmqm QMgrName` control command or IBM MQ Explorer
- If there are problems:
  - Delete queue manager directory and its contents
  - Delete associated log directory and its contents
  - Remove queue manager's stanza from IBM MQ configuration file
  - Remove or change `DefaultQueueManager` stanza if required
- IBM MQ for Windows requires manual changes in Windows Registry



© Copyright IBM Corporation 2014

Figure 8-28. Removing a queue manager manually

WM2091.0

### Notes:

After you stop the queue manager, you can delete it by using the `dltmqm` control command or by using the MQ Explorer.

Deleting a queue manager is a drastic step because you also delete all resources that are associated with the queue manager, including all queues and their messages and all object definitions.

If you use the `dltmqm` command, a confirmation prompt is not displayed; when you press the Enter key all the associated resources are lost.

If you cannot delete the queue manager by using the control command or MQ Explorer, you can delete the queue manager artifacts manually, as described in the figure.

## Channel problem determination

- Verify that the listener program is running on the receiving end of the channel.
  - For `/etc/services`, and `inetd.conf`, check the configuration
- Verify that the queue manager is running.
- Use the TCP/IP `ping` command to verify the connection from sender to receiver.
- Channels come in pairs and they must both have the same name, and are case-sensitive.
- Check error logs on the sending and receiving side of the channel.
- Use MQSC commands on the sending side of the problem channel to correct several channel-related problems.

`STOP CHANNEL (ChannelName)`

`RESET CHANNEL (ChannelName) SEQNUM(1)`

`RESOLVE CHANNEL (ChannelName) ACTION(BACKOUT | COMMIT)`

`START CHANNEL (ChannelName)`

© Copyright IBM Corporation 2014

Figure 8-29. Channel problem determination

WM2091.0

### Notes:

The figure lists some tips for locating problems with channels:

- Verify that the listener program is running on the sending and receiving end of the channel. Some environments use `/etc/services`, and `inetd.conf` instead of the MQ `runmq1sr` program. You must verify that the files are configured correctly. Details regarding the configuration are provided in the *IBM MQ Intercommunication Guide*.
- Verify that the queue manager is running.
- Use a TCP/IP `ping` command to verify the connection from sender to receiver.
- Ensure that the channel pairs have the same name.
- Check the error logs on the sending and receiving channel.

Several MQSC commands are listed in the figure that you can use on the sending side of the channel to correct some channel-related problems.

## Remote administration failures

- AMQ4043 Queue manager not available for connection is generated for one of the following reasons:
  - Listener is not running on the remote queue manager
  - Code page conversion failure
  - Security check failure
- Using IBM MQ Explorer or MQSC commands
  - Check that the IBM MQ command server is running on the remote queue manager.
  - Check that the SYSTEM.ADMIN.SVRCONN channel is defined on the remote queue manager.

© Copyright IBM Corporation 2014

Figure 8-30. Remote administration failures

WM2091.0

### Notes:

This figure lists the common causes of remote administration failures. Remote administration is described in detail in Unit 9, *Implementing distributed queuing*.

Ensure that the following requirements are satisfied before trying to use the MQ Explorer for remote administration. Verify that:

- The MQ server and client are installed on the local and the remote computer.
- A command server is running for every queue manager.
- A TCP/IP listener exists for every queue manager. It can be the MQ listener or the InetD daemon as appropriate for your operating system environment.
- The server-connection channel that is called SYSTEM.ADMIN.SVRCONN, exists on every remote queue manager. This channel is mandatory for every administered remote queue manager.
- The user ID of the initiator must be a member of the "mqm" group on the local and remote computer. The model queue SYSTEM.MQEXPLORER.REPLY.MODEL must exist on every queue manager.

## Channel triggering problems

- Verify that the channel initiator is running.
- Use the **runmqchi** control command to run a channel initiator process.  
Example: **runmqchi -q SYSTEM.CHANNEL.INITQ -M CLIENT**
- Make sure that the channel initiator is monitoring the initiation queue, not the transmission queue.
- Check the error log for channel error messages.
- Try to start the channel manually.
  - If the channel fails to start, or does not successfully move the message from the transmission queue to the remote queue manager, then this problem is a channel problem.

© Copyright IBM Corporation 2014

Figure 8-31. Channel triggering problems

WM2091.0

### Notes:

You can make various checks if you encounter problems with triggering:

- Verify that the channel initiator is running.
- Use the **runmqchi** command to run the channel initiator process.
- Make sure that the channel initiator is monitoring the initiation queue, not the transmission queue
- Check the error log for channel error messages.

When debugging a program with triggering a channel, set a short disconnect interval on the associated channel. The disconnect interval setting stops the channel quickly, with triggering enabled, and makes debugging easier.



## Unit summary

Having completed this unit, you should be able to:

- Determine the possible causes and locations of a missing message
- Analyze the error logs that IBM MQ generates
- Locate First Failure Support Technology (FFST) files on a system
- Use an IBM MQ trace to collect detailed information about IBM MQ operation
- Describe some common problem types and how to approach initial problem determination
- Stop and remove a queue manager manually

© Copyright IBM Corporation 2014

---

Figure 8-32. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. Why might a message be delayed on a transmission queue?
  - a. Queue Manager is at MAXCHL
  - b. Transmission queue is full
  - c. Transmission queue is GET(DISABLED)
  - d. Channel initiator is not running
2. True or False: Persistent messages do not expire.
3. True or False: An uncommitted message can be browsed but not destructively removed.
4. True or False: Applications must handle activity reports.

© Copyright IBM Corporation 2014

Figure 8-33. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.



## Checkpoint answers

1. **a, c, and d.** A full transmission queue returns an MQRC=2053 to the application that performs the MQPUT or MQPUB.
2. **False.** An application can assign an expiration time for persistent messages.
3. **False.** Uncommitted messages are unavailable to any application outside the current unit of work.
4. **False.** Activity reporting is optional.

© Copyright IBM Corporation 2014

Figure 8-34. Checkpoint answers

WM2091.0

### Notes:

## Exercise 6



Running an IBM MQ trace

© Copyright IBM Corporation 2014  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Figure 8-35. Exercise 6

WM2091.0

### Notes:

In this exercise, you configure an MQ trace to monitor activity on a particular queue. You use the `amqspput` and `amqsget` sample programs to send and receive messages to that queue. Finally, you examine the trace report.



## Exercise objectives

After completing this exercise, you should be able to:

- Set up an IBM MQ trace
- Analyze the output from a trace

© Copyright IBM Corporation 2014

---

Figure 8-36. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for detailed instructions.

# Unit 9. Implementing distributed queuing

## What this unit is about

This unit covers the concepts of interconnecting two or more queue managers in a network, allowing the distribution of messages between systems. Channels and the supporting objects that are needed for them to function correctly are explained. Remote queue definitions are explained.

## What you should be able to do

After completing this unit, you should be able to:

- Diagram the connection between two queue managers by using the required components
- Configure IBM MQ channels
- Start and stop channels
- Identify channel states
- Access remote queues
- List considerations for data conversion
- Use the dead-letter queue to find messages that could not be delivered

## How you will check your progress

- Checkpoints
- Hands-on exercises

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Diagram the connection between two queue managers by using the required components
- Configure IBM MQ channels
- Start and stop channels
- Identify channel states
- Access remote queues
- List considerations for data conversion
- Use the dead-letter queue to find messages that could not be delivered

© Copyright IBM Corporation 2014

Figure 9-1. Unit objectives

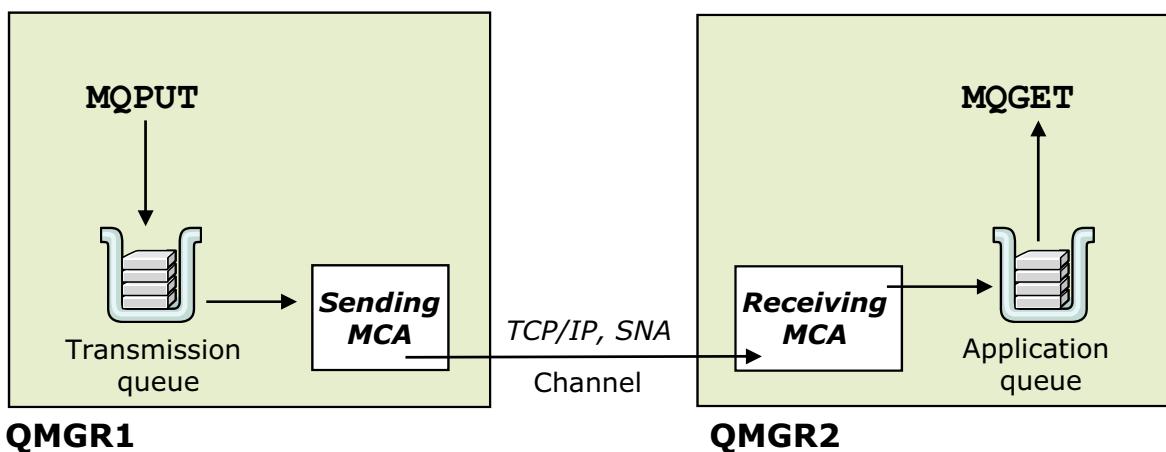
WM2091.0

### Notes:

This unit is, perhaps, one of the most important in the entire course. Channel configuration and operations can be confusing. After you complete this unit and the practical exercise, you should have a better idea of what is involved.

With more practice and use when you return to your job, you can find that they are not as difficult as they might seem.

## Message channel



- One-way link that connects two queue managers by using a *message channel agent* (MCA)

© Copyright IBM Corporation 2014

Figure 9-2. Message channel

WM2091.0

### Notes:

Channels are objects that provide a communication path from one queue manager to another. A *message channel* is a one-way link between two queue managers for the transmission of messages. As shown in the figure, it consists of a message channel agent (MCA) at the sending end, an MCA at the receiving end, and a communications connection between the two. The communications connection might be an SNA LU6.2 conversation or a TCP connection, for example.

Each end of a message channel has a separate definition. Both definitions contain the name of the message channel. Among other things, the definition at each end of a message channel also indicates:

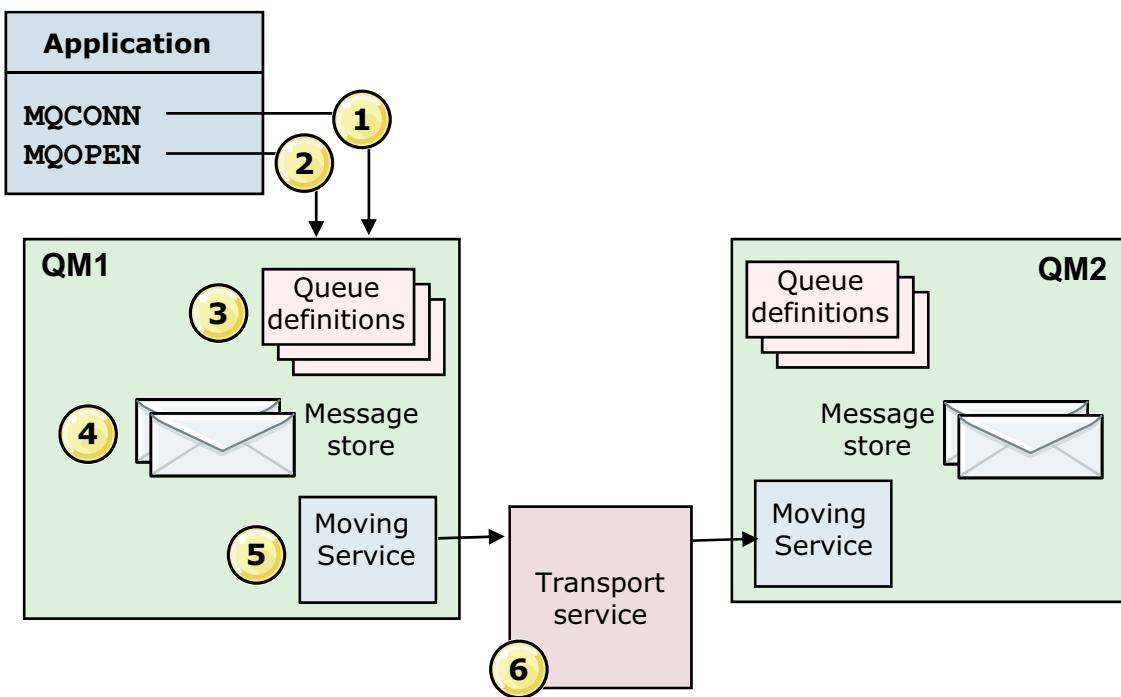
- Whether it is the sending end or the receiving end of the channel
- The communications protocol to use

A *transmission queue* is required for each message channel. A transmission queue is a local queue that has a usage attribute whose value indicates its special role to transmit.

A transmission queue is at the sending end of a message channel. As a result, only the definition of the message channel at the sending end contains the name of the transmission queue.

The queue manager puts any message for a remote queue onto a transmission queue. But how does the queue manager know which transmission queue to use? One way is to give the transmission queue the same name as the name of the destination queue manager.

## Distributed component overview



© Copyright IBM Corporation 2014

Figure 9-3. Distributed component overview

WM2091.0

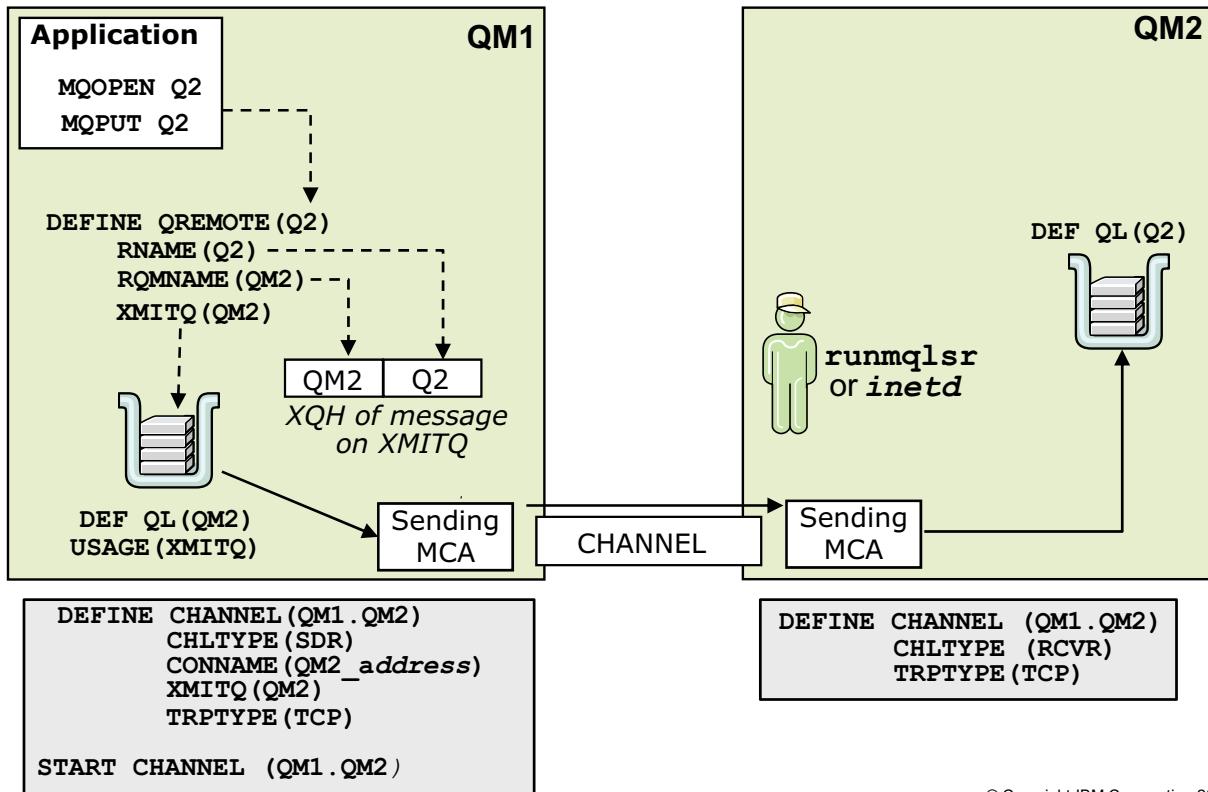
### Notes:

The figure shows the high-level components of distributed queuing. In the figure, an MQ queue manager (QM1) is connected to another queue manager (QM2) so that resources and messages can be shared. An application connects to a queue manager and opens a queue to transmit messages to another queue manager.

1. An application sends an MQCONN to connect to a queue manager.
2. The application sends an MQOPEN to open a queue so that it can put messages on it.
3. The queue manager has a definition for each of its queues, specifying information such as the maximum number of messages that are allowed on the queue.
4. If the messages are destined for a queue on a remote system, the local queue manager holds them in a message store until it is ready to forward them to the remote queue manager. This process might not be apparent to the application.
5. Each queue manager contains communication software that is called the *moving service* component by which the queue manager can communicate with other queue managers.

6. The *transport service* connects to the target queue manager and opens a queue to transmit the messages. The transport service is independent of the queue manager and can be any one of the following methods (depending on the operating system):
  - Transmission Control Protocol/Internet Protocol (TCP/IP)
  - Network Basic Input/Output System (NetBIOS)
  - Sequenced Packet Exchange (SPX)
  - Systems Network Architecture (SNA)

## Distributed queuing overview



© Copyright IBM Corporation 2014

Figure 9-4. Distributed queuing overview

WM2091.0

### Notes:

Whether an application is putting a message on a local queue or to a remote queue, is not apparent to the application. However, an application always gets a message from a local queue.

All queue managers use a common protocol for assured message delivery. A message is not lost in the event of a communications or system failure if the message is persistent, nor is it ever delivered twice.

A message that is destined for a remote queue manager is stored locally on a transmission queue until the MCA can send it.

In the example, a remote queue and transmission queue are defined on queue manager QM1. Channels are defined on both the sender queue manager, QM1, and the receiving queue manager, QM2.

## Distributed queuing components

- Queue that is identified by:
  - Name of the queue
  - Name of the queue manager that owns the queue
- Message channels carry messages from one queue manager to another
- Message channel agents (MCAs)
  - Control the sending and receiving of messages
  - One at each end of the channel
- Transmission queues store messages for a remote queue manager
- Channel initiators and listeners act as trigger monitors for the sender channels
- Channel-exit programs

© Copyright IBM Corporation 2014

Figure 9-5. Distributed queuing components

WM2091.0

### Notes:

Message channels carry messages from one queue manager to another. The definition of each end of a message channel can be sender, receiver, server, requester, cluster sender, or cluster receiver.

A message channel is defined by using the message channel type that is defined at one end, and a compatible type at the other end. Possible combinations are: sender-receiver, requester-server, requester-sender (callback), server-receiver, and cluster sender-cluster receiver.

One MCA takes messages from the transmission queue and puts them on the communication link. The other MCA receives messages and delivers them onto a queue on the remote queue manager.

A message channel agent is a *caller MCA* if it initiated the communication; otherwise it is a *responder MCA*. A caller MCA can be associated with a sender, cluster-sender, server (fully qualified), or requester channel. A responder MCA can be associated with any type of message channel, except a cluster sender.

A *transmission queue* is a special type of local queue stores messages before the MCA transmits them to the remote queue manager. In a distributed-queuing environment, define one transmission queue for each sending MCA, unless you are using MQ queue manager clusters.

You specify the name of the transmission queue in a remote queue definition. If you do not specify the name, the queue manager looks for a transmission queue with the same name as the remote queue manager.

Optionally, you can specify the name of a default transmission queue for the queue manager. This name is used if you do not specify the name of the transmission queue, and a transmission queue with the same name as the remote queue manager does not exist.

A *channel initiator* acts as a trigger monitor for sender channels because a transmission queue might be defined as a triggered queue. When a message arrives on a transmission queue that satisfies the triggering criteria for that queue, a message is sent to the initiation queue, triggering the channel initiator to start the appropriate sender channel.

A channel listener program listens for incoming network requests and starts the appropriate receiver channel when it is needed.

MQ calls channel-exit programs at defined places in the processing carried out by the MCA.

## Basic channel types

- For distributed queuing:
  - Sender
  - Server
  - Receiver
  - Requester
- For clustered environments
  - Cluster-sender
  - Cluster-receiver
- For IBM MQ clients
  - Client-connection
  - Server-connection

© Copyright IBM Corporation 2014

Figure 9-6. Basic channel types

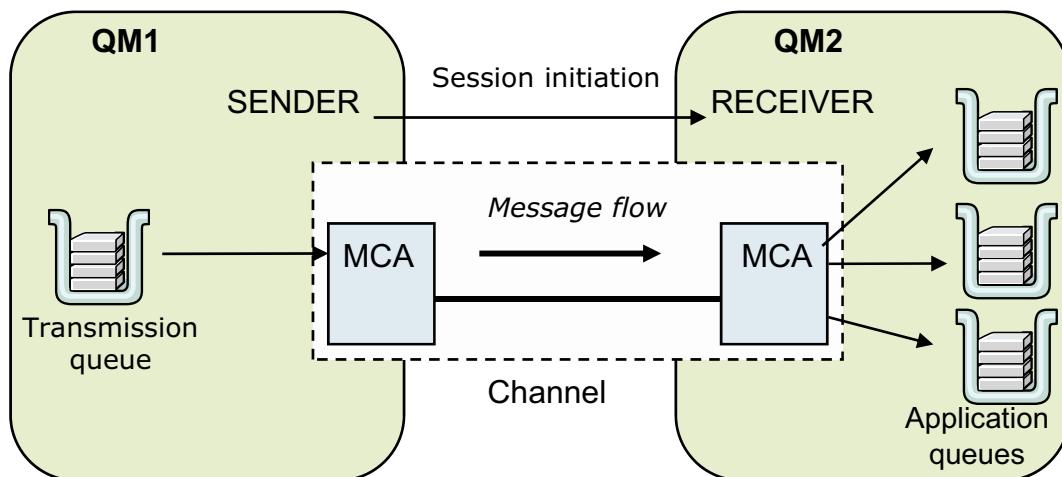
WM2091.0

### Notes:

The figure lists the basic channel types for distributed queuing, clustered environments, and MQ clients.

Clustering is not covered in this course.

## Sender-receiver channels



© Copyright IBM Corporation 2014

Figure 9-7. Sender-receiver channels

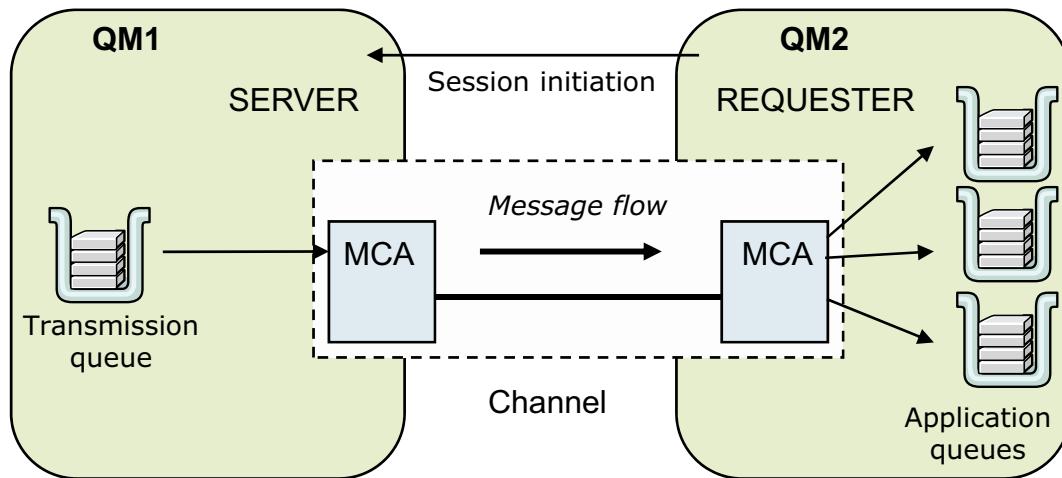
WM2091.0

### Notes:

With a sender-receiver channel, a sender in one system starts the channel so that it can send messages to the other system. The sender requests the receiver at the other end of the channel to start. The sender sends messages from its transmission queue to the receiver. The receiver puts the messages on the destination queue.

A server-receiver channel is similar to a sender-receiver channel except that it applies only to *fully qualified* servers. A fully qualified server has server channels that have the connection name of the server that is specified in the channel definition. Channel startup must be initiated at the server end of the link.

## Requester-server channels



© Copyright IBM Corporation 2014

Figure 9-8. Requester-server channels

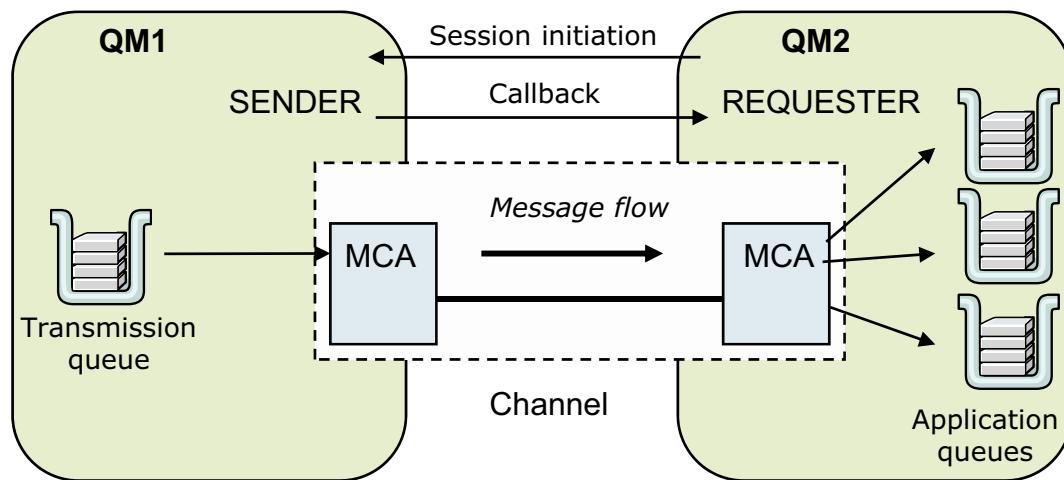
WM2091.0

### Notes:

A requester in one system starts the channel so that it can receive messages from the other system. The requester requests the server at the other end of the channel to start. The server sends messages to the requester from the transmission queue that is defined in its channel definition.

A server channel can also initiate the communication and send messages to a requester, but this initiation applies only to *fully qualified servers*. A requester can start a fully qualified server, or a fully qualified server can initiate a communication with a requester.

## Requester-sender channels



© Copyright IBM Corporation 2014

Figure 9-9. Requester-sender channels

WM2091.0

### Notes:

The requester starts the channel and the sender ends the call. The sender then restarts the communication according to information in its channel definition (known as *callback*). It sends messages from the transmission queue to the destination queue of the requester.

## Choosing a transmission queue

- Specify the transmission queue in local definition of a remote queue.

```
DEFINE QREMOTE(BBB) RNAME(YYY) +
RQMNAME(QM2) XMITQ(EXPRESS)
```

- The name of the transmission queue is inferred from the name of the remote queue manager.
- Specify the default transmission queue attribute on the queue manager.
- ALTER QMGR DEFXMITQ(HOST)**
- Error (the MQOPEN fails).

© Copyright IBM Corporation 2014

Figure 9-10. Choosing a transmission queue

WM2091.0

### Notes:

A queue manager attempts to apply one of the following rules, in the stated order, to determine which transmission queue to use:

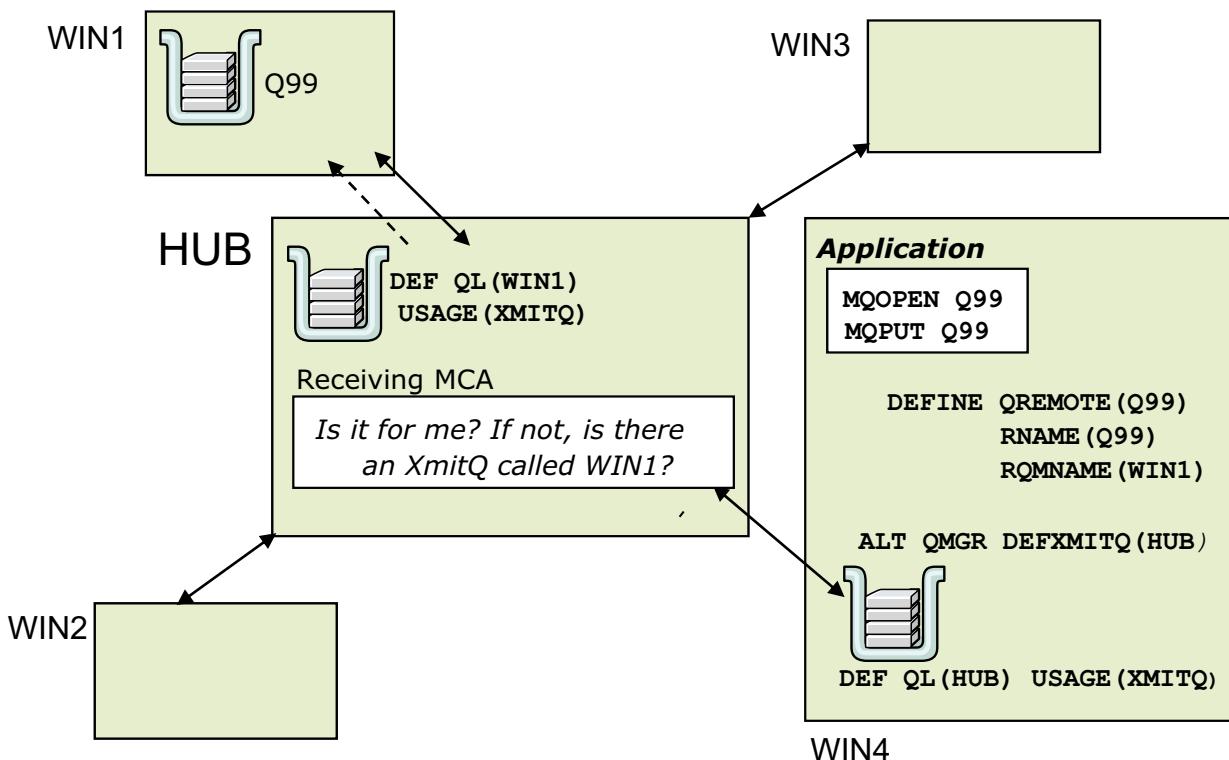
- Use the transmission queue that is named explicitly in a local definition of a remote queue.  
An example MQSC command that creates a transmission queue that is named EXPRESS is:  
**DEF QR(BBB) RNAME(YYY) RQMNAME(QM2) XMITQ(EXPRESS)**
- Use the transmission queue with the same name as the remote queue manager.
- Use the default transmission queue. The default transmission queue is used when no other suitable transmission queue is available for sending messages to another queue manager. If you define a default transmission queue, you must also define a channel to serve the queue. If you do not do this, messages that are put on to the default transmission queue are not transmitted to the remote queue manager and remain on the queue.

An example command for defining a default transmission queue is:

**ALTER QMGR DEFXMITQ(HOST)**

- An error is reported if the queue manager cannot find a transmission queue by attempting to apply the rules.

## Example of using a default transmission queue



© Copyright IBM Corporation 2014

Figure 9-11. Example of using a default transmission queue

WM2091.0

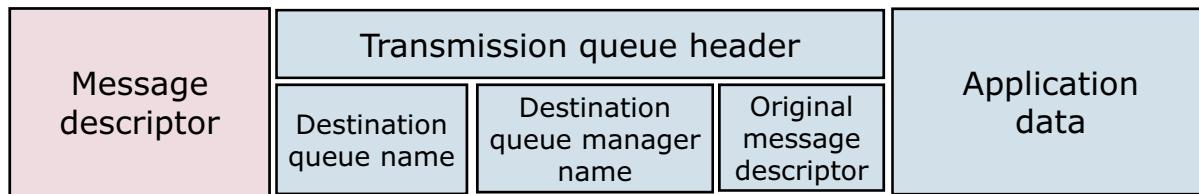
### Notes:

The default transmission queue is the destination for messages that are sent to a remote queue when no other suitable transmission queue is defined. Define it as a local queue, and that queue name is placed in the queue manager parameter **DEFXMITQ**.

Default transmission queues are useful in a hub and spoke network environment. In a hub and spoke environment, each of the spoke queue managers is directly connected only to the hub queue manager. Defining the transmission queue to the hub as the default transmission queue for each spoke queue manager is a logical network design.

The figure shows an example of a hub and spoke design.

## Transmission queue header



© Copyright IBM Corporation 2014

Figure 9-12. Transmission queue header

WM2091.0

### Notes:

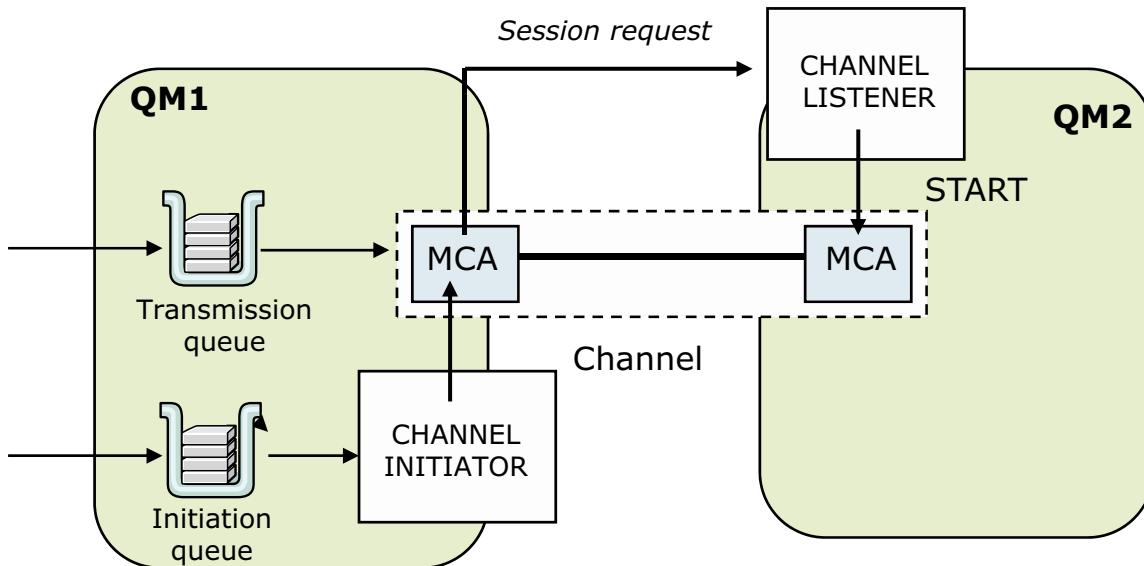
The MQ transmission queue header (MQXQH) structure describes the information that is prefixed to the application message data of messages when they are on transmission queues.

Data in the MQXQH must be in the character set and encoding of the local queue manager.

In addition to a structure identifier and version number, the MQXQH also contains the name of the destination queue, the name of the destination queue manager, and the original message descriptor.

The queue manager generates the message descriptor in the header when the message is placed on the transmission queue. Some of the fields in the separate message descriptor are copied from the message descriptor that the application provides on the MQPUT or MQPUT1 call. The separate message descriptor is the one that is returned to the application when the message is removed from the transmission queue.

## Channel initiators and listeners



© Copyright IBM Corporation 2014

Figure 9-13. Channel initiators and listeners

WM2091.0

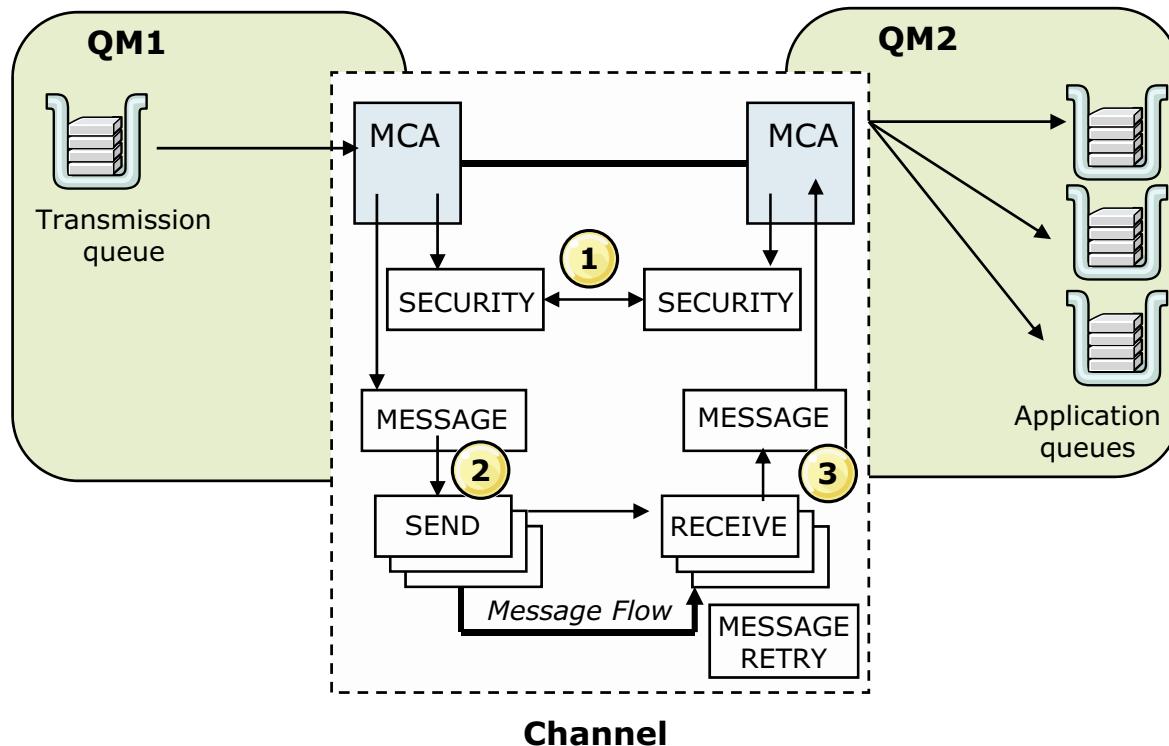
### Notes:

The figure shows a message flow that uses a channel initiator and a listener. A message is put on the initiation queue of QM1. The message triggers the channel initiator for that queue manager that starts the sender channel. The channel listener at QM2 receives the session request and starts the receiving MCA. Messages can then be transmitted from QM1 to QM2.

A channel initiator acts as a trigger monitor for sender channels because a transmission queue can be defined as a triggered queue. When a message arrives on a transmission queue that satisfies the triggering criteria for that queue, a message is sent to the initiation queue, triggering the channel initiator to start the appropriate sender channel. Server channels can also be started in this way if the connection name of the server that is specified in the channel definition. Channels can be started automatically, based on messages that arrive on the appropriate transmission queue.

You need a *channel listener* to start receiving (responder) MCAs. Responder MCAs are started in response to a startup request from the caller MCA; the channel listener detects incoming network requests and starts the associated channel.

## Channel-exit programs



© Copyright IBM Corporation 2014

Figure 9-14. Channel-exit programs

WM2091.0

### Notes:

MQ calls channel-exit programs at defined places in the processing carried out by the MCA. There are four types of channel exits:

- Security exit for security checking, such as authentication of the remote server.
- Message exit for operations on the message, for example, encryption before transmission.
- Send and Receive exits for operations on split messages, for example, data compression and decompression.
- Message-retry exit that is used when a problem arises in putting the message to the destination.

The sequence of processing for channel-exit programs is shown in the figure:

1. Security exits are called after the initial data negotiation between both ends of the channel. Security exits must end successfully for the startup phase to complete and to allow messages to be transferred.
2. The sending MCA calls the Message exit, and then the Send exit is called for each part of the message that is transmitted to the receiving MCA.

3. The receiving MCA calls the Receive exit when it receives each part of the message, and then calls the Message exit when the whole message is received.

Before writing a channel exit, consider other MQ capabilities such as SSL for encryption, or CHLAUTH or CONNAUTH, or both, for authentication.

Channel exits are described in detail in WM212, *IBM MQ V8 Advanced System Administration*.

## Configuration file stanzas for distributed queuing

- Stanzas in the queue manager configuration file (`qm.ini`) for distributed queuing
  - CHANNELS
  - TCP
  - LU62
  - NETBIOS
  - SPX (Windows XP Professional and Windows 2003 Server only)
  - EXITPATH
- Edit configuration files either:
  - Automatically, by using commands or IBM MQ Explorer options that change the configuration of queue managers
  - Manually, by using a text editor

© Copyright IBM Corporation 2014

Figure 9-15. Configuration file stanzas for distributed queuing

WM2091.0

### Notes:

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager. The `qm.ini` file is automatically created when the queue manager with which it is associated is created.

On UNIX and Linux, a `qm.ini` file is held in the root of the directory tree that is occupied by the queue manager. For example, the path and the name for a configuration file for a queue manager that is called QMNAME is: `/var/mqm/qmgrs/QMNAME/qm.ini`

On Windows, the **WorkPath** specified in the `HKEY_LOCAL_MACHINE\Software\IBM\WebSphere MQ` key gives the location of the `qm.ini` file. For example, the path and the name for a configuration file for a queue manager that is called QMNAME is: `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\qm.ini`

A configuration file contains one or more *stanzas*, which are groups of lines in the file that have a common function or define part of a system, such as log functions, channel functions, and installable services. The figure lists the stanzas in the queue manager configuration file for distributed queuing.

## Example queue manager configuration file stanzas

**CHANNELS:**

```
MaxChannels=n      ; Maximum channels allowed. Default is 100
MaxActiveChannels=n ; Maximum active channels allowed at any time.
                     ; Default is the value of MaxChannels
MaxInitiators=n   ; Maximum initiators allowed. Default and maximum
                     ; value is 3
MQIBINDTYPE=type1 ; Whether the binding for applications is "fastpath"
                     ; or "standard". Default is "standard".
ThreadedListener=  ; "YES" to run all channels run as threads of a single
                     ; job. "NO" to start a new responder job for each
                     ; inbound TCP/IP channel. Default is "NO".
AdoptNewMca=chltype ; Stops previous process if channel fails to start.
                     ; The default is "NO".
AdoptNewMcaTimeout= ; Amount of time that the new process waits for the
                     ; old process to end. Default is 60.
AdoptNewMcaCheck=  ; Type of checking required when enabling AdoptNewMca.
                     ; typecheck ; Default is "NAME", "ADDRESS", and "QM".
```

**TCP:**

```
Port=n            ; TCP entries
KeepAlive=NO      ; Port number, the default is 1414
                   ; Switch TCP/IP KeepAlive "ON" or "OFF"
ListenerBacklog=n ; Maximum outstanding connection requests.
ConnectTimeout=n  ; Seconds before an attempt to connect socket times
                   ; out. Default of zero specifies that there is no
                   ; connect timeout.
```

© Copyright IBM Corporation 2014

Figure 9-16. Example queue manager configuration file stanzas

WM2091.0

### Notes:

This figure shows the CHANNELS and TCP stanzas in the queue manager configuration file.

Use these queue manager properties pages in MQ Explorer, MQ commands, or stanzas in the queue manager configuration file to specify channel and network protocol configuration parameters.



#### Note

Only attributes that represent changes to the default values need to be specified in the queue manager configuration file.

For more information about the queue manager configuration file, see the IBM MQ product documentation.

## Minimum channel definitions for remote communication

- On the source queue manager, channel type of SENDER
  - XMITQ specifies the name of the transmission queue
  - CONNAME defines the connection name of the partner system
  - TRPTYPE specifies the transport protocol (default is ‘TCP’)
- On the target queue manager, channel type RECEIVER
  - Same name as sender channel
  - TRPTYPE to specify the transport protocol (default is ‘TCP’)
- Use the `ping` command to test the channel

© Copyright IBM Corporation 2014

Figure 9-17. Minimum channel definitions for remote communication

WM2091.0

### Notes:

To send messages from one queue manager to another, you must define two channels; one on the source queue manager and one on the target queue manager.

On the source queue manager, define a channel with a channel type of SENDER. You must specify:

- The name of the transmission queue to use (the XMITQ attribute)
- The connection name of the remote system (the CONNAME attribute)
- The communication protocol type (the TRPTYPE attribute)

On the target queue manager, define a channel with a channel type of RECEIVER, and the same name as the sender channel. Also, specify the communication protocol on the channel (the TRPTYPE attribute).

The receiver channel definitions can be generic. Generic channels mean that if you have several queue managers that communicate with the same receiver, the sending channels can all specify the same name for the receiver, and one receiver definition applies to them all.

## Define channel command

```
DEF CHL(`channel') CHLTYPE(string) CONNAME(string) +
TRPTYPE(string) XMITQ(string)
```

- Required for definition

**(channel)** Channel name up to 20 characters. Must be the first parameter.

**CHLTYPE (string)** Channel type  
 Sender: **SDR**  
 Receiver: **RCVR**  
 Server: **SVR**  
 Requester: **RQSTR**

**CONNAME (string)** Communication connection identifier for **SDR** and **RQSTR**, optional for **SVR**

**TRPTYPE** Transport type: **LU62**, **NETBIOS**, **SPX**, **TCP** (default)

**XMITQ (string)** Name of the transmission queue from which messages are retrieved for **SDR** and **SVR**

© Copyright IBM Corporation 2014

Figure 9-18. Define channel command

WM2091.0

### Notes:

The MQSC command to define a message channel at one end is **DEFINE CHANNEL**. Related commands are **ALTER CHANNEL**, **DISPLAY CHANNEL**, and **DELETE CHANNEL**.

Attributes not supplied on the **DEFINE CHANNEL** command are taken from the appropriate default channel object:

- SYSTEM.DEF.SENDER
- SYSTEM.DEF.RECEIVER
- SYSTEM.DEF.SERVER
- SYSTEM.DEF.REQUESTER

The rules for naming a channel are the same as the rules for naming a queue except that the name of a channel is limited to 20 characters. The channel definition at each end of a channel must specify the same channel name.

The **CHLTYPE** parameter must be included as the first parameter on both the **DEFINE CHANNEL** and **ALTER CHANNEL** commands.

The value of the **CONNNAME** parameter depends on the communication protocol that is used and, in some cases, on the operating system. For TCP/IP, it might be the IP address or the host name of the system on which the remote queue manager is running.

## Defining channels example

### Hursley

```
DEF CHL('Hursley_Dallas') +
CHLTYPE(SDR) TRPTYPE(TCP) +
CONNNAME('9.84.100.1(1414)') +
XMITQ('Dallas')

DEF QL('Dallas') USAGE(XMITQ)

-----
DEF CHL('Dallas_Hursley') +
CHLTYPE(RCVR) TRPTYPE(TCP)
```

### Dallas

```
DEF CHL('Hursley_Dallas') +
CHLTYPE(RCVR) TRPTYPE(TCP)

-----
DEF CHL('Dallas_Hursley') +
CHLTYPE(SDR) TRPTYPE(TCP) +
CONNNAME('9.20.31.5(1414)') +
XMITQ('Hursley')

DEF QL('Hursley') USAGE(XMITQ)
```

© Copyright IBM Corporation 2014

Figure 9-19. Defining channels example

WM2091.0

### Notes:

Each message channel has a two channel definitions, one at each end.

In the example, the sending end of each channel has a transmission queue with the same name as the queue manager at the other end of the channel.

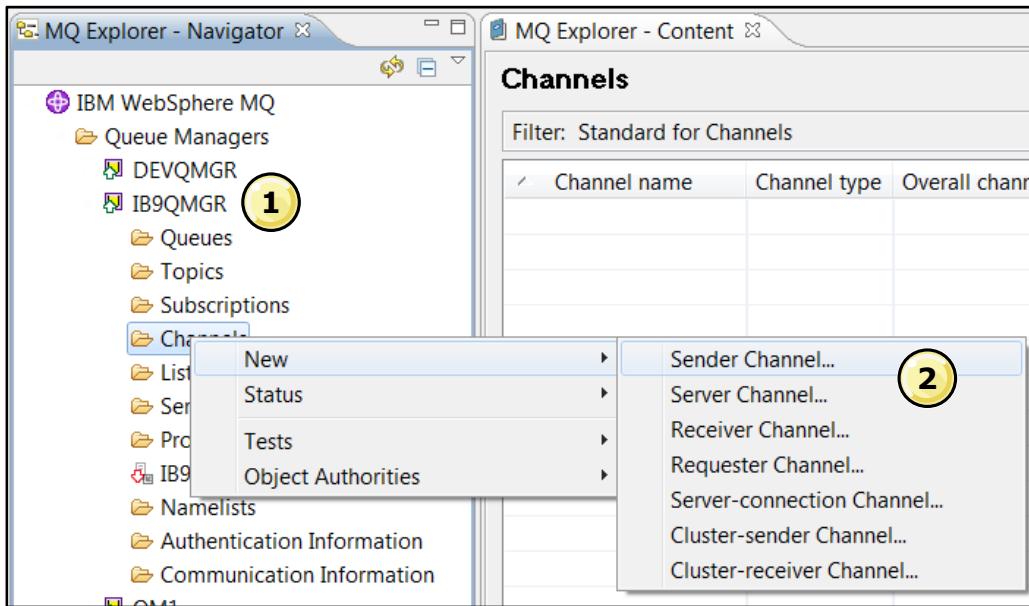
The definition of a channel at each end of the channel must specify the same channel name. The example follows the convention that the name of a transmission queue is the same as the name of the queue manager at the other end channel.

Most users follow the convention that is depicted in the example for naming a channel or adopt some minor variation of it.

For TCP/IP, you can use either the IP address or the host name on the CONNAME parameter.



## Defining channels in IBM MQ Explorer (1 of 2)



© Copyright IBM Corporation 2014

Figure 9-20. Defining channels in IBM MQ Explorer (1 of 2)

WM2091.0

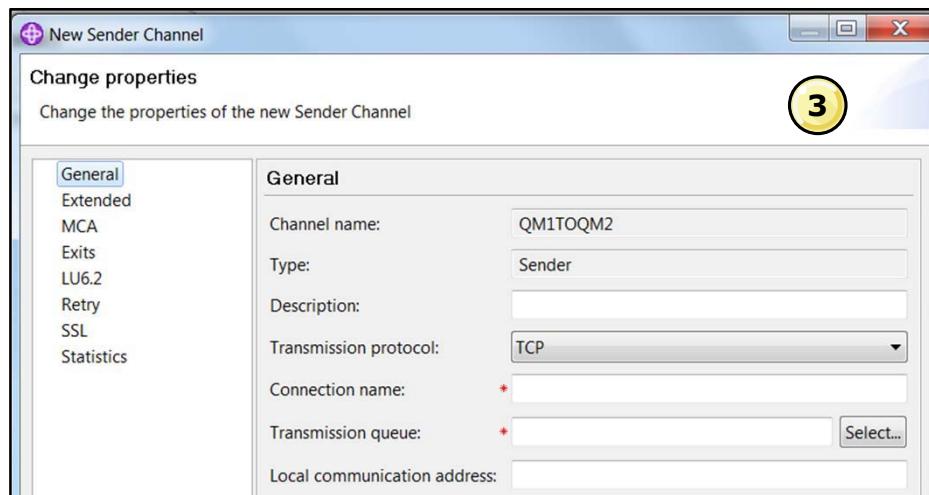
### Notes:

You can also use MQ Explorer to define channels.

1. In the Navigator, expand the queue manager folder.
2. Right-click **Channels**, click **New**, and then select the channel type.



## Defining channels in IBM MQ Explorer (2 of 2)



- **MCA:** Specify how the MCA program of the channel definition runs
- **Exits:** Configure any user exits
- **LU6.2:** Configure the MCA if the MCA uses the LU6.2 transport protocol
- **Retry:** Configure the channel behavior if the channel cannot connect to the remote queue manager
- **SSL:** Specify the SSL settings for this end of the channel
- **Statistics:** Control the collection of statistics and monitoring data for the channel

© Copyright IBM Corporation 2014

Figure 9-21. Defining channels in IBM MQ Explorer (2 of 2)

WM2091.0

### Notes:

3. Configure the channel properties.

The figure summarizes some of the channel definition property tabs. You must provide values for the required properties.

## Controlling the listener process

- Start the IBM MQ listener: **runmq1sr**
  - Run synchronously and waits until the listener process has finished before returning to the caller
  - Must identify the transmission protocol as TCP/IP, SNA LU 6.2 ( Windows only), NetBIOS ( Windows only), or SPX ( Windows only)
- End the IBM MQ listener: **endmq1sr**

© Copyright IBM Corporation 2014

Figure 9-22. Controlling the listener process

WM2091.0

### Notes:

You can use the listener for all the supported communications protocols.

The Internet Assigned Numbers Authority assigns port number 1414 to MQ and, by default, MQ uses this port number. However, if you are running multiple queue managers on a system, with each needing to respond to incoming requests to start a channel, you must specify more port numbers.

## Listener object

- Create a listener object:
  - Optionally, specify **CONTROL (QMR)** so that the listener starts and stops when the queue manager starts and stops

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) +
    PORT(1414) +
    CONTROL(QMGR) +
    REPLACE
```

- Start the listener:

```
START LISTENER(LISTENER.TCP)
```

© Copyright IBM Corporation 2014

Figure 9-23. Listener object

WM2091.0

### Notes:

A channel listener program is defined and run on each queue manager. A channel listener program listens for incoming network requests and starts the appropriate receiver channel when it is needed.

The implementation of channel listeners is operating system specific; however, there are some common features. On all MQ operating systems, the listener can be started by using the MQSC command **START LISTENER**

On MQ for Windows systems and UNIX systems, you can make the listener start automatically with the queue manager by setting the **CONTROL** attribute to **QMGR** or **STARTONLY**

On Windows and UNIX, use either the channel listener program that MQ provides, or the facilities that the operating system provides.

To start the MQ channel listener, use the **rwmqqlsr** command. For example, to start the listener for a queue manager that is named QM1 on port 1414, type:

```
rwmqqlsr -t tcp -p 1414 -m QM1
```

## Starting a message channel

- IBM MQ commands:

**PING CHANNEL (QMA\_QMB)**

**START CHANNEL (QMA\_QMB)**

- Start a sender or requester channel

- From a command: `runmqchl -c Channel -m Qmgr`
- From IBM MQ Explorer

- Channel attributes that are evaluated when channel is started:

<b>BATCHSZ</b>	Batch size
<b>MAXMSGL</b>	Maximum message length
<b>SEQWRAP</b>	Sequence number wrap
<b>HBINT</b>	Heartbeat interval
<b>NPMSPEED</b>	Non-persistent message speed

© Copyright IBM Corporation 2014

Figure 9-24. Starting a message channel

WM2091.0

### Notes:

Check the network first by entering a TCP/IP **ping**, for example.

Use the MQSC **PING CHANNEL** command to test a message channel configuration. It can be used at the sender or server end of a channel, and only when the channel is not started.

You can use the **runmqchl** control command to start a sender (SDR) or a requester (RQSTR) channel. The channel runs synchronously.

Take particular care that the channel definitions are correct. Some attributes that are specified in the channel definition at each end of a message channel are compared when the channel is started:

- **BATCHSZ**: Maximum number of messages that are sent before a sync point is taken. A larger maximum batch size can improve throughput. The lower value from the two channel definitions is used.
- **MAXMSGL**: Maximum message length that the channel can transmit. The lower value from the two channel definitions is used.
- **SEQWRAP**: Highest message sequence number before wrapping. The **SEQWRAP** values in the two channel definitions must match; otherwise the channel fails to start. Messages are

numbered internally in the Message Channel Protocol and the sequence numbers that are used restart at 1 after reaching the value specified on the **SEQWRAP** attribute. Do not specify a low value.

- **HBINT:** The time between heartbeat flows that are sent from a sending MCA to a receiving MCA when there are no messages on the transmission queue. A heartbeat flow can unblock a receiving MCA for which the **STOP CHANNEL** command is entered. The higher value from the two channel definitions is used.
- **NPMSPED:** The speed at which non-persistent messages are sent. You can specify **NORMAL** or **FAST**. The default is **FAST**, which means that a non-persistent message is sent outside of a batch and becomes available for retrieval as soon as it is put on its destination queue. If the definitions at the sending and receiving ends of a channel do not specify the same value, or if one end does not support fast non-persistent messages, then **NORMAL** is used.

If a message cannot be delivered, it is put on the local dead-letter queue. If it cannot be put there, the channel is stopped. However, if a fast non-persistent message cannot be delivered and cannot be put on the dead-letter queue, it is discarded and the channel remains open.

## Reasons why a channel might fail to start

- Definitions at both ends of a channel do not specify the same channel name
- Channel is not defined at both ends with compatible types
- A sequence number mismatch
  - Deleted a channel definition at one end of a message channel and then redefined it
  - Deleted and created a queue manager again

© Copyright IBM Corporation 2014

Figure 9-25. Reasons why a channel might fail to start

WM2091.0

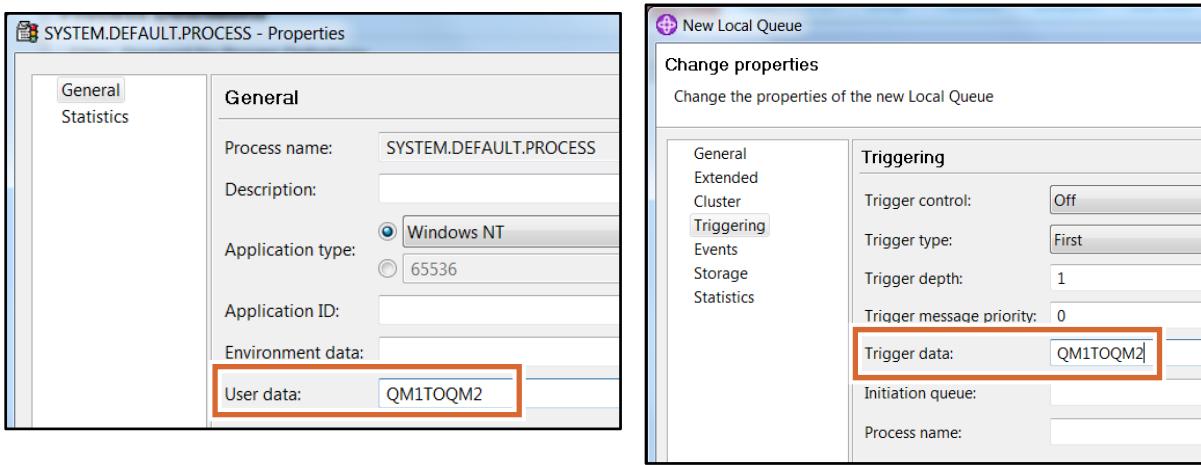
### Notes:

There are several reasons why a channel might fail to start. Some common problems are:

- The definitions at both ends of a channel do not specify the same channel name. Remember, in particular, that the names of channels, like the names of queues, are case-sensitive.
- A channel is not defined at both ends with compatible types.
- There is a sequence number mismatch, often caused by deleting a channel definition at one end of a message channel and then redefining it, or deleting and creating a queue manager again.

## Channel initiator

- Two main functions:
  - Triggering a channel
  - Try channels again
- Options to starts an MCA at sending end of a message channel:
  - Specify the channel name in **User data** attribute of the process object
  - Specify the channel name in **Trigger data** attribute of the transmission queue



© Copyright IBM Corporation 2014

Figure 9-26. Channel initiator

WM2091.0

### Notes:

Instead of starting a channel manually, you can use a *channel initiator* to start the channel automatically. The channel initiator is a special trigger monitor for starting a message channel. It also contains logic in case of difficulty in starting a channel or after an error on a channel.

There are two options for starting an MCA at the sending end of a message channel.

- Specify the name of the channel to start in the **User data** attribute of a process object that triggers the channel.
- Specify the name of the channel to start in the **Trigger data** attribute of the transmission queue

You can configure either of the options by using MQ Explorer or by using MQSC commands to alter the definition of the process or transmission queue.

If you do not specify a channel name, the channel initiator searches the channel definitions until it finds a channel whose definition contains the name of the transmission queue that caused the trigger event.

## Channel control parameters

- Specified on the **DEFINE CHANNEL** or **ALTER CHANNEL** command

<b>DISCINT</b>	Disconnect interval
<b>SHORTRTY, SHORTTMR</b>	Short retry count, short retry interval
<b>LONGRTY, LONGTMR</b>	Long retry count, long retry interval
<b>MRRTY, MRTMR</b>	Message retry count, message retry interval
<b>MCATYPE</b>	Message channel agent type of <b>PROCESS</b> or <b>THREAD</b>
<b>BATCHINT</b>	Batch interval

- Can be configured in IBM MQ Explorer **Channel > Properties**

© Copyright IBM Corporation 2014

Figure 9-27. Channel control parameters

WM2091.0

### Notes:

The channel control parameters that are shown in the figure can be specified on the **DEFINE CHANNEL** or **ALTER CHANNEL** command, or in MQ Explorer.

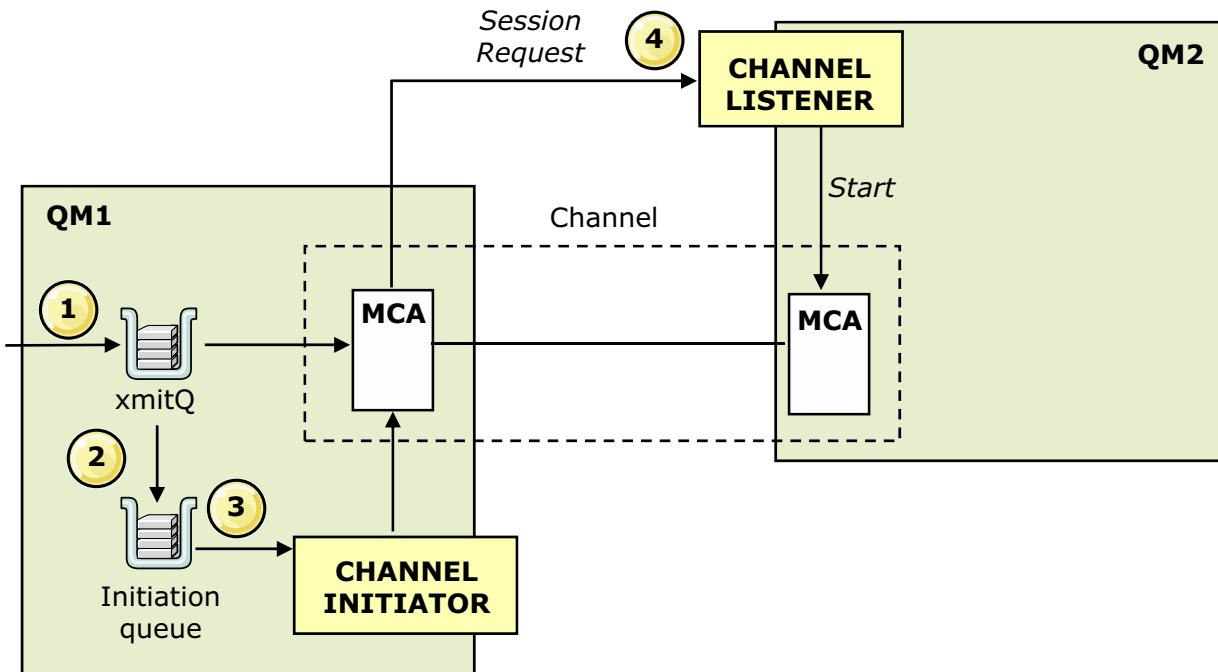
- DISCINT** (SDR and SVR): Maximum time to wait for a message on the transmission queue if it is empty. If no message arrives within this time, the channel closes.
- SHORTRTY, SHORTTMR** (SDR and SVR): Short retry count and short retry time interval to control repeated attempts to establish a communications connection.
- LONGRTY, LONGTMR** (SDR and SVR): Long retry count and long retry time interval to control further repeated attempts to establish a communications connection.
- MRRTY, MRTMR** (RCVR and RQSTR): Message-retry count and message-retry interval when attempting to put a message on a destination queue. If every attempt fails, the MCA decides that it cannot deliver the message.
- MCATYPE** (SDR, SVR, and RQSTR): The value of this parameter can be **THREAD** or **PROCESS**. If **THREAD** is specified, each channel runs as a thread within the channel initiator process. If **PROCESS** is specified, each channel runs as a separate process.

- **BATCHINT** (SDR and SVR): The length of time during which a channel keeps a batch open if there are no messages on the transmission queue. Set this parameter to a value considerably less than the value of **DISCINT**.

The disconnect interval and retry attributes provide some automation in the operation of channels. They allow channels to stop during periods of inactivity and restart when more messages arrive.

Of the parameters that are listed, the channel initiator uses only the **SHORTRTY**, **SHORTTMR**, **LONGRTY**, **LONGTMR**, and **MCATYPE** parameters.

## Channel initiator triggering



© Copyright IBM Corporation 2014

Figure 9-28. Channel initiator triggering

WM2091.0

### Notes:

The figure shows the process for triggering a channel:

1. Message arrives on the transmission queue that satisfies triggering criteria for that queue.
2. Message is sent to the initiation queue.
3. The channel initiator gets the message from the initiation queue and uses it to start the appropriate channel.
4. The sender channel causes the listener to start the remote receiver channel, which then receives messages that are sent from the transmission queue (xmitQ). The channel listener program starts the receiving MCA.

## Channel states: DISPLAY CHSTATUS command (1 of 2)

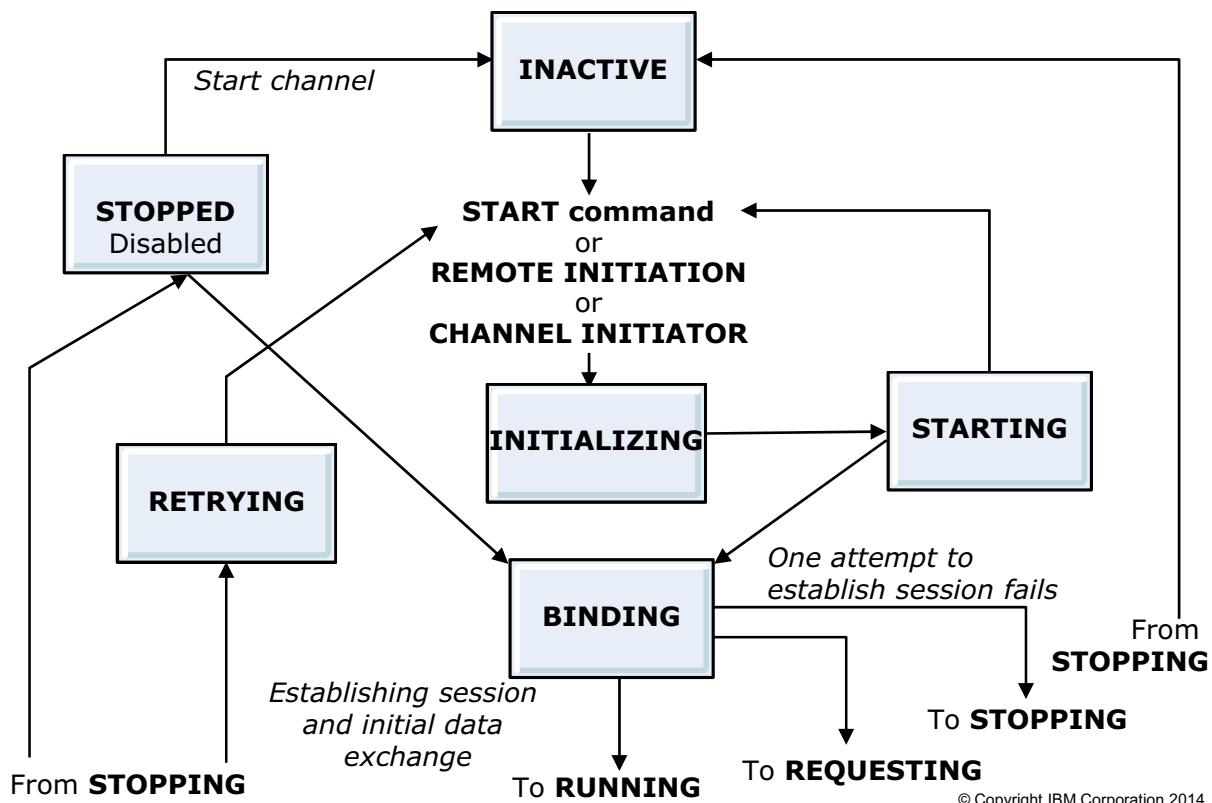


Figure 9-29. Channel states: DISPLAY CHSTATUS command (1 of 2)

WM2091.0

### Notes:

The current state of a channel can be determined by using the MQSC command **DISPLAY CHSTATUS**. This figure, and the figure on the next page, shows the possible states of a channel and the possible flows from one state to the next.

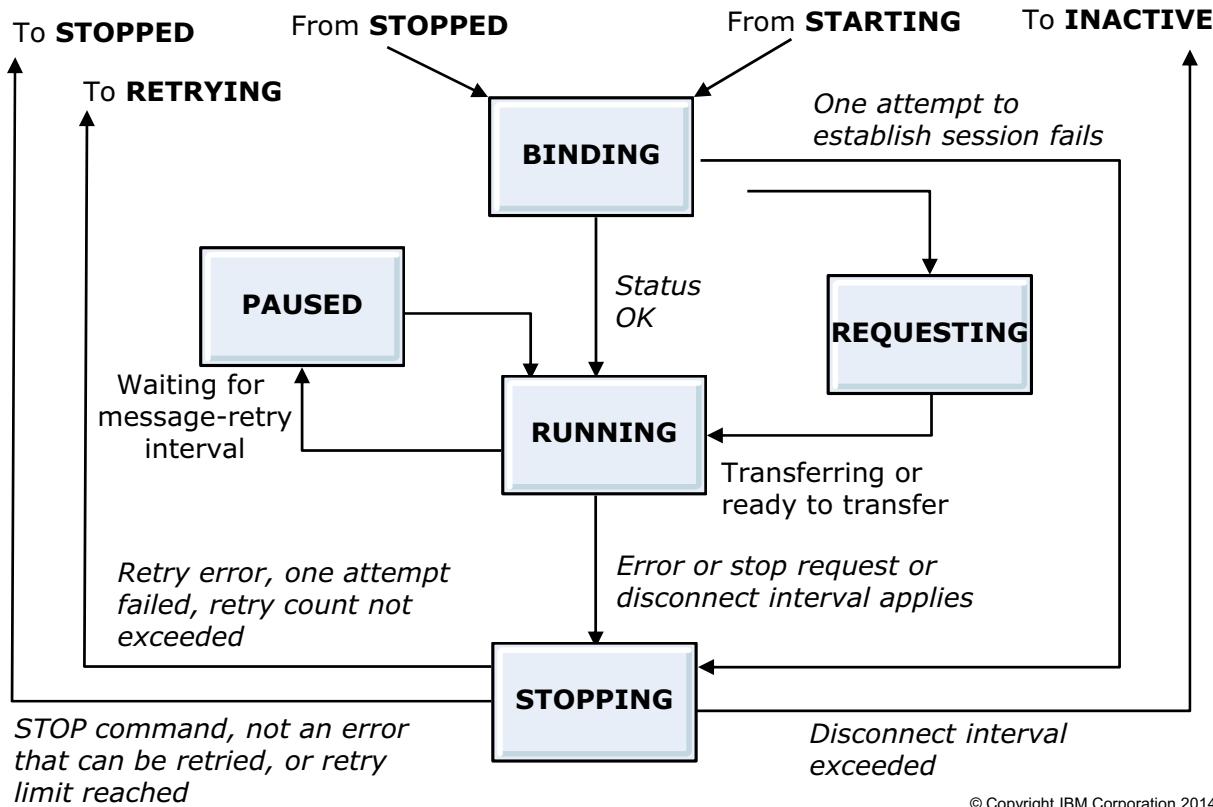
When a channel is in the INITIALIZING, BINDING, REQUESTING, RUNNING, PAUSED, or STOPPING state, it is using resource and a process or thread is running; the channel is active.

START is not really a state. It indicates a start point for when the START CHANNEL command is entered, a transmission queue is triggered, a channel initiator issues a retry, or there is an incoming request to start a channel.

When a channel is in the INITIALIZING state, a channel initiator is attempting to start a channel.

A channel stays in the STARTING state if there is no active slot available. If there is an active slot immediately available, it remains in this state for a short time.

## Channel states: DISPLAY CHSTATUS command (2 of 2)



© Copyright IBM Corporation 2014

Figure 9-30. Channel states: DISPLAY CHSTATUS command (2 of 2)

WM2091.0

### Notes:

During the **BINDING** state, the channel establishes a communications connection and completes the initial data exchange.

In the **REQUESTING** state, a requester is waiting for callback from a sender.

In the **RUNNING** state, messages are being transferred or the channel is waiting for messages to arrive on the transmission queue.

A **PAUSED** channel is waiting for the message-retry interval to complete before attempting to put a message on its destination queue.

In the channel enters the **STOPPING** state if an error occurred, or the **STOP CHANNEL** command was entered, or the disconnect interval expired.

The **RETRYING** state indicates that the channel is waiting until it is time for the channel initiator to make the next attempt to start the channel.

A channel in a **STOPPED** state is disabled and needs manual intervention to start it again.

An **INACTIVE** channel is one that never started or that was disconnected normally.

## Queue definitions for distributed queuing

- Local definition of a remote queue

Example: `DEFINE QREMOTE (BBB) RNAME (YYY) RQMNAME (QM2)`

- Transmission queue must be created at the sending end of each message channel

- `USAGE (XMITQ)` indicates its purpose
- It can have any of the attributes of a local queue

Example: `DEFINE QLOCAL (QM2) USAGE (XMITQ)`

© Copyright IBM Corporation 2014

Figure 9-31. Queue definitions for distributed queuing

WM2091.0

### Notes:

There are two special queues that are required for distributed queuing: a local definition of a remote queue and a transmission queue.

Applications can retrieve messages only from local queues. They can put messages on local queues or remote queues. Therefore, a queue manager might have remote queue definitions and a definition for each of its local queues. Remote queue definitions are definitions for queues that another queue manager owns.

The advantage of remote queue definitions is that they allow an application to put a message to a remote queue without specifying the name of the remote queue, the remote queue manager, or the transmission queue. Remote queues give location independence.

The location of a remote queue can be hidden from an application by creating a local definition of a remote queue by using the MQSC command `DEFINE QREMOTE`. You can also use MQ Explorer to create a local definition of a remote queue.

As described previously, the transmission queue is a local queue that is used when a queue manager forwards messages to a remote queue manager through a message channel. For clarity, give a transmission queue the same name as the remote queue manager.

## Define remote queue (QREMOTE)

- Another queue manager owns remote queues
- Define a remote queue definition

Example:

Define a remote queue that is named PAYROLL.QUERY on the local queue manager. The physical queue is named PAYROLL on a remote queue manager QM2. The local queue manager uses the transmission queue that is named QM2.

```
DEFINE QREMOTE (PAYROLL.QUERY) +
DESCR('Remote queue for QM2') +
REPLACE RNAME (PAYROLL) RQMNAME (QM2) XMITQ (QM2)
```

© Copyright IBM Corporation 2014

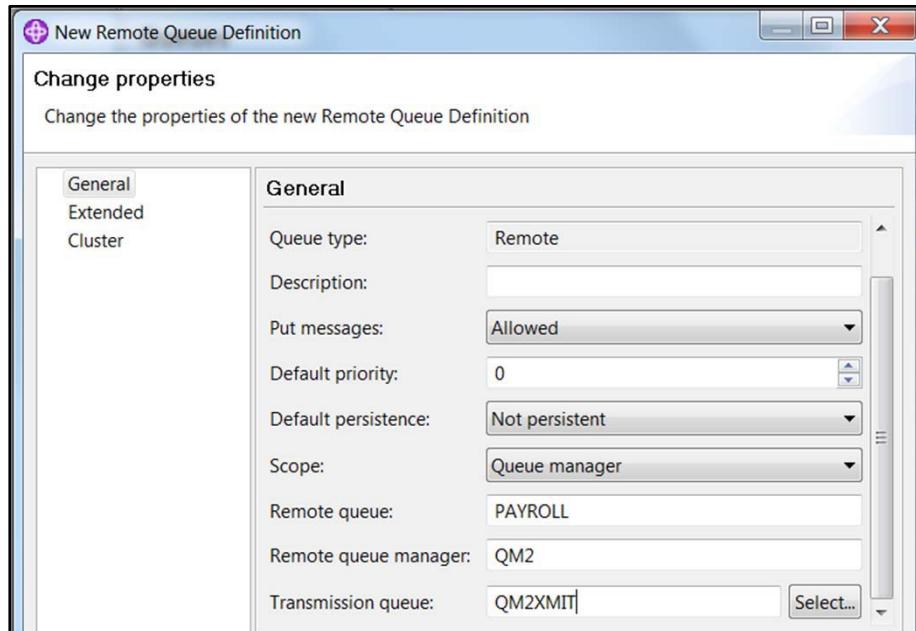
Figure 9-32. Define remote queue (QREMOTE)

WM2091.0

### Notes:

The example in the figure shows that the remote queue definition is not a physical queue. It is a way to direct messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

## Using IBM MQ Explorer to define a remote queue



1. Right-click **Queues**, and then click **New > Remote Queue Definition**.
2. Enter a name for the local definition of the remote queue.
3. Enter the remote queue definition properties.

© Copyright IBM Corporation 2014

Figure 9-33. Using IBM MQ Explorer to define a remote queue

WM2091.0

### Notes:

You can also use MQ Explorer to define a local definition of a remote queue.

This figure lists the steps for creating a local definition of a remote queue. The properties are consistent with the attributes that you would use if you created the queue by using the MQSC **DEFINE QREMOTE** command.



## Managing a remote queue manager with IBM MQ Explorer

1. Create a server connection channel.

Example command:

```
DEF CHL (SYSTEM.ADMIN.SVRCONN) CHLTYPE (SVRCONN)
MCAUSER (MQADMIN)
```

Where **MQADMIN** is the MQ administrator user ID

2. Create a listener to accept incoming network connections.
3. Start the listener.
4. On local IBM MQ Explorer, right-click **Queue Managers**, click **Add remote queue manager**, and then identify the remote queue manager.

© Copyright IBM Corporation 2014

Figure 9-34. Managing a remote queue manager with IBM MQ Explorer

WM2091.0

### Notes:

You can also use MQ Explorer to manage a remote queue manager after the remote queue manager is configured to allow remote management.

The first three steps in this figure list the configuration actions on that must be made on the remote queue manager.

To run the listener in background with no hang-up, type:

```
nohup runmqlsr -t tcp -p port -m QMgrName 2>&1 &
```

To verify that the listener is running on Linux, type:

```
ps -ef | grep runmqlsr
```

After you complete the first three steps, you can add the remote queue manager to MQ Explorer for management.



## Methods for accessing a remote queue manager

- Multi-hopping
  - Using intermediate queue managers
  - Channel and transmission queue definitions required
- Channel sharing
  - Remote queue definitions specify the transmission queue
  - Using different channels
- Clustering

© Copyright IBM Corporation 2014

Figure 9-35. Methods for accessing a remote queue manager

WM2091.0

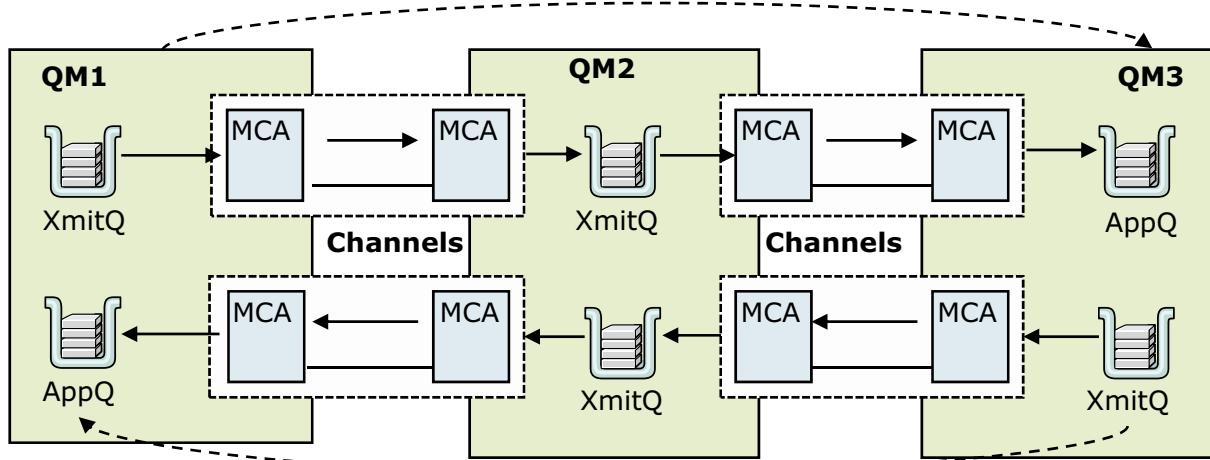
### Notes:

You might not always have one channel between each source and target queue manager. The other methods for accessing a remote queue manager are:

- Multi-hopping
- Channel sharing
- Using clustering

Multi-hopping, channel sharing, and channel sharing by using different channels are described in detail on the next pages.

## Multi-hopping



Pass through one or more intermediate queue managers when there is no direct communication link between the source queue manager and the target queue manager.

© Copyright IBM Corporation 2014

Figure 9-36. Multi-hopping

WM2091.0

### Notes:

It is possible to pass through one or more intermediate queue managers when there is no direct communication link between the source queue manager and the target queue manager. This behavior is known as a *multi-hopping*.

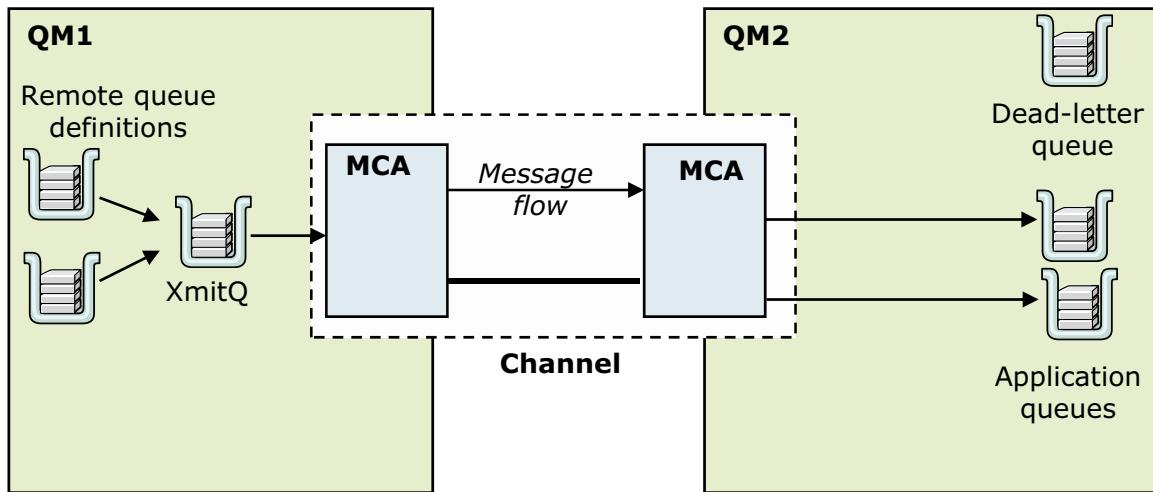
In this configuration, you must define channels between all the queue managers, and transmission queues on the intermediate queue managers.

The figure shows three queue managers. To send messages to the destination queue on QM3, an intermediate queue manager, QM2, is used.

Messages are sent from QM1 to the transmission queue on QM2, and then on to the destination queue on QM3.

Channel definitions exist between QM1 and QM2, and between QM2 and QM3.

## Channel sharing



All messages from all applications that address queues at the same remote location send their messages through the same transmission queue.

© Copyright IBM Corporation 2014

Figure 9-37. Channel sharing

WM2091.0

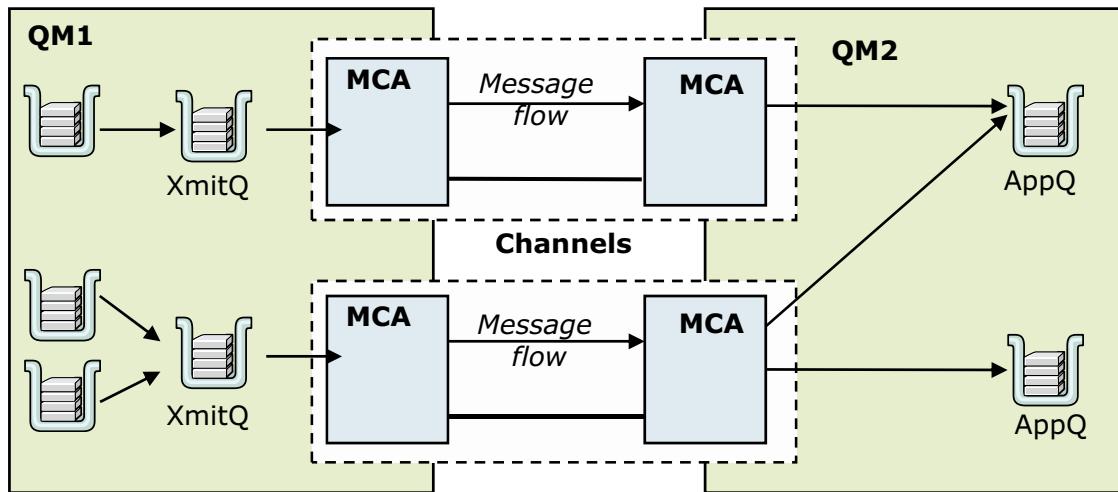
### Notes:

An application designer, has the choice of forcing applications to specify the remote queue manager name and the queue name, or creating a remote queue definition for each remote queue. This definition holds the remote queue manager name, the queue name, and the name of the transmission queue.

Either way, all messages from all applications that address queues at the same remote location have their messages that are sent through the same transmission queue.

This diagram shows a message flow from queue manager QM1 to queue manager QM2. Remote queue definitions on QM1 allow QM1 to put messages to a remote queue manager. The messages are sent by using the transmission queue on QM1.

## Using different channels



If you have messages of different types to send between two queue managers, you can define more than one channel between the them.

© Copyright IBM Corporation 2014

Figure 9-38. Using different channels

WM2091.0

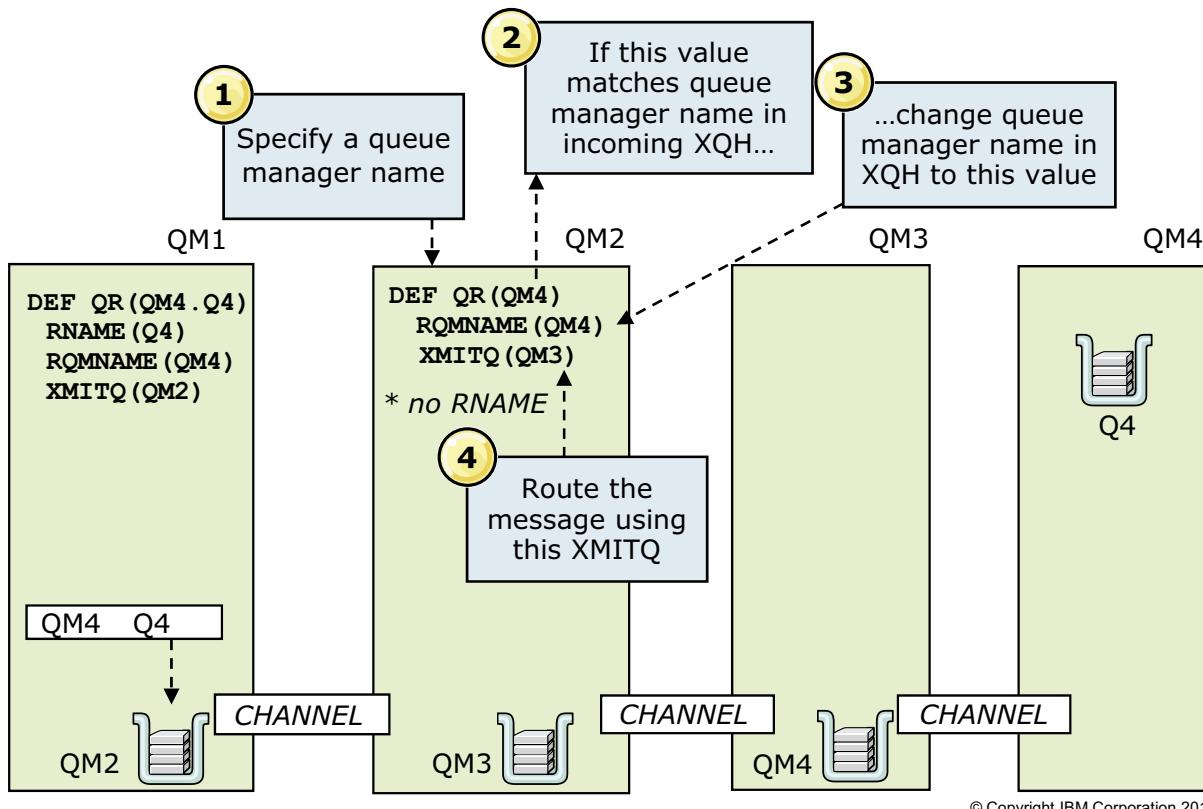
### Notes:

If you have messages of different types to send between two queue managers, you can define more than one channel between the queue managers. There are times when you need alternative channels, perhaps for security purposes, or to trade off delivery speed against message traffic.

To set up a second channel you must define another channel and another transmission queue, and create a remote queue definition that specifies the location and the transmission queue name. Your applications can then use either channel but the messages are still delivered to the same target queues.

The figure shows multiple channels between two queue managers. The sending queue manager, QM1 has two channels that are defined, with a transmission queue for each channel. Messages are sent to the receiving queue manager, QM2 by using both channels.

## Using a queue manager alias



© Copyright IBM Corporation 2014

Figure 9-39. Using a queue manager alias

WM2091.0

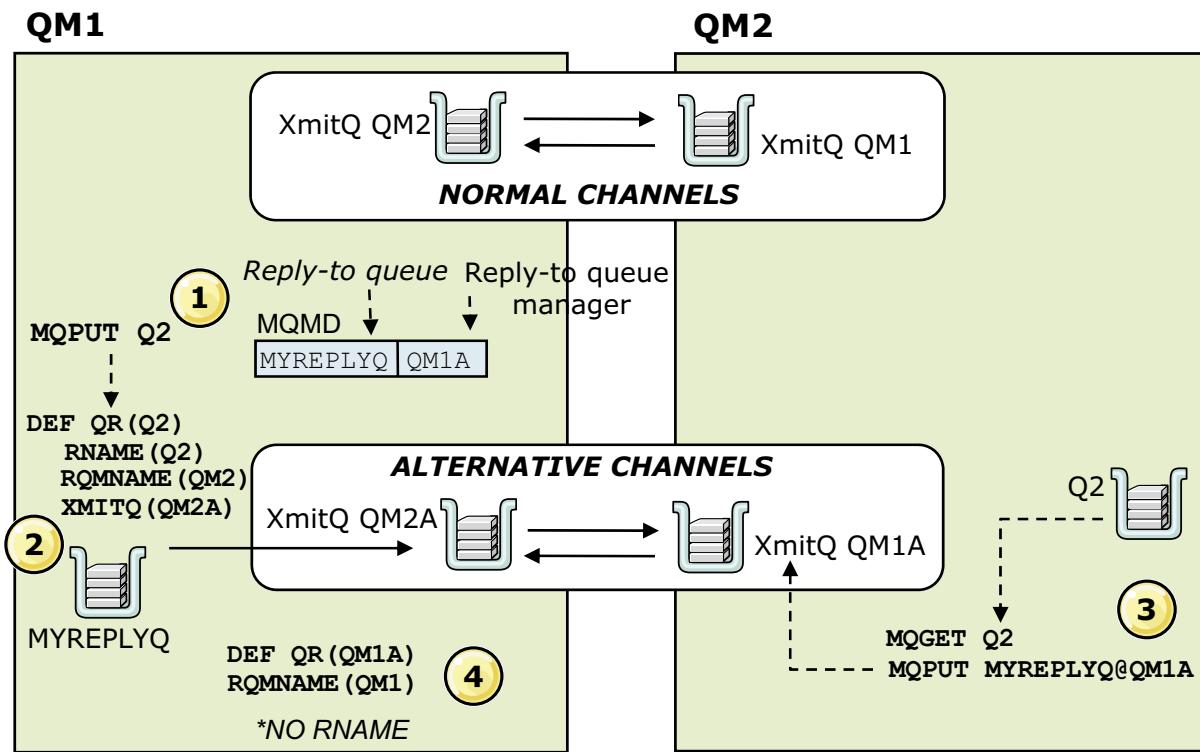
### Notes:

Aliases are used to provide a quality of service for messages. The queue manager alias allows a system administrator to alter the name of a target queue manager without causing you to change your applications. It also allows the system administrator to alter the route to a destination queue manager, or to set up a route that involves passing through a number of other queue managers (multi-hopping). The reply-to queue alias provides a quality of service for replies.

The MQSC command **DEFINE QRREMOTE** is used to define a queue manager alias.

Queue manager aliases and reply-to queue aliases are created by using a remote-queue definition **no RNAME** attribute, as shown in the figure. These definitions do not define real queues; the queue manager uses them to resolve physical queue names, queue manager names, and transmission queues.

## Separating message flows



© Copyright IBM Corporation 2014

Figure 9-40. Separating message flows

WM2091.0

### Notes:

It is possible to separate message flows between two queue managers.

On each queue manager, the normal transmission queue has the same name as the remote queue manager. On queue manager QM1, a local definition of a remote queue specifies an alternative transmission queue, QM2A, that can be used for sending messages to queue manager QM2.

A reply-to queue alias is used to set the value of reply-to queue manager QM1A.

A queue manager alias definition specifies QM1A as an alias of QM1. By using these three definitions, request messages can be separated into two flows between queue managers QM1 and QM2, and the reply messages can be separated into two flows in the reverse direction.

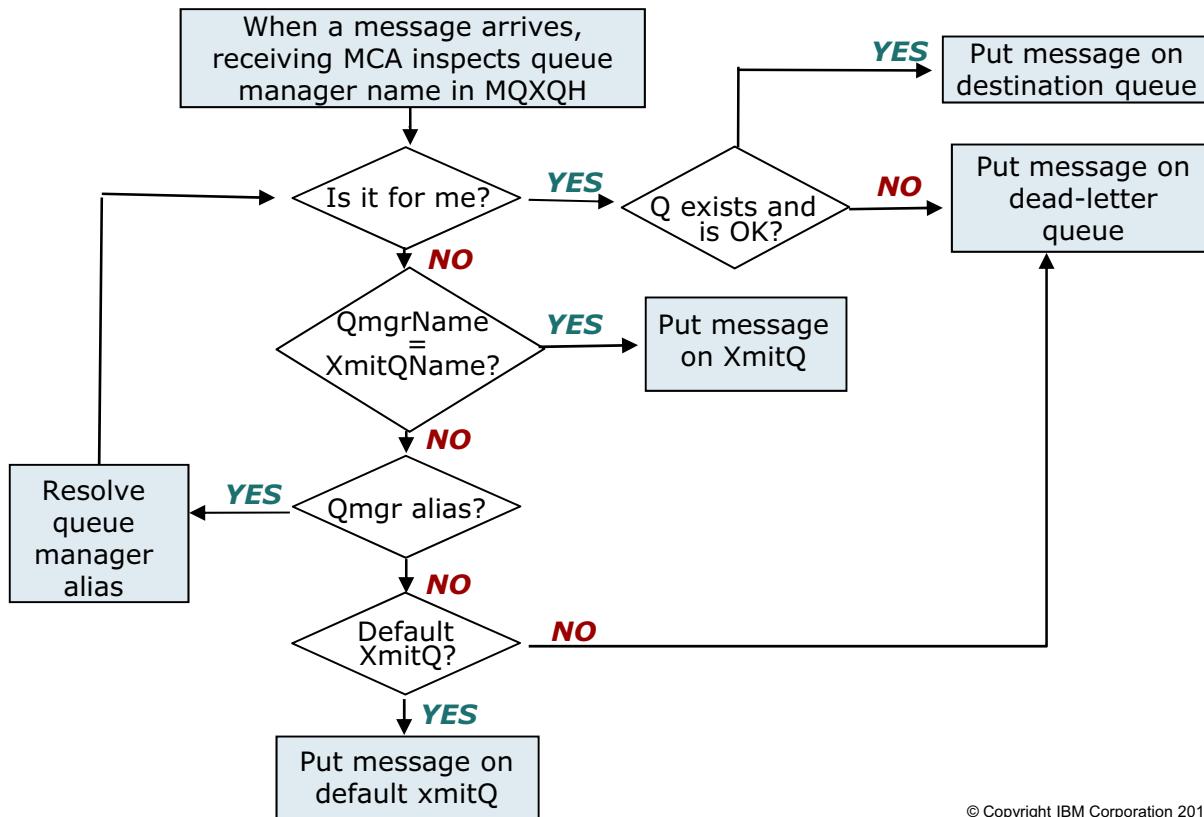
1. A program puts a request message on queue Q2. Q2 is a local definition of a remote queue, which specifies Q2 as the remote queue name QM2 and QM2A as the transmission queue. The message is sent on the channel that serves transmission queue QM2A, not on the "normal" channel, which serves transmission queue QM2.
2. In the message, the program specifies the reply-to queue as MYREPLYQ, which is resolved by using a reply-to queue alias to reply-to queue MYREPLYQ on reply-to queue manager QM1A.

3. Another program gets request messages from queue Q2 on QM2. Each request message includes, in its message descriptor, the name of the reply-to queue and reply-to queue manager.

There is no local definition of a remote queue on queue manager QM2, which explicitly identifies a transmission queue. As a result, a reply message is put on the transmission queue whose name is the same as the reply-to queue manager, such as QM1A in the example. Thus the reply message is sent on the channel that serves transmission queue QM1A, not on the normal channel, which serves transmission queue QM1.

4. On queue manager QM1, a queue manager alias definition specifies QM1A as an alias of QM1. When a reply message arrives at queue manager QM1 addressed to QM1A, the queue manager resolves the queue manager alias and the message is put on the queue MYREPLYQ.

## When a message arrives at a queue manager



© Copyright IBM Corporation 2014

Figure 9-41. When a message arrives at a queue manager

WM2091.0

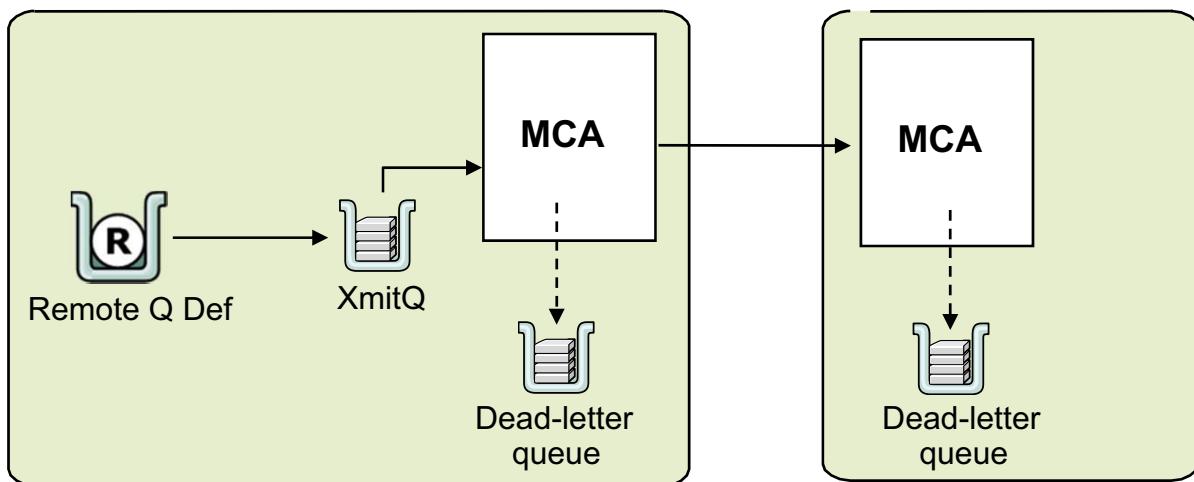
### Notes:

The figure shows the process flow that is followed when a message arrives on a queue manager.

When a message arrives, the receiving MCA inspects the queue manager name in the transmission queue header (MQXQH) to determine whether the message matches its name. If it does, the queue manager checks if the destination queue exists and whether it can accept messages. If it exists and it is running, the queue manager puts the message on the queue. If the destination queue does not exist or is not running, the message is put on the dead-letter queue.

If the destination queue manager name does not match, the destination queue manager name is compared to the transmission queue. If there is a match, the message is put on the transmission queue.

## Dead-letter queue (1 of 2)



- Messages that cannot be delivered are placed on the dead-letter queue
- Dead-letter queue header contains details of the destination queue name and queue manager name

© Copyright IBM Corporation 2014

Figure 9-42. Dead-letter queue (1 of 2)

WM2091.0

### Notes:

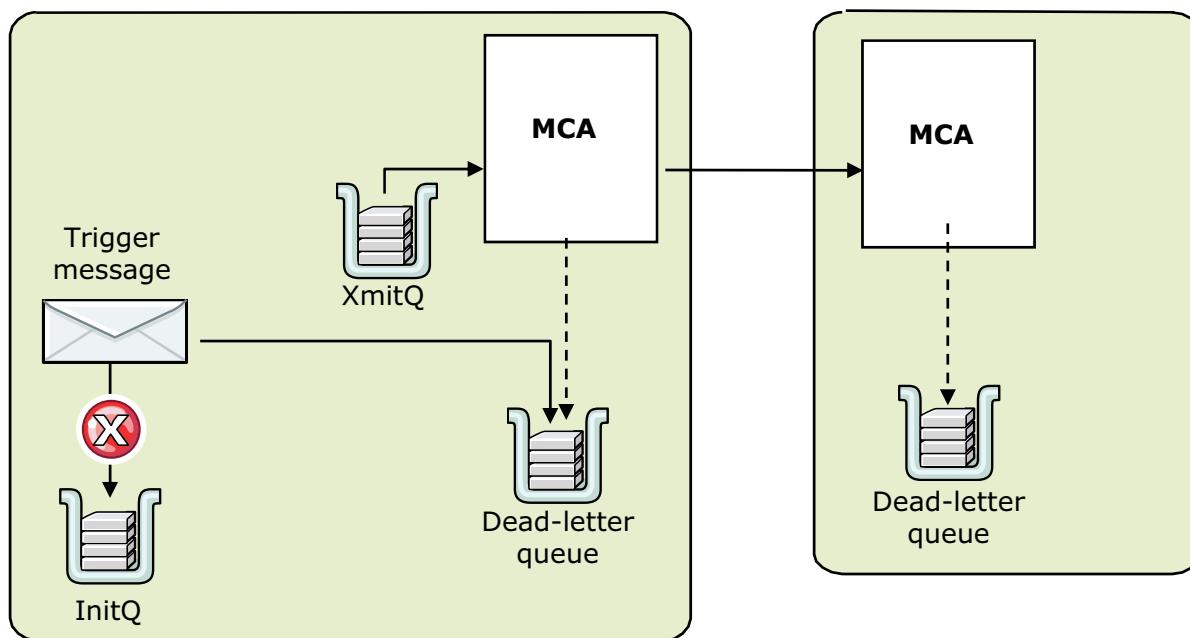
When a problem that relates to a message is detected asynchronously, an exception report is generated if one is requested and the report is sent to the specified reply-to queue. The *Feedback* field in the message descriptor of the report message indicates the reason for the report. The original message is put on the dead-letter queue unless MQRO\_DISCARD\_MSG is requested as a report option.

If a message cannot be delivered, it is put on the dead-letter queue at the receiving end of a message channel, if one is defined. A message-retry at the receiving end of a channel might be useful if the problem is only temporary.

If a message cannot be put on the dead-letter queue, the channel is stopped and the message remains on the transmission queue. A fast non-persistent message, however, is discarded in these circumstances and the channel remains open.

Always create a dead-letter queue. If it does not exist when a message cannot be delivered, the channel stops and the message remains on the transmission queue.

## Dead-letter queue (2 of 2)



- If a trigger message is created but cannot be put on the initiation queue, the trigger message is put instead on the dead-letter (undelivered message) queue. For example, if the queue is full or the length of the trigger message is greater than the maximum message length specified for the initiation queue, the message is put on the dead-letter queue.

© Copyright IBM Corporation 2014

Figure 9-43. Dead-letter queue (2 of 2)

WM2091.0

### Notes:

If a trigger message is created but cannot be put on the initiation queue, the trigger message is put instead on the dead-letter (undelivered message) queue. For example, if the queue is full or the length of the trigger message is greater than the maximum message length specified for the initiation queue, the message is put on the dead-letter queue.

If the PUT operation to the dead-letter queue cannot complete successfully, the trigger message is discarded and a warning message is placed in the error log.

## Using dead-letter queues

- Create a dead-letter queue on all queue managers
- Use message retry on message channels for transient conditions
- Consider a "return to sender" function in the application

```
MQRO_PASS_MSG_ID +
MQRO_PASS_CORREL_ID +
MQRO_EXCEPTION_WITH_FULL_DATA +
MQRO_DISCARD_MSG
```

- Regularly run a routine that processes messages on the dead-letter queue to ensure that it does not become full
- WebSphere MQ Dead-letter Queue Handler utility checks messages that are on the dead-letter queue and processes them according to a set of rules that you supply to prevent an application dead letter queue from becoming full

© Copyright IBM Corporation 2014

Figure 9-44. Using dead-letter queues

WM2091.0

### Notes:

This figure lists some guidelines for using dead-letter queues.

Create a dead-letter queue on all queue managers.

Use message-retry on message channels to allow for transient conditions.

Consider the combination of report options that implements "return to sender" as an alternative to putting a message on the dead letter queue. The use of the "return to sender" function might mean that some messages targeted for the dead letter queue might be returned.

Do not allow an application dead-letter queue to become full.

You can use the dead-letter queue handler to prevent the dead-letter queue from becoming full. The dead-letter queue handler is a stand-alone utility that checks the messages on the dead-letter queue and processes them according to a set of rules. The arrival of a message on the dead-letter queue can trigger the dead-letter queue handler.

If there is no reasonable retry option, forward the message to an application dead-letter queue.

## Dead-letter header (MQDLH)

- Prefixes application message data of messages on the dead-letter queue
- Applications that put messages directly on the dead-letter queue must prefix message data with an MQDLH structure, and initialize fields with appropriate values
- Queue manager does not require that an MQDLH structure is present, or that valid values are specified for the fields
- Fields include:
  - Reason why message was put on dead-letter queue
  - Name of original destination queue and queue manager
  - Date and time message was put on queue

© Copyright IBM Corporation 2014

Figure 9-45. Dead-letter header (MQDLH)

WM2091.0

### Notes:

The MQ dead-letter queue header (MQDLH) is the information that prefixes the application message data of the messages on the dead-letter queue.

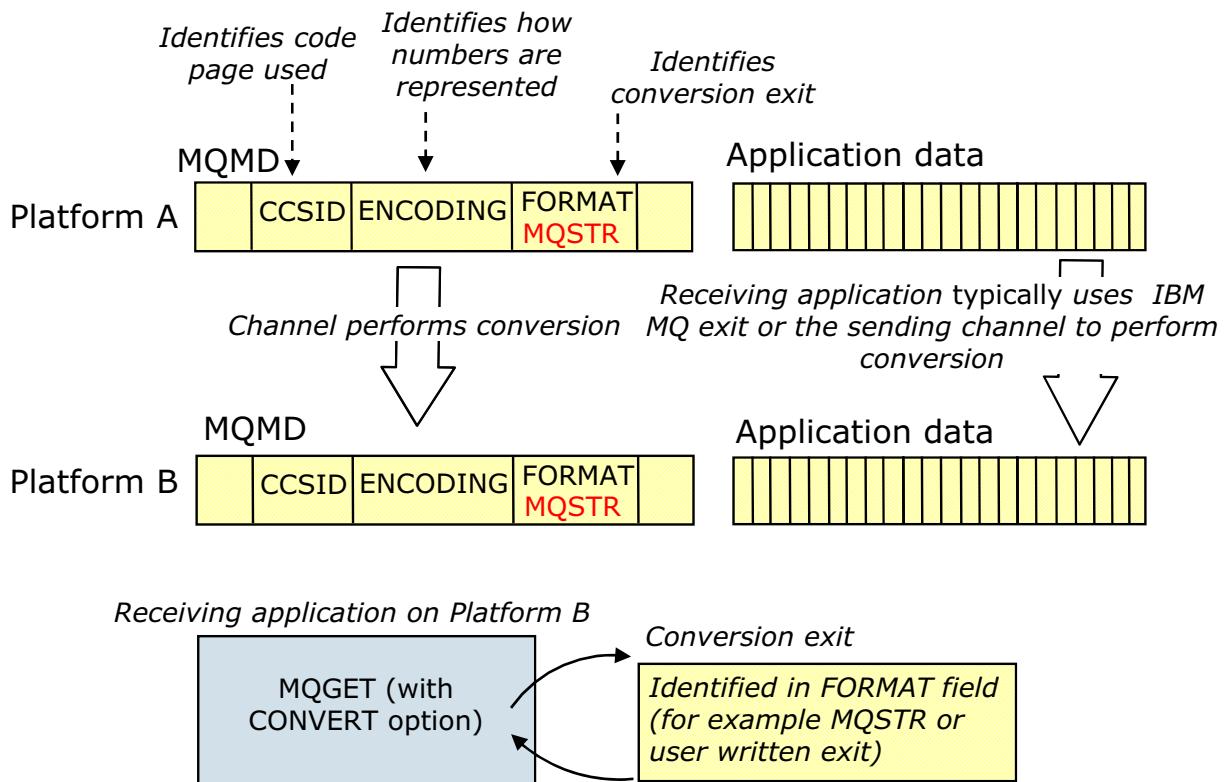
Applications that put messages directly on the dead-letter queue must prefix message data with an MQDLH structure, and initialize fields with appropriate values. However, the queue manager does not require an MQDLH structure, or that valid values are specified for the fields.

Fields in the MQDLH structure include:

- The reason why the message was put on the dead-letter queue
- The name of the original destination queue and queue manager
- The date and time the message was put on the dead letter queue

See the MQ product documentation for a detailed description of all fields in the MQDLH.

## Data conversion



© Copyright IBM Corporation 2014

Figure 9-46. Data conversion

WM2091.0

### Notes:

When sending messages between queue managers, you must consider the message code page and encoding, especially when sending data between countries and operating systems.

A message descriptor accompanies every message and is delivered to the receiving application with the application data. The message descriptor is always converted to the representation of the destination system by all queue managers. When a message channel is started between two queue managers, the two MCAs determine what conversion is required and decide which of them does it.

The conversion of the application data in a message, however, requires more attention.

## Data representation

- Messages in a heterogeneous network
  - Character fields might need translation
  - Numeric fields might need transformation
- Message descriptor accompanies every message
  - Delivered to the receiving application
  - Always converted by IBM MQ
- Application data
  - Fields in the message descriptor describe the format and representation
  - Data conversion support is available

© Copyright IBM Corporation 2014

Figure 9-47. Data representation

WM2091.0

### Notes:

When messages are being routed through a heterogeneous network of queue managers, there is a requirement to be able to handle different data representations.

- Some character fields might require translation from one character set to another.
- Some numeric fields might require transformation, such as byte reversal for integers.



## Data representation fields in the MQMD

- **Encoding:** Representation of the numeric data in the message

MQENC\_NATIVE

- **CodedCharSetId:** Representation of the character data in the message

MQCCSI\_Q\_MGR

- **Format**

- Indicates the nature of the data in the message
  - Values starting with “MQ” are reserved for IBM MQ

© Copyright IBM Corporation 2014

Figure 9-48. Data representation fields in the MQMD

WM2091.0

### Notes:

There are three fields in the MQ message descriptor that are used to support application data conversion. The first two fields specify the numeric and character representations of the application data. A message that is put with the initial values of these fields is correctly described. The **Format** field indicates the nature of the application data in a message.



## Requesting application data conversion options

- Request data conversion on MQGET: **MQGMO\_CONVERT**
  - **Encoding** and **CodedCharSetId**
    - On input, requested representation of the message
    - On output, what the application receives
  - Conversion performed based on **Format** field
  - A warning and message is returned in its original form, if the conversion fails

© Copyright IBM Corporation 2014

Figure 9-49. Requesting application data conversion options

WM2091.0

### Notes:

No application data is converted by default. The application requests application data conversion by using the MQGMO\_CONVERT option on the MQGET call.

## What application data conversion can be done

- Some formats are built in, and data conversion is performed by a built-in conversion routine
  - A message that is entirely character data
  - A message structure defined by IBM MQ
- User written data conversion exit is required when:
  - Application defines format of a message, not by IBM MQ
  - Message with a built-in format fails to convert
- IBM MQ utility to help write a data conversion exit

© Copyright IBM Corporation 2014

Figure 9-50. What application data conversion can be done

WM2091.0

### Notes:

A built-in conversion routine can convert a message that consists entirely of characters or a message with a structure defined by MQ.

If the application defines the format of a message, a user-written data conversion exit must convert the data. The `crtmqcvx` utility is supplied with MQ to help in the writing of a data conversion exit.

## Unit summary

Having completed this unit, you should be able to:

- Diagram the connection between two queue managers by using the required components
- Configure IBM MQ channels
- Start and stop channels
- Identify channel states
- Access remote queues
- List considerations for data conversion
- Use the dead-letter queue to find messages that could not be delivered

© Copyright IBM Corporation 2014

Figure 9-51. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. True or false: A transmission queue is required for every remotely connected queue manager.
2. True or false: A common naming convention for a transmission queue is to use the same name as the targeted remote queue manager.
3. What action prompts the channel initiator to start a sender channel?
4. What IBM MQ control command shows the current state of a channel?

© Copyright IBM Corporation 2014

Figure 9-52. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.



## Checkpoint answers

1. **True.** If the remote queue manager is not known, a default transmission queue is used.
2. **True.**
3. When a message is placed on the initiation queue of the sending queue manager.
4. Use the **DISPLAY CHSTATUS** command.

© Copyright IBM Corporation 2014

---

Figure 9-53. Checkpoint answers

WM2091.0

### Notes:

## Exercise 7



Connecting queue managers

© Copyright IBM Corporation 2014  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Figure 9-54. Exercise 7

WM2091.0

### Notes:

In this exercise, you create channels between two queue managers, with one queue manager that simulates a remote queue manager. You use the `amqsput` and `amqsget` sample programs to send and receive messages to and from queues.



## Exercise objectives

After completing this exercise, you should be able to:

- Design a networked IBM MQ architecture that consists of two or more interconnected queue managers
- Use IBM MQ commands to create the channels and supporting objects to implement distributed queuing
- Use the IBM MQ sample programs to test the client connection to IBM MQ

© Copyright IBM Corporation 2014

---

Figure 9-55. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for detailed instructions.

# Unit 10. Implementing basic security

## What this unit is about

This unit describes how IBM MQ uses access control lists (ACLs) to protect its objects, and how the object authority manager (OAM) uses these ACLs whenever a user attempts to access these objects. The unit also describes how to manage MQ object authorizations and introduces Secure Socket Layer (SSL) support.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the role of the object authority manager (OAM) to provide security to MQ resources
- Protect MQ resources by using the OAM
- Use some of the OAM control commands
- Describe the Secure Sockets Layer (SSL) support provided in MQ
- Implement basic channel authentication

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Describe the role of the object authority manager (OAM) to provide security to MQ resources
- Protect MQ resources by using the OAM
- Use some of the OAM control commands
- Describe the Secure Sockets Layer (SSL) support provided in MQ
- Implement basic channel authentication

© Copyright IBM Corporation 2014

---

Figure 10-1. Unit objectives

WM2091.0

### Notes:

## Security mechanisms

- **Identification**

Uniquely identify users of a system or application that is running in the system

- **Authentication**

Prove that a user or application is genuinely that person or what that application claims to be

- **Access control**

Protect resources in a system by limiting access only to authorized users and their applications

- **Confidentiality**

Encrypt messages to protect data

- **Auditing**

Track users and applications that access the system

© Copyright IBM Corporation 2014

Figure 10-2. Security mechanisms

WM2091.0

### Notes:

There are five main requirements for a comprehensive security implementation:

- The ability to uniquely identify users of a system or application
- The ability to prove that a user or application is authentic
- The ability to protect resources by limiting access to authorized users and applications
- The ability to protect confidential data
- The ability to track users and applications that access the system and the data

## IBM MQ security implementations

- OAM installable service
- SSL for channel security
- Channel authentication rules for channel access control
- Connection authorization for queue manager access control
- Connection refusal events

© Copyright IBM Corporation 2014

Figure 10-3. IBM MQ security implementations

WM2091.0

### Notes:

MQ security supports the five security implementations.

The OAM installable service provides authorization for MQI calls, commands, and access to objects to protect the local MQ resources.

The SSL protocol provides industry-standard channel security, with protection against eavesdropping, tampering, and impersonation to control access to the network. An MQ channel can be configured to receive and authenticate an SSL certificate from an SSL client.

MQ channel authentication allows you to provide more precise control over the access that is granted to connecting systems at a channel level.

MQ connection authorization uses the user ID and password that is supplied to check whether a user has authority to access resources.

MQ security provides descriptive connection refusal events, which are written to a channel event queue.

## Planning for security

1. Identify users that need authority to administer IBM MQ
2. Identify applications that need authority to work with IBM MQ objects
3. User ID associated with MCAs authority to access various IBM MQ resources

© Copyright IBM Corporation 2014

Figure 10-4. Planning for security

WM2091.0

### Notes:

You can use MQ for a wide variety of applications on a range of operating systems and hardware. The security requirements are likely to be different for each application.

You must consider certain aspects of security when implementing MQ. If you ignore security aspects and do nothing on some operating systems, such as IBM i, UNIX, Linux, and Windows, you cannot use MQ. On z/OS, the effect of ignoring security is that your MQ resources are unprotected. That is, all users can access and change all MQ resources.

First, identify users that need the authority to administer MQ. These users are the users that need to be able to enter commands and use MQ Explorer to access queue managers, queues, channels, and processes.

Second, identify applications that need to access to MQ queue managers, queues, processes, namelists, and topics by using MQI calls.

Third, identify the user IDs that are associated with the applications that need to administer channels and channel initiators, and open transmission queues.

For more information about security planning, see the IBM MQ product documentation.

## IBM MQ access control overview

- Granular access control facilities
  - Provided by IBM MQ installable services
  - Which user? Which resource? What types of access?
  - Channel access control that is based on IP address, queue manager, SSL distinguished name, and asserted identity
- IBM MQ access control at user and group level
- Alternate user IDs can be specified when suitably authorized
- User needs access to the first named resource and not the alias queue or remote queue resolved resource

© Copyright IBM Corporation 2014

Figure 10-5. IBM MQ access control overview

WM2091.0

### Notes:

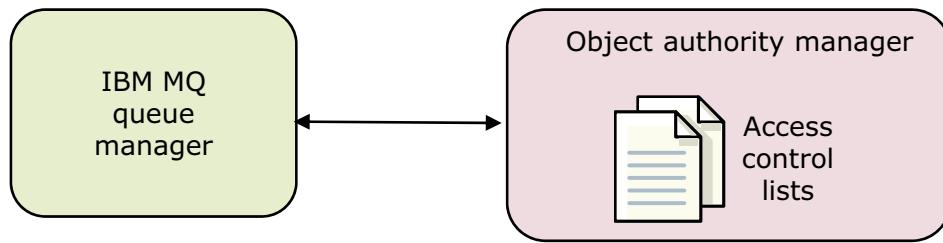
MQ access control is the primary security component. Access control allows MQ to control which users and applications are granted which types of access to which MQ resources. Resources that might be controlled in this way are the queue manager, queues, and processes.

All distributed queue managers provide access control facilities to control which users have access to which MQ resources. The distributed queue managers use the installable services component of MQ to provide access control for MQ resources.

MQ captures the user ID that is associated with an application with the MQCONN call. This user ID is used for the access control checks. It is possible for authorized users to use an alternative user ID instead of the logged on user ID. The name that is specified in the IBM MQ API command is used when MQ checks to see whether a user is authorized to access a particular resource.

For an alias or remote queue definition, the user needs access to the queue that is specified in the IBM MQ API command and not the resolved name. For model queues, there might be instances where MQ generates the name of the dynamic queue. In this case, the user ID creating a dynamic queue is automatically given full access rights to the queue.

## OAM installable service



- IBM MQ authorization service component
- Common access control list (ACL) manager for distributed queue managers
- Authorizations can be granted or revoked at the principal or group level
- IBM MQ Explorer and IBM MQ control commands can be used to configure and manage ACLs

© Copyright IBM Corporation 2014

Figure 10-6. OAM installable service

WM2091.0

### Notes:

The MQ OAM component is an MQ installable service.

The MQ OAM provides a full set of access control facilities for MQ including access control checking and commands to set, change, and inquire on MQ access control information.

Access to MQ entities is controlled through MQ user groups and the OAM. A command interface and MQ Explorer can be used to identify administrators, and to grant or revoke authorizations as required.

The authorization service maintains an access control list (ACL) for each MQ object to which it is controlling access. An ACL contains a list of all the group IDs that can access the object; on Windows, the ACL can contain user IDs and group IDs.

Administrators can use MQ Explorer and control commands to configure and manage ACLs objects.

## Principals and groups

- On UNIX and Linux
  - Principal is a user ID, or an ID that is associated with an application program that is running on behalf of a user
  - Group is a system-defined collection of principals.
  - ACLs are based on groups by default
  - Queue manager can be configured to support principals at creation with `-oa user` option of by editing `qm.ini`
- On Windows
  - Principal is a Windows user ID, or an ID that is associated with an application program that is running on behalf of a user
  - Group is a Windows group
  - ACLs are based on user IDs and groups
- Changes to a principal's group membership are not recognized until the queue manager is restarted or a **REFRESH SECURITY MQSC** command is executed

© Copyright IBM Corporation 2014

Figure 10-7. Principals and groups

WM2091.0

### Notes:

Principals can belong to groups. You can grant access to a particular resource to groups rather than to individuals to reduce the amount of administration required. For example, you might define a group that consists of users who want to run a particular application. Other users can be given access to all the resources they require by adding their user ID to the appropriate group.

On UNIX and Linux, all ACLs are based on groups by default. You can change the queue manager to support principals, similar to Windows.

On Windows, ACLs are based on user IDs and groups.

Any changes that you make to a principal's group membership are not recognized until the queue manager is restarted, or you enter the **REFRESH SECURITY MQSC** command.

## Access control with the OAM

- Access control lists are specific to IBM MQ
  - Not integrated with system-level security
  - Changes to user's operating system authority is not recognized until queue manager restart or security refresh
  - One ACL for each queue manager; not shared between queue managers
- Access control for IBM MQ objects
  - Queue manager
  - Queues
  - Processes
  - Namelists
  - Channels
  - Authentication information objects
  - Listeners
- OAM can be disabled
  - Remove entry from `mqsc.ini` or Windows registry
  - Not recommended
  - Difficult to reestablish uniform checking of authority

© Copyright IBM Corporation 2014

Figure 10-8. Access control with the OAM

WM2091.0

### Notes:

There is a set of access control lists for each queue manager, so ACLs cannot be (automatically) shared with multiple queue managers. The ACLs are not integrated with system-level security.

The OAM provides access control facilities only for MQ objects: the queue manager, all queues, processes, namelists, channels, authentication information objects, listeners, and services.

It is possible, though not good practice, to disable access control checking by setting the appropriate environment variable or deleting the authorization service stanza in the queue manager configuration file `qm.ini` (or Registry on Windows systems).

If the OAM is disabled, then access permissions, which are normally automatically set when objects are created, are no longer created. If the OAM is enabled later, these permissions do not exist and must be created manually.

## OAM access control lists

- One authorization file for each object plus global permissions files
- Windows OAM bypasses authorization files for certain classes of principal:
  - SYSTEM
  - Local “Administrators” group
  - Local “mqm” group
- Uses IBM MQ object authorizations
  - Context, such as passall, passid
  - MQI, such as browse, put, get, set
  - Administration, such as dsp, chg, dlt
  - Generic, such as all, alladm, allmqi, none

© Copyright IBM Corporation 2014

Figure 10-9. OAM access control lists

WM2091.0

### Notes:

Each file contains a set of access control stanzas. There is one stanza per principal for which access is controlled, where a principal is either a user ID or a group.

The principals (user IDs, groups, or both) which have access to the appropriate object are listed in this file. This listing includes a bit string (in hex) that represents the access rights that are associated with that entity.

Certain principals or groups are granted automatic access to resources:

- Members of the “mqm” group or the “mqm” user
- On Windows:
  - Administrator user and local group
  - SYSTEM user ID
  - The user or principal group that creates a resource

The authorizations that can be given are categorized as follows:

- Authorizations for sending MQI calls
- Authorizations for MQI context
- Authorizations for entering commands for administration tasks
- Generic authorizations

## Set or reset authorization

- Use **setmqaut** command to set or reset authorization by object type
  - Prefix authorization with a plus sign (+) to add
  - Prefix authorization with a minus sign (-) to revoke
- Command is cumulative
  - Set authorization explicitly on each **setmqaut** command to avoid retaining unwanted pre-existing authorities
  - Granting and revoking is achieved by specifying **-all** to remove all authorization and then granting required authorizations
- Use **dspmqaut** command to display and verify authorizations

Format: **setmqaut -m QMgr -t Objtype -n Profile  
[-p Principal | -g Group] permissions**

Example: **setmqaut -m JUPITER -t queue -n MOON.\* -g VOYAGER  
+browse +get -put**

© Copyright IBM Corporation 2014

Figure 10-10. Set or reset authorization

WM2091.0

### Notes:

Three control commands provide control of the security environment for MQ: **setmqaut**, **dspmqaut**, and **dmpmqaut**. These programs require a connection to the authorization service; they can be used when the target queue manager is active and the OAM is enabled. All of the responses to these commands are displayed on the screen.

The **setmqaut** command sets the access to a particular resource by a principal or group. This command can be used to add or revoke privileges.

The **setmqaut** command can use generic profiles. In the example, the **setmqaut** command allows members of the group VOYAGER to get and browse but not put messages on the queues that start with the characters **MOON**. that the JUPITER queue manager owns.

If you use the **setmqaut** command to set authorizations for an individual user ID, the authorizations are held at the level of the individual user ID. For MQ on UNIX however, authorizations are held at the level of the primary group of the user ID and so all members of that group acquire the same authorizations.

## Display authorization

- Use the **dsprmqaout** command to verify that object authorization is correct
  - By object type
  - For a principal or group

Format:

```
dsprmqaout -m QMgr -t ObjType -n ObjName
[-p Principal | -g Group] [-s Service]
```

Example:

```
dsprmqaout -m SATURN -t q -n APPL.Q1 -p mquser
Entity mquser has the following authorizations for object
APPL.Q1:
  get
  browse
  ...
```

© Copyright IBM Corporation 2014

Figure 10-11. Display authorization

WM2091.0

### Notes:

The **dsprmqaout** command displays current authorizations for MQ objects that include queues, queue managers, and processes.

This command does support generic profiles. If a user ID is a member of one or more groups, the display authorization command displays the combined authorizations of all the groups.

Only one group or principal can be specified.

The example command displays the authorizations for the queue that is named APPL.Q1 on queue manager SATURN for the user that is named “mquser”.

## Create authorization report

- Use **dmpmqaut** command to generate a report of current authorizations
  - By object type
  - For a principal or group

Format:

```
dmpmqaut -m Qmgr -t Objtype [-n Profile | -1]
[-p Principal | -g Group]
```

Example:

```
dmpmqaut -m qm1 -t q -n a.b.c -p user1
```

```
profile: a.b.*  
object type: queue  
entity: user1  
type: principal  
authority: get, browse, put, inq
```

© Copyright IBM Corporation 2014

Figure 10-12. Create authorization report

WM2091.0

### Notes:

Use MQ **dmpmqaut** control command to create a report of the current authorizations that are associated with a specified profile.

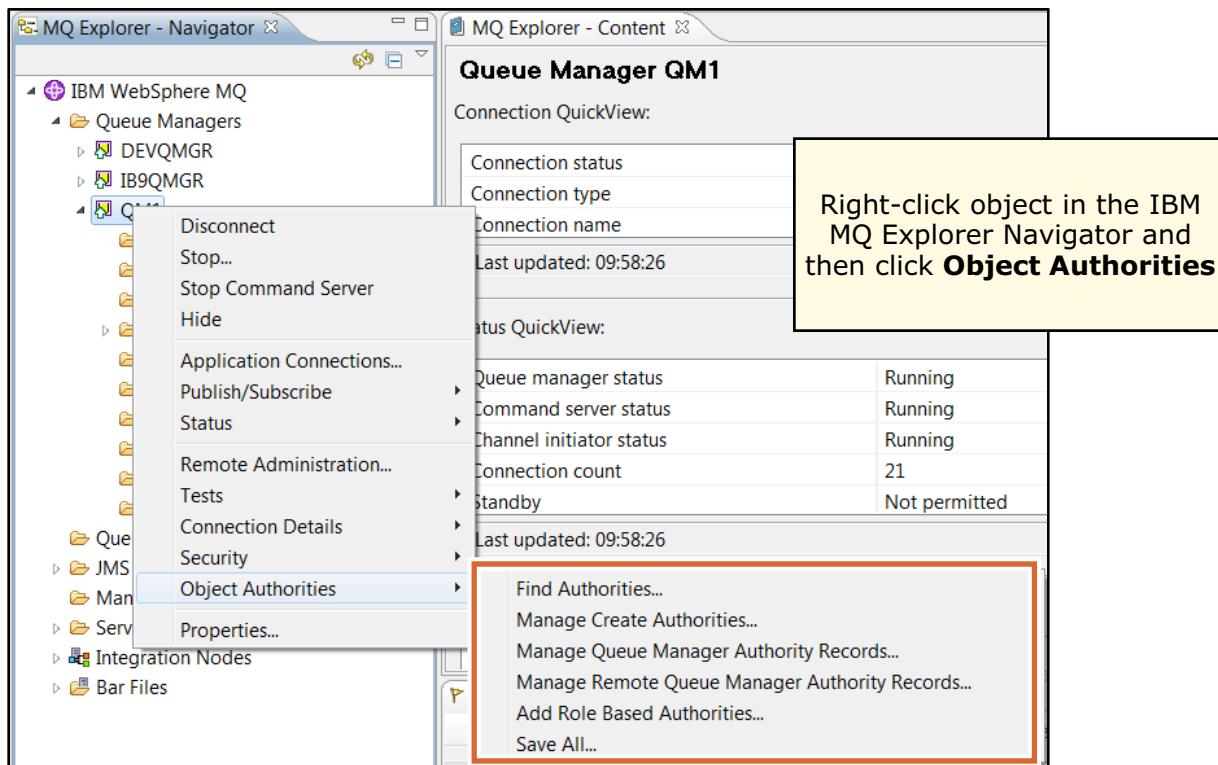
The **-1** parameter allows you to create a report with a terse list of all profiles names and types.

Group names must exist and you can specify one group name on the **dmpmqaut** command. MQ for Windows allows the use of local groups only.

The example in the figure creates a report that shows the object authority for the queues on queue manager **qm1** for the user **user1**.



## Managing authorization in IBM MQ Explorer



© Copyright IBM Corporation 2014

Figure 10-13. Managing authorization in IBM MQ Explorer

WM2091.0

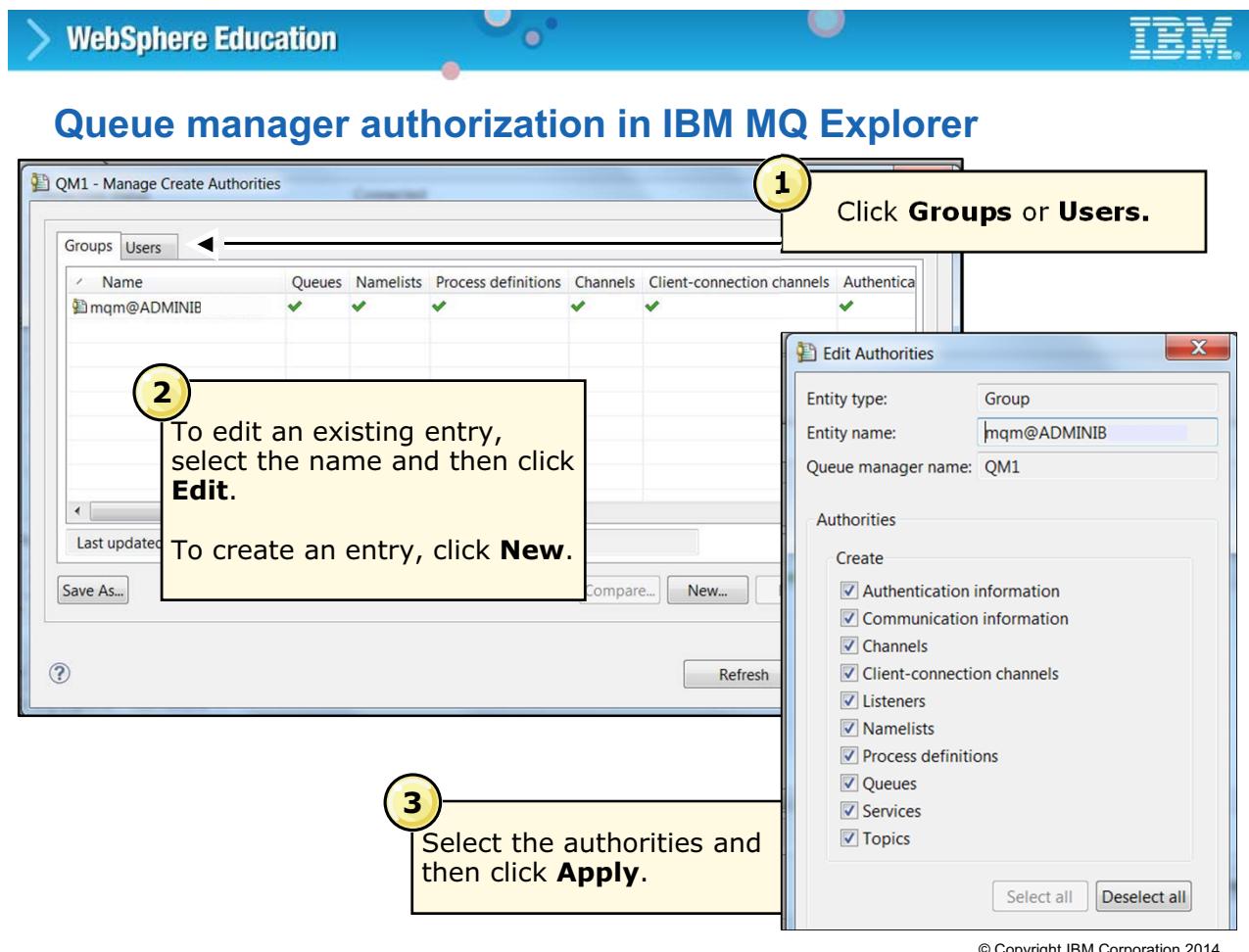
### Notes:

You can also manage object authorities with MQ Explorer.

In MQ Explorer, you can:

- Find authorities
- Manage “create” authorities
- Manage queue manager authority records
- Manage remote queue manager authority records
- Add role-based authorities

To manage the authority records for a queue manager, right-click the queue manager and then click **Object Authorities**.



© Copyright IBM Corporation 2014

Figure 10-14. Queue manager authorization in IBM MQ Explorer

WM2091.0

## Notes:

To modify an existing authority record:

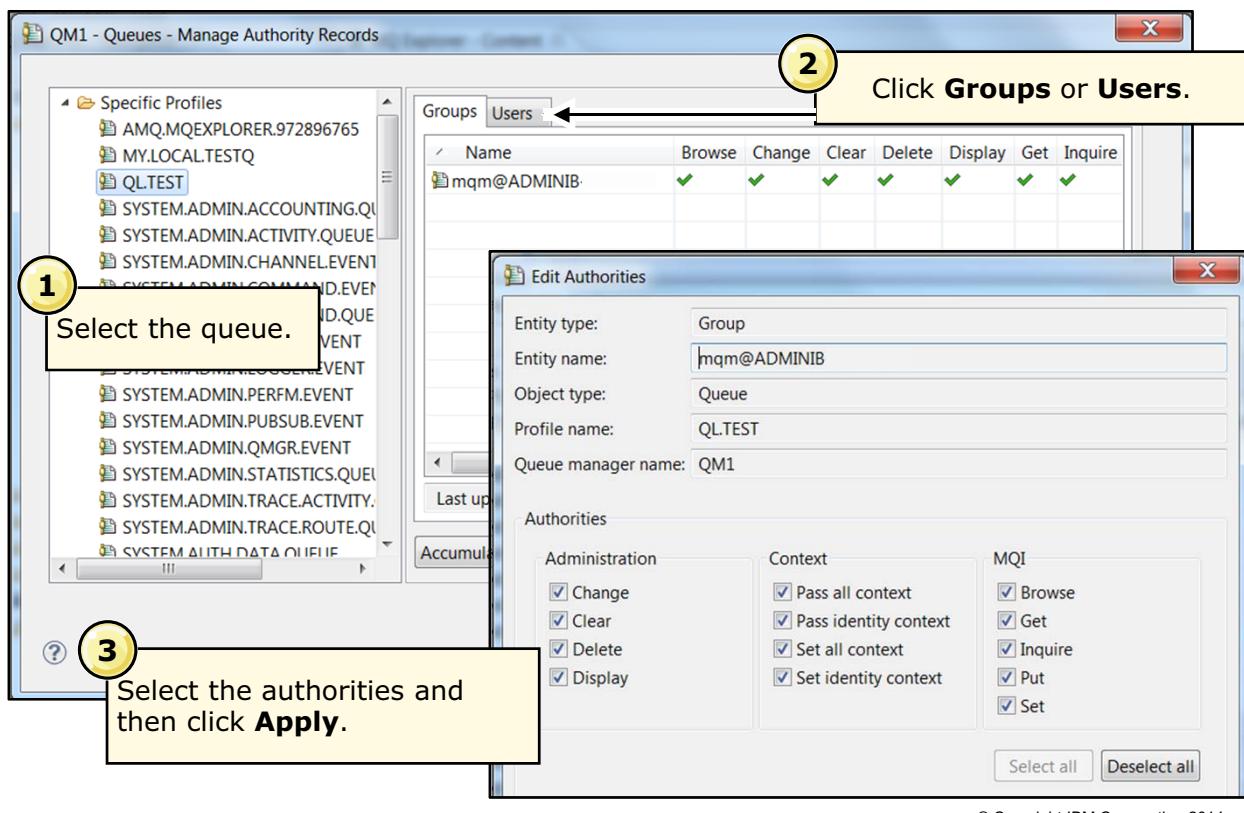
1. Click the **Groups** tab to modify a group record, or click the **Users** tab to modify the record for a specific user.
2. Select the group or user and then click **Edit**.
3. Modify the authorities and then click **OK**.

To create a new authority record:

1. Click the **Groups** tab to add a record for a group, or click the **Users** tab to add a record for a specific user.
2. Click **New**.
3. Enter an entity name, select the authorities, and then click **OK**.



## Queue authorization in IBM MQ Explorer



© Copyright IBM Corporation 2014

Figure 10-15. Queue authorization in IBM MQ Explorer

WM2091.0

### Notes:

You can modify or add queue authority records in MQ Explorer, by right-clicking the queue in the **Queue** content view and then clicking **Object Authorities > Manage Authority Records**.

On the Manage Authority Records view:

1. Select the queue.
2. Click the **Groups or Users** tab. Select the record and then click **Edit** to modify an existing record, or click **Add** to add a record.
3. Select the authorities and then click **Apply**.

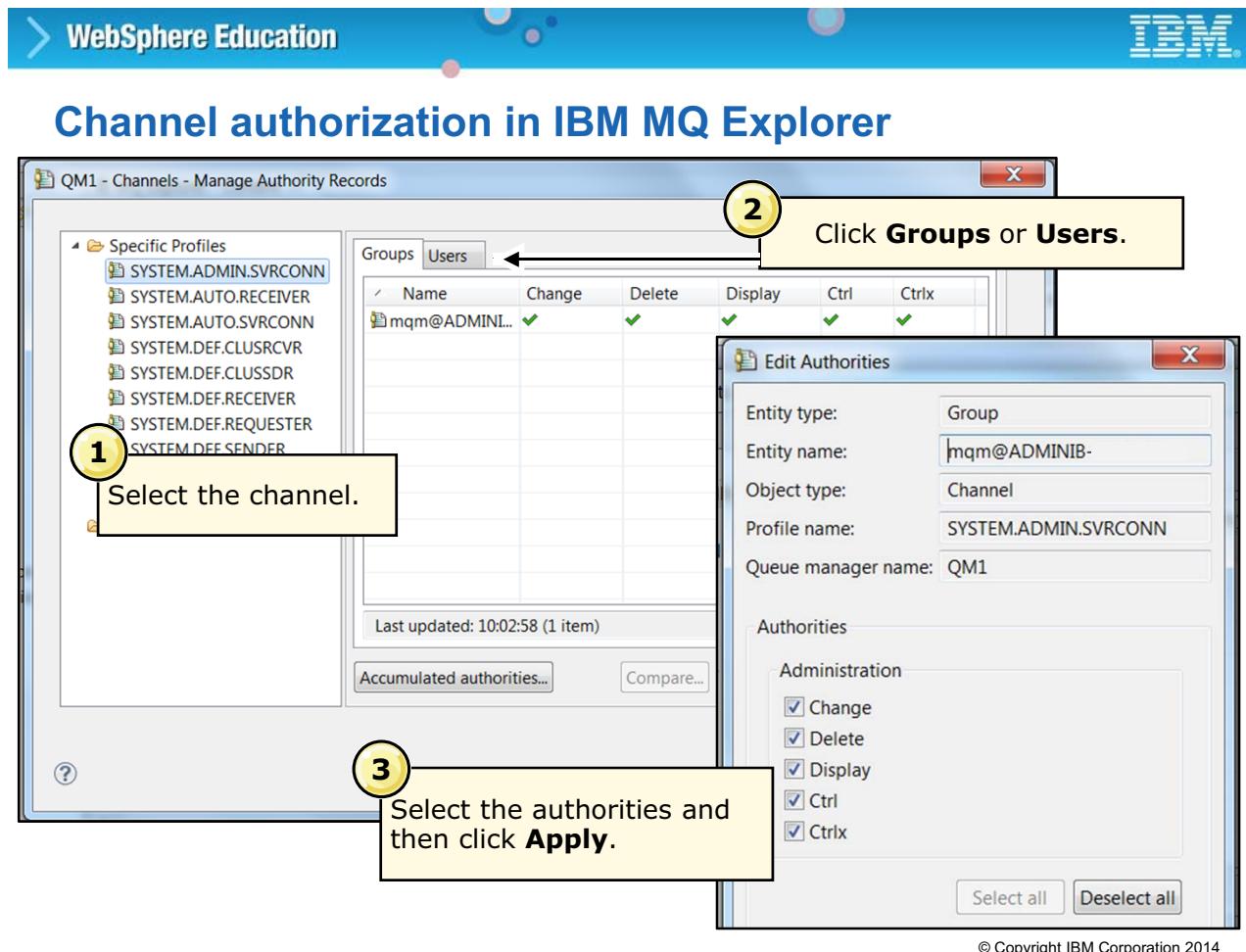


Figure 10-16. Channel authorization in IBM MQ Explorer

WM2091.0

## Notes:

You can also add or modify channel authority records in MQ Explorer by using the same technique that is used to add or modify queue authority records.

Right-click the channel in the **Channels** content view and then click **Object Authorities > Manage Authority Records**.

On the Manage Authority Records view:

1. Select the channel.
2. Click the **Groups** or **Users** tab. Select the record and then click **Edit** to modify an existing record, or click **Add** to add a record.
3. Select the authorities and then click **Apply**.

## Access control for IBM MQ control programs

- Restrictive use of most IBM MQ control programs  
Example: `crtmqm`, `strmqm`, `runmqsc`, `setmqaut`
- UNIX restricts users to “mqm” group
  - Configuration as a part of IBM MQ installation
  - Control that the operating system imposes, not OAM
- Windows allows:
  - “mqm” group
  - “Administrators” group
  - System user ID

© Copyright IBM Corporation 2014

Figure 10-17. Access control for IBM MQ control programs

WM2091.0

### Notes:

Access control can be configured for most MQ control programs. Control programs include commands for creating queue managers, starting queue managers, running MQSC, and setting authorization.

By default, UNIX restricts access to control programs to members of the “mqm” group.

By default, Windows restricts access to control programs to members of the “mqm” and “Administrators” groups, and the Windows System ID.

## Security and distributed queuing

- Put authority option for the receiving end of a message channel
  - Default user identifier is used
  - Context user identifier is used
- Transmission queue
  - Messages that are destined for a remote queue manager are put on a transmission queue by the local queue manager
  - An application does not normally need to put messages directly on a transmission queue, or need authorization to do so
  - Only special system programs should put messages directly on a transmission queue and have the authorization to do so

© Copyright IBM Corporation 2014

Figure 10-18. Security and distributed queuing

WM2091.0

### Notes:

On the receiving end of a message channel, you can specify the user ID to use for checking the authority of the receiving MCA to open a destination queue.

You can choose one of the following options:

- **Default user identifier**

Use the receiving MCAs default user identifier. A security exit or setting the MCAUSER attribute in the channel definition at the receiving end of the message channel can change this user identifier.

- **Context user identifier**

Use the user identifier in the context of the message.

Allow special system programs only to put messages directly on a transmission queue.

## Security authorization for remote queues

- Distributed platforms have authorizations for remote and clustered queues
- For applications that explicitly open `queue@qmgr`, which is a common pattern when using reply to information

Example:

```
setmqaut -m QM1 -t rqmname -n QM2 -p mquser +put
```

© Copyright IBM Corporation 2014

Figure 10-19. Security authorization for remote queues

WM2091.0

### Notes:

MQ supports security authorization for remote and clustered queues by using a remote queue manager object that the OAM recognizes.

An example of a command to set security authorization for non-local queue is provided in the figure. The example gives the user “mquser” put authority on the remote queue that is named QM2.

When an application is attempting to access an MQ object, the general rule is that its authority is checked on the first object in the resolution path. For example, if the object descriptor supplies the name of a remote queue and remote queue manager, authority checking occurs on the transmission queue, which has the same name as the remote queue manager. If the application attempts to open a local definition of a remote queue, authority checking occurs against that object. For an alias queue, authority checking occurs at the level of the alias queue, not at the level of the queue to which it resolves.

Limit the ability to define queues to privileged users. Otherwise, normal access control can be bypassed by creating an alias queue. The use of the `+crt` authorization on the `setmqaut` control command allows you to specify which users are allowed to create queues.

Example: `setmqaut -m QMC01 -t queue -g GROUPB +crt`

## Authorization checking in the MQI

- MQI calls with security checking
  - MQCONN and MQCONNX
  - MQOPEN
  - MQPUT1 (implicit MQOPEN)
  - MQSUB
  - MQCLOSE (for dynamic queues)
- IBM MQ events are written to SYSTEM.ADMIN.QMGR.EVENT queue as audit records
- Reason code **MQRC\_NOT\_AUTHORIZED** is returned if not authorized

© Copyright IBM Corporation 2014

Figure 10-20. Authorization checking in the MQI

WM2091.0

### Notes:

This figure lists the MQI calls with security checking.

Applications that send the MQCONN or MQCONNX call must be authorized to connect to the queue manager.

For MQOPEN and MQPUT1, the authority check is made on the name of the object that is opened, and not on the name, or names, resulting after a name is resolved. For example, an application might be granted authority to open an alias queue without having authority to open the base queue to which the alias resolves. The rule is to check the first definition that is encountered during the process of resolving a name that is not a queue manager alias, unless the queue manager alias definition is opened directly.

When an MQSUB call is sent, the queue manager verifies that the user identifier under which the application is running has the appropriate level of authority to subscribe to the topic object.

If the MQCLOSE call has options for deleting and purging permanent dynamic queues. If one of these options is set, there is a check to determine whether the user is authorized to delete the queue. This check is not done if the application that created the queue is attempting to delete the queue.

If authorization fails for any of the MQI calls, an MQRC\_NOT\_AUTHORIZED event is written to the SYSTEM.ADMIN.QMGR.EVENT event queue. This event indicates that a command is entered by a user ID that is not authorized to access the object that is specified in the command.



## Channel authentication control

- Enabled by default
- Rules are based on:
  - Connecting IP address
  - Connecting queue manager name
  - SSL distinguished name
  - Asserted identity (including \*MQADMIN option)
  - Derived identity from distinguished name mapping
- Rules can be applied in WARNING mode to allow connection but generate errors

© Copyright IBM Corporation 2014

Figure 10-21. Channel authentication control

WM2091.0

### Notes:

In MQ, channel authorization (CHLAUTH) records define the rules that are applied when a queue manager or client attempts to connect through a channel.

Channel authentication uses rules to control access to a channel. A wildcard can be used with the MQ administrator ID (\*MQADMIN) in these rules to cover the use of any ID that would otherwise gain automatic administrative rights over a queue manager. Having a generic user ID, such as \*MQADMIN, makes it easier to have the same rules on all operating systems, where the actual definition of who is an administrator might vary.

Rules can also be defined as “WARN”, generating authorization events, without blocking the connection. This behavior might help when changing to a secure environment, by not turning off connections immediately.

## Channel authentication commands

- Use the MQSC command **SET CHLAUTH** to create or modify a channel authentication record
- Use the MQSC command **DISPLAY CHLAUTH** to test rules that you define
- Use the MQSC command **ALTER QMGR CHLAUTH(DISABLED)** to disable channel authentication

© Copyright IBM Corporation 2014

Figure 10-22. Channel authentication commands

WM2091.0

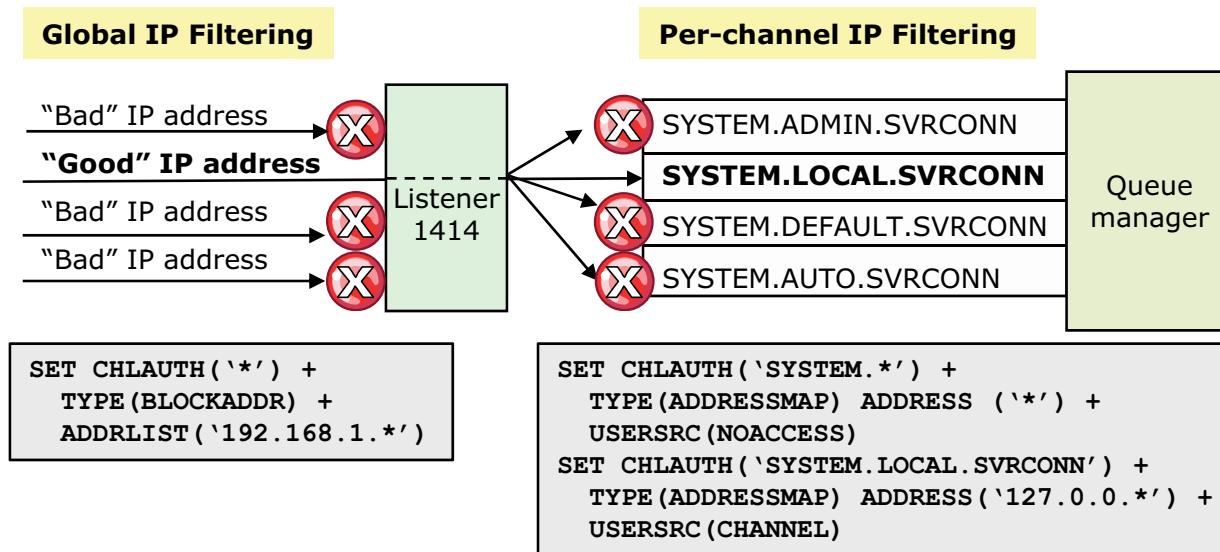
### Notes:

Use the MQSC command **SET CHLAUTH** to configure channel authentication control.

Use the MQSC command **DISPLAY CHLAUTH** with the **MATCH(RUNCHECK)** option to verify a simulated connection. Rules can be tested from the console without making a real connection.

If necessary, such in a development environment, you can disable channel authentication by using the MQSC command **ALTER QMGR CHLAUTH(DISABLED)**

## Channel authentication example



- Filter connection requests based on IP address of requester
- Per-channel rules match the least-specific to most-specific
- Global blocking rules occur at the listener before the channel name is known and take precedence over per-channel rules

© Copyright IBM Corporation 2014

Figure 10-23. Channel authentication example

WM2091.0

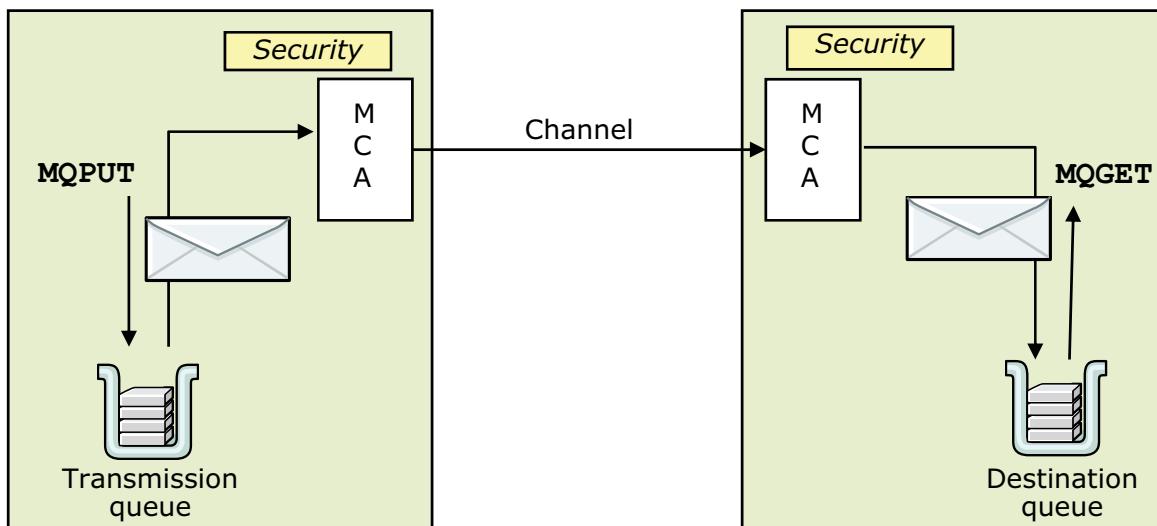
### Notes:

In the **SET CHLAUTH** command, the address list (**ADDRLIST**) is the address from which you are refusing connections. The address can be a specific IP address or a pattern that includes the asterisk (\*) as a wildcard or the hyphen (-) to indicate a range that matches the address. In this example in the figure, the **SET CHLAUTH** command on left blocks all IP addresses beginning with 192.168.1.

In the example on the right, the first **SET CHLAUTH** command restricts access to all SYSTEM queues from any IP address because the **USERSRC** parameter is set to **NOACCESS**. The **NOACCESS** option means that inbound connections that match this mapping do not have access to the queue manager and the channel ends immediately.

The second command gives the local host (127.0.0) access to the channel that MQ Explorer uses because the **USERSRC** parameter is set to **CHANNEL**. The **CHANNEL** option means that inbound connections that match this mapping use the flowed user ID or any user that is defined on the channel object in the **MCAUSER** field.

## IBM MQ security exits



- Allows an MCA to authenticate its partner
- Usually work in pairs at each end of channel
- Called immediately after the initial data negotiation completes on channel startup, but before any messages start to flow
- Format of a security message and security exit program are defined by the user

© Copyright IBM Corporation 2014

Figure 10-24. IBM MQ security exits

WM2091.0

### Notes:

As an option, you can use a channel security exit for added security.

A security exit forms a secure connection between two security exit programs, where one program is for the sending message channel agent (MCA), and one is for the receiving MCA.

Security exits normally work in pairs, one at each end of a channel. They are called immediately after the initial data exchange negotiation completes on channel startup, but before any messages start to flow. One possible outcome of the conversation between the security exits is that the channel is closed and message flow is not allowed to proceed.

The name of the security exit is specified as a parameter on the channel definition at each end of a channel.



**Note**

Try using MQ options such as SSL or connection authentication before writing a custom exit.

## Connection authentication

- Client application can provide a user ID and password
- Queue manager can be configured to act on a supplied user ID and password
- External repository can be used to determine whether a user ID and password combination is valid
- Application user ID, which is the usual operating system user ID presented to IBM MQ, might be different from the user ID provided by the application

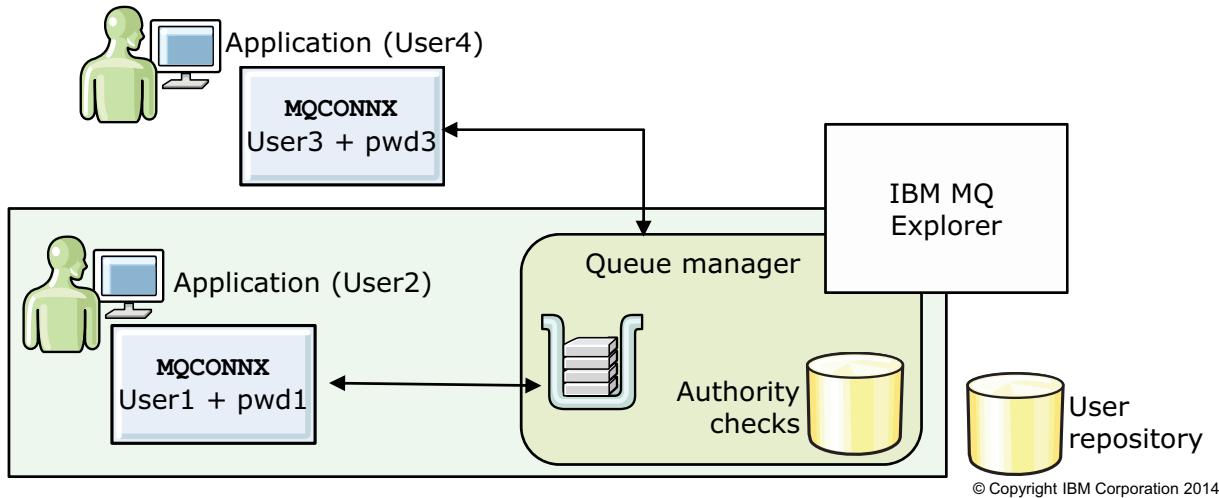


Figure 10-25. Connection authentication

WM2091.0

### Notes:

Connection authentication can be achieved in various ways:

- An application can provide a user ID and password. The application can be either a client, or it can use local bindings.
- A queue manager can be configured to act on a supplied user ID and password.
- A repository can be used to determine whether a user ID and password combination is valid.

In the diagram, two applications are making connections with a queue manager, one application as a client and one using local bindings. Applications might use various APIs to connect to the queue manager, but all can provide a user ID and a password. The user ID that the application is running under, User1 and User3 in the diagram, might be different from the user ID that is provided by the application (User2 and User4).

## Enabling connection authentication on a queue manager

- Define the authentication information object with the **DEFINE AUTHINFO** command
  - **IDPWOS** Indicates that the queue manager uses the local operating system to authenticate the user ID and password
  - **IDPWLDAP** Indicates that the queue manager uses an LDAP server to authenticate the user ID and password
- Set the **CONNAUTH** attribute on the queue manager to the name of an authentication information object
- Use the **REFRESH SECURITY** command to refresh the cached view of the configuration for connection authentication

© Copyright IBM Corporation 2014

Figure 10-26. Enabling connection authentication on a queue manager

WM2091.0

### Notes:

To configure a queue manager to use a supplied user ID and password to check whether a user has authority to access resources, enable connection authentication on the queue manager.

First, define an authentication object. You can define the authentication object to use the local operating system to authenticate the user ID and password or to use an LDAP server to authenticate the user ID and password.

After you define the authentication object, set the **CONNAUTH** attribute on the queue manager to the name of an authentication information object.

You must refresh the configuration before the queue manager recognizes the changes.

## Enabling connection authentication examples

Example 1: Define an authentication object that uses the local file system to authenticate the user ID and password.

```
DEFINE AUTHINFO (USE.OS) AUTHTYPE (IDPWOS)
ALTER QMGR CONNAUTH (USE.OS)
REFRESH SECURITY TYPE (CONNAUTH)
```

Example 2: Define an authentication object that uses an LDAP server to authenticate the user ID and password.

```
DEFINE AUTHINFO (USE.LDAP) AUTHTYPE (IDPWLDAP) +
CONNNAME ('ldap1(389),ldap2(389)') +
LDAPUSER ('CN=QMGR1') LDAPPWD ('passw0rd')
ALTER QMGR CONNAUTH (USE.LDAP)
REFRESH SECURITY TYPE (CONNAUTH)
```

© Copyright IBM Corporation 2014

Figure 10-27. Enabling connection authentication examples

WM2091.0

### Notes:

This figure provides two connection authentication examples.

In Example 1:

- The first MQSC command defines an authentication object that is named USE.OS. The AUTHTYPE(IDPWOS) attribute directs this authentication object to use the local operating system to authenticate the user ID and password.
- The second MQSC command enables connection authorization on the queue manager by using the USE.OS authentication object.
- The third MQSC command refreshes connection authorization security on the queue manager so that the queue manager recognizes the changes.

In Example 2:

- The first MQSC command defines an authentication object that is named USE.LDAP that uses an LDAP server to authenticate the user ID and password.
- The second MQSC command enables connection authorization on the queue manager by using the USE.LDAP authentication object.

- The third MQSC command refreshes connection authorization security on the queue manager so that the queue manager recognizes the changes.

## Secure sockets layer (SSL)

- Protocol that allows transmission of secure data over an insecure network
- Combines these techniques:
  - Symmetric and secret key encryption
  - Asymmetric and public key encryption
  - Digital signature
  - Digital certificates
- Protection
  - Client/server
  - Queue manager and queue manager channels
- Combats security problems
  - Eavesdropping: Encryption techniques
  - Tampering: Digital signature
  - Impersonation: Digital certificates

© Copyright IBM Corporation 2014

Figure 10-28. Secure sockets layer (SSL)

WM2091.0

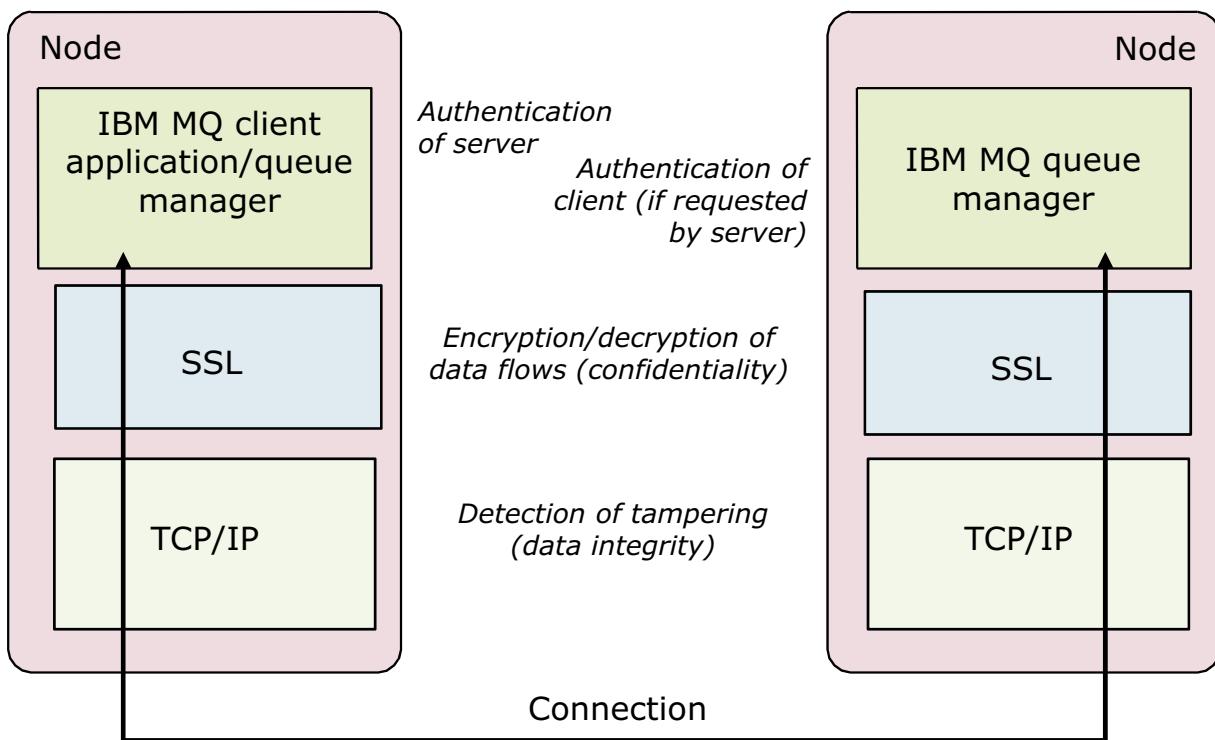
### Notes:

SSL is an industry-standard protocol for secure communications, involving encryption, authentication, and integrity of data. SSL is supported in both client/server and queue manager/queue manager channels (including clusters). There are many flexible capabilities built-in, including the ability to select who are prepared to accept communications from based on their fully authenticated identity. This feature removes, for many people, the requirement to set up channel exits, where they were used for security purposes.

SSL, widely accepted in the Internet community, was subjected to significant testing by the hacker community.

SSL uses a combination of public key and symmetric key encryption to ensure message privacy. Before exchanging messages, an SSL server and SSL client do an electronic handshake during which they agree to use a session key and encryption algorithm. All messages between the client and the server are then encrypted. Encryption ensures that the messages remain private even if eavesdroppers intercept it.

## IBM MQ SSL support



© Copyright IBM Corporation 2014

Figure 10-29. IBM MQ SSL support

WM2091.0

### Notes:

MQ provides the following support for SSL:

- Authentication of the server on the client
- Authentication of the client, if the server requests it
- Encryption/decryption of data flows
- Detection of tampering

## Enabling SSL on the queue manager

- Use IBM MQ Explorer or the MQSC command **ALTER QMGR**

<b>SSLCRNLN</b>	Name of a namelist of authentication information objects that provide certificate revocation locations (CRLs) to allow enhanced TLS/SSL certificate verification
<b>SSLCRYP</b>	Name of the parameter string that is required to configure the cryptographic hardware present on the system
<b>SSLKEYR</b>	Name of the SSL key repository
<b>SSLTASKS</b>	Number of server subtasks to use for processing SSL calls
<b>SSLEV</b>	Enables or disables SSL event messages
<b>SSLFIPS</b>	Specifies whether only FIPS-certified algorithm is used if cryptography is carried out in IBM MQ, rather than in cryptographic hardware

© Copyright IBM Corporation 2014

Figure 10-30. Enabling SSL on the queue manager

WM2091.0

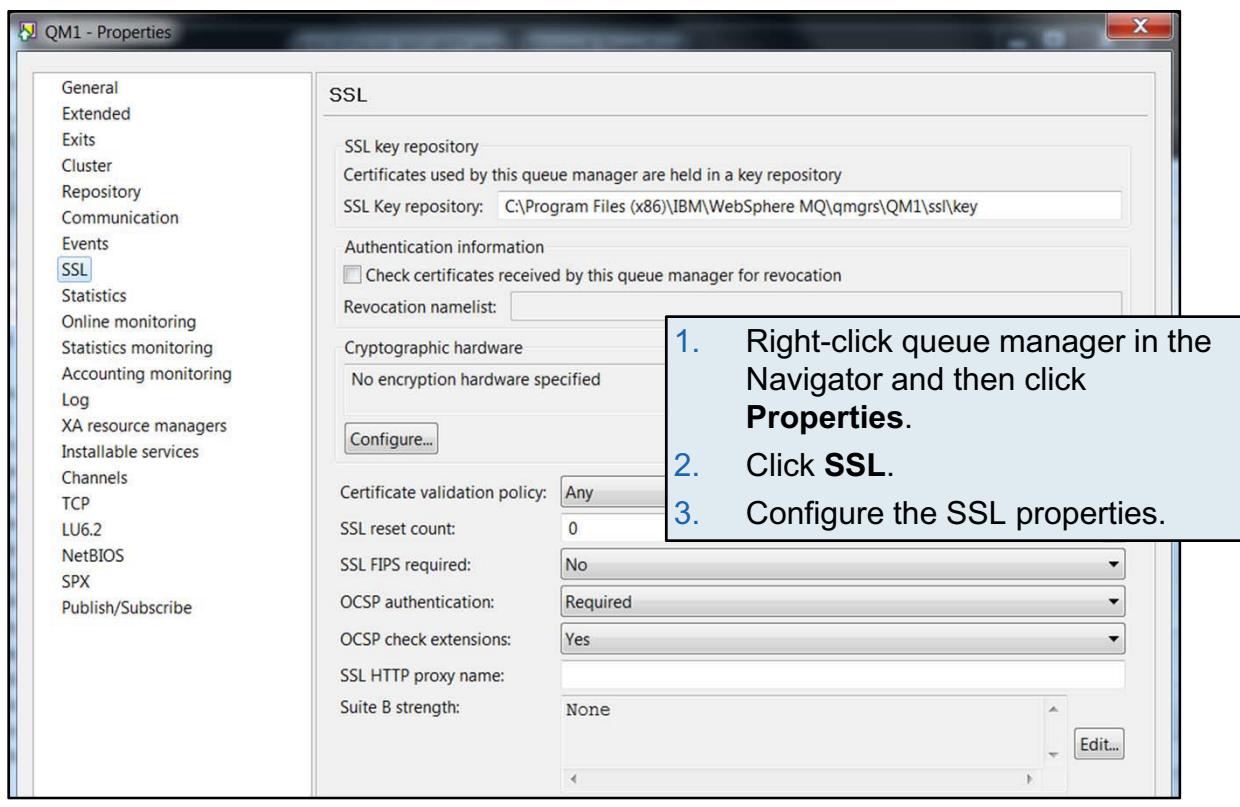
### Notes:

The figure lists the queue manager attributes that can be modified for SSL by using the **ALTER QMGR** command.

- **SSLKEYR** (SSL key repository), holds the name of the SSL key repository.
- **SSLCRNLN** (SSL CRL namelist), holds the name for the namelist of authentication information objects.
- **SSLCRYP** (SSL CryptoHardware), holds the name of the parameter string that is required to configure the cryptographic hardware present on the system. This parameter applies to UNIX only.
- **SSLTASKS** (SSL tasks), holds the number of server subtasks to use for processing SSL calls. If you use SSL channels, you must provide two for these tasks.
- **SSLEV** enables or disables the sending of SSL event messages to the channel event queue, when a channel fails to establish an SSL connection.
- **SSLFIPS** specifies whether only FIPS-certified algorithms are used if cryptography is carried out in MQ (UNIX and Windows only).



## Enabling SSL by using IBM MQ Explorer



© Copyright IBM Corporation 2014

Figure 10-31. Enabling SSL by using IBM MQ Explorer

WM2091.0

### Notes:

You can also use MQ Explorer to configure SSL on a queue manager.



## Channel attributes for SSL

- Specify cipher specification to use when communicating with the queue manager and ensure that it matches the cipher specification on the target channel.

- Use IBM MQ Explorer or the MQSC command **ALTER CHANNEL**

**SSLCIPH** Defines a single cipher specification for an SSL connection

**SSLPEER** Specifies distinguished name that is used in SSL channel negotiation

**SSLCAUTH** Specifies whether the channel uses SSL for client authentication



© Copyright IBM Corporation 2014

Figure 10-32. Channel attributes for SSL

WM2091.0

### Notes:

You can configure the cipher specification to use when communicating with the queue manager by using the MQSC commands **DEFINE CHANNEL** or **ALTER CHANNEL**, or by using MQ Explorer.

The figure lists the attributes available with the **DEFINE CHANNEL** and **ALTER CHANNEL** command for use with SSL and includes an example of the SSL properties in MQ Explorer.

- SSLCIPH** specifies the encryption strength and cipher specifications, for example `NONE_MD5` or `RC4_MD5_US`. The cipher specifications must match at both ends of the channel.
- SSLPEER** specifies the distinguished name (unique identifier) allowed.
- SSLCAUTH** defines whether MQ requires and validates a certificate from the SSL client.

## Default security configuration in IBM MQ Explorer

- Use the **Client Connections** tabs to set default security options on all new client connections
- Default values can be overridden when a new queue manager is added



© Copyright IBM Corporation 2014

Figure 10-33. Default security configuration in IBM MQ Explorer

WM2091.0

### Notes:

Client security configuration support is provided in MQ Explorer. The default security preferences for client connections are part of the **Preferences** dialog box.

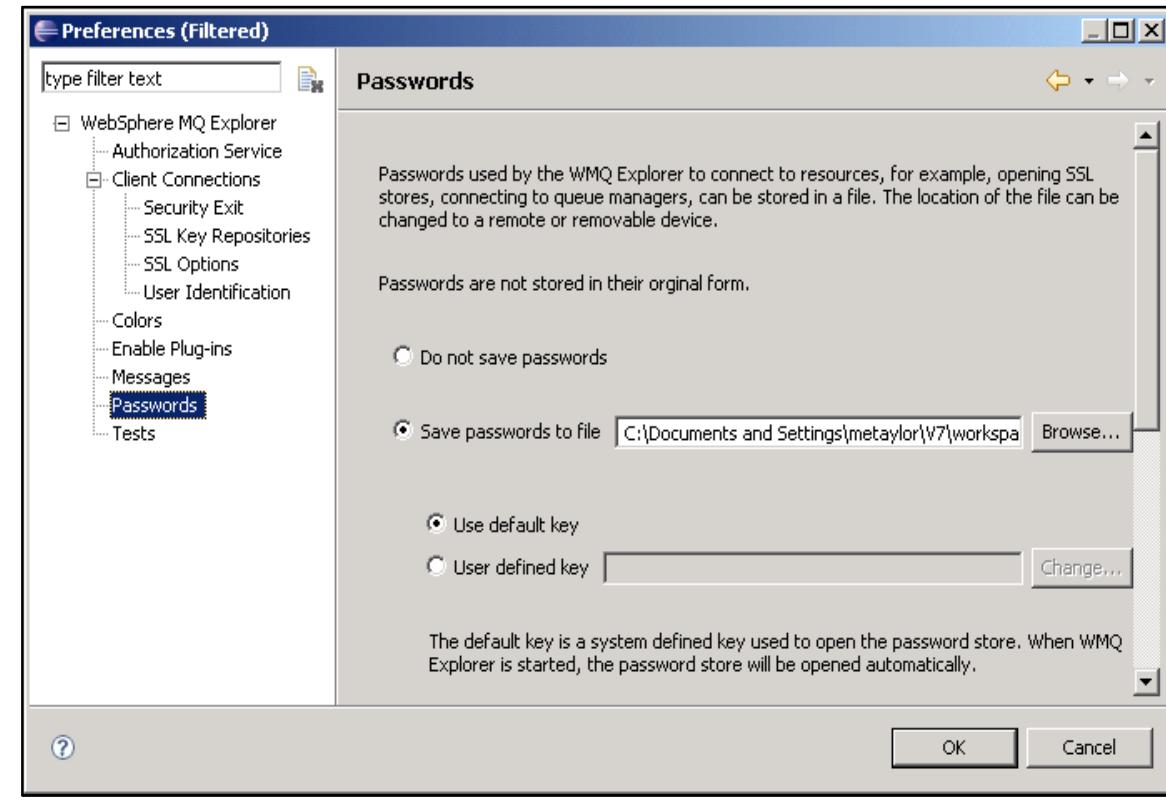
To modify the default security preferences for client connections in MQ Explorer:

1. Click **Windows > Preferences** to open the **Preferences** dialog box.
2. Expand **WebSphere MQ Explorer**.
3. **Expand Client Connections.** The default security settings are now accessible.

The figure shows the **User Identification** client connections preferences. In this option, the user ID and password are passed to the server and a server security exit can use it to establish the identity of the MQ client.

 WebSphere Education

## Storing IBM MQ Explorer passwords



© Copyright IBM Corporation 2014

Figure 10-34. Storing IBM MQ Explorer passwords

WM2091.0

### Notes:

Using MQ Explorer, you can store passwords to a file so that you do not have to enter them every time you want to connect to resources.

The password file can be stored locally, to a remote device, or to a removable device.

To open the **Passwords** preference window:

1. Click **Windows > Preferences**.
2. Expand **WebSphere MQ Explorer**.
3. Select **Passwords** to show the Passwords view.

## Unit summary

Having completed this unit, you should be able to:

- Describe the role of the object authority manager (OAM) to provide security to MQ resources
- Protect MQ resources by using the OAM
- Use some of the OAM control commands
- Describe the Secure Sockets Layer (SSL) support provided in MQ
- Implement basic channel authentication

© Copyright IBM Corporation 2014

Figure 10-35. Unit summary

WM2091.0

### Notes:



## Checkpoint questions

1. True or false: You must either restart the queue manager or use the **REFRESH SECURITY** command to refresh the cached view of the configuration for connection authentication.
2. For an application to open a queue by using an alternate user ID, the initial user ID requires:
  - a. OAM delegate authority
  - b. OAM alternate user authority
  - c. Password of alternate user ID
3. Using OAM, how would you protect local queue REBATE.IN on MY.QMGR for:
  - a. PUT access only by group ASSESSORS
  - b. GET by using the ID of PROCESSOR
  - c. GET browse only by group AUDITOR

© Copyright IBM Corporation 2014

Figure 10-36. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.

## Checkpoint answers

1. True.
2. b.
3. a. PUT access only by group ASSESSORS:  
`setmqaut -m MY.QMGR -t q -n REBATE.IN -g ASSESSORS +put`  
b. GET by using the ID of PROCESSOR  
`setmqaut -m MY.QMGR -t q -n REBATE.IN -p PROCESSOR +get`  
c. GET browse only by group AUDITOR:  
`setmqaut -m MY.QMGR -t q -n REBATE.IN -g AUDITOR +browse`

© Copyright IBM Corporation 2014

Figure 10-37. Checkpoint answers

WM2091.0

### Notes:

## Exercise 8



### Controlling access to IBM MQ

© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.O

Figure 10-38. Exercise 8

WM2091.0

### Notes:

In this exercise, you use the OAM utilities to set access control on a queue, and then use sample programs to see the effect of attempting to breach security.



## Exercise objectives

After completing this exercise, you should be able to:

- Use the `setmqaut` command to define access control on a queue
- Use the `dspmqaut` command to display the access control on a queue
- Use IBM MQ Explorer to manage authority records
- Test security by using IBM MQ sample programs

© Copyright IBM Corporation 2014

Figure 10-39. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for detailed instructions.



# Unit 11. IBM MQ clients

## What this unit is about

This unit describes the ways that IBM MQ clients can attach to an IBM MQ server. Different connection methods are explained.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the various ways of connecting a client to a queue manager
- Analyze client connection requirements to determine the best approach
- Describe the limitations of various client connection methods
- Propose methods to ensure security with client connections

## How you will check your progress

- Checkpoint questions
- Hands-on exercise

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Describe the various ways of connecting a client to a queue manager
- Analyze client connection requirements to determine the best approach
- Describe the limitations of various client connection methods
- Propose methods to ensure security with client connections

© Copyright IBM Corporation 2014

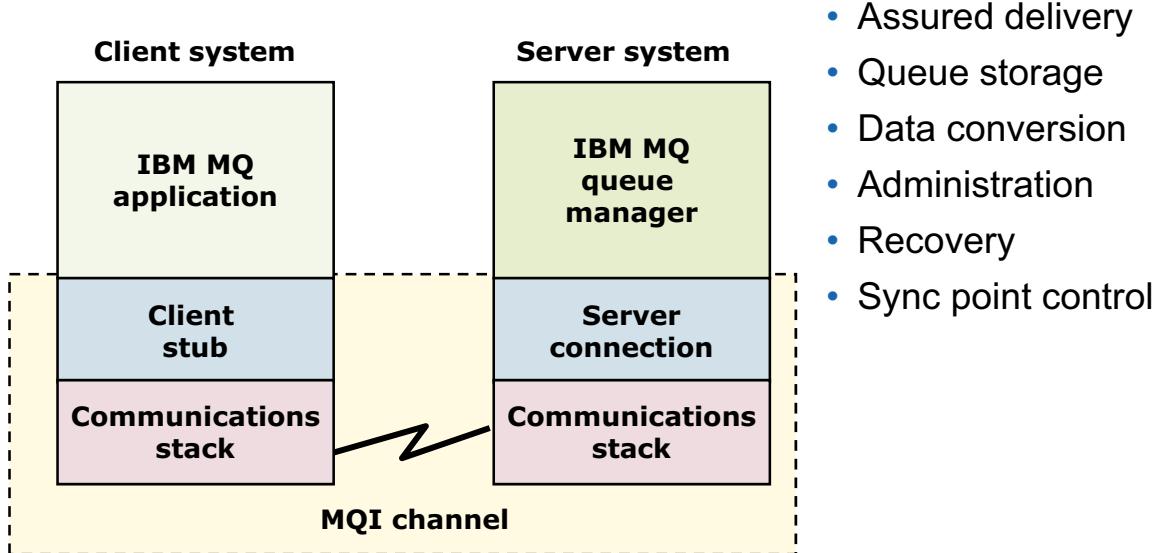
---

Figure 11-1. Unit objectives

WM2091.0

### Notes:

## IBM MQ client



- Assured delivery
- Queue storage
- Data conversion
- Administration
- Recovery
- Sync point control

© Copyright IBM Corporation 2014

Figure 11-2. IBM MQ client

WM2091.0

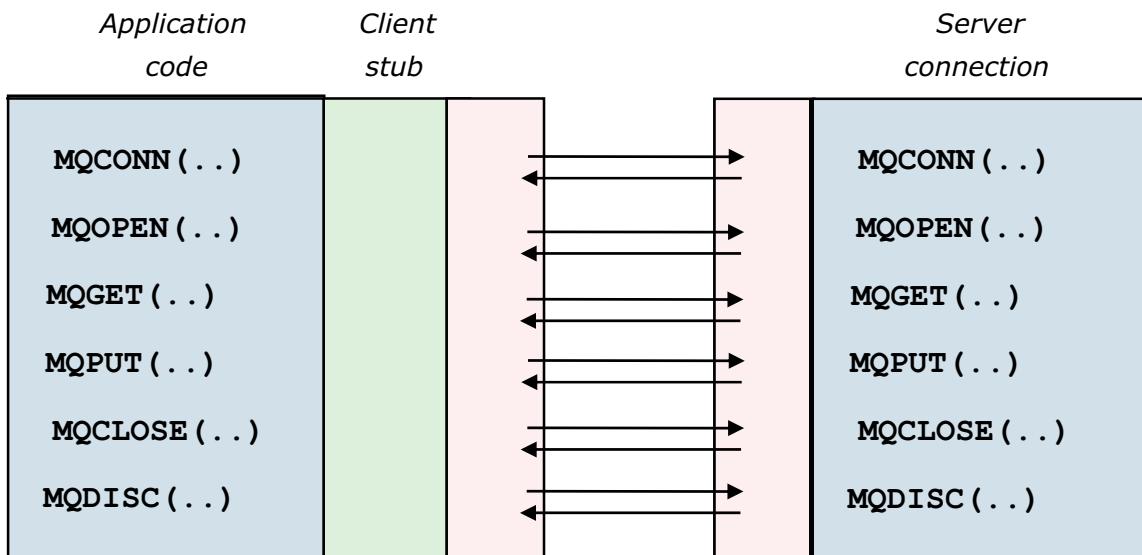
### Notes:

With an MQ client, an application that is running on the same system as the client can connect to a queue manager that is running on another system and issue MQI calls to that queue manager. This type of application is referred to as an *MQ client application* and the queue manager is referred to as the *MQ server queue manager*.

An MQ client can be installed on a system on which no queue manager runs.

An MQ client application and a server queue manager communicate with each other by using an MQI channel.

## IBM MQ clients explained



© Copyright IBM Corporation 2014

Figure 11-3. IBM MQ clients explained

WM2091.0

### Notes:

An MQI channel starts when the client application issues an MQCONN or MQCONNX call to connect to the server queue manager. It ends when the client application issues an MQDISC call to disconnect from the server queue manager.

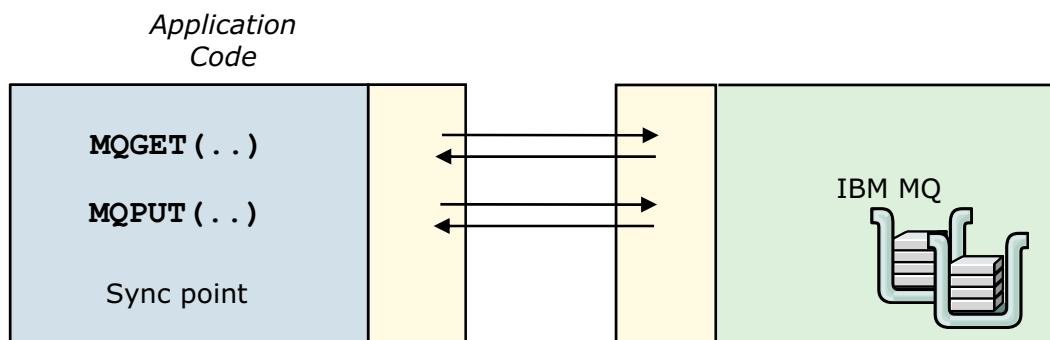
The input parameters of an MQI call flow in one direction on an MQI channel and the output parameters flow in the opposite direction.

A client application can be connected to more than one queue manager server simultaneously. Each MQCONN call to a different queue manager returns a different connection handle. This behavior does not apply if the application is not running as an MQ client.

The MQI stub that is linked with an application when running as a client is different from the one that is used when the application is not running as a client. An application receives the reason code MQRC\_Q\_MGR\_NOT\_AVAILABLE on an MQCONN call if it is linked with the wrong MQI stub.

The full range of MQI calls and options is available to an MQ client application.

## Sync point control on a base client



- IBM MQ client application:
  - Can participate in a local unit of work that uses IBM MQ resources
  - Cannot participate in a global unit of work that uses IBM MQ resources

© Copyright IBM Corporation 2014

Figure 11-4. Sync point control on a base client

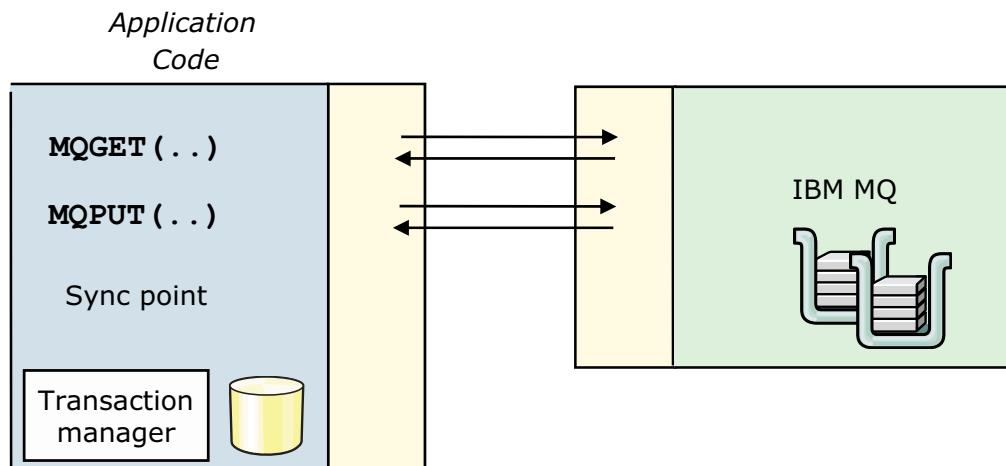
WM2091.0

### Notes:

An MQ client application can participate in a local unit of work that involves the resources of the queue manager it is connected to. It uses the MQCMIT and MQBACK calls in the normal way.

An MQ client application cannot participate in a global unit of work. An application can send MQI calls to another resource manager or sync point coordinator, on the same system as the application or on another system. It can participate in a unit of work that is associated with that resource manager or sync point coordinator. At the same time, it can also be participating in a local unit of work that requires the resources of the queue manager it is connected to. In which case, the two units of work are independent of each other.

## Extended transactional client



- Can update resources that are managed by another resource manager, under the control of an external transaction manager
  - XA transactional client is supplied with IBM MQ
  - Transaction manager runs on client system
  - Transaction manager provides sync point processing

© Copyright IBM Corporation 2014

Figure 11-5. Extended transactional client

WM2091.0

### Notes:

An *MQ extended transactional client* is an MQ MQI client with some additional features.

An MQ extended transactional client application, within the same unit of work, can:

- Put messages to, and get messages from, server queue manager queues
- Update the resources of a resource manager other than an MQ queue manager

An external transaction manager that is running on the same system as the client application must manage this unit of work. The queue manager to which the client application is connected cannot manage the unit of work; the queue manager can act only as a resource manager, not as a transaction manager. The client application can commit or back out the unit of work by using only the API provided by the external transaction manager. The client application cannot, therefore, use the MQI calls, MQBEGIN, MQCMIT, and MQBACK.



## Client configuration methods

- Method 1: MQSERVER environment variable on client
- Method 2: Client configuration file on the client
- Method 3: Client channel definition table
- Method 4: Encapsulated connection object in Java (application code)
- Method 5: External JNDI lookup for JMS (application code)

© Copyright IBM Corporation 2014

Figure 11-6. Client configuration methods

WM2091.0

### Notes:

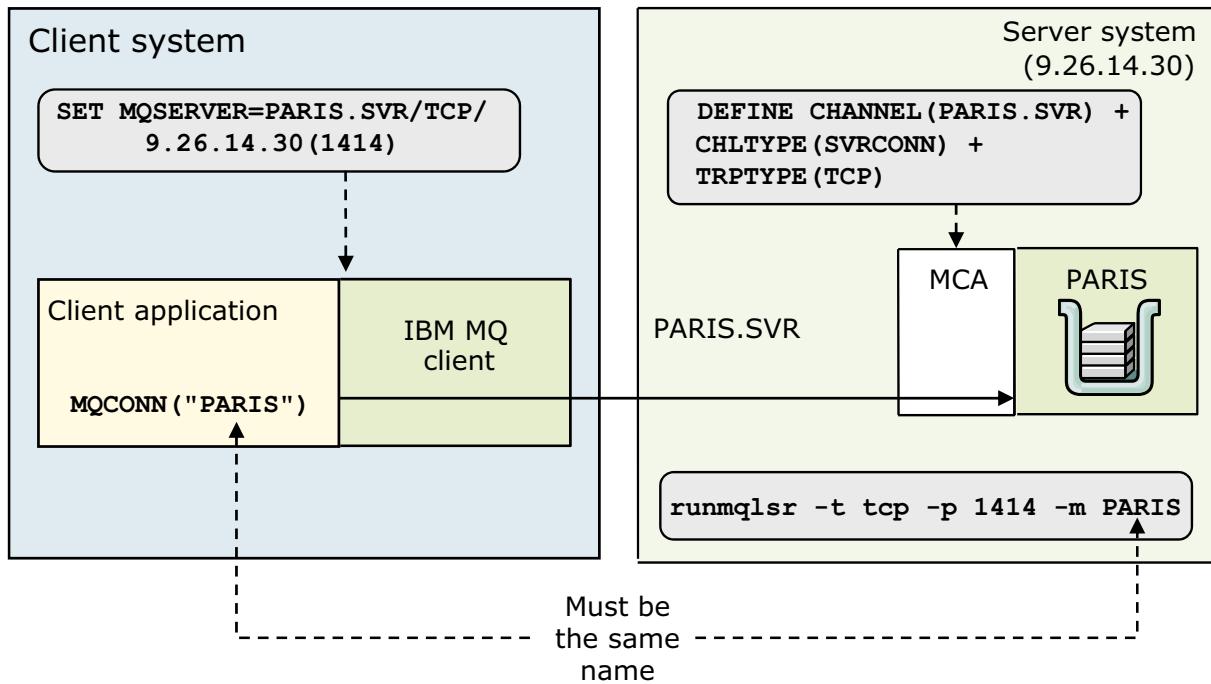
The figure lists the ways to configure clients connections:

1. Set the MQSERVER environment variable on the client.
2. Modify the client configuration file (`mqclient.ini`)
3. Create a client channel definition table.
4. Use an encapsulated connection object in Java
5. Create an external JNDI lookup for JMS.

The first two methods are relevant from an administration point of view. These methods are examined in more detail on the next pages.

The last two methods require user-defined application code.

## Method 1: Using MQSERVER



© Copyright IBM Corporation 2014

Figure 11-7. Method 1: Using MQSERVER

WM2091.0

### Notes:

Method 1 is the easier of the two methods for setting up a client. On the server, you use the MQSC `DEFINE CHANNEL` command to define a server connection (SVRCONN) and the `runmqlsr` command to start the listener.

On the client system, you define a simple client connection by setting the environment variable `MQSERVER` to the following value:

`ChannelName/TransportType/ConnectionName`

For example, on Windows, type: `SET MQSERVER=VENUS.SVR/TCP/9.20.4.56`

For example, on a UNIX system, type: `export MQSERVER=VENUS.SVR/TCP/9.20.4.56`

As shown in the figure, the connection name in the `MQCONN` call must be the same as the queue manager name in the `runmqlsr` command.

## Defining an MQI channel for a client connection

- Use **DEFINE CHANNEL** command with parameters:

<b>CHLTYPE:</b>	CLNTCONN or SVRCONN
<b>TRPTYPE:</b>	LU62, NETBIOS, SPX, or TCP
<b>CONNNAME (<i>string</i>)</b>	For a client connection only
<b>QMNAME (<i>string</i>)</b>	For a client connection only

- No operational involvement on an MQI channel
  - MQI channel starts when a client application issues **MQCONN** (or **MQCONNX**)
  - MQI channel stops when a client application issues **MQDISC**

© Copyright IBM Corporation 2014

Figure 11-8. Defining an MQI channel for a client connection

WM2091.0

### Notes:

As with a message channel, an MQI channel requires a definition at both ends of the channel and each end of an MQI channel has a type. The two permissible types are:

- **CLNTCONN:** Client connection, for the end of the MQI channel on the client system.
- **SVRCONN:** Server connection, for the end of the MQI channel on the server system.

MQI channels do not have to be started. Run the client application that calls an **MQCONN** or **MQCONNX**.

An MQI channel can be used to connect a client to a single queue manager, or to a queue manager that is part of a queue-sharing group.

The MQ listener must be running on the server system.

## Method 2: Client configuration file

- Configure clients by using attributes in the `mqclient.ini` text file
  - Default location on UNIX and Linux is `/var/mqm`
  - Default location on Windows is `C:\ProgramData\IBM\MQ`
  - Location can be specified by **MQCLNTCF** environment variable
- Configuration applies to all connections a client application makes to any queue managers
- Attributes can be overridden by environment variables

Example:

```
/* Module Name: mqclient.ini
/* Type : IBM MQ MQI client configuration file
# Function : Define the configuration of a client
#####
ClientExitPath:
    ExitsDefaultPath=/var/mqm/exits
    ExitsDefaultPath64=/var/mqm/exits64
CHANNELS:
    DefRecon=YES
    ServerConnectionParms=SALES.SVRCONN/TCP/hostname.x.com(1414)
```

© Copyright IBM Corporation 2014

Figure 11-9. Method 2: Client configuration file

WM2091.0

### Notes:

You configure your MQI clients by using a text file, similar to the queue manager configuration file. The file contains stanzas, each of which contains lines of the format **attribute-name=value**

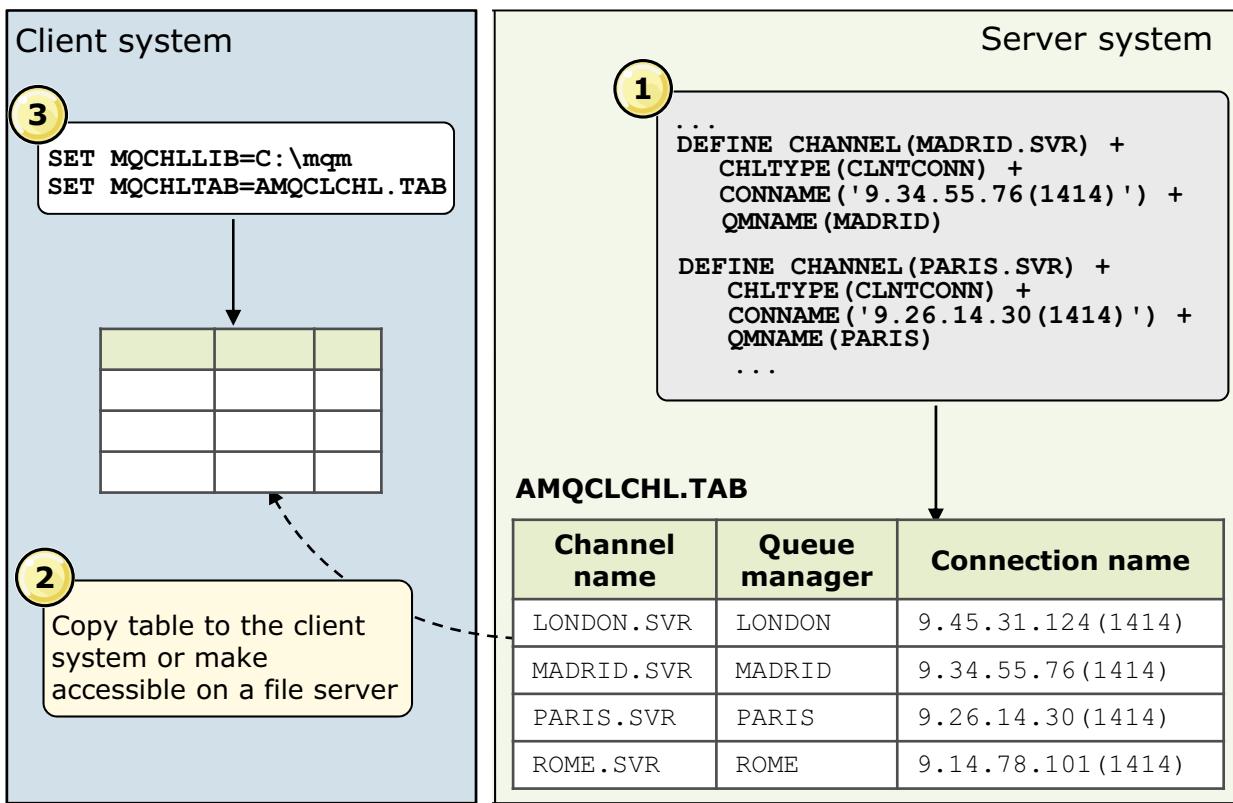
The configuration features apply to all connections a client application makes to any queue managers, rather than being specific to an individual connection to a queue manager.

The CHANNELS stanza in the client configuration file specifies information about client channels.

The **DefRecon** attribute provides an administrative option to enable client programs to automatically reconnect, or to disable the automatic reconnection of a client program that is written to reconnect automatically.

The **ServerConnectionParms** attribute is equivalent to the **MQSERVER** environment parameter and specifies the location of the MQ server and the communication method. The **ServerConnectionParms** attribute defines only a simple channel; you cannot use it to define an SSL channel or a channel with channel exits. It is a string of the format **ChannelName/TransportType/ConnectionName**. The **ConnectionName** must be a fully qualified network name.

## Method 3: Using a client channel definition table



© Copyright IBM Corporation 2014

Figure 11-10. Method 3: Using a client channel definition table

WM2091.0

### Notes:

Method 3 is a more complex method and uses the definition of a client connection and a server connection on the server system.

As shown in the figure, the client connection definition is stored in the *client channel definition table*, file name **AMQCLCHL.TAB**. This table must be accessible from the client system by existing on a file server, or it must be copied to the client system.

On the client system, the environment variables **MQCHLLIB** and **MQCHLTAB** specify the location and the name of the client channel definition table.

On Windows, the syntax is:

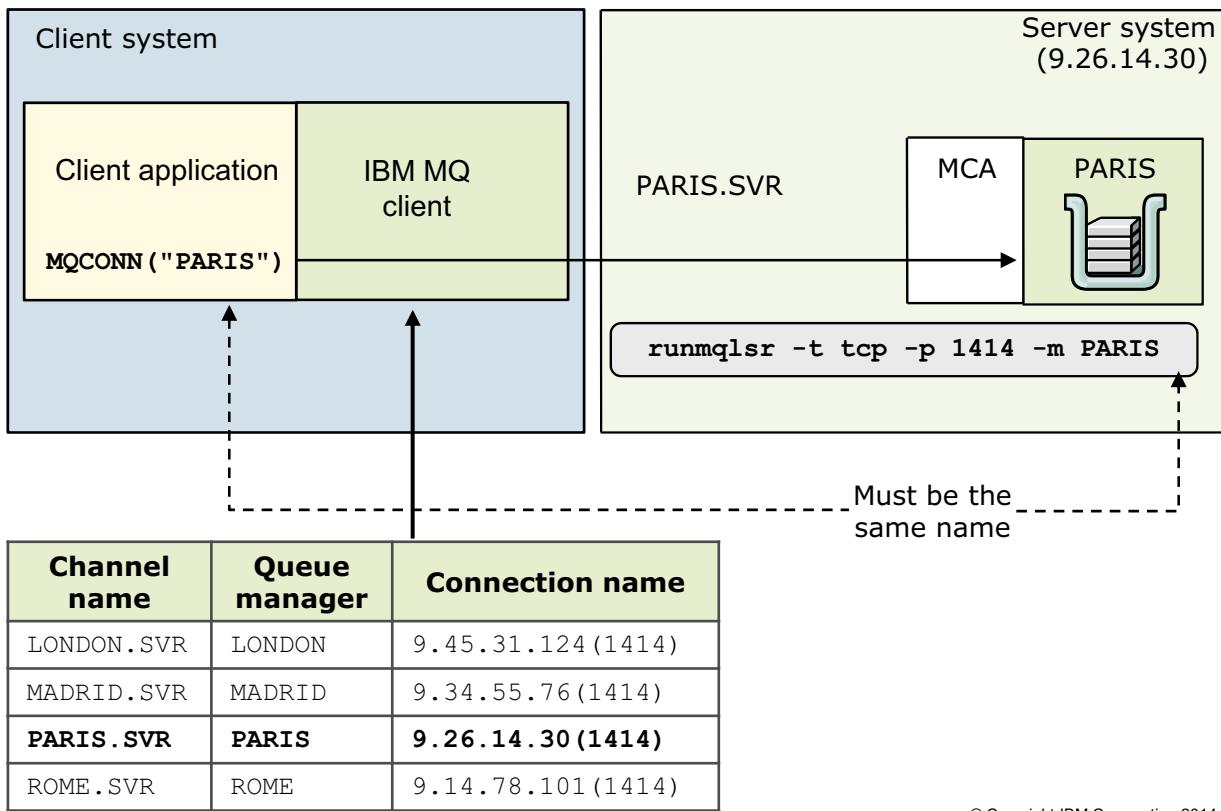
```
SET MQCHLLIB=C:\Program Files\IBM\WebSphere MQ
SET MQCHLTAB=AMQCLCHL.TAB
```

On a UNIX system, the syntax is:

```
export MQCHLLIB=/var/mqm/
export MQCHLTAB=AMQCLCHL.TAB
```

Although method 1 is the simpler approach for configuring a client, method 2 provides more flexibility. For example, with method 1, only one MQI channel can be made available to a client application because the environment variable MQSERVER can be set to one value only. With method 3, the client channel definition table can contain multiple client connection definitions and so a client application can have access to multiple MQI channels.

## Accessing a client channel definition table



© Copyright IBM Corporation 2014

Figure 11-11. Accessing a client channel definition table

WM2091.0

### Notes:

As shown in the figure, the connection name in the `MQCONN` call from the client must match the queue manager name in the `runmqqlsr` command on the server. The client uses the client channel definition table to get the connection information for the server.

## Weighted selection on CLNTCONN channels

### Client channel definition table

Group name	Channel name	Queue manager	Connection name	Client weight	Affinity
GRP1	LONDON.SVR	LONDON	9.45.31.124 (1414)	20	NONE
GRP1	MADRID.SVR	MADRID	9.34.55.76 (1414)	50	PREFERRED
GRP1	PARIS.SVR	PARIS	9.26.14.30 (1414)	30	NONE
GRP1	ROME.SVR	ROME	9.14.78.101 (1414)	0	PREFERRED

Application 1:  
MQCONN ("\*GRP1")

First attempt: ROME.SVR  
Subsequent attempts:  
ROME.SVR

Application 2:  
MQCONN ("\*GRP1")

First attempt: ROME  
Next attempt: 50%  
chance MADRID  
then 100% MADRID

Application 3:  
MQCONN ("\*GRP1")

First attempt: ROME  
Next attempt: 30%  
chance PARIS then 100%  
ROME (if up) or MADRID,  
PARIS, or LONDON

\* Distribution is not guaranteed

© Copyright IBM Corporation 2014

Figure 11-12. Weighted selection on CLNTCONN channels

WM2091.0

### Notes:

MQ has two attributes on the client connection channel definition for use in client channel definition tables: client weighting (CLNTWGHT) and affinity (AFFINITY). These two attributes allow load sharing and selection of a channel that is based on the weights that are assigned to the channel at definition time. The special channel weighting of 0 means to always use this channel definition. However, if this connection is not available, then further matching channels are tried. If there are multiple matching channels with a weighting of 0, then these channels are tried first, in alphabetical order.

For nonzero client weightings, weights are summed; then a random number between 1 and that sum is generated. The sum is used to select the matching channel. In the example in the figure, a weight of 1-20 causes LONDON to be selected, 21-70 selects MADRID, and 71-99 selects PARIS.

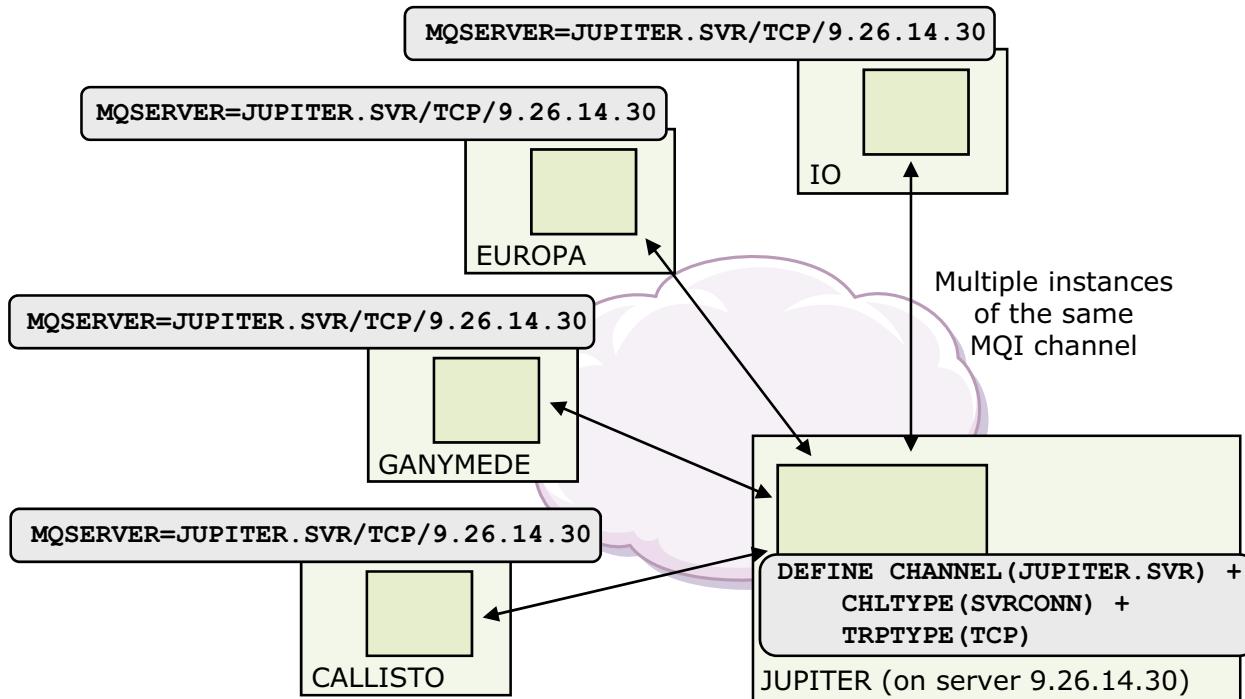
The channel affinity attribute is used so client applications that connect multiple times with the same queue manager name can choose whether to use the same client channel definition for each connection.

In the example, Application 1 chooses the connection to and ROME because the client weighting is zero. The Affinity value is set to PREFERRED, so subsequent connections come back to this connection if it is available; otherwise the next matching connection is tried.

Application 2, also tried to connect to ROME, but it is not responding for some reason. In the absence of any other weight 0 channels, the channel selection algorithm chooses MADRID (50% of the time) which also has an affinity of PREFERRED. Subsequent connections from this client return to MADRID while it is available.

Application 3 also fails to connect to ROME and connects to PARIS. On PARIS, AFFINITY is set to NONE so subsequent connections from this client start a new selection process. If ROME becomes available, that connection is re-established; otherwise it is distributed among the remaining three channels that are based on their weights.

## Channel instances



© Copyright IBM Corporation 2014

Figure 11-13. Channel instances

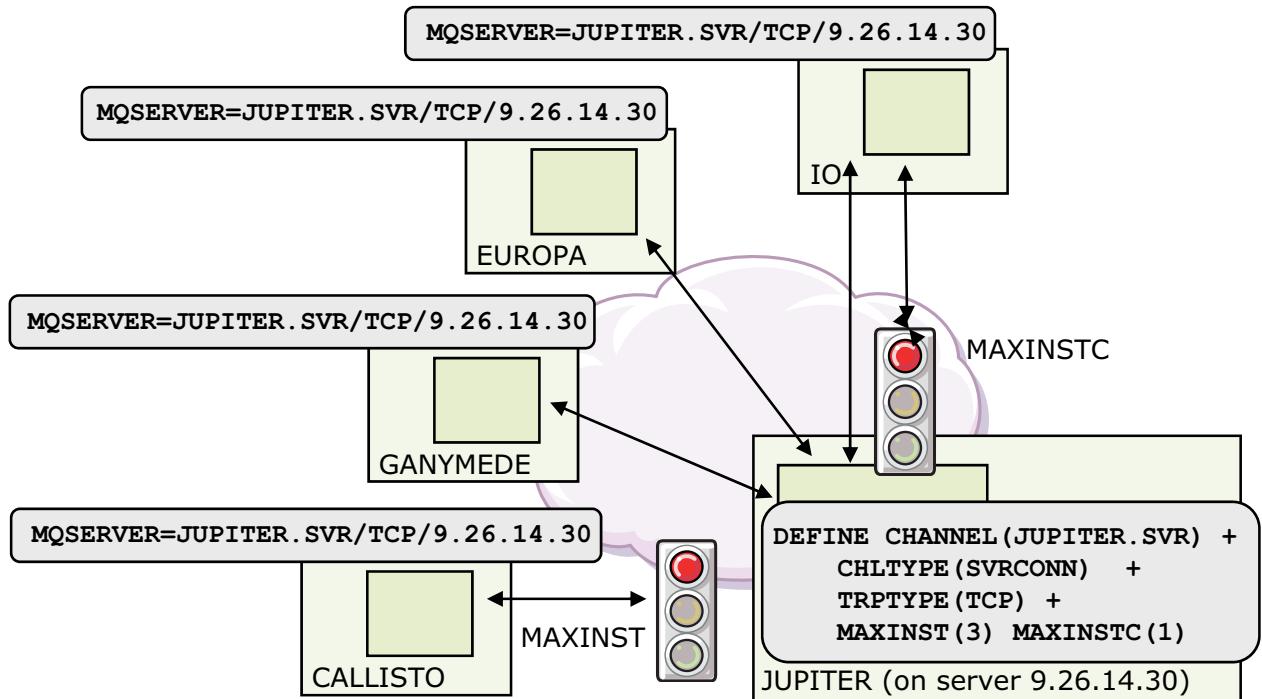
WM2091.0

### Notes:

Each of the four MQI channels that are shown in the figure is an *instance* of the same MQI channel. The main points to note are:

- The environment variable MQSERVER has an identical value on each of the client systems with host names IO, EUROPA, GANYMEDE, and CALLISTO.
- On the server system, JUPITER, the queue manager requires just one server connection definition for the MQI channel JUPITER.SVR.
- Each client application is using a different instance of the MQI channel JUPITER.SVR.

## Instance limits on SVRCONN channels



© Copyright IBM Corporation 2014

Figure 11-14. Instance limits on SVRCONN channels

WM2091.0

### Notes:

The MAXINST channel attribute specifies the maximum number of simultaneous instances of a server-connection channel that can be started. A value of zero prevents all client access on this channel.

The MAXINSTC attribute specifies the maximum number of simultaneous instances of a server-connection channel that can be started from a single client. This attribute can be set from zero through 999 999 999. A value of zero indicates that no client connections are allowed on this channel.

If the MAXINST or MAXINSTC value is dynamically reduced, no new channel instances are started from this client and running channel instances remain running.

The example shows four separate clients, all attempting to connect over the SVRCONN channel that is called JUPITER.SVR. Three of the clients connect with one instance each: IO, EUROPA, and GANYMEDE. The fourth client CALLISTO is refused connection because of the MAXINST value of 3 on the SVRCONN definition. IO tries to start a second client connection over the same JUPITER.SVR channel. The connection is refused also because it exceeds the MAXINSTC value of JUPITER.SVR.

## Auto-definition of channels

- Applies only to the end of a channel with type:
  - Receiver
  - Server connection
- Function that is started when an incoming request is received to start a channel but there is no channel definition
- Channel definition is created automatically by using the model:
  - SYSTEM.AUTO.RECEIVER
  - SYSTEM.AUTO.SVRCONN
- Connection partner values are used for:
  - Channel name
  - Sequence number wrap value

© Copyright IBM Corporation 2014

Figure 11-15. Auto-definition of channels

WM2091.0

### Notes:

In MQ, you can enable the automatic definition of a channel if there is no appropriate channel definition for a receiver or server-connection channel. The function is started if there is an incoming request to start a message channel or MQI channel but no channel definition exists for the specific channel.

To enable the automatic definition of channels, the attribute **ChannelAutoDef** of the queue manager object must be set to MQCHAD\_ENABLED. The corresponding parameter on the **ALTER QMGR** command is **CHAD(ENABLED)**

The definition is created by using the appropriate model channel definition (SYSTEM.AUTO.RECEIVER or SYSTEM.AUTO.SVRCONN) and the channel name and sequence wrap values from the client.

After a channel definition is created automatically, the definition can be used as though it always existed.

Channel auto-definition events can be enabled by setting CHADEV(ENABLED) with the **ALTER QMGR** command.



## IBM MQ client limitations

- Remote to IBM MQ server relies on network connectivity
- Client-based trigger monitors
- Distribution of IBM MQ maintenance
- Control of IBM MQ applications

© Copyright IBM Corporation 2014

Figure 11-16. IBM MQ client limitations

WM2091.0

### Notes:

There are some limitations when you use MQ clients:

- Clients use network connections. This limitation introduces a dependency on network availability. Network availability also applies when the MQ client and server are on the same server.
- MQ client trigger monitors are available. Trigger monitors work just like server-based trigger monitors but are on the client side.
- Distribution of MQ client maintenance can be a problem, particularly if control of the client computer is with another organization. As with any distributed component, keeping the MQ client software up to date becomes a challenge, particularly if control of the client computer is the responsibility of another organization.
- The MQ administrator might not have access to control the client application.



## IBM MQ client security

- MCAUSER
  - Default setting is wide open, especially for client attachment
  - You must restrict who can access your queue manager
- Logged-in user name is automatically used but not authenticated at the server
- Configuration support in MQ Explorer
  - Setting the default security exit for all client connections
  - Enabling the default SSL options for all client connections
  - Enabling the default SSL stores
  - Specifying the user ID and password server security exit users use to establish the identity of the IBM MQ client

© Copyright IBM Corporation 2014

Figure 11-17. IBM MQ client security

WM2091.0

### Notes:

You must consider MQ client security so that the client applications do not have unrestricted access to resources on the server.

There are two aspects to security between a client application and its queue manager server: authentication and access control. Client access control is based on user IDs.

Use the client security configuration support in MQ Explorer to set default security options on all new client connections. These default values can be overridden when a new queue manager is added.

## Access control for an IBM MQ client

- Access control that is based on user ID used by server connection process (**MCAUserIdentifier** in MQCD)
- Security exits at both ends of the MQI channel
  - Client security exit can flow a user ID and password
  - Server security exit can authenticate the user ID and set **MCAUserIdentifier**
- No security exit at the client end of the MQI channel
  - Value of logged in user ID flows to the server system
  - Server security exit can authenticate the user ID and set **MCAUserIdentifier**
- No security exit at either end of the MQI channel
  - **MCAUserIdentifier** has the value of **MCAUSER** if it is nonblank
  - **MCAUserIdentifier** has the value of flowed user ID otherwise

© Copyright IBM Corporation 2014

Figure 11-18. Access control for an IBM MQ client

WM2091.0

### Notes:

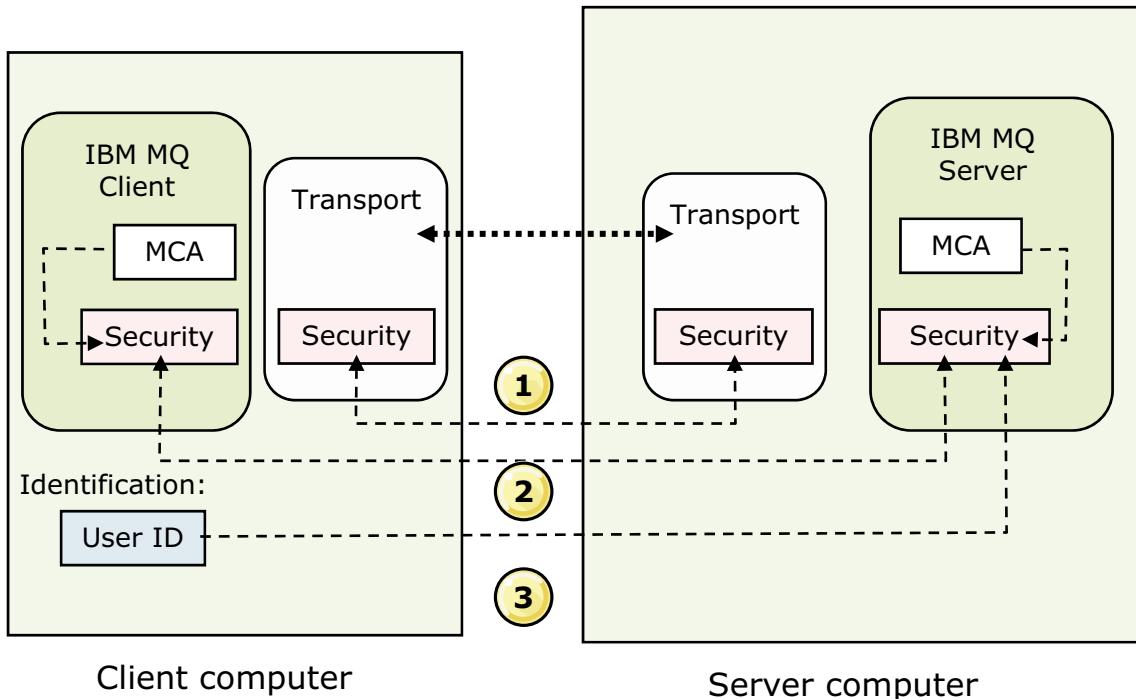
An MQ client application might want to connect to the queue manager, or open a queue, for example. When an MQ client application wants to access an MQ object on the server queue manager, access control is based on a user ID of the server connection process. The reason for this is that it is the server connection that sends the MQI calls.

The value of the **MCAUserIdentifier** field in the active channel definition at the server end of the MQI channel instance determines the user ID.

Depending on the combination of settings, the user-identifier fields are set to appropriate values. If a server-connection security exit is provided, the exit can set the user-identifier fields. Otherwise, they are determined as follows:

- If the server-connection channel MCAUSER attribute is nonblank, this value is used.
- If the server-connection channel MCAUSER attribute is blank, the user ID received from the client is used.

## IBM MQ client security checks



© Copyright IBM Corporation 2014

Figure 11-19. IBM MQ client security checks

WM2091.0

### Notes:

There are three levels of security to consider for MQ clients:

1. Transport level checks, such as IP filtering.
2. Custom channel exits, such as to check security to custom specification
3. Default user ID propagation such as, MQ passes the currently logged-on user ID that a server-side security exit can use to authenticate.
- 4.

## Unit summary

Having completed this unit, you should be able to:

- Describe the various ways of connecting a client to a queue manager
- Analyze client connection requirements to determine the best approach
- Describe the limitations of various client connection methods
- Propose methods to ensure security with client connections

© Copyright IBM Corporation 2014

Figure 11-20. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. Which best describes an IBM MQ client?
  - a. A channel to an IBM MQ server
  - b. An API for IBM MQ
  - c. A customer who purchases IBM MQ
  - d. Libraries and communications that allows an application to use a remote IBM MQ server
2. True or false: An IBM MQ client can participate in global (XA) units of work.
3. Is it true that a client channel with weighting = 0 is always selected?
4. Which of the following statements are true?
  - a. MAXINST limits the total number of client connections for a specific channel
  - b. MAXINSTC limits the total number of connections for a specific channel from an individual client
  - c. Affinity causes subsequent client connections from a specific client to return to the same channel.
  - d. All of the above.

© Copyright IBM Corporation 2014

Figure 11-21. Checkpoint questions

WM2091.0

### Notes:

Enter your answers here:

- 1.
- 2.
- 3.
- 4.



## Checkpoint answers

1. c and d.
2. **False.** The IBM MQ extended transactional client is required to participate in global transactions.
3. **Generally true.** A client channel with weighting = 0 is always selected, but if the target queue manager is not available, then it is skipped.
4. d.

© Copyright IBM Corporation 2014

Figure 11-22. Checkpoint answers

WM2091.0

### Notes:

## Exercise 9

Connecting an IBM MQ client



© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.O

Figure 11-23. Exercise 9

WM2091.0

### Notes:

In this exercise, you configure a local client that is connected to an IBM MQ server system. You use two different methods (MQSERVER environment variable and CLNTCONN channel definition) to gain experience with each method. You encounter security issues and use the MCAUSER attribute to solve this problem.



## Exercise objectives

After completing this exercise, you should be able to:

- Create a server connection (SVRCONN) channel to support client connections
- Use the MQSERVER environment variable to set up a client connection
- Use a client configuration file to specify information about client channels

© Copyright IBM Corporation 2014

Figure 11-24. Exercise objectives

WM2091.0

### Notes:



# Unit 12. Backing up and restoring IBM MQ object definitions

## What this unit is about

This unit describes the ways in which a queue manager can be backed up and restored on the file system where it is installed.

## What you should be able to do

After completing this unit, you should be able to:

- Develop a method for backing up the IBM MQ environment
- Save the queue manager object definitions

## How you will check your progress

- Checkpoint questions
- Hands-on exercises

## References

IBM MQ product documentation

## Unit objectives

After completing this unit, you should be able to:

- Develop a method for backing up the IBM MQ environment
- Save the queue manager object definitions

© Copyright IBM Corporation 2014

Figure 12-1. Unit objectives

WM2091.0

### Notes:



## Overview of the IBM MQ file system structure

- Windows
  - Programs: C:\Program Files\IBM\WebSphere MQ\
  - Data: C:\ProgramData\IBM\MQ
- UNIX and Linux
  - Programs: /usr/mqm
  - Data: /var/mqm

© Copyright IBM Corporation 2014

Figure 12-2. Overview of the IBM MQ file system structure

WM2091.0

### Notes:

Before you back up, you must know the locations of the MQ components on the file system. There are some variations in the file system layout for MQ depending on the operating system.

The figure lists the default installation location for MQ software components and data. The paths might be different for your installation.

## The need to back up

- Allow recovery of queue managers against possible corruption or loss of data that hardware failures cause
- IBM MQ file system backup can be done as part of a full system backup



© Copyright IBM Corporation 2014

Figure 12-3. The need to back up

WM2091.0

### Notes:

Periodically backing up MQ and its objects allows the recovery of queue managers against possible corruption that hardware failures cause.

If the hardware fails, a queue manager is forced to stop. If any queue manager log data is lost because of the hardware failure, the queue manager might be unable to restart. Through backing up queue manager data, you might be able to recover some, or all, of the lost queue manager data.

## When to back up

- File system backups can be done as part of the regular full system backup
- Backups can be run at any time interval
  - Nightly
  - Weekly
- Run backups regularly
- Backups must be conducted when IBM MQ is not running

© Copyright IBM Corporation 2014

Figure 12-4. When to back up

WM2091.0

### Notes:

In general, the more frequently you back up queue manager data, the less data you lose if the hardware fails and results in the loss of integrity in the recovery log.

Shut down MQ when you back up to ensure that the backup data is consistent when the files were copied. Shutting down MQ means that it is unavailable, which might cause problems if it is required for high availability.



#### Important

MQ must be stopped when backing up the file system. If MQ is running, updates that are in-progress are not backed up.

## How to back up

- Ensure that queue manager is not running
- Locate directories where the queue manager places its data and log files
- Take copies of all the data from the queue manager and log file directories, including subdirectories
- Restart the queue manager

© Copyright IBM Corporation 2014

Figure 12-5. How to back up

WM2091.0

### Notes:

As previously mentioned, The queue manager must be stopped when backing up. If you try to back up a running queue manager, the backup might not be consistent because of updates in progress when the files were copied. If possible, stop your queue manager in an orderly way.

Use the information in the configuration files to find the directories under which the queue manager places its data and its log files.

Some of the directories might be empty, but you need them all to restore from the backup, so save all directories and subdirectories.

Preserve the ownerships of the files. For MQ on UNIX, you can preserve ownership by using the UNIX `tar` command.

Restart the queue manager after the backup is complete.

## Overview of IBM MQ objects

- IBM MQ object types include:
  - Queue managers
  - Queues
  - Channels
  - Process definitions
  - Listeners
  - Namelists
  - Client connection channels
  - Services
  - Authentication information objects
- Anything that you define to IBM MQ by using MQSC

© Copyright IBM Corporation 2014

Figure 12-6. Overview of IBM MQ objects

WM2091.0

### Notes:

This figure is a review of the various MQ object types that are defined to MQ. Each object has an object definition that can be backed up.

## Why back up IBM MQ object definitions

- To recreate objects that were deleted or missing
- To simplify administration tasks in IBM MQ including:
  - Migration
  - Cloning systems
  - Automation of application migrations
  - Auditing

© Copyright IBM Corporation 2014

Figure 12-7. Why back up IBM MQ object definitions

WM2091.0

### Notes:

When updating applications, you might want to keep the same object definitions from the queue managers of source environment. By backing up the MQ object definitions you can restore them on the new queue manager quickly and easily. Otherwise, a manual process would be required to restore the queue managers.

Similarly, when cloning systems (typically for clustering) or for automation of application migrations, a backup provides a quick and simple way to copy the object definitions from one queue manager to another.

If you are auditing your MQ system, backing up the object definition provides a view of all the definitions in one place.

Fundamentally, backing up object definitions is about saving time and effort by using the backup file to re-create the object definition on a new queue manager.

## Overview of facilities available

- Save IBM MQ configuration command: `dmpmqcfg`
  - Use IBM MQ programmable command format (PCF) commands to save queue manager object definitions
  - Can connect to local queue managers or use client connections to remote systems
- Output from MQSC `DISPLAY` command

© Copyright IBM Corporation 2014

Figure 12-8. Overview of facilities available

WM2091.0

### Notes:

There are two options available for backing up object definitions:

- Save the output from the MQSC display queue manager, queue, and channel commands in an external file.
- Save the MQ configuration by using the integrated `dmpmqcfg` command (the most comprehensive option).

## Backing up resource definitions with runmqsc

```
C:\> runmqsc QmgrName
DISPLAY QL(*) ALL
DISPLAY QR(*) ALL
DISPLAY CHL(*) CHLTYPE(*) ALL
```

- Simple and quick method to see all the object definitions
- **runmqsc** command has an SQL-like syntax so that queries can be easily refined
- Output is formatted in single or multiple column format
  - Needs reformatting to be used for restore purposes

© Copyright IBM Corporation 2014

Figure 12-9. Backing up resource definitions with runmqsc

WM2091.0

### Notes:

To use MQSC commands to see all the object definitions, enter **DISPLAY** and the **ALL** attribute. This command reports the resource definitions for the specified queue manager.

The output is useful for a quick overview of the object definitions; however, it is formatted in single or multi-columns, making it difficult to be used for restore purposes. Reformatting is required to use the output for restoration purposes.

This option does not provide a complete backup of all objects.

## Save IBM MQ configuration: `dmpmqcfg`

- Back up object definitions and authorities, restore with MQSC
- Rebuild queue manager on new version of IBM MQ
- Generate reports on objects and their access control
- Must have appropriate authority to each object that is inquired
- Variety of output formats supported
  - “Classic” MQSC
  - One line MQSC
  - `setmqaut`
  - `grtmqaut` (Linux only)
- Supported on distributed platforms only but can connect to a remote z/OS queue manager

© Copyright IBM Corporation 2014

Figure 12-10. Save IBM MQ configuration: `dmpmqcfg`

WM2091.0

### Notes:

The `dmpmqcfg` command extracts a queue manager's configuration and displays it in MQSC syntax.

The `dmpmqcfg` command can connect to local queue managers or use client connections to remote systems.

You can specify the syntax of the backup report in the `dmpmqcfg` command. The syntax options are:

- Multi-line MQSC that can be used as direct input on a `rungqsc` control command.
- MQSC with all attributes on a single line for line comparison.
- MQ set authority access (`setmqaut`) statements for UNIX and Windows queue managers.
- On Linux, grant MQ authority access (`grtmqaut`) for granting access to the objects on iSeries.

## Backing up queue manager configuration

To take a backup copy of a queue manager's configuration:

1. Ensure that the queue manager is running.
2. Run `dmpmqcfg` command with:
  - Formatting option of (`-o mqsc`) to create multi-line MQSC that can be used as direct input to MQSC
  - All attributes (`-a`)
  - Standard output redirection to store the definitions into a file

Example:

```
dmpmqcfg -o mqsc -m MYQMGR -a > /mq/backups/MYQMGR.mqsc
```

© Copyright IBM Corporation 2014

Figure 12-11. Backing up queue manager configuration

WM2091.0

### Notes:

This figure lists the steps for backing up a queue manager configuration by using the `dmpmqcfg` control command.



#### Important

When backing up the MQ file system, the queue manager must be stopped.

When backing up an MQ configuration by using the `dmpmqcfg` command, the queue manager must be running.

The example command in the figure creates a backup of all attributes (`-a`) of the queue manager that named MYQMGR (`-m MYQMGR`) in a multi-line MQSC format (`-o mqsc`). The results of the command `dmpmqcfg` are redirected to the `MYQMGR.mqsc` in the `/mq/backups/` directory.

## Using `dmpmqcfg`: Syntax examples

```
dmpmqcfg [-m QMgrName] [-n ObjName] [-c Connection]
[-t ObjType] [-x ExportType] [-o Format] [-a] [-z]
```

- Example 1: Local queue manager, all objects, and authorities:

```
dmpmqcfg -m MYQMGR
```

- Example 2: Local queue manager, all object definitions of SYSTEM queues showing all attributes

```
dmpmqcfg -m MYQMGR -n SYSTEM.* -t queue -x object -a
```

- Example 3: Local queue manager, all channel authentication records, silence any warnings

```
dmpmqcfg -m MYQMGR -x chlauth -z
```

- Example 4: Local queue manager, all authorities in `setmqaut` format

```
dmpmqcfg -m MYQMGR -o setmqaut
```

- Example 5: Dynamic client connection to remote queue manager

```
dmpmqcfg -m RMTQMGR -c "DEFINE CHANNEL(SYSTEM.DEF.SVRCONN)
CHLTYPE(CLNTCONN) CONNAME('idev01.hursley.ibm.com(1414)')"
```

© Copyright IBM Corporation 2014

Figure 12-12. Using `dmpmqcfg`: Syntax examples

WM2091.0

### Notes:

The figure shows examples of the `dmpmqcfg` control command.

## Using dmpmqcfg: Output MQSC

```
*****
* Script generated on 2013-10-31 at 15.44.12
* Script generated by user 'JSMITH' on host 'dev.IBM.COM'
* Queue manager name: MYQMGR
* Queue manager platform: IBM i
* Queue manager command level: (710/710)
* Command issued: QMQM/DMPMQCFG -m MYQMGR
*****
ALTER QMGR +
* ALTDATE(2013-10-31) +
* ALTTIME(14.32.42) +
CCSID(37) +
CLWLUSEQ(LOCAL) +
* COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) +
* CRDATE(2013-10-31) +
* CRTIME(14.32.42) +
* PLATFROM(AS400) +
* QMID(MYQMGR_2011-10-31_14.32.42) +
SSLKEYR('/QIBM/UserData/ICSS/Cert/Server/Default') +
* VERSION(08000000) +
FORCE
SET AUTHREC +
PROFILE('self') +
PRINCIPAL('QMQM') +
OBJTYPE(QMGR) +
. . .
```

© Copyright IBM Corporation 2014

Figure 12-13. Using dmpmqcfg: Output MQSC

WM2091.0

### Notes:

The figure shows an example of a multi-line MQSC report that the `dmpmqcfg` command creates when the `-o mqsc` option is specified.

## Using dmpmqcfg: Output setmqaut

```
#####
# Script generated on 2013-10-31      at 15.59.09
# Script generated by user 'JSMITH' on host 'dev.IBM.COM'
# Queue manager name: JMSMITH
# Queue manager platform: IBM i
# Queue manager command level: (710/710)
# Command issued: QMQM/DMPMQCFG -m MYQMGR -o setmqaut
#####
setmqaut -m MYQMGR -t qmgr -p QMQM +altusr +chg +connect +crt +dlt +dsp
+inq +set +setall +setid +ctrl +system
setmqaut -m MYQMGR -t qmgr -g QMQMADM +altusr +chg +connect +dlt +dsp
+inq +set +setall +setid +ctrl +system
setmqaut -m MYQMGR -n SYSTEM.ADMIN.ACCTING.QUEUE -t queue -p QMQM
+browse +chg +clr +dlt +dsp +get +inq +put +passall +passid +set
+setall +setid
setmqaut -m MYQMGR -n SYSTEM.ADMIN.ACCTING.QUEUE -t queue -g QMQMADM
+browse +chg +clr +dlt +dsp +get +inq +put +passall +passid +set
+setall +setid
setmqaut -m MYQMGR -n SYSTEM.ADMIN.ACTIVITY.QUEUE -t queue -p QMQM
+browse +chg +clr +dlt +dsp +get +inq +put +passall +passid +set
+setall +setid
setmqaut -m MYQMGR -n SYSTEM.ADMIN.ACTIVITY.QUEUE -t queue -g QMQMADM
+browse +chg +clr +dlt +dsp +get +inq +put +passall +passid +set
+setall +setid
...
...
```

© Copyright IBM Corporation 2014

Figure 12-14. Using dmpmqcfg: Output setmqaut

WM2091.0

### Notes:

The figure shows an example of a `dmpmqcfg` report that is created in the `setmqaut` format. You can run this report against a queue manager to restore MQ object authority.



## Overview of security definitions

- Security definitions in this context refer to IBM MQ OAM security definitions; not UNIX or Windows security profiles or SSL data
- OAM security definitions
  - Define the authorizations that are given to a specified user group
  - Use `setmqaut` to create

© Copyright IBM Corporation 2014

Figure 12-15. Overview of security definitions

WM2091.0

### Notes:

The `setmqaut` commands that the `dmpmqcfg -o setmqaut` command generates are MQ OAM commands. They do not include UNIX or Windows security profiles or SSL data.



## Need to back up security definitions

- Restoration of security definitions
- Migration between environments
- Audit purposes

© Copyright IBM Corporation 2014

Figure 12-16. Need to back up security definitions

WM2091.0

### Notes:

Similar to backing up object definitions, backing up security definitions allows users to save time and effort by using the backup file to re-create the security definitions rather than doing it manually.

With regards to audits, backing up security definitions provides a way to view the security definition information in one place and in a format that might be more meaningful to the viewer.

## Backing up security definitions

- Use the **dmpmqaut** command to create a report of the current authorizations

Sample output:

```
profile:      a.b.*  
object type: queue  
entity:       user1  
type:        principal  
authority:   get, browse, put, inq
```

- Use the **amqoamd -s** command to create a report the current authorizations in a **setmqaut** format

Sample output:

```
setmqaut -m AJGMQ1 -n @CLASS -t channel -p andrew@IBM-L3M1562 +crt  
setmqaut -m AJGMQ1 -n @CLASS -t channel -g mqm +crt  
setmqaut -m AJGMQ1 -n @CLASS -t authinfo -p andrew@IBM-L3M1562 +crt  
setmqaut -m AJGMQ1 -n @CLASS -t authinfo -g mqm +crt  
setmqaut -m AJGMQ1 -n SYSTEM.BROKER.ADMIN.STREAM -t queue -p  
    MUSR_MQADMIN@IBM-L3M1562 +browse +chg +clr +dlt +dsp +get +inq +put  
    +passall +passid +set +setall +setid
```

© Copyright IBM Corporation 2014

Figure 12-17. Backing up security definitions

WM2091.0

### Notes:

There are three ways to back up the OAM security definitions: **dmpmqcfg**, **dmpmqaut**, and **amqoamd -s**.

The output from the **dmpmqaut** command, is formatted in a report style. This output cannot be easily used for restoring security definitions unless it is reformatted.

The output from the **amqoamd -s** command, which is shown in the figure, is generated as a series of **setmqaut** commands. These commands can be used in a command window to re-create the OAM security definitions.



## dmpmqaut command

```
dmpmqaut -m QMgrName [-n Profile | -l | -a] -t ObjectType
-s ServiceComponent [-p PrincipalName | -g GroupName]
[-e | -x]
```

- n**      Profile The name of the profile for which to dump authorizations.
- l**      Dump only the profile name and type.
- a**      Generate set authority commands.
- t**      ObjectType The type of object for which to dump authorizations. Possible values are authinfo, channel, clntconn, listener, namelist, process, queue, qmgr, rqmname, service, or topic
- e**      Display all profiles that are used to calculate the cumulative authority that the entity has to the object specified in **-n Profile**
- x**      Display all profiles with the same name as specified in **-n Profile**

© Copyright IBM Corporation 2014

Figure 12-18. dmpmqaut command

WM2091.0

### Notes:

You can use the MQ **dmpmqaut** control command to create a report of the current authorizations that are associated with a specified profile.

This figure describes the syntax for the **dmpmqaut** command.

## Backing up security definitions in IBM MQ Explorer

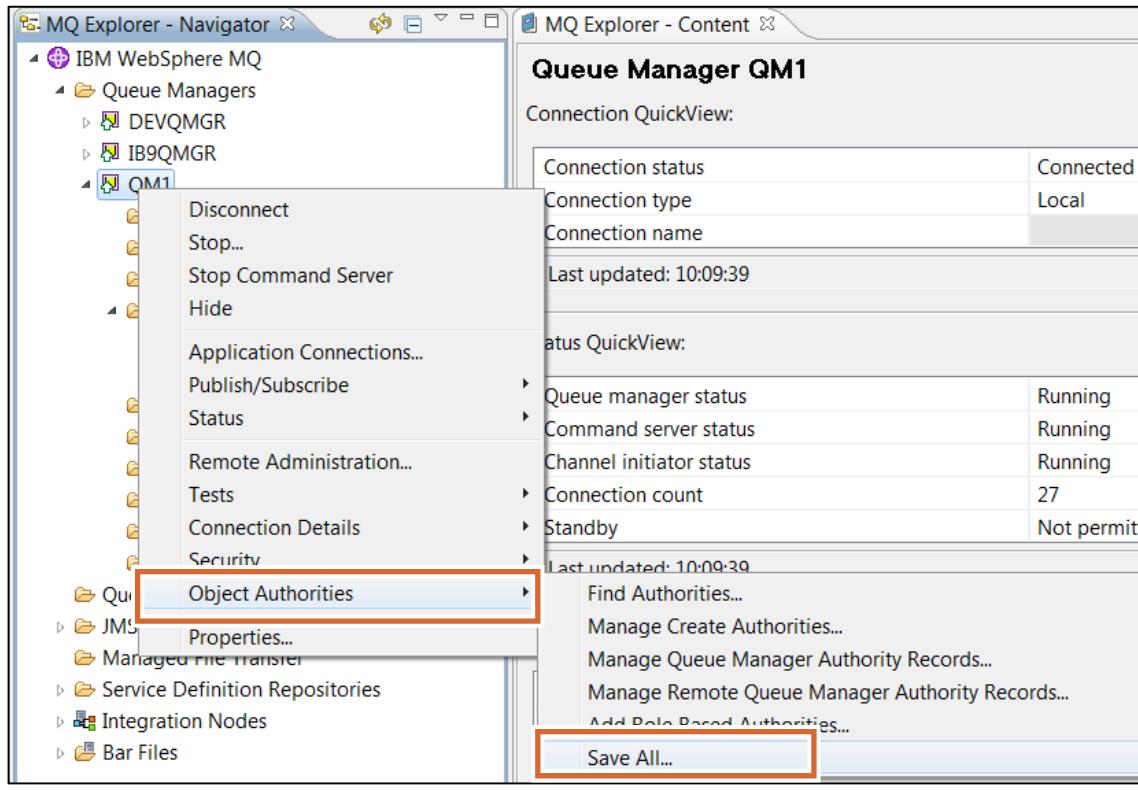


Figure 12-19. Backing up security definitions in IBM MQ Explorer

WM2091.0

### Notes:

As seen in the figure, the security definitions can be exported from the MQ Explorer.

The security definitions that are exported from MQ Explorer are in a format that is similar to the output from `amqoamd -s` command, which is a series of `setmqaut` commands.



## Unit summary

Having completed this unit, you should be able to:

- Develop a method for backing up the IBM MQ environment
- Save the queue manager object definitions

© Copyright IBM Corporation 2014

Figure 12-20. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. If you want to re-create your security definitions what would be the best method to back them up?
  - a. **dmpmqaut**
  - b. **amqoamd -s**
  - c. **bkupaut**
2. True or False: It is acceptable to back up IBM MQ files while the IBM MQ queue managers are running.

© Copyright IBM Corporation 2014

Figure 12-21. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

1.

2.



## Checkpoint answers

1. b. The output is given as a series of `setmqaut` commands, which can be used on the command interface to re-create the security definitions.
2. **False.** The backup is not consistent because of updates in progress when the files were copied.

© Copyright IBM Corporation 2014

Figure 12-22. Checkpoint answers

WM2091.0

### Notes:

## Exercise 10



### Backing up and restoring IBM MQ object definitions

© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.O

Figure 12-23. Exercise 10

WM2091.0

### Notes:

In this exercise, you use the `dmpmqcfg` command to unload a queue manager's object definitions. You then create a queue manager and load the same definitions, and use MQSC commands or MQ Explorer to show that the definitions are the same.



## Exercise objectives

After completing this exercise, you should be able to:

- Use IBM MQ commands to back up object definitions of a queue manager
- Use IBM MQ commands to upload object definitions to another queue manager

© Copyright IBM Corporation 2014

Figure 12-24. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for exercise instructions.



# Unit 13. Monitoring and configuring IBM MQ for performance

## What this unit is about

In this unit, you learn about the information that the accounting and statistics system management utilities provide for monitoring an IBM MQ network.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the statistics and accounting data that MQ provides
- View and generate accounting and statistical data
- Interpret statistics and accounting data to identify possible system performance benefits
- Configure and tune MQ for improved performance

## How you will check your progress

- Checkpoint questions
- Hands-on exercise

## References

IBM MQ product documentation

IBM WebSphere MQ SupportPac MS0P

## Unit objectives

After completing this unit, you should be able to:

- Describe the statistics and accounting data that MQ provides
- View and generate accounting and statistical data
- Interpret statistics and accounting data to identify possible system performance benefits
- Configure and tune MQ for improved performance

© Copyright IBM Corporation 2014

---

Figure 13-1. Unit objectives

WM2091.0

### Notes:

## Collecting statistics and accounting data

- IBM MQ collects sets of data to be written as messages to predefined queues
  - Data can be post-processed to give information about system activity
  - Data can be used for capacity planning, chargeback, and other information
- Statistical data collection is divided into three categories
  - MQI statistics
  - Queue statistics
  - Channel statistics
- Collection of data for each class can be selected independently
- Queue manager and queue or channel attributes control collection
- Accounting data collects information about MQI applications

© Copyright IBM Corporation 2014

Figure 13-2. Collecting statistics and accounting data

WM2091.0

### Notes:

A statistics message records the information about the activities that occur in an MQ system. A statistics message is a PCF message that contains PCF structures. They are delivered to the system queue (SYSTEM.ADMIN.STATISTICS.QUEUE) at configured intervals.

The information that is contained within statistics messages can be used for:

- Accounting for application resource use
- Recording application activity
- Planning for capacity
- Detecting problems in your queue manager network
- Helping to determine the causes of problems in your queue manager network
- Improving the efficiency of your queue manager network
- Familiarizing yourself with the running of your queue manager network
- Confirming that your queue manager network is running correctly

Statistical data is available for MQI, queues, and channels.

## MQI statistics

- Generated only for queues that are opened after statistics collection has been enabled
- Contain information relating to the number of MQI calls made during a configured interval
  - Success and failure counts for each MQI verb
  - MQI and byte counts for PUT and GET on queues for persistent and nonpersistent messages
- Written in the form of PCF records to SYSTEM.ADMIN.STATISTICS.QUEUE
- To enable, use the MQSC command **ALTER QMGR** with **STATMQI (ON)**

© Copyright IBM Corporation 2014

Figure 13-3. MQI statistics

WM2091.0

### Notes:

MQI statistics messages contain information that relates to the number of MQI requests processed during a configured interval. For example, the statistics message information can include the number of MQI commands that a queue manager processes.

MQI statistics are written to the system queue SYSTEM.ADMIN.STATISTICS.QUEUE.

## Queue statistics

- Can be configured at both system and queue level
- Minimum and maximum depth of queue
- Average time-on-queue for messages that are retrieved from the queue
- API and byte counts for GET, PUT, BROWSE (persistent and non-persistent)
- To enable, alter the queue definition to use the **STATQ (ON)** attribute

© Copyright IBM Corporation 2014

Figure 13-4. Queue statistics

WM2091.0

### Notes:

Queue statistics messages contain information about the activity of a queue during a configured interval. The information includes the number of messages that are put on and retrieved from the queue, and the total number of bytes that a queue processes.

Each queue statistics message can contain up to 100 records. Each record contains activity information for each queue for which statistics were collected.

## Channel statistics

- Can be configured at both system and channel level
  - Set **STATCHL** channel attribute to enable or disable channel statistic information collection  
**STATCHL (None | Off | Low | Medium | High)**
  - Set **STATACLS** queue manager attribute to control automatically defined cluster-sender channels
- Messages are written at end of the interval that contains up to 100 channel records
- Only channels “active” in the time interval have statistics that are recorded

© Copyright IBM Corporation 2014

Figure 13-5. Channel statistics

WM2091.0

### Notes:

Channel statistics messages contain information about the activity of a channel during a configured interval. For example, the message can include the number of messages that the channel transferred, or the number of bytes that the channel transferred.

Each channel statistics message contains up to 100 records. Each record contains the activity for each channel for which statistics were collected.

Channel statistics monitoring is set at either the channel level or at the queue manager level. The channel attribute **STATCHL** on the **ALTER CHL** command controls individual channels. The queue manager attribute **STATCHL** on the **ALTER QMGR** command controls many channels together. The choice of which level to use depends on your system.

Channel statistics information collection can be set to one of the three monitoring levels: low, medium, or high. Collecting statistics information data might require that the process runs some instructions that can affect performance. To reduce the impact of channel statistics collection, the medium and low monitoring options measure a sample of the data at regular intervals rather than collecting data all the time.

Automatically defined cluster-sender channels are not MQ objects, so they do not have attributes. To control automatically defined cluster-sender channels, use the queue manager attribute **STATACLs**. This attribute determines whether automatically defined cluster-sender channels within a queue manager are enabled or disabled for channel statistics information collection.

## Flush statistics

- Forces writing statistics immediately instead of waiting for an interval:  
**RESET QMGR TYPE(STATISTICS)**

OR  
by using  
WebSphere MQ  
SupportPac MSOP

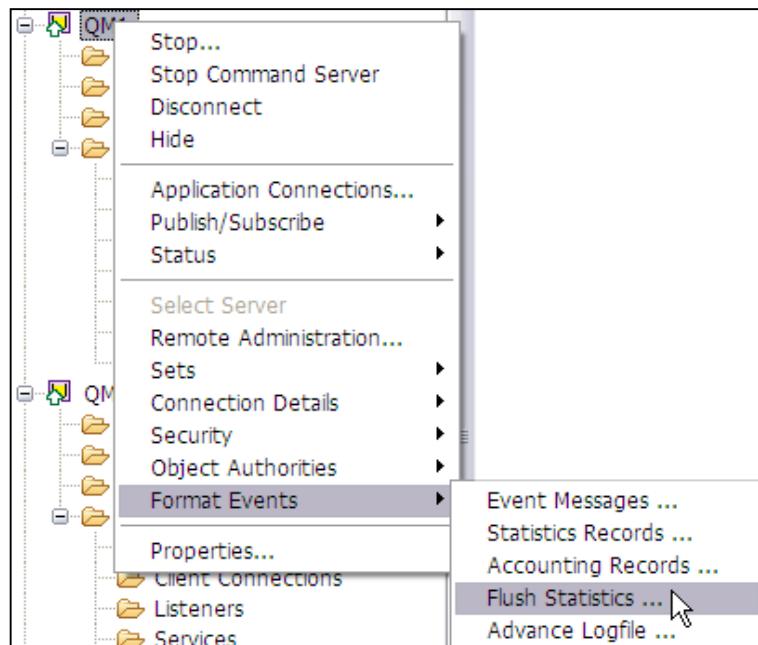


Figure 13-6. Flush statistics

WM2091.0

### Notes:

Statistics can be written before the end of the interval by entering a forced statistic write. To force writing statistics immediately, enter the MQSC command **REST QMGR TYPE(STATISTICS)**.

If you have the WebSphere MQ Explorer SupportPac MSOP installed, there is a menu option to flush (write) the statistics.

## IBM MQ accounting data collection

- Can collect connection-level and queue information for each connection
- Connection-level information includes:
  - Context details such as application name, process ID, connection type, and connect time
  - MQI counts
  - Message details (counts, bytes) for MQPUT, MQGET, MQGET (browse)
- Written in the form of a PCF monitoring message to SYSTEM.ADMIN.ACCTOUNTING.QUEUE

© Copyright IBM Corporation 2014

Figure 13-7. IBM MQ accounting data collection

WM2091.0

### Notes:

Accounting messages record information about the MQI operations that MQ applications complete. An accounting message is a PCF message that contains a number of PCF structures.

When an application disconnects from a queue manager, an accounting message is generated and delivered to the system accounting queue SYSTEM.ADMIN.ACCTOUNTING.QUEUE. For long-running MQ applications, intermediate accounting messages are generated as follows:

- When the time since the connection was established exceeds the configured interval
- When the time since the last intermediate accounting message exceeds the configured interval

The information that is contained within accounting messages can be used for the following purposes:

- Accounting for application resource use
- Recording application activity
- Detecting problems in your queue manager network
- Helping to determine the causes of problems in your queue manager network
- Improving the efficiency of your queue manager network
- Familiarizing yourself with the running of your queue manager network
- Confirming that your queue manager network is running correctly

MQI accounting messages contain information that is related to the number of MQI requests that were run by using a connection to a queue manager.

## IBM MQ queue accounting data

- Queue data might be up to 100 queue details per message
- Queue information includes:
  - Queue details: Name, type
  - Open details: First open time, last close time
  - MQPUT, MQGET, MQGET(browse) details

© Copyright IBM Corporation 2014

Figure 13-8. IBM MQ queue accounting data

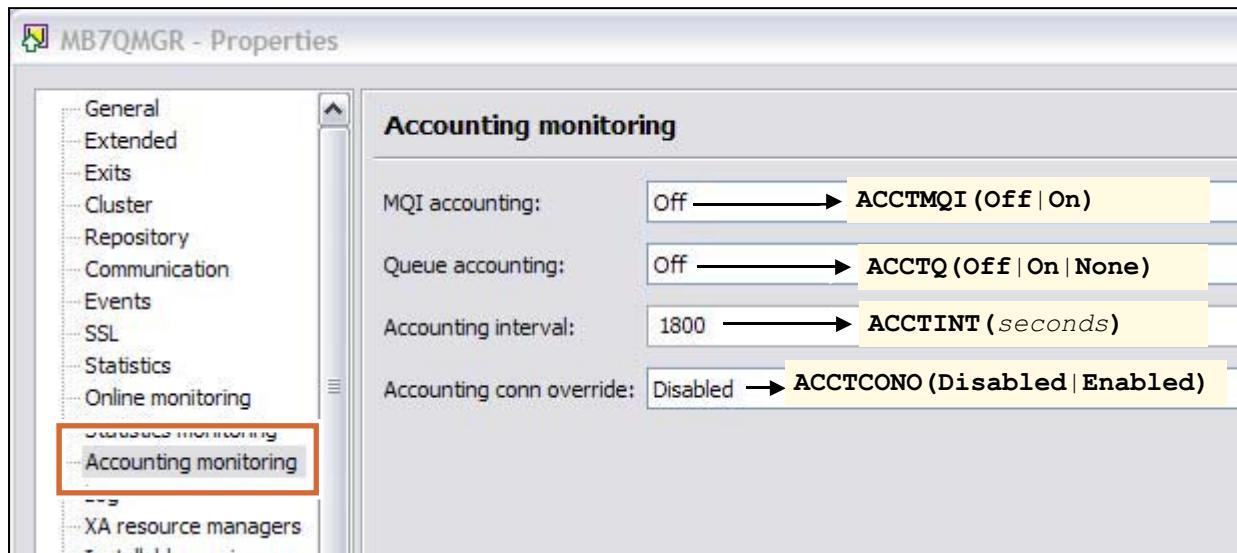
WM2091.0

### Notes:

Queue accounting messages contain information about the number of MQI requests that were processed by using connections to a queue manager, concerning specific queues.

Each queue accounting message can contain up to 100 records. Every record contains information about application activity for a specific queue.

## Queue manager accounting monitoring



© Copyright IBM Corporation 2014

Figure 13-9. Queue manager accounting monitoring

WM2091.0

### Notes:

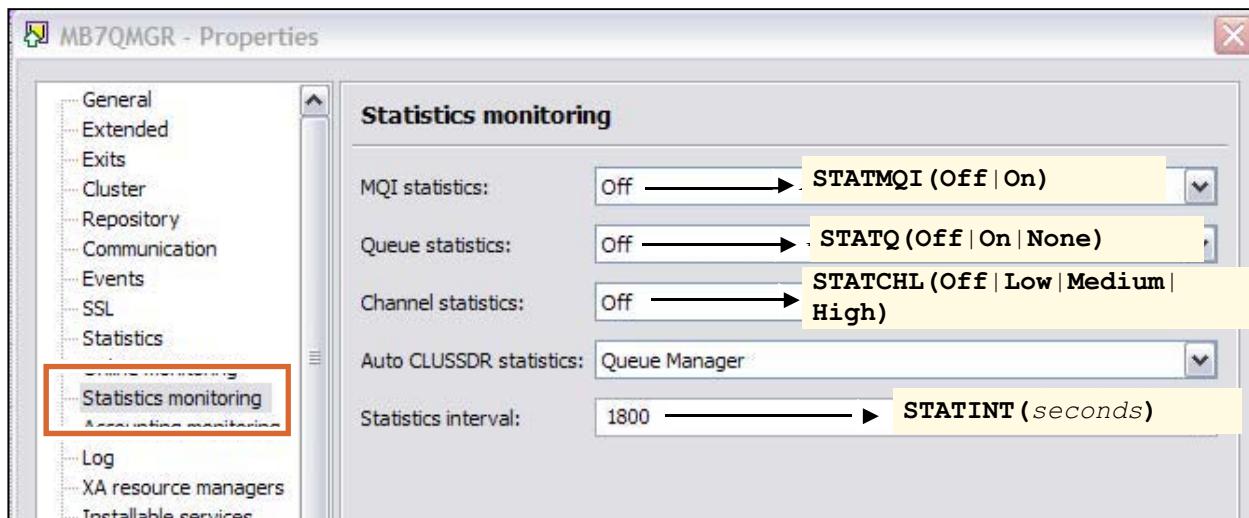
You can enable MQI and queue accounting on the queue manager by using MQSC commands or the queue manager **Accounting monitoring** properties in MQ Explorer. The figure shows the MQ Explorer accounting properties and the associated MQSC command.

The queue manager **MQI accounting** property (**ACCTMQI**) controls the collection of MQI accounting information. When **MQI accounting** is set to **On**, accounting information is collected for every connection to the queue manager.

The queue manager **Queue accounting** property (**ACCTQ**) controls the collection of queue accounting information for any queues with the **Queue accounting** property set to **Queue Manager**.



## Queue manager statistics monitoring



© Copyright IBM Corporation 2014

Figure 13-10. Queue manager statistics monitoring

WM2091.0

### Notes:

You can enable statistics collection for MQI, queues, and channels on the queue manager by using the MQSC commands or the queue manager **Statistics monitoring** properties in MQ Explorer. The figure shows the MQ Explorer statistics property and associated MQSC commands.

The queue manager **MQI statistics** property (**STATMQI**) controls the collection of MQI statistics. When the **MQI statistics** property is set to **On**, the MQI statistics information is collected for every connection to the queue manager.

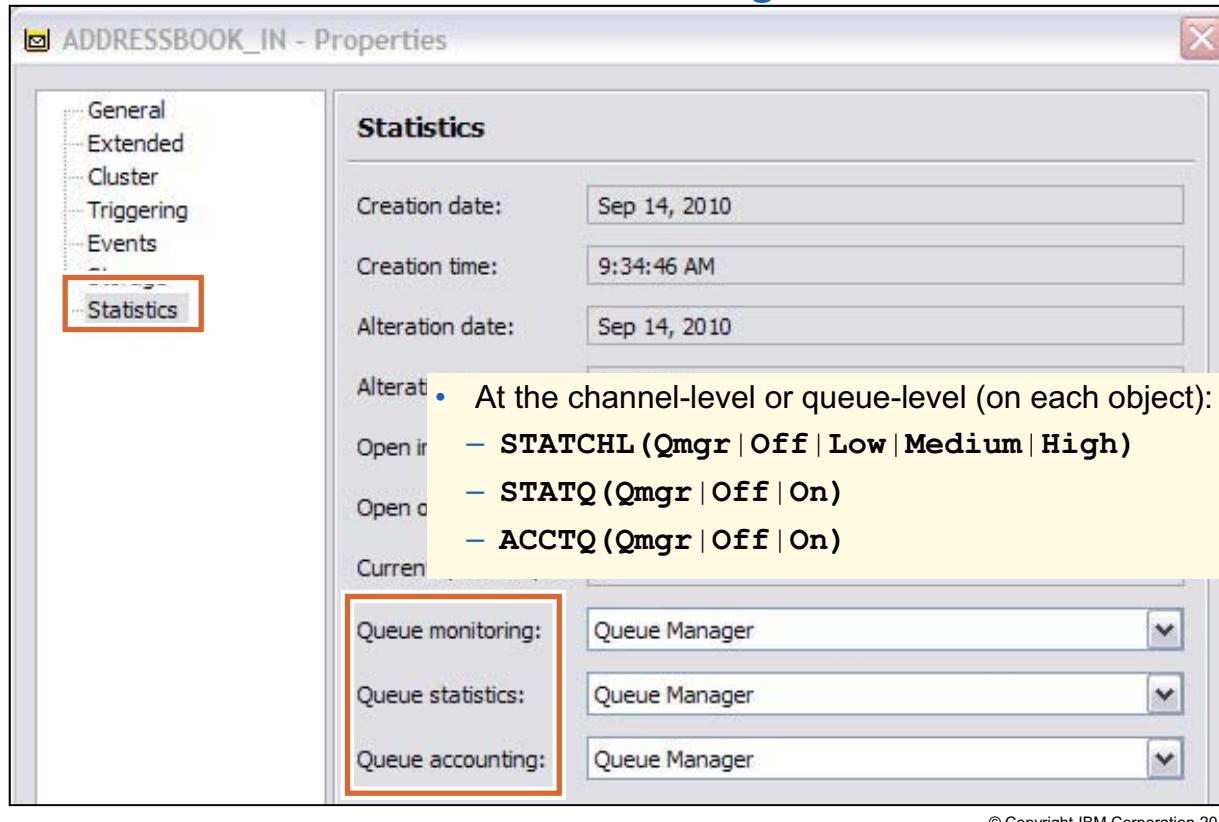
Queue statistics can be enabled for individual queues, or for multiple queues at the queue manager level by using the queue manager **Queue statistics** property (**STATQ**). When **Queue statistics** is set to **On**, queue statistics are collected for queues that have the **Queue statistics** property set to **Queue Manager**.

The queue manager **Channel statistics** property (**STATCHL**) controls the collection of channel statistics for all channels, or individual channels.

The Statistics interval property (**STATINT**) specifies the interval, in seconds, between the generation of statistics messages. The default statistics interval is 1800 seconds (30 minutes).



## Queue and channel statistics configuration



© Copyright IBM Corporation 2014

Figure 13-11. Queue and channel statistics configuration

WM2091.0

### Notes:

By using MQ Explorer or MQSC commands you can enable queue and channel statistics at the queue manager level or at the individual queue or channel level.

If the channel or queue **Statistics** properties are set to **Queue Manager**, the queue manager controls statistics collection. If the channel or queue **Statistics** properties are set to **On**, statistics information is collected for every connection to the queue manager that opens the queue or channel.

## Displaying statistics and accounting data

- Use **amqsmmon** sample program to display the information that is contained within accounting and statistics messages in a formatted form
  - Accounting messages are read from the accounting queue, SYSTEM.ADMIN.ACOUNTING.QUEUE
  - Statistics messages are read from the statistics queue, SYSTEM.ADMIN.STATISTICS.QUEUE
- Source code is provided
- Examples:

```
amqsmmon -m QM1 -t statistics
amqsmmon -m QMGR1 -t accounting
```

© Copyright IBM Corporation 2014

Figure 13-12. Displaying statistics and accounting data

WM2091.0

### Notes:

Accounting and statistics messages are written to the system accounting and statistics queues. To use the information that is recorded in these messages, you must use an application to transform the recorded information into a format that is suitable for use.

The MQ sample program, **amqsmmon**, processes messages from the accounting and statistics queues and shows the information in a readable format.

The supplied source code for the **amqsmmon** program can be used as a template for writing your own application to process accounting or statistics messages. You can also modify the **amqsmmon** source code to meet your own particular requirements.

The required parameter is the type (-t) of messages to process:

- If **-t** is set to **accounting**, accounting record messages are processed from the system queue SYSTEM.ADMIN.ACOUNTING.QUEUE
- If **-t** is set to **statistics**, statistics records are processed. Messages are read from the system queue SYSTEM.ADMIN.STATISTICS.QUEUE

For a complete description the **amqsmmon** parameters, see the IBM MQ product documentation.

## Examples of accounting output

```
QueueManager: QMGR1
IntervalEndDate: 2013-08-10
IntervalEndTime: 14:39:50
CommandLevel: 600
SeqNumber: 0
ApplName: amqspput.exe
ApplicationPid: 9408
ApplicationTid: 1
UserId: 'admin01'
ObjectCount: 1
```

```
OBJECTS:
QueueName: 'APP.QUEUE.X'
QueueType: Predefined
QueueDefType: Local
OpenCount: 1
OpenDate: 2013-08-10
OpenTime: 14:39:49
CloseCount: 1
CloseDate: 2013-08-10
CloseTime: 14:39:50
```

```
PutCount: [0, 1]*
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [0, 4]
PutMinBytes: [0, 4]
PutMaxBytes: [0, 4]
*array shows nonpersistent,
persistent
```

© Copyright IBM Corporation 2014

Figure 13-13. Examples of accounting output

WM2091.0

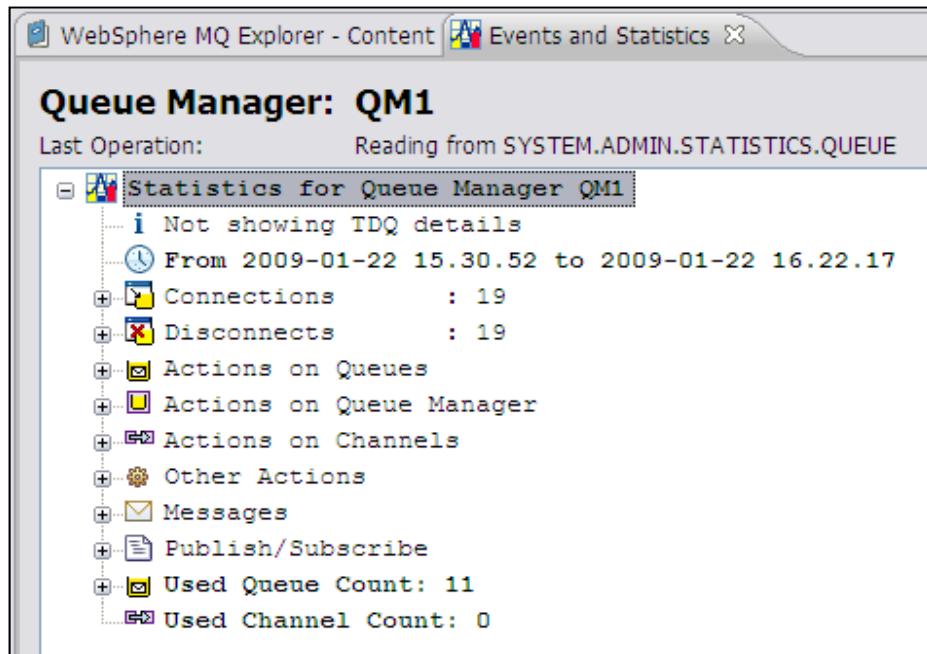
### Notes:

The figure includes three examples of accounting information that the **amqsmon** sample program generates.

The first example, show information about the queue manager. The second example, shows information about the objects that the queue manager controls, such as a queue. The third example shows more information about the queue.

## Display accounting and statistics

- MSOP SupportPac: Configuration and Display Extension plug-ins for WebSphere MQ



© Copyright IBM Corporation 2014

Figure 13-14. Display accounting and statistics

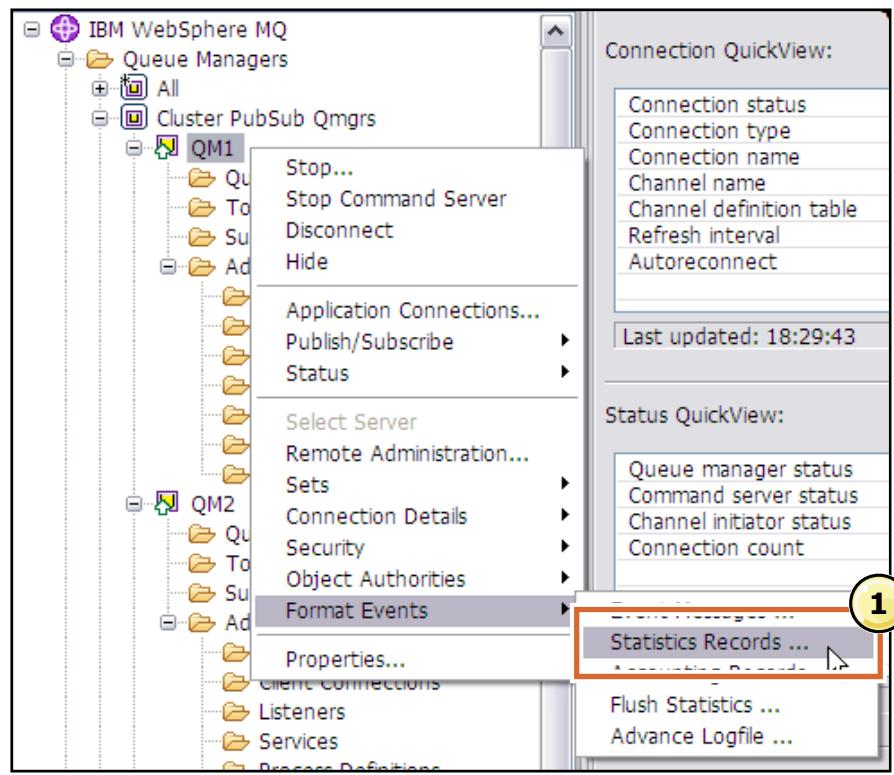
WM2091.0

### Notes:

The WebSphere MQ MSOP SupportPac provides an integrated facility to view statistics with the ability to drill down to the information required. Among its facilities is the ability to aggregate the statistics and accounting reports that MQ generates. That means that there is just one single statistics report for each queue that was accessed during the period for which the events are available.

This MSOP SupportPac provides a set of plug-ins for the MQ Explorer, extending its capabilities for configuration of queue managers, and showing additional information about those queue managers.

## MSOP accessing statistics and accounting data (1 of 2)



© Copyright IBM Corporation 2014

Figure 13-15. MSOP accessing statistics and accounting data (1 of 2)

WM2091.0

### Notes:

This figure and the next show the steps for accessing accounting statistics and data in MQ Explorer and the MSOP SupportPac. These steps assume that the MSOP SupportPac is installed and enabled.

1. Right-click the queue manager in the MQ Explorer navigator, and then click **Format Events > Statistics Records**.



## MSOP accessing statistics and accounting data (2 of 2)

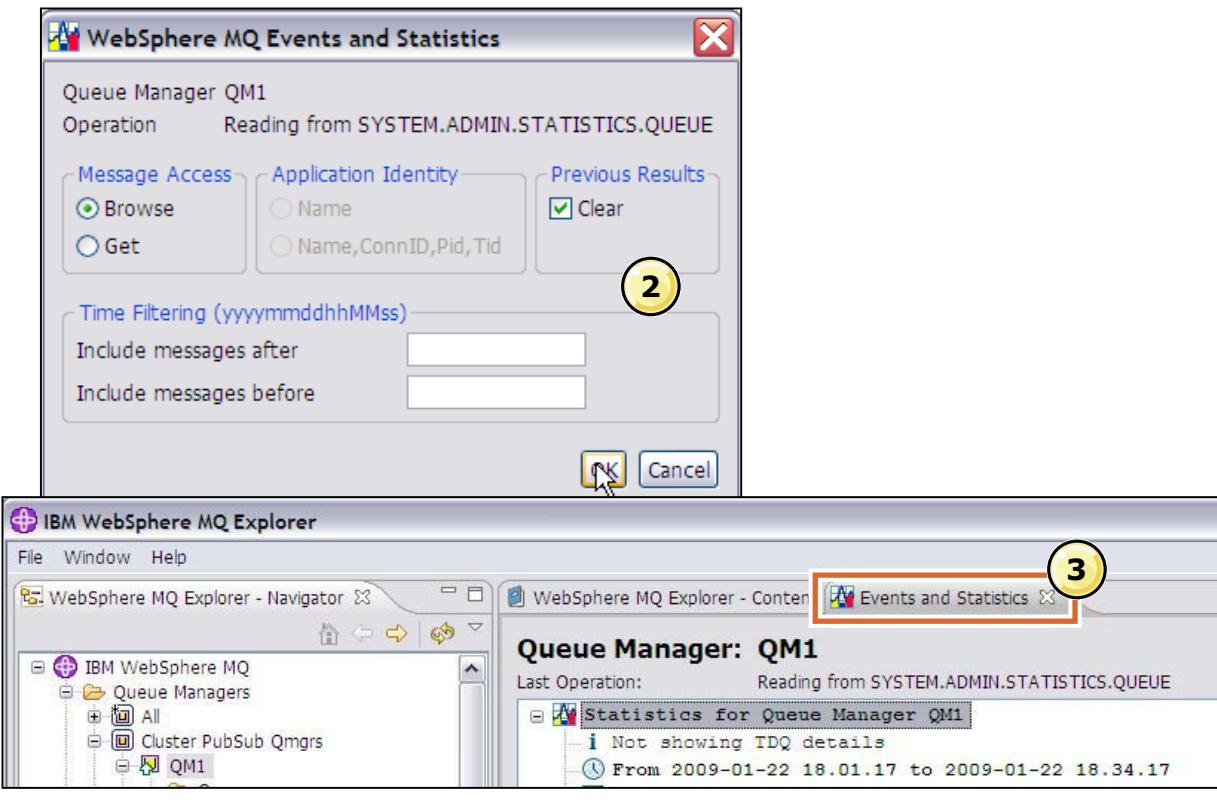


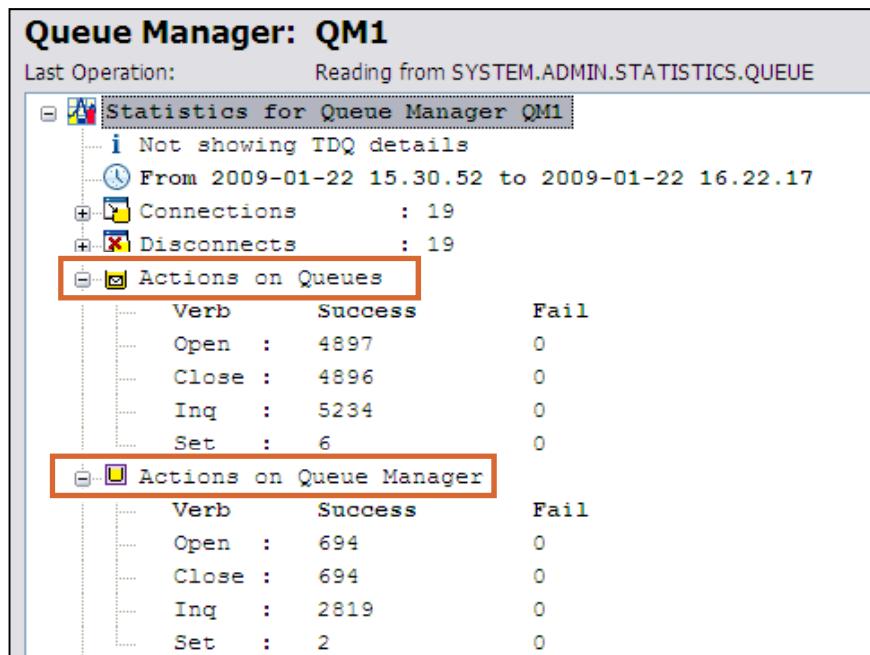
Figure 13-16. MSOP accessing statistics and accounting data (2 of 2)

WM2091.0

### Notes:

2. Specify the startup options. In most cases, you can accept the default settings. Click **OK**.
3. Select the MSOP SupportPac **Events and Statistics** tab. To return to the standard MQ views, you must click the **MQ Explorer - Content** tab.

## MSOP: Example statistics data



© Copyright IBM Corporation 2014

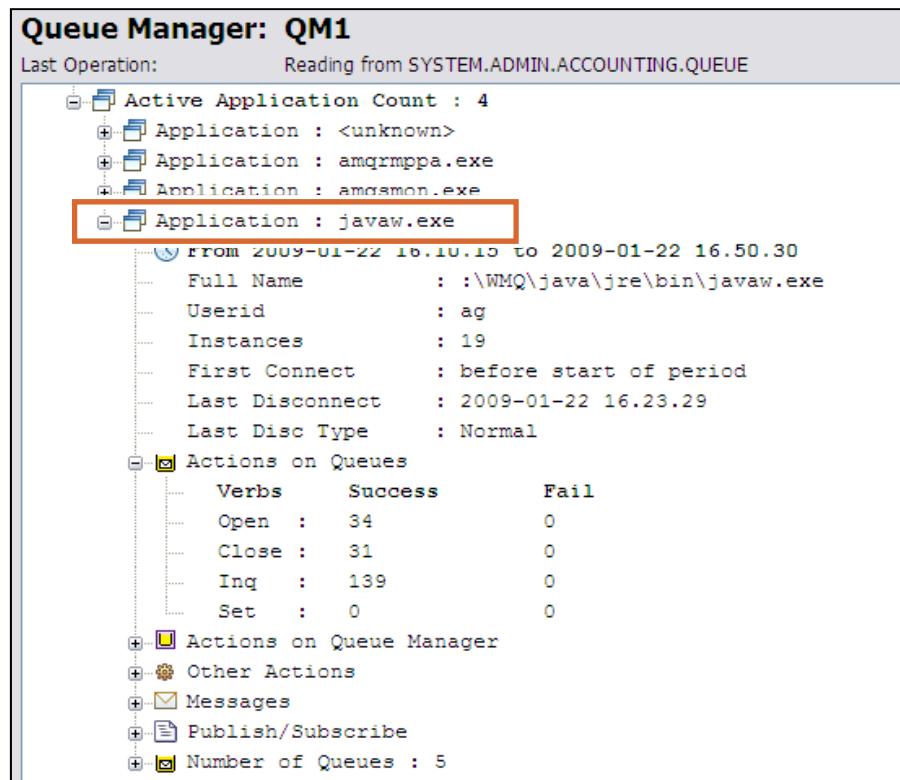
Figure 13-17. MSOP: Example statistics data

WM2091.0

### Notes:

The figure provides an example of the MSOP **Statistics and Events** tab. It shows queue manager and queue operations in each collection interval. A typical use of statistics events would be to monitor any trends in increasing activity of a queue manager for capacity planning.

## MSOP: Example accounting data



© Copyright IBM Corporation 2014

Figure 13-18. MSOP: Example accounting data

WM2091.0

### Notes:

The figure shows an example of the MSOP accounting events.

Accounting events show the activity of each application that was connected to the queue manager. A typical use of these events would be for chargeback of processing costs from the enterprise infrastructure managers to the application owner.

There is one report for each identifiable application and a summary of the activity for each application that accesses a queue.

## MSOP: Example queue accounting data

**Queue Manager: QM1**

Last Operation: Reading from SYSTEM.ADMIN.ACOUNTING.QUEUE

- Application Accounting for Queue Manager QM1
  - Not showing TDQ details
  - Active Application Count : 3
    - Application : amqrmpa.exe
    - Application : **amqspput.exe**
      - From 2009-01-22 18.12.28 to 2009-01-22 18.13.49
        - Full Name : c:\WMQ\bin\amqspput.exe
        - Userid : ag
        - Instances : 3
        - First Connect : before start of period
        - Last Disconnect : 2009-01-22 18.13.49
        - Last Disc Type : Normal
      - Actions on Queues
      - Other Actions
      - Messages
      - Publish/Subscribe
    - Number of Queues : 3
      - Queue Name : **DEPOSITI**
        - Created : 2008-12-18 20.41.59
        - Queue Type : Local
        - Def Type : Predefined
        - Open Time : 2009-01-22 18.13.49
        - Close Time : 2009-01-22 18.13.49
        - Access Success Fail
 

Access	Success	Fail
Open	1	0
Close	1	0

© Copyright IBM Corporation 2014

Figure 13-19. MSOP: Example queue accounting data

WM2091.0

### Notes:

The figure shows an example of the MSOP SupportPac queue accounting data.

The accounting data can be used to show the number of IBM MQ API operations against each queue that an application uses.



## MSOP command-line interface

- Windows batch file is provided
  - In the plug-ins directory
  - Tailor batch file to the local directory layout
  - Create a shell script to run on Linux
- Use it for running batch reports

Example:

Read the top 50 messages non-destructively from the Performance Events queue and formats them to the console

```
amqsmmon -m QMA -q SYSTEM.ADMIN.PERFM.EVENT -l 50
```

© Copyright IBM Corporation 2014

Figure 13-20. MSOP command-line interface

WM2091.0

### Notes:

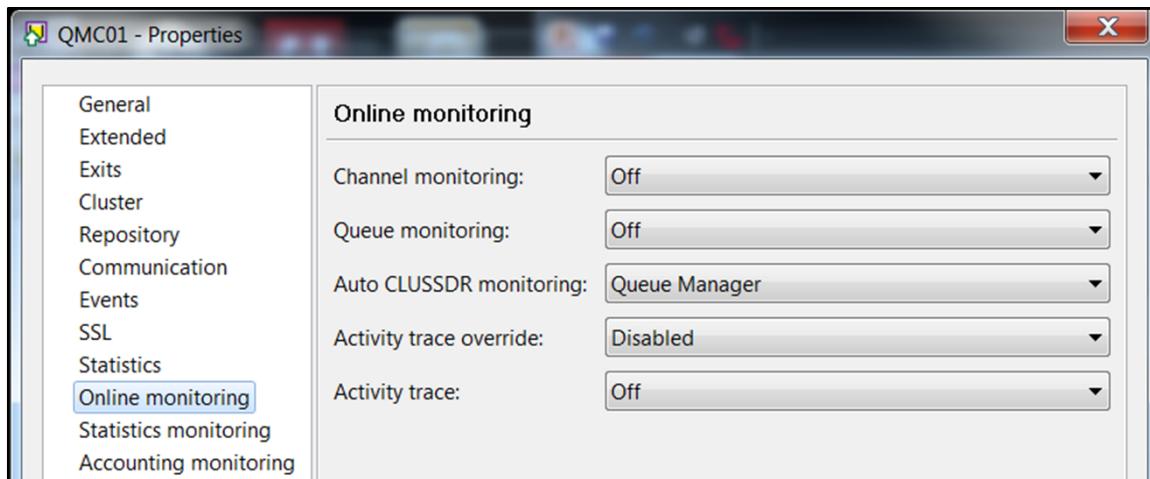
When you install the MSOP SupportPac, a batch file for Windows is included in the MQ plug-ins subdirectory. The batch file is provided so that event formatting can be run from a command line instead of MQ Explorer. It might be necessary to edit this batch file to change the location of the installation directories.

Windows batch files are limited to nine parameters, so you might not be able to put all of the options on the command. You can use the default values, or you can directly modify the batch file with the extra parameters.



## Online monitoring

- Collect monitoring data for queues and channels
- Configure by using MQSC commands or MQ Explorer queue manager properties
- Use MQ Explorer content views or MQSC **DISPLAY QSTATUS** and **DISPLAY CHSTATUS** with **MONITOR** option



© Copyright IBM Corporation 2014

Figure 13-21. Online monitoring

WM2091.0

### Notes:

The **Online monitoring** queue manager properties specify whether to collect online monitoring data about the current performance of channels that are hosted by the queue manager.

- To disable online monitoring data collection for the queue manager's channels that have the value **Queue Manager** in their **Channel monitoring** attribute, select **Off**.
- To disable online monitoring data collection for all queue manager channels regardless of the setting of the channel's **Channel monitoring** attribute, select **None**.

The **Queue monitoring** property specifies whether to collect online monitoring data about the current performance of queues that are hosted by the queue manager.

The **Auto CLUSSDR monitoring** property specifies whether to collect online monitoring data about the current performance of automatically defined cluster-sender channels.

The Activity trace property specifies whether to enable an activity trace. Activity trace was described in detail in Unit 8, "Identifying the cause of problems".

## Configuring and tuning IBM MQ for performance

- Default properties are configured to produce a fully functioning queue manager by using reasonable amounts of memory and disk space, but it is not optimized for performance
- Apply tuning to all connected queue managers because messaging performance by using more than one queue manager depends on the performance of those other queue managers
- Tuning options:
  - Queue manager log
  - Queue manager channels
  - Queue manager listeners
  - Queue buffer sizes

© Copyright IBM Corporation 2014

Figure 13-22. Configuring and tuning IBM MQ for performance

WM2091.0

### Notes:

An MQ queue manager that is created with default properties is configured to produce a fully functioning queue manager by using reasonable amounts of memory and disk space. It is not optimized for performance.

You can make some configuration changes to improve the performance of message processing with MQ.



#### Warning

It might not be necessary to implement the tuning guidelines that are described in this unit, especially if the message throughput and response time of the queue manager already meets the required level. Implementation of some of the tuning options can degrade the performance of a previously balanced system if applied inappropriately.

Carefully monitor the results of tuning the queue manager to ensure that there are no adverse effects.

## Tuning options for message types

	Applies to non-persistent messages	Applies to persistent messages
Queue manager log	No	Yes
Queue manager channels	Yes	Yes
Queue manager listeners	Yes	Yes
Queue buffer sizes	Yes	Yes

© Copyright IBM Corporation 2014

Figure 13-23. Tuning options for message types

WM2091.0

### Notes:

As shown in the figure, tuning options apply to both persistent and non-persistent messages except for the queue manager log. Non-persistent messages are held in the main memory, and the file system when the queues become deep. They are not written to the log. Only persistent messages are synchronously written to the log.



#### Note

Use persistent messages only if the message must survive a queue manager restart (made by the administrator or as the result of a power failure, communications failure, or hardware failure). The path through the queue manager for a persistent message is three times longer than for a non-persistent message.

## Queue manager log

- Only an issue when persistent messages are being processed within the queue manager
- Default settings that are configured in:
  - Default log settings in MQ Explorer properties
  - LogDefaults stanza in the `mqsc.ini` file
- Performance factors:
  - Log file location\*
  - Level of log write integrity
  - Type of logging\*
  - Log file pages
  - Log buffer pages
  - Number of primary and secondary log files
  - Number of concurrent applications
  - Application processing within a unit of work

\*Must be specified when the queue manager is created

© Copyright IBM Corporation 2014

Figure 13-24. Queue manager log

WM2091.0

### Notes:

The location, size, and number of queue manager logs can affect performance. Some performance factors, such as log file location and the type of logging, must be specified when the queue manager is created. Other performance factors such as level of log integrity, log file pages, and log buffer pages can be changed at any time but might require a restart.

Queue manager log settings can be configured in the MQ Explorer queue manager properties and the **LogDefaults** stanza in the `mqsc.ini` file.

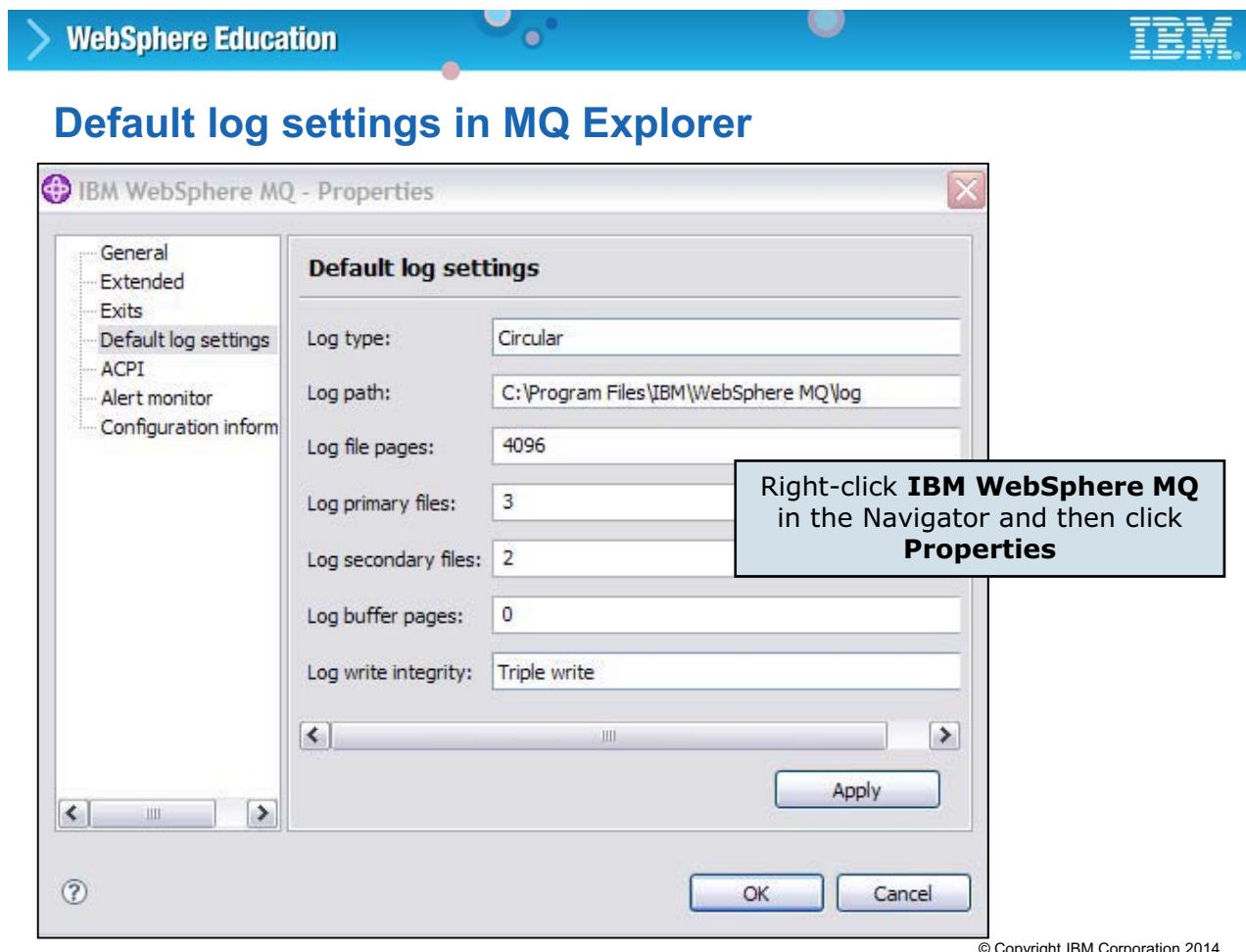


Figure 13-25. Default log settings in MQ Explorer

WM2091.0

## Notes:

The queue manager log properties that can be configured in the MQ Explorer are:

- Log type
- Log path
- Log file pages
- Log primary files
- Log secondary files
- Log buffer pages
- Log write integrity

To access the **Default log settings** in MQ Explorer:

1. Right-click the queue manager in the MQ Explorer navigator and then click **Properties**.
2. From the **Properties** list, select **Default log settings**.

## Log file location

- Locate the queue manager log on its own disk, particularly when processing large messages, or high message volumes (> 50 messages per second)
  - On Windows, create a directory
  - On UNIX, allocate a file system
- When possible, allocate the log on a device with a battery-backed write cache or use fastest local disk available

© Copyright IBM Corporation 2014

Figure 13-26. Log file location

WM2091.0

### Notes:

The log file location can affect performance.

Non-persistent messages are held in main memory and then saved to the file system as the queue depth increases. Persistent messages are synchronously written to the log.

Queue and log I/O contention can occur due to the queue manager simultaneously updating a queue file and log extent on the same disk. To avoid potential queue and log I/O contention, put queues and logs on separate and dedicated physical devices.

## Log write integrity level

- Method that the queue manager logger uses to reliably write log records
- Dependent on hardware
- Specified in the **Log** stanza of the queue manager configuration with the **LogWriteIntegrity** parameter
  - **SingleWrite**: Safe for the logger to write log records in a single write as the hardware assures full write integrity
  - **TripleWrite**: Default value that assures write integrity because full write integrity hardware is not available
- For performance, choose **SingleWrite** if hardware assures full write integrity

© Copyright IBM Corporation 2014

Figure 13-27. Log write integrity level

WM2091.0

### Notes:

MQ provides full write integrity of logs where hardware that assures write integrity is not available.

Some hardware guarantees that, if a write operation writes a page and fails for any reason, a subsequent read of the same page results in each byte in the buffer that is the same as before the write.

On this type of hardware, it is safe for the logger to write log records in a single write as the hardware assures full write integrity. This method provides the highest level of performance.

The log write integrity level can be configured in the MQ Explorer queue manager properties and in the **Log** stanza of the queue manager configuration file.

## Type of logging

- Linear logging file extents are continually allocated as required
  - Choose to be able to forward recover queue data that follows a failure or recover from media failure of the device that contains the log
- Circular logging log file extents are reused after they no longer contain active log data
- For performance, choose circular logging if linear logging not required

© Copyright IBM Corporation 2014

Figure 13-28. Type of logging

WM2091.0

### Notes:

The type of logging can affect performance.

Circular logging is more efficient because log files are reused when they no longer contain active log data. If circular logging is enabled, ensure that **Log primary files** are greater than 20. Ensure that primary logs satisfy circular logging because secondary logs are formatted each time that they are reused.

## Log file pages

- Specified on queue manager creation and cannot be changed
- Defines the size of one physical disk extent in units of 4 KB pages
- For Windows:
  - Default number of log file pages is 4096, giving a log file size of 16 MB
  - Minimum number of log file pages is 32
  - Maximum number of log file pages is 65536
- For UNIX:
  - Default number of log file pages is 4096, giving a log file size of 16 MB
  - Minimum number of log file pages is 64
  - Maximum number of log file pages is 65,535
- For performance, allocate maximum size if disk space is available

© Copyright IBM Corporation 2014

Figure 13-29. Log file pages

WM2091.0

### Notes:

The **Log file pages** option in the queue manager log properties defines the size of one physical disk extent.

The size of the disk extent is directly related to the elapsed times between changing disk extents. It is better to have a smaller number of large extents. The largest size reduces the frequency of switching extents.

Log file page size can be configured in the MQ Explorer queue manager properties.

## Log buffer pages

- Log buffer is a circular piece of main memory where the log records are concatenated so that multiple log records can be written to the log file in a single I/O operation
- Log buffer size
  - Default is 512 pages
  - Maximum is 4096 pages
- Improve persistent message throughput of large messages (message size > 1 MB) by increasing **Log buffer pages** to improve likelihood of messages that need only one I/O to get to the disk

© Copyright IBM Corporation 2014

Figure 13-30. Log buffer pages

WM2091.0

### Notes:

To improve persistent message throughput of large messages, increase the value in the **Log buffer pages** property to improve the likelihood of messages that need only one I/O to get to the disk. A large message is a message greater than 1 MB.

You can reduce the memory by using a smaller log buffer page without impacting throughput in environments that process a few (under 100) small persistent messages. A small message is a message less than 10 KB.

## Number of log files

- Queue manager allocates and formats primary log file extents when it is first started or when extra extents are added
  - Minimum number is 2
  - Maximum number is 254 on Windows, or 510 on UNIX
  - Default number is 3
- Queue manager dynamically allocates secondary log file extents when the primary files are exhausted
  - Minimum number is 1
  - Maximum number is 253 on Windows, or 509 on UNIX
  - Default number is 2
- Total number of primary and secondary log files:
  - Must not exceed 255 on Windows, or 511 on UNIX
  - Must not be less than 3
- For performance, ensure that there are a reasonable number of secondary extents for system activity

© Copyright IBM Corporation 2014

Figure 13-31. Number of log files

WM2091.0

### Notes:

Records are written into the log buffer with each put, get, and commit. This information is moved onto the log disk. Periodically the checkpoint process decides how many of these log file extents are in the *active* log and must be kept online for recovery purposes. The log extents no longer in the active log are available for achieving when using linear logs or available for reuse when using circular logs.

Ensure that there are sufficient primary logs to hold the active log plus the new log extents that are used until the next checkpoint; otherwise some secondary logs are temporarily included in the log set. Secondary logs must be instantly formatted which is an unnecessary delay when using circular logging.

The value for the number of logs is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

## Queue manager channels performance tuning

- Run as trusted or fast path IBM MQ applications to give a performance benefit through reduced code pathlength
- Configure by using one of two options:
  - Specify a value of **MQIBindType=FASTPATH** in the **Channels** stanza of the **qm.ini** or registry file
  - Set the environment variable **MQ\_CONNECT\_TYPE = FASTPATH** in the environment in which the channel is started
- Considerations:
  - If channel exits are used, there is the potential for the exit to corrupt the queue manager if the exits are not correctly written and thoroughly tested
  - If **STOP CHANNEL (TERMINATE)** command is used
  - If the environment is unstable with regular component failure

© Copyright IBM Corporation 2014

Figure 13-32. Queue manager channels performance tuning

WM2091.0

### Notes:

Fastpath channels, can increase throughput for both non-persistent and persistent messaging. For persistent messages, the improvement is for the path through the queue manager, and does not affect performance when writing to the log disk.



#### Note

Because the greater proportion of time for persistent messages is in the process of writing to the log disk, the performance improvement for fastpath channels is less apparent with persistent messages than with non-persistent messages.

Setting the **MQIBindType** attribute in the **qm.ini** file to **FASTPATH** causes the channel to run in ‘trusted’ mode. Trusted applications do not use a thread in the agent process. There is no interprocess communication (IPC) between the channel and agent because the agent does not exist in this connection.

If the channel is run in STANDARD mode, then any messages that are passed between the channel and agent use IPC memory that is dynamically obtained and only held for the lifetime of the MQGET. Standard channels each require an extra 80 KB of memory. When the message rate increases, more IPC memory is used in parallel.

## Queue manager listeners performance tuning

- Run as trusted or fast path IBM MQ applications to give a performance benefit through reduced code pathlength
- Set the environment variable **MQ\_CONNECT\_TYPE=FASTPATH** in the environment in which the listener is started
  
- Works for listeners that are started when the **runmq1sr** command is run manually or in a script
  - **MQ\_CONNECT\_TYPE=FASTPATH** needs to be present only in the shell from which the **runmq1sr** command is entered
- Works for listeners that were defined by using the **DEFINE LISTENER** MQSC command
  - **MQ\_CONNECT\_TYPE=FASTPATH** must be set in the environment in which the queue manager is started

© Copyright IBM Corporation 2014

Figure 13-33. Queue manager listeners performance tuning

WM2091.0

### Notes:

Running queue manager listeners as trusted or fastpath applications might give a performance benefit through reduced code path length.

More resource savings are available by using the **runmq1sr** listener command rather than InetD, including a reduced requirement on virtual memory, number of processes, and file handles.



#### Note

The performance improvement for fastpath channels is less apparent with persistent messages than with non-persistent messages.

## Queue buffer sizes

- Each queue has two buffers to hold messages
  - Buffer for non-persistent messages has a default size of 64 K
  - Buffer for persistent messages has a default size of 256 K for 64-bit queue managers
  - Maximum size that is supported for each queue buffer is 1 Mb
- You can change the buffer sizes to limit times when buffers fill and messages are written to the file system
  - When choosing values, determine the average size and average number of messages on the queue
  - Defining queues by using large queue buffers can degrade performance if the system is short of real memory

© Copyright IBM Corporation 2014

Figure 13-34. Queue buffer sizes

WM2091.0

### Notes:

Increasing the queue buffer provides the capability to absorb peaks in message throughput at the expense of real storage. After these queue buffers are full, the additional message data is given to the file system that eventually finds its way to the disk.

Defining queues by using large non-persistent or persistent queue buffers can degrade performance if the system is short of real memory because many queues were already defined with large buffers.

 **Note**

The queue buffers are allocated in shared storage so consideration must be given to whether the agent process or application process has the memory addressability for all the required shared memory segments.

## Changing queue buffer sizes

1. Delete any existing queues that you want to change.
2. Stop the queue manager.
3. Add or change the **TuningParameters** stanza in the Windows Registry or **qm.ini** file.
  - Non-persistent queue buffer size is specified in **DefaultQBufferSize**
  - Persistent queue buffer size is specified in **DefaultPQBufferSize**
4. Restart the queue manager.
5. Define the queues.

© Copyright IBM Corporation 2014

Figure 13-35. Changing queue buffer sizes

WM2091.0

### Notes:

This figure lists the steps for changing queue buffer sizes.

Queues can be defined with different values of **DefaultQBufferSize** and **DefaultPQBufferSize**. The value is taken from the **TuningParameters** stanza in use by the queue manager when the queue was defined.

When the queue manager is restarted, existing queues keep their earlier definitions and new queues are created with the current setting. When a queue is opened, resources are allocated according to the definition held on disk from when the queue was created.



## Unit summary

Having completed this unit, you should be able to:

- Describe the statistics and accounting data that MQ provides
- View and generate accounting and statistical data
- Interpret statistics and accounting data to identify possible system performance benefits
- Configure and tune MQ for improved performance

© Copyright IBM Corporation 2014

---

Figure 13-36. Unit summary

WM2091.0

### Notes:

## Checkpoint questions

1. Which of the following are types of statistics collectable by IBM MQ?
  - a. Queue
  - b. Process
  - c. Channel
  - d. Queue manager
  
2. True or False: The following command would clear statistics: **RESET QMGR TYPE (STATISTICS)**
  
3. Select all answers that are correct for the following statement.  
Accounting data includes:
  - a. Application data
  - b. Queue data
  - c. Publish/subscribe data
  - d. Byte counts
  - e. All of the above

© Copyright IBM Corporation 2014

Figure 13-37. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

- 1.
  
- 2.
  
- 3.



## Checkpoint answers

1. **a, c, and d**
2. **False.** That command flushes statistics to the statistics queue before the expiration of the collection interval. It does not clear the statistics.
3. **e**

© Copyright IBM Corporation 2014

Figure 13-38. Checkpoint answers

WM2091.0

### Notes:

## Exercise 11



Monitoring and configuring IBM MQ  
for performance

© Copyright IBM Corporation 2014

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

R.O

Figure 13-39. Exercise 11

WM2091.0

### Notes:

In this exercise, you enable and configure the online monitoring, statistics, and accounting features of MQ.



## Exercise objectives

After completing this exercise, you should be able to:

- Enable accounting and statistics collection in IBM MQ
- View accounting and statistics data

© Copyright IBM Corporation 2014

---

Figure 13-40. Exercise objectives

WM2091.0

### Notes:

See the *Student Exercises Guide* for detailed instructions.

# Unit 14. JMS administration overview

## What this unit is about

This unit provides an overview of how IBM MQ supports the Java Message Service.

## What you should be able to do

After completing this unit, you should be able to:

- Describe IBM MQ as a Java Message Service (JMS) provider
- Describe the advantages and disadvantages of using JMS instead of Java IBM MQ calls

## How you will check your progress

- Checkpoints

## References

IBM MQ product documentation



## Unit objectives

After completing this unit, you should be able to:

- Describe IBM MQ as a Java Message Service (JMS) provider
- Describe the advantages and disadvantages of using JMS instead of Java IBM MQ calls

© Copyright IBM Corporation 2014

---

Figure 14-1. Unit objectives

WM2091.0

### Notes:

## What is JMS?

- Messaging standard that allows application components that are based on the Java Enterprise Edition to create, send, receive, and read messages
- Supports point-to-point and publish/subscribe messaging styles
- Requires a JMS provider
- Current version is JMS 2.0
  - Deferred message delivery
  - Shared subscriptions
  - Asynchronous send operation

© Copyright IBM Corporation 2014

Figure 14-2. What is JMS?

WM2091.0

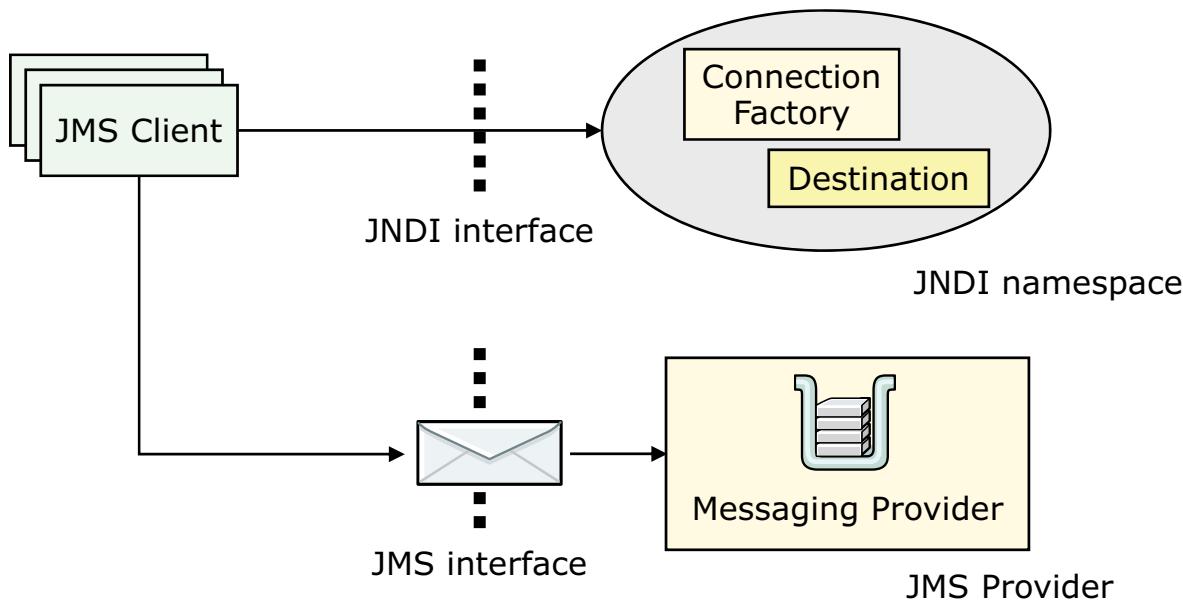
### Notes:

JMS is a messaging API specification. It defines how a Java application accesses messaging resources with the Java Platform, Enterprise Edition framework, but not exclusively so. That is, stand-alone JMS applications are valid. The current version of the JMS standard is V2.0.

Like all Java specifications, operating system neutrality is a key design focus. Object code portability flows from this neutrality.

JMS supports both the point-to-point and publish/subscribe messaging styles.

## JMS architecture



© Copyright IBM Corporation 2014

Figure 14-3. JMS architecture

WM2091.0

### Notes:

This figure provides an overview of the JMS architecture.

Messaging applications generally consist of the following parts:

- Messaging provider: The enterprise messaging system, such as MQ
- JMS clients: Java programs that produce and use messages by using JMS
- Messages: The objects that communicate information between clients
- Administered objects: Preconfigured JMS objects that an administrator creates for the clients

JNDI (Java Naming and Directory Interface) is a standard Java extension that provides a uniform API for accessing various directory and naming services. JMS clients use JNDI to browse a naming service to obtain references to administered objects. Administered objects are the JMS connection factory and JMS destination objects, where JMS destination objects are topics and queues. A system administrator creates and administers the destination objects and connection factory.

JNDI-administered objects are stored in the bindings. These bindings can be stored in a file system or in an LDAP repository. A naming service associates name with distributed objects so that the administered objects are located through their names, not complex network addresses. JNDI

provides an abstraction that hides the specifics of the naming service, which makes client applications more portable.

A JMS client specifies a JNDI *InitialContext* to obtain a JNDI connection to the JMS messaging server. The *InitialContext* is the starting point in any JNDI lookup and acts like the root of a file system. The JMS directory service that is being used determines the properties that are used to create an *InitialContext*.



## JMS support in IBM MQ

- IBM MQ for JMS and JMS are both message-oriented middleware
- IBM MQ is a JMS provider and supports JMS 2.0
- MQ Explorer supports JMS object administration
  - Queues
  - Connection factories
  - Destinations

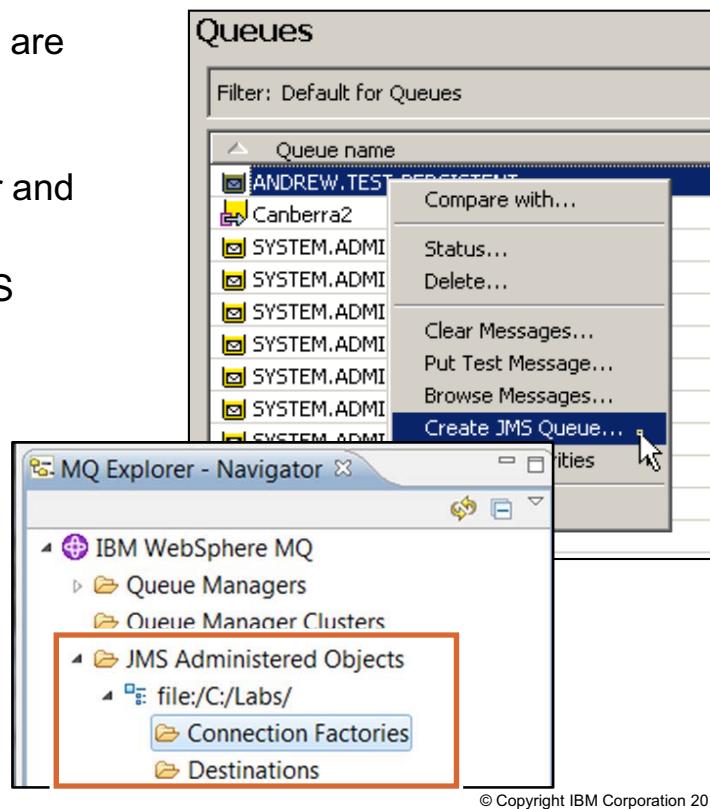


Figure 14-4. JMS support in IBM MQ

WM2091.0

### Notes:

The implementation of MQ classes for JMS is not dependent on MQ classes for Java. MQ classes for Java and MQ classes for JMS are peers that use a common Java interface to the MQI.

You can administer JMS objects in MQ by using MQ Explorer or the JMS Administration tool.

## IBM MQ classes for JMS or JMS

### IBM MQ classes for JMS

Encapsulate MQI

Easy to implement for anyone familiar with MQI in other languages

Can take advantage of the full features of IBM MQ

Similar object model to other object-oriented language interfaces such as C++ and .NET

### JMS

Implement JMS interfaces

Easy to implement for anyone with JMS skills

Can use message-driven beans

Can connect to other JMS providers

Bridges applications between JMS providers

© Copyright IBM Corporation 2014

Figure 14-5. IBM MQ classes for JMS or JMS

WM2091.0

### Notes:

The figure lists some of the considerations for choosing between JMS and MQ classes for JMS.

The MQ classes JMS have a close correlation to the underlying MQI. If you are familiar with programming in the MQI, you should have little trouble adapting that knowledge to the classes for JMS. All full MQ functions can be used.

Some of the JMS considerations are already described. Message-driven beans are method for driving code when a message arrives on a destination.

If the need arises to bridge JMS providers of any type, it is easy with a JMS implementation.

## IBM JMS extensions

- Provide a greater level of consistency across IBM JMS providers
- Can be used with any messaging provider
- Use these extensions to:
  - Create connection factories and destinations dynamically at run time instead of retrieving them from JNDI namespace
  - Set the properties of IBM MQ classes for JMS objects
  - Obtain detailed information about a problem
  - Control tracing
  - Obtain version information about IBM MQ classes for JMS

© Copyright IBM Corporation 2014

Figure 14-6. IBM JMS extensions

WM2091.0

### Notes:

IBM MQ V8 classes for JMS provide a more generic set of extensions to the JMS API, which are not specific to MQ as the messaging system. These extensions are known as the IBM JMS extensions.

## IBM MQ JMS administration utility

- Command utility to define administered object to JNDI namespace
- Uses a configuration file (`JMSAdmin.config`) to set the values of certain properties
  - Initial context factory
  - Service provider URL
  - Security authentication type

Example configuration file

```
#Set the service provider
    INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
#Set the initial context
    PROVIDER_URL=ldap://polaris/o=ibm_us,c=us
#Set the authentication type
    SECURITY_AUTHENTICATION=none
```

© Copyright IBM Corporation 2014

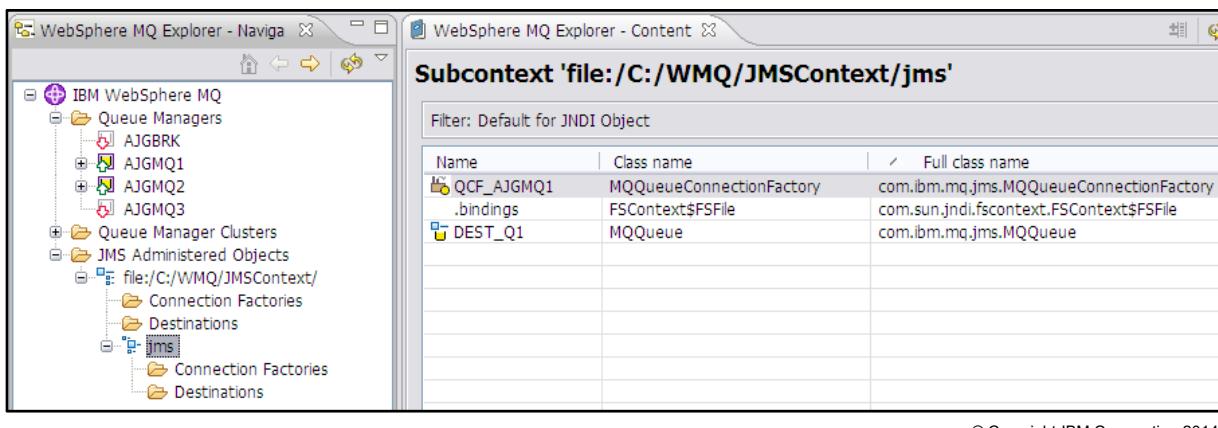
Figure 14-7. IBM MQ JMS administration utility

WM2091.0

### Notes:

## MQ Explorer administration for JMS

- Add an initial context to define the root of the JNDI namespace
- Create a JMS connection factory to connect JMS client to IBM MQ by using:
  - MQ queue manager
  - MQ channel
  - MQ listener
- Create destination objects that represent queues or topics



© Copyright IBM Corporation 2014

Figure 14-8. MQ Explorer administration for JMS

WM2091.0

### Notes:

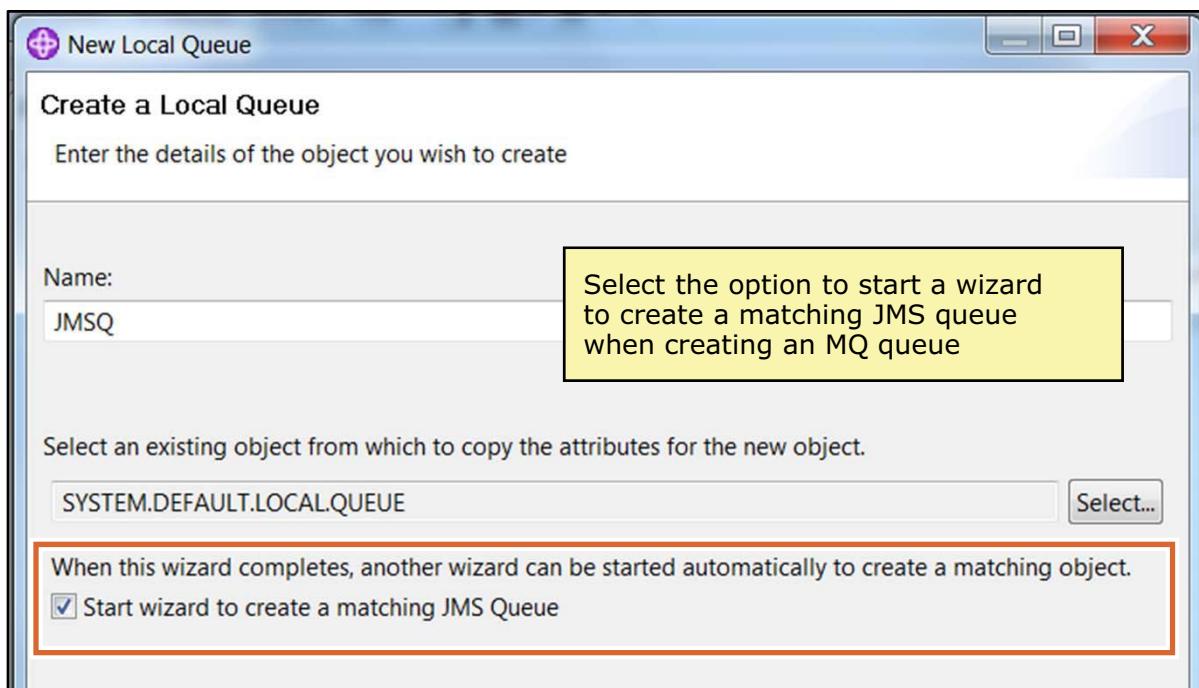
MQ can administer JMS queues and objects. The first step is to identify the type and location of the JNDI repository by configuring the Initial Context.

The information in the figure shows how to start defining the initial context. The bottom screen capture is a summary of the JMS subcontext.

After it is connected, the MQ Explorer can be used to complete most JMS administration tasks.



## Create JMS and MQ queues simultaneously



© Copyright IBM Corporation 2014

Figure 14-9. Create JMS and MQ queues simultaneously

WM2091.0

### Notes:

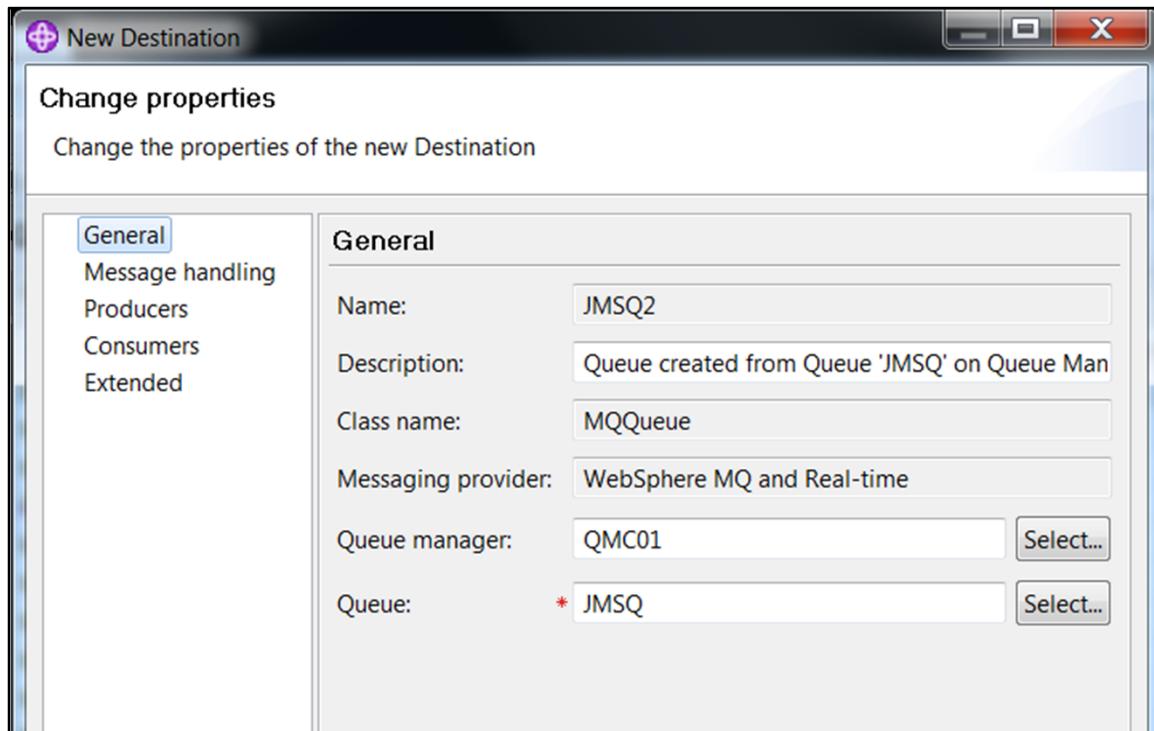
You can create a JMS queue the same time you create a local queue in MQ Explorer by clicking **Start wizard to create a matching JMS Queue**. The **New JMS queue** wizard starts after the local queue is created.

After the JMS queue is created, it appears in the **Destinations** view.

WebSphere Education

IBM

## JMS queue properties



© Copyright IBM Corporation 2014

Figure 14-10. JMS queue properties

WM2091.0

### Notes:

You define and modify the JMS queue properties by right-clicking the JMS queue in the **Destinations** view and then clicking **Properties**.

The JMS queue properties include:

- Message handling properties
- Producer and consumer properties
- Extended properties

An application uses a message producer to send messages to a destination and a message consumer to receive messages that are sent to a destination.

## Unit summary

Having completed this unit, you should be able to:

- Describe IBM MQ as a Java Message Service (JMS) provider
- Describe the advantages and disadvantages of using JMS instead of Java IBM MQ calls

© Copyright IBM Corporation 2014

Figure 14-11. Unit summary

WM2091.0

### Notes:



## Checkpoint questions

1. Which of the following statements are NOT true:
  - a. JMS is an API
  - b. JMS is platform neutral
  - c. JMS part of Java Platform, Enterprise Edition
  - d. JMS is programming language neutral
  
2. True or False: You must use MQ Explorer to define the JMS administered objects.

© Copyright IBM Corporation 2014

---

Figure 14-12. Checkpoint questions

WM2091.0

### Notes:

Write your answers here:

1.

2.



## Checkpoint answers

1. **d.** JMS allows application components that are based on the Java Enterprise Edition (Java EE) to create, send, receive, and read message.
2. **False.** You can also use the MQ Administration tool or the tools that the application server provides.

© Copyright IBM Corporation 2014

Figure 14-13. Checkpoint answers

WM2091.0

### Notes:



# Unit 15. Course summary

## What this unit is about

This unit summarizes the course.

## What you should be able to do

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study



## Unit objectives

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2014

---

Figure 15-1. Unit objectives

WM2091.0

### Notes:

## Course learning objectives

Having completed this course, you should be able to:

- Plan the implementation of IBM MQ
- Install IBM MQ
- Perform basic customization and administration tasks
- Enable a queue manager to exchange messages with another queue manager
- Use a trigger message and a trigger monitor to start an application to process messages
- Implement basic queue manager restart and recovery procedures
- Identify the cause of a problem
- Plan for and implement basic IBM MQ security features
- Use accounting and statistics messages to monitor the activities in an IBM MQ system
- Use IBM MQ sample programs to test queue manager, queue, channel, and client configuration

© Copyright IBM Corporation 2014

Figure 15-2. Course learning objectives

WM2091.0

### Notes:



## To learn more on the subject

- IBM Training website:

[www.ibm.com/training](http://www.ibm.com/training)

- IBM MQ V8 product website:

<http://www.ibm.com/software/products/en/ibm-mq>

- IBM Knowledge Center for IBM MQ V8:

[http://www.ibm.com/support/knowledgecenter/SSFKSJ\\_8.0.0/welcome/WelcomePagev8r0.html](http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/welcome/WelcomePagev8r0.html)

© Copyright IBM Corporation 2014

Figure 15-3. To learn more on the subject

WM2091.0

### Notes:

## Unit summary

Having completed this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2014

Figure 15-4. Unit summary

WM2091.0

### Notes:



# Appendix A. List of abbreviations

<b>ACL</b>	Access control list
<b>API</b>	Application programming interface
<b>FFST</b>	First Failure Support Technology
<b>IBM</b>	International Business Machines Corporation
<b>IPC</b>	interprocess communication
<b>JMS</b>	Java Message Service
<b>JNDI</b>	Java Naming and Directory Interface
<b>MCA</b>	Message channel agent
<b>MQAI</b>	IBM MQ Administration Interface
<b>MQI</b>	IBM MQ Message Queue Interface
<b>MQDLH</b>	IBM MQ dead-letter header
<b>MQMD</b>	IBM MQ message descriptor
<b>MQTMC2</b>	IBM MQ trigger message 2 character format
<b>MSCS</b>	Microsoft Cluster Service
<b>NetBIOS</b>	Network Basic Input/Output System
<b>OAM</b>	Object Authority Manager
<b>PCF</b>	Programmable control format
<b>RCP</b>	Rich client platform
<b>SPX</b>	Sequenced Packet Exchange
<b>SNA</b>	Systems Network Architecture
<b>SSL</b>	Secure Sockets Layer
<b>TCP/IP</b>	Transmission Control Protocol/Internet Protocol



# Appendix B. Resource guide

Completing this WebSphere Education course is a great first step in building your WebSphere, CICS, and SOA skills. Beyond this course, IBM offers several resources to keep your WebSphere skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites.

## Training

- **IBM Training website**
  - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
  - For more information, see: <http://www.ibm.com/training>
- **IBM Training News**
  - Review or subscribe to updates from IBM and its Business Partners.
  - For more information, see: <http://bit.ly/IBMTrafficEN>
- **IBM Certification**
  - You can demonstrate to your employer or clients your new WebSphere, CICS, or SOA mastery through achieving IBM Professional Certification. WebSphere certifications are available for developers, administrators, and business analysts.
  - For more information, see: <http://www.ibm.com/certify>
- **Training paths**
  - Find your next course easily with IBM training paths. Training paths provide a visual flow-chart style representation of training for many WebSphere products and roles, including developers and administrators.
  - For more information, see: <http://www.ibm.com/services/learning/sites.wss/us/en?pageType=page&c=a0003096>

## Social media links

You can keep in sync with WebSphere Education, including new courses and certifications, course previews, and special offers, by visiting any of the following social media websites.

- **Twitter**

- Receive short and concise updates from WebSphere Education a few times each week.
- Follow WebSphere Education at: [twitter.com/websphere\\_edu](http://twitter.com/websphere_edu)
- **Facebook:**
  - Become a fan of IBM Training on Facebook to keep in sync with the latest news and career trends, and to post questions or comments.
  - Find IBM Training at: [facebook.com/ibmtraining](http://facebook.com/ibmtraining)
- **YouTube:**
  - Visit the IBM Training YouTube channel to learn about IBM training programs and courses.
  - Find IBM Training at: [youtube.com/IBMTutorial](http://youtube.com/IBMTutorial)

## **Support**

- **WebSphere Support portal**
  - The WebSphere Support website provides access to a portfolio of support tools. From the WebSphere Support website, you can access several downloads, including troubleshooting utilities, product updates, drivers, and Authorized Program Analysis Reports (APARs). To collaboratively solve issues, the support website is a clearing house of links to online WebSphere communities and forums. The IBM support website is now customizable so you can add and delete portlets to the information most important to the WebSphere products you work with.
  - For more information, see:  
<http://www.ibm.com/software/websphere/support>
- **IBM Support Assistant**
  - The IBM Support Assistant is a local serviceability workbench that makes it easier and faster for you to resolve software product issues. It includes a desktop search component that searches multiple IBM and non-IBM locations concurrently and returns the results in a single window, all within IBM Support Assistant.
  - IBM Support Assistant includes a built-in capability to submit service requests; it automatically collects key problem information and transmits it directly to your IBM support representative.
  - For more information, see: <http://www.ibm.com/software/support/isa>
- **WebSphere Education Assistant**
  - IBM Education Assistant is a collection of multimedia modules that are designed to help you gain a basic understanding of IBM Software products

and use them more effectively. The presentations, demonstrations, and tutorials that are part of the IBM Education Assistant are an ideal refresher for what you learned in your WebSphere Education course.

- For more information, see:

<http://www.ibm.com/software/info/education/assistant/>

## WebSphere documentation and tips

- **IBM Redbooks**

- The IBM International Technical Support Organization develops and publishes IBM Redbooks publications. IBM Redbooks are downloadable PDF files that describe installation and implementation experiences, typical solution scenarios, and step-by-step “how-to” guidelines for many WebSphere products. Often, Redbooks include sample code and other support materials available as downloads from the site.
- For more information, see: <http://www.ibm.com/redbooks>

- **IBM documentation and libraries**

- Information centers and product libraries provide an online interface for finding technical information on a particular product, offering, or product solution. The information centers and libraries include various types of documentation, including white papers, podcasts, webcasts, release notes, evaluation guides, and other resources to help you plan, install, configure, use, tune, monitor, troubleshoot, and maintain WebSphere products. The WebSphere information center and library are located conveniently in the left navigation on WebSphere product web pages.

- **developerWorks**

- IBM developerWorks is the web-based professional network and technical resource for millions of developers, IT professionals, and students worldwide. IBM developerWorks provides an extensive, easy-to-search technical library to help you get up to speed on the most critical technologies that affect your profession. Among its many resources, developerWorks includes how-to articles, tutorials, skill kits, trial code, demonstrations, and podcasts. In addition to the WebSphere zone, developerWorks also includes content areas for Java, SOA, web services, and XML.
- For more information, see: <http://www.ibm.com/developerworks>

## WebSphere Services

- IBM Software Services for WebSphere are a team of highly skilled consultants with broad architectural knowledge, deep technical skills, expertise on suggested practices, and close ties with IBM research and development labs. The WebSphere Services team offers skills transfer, implementation, migration, architecture, and design services, plus customized workshops. Through a worldwide network of services specialists, IBM Software Service for WebSphere makes it easy for you to design, build, test, and deploy solutions, helping you to become an on-demand business.
- For more information, see:  
<http://www.ibm.com/developerworks/websphere/services/>



