

Course Guide

# IBM MQ V9 Advanced System Administration (Distributed)

Course code WM213 / ZM213 ERC 1.0



## May 2017 edition

### Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

### Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

**© Copyright International Business Machines Corporation 2017.**

**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>xiv</b>
<b>Agenda .....</b>	<b>xvi</b>
<b>Unit 1. Managing clients and client connections .....</b>	<b>1-1</b>
How to check online for course material updates .....	1-2
Unit objectives .....	1-3
1.1. Sharing conversations .....	1-4
Sharing conversations overview .....	1-5
Sharing conversation requirements .....	1-6
SHARECNV channel property .....	1-7
Sharing conversations configuration: Server-connection .....	1-9
Sharing conversations configuration: Client-connection .....	1-10
Sharing conversations example .....	1-11
Sharing conversations channel status fields .....	1-12
Sharing conversations and existing channel status fields .....	1-13
Sharing conversations and channel exit considerations .....	1-14
Sharing conversation performance .....	1-15
1.2. Read ahead .....	1-16
Standard MQGET processing .....	1-17
Read ahead overview .....	1-18
Read ahead advantages .....	1-19
Read ahead considerations .....	1-20
Default read ahead configuration (1 of 2) .....	1-21
Read ahead configuration (2 of 2) .....	1-22
Read ahead memory buffer size .....	1-23
Read ahead buffer updates .....	1-24
Using IBM MQ Explorer to get read ahead status .....	1-25
Using MQSC to get read ahead status .....	1-26
1.3. Asynchronous put .....	1-27
Asynchronous put introduction .....	1-28
MQPUT processing .....	1-29
MQPUT in client mode with asynchronous put .....	1-30
When asynchronous put applies .....	1-31
Sync point .....	1-32
Setting the default put response type on a queue .....	1-33
1.4. Client modes for MQSC .....	1-34
MQSC client modes .....	1-35
MQSC client-local mode (-n) .....	1-36
MQSC client-connect mode (-c) .....	1-37
1.5. Troubleshooting tips .....	1-38
Troubleshooting IBM MQ channels .....	1-39
Listener problems .....	1-40
Channel configuration problems .....	1-41
Network connectivity problems .....	1-42
In-doubt channels .....	1-43
MQSC connection and channel status commands .....	1-44
Channel monitoring .....	1-45

Channel monitoring example .....	1-46
Troubleshooting IBM MQ clients .....	1-47
Unit summary .....	1-48
Review questions (1 of 2) .....	1-49
Review questions (2 of 2) .....	1-50
Review answers (1 of 2) .....	1-51
Review answers (2 of 2) .....	1-52
<b>Unit 2. Securing IBM MQ channels with TLS .....</b>	<b>2-1</b>
Unit objectives .....	2-2
2.1. TLS overview .....	2-3
The security problems .....	2-4
Data encryption and signing .....	2-5
Elementary encryption: Symmetric key .....	2-6
Elementary encryption: Asymmetric keys .....	2-7
Digital signature .....	2-8
Relative authentication .....	2-9
The key distribution problem .....	2-10
Digital certificates .....	2-11
Trusting a digital certificate .....	2-12
Certificate revocation .....	2-13
Digital certificate details .....	2-14
Distinguished name .....	2-15
Transport Layer Security (TLS) concepts .....	2-16
2.2. Implementing TLS in IBM MQ .....	2-17
TLS support in IBM MQ .....	2-18
Cipher specification support in IBM MQ .....	2-19
IBM MQ connection procedure with TLS .....	2-20
TLS and IBM MQ certificate exchange .....	2-21
Configuring TLS on queue managers .....	2-22
Managing certificates .....	2-23
Creating certificates .....	2-24
Creating a key repository with IBM Key Management (1 of 2) .....	2-25
Creating a key repository with IBM Key Management (2 of 2) .....	2-26
Using IBM Key Management to create a self-signed personal certificate .....	2-27
Using IBM Key Management to add a CA certificate .....	2-28
Command options for managing certificates and keys .....	2-29
Using runmqakm to create and initialize a key database .....	2-30
Using runmqakm to generate a self-signed certificate .....	2-31
Key repository .....	2-32
Labeling the certificate .....	2-34
Benefits of labeling the certificate .....	2-35
Configuring TLS on queue manager with IBM MQ Explorer .....	2-36
Configuring TLS on channels .....	2-37
Distinguished Name matching support in IBM MQ .....	2-38
Defining certificates for channels (1 of 2) .....	2-39
Defining certificates for channels (2 of 2) .....	2-40
Channel attributes for TLS .....	2-41
SSLPEER .....	2-42
SSLPEER mapping .....	2-43
Displaying certificate details .....	2-44
Checking certificate status with OCSP .....	2-45
Setting OSCP properties in IBM MQ Explorer (1 of 2) .....	2-46
Setting OSCP properties in IBM MQ Explorer (2 of 2) .....	2-47
Accessing CRLs and ARLs .....	2-48
Accessing CRLs and ARLs by using IBM MQ Explorer .....	2-50

Possible authentication failures . . . . .	2-51
Using secret key reset . . . . .	2-52
Refreshing TLS on IBM MQ . . . . .	2-53
Security administration tasks . . . . .	2-54
TLS implementation scenario (1 of 3) . . . . .	2-55
TLS implementation scenario (2 of 3) . . . . .	2-56
TLS implementation scenario (3 of 3) . . . . .	2-57
Security problems and solutions . . . . .	2-58
IBM MQ Advanced Message Security . . . . .	2-59
Unit summary . . . . .	2-60
Review questions . . . . .	2-61
Review answers . . . . .	2-62
Exercise: Securing channels with TLS . . . . .	2-63
Exercise objectives . . . . .	2-64
<b>Unit 3. Authenticating channels and connections . . . . .</b>	<b>3-1</b>
Unit objectives . . . . .	3-2
3.1. Authentication and authorization overview . . . . .	3-3
Authentication . . . . .	3-4
Authorization . . . . .	3-5
Identification and authentication in IBM MQ . . . . .	3-6
Approach to planning security . . . . .	3-7
Secure connections in IBM MQ Explorer . . . . .	3-8
Setting client connection default values . . . . .	3-9
Setting passwords that IBM MQ Explorer uses . . . . .	3-10
Security for remote queue manager connection . . . . .	3-11
Security settings for remote queue managers (1 of 4) . . . . .	3-12
Security settings for remote queue managers (2 of 4) . . . . .	3-13
Security settings for remote queue managers (3 of 4) . . . . .	3-14
Security settings for remote queue managers (4 of 4) . . . . .	3-15
User ID when using bindings mode . . . . .	3-16
Queue manager to queue manager . . . . .	3-17
Why using TLS can lead to a false sense of security . . . . .	3-18
IBM MQ Advanced Message Security . . . . .	3-19
3.2. Object authorizations review . . . . .	3-20
Object authority manager (OAM) . . . . .	3-21
Managing object authorities in IBM MQ Explorer (1 of 2) . . . . .	3-22
Managing object authorities in IBM MQ Explorer (2 of 2) . . . . .	3-23
Managing authority records in MQSC . . . . .	3-24
Application to queue manager . . . . .	3-25
Blank MCAUSER: Main cause of security exposures . . . . .	3-26
MCAUSER remedy . . . . .	3-27
Secure remote administration . . . . .	3-28
Authorizing groups and principals . . . . .	3-30
Interactive users with OAM . . . . .	3-31
Considerations for applications with OAM . . . . .	3-32
OAM and queue manager to queue manager . . . . .	3-33
A note about transmission queues . . . . .	3-34
PUTAUT: Put Authority for channels . . . . .	3-35
Security guidelines . . . . .	3-36
3.3. Connection authentication . . . . .	3-37
Connection authentication . . . . .	3-38
Enabling connection authentication on a queue manager . . . . .	3-39
Connection authentication configuration . . . . .	3-40
Connection authentication error notification . . . . .	3-42
Connection authentication and authorization . . . . .	3-43

Connection authentication user repositories . . . . .	3-45
LDAP user repository . . . . .	3-46
Secure connection to an LDAP server . . . . .	3-48
Connection authentication and authorization: LDAP . . . . .	3-50
User ID and password for IBM MQ Explorer (1 of 2) . . . . .	3-51
User IDs and passwords in IBM MQ programs . . . . .	3-53
Connection authentication summary . . . . .	3-54
3.4. Channel authentication . . . . .	3-55
Channel authentication rules . . . . .	3-56
Channel access blocking points . . . . .	3-57
Channel authentication example . . . . .	3-58
Precedence order . . . . .	3-59
Precedence order example . . . . .	3-60
Using IP addresses in channel authentication rules . . . . .	3-61
Using host names in channel authentication rules . . . . .	3-62
Getting the host name . . . . .	3-63
Connection authentication configuration granularity . . . . .	3-65
Diagnosing host name lookup failures . . . . .	3-66
Using MATCH(RUNCHECK) example . . . . .	3-67
More security options . . . . .	3-68
Dynamic mapping of MCAUSER . . . . .	3-69
Dynamic MCAUSER mapping example . . . . .	3-70
User ID blocking with CHLAUTH BLOCKUSER . . . . .	3-71
Filtering by IP address . . . . .	3-72
Filtering by IP address: Example . . . . .	3-73
Filtering by IP address: Guidelines . . . . .	3-74
Step-by-step guide to a basic secure setup . . . . .	3-75
Step 1: Set MCAUSER . . . . .	3-76
Step 2: Provision user IDs and groups . . . . .	3-77
Step 3: Set channels that are not SYSTEM channels . . . . .	3-78
Step 4: Authorize groups . . . . .	3-79
Step 5: Set up strong authentication . . . . .	3-80
3.5. Channel exits . . . . .	3-81
Channel exit programs . . . . .	3-82
Channel exits for link level security . . . . .	3-83
Channel auto-definition exit . . . . .	3-84
Channel exits on message channels . . . . .	3-85
Channel exit configuration . . . . .	3-86
Location of exit modules . . . . .	3-87
Client-side security exit . . . . .	3-88
Unit summary . . . . .	3-89
Review questions . . . . .	3-90
Review answers . . . . .	3-91
Exercise: Implementing connection authentication . . . . .	3-92
Exercise objectives . . . . .	3-93
<b>Unit 4. Implementing workload management in an IBM MQ cluster . . . . .</b>	<b>4-1</b>
Unit objectives . . . . .	4-2
4.1. Queue manager cluster concepts review . . . . .	4-3
What is a queue manager cluster? . . . . .	4-4
Benefits of clustering . . . . .	4-5
Overview of cluster components . . . . .	4-6
Considerations for a full repository . . . . .	4-8
Cluster channels . . . . .	4-9
Steps to set up a basic cluster . . . . .	4-10
Controlling clusters with MQSC commands . . . . .	4-11

Monitoring clusters with IBM MQ Explorer .....	4-13
Independent clusters .....	4-14
Overlapping clusters .....	4-15
Cluster gateways .....	4-16
Cluster troubleshooting .....	4-17
4.2. Workload management in queue manager clusters .....	4-18
Workload balancing in queue manager clusters .....	4-19
Cluster workload management options .....	4-20
When workload balancing can occur .....	4-21
Application binding overview .....	4-23
Application binding options .....	4-24
Setting default bind type for a queue .....	4-25
Cluster workload management attributes .....	4-26
Cluster workload management algorithm: Summary .....	4-27
Queue manager cluster scenario .....	4-28
Use-queue basics .....	4-29
Configuring CLWLUSEQ on a queue manager .....	4-30
Configuring CLWLUSEQ on a queue .....	4-31
Use-queue example .....	4-32
Rank basics .....	4-33
Configuring CLWLRANK .....	4-34
Channel rank example .....	4-35
Channel and queue rank example .....	4-36
Channel and queue priority basics .....	4-37
Using CLWLPRTY to identify primary and backup servers .....	4-38
Configuring channel and queue priority .....	4-39
Priority example (1 of 2) .....	4-40
Priority example (2 of 2) .....	4-41
Most recently used channels .....	4-42
Most recently used channels basics .....	4-43
Configuring CLWLMRUC on a queue manager .....	4-44
Using processing power to control the workload .....	4-45
Channel weight basics .....	4-46
Configuring CLWLWGHT .....	4-47
Channel weight example .....	4-48
Cluster workload management algorithm: Summary .....	4-49
Multiple cluster transmission queues .....	4-50
Automatic cluster transmission queues .....	4-51
Manual cluster transmission queues .....	4-52
Unit summary .....	4-53
Review questions .....	4-54
Review answers .....	4-55
Exercise: Implementing workload management in a cluster .....	4-56
Exercise objectives .....	4-57
<b>Unit 5. More troubleshooting tools and techniques .....</b>	<b>5-1</b>
Unit objectives .....	5-2
5.1. Event and activity monitoring .....	5-3
Instrumentation event monitoring .....	5-4
Examples of conditions that cause events .....	5-5
Event queues .....	5-6
Queue manager events .....	5-7
Controlling queue manager events .....	5-8
Controlling queue events .....	5-9
Configuration events .....	5-10
Configuration event examples .....	5-11

Logger events . . . . .	5-12
Logger event examples . . . . .	5-13
SupportPac MH05: WebSphere MQ Events Display Tool . . . . .	5-14
Sample output from MH05 SupportPac . . . . .	5-15
Application activity trace . . . . .	5-16
Tracing application activity . . . . .	5-17
Enabling activity trace in IBM MQ Explorer . . . . .	5-18
Choosing the application activity trace level . . . . .	5-19
Subscribing to activity trace data . . . . .	5-20
Example activity trace topic strings . . . . .	5-21
Formatting application activity trace output . . . . .	5-22
Activity trace example . . . . .	5-23
<b>5.2. Message monitoring . . . . .</b>	<b>5-24</b>
Where is the message? . . . . .	5-25
Message monitoring . . . . .	5-26
Recording message activity . . . . .	5-27
Activate message activity recording . . . . .	5-29
Trace-route messaging . . . . .	5-30
Enabling queue manager for trace-route messaging . . . . .	5-31
Display route application (dspmqrt) . . . . .	5-32
Display route application parameters . . . . .	5-34
dspmqrt sample output (1 of 2) . . . . .	5-35
dspmqrt sample output (2 of 2) . . . . .	5-36
Hints and tips for using trace route . . . . .	5-37
Comparing activity recording and trace route . . . . .	5-38
Exercise: Tracing message routes . . . . .	5-39
Exercise objectives . . . . .	5-40
Dead-letter queues . . . . .	5-41
Using dead-letter queues . . . . .	5-42
IBM MQ dead-letter queue handler . . . . .	5-43
Dead-letter queue handler sample . . . . .	5-44
Using a dead-letter queue handler . . . . .	5-45
<b>5.3. Disaster recovery . . . . .</b>	<b>5-46</b>
What is disaster recovery? . . . . .	5-47
What to do if a queue manager stops unexpectedly . . . . .	5-48
What is the point of logging? . . . . .	5-49
DO, REDO, and UNDO operations . . . . .	5-51
Phases of restart recovery . . . . .	5-53
First manual restart of the queue manager . . . . .	5-55
Cold start of IBM MQ . . . . .	5-56
Cold start considerations . . . . .	5-57
Cold start procedure (1 of 2) . . . . .	5-58
Cold start procedure (2 of 2) . . . . .	5-59
Rebuilding a queue manager . . . . .	5-60
Copying or moving contents of a queue to a file . . . . .	5-61
Using the dmpmqmsg utility . . . . .	5-62
Planning for recovery . . . . .	5-63
Unit summary . . . . .	5-64
Review questions . . . . .	5-65
Review answers . . . . .	5-66
Exercise: Handling messages on the dead-letter queue . . . . .	5-67
Exercise objectives . . . . .	5-68
<b>Unit 6. High availability . . . . .</b>	<b>6-1</b>
Unit objectives . . . . .	6-2
<b>6.1. Planning for high availability . . . . .</b>	<b>6-3</b>

High availability objectives .....	6-4
Potential outage types .....	6-5
Some terms .....	6-6
Single points of failure .....	6-7
Failover strategies .....	6-8
IBM MQ high availability technologies .....	6-9
IBM MQ clusters as a high availability strategy .....	6-10
IBM MQ clustering high availability considerations .....	6-11
Support for networked storage .....	6-12
Client configuration for availability .....	6-13
Automatic client reconnection .....	6-14
Failover and IBM MQ .....	6-15
Failover considerations .....	6-16
<b>6.2. HA clusters .....</b>	<b>6-17</b>
IBM MQ in HA clusters .....	6-18
Standby failover .....	6-19
Standby failover complete .....	6-20
Active-active configuration .....	6-21
Active-active takeover .....	6-22
Benefits of using IBM MQ with HA clusters .....	6-23
Implementing HA clusters on UNIX .....	6-24
Configuring the shared disk .....	6-25
Creating the queue manager for the HA cluster .....	6-26
Adding queue manager to other nodes in HA cluster .....	6-27
Starting the queue manager under HA control .....	6-28
IBM PowerHA cluster .....	6-29
Microsoft Cluster Service (MSCS) .....	6-30
<b>6.3. Multi-instance queue managers .....</b>	<b>6-31</b>
Multi-instance queue managers .....	6-32
Creating multi-instance queue managers .....	6-33
Standby configuration .....	6-34
Standby failover .....	6-35
Handling multiple IP addresses in multi-instance .....	6-36
Multi-instance queue manager administration .....	6-37
Managing multi-instance queue manager in IBM MQ Explorer .....	6-38
Comparing multi-instance queue managers and HA clusters .....	6-39
Unit summary .....	6-40
Review questions .....	6-41
Review answers .....	6-42
<b>Unit 7. Introduction to distributed publish/subscribe.....</b>	<b>7-1</b>
Unit objectives .....	7-2
<b>7.1. Publish/subscribe overview .....</b>	<b>7-3</b>
What is publish/subscribe? .....	7-4
Publish/subscribe example .....	7-5
Publish/subscribe patterns .....	7-6
Distributed publish/subscribe topologies .....	7-7
Publications, subscriptions, and topics .....	7-8
Topic strings and topic tree .....	7-9
Matching publications to subscriptions .....	7-10
Summary of IBM MQ publish/subscribe .....	7-11
IBM MQ publish/subscribe terminology .....	7-12
Publish/subscribe in IBM MQ .....	7-13
Publish/subscribe queue manager properties .....	7-14
<b>7.2. Managing topics .....</b>	<b>7-16</b>
Defining the topic tree .....	7-17

Why worry about the size and shape of the topic tree?	7-18
Defining a topic object with MQSC	7-20
Topic object attributes	7-21
Defining a topic object in IBM MQ Explorer (1 of 2)	7-22
Defining a topic object in IBM MQ Explorer (2 of 2)	7-23
Managing topic object definitions	7-24
Getting topic status in IBM MQ Explorer	7-25
Getting topic status with MQSC	7-26
Display topic status example	7-27
Publisher topic status in IBM MQ Explorer	7-28
Finding publishers (1 of 2)	7-29
Finding publishers (2 of 2)	7-30
Retained publications	7-31
<b>7.3. Managing subscriptions</b>	<b>7-32</b>
Subscription management	7-33
Creating administrative subscriptions in IBM MQ Explorer	7-34
Subscription lifetime	7-35
Resolving ASPARENT for durable subscriptions	7-36
Managing subscription queues	7-37
Monitoring application subscriptions in IBM MQ Explorer	7-38
Monitoring subscriptions with MQSC (1 of 2)	7-39
Monitoring subscriptions with MQSC (2 of 2)	7-40
Display subscription example	7-41
Display subscription status example	7-42
Display connection status for the subscription	7-43
Testing publish/subscribe in IBM MQ Explorer (1 of 2)	7-44
Testing publish/subscribe in IBM MQ Explorer (2 of 2)	7-45
<b>7.4. Security overview</b>	<b>7-46</b>
Publish/subscribe security	7-47
Publish/subscribe security relationships	7-48
Topic security	7-49
Controlling topic security with IBM MQ Explorer (1 of 3)	7-50
Controlling topic security with IBM MQ Explorer (2 of 3)	7-51
Controlling topic security with IBM MQ Explorer (3 of 3)	7-52
Controlling security topic with setmqaut	7-53
Basic IBM MQ security on publish/subscribe queues	7-54
<b>7.5. Publish/subscribe topologies</b>	<b>7-55</b>
Publish/subscribe hierarchies	7-56
Publish/subscribe hierarchy configuration	7-57
Display publish/subscribe hierarchy status	7-58
Publish/subscribe clusters	7-59
Cluster topics	7-61
Cluster topic example	7-63
Display cluster topic example	7-64
Cluster modes for routing publications	7-65
Cluster topic routing	7-66
Routing behavior for publish/subscribe clusters	7-68
Topic host routing requirements	7-69
Subscription propagation	7-70
Viewing proxy subscriptions	7-71
<b>7.6. Comparing topologies</b>	<b>7-72</b>
Proxy subscription in a hierarchy	7-73
Proxy subscriptions in a cluster	7-74
Scaling	7-75
Publication flows	7-76
Configuration	7-77

Availability . . . . .	7-78
Comparison summary . . . . .	7-79
<b>7.7. Troubleshooting . . . . .</b>	<b>7-80</b>
Tips . . . . .	7-81
Problem determination (1 of 2) . . . . .	7-82
Problem determination (2 of 2) . . . . .	7-83
Loop detection . . . . .	7-85
Stopping a queue manager . . . . .	7-86
Unit summary . . . . .	7-87
Review questions . . . . .	7-88
Review answers . . . . .	7-89
Exercise: Configuring distributed publish/subscribe . . . . .	7-90
Exercise objectives . . . . .	7-91
<b>Unit 8. Supporting JMS with IBM MQ . . . . .</b>	<b>8-1</b>
Unit objectives . . . . .	8-2
<b>8.1. IBM MQ and JMS . . . . .</b>	<b>8-3</b>
What is JMS? . . . . .	8-4
JMS elements . . . . .	8-5
JMS administered objects . . . . .	8-6
JMS providers . . . . .	8-7
IBM MQ classes for Java or JMS . . . . .	8-8
IBM MQ classes for JMS and IBM MQ classes for Java . . . . .	8-9
JMS message properties . . . . .	8-10
Support for JMS 2.0 in IBM MQ . . . . .	8-11
<b>8.2. Managing JMS resources with IBM MQ Explorer . . . . .</b>	<b>8-12</b>
Configuring JMS administered objects in IBM MQ . . . . .	8-13
IBM MQ Explorer and JMS . . . . .	8-14
Adding an initial context . . . . .	8-15
Connecting the initial context . . . . .	8-16
Creating a connection factory (1 of 3) . . . . .	8-17
Creating a connection factory (2 of 3) . . . . .	8-18
Creating a connection factory (3 of 3) . . . . .	8-19
Creating a JMS destination (1 of 3) . . . . .	8-20
Creating a destination (2 of 3) . . . . .	8-21
Creating a destination (3 of 3) . . . . .	8-22
Mapping between IBM MQ and JMS objects (1 of 2) . . . . .	8-23
Mapping between IBM MQ and JMS objects (2 of 2) . . . . .	8-24
JMS object creation wizards . . . . .	8-25
IBM MQ classes for JMS problem determination . . . . .	8-26
Unit summary . . . . .	8-27
Review questions . . . . .	8-28
Review answers . . . . .	8-29
<b>Unit 9. Introduction to the IBM MQ Console . . . . .</b>	<b>9-1</b>
Unit objectives . . . . .	9-2
<b>9.1. IBM MQ Console overview . . . . .</b>	<b>9-3</b>
The IBM MQ Console . . . . .	9-4
IBM MQ Console dashboard . . . . .	9-5
Advantages of the IBM MQ Console . . . . .	9-6
Administration with the IBM MQ Console . . . . .	9-7
Dashboard components . . . . .	9-8
IBM MQ Console security . . . . .	9-9
Managing queue managers . . . . .	9-10
Queue manager status . . . . .	9-11
Monitoring resources . . . . .	9-12

IBM MQ Console and the administrative REST API .....	9-13
9.2. IBM MQ Console setup .....	9-14
Getting started .....	9-15
Configuring users and roles .....	9-16
Sample user registry files .....	9-17
IBM MQ Console roles .....	9-18
Sample basic_registry.xml file (1 of 2) .....	9-19
Sample basic_registry.xml file (2 of 2) .....	9-20
Commands to control IBM MQ Console web server .....	9-21
Working with queue managers .....	9-22
Adding a widget .....	9-23
Configuring the dashboard layout .....	9-24
Monitoring system resource usage .....	9-25
More information on the IBM MQ Console .....	9-26
Unit summary .....	9-27
Review questions .....	9-28
Review answers .....	9-29
Exercise: Getting started with the IBM MQ Console .....	9-30
Exercise objectives .....	9-31
<b>Unit 10. Course summary .....</b>	<b>10-1</b>
Unit objectives .....	10-2
Course objectives .....	10-3
To learn more on the subject .....	10-4
Enhance your learning with IBM resources .....	10-5
Unit summary .....	10-6
Course completion .....	10-7
<b>Appendix A. List of abbreviations .....</b>	<b>A-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®  
First Failure Support  
Technology™  
WebSphere®

Approach®  
Notes®  
z/OS®

FFST™  
PowerHA®

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.

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

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

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

# Course description

## IBM MQ V9 Advanced System Administration (Distributed)

**Duration: 4 days**

### Purpose

This course expands the basic skill sets that are developed in the IBM courses WM103/ZM103, *Technical Introduction to IBM MQ*, and WM153/ZM153, *IBM MQ V9 System Administration (using Windows for labs)* or WM154, *IBM MQ V9 System Administration (using Linux for labs)*.

The course focuses on advanced features of IBM MQ, such as implementing workload management by using a queue manager cluster, and authenticating connections, channels, and users. It also covers securing channels with Transport Layer Security (TLS), advanced client connection features, event and message monitoring, and publish/subscribe administration.

In addition to the instructor-led lectures, you participate in hands-on lab exercises that reinforce lecture content. The lab exercises give you practical experience with tasks such as implementing security, configuring workload management for a queue manager cluster, and advanced troubleshooting techniques.

Completing this course can also help you prepare for the appropriate IBM MQ Administrator certifications.

### Audience

This advanced skills course is designed for technical professionals who require advanced administrator skills for IBM MQ on distributed operating systems, or who provide support to others who administer IBM MQ.

### Prerequisites

Before taking this course, you should possess the skills that are required to complete basic IBM MQ system administration tasks in a distributed environment. You can obtain these skills through practical experience or by successfully completing one of the IBM MQ V9 system administration courses for distributed operating systems:

- *IBM MQ V9 System Administration (using Windows for labs)* (WM153G)
- *IBM MQ V9 System Administration (using Windows for labs)* (ZM153G)
- *IBM MQ V9 System Administration (using Linux for labs)* (WM154G)

### Objectives

- Use conversation sharing, read-ahead, and asynchronous put to improve the performance of MQI client connections

- Use Transport Layer Security (TLS) to secure TCP/IP channels
- Authenticate IBM MQ channels, connections, and users
- Manage the workload in an IBM MQ queue manager cluster
- Implement IBM MQ high availability
- Monitor application activity, events, and messages
- Use the IBM MQ dead-letter queue message handler to manage a dead-letter queue
- Administer distributed publish/subscribe networks
- Use the IBM MQ Console to administer IBM MQ objects and resource usage
- Administer Java Message Service (JMS) in MQ

## Contents

- Implementing security
- Workload management

## Curriculum relationship

This course is a follow-on course to WM153, ZM153, or WM154, *IBM MQ V9 System Administration*.

---

# Agenda

---

**Note**

The following unit and exercise durations are estimates, and might not reflect every class experience.

---

## Day 1

- (00:30) Course introduction
- (01:30) Unit 1. Managing clients and client connections
- (01:30) Unit 2. Securing IBM MQ channels with TLS
- (01:30) Exercise 1. Securing channels with TLS

## Day 2

- (02:00) Unit 3. Authenticating channels and connections
- (01:30) Exercise 2. Implementing connection authentication
- (01:00) Unit 4. Implementing workload management in an IBM MQ cluster
- (01:00) Exercise 3. Implementing workload management in a cluster

## Day 3

- (02:00) Unit 5. More troubleshooting tools and techniques
- (01:00) Exercise 4. Tracing message routes
- (01:00) Exercise 5. Handling messages on the dead-letter queue
- (01:00) Unit 6. High availability

## Day 4

- (01:30) Unit 7. Introduction to distributed publish/subscribe
- (01:30) Exercise 6. Configuring distributed publish/subscribe
- (00:30) Unit 8. Supporting JMS with IBM MQ
- (01:00) Unit 9. Introduction to the IBM MQ Console
- (01:00) Exercise 7. Getting started with the IBM MQ Console
- (00:30) Unit 10. Course summary

---

# Unit 1. Managing clients and client connections

## Estimated time

01:30

## Overview

In this unit, you learn about the ways that IBM MQ clients can attach to an IBM MQ server. You also learn about channel performance and monitoring channel activity.

## How you will check your progress

- Review questions

## References

IBM Knowledge Center for IBM MQ V9

## How to check online for course material updates



**Note:** If your classroom does not have internet access, ask your instructor for more information.

### Instructions

1. Enter this URL in your browser:  
[ibm.biz/CloudEduCourses](http://ibm.biz/CloudEduCourses)
2. Find the product category for your course, and click the link to view all products and courses.
3. Find your course in the course list and then click the link.
4. The wiki page displays information for the course. If a course corrections document is available, it is found here.
5. If you want to download an attachment, such as a course corrections document, click the **Attachments** tab at the bottom of the page.
6. To save the file to your computer, click the document link and follow the prompts.

[Comments \(0\)](#) [Versions \(1\)](#) **Attachments (1)** [About](#)

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-1. How to check online for course material updates

## Unit objectives

- Manage client performance by sharing conversations, using read ahead, and using asynchronous put
- Describe the client modes that MQSC supports
- Use troubleshooting tools and techniques to monitor and manage clients and connections

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-2. Unit objectives

IBM MQ supports two types of channels: message channels and MQI channels.

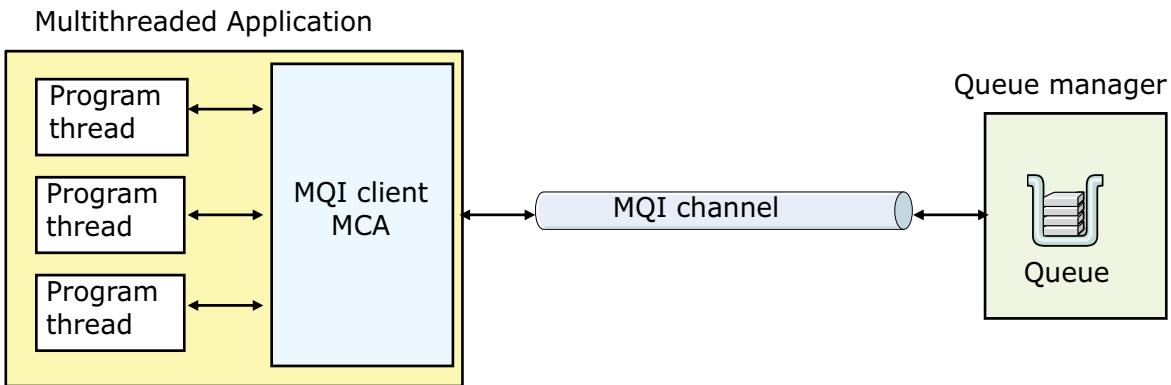
- A message channel provides a one-way communications link between two queue managers.
- An MQI channel provides a two-way communications link between an application and a queue manager.

MQI applications and IBM MQ can implement three main options to provide performance scalability for MQI channels: sharing conversations, read ahead, and asynchronous put. Each of these options is described in detail in this unit.

## 1.1. Sharing conversations

## Sharing conversations overview

- MQI channels
  - Bidirectional communications link between an MQI client application and a queue manager
  - Used to transfer MQI calls and responses between MQI client applications and queue managers
- A single MQI channel can support multiple threads



[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-3. Sharing conversations overview

The two-way channels and multiplexing support in IBM MQ allows an application to have multiple threads that share a client-channel connection.

In this context, an application means a single process. An example would be a Java or C program that runs multiple threads, with each thread that is processing a message. Separate processes, for example, two separate Java or C programs, cannot share a client channel.

## Sharing conversation requirements

- Both ends of the client/server connection must be configured for sharing conversations
- Client channel **Sharing conversation** value must match the value that is supplied on the client MQCONN call or the value in the client channel definition table (CCDT)
- Sharing conversations limit on the server side is not exceeded

Managing clients and client connections

© Copyright IBM Corporation 2017

*Figure 1-4. Sharing conversation requirements*

Sharing conversations requires configuration at the client-connection and server-connection ends of the channel. This figure summarizes the requirements for sharing conversations.

The MQCD structure contains the parameters that control execution of a channel. It is passed to each channel exit that is called from an MCA.

## SHARECNV channel property

- Can be configured on server-connection (SVRCONN) and client-connection (CLNTCONN) channels
- Applications can set **SHARECNV** on the MQCONN call or by setting the **sharingConversations** field of the MQEnvironment and MQChannelDefinition classes in Java
- If you do not need to share conversations:
  - Set **SHARECNV** to 1 to eliminate contention and use features such as bidirectional heartbeats and performance improvements added in IBM MQ V8
  - If you have existing applications that do not run correctly when you set **SHARECNV** to 1 or greater, set **SHARECNV** to 0

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-5. SHARECNV channel property

The IBM MQ sharing conversations option is configured on the client-connection channel. The client-connection can be configured dynamically by using the MQSERVER environment variable. It can also be configured by the application in the MQCONN call, by using the IBM MQ Java and JMS classes, or by using a pre-connect exit.

You use the SHARECNV channel property to specify the maximum number of conversations that can be shared over a particular TCP/IP client channel instance.

By default, the SHARECNV value for a new queue manager is set to 10 in any SRVCONN channel definitions. The SYSTEM.DEF.SRVCONN channel definition, which is used to supply default values when a new SRVCONN channel is defined, also has SHARECNV set to 10. You can change this default if needed.

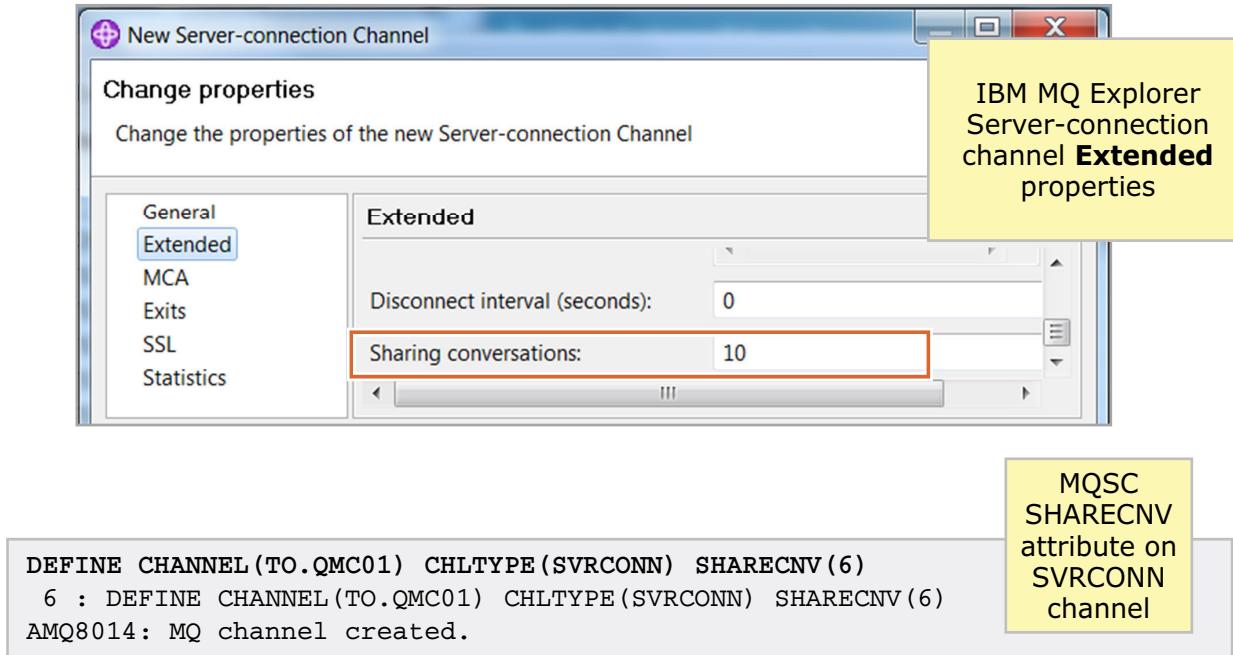
You set the SHARECNV property to a numeric value.

- SHARECNV(0). This value specifies no sharing of conversations over a TCP/IP socket. Use a value of 0 if you have WebSphere MQ V6 client applications.
- SHARECNV(1). This value allows one conversation on the channel instance. Client heartbeats and read ahead are available, and channel quiescing is more controllable. SHARECNV(1) is the suggested value for IBM MQ V9 if you do not need to share conversations.

- SHARECNV(2) to SHARECNV(999999999). Each of these values specifies the number of shared conversations. If the client-connection SHARECNV value does not match the server-connection SHARECNV value, then the lowest value is used. The default value is SHARECNV(10), which specifies 10 threads to run up to 10 client conversations per channel instance. However, distributed servers might exhibit performance problems with SHARECNV channels that can be eased by setting SHARECNV(1) wherever possible.



## Sharing conversations configuration: Server-connection



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-6. Sharing conversations configuration: Server-connection

You can use MQSC commands and IBM MQ Explorer to configure sharing conversations on the server-connection (SVRCONN) channel.

In IBM MQ Explorer, the **Sharing conversations** property is on the **Extended** properties page.



## Sharing conversations configuration: Client-connection

**IBM MQ Explorer Client-connection channel General properties**

Sharing conversations: 10

```
DEFINE CHANNEL (TO.QMC01) CHLTYPE(CLNTCONN) SHARECNV(4) CONNAME(10.1.2.3)
8 : DEFINE CHANNEL (TO.QMC01) CHLTYPE(CLNTCONN) SHARECNV(4)
CONNAME(10.1.2.3)
AMQ8014: MQ channel created.
```

**MQSC SHARECNV attribute on CLNTCONN channel**

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-7. Sharing conversations configuration: Client-connection

You can use MQSC commands `DEFINE CHANNEL` and `ALTER CHANNEL` or IBM MQ Explorer to configure sharing conversations on the client-connection (SVRCONN) channel.

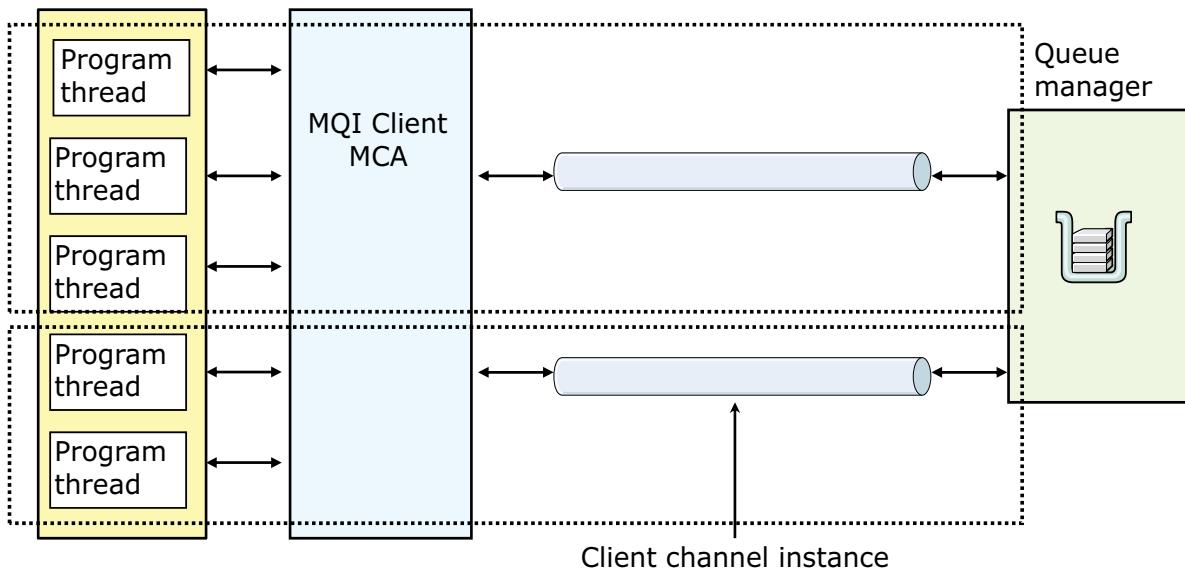
In IBM MQ Explorer, the **Sharing conversations** property is on the **General** properties page for client-connection channels.

## Sharing conversations example

```
DEFINE CHANNEL (SALES)
  TYPE (CLNTCONN)  SHARECNV (3)
```

```
DEFINE CHANNEL (SALES)
  TYPE (SVRCONN)  SHARECNV (5)
```

Multithreaded Application



[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

*Figure 1-8. Sharing conversations example*

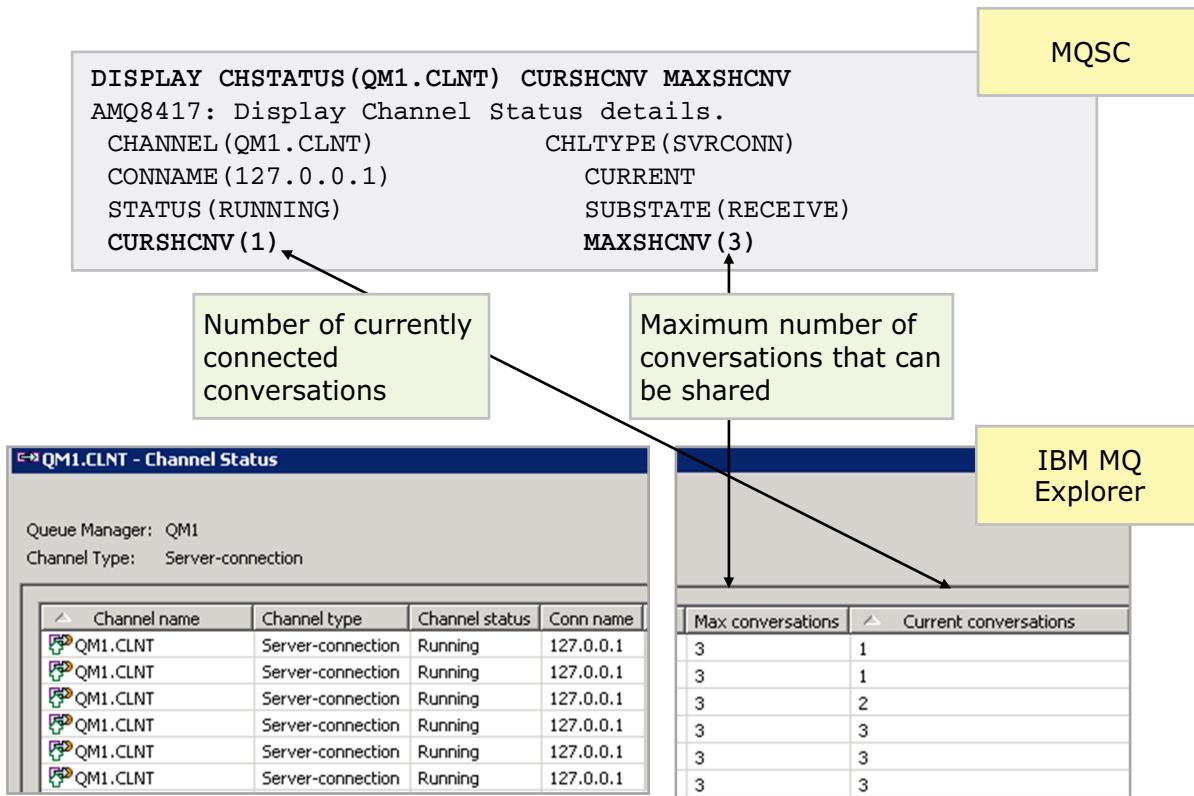
The example shows that the client-channel definition SHARECNV attribute is set to 3. The corresponding server-connection channel definition SHARECNV attribute is set to 5.

Because the channel SHARECNV values are different, the lowest value applies. So in this example, the maximum number of threads that can share a single client channel is 3. The figure shows three threads that share one client-channel instance, and another two threads in the same application that share a different client-channel instance.

You cannot specify the distribution of the number of threads evenly over client channels.



## Sharing conversations channel status fields



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-9. Sharing conversations channel status fields

Fields that are associated with sharing conversations are shown when you view channel status on server-connection type channels for a queue manager. The sharing conversations values can be viewed by using the MQSC DISPLAY CHSTATUS command or the IBM MQ Explorer **Channel Status** view.

- The **Current conversations** (CURSHCNV) property specifies the number of conversations that are currently connected to the queue manager on the channel instance.
- The **Max conversations** (MAXSHCNV) property specifies the maximum number of conversations that can share a channel instance. This value is the lowest value of the SHARECNV attribute on the associated client-connection channel and server-connection channel.

A value of 0 for CURSHCNV and MAXSHCNV indicates that sharing conversations is not enabled for this channel.

## Sharing conversations and existing channel status fields

```
DISPLAY CHSTATUS (QM1.CLNT) ALL
AMQ8417: Display Channel Status details.
CHANNEL (QM1.CLNT)          CHLTYPE (SVRCONN)
BUFSRCVD (2)                 BUFSSENT (2)
BYTSRCVD (544)                BYTSSENT (536)
CHSTADA (2017-05-22)          CHSTATI (06.31.42)
COMPHDR (NONE,NONE)           COMPMMSG (NONE,NONE)
COMP RATE (0,0)                 COMPTIME (0,0)
CONNNAME (127.0.0.0)            CURRENT
EXITTIME (0,0)                  HBINIT (300)
JOBNAME (0000A2800001498)      LOCLADDR ( )
LSTMSGDA (2017-05-22)          LSTMSGTI (02.43.51)
MCASTAT (RUNNING)              MCAUSER (MUSER_ADMIN)
MONCHL (OFF)                   MSGS (2)
RAPPLTAG ( )                   SSLCERTI ( )
SSLKEYDA ( )                   SSLKEYTI ( )
SSLPEER ( )                    SSLKEYS (0)
STATUS (RUNNING)                STOPREQ (NO)
SUBSTATE (RECEIVE)              CURSHCNV (1)
MAXSHCNV (3)                   RVERSION (080000000)
RPRODUCT (MQJB)
```

LSTMSGDA and LSTMSGTI are the date and time of the most recent MQI call that any thread that is sharing the channel instance makes

If threads have a different user ID, then \* is shown for MCA user ID

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-10. Sharing conversations and existing channel status fields

Sharing conversations affects the display of server-connection (SVRCONN) type channels in a queue manager.

The **MCA user ID** (MCAUSER) is the effective user ID of the channel instance. If multiple program threads all have the same user ID, the value is shown. If any threads have a different user ID, an asterisk (\*) is shown.

**Last message date** (LSTMSGDA) and **Last message time** (LSTMSGTI) indicate the date and time of the most recent MQI call that any thread that is sharing the channel instance makes.

## Sharing conversations and channel exit considerations

- Consider channel exit behavior if conversation sharing is used on client channels
- An exit sees the information flows for many independent client threads and might need to serialize access to the MQCD data type
- Review existing exits on client-connection channels and server-connection channels for their possible impact on their correct operation

Managing clients and client connections

© Copyright IBM Corporation 2017

*Figure 1-11. Sharing conversations and channel exit considerations*

By using a pre-connect exit, MQI clients can be configured to retrieve client channel definitions from an external repository, such as LDAP and WebSphere Services Registry and Repository. The details of the exit library and function to call are specified in the `mqclient.ini` configuration file.

## Sharing conversation performance

- Consider the possible impact on applications when multiple threads share a single client channel
 

Example:  
An MQOPEN call in one thread might take longer than expected to run because many other threads are calling MQPUT and MQGET to process messages.  
The tradeoff is that the queue manager is managing fewer client channels
- Set **SHARECNV** to 1 on the server connection channels to eliminate contention and improve performance



For consistency with previous releases of IBM MQ, the default **SHARECNV** value on a server-connection is 10.

*Figure 1-12. Sharing conversation performance*

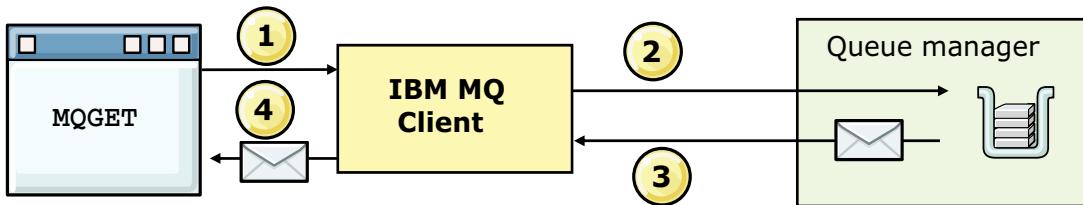
The advantage of sharing conversations is that fewer active client-channel definitions are needed to support the environment and fewer resources are required, such as less memory and processor power.

You must consider the possible impact of sharing conversations on multithreaded applications. Some threads can be delayed when a multithreaded application issues an IBM MQ call because of the activity of other threads. If you know that threads in an application are each going to be making high use of a client channel, then do not share conversations. Instead, set the SHARECNV attribute so that each thread gets its own channel. If you know that each thread uses client channels relatively infrequently, then sharing conversations is appropriate.

## 1.2. Read ahead

## Standard MQGET processing

1. Application sends an MQGET call
2. MQ client requests one message from the queue on the queue manager
3. Queue manager sends one message to MQ client
4. MQ client passes the message to the application
5. Loop back to Step 1 and repeat until all of the messages are read



- Disadvantage:
  - Application client must wait for each MQGET operation to complete before it can process messages
  - Wait time can be long, depending on the network

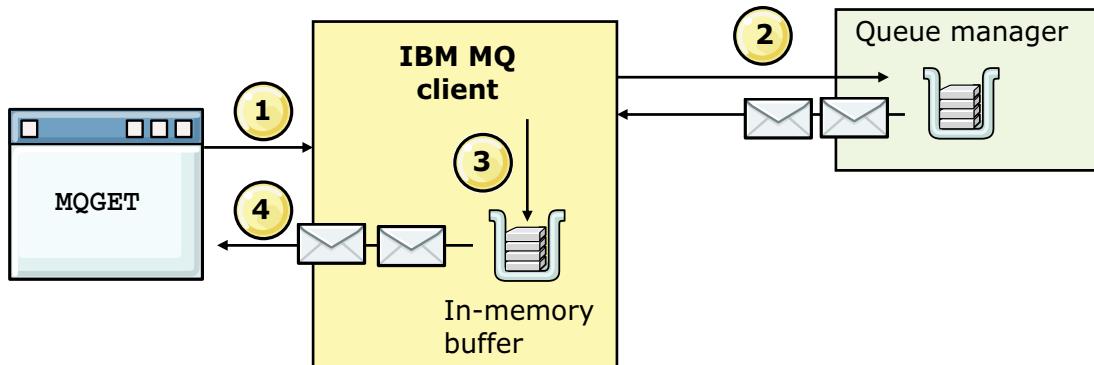
Figure 1-13. Standard MQGET processing

With standard MQGET processing, the IBM MQ client gets a message from the queue manager when the application sends an MQGET call and waits for a response from the queue manager before it sends another MQGET. The figure shows standard MQGET processing.

The disadvantage of standard MQGET processing is that the application client must wait for each MQGET operation to complete before it can process a message, which causes delays in message processing.

## Read ahead overview

1. Application sends an MQGET call
2. Client reads messages from the queue on the queue manager
3. Client stores the messages in an in-memory buffer
4. Client passes the messages to the application from the in-memory buffer



[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-14. Read ahead overview

The figure shows the read ahead feature in IBM MQ.

When an application sends the first MQGET call, the IBM MQ client can independently get several messages from the queue manager and store them in an in-memory buffer. When the application issues subsequent MQGET calls, it does not have to wait for the IBM MQ client to get the message from the queue manager. Instead, because the messages are stored in a memory buffer, the IBM MQ client can immediately pass the next message to the application in response to an MQGET call.

The message buffer is an 'in memory' queue of messages. If the application ends or the server fails, these messages are lost.

Because this mechanism is designed to remove the network delay, it currently only benefits client applications.

The read ahead feature can also be used for browsing.

## Read ahead advantages

- While the application is processing a message, the IBM MQ client is proactively:
  - Reading ahead
  - Getting messages from the queue on the queue manager
  - Storing the messages in an in-memory buffer
- When application sends an MQGET for the next message, the IBM MQ client can pass the message immediately from the in-memory buffer

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-15. Read ahead advantages

Read ahead is useful for applications that get large numbers of non-persistent messages, outside of syncpoint where they are not changing the selection criteria regularly. For example, applications that get responses from a command server or a query, such as a list of airline flights.

A large proportion of the cost of an MQGET from a client is the turnaround time of the network connection. When IBM MQ uses read ahead (also referred to as *streaming*), the IBM MQ client makes a request for more than one message from the server. The server sends as many non-persistent messages that match the criteria as it can up to the limit set by the client. The largest speed benefit is seen where the application processes many similar non-persistent messages and where the network is slow.

If an application requests read ahead but the messages are not suitable, for example, they are all persistent, then only one message is sent to the client at any one time. Read ahead is effectively turned off until a sequence of non-persistent messages are on the queue again.

This performance improvement is available to both MQI and JMS applications.

## Read ahead considerations

- Read ahead is supported for non-persistent messages only
- Messages that are streamed into memory on the client system are destructively removed from the queue manager
- If the client reads a persistent message, no further messages are read from the queue until the application reads the persistent message
- If the queue is closed, application MQCLOSE option determines what happens to any messages that are stored in the read ahead buffer
- Messages are lost if:
  - Client program ends
  - Client system fails before the application gets all the messages

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

*Figure 1-16. Read ahead considerations*

Not all client application designs are suited to using read ahead, so read ahead is not enabled by default. For example, read is not supported for persistent messages. Do not use read ahead when the application processes persistent messages.

The administrator can enable read ahead at the queue or application level.

When an application calls MQOPEN with MQOO\_READ\_AHEAD, the IBM MQ client enables read ahead if certain conditions are met.

- Both the client and remote queue manager must be at WebSphere MQ Version 7 or higher.
- The client application must be compiled and linked against the threaded MQI client libraries.
- The client channel must be using TCP/IP protocol.

The two MQCLOSE options allow applications to configure what happens to any messages that are being stored in the read ahead buffer if the queue is closed. The application can specify to discard messages in the read ahead buffer or to get all of the messages before the queue is closed.

## Default read ahead configuration (1 of 2)

1. Configure read ahead at the queue level
  - To enable read ahead, set the **Default read ahead (DEFREADA)** queue attribute to **YES**
  - To use read ahead only when a client requests it, set the **Default read ahead (DEFREADA)** queue attribute to **NO**
2. Configure read ahead at the application level option on the MQOPEN call
  - To use read ahead whenever possible, set to **MQOO\_READ\_AHEAD**
  - If the **Default read ahead (DEFREADA)** queue attribute determines read ahead behavior, set to **MQOO\_READ\_AHEAD\_AS\_Q\_DEF**
3. Ensure that the channel has a nonzero value for the **Sharing conversations (SHARECNV)** setting on both the client and server channel definitions

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-17. Default read ahead configuration (1 of 2)

Whether an application uses the read ahead feature is due to a combination of the settings that are used with the MQOPEN call from the application and the values that are set in the **Default read ahead (DEFREADA)** attribute on the queue.

The figure lists the steps for enabling read ahead.

1. To configure read ahead at the queue level, set the queue attribute **Default read ahead (DEFREADA)** to **YES**.
2. To use read ahead at the application level, implement one of the following options:
  - Use the **MQOO\_READ\_AHEAD** option on the MQOPEN function call to use read ahead wherever possible. It is not possible for the client application to use read ahead if the DEFREADA queue attribute is set to DISABLED.
  - Use the **MQOO\_READ\_AHEAD\_AS\_Q\_DEF** option on the MQOPEN function call to use read ahead only when read ahead is enabled on a queue.
3. The channel must have a nonzero **Sharing conversations (SHARECNV)** values in both the client and server channel definitions.



`ALTER QLOCAL(QL.A) DEFREADA(NO)`

MQSC DEFREADA  
attribute on local  
queue

Figure 1-18. Read ahead configuration (2 of 2)

The **Default read ahead** (DEFREADA) parameter on a queue can be set by using the IBM MQ Explorer or the MQSC ALTER QLOCAL command. The valid values are NO, YES, and DISABLED.

- NO indicates that read ahead is not enabled. Setting DEFREADA to NO is equivalent to the behavior in IBM MQ versions before WebSphere MQ V7.0 in which messages are transported from the queue manager at the time they are requested.
- YES indicates that read ahead is enabled.
- DISABLED indicates that read ahead is disabled, even if an application that opens the queue specifies the MQOO\_READ\_AHEAD option.

## Read ahead memory buffer size

- When the read ahead feature is in use:
  - Client receives messages ahead of an application that requests them, storing them in a buffer in memory on the client
  - Default behavior is that the client manages the size of the buffer
- Buffer size can be altered by adjusting the **MaximumSize** attribute of the **MessageBuffer** stanza on the client configuration file
  - Set to a value 1 - 999999
  - When **MaximumSize = -1**, the client determines the appropriate value
  - When **MaximumSize = 0**, read ahead is disabled for the client

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-19. Read ahead memory buffer size

The read ahead memory buffer attributes are configurable.

Use the **MessageBuffer** stanza of the client configuration file to specify information about message buffers.

The **MaximumSize** attribute is an integer value that indicates the size of the read-ahead buffer, in the range of 1 KB through 999999 KB.

For more information about updating the client configuration file, see the IBM Knowledge Center for IBM MQ.

## Read ahead buffer updates

- Default behavior is that the client keeps the memory buffer filled with messages so that the client can pass a message immediately to the application when it sends an **MQGET**
- **UpdatePercentage** and **MaximumSize** attributes in the client configuration determine when to get more messages

Example:

**MaximumSize** = 100 Kb and **UpdatePercentage** = 20

$$\text{Threshold} = (100 - 20) = 80 \text{ Kb}$$

When **MQGET** calls remove 80 Kb from a queue, the client makes a new request automatically

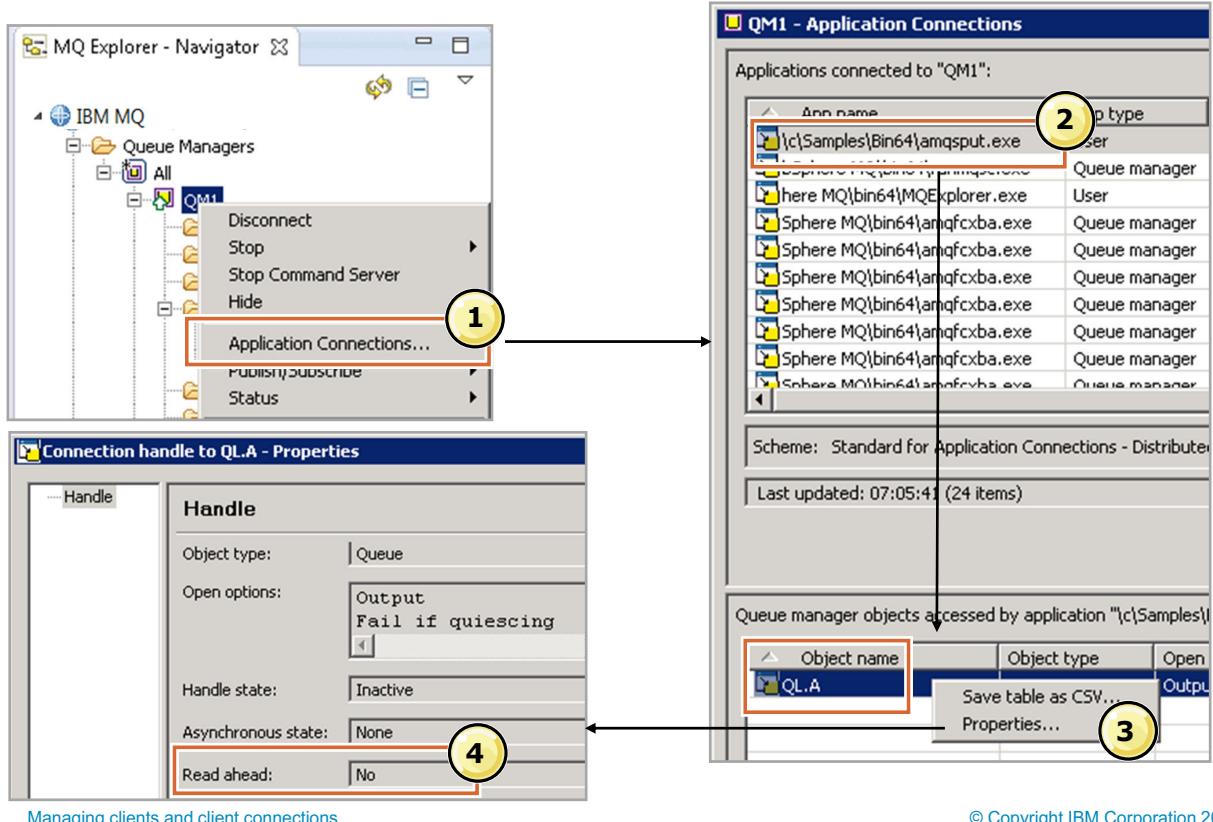
Figure 1-20. Read ahead buffer updates

The read ahead **UpdatePercentage** and **MaximumSize** attributes determine when client gets more messages.

The **UpdatePercentage** value, in the range of 1 - 100, is used in calculating the threshold value to determine when a client application makes a new request to the server. The special value of “-1” indicates that the client determines the appropriate value.



## Using IBM MQ Explorer to get read ahead status



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-21. Using IBM MQ Explorer to get read ahead status

The status of read ahead on connected applications can be viewed by using IBM MQ Explorer.

1. Right-click the queue manager in the **MQ Explorer - Navigator** view and then click **Application Connections**.
2. Click the application.
3. Right-click the application object name and then click **Properties**.
4. The **Read ahead** status field shows one of the following values:
  - **Yes** indicates that read ahead is enabled and is being used efficiently.
  - **No** indicates that read ahead is not enabled on any open queues.
  - **Inhibited** indicates that an application requested read ahead but it is inhibited because of incompatible options that are specified on the first call to MQGET.
  - **Backlog** indicates that read ahead is enabled but is not being used efficiently. **Backlog** can indicate that non-persistent messages were streamed into memory on the client system, but the client program is not requesting them. For example, the program might be using MQGET for specific correlation IDs.

## Using MQSC to get read ahead status

```
DISPLAY CONN(*) TYPE(HANDLE) WHERE(READA EQ YES) ALL
CONN(301AEB4820008C02)
  EXTCCONN(414D5143514D43303120202020202020)
    TYPE(CONN)
    .
    .
    ASTATE(ACTIVE)
    OBJNAME(QL.1)          OBJTYPE(QUEUE)
    OPENOPTS(MQOO_INPUT_SHARED,MQOO_OUTPUT)
    HSTATE(ACTIVE)          READA(YES)
```

Managing clients and client connections

© Copyright IBM Corporation 2017

*Figure 1-22. Using MQSC to get read ahead status*

The figure shows an example of using the display connection (`DISPLAY CONN`) command in MQSC to show the read ahead status.

The **READA** field can show the following values:

- **YES** indicates that read ahead is enabled on an open queue and is being used efficiently.
- **NO** indicates that read ahead is not enabled on any open queues.
- **INHIBITED** indicates that an application requested read ahead but it is inhibited because of incompatible options that are specified on the first call to MQGET.
- **BACKLOG** indicates that read ahead is enabled but is not being used efficiently. BACKLOG can indicate that non-persistent messages were streamed into memory on the client system, but the client program is not requesting them. For example, the application might be using MQGET for specific correlation IDs.

## 1.3. Asynchronous put

## Asynchronous put introduction

- One of the strengths of MQ is assured message delivery
- High qualities of service are not required for all messages in all applications
- With an asynchronous response from an **MQPUT** (or “fire and forget”) messages can be streamed down a network link without waiting for a return code
- Relevant in a client connection where a high quality of service is required
- In the right circumstances, significant performance improvement can be seen
- Enabled by a combination of API settings and queue and topic definitions

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-23. Asynchronous put introduction

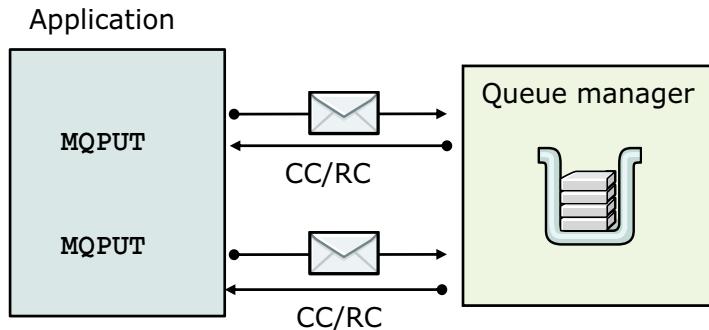
Asynchronous put (also known as *fire and forget*) recognizes that a large proportion of the cost of an MQPUT from a client is the turnaround of the network connection. When the application uses asynchronous put, it sends the message to the server but does not wait for a return code before it sends more MQI calls.

You can use asynchronous put to improve messaging performance in some situations. The largest performance improvement is seen in cases where the application sends many MQPUT calls and where the network is slow.

After the application completes the PUT sequence, it sends other calls, such as MQCMIT or MQDISC, which clears any MQPUT calls that are not complete.

## MQPUT processing

- Normally, when an application puts a message on a queue, the application must wait for the queue manager to confirm that it processed the MQI request
- A completion code (CC) and reason code (RC) is returned after each message is written



[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

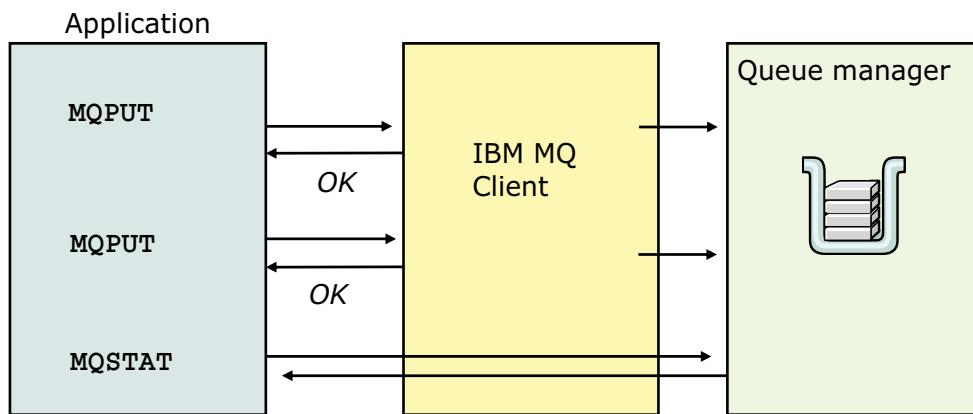
Figure 1-24. MQPUT processing

When a message is PUT by an application in bindings mode, the message is written directly to the queue in the queue manager.

Normally, when an application puts messages on a queue, by using MQPUT or MQPUT1, the application must wait for the queue manager to confirm that it processed the MQI request. The queue manager confirmation includes a completion code and a reason code.

## MQPUT in client mode with asynchronous put

- Application puts message but the queue manager does not return the success or failure of each call
  - Client returns CC and RC with assumption that put was successful
  - An “OK” return from asynchronous put does mean that the message was delivered
- Application should use **MQSTAT** call to get the status information periodically



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-25. MQPUT in client mode with asynchronous put

The figure shows an application with the asynchronous put function.

When the IBM MQ client receives the message from the application, it sends the message over the client channel. The client then returns an “OK” return code to the application without waiting for any reply from the queue manager.

When the application uses asynchronous put, the application should periodically send an MSTAT call to get the status information.

## When asynchronous put applies

- Determined by combination of options in the MQPUT call and attributes of the queue
  - Application sets MQPMO\_ASYNC\_RESPONSE or MQPMO\_RESPONSE\_AS\_Q\_DEF in the MQPUT message options (MQPMO)
  - AND message is nonpersistent OR message is persistent and put within a unit of work
  - AND application client is connected
- If the conditions are not met, then a “normal” PUT occurs
- Queue attribute identifies default put response type

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-26. When asynchronous put applies

To qualify for *fire and forget* put delivery, a message must meet a set of criteria. The process of determining whether it meets the criteria is carried out over two stages, which are information from the application.

In a nondistribution list scenario, an application put request is eligible for asynchronous delivery if one of the following conditions is true:

- The application MQPMO option field in the MQPUT call specifies MQPMO\_ASYNC\_RESPONSE.
- The application MQPMO options field in the MQPUT call specifies MQPMO\_RESPONSE\_AS\_Q\_DEF (or its synonym MQPMO\_RESPONSE\_AS\_TOPIC\_DEF) and the DEFRESP attribute was set to an asynchronous value at the time it was opened.

Finally, one or more of the following conditions must also be true:

- The message is a persistent message that is being put within a unit of work.
- The message MQMD persistence is set to MQPER\_NOT\_PERSISTENT.
- In the MQPUT call, the message MQMD **Persistence** attribute is set to MQPER\_AS\_Q\_DEF (or its synonym MQPER\_AS\_TOPIC\_DEF) and the object DEFPSIST attribute was set to non-persistent when the object was opened.

## Sync point

- A successful commit for a unit of work means that all asynchronous puts were carried out successfully
- A subsequent **MQGET** (in the same unit of work) for a message that was PUT asynchronously, succeeds as normal

Managing clients and client connections

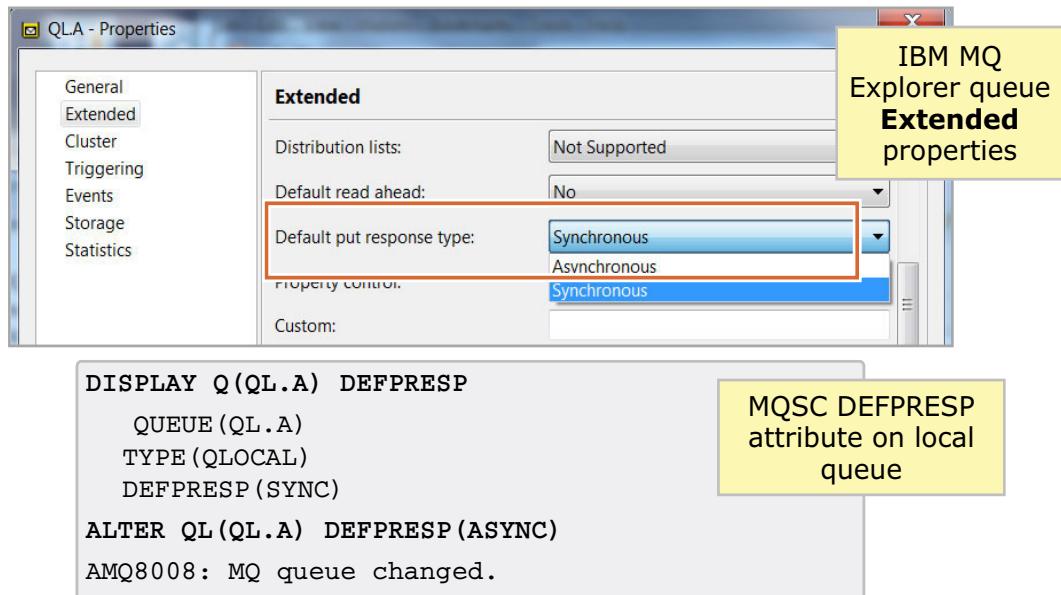
© Copyright IBM Corporation 2017

Figure 1-27. Sync point

If the application successfully commits a unit of work, then all messages put asynchronously were successfully written to the queue.

## Setting the default put response type on a queue

- If the application uses the MQPMO\_RESPONSE\_AS\_Q\_DEF value in the MQPMO, then the **Default put response type (DEFPRESP)** attribute of the queue controls whether the application can use the asynchronous put capability



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-28. Setting the default put response type on a queue

If the application uses the MQPMO\_RESPONSE\_AS\_Q\_DEF value in the put message options, then the **Default put response type (DEFPRESP)** property of the queue controls whether the application can use the asynchronous put capability.

The **Default put response type** property can be set by using IBM MQ Explorer or MQSC as shown in the figure.

- Synchronous** ensures that the PUT operations to the queue that specifies MQPMO\_RESPONSE\_AS\_Q\_DEF are sent as if MQPMO\_SYNC\_RESPONSE was specified instead. **Synchronous** is the default that is supplied with IBM MQ, but your installation might change it.
- Asynchronous** ensures that the PUT operations to the queue that specifies MQPMO\_RESPONSE\_AS\_Q\_DEF are sent as if MQPMO\_ASYNC\_RESPONSE was specified instead.

## 1.4. Client modes for MQSC

## MQSC client modes

- Client-local mode (-n) offers a limited set of MQSC commands to manage client channel definition files
- Client-connect mode (-c) connects to a remote queue manager by using a client channel and issues escaped PCF commands

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-29. MQSC client modes

As shown in the figure, IBM MQ supports two client modes.

Client-local mode sends MQSC commands to a queue manager through either a local or a client connection.

Client-connect mode sends MQSC commands against the client channel table without requiring a connection to a queue manager.

## MQSC client-local mode (-n)

- Modifies the `runmqsc` command to not connect to a queue manager
  - All other command parameters must be omitted
  - Requires that the client libraries are installed
- MQSC commands that are entered in this mode are limited to managing the local channel definition file
- Limited to the following MQSC commands:
  - ALTER, DEFINE, DELETE, DISPLAY AUTHINFO (of the type CRLLDAP or OCSP only)
  - ALTER, DEFINE, DELETE, DISPLAY CHANNEL (of the type CLNTCONN only)

Figure 1-30. MQSC client-local mode (-n)

The client-local mode is started by adding the `-n` option to the `runmqsc` command. If this parameter is specified, all other command parameters must be omitted, otherwise an error message is generated.

Client-local mode requires the client libraries. If the client libraries are not installed on the client computer, an error message is generated.

MQSC commands that are entered in client-local mode are limited to managing the local channel definition file, which is located through the `MQCHLLIB` and `MQCHLTAB` environment variables, or the default values if not defined.

## MQSC client-connect mode (-c)

- Modifies the `runmqsc` command to connect to a queue manager by using a client connection
- Client channel definitions that are used to connect to the queue manager are located by using the following environment variables in this order of precedence: MQSERVER, MQCHLLIB, and MQCHLTAB
- Requires that the client is installed

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-31. MQSC client-connect mode (-c)

The `-c` option modifies the `runmqsc` command to connect to a queue manager by using a client connection. The client channel definitions that are used to connect to the queue manager are located by using the following environment variables in this order of precedence: MQSERVER, MQCHLLIB, and MQCHLTAB.

This option requires the client libraries. If the client libraries are not installed on the client computer, an error message reports that client libraries are missing.

## 1.5. Troubleshooting tips

## Troubleshooting IBM MQ channels

- Listener problems
- Channel configuration problems
- Network connectivity problems
- In-doubt channels

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-32. Troubleshooting IBM MQ channels

Most problems with IBM MQ channels can be identified as listener problems, channel configuration problems, network connectivity problems, or in-doubt channels. Tips and techniques for identifying and fixing each of these problems are described in this topic.

## Listener problems

- If you configured the listener as an MQ object, make sure that it is configured correctly and it is running
- If you are starting the listener manually, make sure that the listener process is active and listening
- List network status on the system to ensure that the listener process is active on the correct port
- Make sure that the queue manager is running

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-33. Listener problems

This figure lists the steps to take to identify listener problems.

First, make sure that the listener is running. For example, list the processes on your system and make sure that the `runmqclsr` process is present.

You can also get listener status by entering the following MQSC commands:

```
DISPLAY LISTENER(B.LISTENER) ALL  
DISPLAY LSSTATUS(B.LISTENER) ALL  
START LISTENER(B.LISTENER)
```

On Windows, you can run the `netstat -an` command to get a list of active ports and verify that the IBM MQ listener port is not already in use by another program.

On UNIX and Linux, you can use the `lsof` program to determine whether another application is using the port. For example, type: `lsof -i tcp:1414`

## Channel configuration problems

- Make sure that the channel definitions on the sending and receiving side match
- Ensure that channel names must match exactly
- Ensure that you used the correct channel type on both sides of the connection
- If the channel definition relies on the default port value of 1414, try specifying the port number explicitly
- Verify that the receiving channel is enabled

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

*Figure 1-34. Channel configuration problems*

Make sure that the channel definitions on the sending and receiving side match. Use the MQSC DISPLAY CHANNEL and DISPLAY CHSTATUS commands with the ALL option to display the definitions and the channel status on both sides of the channel.

Channel names must match exactly. Be sure to use single quotation marks around lowercase or mixed-case channel names in MQSC. For example, type `DEFINE CHANNEL ('a.to.b')`. Or better yet, always use uppercase characters for channel names.

Make sure that you used the correct channel type on both sides of the channel. For example, SENDER/RECEIVER, SERVER/REQUESTER, and CLUSSDR/CLUSRCVR channels are compatible, but a SENDER/SENDER pair is not valid.

If the channel definition relies on the default port of 1414, try specifying the port number explicitly. The default of 1414 does not apply if you define a TCP service that is called "MQSeries" on a different port number.

Make sure that the receiving channel is not disabled. If you stopped a receiver channel with the `STOP CHANNEL (A.TO.B)` command instead of the `STOP CHANNEL (A.TO.B) STATUS (INACTIVE)` command, then that channel is disabled and the sending side cannot start it. In this case, you must run `START CHANNEL (A.TO.B)` against the receiver channel to re-enable it, and then enter `START CHANNEL (A.TO.B)` on the sending side to get the channel to run.

## Network connectivity problems

- Use the TCP ping command to test network connectivity between the two systems
- Use Telnet to test the MQ port on the remote system  
Example: `telnet server1 1414`
- Use the MQSC PING CHANNEL command to test a channel by sending a special message to the remote queue manager and checking that the data is returned
- If the channel definition uses host names instead of IP addresses, make sure that they resolve to the expected address
- If a firewall exists between the systems, make sure that it is configured to allow MQ traffic
- On server-connection channels, set **SHARECNV** to 1 or higher to ensure that MQ sends heartbeats whenever the channel is otherwise idle, which prevents firewalls from ending the connection due to inactivity

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

*Figure 1-35. Network connectivity problems*

This figure lists some tips and techniques for finding common network connectivity problems.

You can use the TCP/IP `ping` command, the MQSC PING CHANNEL command, or Telnet to test network connectivity. For example, to use Telnet to test the IBM MQ port on the remote system, type: `telnet server1 1414`

If the channel uses host names instead of IP address, make sure that they resolve to the expected address. For example, use the `nslookup` or `dig` commands, where available, to ensure that the domain name service (DNS) servers are providing the right address information. Test with IP addresses in your channel definition if you are not sure whether the DNS is causing the problem.

If the network includes a firewall, make sure that it is configured to allow IBM MQ traffic. If necessary, you can use the LOCALADDR attribute on sending channels to ensure that the IBM MQ channel uses ports that the firewall allows.

## In-doubt channels

- MQ channels can show an “in doubt” status when the sending side does not know whether a previous batch of messages was received
- In most cases, starting the channel is sufficient to clear this status
- Might be necessary to check manually whether the receiving side received the last batch of messages

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

*Figure 1-36. In-doubt channels*

An in-doubt channel is a channel that is in doubt with a remote channel about which messages were sent and received. This figure lists some tips for identifying in-doubt channels.

In-doubt channel problems are typically resolved automatically. Even when communication is lost, and a channel is placed in doubt with a message batch at the sender with receipt status unknown, the situation is resolved when communication is reestablished.

You can, when necessary, resynchronize the channel manually. You can enter MQSC commands on the sending side to resolve in-doubt channels. For example:

```
STOP CHANNEL (A.TO.B)
RESOLVE CHANNEL (A.TO.B) ACTION(BACKOUT or COMMIT)
START CHANNEL (A.TO.B)
```

The RESOLVE CHANNEL command requests a channel to commit or back out in-doubt messages. This command is used when the other end of a link fails during the confirmation period, and for some reason it is not possible to reestablish the connection. In this situation, the sending end remains in doubt whether the messages were received. Any outstanding units of work must be resolved by being backed out or committed.

The RESOLVE CHANNEL command can be used only for sender (SDR), server (SVR), and cluster-sender (CLUSSDR) channels.

## MQSC connection and channel status commands

- **DISPLAY CONN**
  - Display connection information about the applications that are connected to the queue manager
  - Use the results to identify applications with long-running units of work
- **DISPLAY CHSTATUS**
  - Display the status of one or more channels

Managing clients and client connections

© Copyright IBM Corporation 2017

*Figure 1-37. MQSC connection and channel status commands*

You can use the **DISPLAY CONN** and **DISPLAY CHSTATUS** commands to get more information about a channel.

## Channel monitoring

- To enable, set **MONCHL** attribute on the channel
  - Set to **LOW**, **MEDIUM**, or **HIGH** to control at the channel level
  - Set to **QMGR** and then set **MONCHL** value in the queue manager to control at queue manager level
- To display monitoring information for a channel, use IBM MQ Explorer or the MQSC command **DISPLAY CHSTATUS () MONITOR**

Managing clients and client connections

© Copyright IBM Corporation 2017

*Figure 1-38. Channel monitoring*

If you require real-time monitoring information for a channel, you can set the monitoring level either at the object level or at the queue manager level.

You can set channel monitoring to **LOW**, **MEDIUM**, or **HIGH**.

- **LOW** measures a small sample of the data, at regular intervals. Use this level for objects that process a high volume of messages.
- **MEDIUM** measures a sample of the data, at regular intervals. Use this level for most objects.
- **HIGH** measures all data, at regular intervals. Use this level for objects that process only a few messages per second, on which the most current information is important.

You can use the IBM MQ Explorer or the **DISPLAY CHSTATUS** command with the **MONITOR** option to show the channel monitoring information.

## Channel monitoring example

- Sender channel QM1.TO.QM2 **MONCHL** attribute is set to the default value of **QMGR**
- Queue manager that owns the channel **MONCHL** attribute is set to **MEDIUM**

```
DISPLAY CHSTATUS (QM1.TO.QM2) MONITOR
  CHSTATUS (QM1.TO.QM2)
  XMITQ (Q1)
  CONNAME (127.0.0.1)
  CURRENT CHLTYPE (SDR)
  STATUS (RUNNING)
  SUBSTATE (MQGET)
  MONCHL (MEDIUM)
  XQTIME (755394737,755199260)
  NETTIME (13372,13372)
  EXITTIME (0,0)
  XBATCHSZ (50,50)
  COMPTIME (0,0)
  STOPREQ (NO)
  RQMNAME (QM2)
```

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-39. Channel monitoring example

This figure shows an example of using the `DISPLAY CHSTATUS` command to display the monitoring information for a channel that is named QM1.TO.QM2. The example assumes that the monitoring level is set to **MEDIUM**.

## Troubleshooting IBM MQ clients

- Make sure that the client application environment variables are set to the correct values
- If you are using a client channel definition file, examine the file and verify the settings
- If client application set its connection parameters programmatically, rather than relying on any external files or environment variables, check the client application code to ensure that it is using the correct client-channel definition attributes
- Check the security attributes of the server-connection channel to which the client connects
- Check the error logs on the client and queue manager servers

[Managing clients and client connections](#)

© Copyright IBM Corporation 2017

Figure 1-40. Troubleshooting IBM MQ clients

This figure lists some tips for troubleshooting IBM MQ clients.

You can check the client channel definition table without connecting to the queue manager by using the `runmqsc -n` command to run MQSC in client-local mode.

To display information about the client channel configuration, type:

```
DISPLAY CHANNEL(*) CHLTYPE(CLNTCONN) ALL
```

Be sure to check the security attributes of the server-connection channel.

## Unit summary

- Manage client performance by sharing conversations, using read ahead, and using asynchronous put
- Describe the client modes that MQSC supports
- Use troubleshooting tools and techniques to monitor and manage clients and connections

Managing clients and client connections

© Copyright IBM Corporation 2017

*Figure 1-41. Unit summary*

## Review questions (1 of 2)

1. Which of the following best describes the shared conversations option?
  - A. With shared conversations, multiple applications can share a single client channel.
  - B. With shared conversations, threads in different applications can share a single client channel.
  - C. With shared conversations, multiple threads in a single multi-threaded application can share a single client channel.
  - D. With shared conversations, applications on different servers can share a single client channel.
2. True or False: **MAXSHCNV** shows the highest value of the **SHARECNV** attribute on the associated client-connection and server-connection channels.



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-42. Review questions (1 of 2)

Write your answers here:

- 1.
- 2.

## Review questions (2 of 2)

3. Which of the following statements best describe read ahead?
  - A. IBM MQ client can read messages from the queue and store them in a local memory buffer.
  - B. IBM MQ client can read through the messages on the queue to determine how many messages there are.
  - C. IBM MQ client can use the read ahead capability to read messages that are not yet committed to the queue.
  
4. True or False: An IBM MQ client with read ahead processing stops reading messages off the queue on the queue manager if the message that it last read is persistent.
  
5. True or False: Asynchronous PUT is the default for non-persistent messages that are put from an IBM MQ client application.



Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-43. Review questions (2 of 2)

Write your answers here:

3.

4.

5.

## Review answers (1 of 2)



1. Which of the following best describes the shared conversations option?
  - A. With shared conversations, multiple applications can share a single client channel.
  - B. With shared conversations, threads in different applications can share a single client channel.
  - C. With shared conversations, multiple threads in a single multi-threaded application can share a single client channel.
  - D. With shared conversations, applications on different servers can share a single client channel.

The answer is C.

2. True or False: MAXSHCNV shows the highest value of the SHARECNV attribute on the associated client-connection and server-connection channels.

The answer is False. MAXSHCNV shows the lowest value of the SHARECNV attribute on the associated client-connection and server-connection channels.

## Review answers (2 of 2)

3. Which of the following statements best describe read ahead?
  - A. IBM MQ client can read messages from the queue and store them in a local memory buffer.
  - B. IBM MQ client can read through the messages on the queue to determine how many messages there are.
  - C. IBM MQ client can use the read ahead capability to read messages that are not yet committed to the queue.

The answer is A.
4. True or False: An IBM MQ client with read ahead processing stops reading messages off the queue on the queue manager if the message that it last read is persistent.

The answer is True.
5. True or False: Asynchronous PUT is the default for non-persistent messages that are put from an IBM MQ client application.

The answer is False. The default is always synchronous PUT.

Managing clients and client connections

© Copyright IBM Corporation 2017

Figure 1-45. Review answers (2 of 2)

---

# Unit 2. Securing IBM MQ channels with TLS

## Estimated time

01:30

## Overview

In this unit, you learn how to use Transport Layer Security (TLS) to secure IBM MQ channel communications that include mutual authentication.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Describe the certificate infrastructure that is supported in IBM MQ  
Manage certificates with IBM Key Management
- Manage certificates with IBM MQ Key Management
- Describe cipher specifications and their support in IBM MQ
- Use certificate revocation lists or Online Certificate Status Protocol (OCSP) to validate currency of certificates
- Use TLS to secure IBM MQ channel communications

Securing IBM MQ channels with TLS

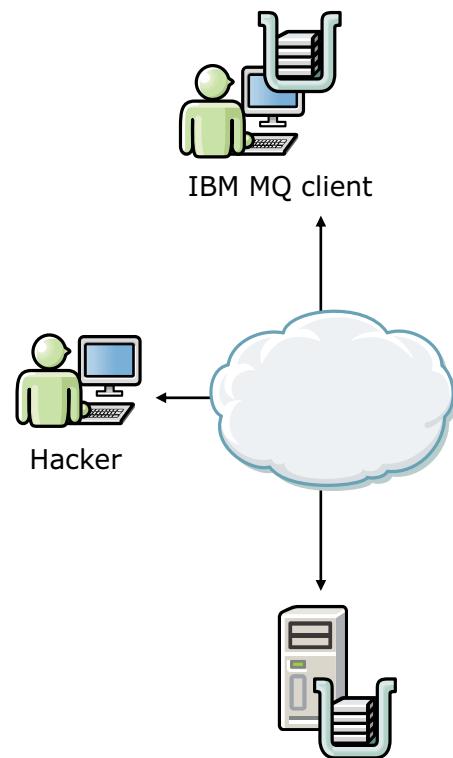
© Copyright IBM Corporation 2017

*Figure 2-1. Unit objectives*

## 2.1. TLS overview

## The security problems

- **Eavesdropping and message privacy**  
How can you stop someone from seeing the information that you send?
- **Tampering and message integrity**  
How can you stop someone from changing the information that you send?
- **Impersonation and authentication**  
How do you know that a person is who they say they are?
- **Replay**  
An extension of eavesdropping



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-2. The security problems

Whenever data is being transmitted over a network, many security problems can occur.

- The data can be captured and read, and a hacker can use the information. The problem is message privacy.
- The data can be damaged or changed, either accidentally or intentionally, before it arrives at its destination. The problem is message integrity.
- The communicating parties normally want to be sure about the identity of the partner. The sender of a message wants to know who receives the data, and the receiver wants to be sure of the identity of the originator. The problem is user authentication.

This unit looks at the different ways you can encrypt your message and the problems that are associated with security.

## Data encryption and signing

- Encrypt data with the public key of the owner
  - Only the owner can decrypt it
- Encrypt with a private key
  - If they have the public key, anyone can decrypt data
  - Identifies message source; it is a signed message
- Encrypt with a private key and a public key
  - Only you can decrypt it
  - Identifies message source; it is a signed message
- Hash the message and encrypt the hash (change detection)
  - Message digest or Message Authentication Code (MAC)
  - Easy to compute
  - Difficult to reverse
- Key ring or keystore: a repository for keys
  - Keep private key private

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-3. Data encryption and signing

To deal with the problem of eavesdropping, encrypt the information before you send it so that an eavesdropper cannot read the information.

Various combinations of encryption that use both public and private keys are shown here. The implications of public and private keys are also described.

*Message digests* are fixed-size numeric representations of the contents of messages, which are inherently variable in size. A *hash function* computes a message digest, which is a transformation that meets two criteria:

1. The hash function must be one way. It must not be possible to reverse the function to find the message corresponding to a message digest, other than by testing all possible messages.
2. It must be computationally infeasible to find two messages that hash to the same digest. A message digest is also known as a *message authentication code* because it can provide assurance that the message was not modified. The message digest is sent with the message itself. The receiver can generate a digest for the message and compare it with the sender digest. If the two digests are the same, the message integrity is verified. Any tampering with the message during transmission almost always results in a different message digest.

## Elementary encryption: Symmetric key

- Monoalphabetic cipher: Change A to D, B to Z, and C to Q
  - With enough data, it can be decrypted ABBA = DZZD
  - Low-cost encryption
  - If they know the key, anyone can read data
  
- Polyalphabetic cipher
  - Multiple schemes
  - Harder to break
  - Low-cost encryption
  - If they know the keys and schemas, anyone can read data

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-4. Elementary encryption: Symmetric key

Cryptography is the process of converting between readable text, called *plaintext*, and an unreadable form, called *ciphertext*.

1. The sender converts the plaintext message to ciphertext. This part of the process is called *encryption*.
2. The ciphertext is transmitted to the receiver.
3. The receiver converts the ciphertext message back to its plaintext form. This part of the process is called *decryption*.

The conversion involves a sequence of mathematical operations that change the appearance of the message during transmission but do not affect the content. Cryptographic techniques can ensure confidentiality and protect messages against unauthorized viewing because an encrypted message is not understandable.

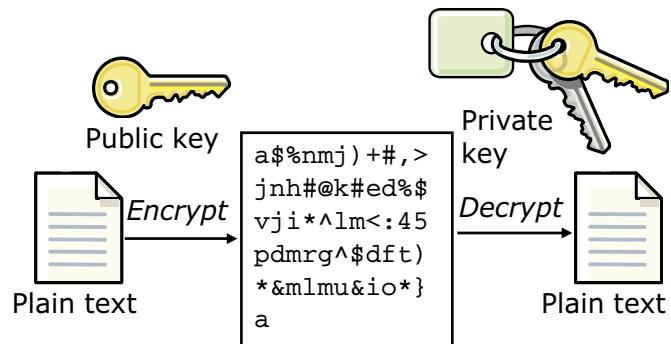
Cryptographic techniques involve a general algorithm, which is made specific by the use of keys. Cryptographic techniques that require both parties to use the same secret key are known as symmetric algorithms.

Monoalphabetic cipher is also known as a cryptogram, which uses a key that consists of rearranging the letters of the alphabet. Polyalphabetic substitution uses several substitution alphabets instead of just one.

## Elementary encryption: Asymmetric keys

- Uses key pairs
  - Encryption key is public
  - Decryption key is private
- Asymmetric keys cannot decrypt even if you know both the public key and algorithm
- Each person has a unique key
- Based on large prime numbers
- Standard key sizes:
 

▪ 512 bits	Low-strength key
▪ 768 bits	Medium-strength key
▪ 1024 - 4096 bits	High-strength key
- Asymmetric algorithms are much slower than symmetric ones



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-5. Elementary encryption: Asymmetric keys

Asymmetric keys can be used to implement digital signing, as anything encrypted with the private key of the owner can be used to show that it came from them, if the private key is secure.

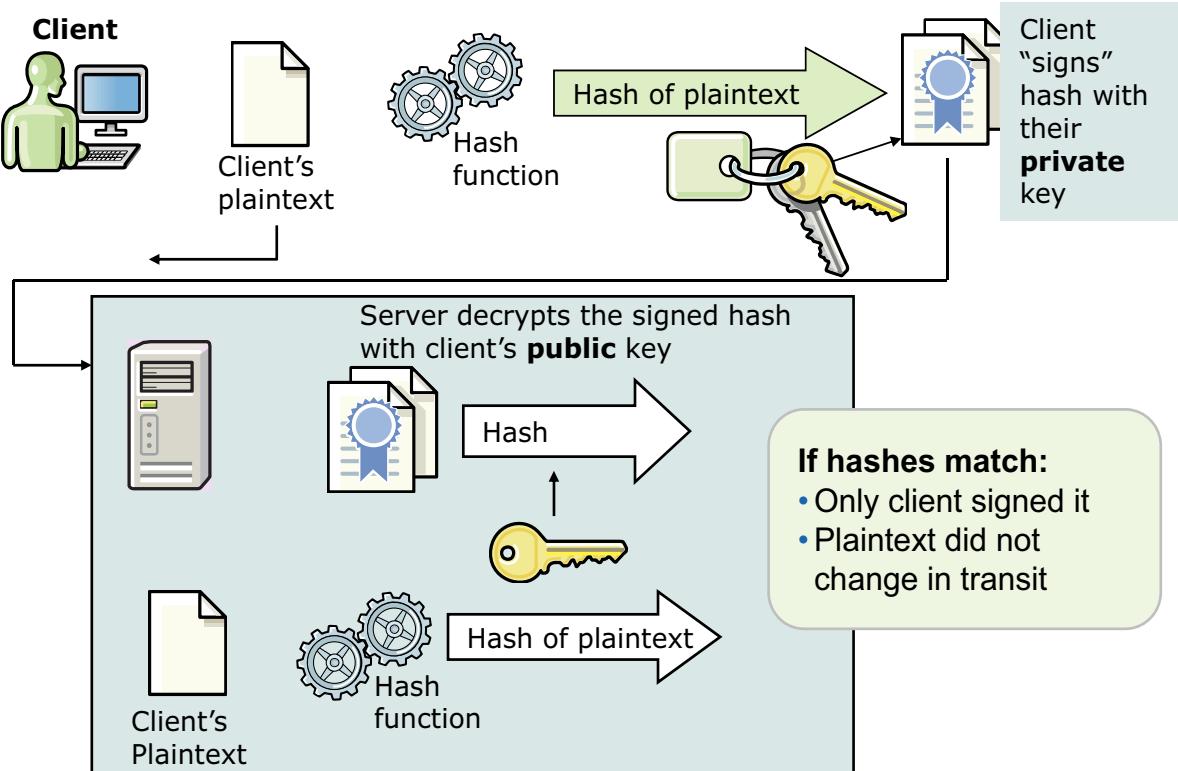
These two keys are mathematically related and they form a key pair. One of these two keys is kept private and the other can be made public.

A private key is typically used for encrypting the message-digest; in such an application, private-key algorithm is called *message-digest encryption algorithm*. A public key is typically used for encrypting the secret-key; in this application, a private-key algorithm is called a *key encryption* algorithm.

Key sizes determine the strength of level of security. For example, for ordinary use a key size of 768 bits might be acceptable, but for corporate use a key size of 1024 - 4096 bits provides a higher level of encryption.



## Digital signature



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

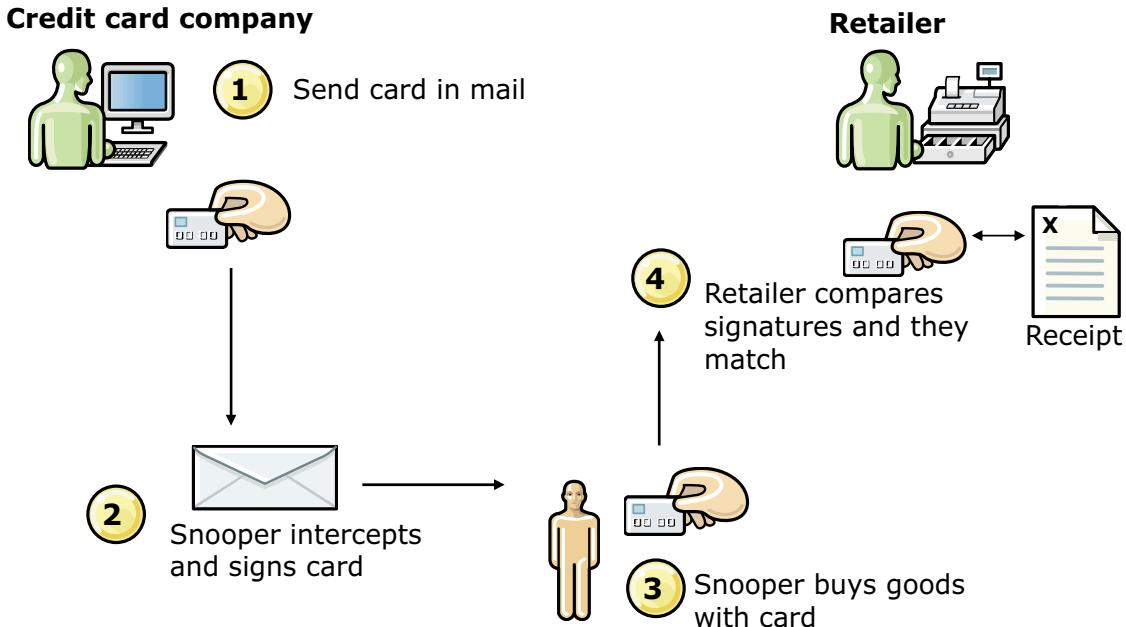
Figure 2-6. Digital signature

Digital signatures combine the use of the one-way hash function and public/private key encryption.

The message is hashed to provide a small message digest or hash. For demonstration purposes, think of it as a unique number that is generated from the plaintext message.

The client private key encrypts this hash or number to create the digital signature. The recipient of the message can also hash the received plaintext message to get a hash number. It can use the client public key to decrypt the digital signature to get the original hash number. If these numbers match, then the message came from the client and it did not change since it was signed.

## Relative authentication



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-7. Relative authentication

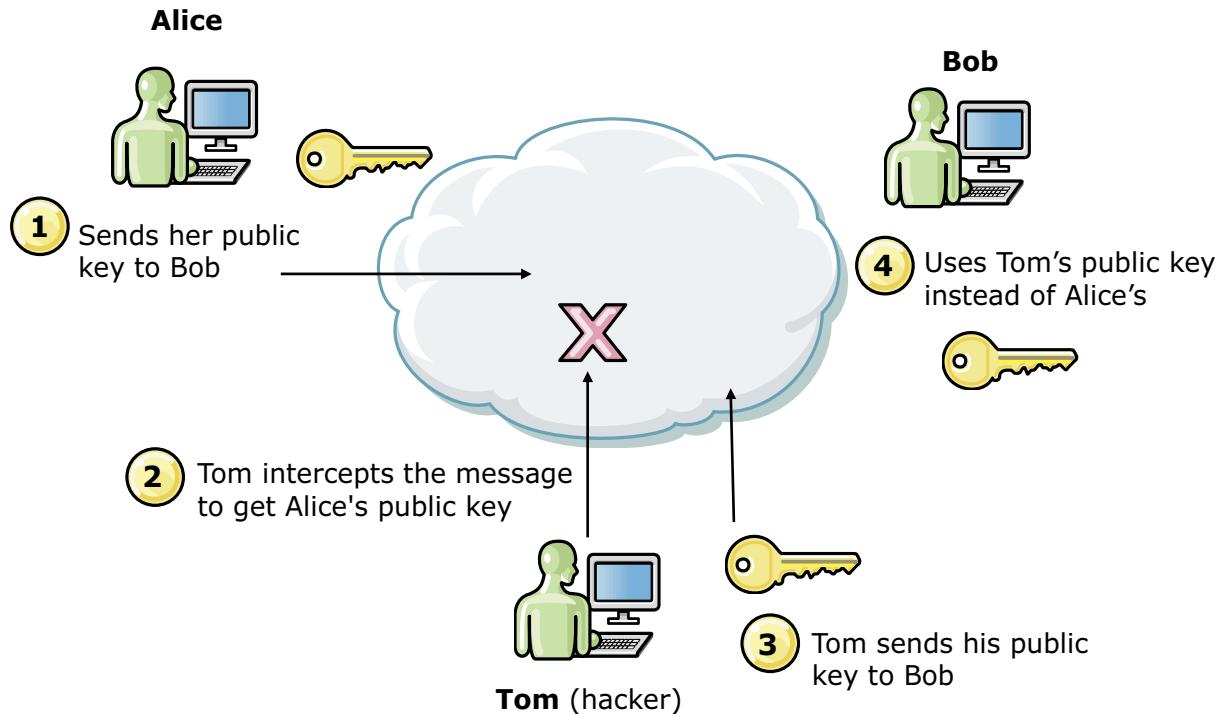
The figure shows examples of relative authentication and the problems that are associated with it.

1. When a credit card company sends a new credit card, they send it in the mail to your address.
2. If someone else picks up the mail before you, they can take your new credit card and sign your name (which is printed on the front of the card) in their own handwriting.
3. The identify thief can then use the credit card to purchase goods.
4. A retailer that checks the signature on the receipt would see that they matched, but this step verifies only that you are the person who signed the card.

The signature on the back of your credit card is an example of *relative authentication*. All the retailer can check is if the signature on the back of the card is the same as the signature on the receipt. This process does not verify that you are the person that is named on the credit card. Another authentication method is necessary to verify that you are the person that is named on the credit card. Other authentication methods include checking a driver's license or requiring a personal identification number (PIN).

Sometimes relative authentication is all that is required but it might not provide enough security.

## The key distribution problem



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-8. The key distribution problem

The initial transfer of public keys can be subject to a *man-in-the-middle* attack.

For example, Alice wants to send her public key to Bob so that he can use it to encrypt transmissions back to her. However, a hacker, Tom, intercepts the message from Alice and replaces Alice's public key with his own before he sends it to Bob. When Bob receives the key and encrypts his message with it, Tom can intercept the message and decrypt it with his private key to read their messages.

All Bob can prove is that the person who owns the private key that is paired with the public key he was sent is the same person that writes the message that Bob gets. This scenario is another example of relative authentication. Bob cannot authenticate that the public key belongs to Alice.

## Digital certificates

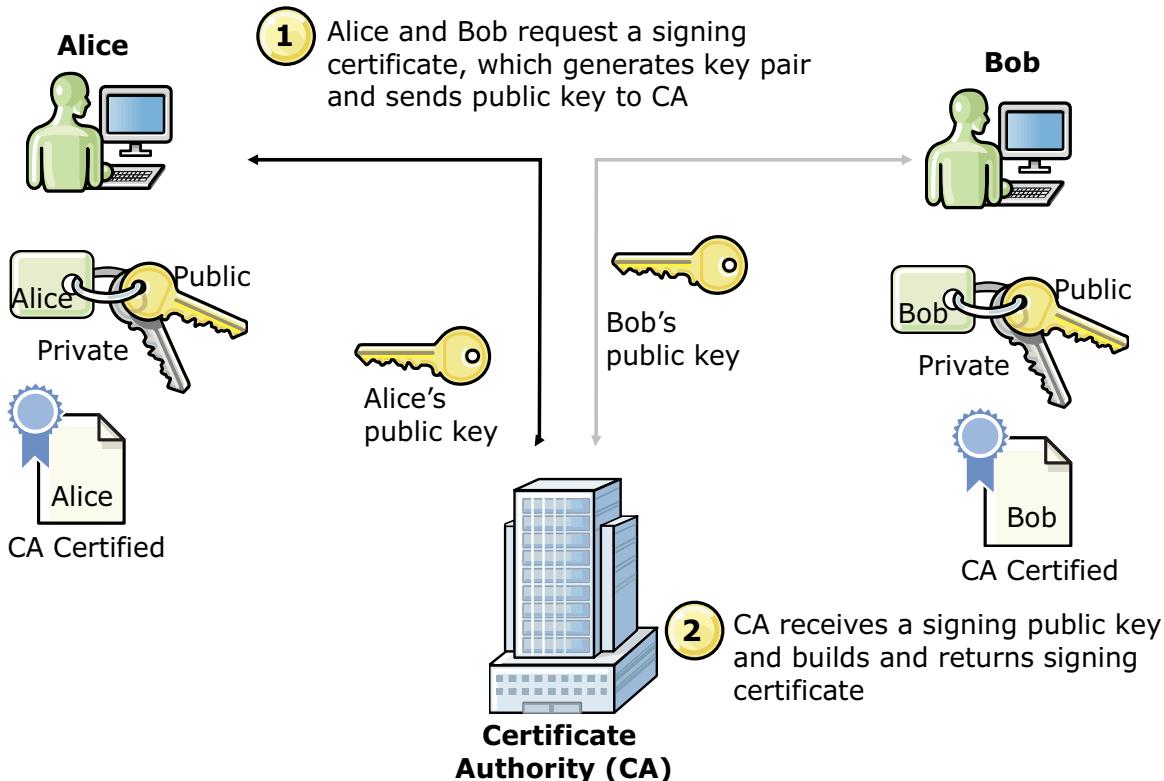


Figure 2-9. Digital certificates

A digital certificate contains information about the individual, for example, their name, company, and public key. The certificate is signed with a digital signature by a certificate authority (CA), which is a trustworthy authority.

To resolve the problem that is described in the previous example, Alice and Bob would request a signing certificate from a CA. Instead of sending each other their public keys, they would send them to the CA. The CA would verify the identity of the sender, and then create the certificates for Alice and Bob.

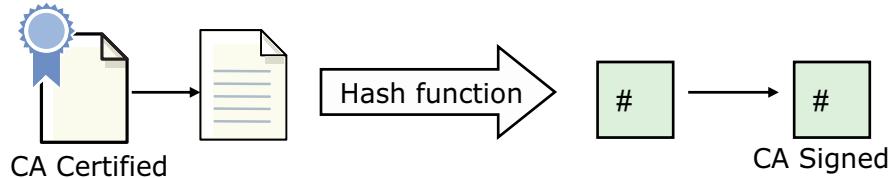
A CA is an independent and trusted third party that provides digital certificates. Digital certificates provide you with an assurance that the public key of an entity truly belongs to that entity.

A CA has the following roles:

- On receiving a request for a digital certificate to verify the identity of the requester before building, signing, and returning the personal certificate.
- To provide the public key of the CA in its CA certificate.
- To publish lists of certificates that are no longer trusted in a certification revocation list (CRL).

## Trusting a digital certificate

- Digital certificate is plain text
- Can be subject to tampering
- CA signs at creation



**Digital signature of the CA allows tampering to be detected:**

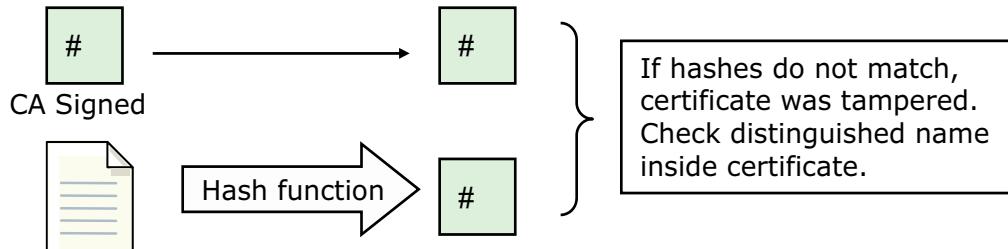


Figure 2-10. Trusting a digital certificate

If a key can be intercepted, can a certificate also be intercepted? How can you trust the certificate that was sent to you? Can someone tamper with a certificate to pretend to be someone they are not?

A digital certificate can be thought of as a piece of plaintext that can be subject to tampering. It is a file on your computer.

The “Digital certificate” example on the previous page showed Alice signing the plaintext document before she sent it to Bob. Bob can then check the signature to ensure that the message from Alice was not altered. The same technique is used to determine whether anyone tampered with a digital certificate.

The CA calculates the hash value of the plaintext (your certificate) and then signs that hash value with the CA private key to generate a CA digital signature. To check that the certificate is valid, the CA digital signature can be decrypted by using the CA public key to check that the hash values match. Well-known CA public keys are installed in many of the security products that use SSL/TLS for security.

Digital certificates do not contain your private key. You must keep your private key secret.

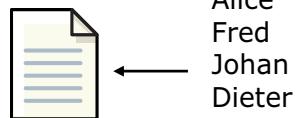
## Certificate revocation

- What happens if a certificate is no longer trusted?



Valid from Jan 1, 2016  
Valid to Jan 1, 2017

- Certificate revocation list (CRL) holds a list of certificates that CA marks as no longer trusted
- Authority revocation list (ARL) is a form of CRL that contains certificates that are issued to CA



*Figure 2-11. Certificate revocation*

A digital certificate has two dates that are associated with it. It has a date from which it is valid and it has a date after which it is invalid, or an expiration date.

What happens when a certificate was sent and it is no longer considered to be trusted before its expiration date passed? A certificate authority can revoke a certificate that is no longer trusted by publishing it in a certificate revocation list (CRL). When a certificate is received, it can be checked against this list to ensure that it was not revoked.

An authority revocation list (ARL) is a form of CRL that contains certificates that are sent to certifying authorities; contrary to CRL, which contains revoked end-entity certificates.

## Digital certificate details

- X.509 standard
- Digital certificates contain at least the following information about the entity that is certified:
  - Public key of the owner
  - Distinguished name of the owner
  - Distinguished name of the CA that is sending the certificate
  - Date from which the certificate is valid
  - Expiration date of the certificate
  - Version number
  - Serial number
- X.509 V2 certificates also contain:
  - Issuer identifier
  - Subject identifier
- X.509 V3 certificates can contain even more information

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-12. Digital certificate details*

Digital certificates that IBM MQ uses comply with the X.509 standard. The X.509 standard determines the specific pieces of information the digital certificates contain and the format for sending digital certificates.

Digital certificates contain at least the following information about the entity that is certified:

- The public key of the owner
- The distinguished name of the owner
- The distinguished name of the CA that is providing the certificate
- The date from which the certificate is valid
- The expiration date of the certificate
- A version number
- A serial number

An X.509 V2 certificate also contains an issuer identifier and a subject identifier.

An X.509 V3 certificate can contain some other extensions such as the basic constraint extension. Some extensions are standard, but others are implementation-specific. An extension can be critical, in which case a system must be able to recognize the field; otherwise, the system must reject the certificate. If an extension is not critical, the system can ignore the extension.

## Distinguished name

- Uniquely defines a user or entity
- X.509 format is well-defined

Example:

```
CN="Thomas J Watson" L="Yorktown Heights" ST=NY O=IBM
OU="IBM Headquarters" C=US
```

CN	Common name
T	Title
L	Locality name
ST/SP/S	State or province name
O	Organization name
OU	Organizational unit name
C	Country

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-13. Distinguished name

The distinguished name (DN) uniquely identifies an entity in an X.509 certificate. The format for the DN is described in the figure. The X.509 standard provides for a DN that is specified in a string format.

In the DN, the common name (CN) can describe an individual user or any other entity, for example, a web server.

The DN can contain multiple organization unit (OU) attributes, but only one instance of each of the other attributes is allowed. The order of the OU entries is significant. The order specifies a hierarchy of organizational unit names, with the highest-level unit first.

The X.509 standard defines other attributes that do not usually form part of the DN but can provide optional extensions to the digital certificate.

## Transport Layer Security (TLS) concepts

- Provides privacy and data integrity for data that are exchanged over a network
- Composed of two layers
  - TLS Record Protocol layer provides connection security
  - TLS Handshake Protocol layer supports client and server authentication and encryption algorithm and cryptographic key negotiation before any data is exchanged
- Used whenever a cipher specification is prefaced with “TLS” or “ECDHC” is specified

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-14. Transport Layer Security (TLS) concepts

The TLS protocol allows two parties to communicate with privacy and data integrity. Although they are similar, TLS and SSL are not interoperable.

The TLS protocol provides communications security over the internet, and allows client/server applications to communicate in a way that is private and reliable. The protocol has two layers:

- TLS record protocol
- TLS handshake protocol

The TLS layer requires a transport protocol such as TCP/IP.

For more information about the TLS protocol, see the information that the TLS Working Group provides on the website of the Internet Engineering Task Force at <http://www.ietf.org>.

## 2.2. Implementing TLS in IBM MQ

## TLS support in IBM MQ

- Authentication
  - Queue managers or clients that initiate an TLS-enabled connection are assured of the identity of the queue manager that they are connecting to
  - Queue managers that are receiving connections can check the identity of the queue manager or client that initiates the connection
- Message privacy
  - If configured, TLS uses a unique session key that encrypts all information that is exchanged over the connection to ensure that unauthorized parties cannot view information
- Message integrity
  - Data cannot be tampered with over the connection
- Certificate authority chain
  - Each certificate in the CA chain is signed by the entity that its parent certificate in the chain identifies
  - Root CA always signs root CA certificate at the head of the chain
  - Signatures of all certificates in the chain must be verified

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-15. TLS support in IBM MQ*

IBM MQ supports some cipher specifications that use the TLS V1.0 and V1.2 protocols.

An optional feature on the TLS handshake is the authentication of the client certificate and the server certificate.

SSL can allow for the client to pass many cipher specifications on the SSL handshake and the server can choose one that it supports. IBM MQ imposes the restriction that only one cipher specification can be supplied on the channel definition, which must match at both ends.

SSL can allow its sessions to be reused. Session reuse might be useful for short-lived web queries. However, IBM MQ channels are likely to be longer-running, so the session reuse feature of SSL is not used.

TLS is the same as SSL in terms of IBM MQ resource definitions. The cipher specification is the only setting for TLS.

## Cipher specification support in IBM MQ

IBM MQ V9 on Windows, Linux, and UNIX supports the following TLS cipher specifications:

- TLS 1.0
  - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (TLS 1.0)
  - TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (TLS 1.0)
- TLS 1.2
  - ECDHE\_ECDSA\_AES\_128\_CBC\_SHA256
  - ECDHE\_ECDSA\_AES\_256\_CBC\_SHA384
  - ECDHE\_ECDSA\_AES\_128\_GCM\_SHA256
  - ECDHE\_ECDSA\_AES\_256\_GCM\_SHA384
  - ECDHE\_RSA\_AES\_128\_CBC\_SHA256
  - ECDHE\_RSA\_AES\_256\_CBC\_SHA384
  - ECDHE\_RSA\_AES\_128\_GCM\_SHA256
  - ECDHE\_RSA\_AES\_256\_GCM\_SHA384
  - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256
  - TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256
  - TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256
  - TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-16. Cipher specification support in IBM MQ

Not all of the TLS cipher specifications are available for specification on IBM MQ channels.

From IBM MQ Version 8.0.2, SSLv3 protocol and cipher specifications are deprecated.

From IBM MQ Version 8.0.3, some TLS1.0 and TLS1.2 cipher specifications are deprecated.

By default, you are not allowed to specify a deprecated cipher specification on a channel definition.

If you attempt to specify a deprecated cipher specification, you receive message AMQ8242:

*SSL/CIPH definition wrong*, and PCF returns MQRCCF\_SSL\_CIPHER\_SPEC\_ERROR.

You cannot start a channel with a deprecated cipher specification. If you attempt to start a channel with a deprecated cipher specification, the system returns *MQCC\_FAILED* (2) with a reason of *MQRC\_SSL\_INITIALIZATION\_ERROR* (2393) to the client.

It is possible for you to re-enable one or more of the deprecated cipher specifications for defining channels at run time on the server by setting the environment variable *AMQ\_SSL\_WEAK\_CIPHER\_ENABLE*.

## IBM MQ connection procedure with TLS

- When a queue manager connects to another queue manager
  - Queue managers exchange and validate certificates
  - Must configure both queue managers and the channels with appropriate certificate settings
- When messages are sent from one queue manager to another queue manager along a channel
  - Data is generally encrypted by using a session key that is established during the certificate exchange
  - Must configure the channels with appropriate cipher specifications

Securing IBM MQ channels with TLS

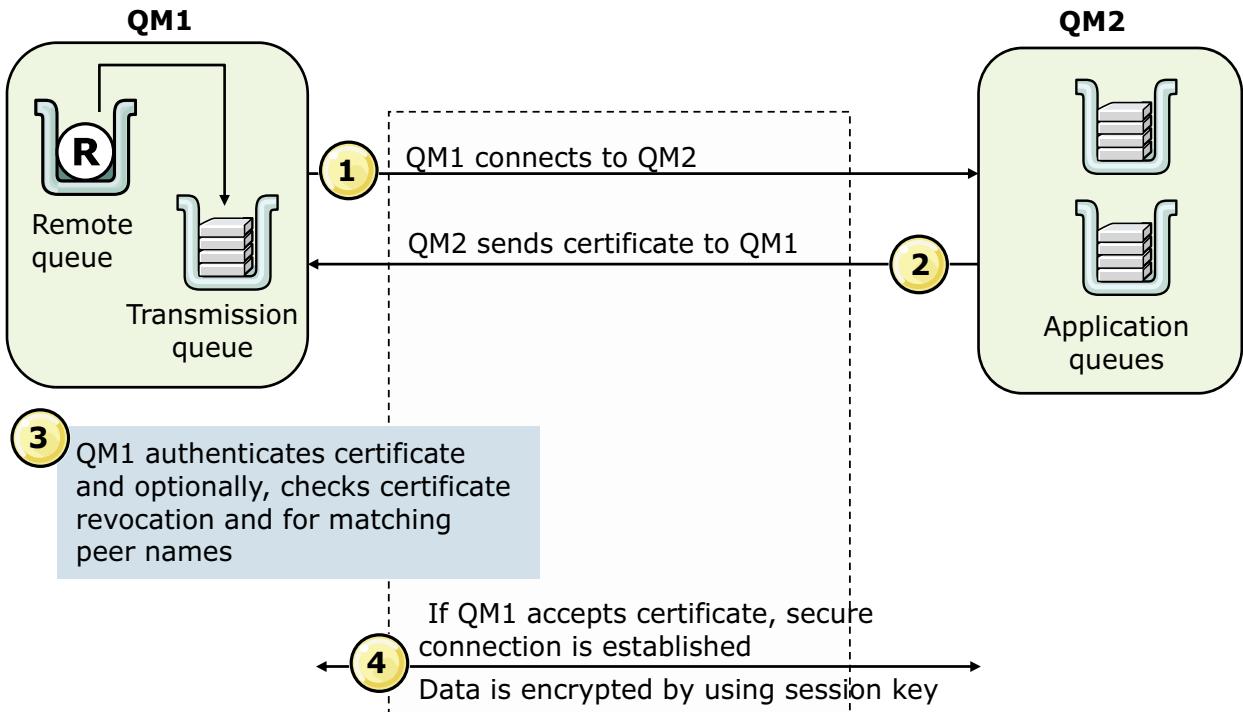
© Copyright IBM Corporation 2017

Figure 2-17. IBM MQ connection procedure with TLS

You can implement mutual authentication between two queue managers by using self-signed TLS certificates.

When messages are sent on a message channel, the data is generally encrypted by using a session key that is established during the queue manager certificate exchange. Message encryption assumes that the channels are configured with appropriate cipher specifications.

## TLS and IBM MQ certificate exchange



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-18. TLS and IBM MQ certificate exchange

This figure summarizes the queue manager certificate exchange process.

1. QM1 connects to QM2.
2. The personal certificate that is used by QM2 is sent to QM1.
3. QM1 authenticates the personal certificate against the chain of certificate authority certificates. QM1 optionally checks for certificate revocation if Online Certificate Status Protocol (OCSP) is supported on the server. OCSP is an Internet protocol that is used for obtaining the revocation status of an X.509 digital certificate.  
QM1 optionally checks the personal certificate against the CRL.  
QM1 optionally applies a filter to accept personal certificates that meet any defined peer names only.
4. If QM1 accepts the personal certificate from QM2, the secure connection is established.

## Configuring TLS on queue managers

1. Create the queue manager key repository
  - Store certificates that the queue manager uses
  - Use IBM Key Management
  
2. Specify the queue manager key repository location by using one of these options:
  - In IBM MQ Explorer queue manager properties
  - By using the `ALTER QMGR SSLKEYR` command
  - By specifying the **Key Repository** queue manager attribute
  
3. Use CRL or OCSP to ensure that the certificate is valid
  
4. Configure for cryptographic hardware (if support is required)

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-19. Configuring TLS on queue managers

This figure lists the configuration tasks that are required for setting up IBM MQ to use TLS.

The first step is to create the queue manager key repository by using the IBM Key Management application or commands. The key repository is a file that stores the certificates that the queue manager uses. On Windows, Linux, and UNIX, the key repository is known as the *key database file*.

After you create the key repository, you must configure the queue manager with the location of the key repository.

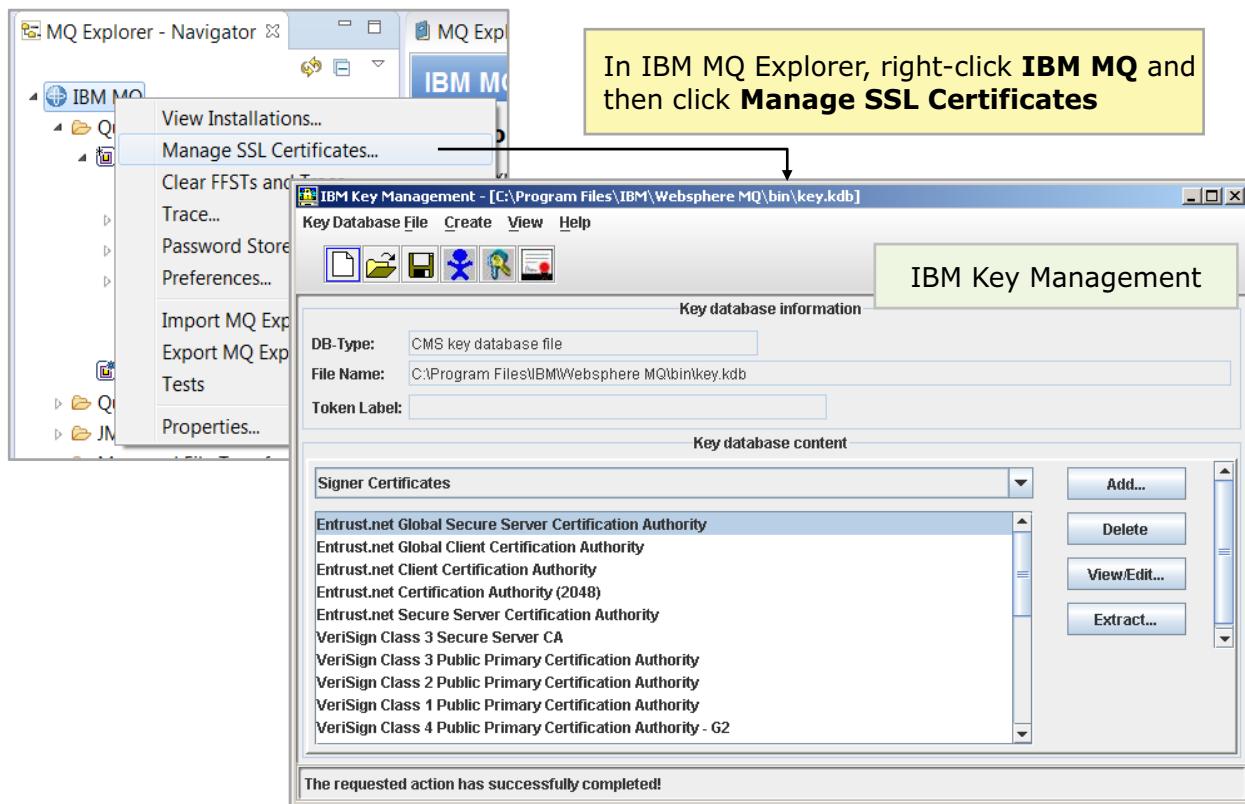
CAs can revoke certificates that are no longer trusted by publishing them in a CRL. The next step is to configure CRL checking, if CRL checking is required. When a queue manager receives a certificate, it can be checked against the CRL on an LDAP server or by using OCSP to ensure that it is not revoked. CRL checking is not mandatory for messaging with TLS, but it can ensure the trustworthiness of user certificates.

IBM MQ can support cryptographic hardware when necessary. The queue manager must be configured to support cryptographic hardware.

IBM Training

## Managing certificates

In IBM MQ Explorer, right-click **IBM MQ** and then click **Manage SSL Certificates**



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

*Figure 2-20. Managing certificates*

The IBM Key Management (`iKeyman`) application can manage the TLS certificates that IBM MQ provides.

On Linux or Windows, start the IBM Key Management application by running the `strmqikm` command.

On Windows, you also start the IBM Key Management application by clicking **IBM MQ > IBM Key Management** from the Windows **Program** menu.

If you have IBM MQ Explorer installed, you can start the IBM Key Management application by right-clicking **IBM MQ > Manage SSL Certificates**.

## Creating certificates

- Certificates contain the distinguished name
1. Create key database file by using IBM Key Management
  2. Specify **Generate a certificate request**
    - This request is written to a file or data set
    - Send it to the CA (can be sent by using their website)
    - Receive your signed certificate from the CA
  3. Import the certificate into the repository and label it

Securing IBM MQ channels with TLS

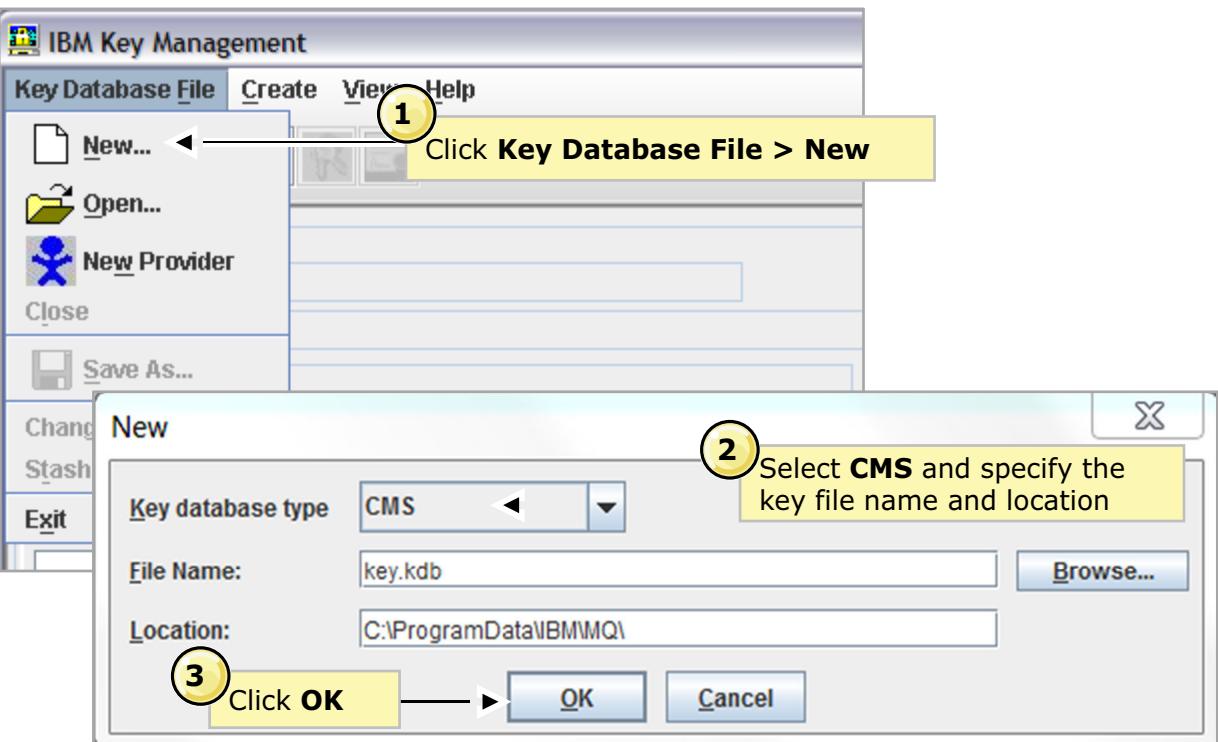
© Copyright IBM Corporation 2017

Figure 2-21. Creating certificates

This figure summarizes the steps for creating certificates.

A digital certificate can be created on your system and self-signed or that your site CA signed. These types of certificates might be useful for internal use certificates or for testing purposes. If you want to communicate with an external entity, you must obtain a certificate that a CA signed.

1. A TLS connection requires a key repository at each end of the connection. The queue manager and IBM MQ client must have access to a key repository. Create the key repository, also known as the key database file, by using IBM Key Management.
2. Generate a certificate request, or make a request that is based on an existing certificate in your repository, and send the certificate request to the CA.
3. After you receive a signed certificate, import it into the repository. The certificates are associated with individual queue managers by using a label, or with client logons for client applications.



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-22. Creating a key repository with IBM Key Management (1 of 2)

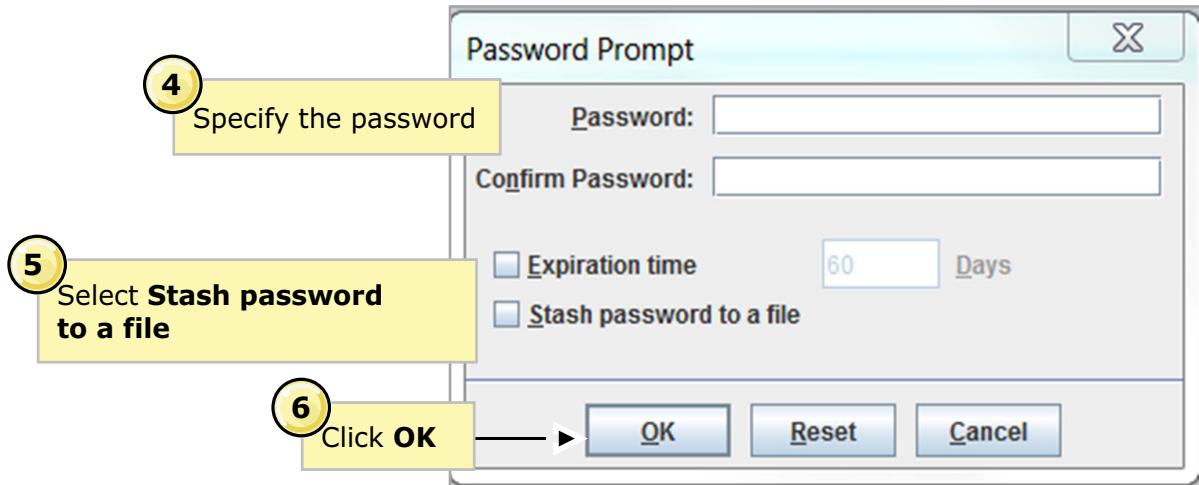
To create a key repository with IBM Key Management:

1. Select **Key Database File > New**.
2. Select **CMS** and specify the key file name and location.

The **File Name** field already contains the text `key.kdb`. If your stem name is `key`, leave this field unchanged. If you specified a different stem name, replace `key` with your stem name but you must not change the `.kdb` extension.

3. Click **OK**.

## Creating a key repository with IBM Key Management (2 of 2)



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-23. Creating a key repository with IBM Key Management (2 of 2)

4. Specify a password for the key repository.
5. Select the **Stash the password to a file** option.

If you do not stash the password, attempts to start TLS channels fail because they cannot obtain the password that is required to access the key database file.

6. Click **OK**. The signer certificates window displays, containing a list of the CA certificates that are provided with IBM Key Management and preinstalled in the key database. Set the access permissions.

When IBM Key Management is used to create the key database file, the access permissions for the key database file are set to give access only to the user ID that used IBM Key Management.

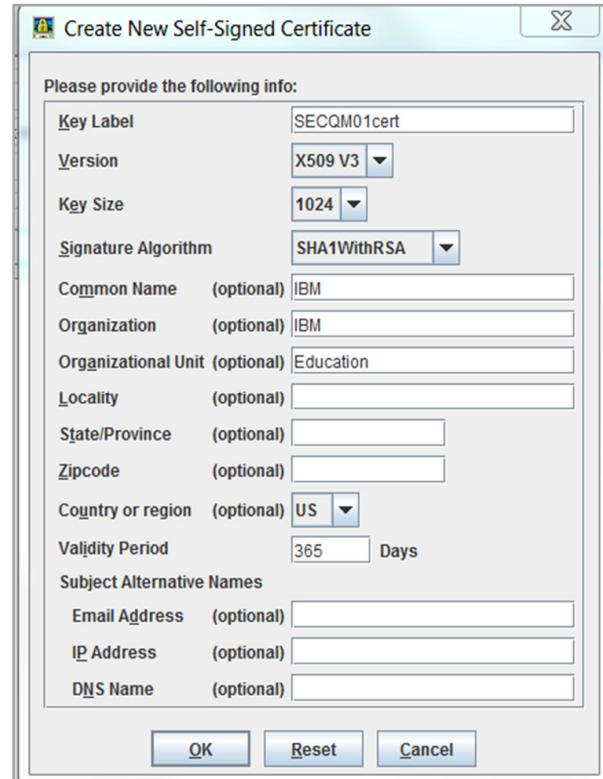
The MCA accesses the key database file, so ensure that the user ID under which the MCA runs has permission to read both the key database file and the password stash file.

MCAs can run under the “mqm” user ID, which is in the “mqm” group. After you create your queue manager key database file, work with the same user ID to add read permission for the “mqm” group.

# IBM Training

## Using IBM Key Management to create a self-signed personal certificate

1. Click **Key Database File > Open**
2. Click **Key database type** and select **CMS**
3. Specify the name and location of the key database file in which you want to save the certificate
4. Type the password that you set when you created the key
5. Click **Create > New Self-Signed Certificate**
6. In the **Key Label** field, enter the certificate label
7. Type or select a value for any field in the distinguished name



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

*Figure 2-24. Using IBM Key Management to create a self-signed personal certificate*

For testing purposes, you can create a self-signed personal certificate. This figure lists the steps for creating a self-signed personal certificate by using IBM Key Management.

The **Key Label** value must be one of the following values:

- The value of the `CERTLBL` attribute, if it is set on the queue manager or client
- The default `ibmwebspheremq` with the name of the queue manager that is appended, all in lowercase characters
- The default `ibmwebspheremq` with MQI client logon user ID appended, all in lowercase

## Using IBM Key Management to add a CA certificate

1. Click **Key Database File > Open**
2. Click **Key database type** and select **CMS**
3. Specify the key database file to which you want to add the certificate
4. Enter the key database password
5. In the **Key database content** field, select **Signer Certificates**
6. Click **Add**
7. Specify the certificate file name and location
8. In the **Enter a Label** window, type the name of the certificate

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-25. Using IBM Key Management to add a CA certificate

This figure lists the steps to use IBM Key Management to add a CA certificate or the public part of a self-signed certificate to the key repository.

If the certificate that you want to add is in a certificate chain, you must also add all the certificates that are above it in the chain. You must add the certificates in the descending order and start from the root, followed by the CA certificate immediately below it in the chain.

## Command options for managing certificates and keys

- **`runmqckm`** and **`runmqakm`** commands
  - Create the type of CMS key database files that IBM MQ requires
  - Create certificate requests
  - Import personal certificates
  - Import CA certificates
  - Manage self-signed certificates
  - Include `-help` to display syntax and options

	<code>runmqckm</code>	<code>runmqakm</code>
Creation of certificates and certificate requests with Elliptic Curve public keys	No	Yes
Stronger encryption of the key repository file	No	Yes
Certified as FIPS 140-2 compliant	No	Yes
Supports JKS and JCEKS key repository file formats	Yes	No

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-26. Command options for managing certificates and keys

As an option, you can use the `runmqckm` command (Windows and UNIX), and the `runmqakm` command (Windows, UNIX and Linux) to manage keys, certificates, and certificate requests.

You can use these commands to create the type of CMS key database files that IBM MQ requires, create certificate requests, import certificates, and manage self-signed certificates.

The table in the figure identifies the differences between the two commands.

## Using runmqakm to create and initialize a key database

- To create and initialize a key database file:

```
runmqakm -keydb -create -populate -db <filename>.kdb  
-pw <password> -stash
```

- **db** is the file name for the new key database
- **pw** is the password to use to protect the key database file
- **populate** populates the key database with some predefined trusted CA certificates (optional)
- **stash** saves the key database password locally in the .sth file so that it does not have to be entered on the command line

Figure 2-27. Using runmqakm to create and initialize a key database

This figure summarizes the syntax for creating and initializing a key database by using the `runmqakm` command.

## Using runmqakm to generate a self-signed certificate

- To generate a self-signed certificate and store it in the key database:

```
runmqakm -cert -create -db server.kdb -stashed  
-dn "CN=myserver,OU=mynetwork,O=mycompany,C=mycountry"  
-expire 7300 -label "My self-signed certificate"  
-default_cert yes
```

- dn specifies the distinguished name to use on the public key certificate
- expire indicates the number of days the certificate is valid
- label is a name to use for the self-signed certificate within the key database
- default\_cert makes the new certificate the default (optional)

Figure 2-28. Using runmqakm to generate a self-signed certificate

This figure summarizes the syntax for using the `runmqakm` to generate a self-signed certificate and store it in the key database.

## Key repository



- Contains digital certificate of entity
  - For queue manager, default label of `ibmwebspheremq<QmgrName>`
  - For client, with default label of `ibmwebspheremq<logonUserId>`
  - From various certificate authorities
- On queue manager, specify key repository path  
Example: `ALTER QMGR SSLKEYR ('/var/mqm/qmgrs/QM1/tls/key')`
- On clients, specify **SSLKeyRepository** entry in **SSL** stanza in `mqclient.ini` file  
Example: `SSL:  
SSLKeyRepository=C:\key`
- In environment variable, specify `MQSSLKEYR`  
Example: `export MQSSLKEYR=/var/mqm/ssl/key`

Do not include the **.kdb** extension in the file name as the queue manager appends this extension automatically

Figure 2-29. Key repository

Apart from the certificates for the queue manager and client, the key repository can also contain many signed digital certificates from various CAs. These signed digital certificates allow the key repository to be used to verify certificates that it receives from its partner at the remote end of the connection.

The naming of the certificate is crucial; it allows a queue manager to locate its own certificate. The structure and capitalization of the certificate name must be correct for the queue manager's operating system.

- By default, the certificate name is `ibmwebspheremq<qmgr-name>`, all in lowercase, on both. The label name can be overridden with a custom value.
- On clients, the default label is `ibmwebspheremq` followed by the logon user ID, mapped to lowercase characters. The label name can be overridden with a custom value.

On the queue manager, a digital certificate contains the identity of the owner of that certificate. Each queue manager has its own certificate. On all operating systems, this certificate is stored in a key repository by using your digital certificate management tool, such as IBM Key Management.

The key repository is specified on the queue manager object by using the `ALTER QMGR` command. On distributed operating systems, this name is the path and the stem of the file name for the key database file.

On an IBM MQ client, each user typically has a separate key repository file with access that is restricted to that user. This key repository file is accessed by using the environment variable `MQSSLKEYR`. An application can also specify it on the `MQCONN` `KeyRepository` parameter.

## Labeling the certificate

- Identifies which personal certificate in the key repository is sent to the remote peer
- For the queue manager, use the **CERTLBL** attribute

Example: `ALTER QMGR SSLKEYR('/var/mqm/qmgrs/QM1/ssl/key') CERTLBL('QM1Certificate')`

- For a client, use the **CertificateLabel** entry of the SSL stanza

Example:

```
SSL:
  SSLKeyRepository=C:\key
  CertificateLabel=MyCert
```

- For the environment, use the environment variable **MQCERTLBL**

Example: `export MQCERTLBL=MyCert`

*Figure 2-30. Labeling the certificate*

You can provide a label name for the IBM MQ queue manager or client to use. For the queue manager, you can use **CERTLBL** attribute on the **ALTER QMGR** command.

For clients, the certificate label can be provided in the locations:

- By the application in the MQSCO structure with the **SSLKeyRepository** location
- In the SSL stanza in the `mqclient.ini` file with the **SSLKeyRepository** location
- By using the environment variable **MQCERTLBL**

You do not need to run the `REFRESH SECURITY TYPE(SSL)` command if you change the certificate label. However, you must run a `REFRESH SECURITY TYPE(SSL)` command if you change the certificate label name on the queue manager.

## Benefits of labeling the certificate

- Administrator can label certificates to follow company policy
- Simplifies migration to new certificate when current certificate is ready to expire

Example:

Current certificate needs to change to new certificate 'QM1 Cert 2017'.

```
ALTER QMGR CERTLBL('QM1 Cert 2017')
REFRESH SECURITY TYPE(SSL)
```

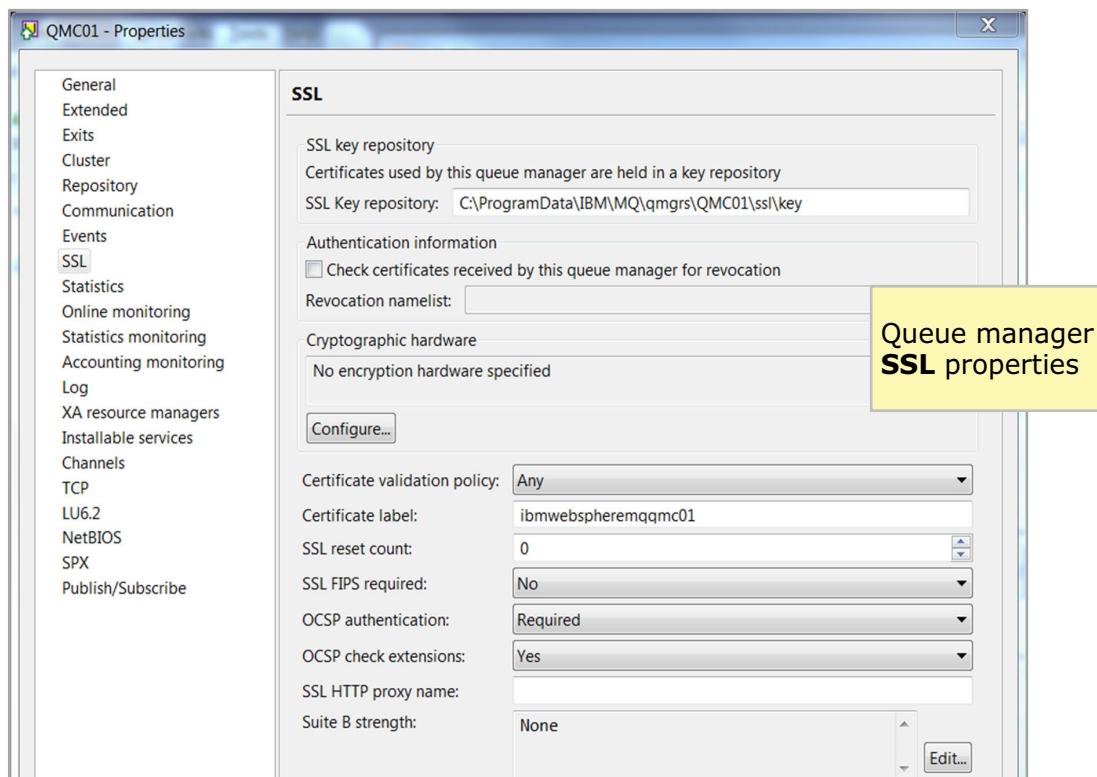
Figure 2-31. Benefits of labeling the certificate

A benefit of labeling a certificate is that you can label your own certificate instead of following the pattern that IBM MQ mandates.

Another benefit of labeling the certificate is that the administrator can change certificates that the channel or client uses. The job of installing the new certificate can be done at any prior point and labeled as you want. That label does not have to change to get the queue manager to use it; the IBM MQ administrator just needs to identify the label to the queue manager and then refresh the queue manager.



## Configuring TLS on queue manager with IBM MQ Explorer



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-32. Configuring TLS on queue manager with IBM MQ Explorer

You can configure TLS properties by using MQSC or IBM MQ Explorer. The configuration properties for TLS are on the **SSL** properties page.

## Configuring TLS on channels

- Define the cipher specification by using MQSC or **SSL** channel properties in IBM MQ Explorer
- Option to configure a channel to accept only certificates with attributes in the distinguished name of the owner that match given values
  - Can use the wildcard character (\*) at the beginning or the end of the attribute value in place of any number of characters
- Option to configure a queue manager channel so that the queue manager refuses the connection if the initiating party does not send its own personal certificate

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-33. Configuring TLS on channels

## Distinguished Name matching support in IBM MQ

Attribute name	Description
SERIALNUMBER	Certificate serial number
MAIL	Email address
UID or USERID	User identifier
CN	Common name
T	Title
OU	Organizational unit name
DC	Domain component
O	Organization name
STREET	Street or first line of address
L	Locality name
ST (or SP or S)	State or province name
PC	Postal code
C	Country
UNSTRUCTUREDNAME	Host name
UNSTRUCTUREDADDRESS	IP address
DNQ	Distinguished name qualifier

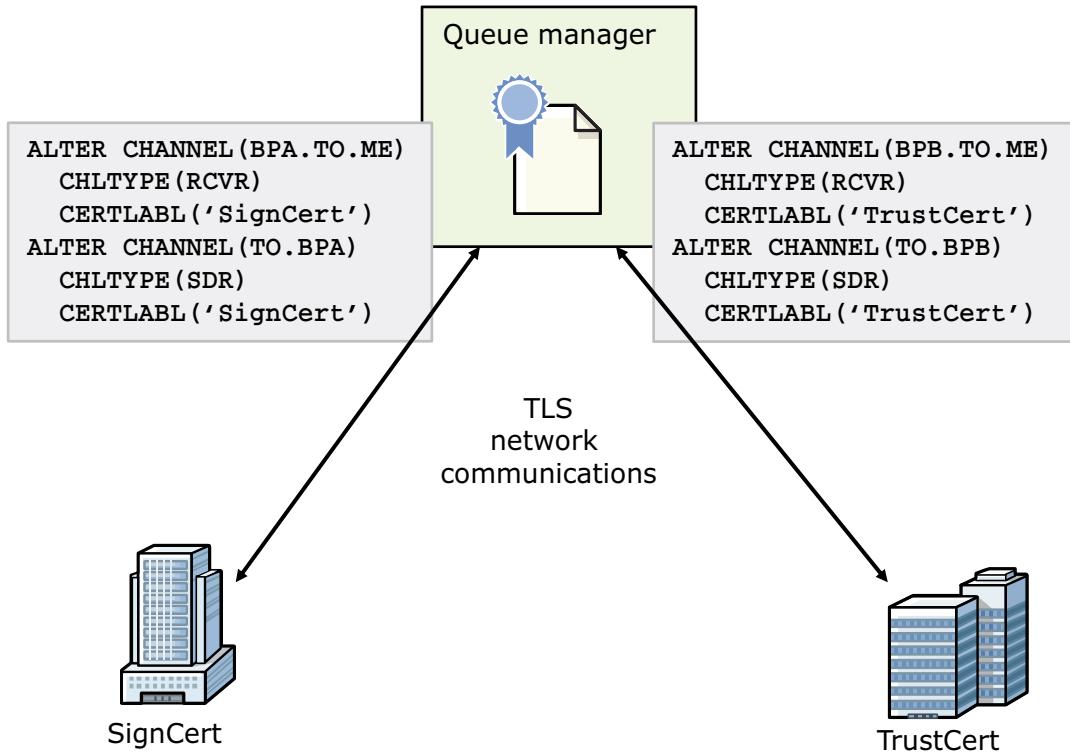
[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-34. Distinguished Name matching support in IBM MQ

This table summarizes the DN matching support in IBM MQ.

## Defining certificates for channels (1 of 2)



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-35. Defining certificates for channels (1 of 2)

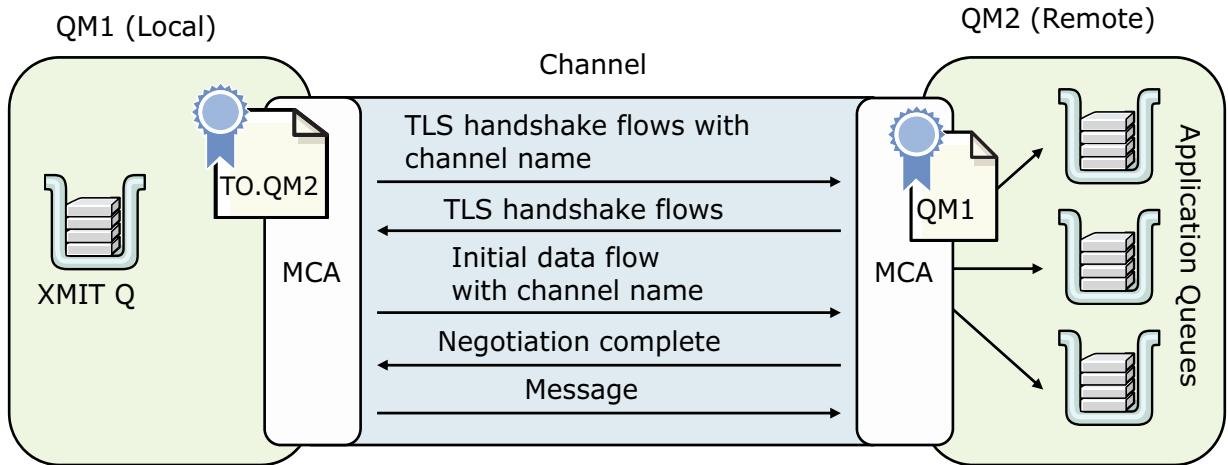
With IBM MQ, you can define certificates for each channel.

The feature provides more flexibility than a single queue manager certificate for connecting to partners with different CA requirements and connecting to servers with different levels of SSL support.

Channel certificates are configured by using the CERTLBL attribute on the channel definition.

- If the CERTLBL attribute is blank, the channel uses the queue manager certificate.
- If the CERTLBL attribute is set to a value, the channel uses the certificate that the label identifies. You can use a CERTLBL for TLS cipher specifications.

## Defining certificates for channels (2 of 2)



- **CERTLABEL** channel attribute identifies which personal certificate in the key repository is sent to the remote peer
- Both ends of the channel must be same version of IBM MQ

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-36. Defining certificates for channels (2 of 2)

Enhancements to the TLS protocol allow the provision of information as part of the TLS handshake. This information can be used to determine which certificate to use for this particular connection. This enhancement is known as Server Name Indication (SNI).

IBM MQ uses SNI to provide a channel name in the TLS handshake.

- The sender (or client) end of the channel puts the channel name into the SNI hint for the TLS handshake.
- The receiver (or server-connection) end of the channel retrieves the channel name from the SNI hint and selects the appropriate certificate that is based on that information.

## Channel attributes for TLS

- **CERTLBL** identifies certificate label for this channel to use
- **SSL\_CIPHER** specifies cipher specification that is used on the channel
- **SSLPEER** that is used by the local queue manager or client to filter the certificate that the peer queue manager or client at the other end of the channel sends
- **SSLCAUTH** defines whether IBM MQ requires a certificate from the TLS client
  - Can request whether the client end is required to provide a certificate for authentication
  - Valid for receiver, server-connection, cluster-receiver, server, and requester channels
  - For channels with **SSL\_CIPHER** specified

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-37. Channel attributes for TLS*

This figure lists the channel attributes for configuring TLS at the channel level.

The end of the channel that initiates the TLS connection is the client. The client always authenticates the server certificate, but might not necessarily send a certificate to the server to be authenticated. The SSLCAUTH attribute is used on the TLS server-end of the connection to indicate whether to expect a certificate for authentication from the TLS client, or if only the server-end has its certificate authenticated. The server-end certificate authentication can be useful for IBM MQ client connections, for lightweight queue managers, or for any initiating partner where authentication is not necessary.

The SSLCAUTH attribute can have the value of either OPTIONAL or REQUIRED. The default is REQUIRED.

## SSLPEER

- Used to check the distinguished name (DN) of the certificate from the peer queue manager or client at the other end of an IBM MQ channel
- Can use wildcards
- Multiple organizational units (OU)
  - Must be specified in descending hierarchical order (such as: OU=Big Unit, OU=Medium Unit, OU=Small Unit)
- Certificate DN mapping with expanded pattern matching and allow/deny capability

Examples:

```
SSLPEER ('CN="Thomas J Watson", O=IBM')
SSLPEER ('OU=Enablement*, O=IBM')
```

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-38. SSLPEER

The SSLPEER attribute identifies the DN of the certificate of the partner. This field can have wildcards to allow generic matching.

If the DN received from the peer does not match the SSLPEER value, the channel does not start.

SSLPEER is an optional attribute. If the field is left blank or is not present, then the peer DN is not checked when the channel is started.

For more information about the IBM MQ rules for SSLPEER values, see the IBM Knowledge Center for IBM MQ V9.

## SSLPEER mapping

- SSLPEER attribute of the channel filters connections that are based on DN of the connection requester
- An alternative way of restricting connections into channels by matching against the TLS subject DN is to use channel authentication records
  - Map certificate DNs to MCAUSER values or specify certificate DNs for which access is denied
  - Rules are hierarchical so it is possible to set “deny all” policy and then override with more specific access

Example:

```
SET CHLAUTH(*) TYPE(SSLPEERMAP) SSLPEER('CN=*')
  USERSRC(NOACCESS)
SET CHLAUTH(*) TYPE(SSLPEERMAP) SSLPEER('OU=ADMIN',O=IBM')
  USERSRC(CHANNEL)
SET CHLAUTH(*) TYPE(SSLPEERMAP) SSLPEER('CN="SWATTS",
  OU=ADMIN, O=IBM') USERSRC(NOACCESS)
```

*Figure 2-39. SSLPEER mapping*

An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records.

With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns to connect.

SSLPEER mapping is described in more detail in Unit 3.

## Displaying certificate details

- DISPLAY CHSTATUS shows partner certificate details
  - SSLPEER: The distinguished name of the partner
  - SSLCERTI: The certificate issuer's distinguished name of the partner

Examples:

- CA-signed:

```
SSLPEER (CN=MQ01, T=QMGR, O=IBM, L=Sydney, ST=NSW, C=AUST)
SSLCERTI (CN=IBM CA, O=IBM, L>New York, ST=NY, C=US)
```

- Self-signed:

```
SSLPEER (CN=MQ23, T=QMGR, O=IBM, L=Hursley, ST=Hampshire, C=UK)
SSLCERTI (CN=MQ23, T=QMGR, O=IBM, L=Hursley, ST=Hampshire, C=UK)
```

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-40. Displaying certificate details*

The DISPLAY CHSTATUS command provides some information about SSL partner certificate in the SSLPEER and SSLCERTI fields.

Do not confuse SSLPEER on DISPLAY CHSTATUS with SSLPEER on DISPLAY CHANNEL.

- On the channel definition, SSLPEER contains the filter that is matched against the partner distinguished name.
- On channel status, SSLPEER shows the actual DN of the partner certificate.

SSLCERTI contains the issuer DN from the partner certificate.

If SSLPEER and SSLCERTI are the same, the certificate is self-signed. These fields are also passed to the security exit so that more complicated decisions can be made based on this information.

## Checking certificate status with OCSP

- IBM MQ determines which OCSP responder to contact in one of two ways:
  - By using the AuthorityInfoAccess (AIA) certificate extension in the certificate
  - By using a URL that is specified in an authentication information object or specified by a client application
- To allow TLS channels to AIA extensions, ensure that the OCSP server that is named in them is available, is correctly configured, and is accessible over the network
- If IBM MQ receives an OCSP outcome of “Unknown”, its behavior depends on the setting of the **OCSP authentication** attribute
  - For queue managers, set the **OCSPAuthentication** in the **SSL** stanza of the `qm.ini` file to **WARN** or **OPTIONAL** (**REQUIRED** is the default)
  - For clients, set the **ClientRevocationChecks** attribute in the **SSL** stanza of the client configuration file `client.ini` to **OPTIONAL** or **DISABLED** (**REQUIRED** is the default)

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

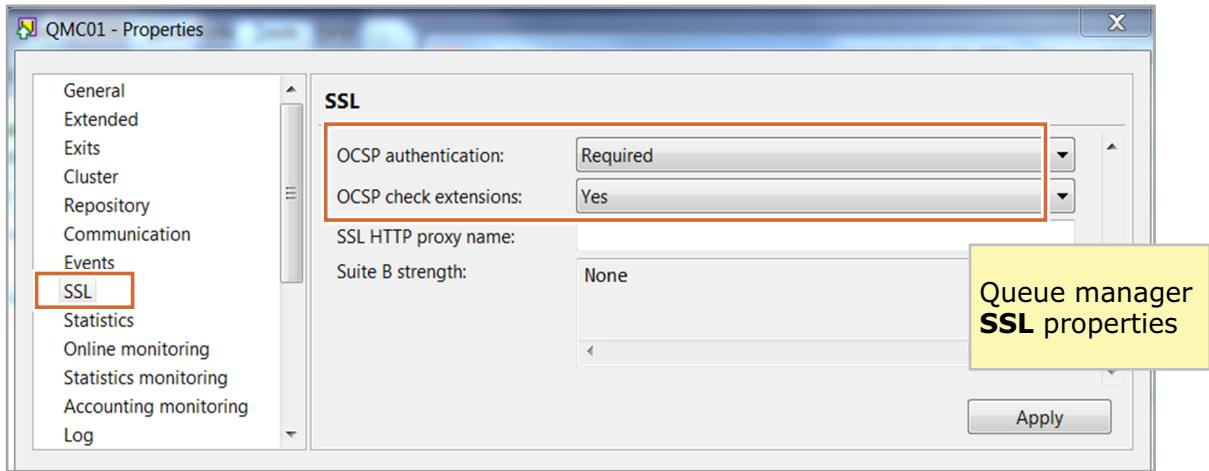
*Figure 2-41. Checking certificate status with OCSP*

IBM MQ supports OCSP for checking status. Based on configuration settings, IBM MQ determines the responder to use, and handles the response that is received.

IBM MQ determines which OCSP responder to contact by using the AIA certificate extension in the certificate or by using a URL that is specified in an authentication information objects. A URL specified in an authentication information object or by a client application takes priority over a URL in an AIA certificate extension.

The **OCSP authentication** attribute determines the behavior of an “unknown” outcome. On a queue manager, this attribute can be set to REQUIRED, WARN, or OPTIONAL. On a client, the attribute can be set to REQUIRED, OPTIONAL, or DISABLED. REQUIRED is the default value for both clients and queue managers.

## Setting OSCP properties in IBM MQ Explorer (1 of 2)



- **OCSP authentication** dictates outcome of a connection when the OCSP call returns “Unknown” response
  - **Required:** IBM MQ rejects the connection
  - **Optional:** Connection succeeds
  - **Warn:** Connection succeeds but IBM MQ writes an AMQ9717 message into the error log
- **OCSP check extensions** controls whether to use the OCSP server details in the AIA certificate extensions for digital revocation check

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-42. Setting OSCP properties in IBM MQ Explorer (1 of 2)

You can configure the OSCP properties in IBM MQ Explorer on the queue manager **SSL** properties page.

## Setting OSCP properties in IBM MQ Explorer (2 of 2)

To override the OSCP responder in the AIA certificate extension:

1. Under queue manager in the **MQ Explorer – Navigator** view, click **Authentication Information > New > OCSP Authentication Information**
2. Enter a name
3. Specify the OCSP responder URL

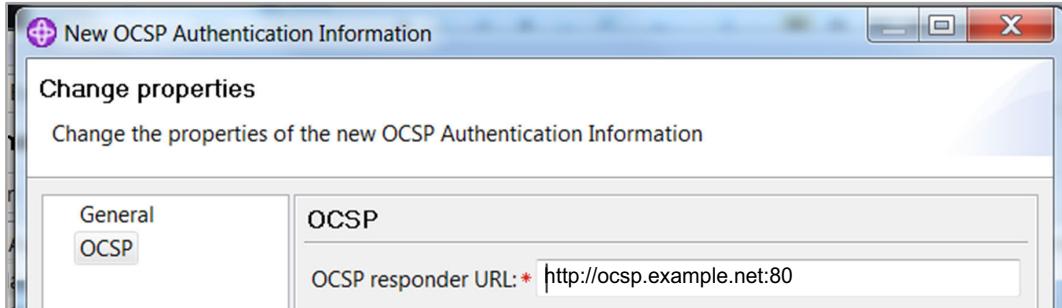


Figure 2-43. Setting OSCP properties in IBM MQ Explorer (2 of 2)

The URL for OSCP responder that is set in the AIA can be overridden by using IBM MQ Explorer. This figure lists the steps for overriding the OSCP responder URL.

## Accessing CRLs and ARLs

- Can be stored in and accessed from LDAP servers

1. Define on AUTHINFO objects

```
DEFINE AUTHINFO(LDAP1)
AUTHTYPE(CRLLDAP)
CONNNAME('ldap(389)')
LDAPUSER('cn=user')
LDAPPWD(...)
```

LDAP Connname  
LDAP Username  
LDAP Password



2. Put AUTHINFO objects into a namelist

```
DEFINE NL(LDAPNL) NAMES(LDAP1, LDAP2, ...)
```

LDAP Server

3. Associate namelist with the queue manager

```
ALTER QMGR SSLCRLNL(LDAPNL)
```



- Clients

- Client channel table contains the LDAP definitions that are present when the table is copied
- Details of the LDAP servers can be held in the Active Directory (Windows only)

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-44. Accessing CRLs and ARLs

As an alternative to OCSP, you can verify certificates by using CRLs and ARLs.

### Queue managers

For queue managers, CRLs and ARLs can be stored in and accessed from LDAP servers. The LDAP CRL and ARL servers are generally defined to be publicly readable.

A few parameters must be specified to access an LDAP server that contains CRLs and ARLs. These parameters are the DNS name or IP address of the LDAP server with an optional TCP/IP port number. The DN of the entry that binds to the directory and the password that is associated with the distinguished name can also be included as optional parameters.

These parameters are defined on a queue manager AUTHINFO object. Several of these objects might be needed to ensure redundancy. Redundancy is provided so that, for example, if the first LDAP server to which a queue manager connects is down, it can connect to another server that can supply the same information. On distributed operating systems, up to 10 of these AUTHINFO objects can be supplied for CRL and ARL checking.

The list of AUTHINFO objects is named in a namelist and this namelist is specified on the SSLCRLNL queue manager attribute by using the ALTER QMGR command.

## Clients

The client channel definition table LDAP CRL and ARL server information does not have to match the definitions that are current on the queue manager system when the channel runs.

The MQCONN call provides a structure for IBM MQ authority information records that specify the LDAP CRL and ARL server information.

On Windows, records that specify the access details of the LDAP servers that hold CRL and ARL information are created in the Active Directory by using the `setmqcrl` command. Details of the LDAP server can be held in Active Directory, but the Active Directory cannot be used for CRLs or ARLs.

The CLNTCONN attributes for LDAP servers do not have to match the CLNTCONN attributes of the corresponding SVRCONN channel.

## Accessing CRLs and ARLs by using IBM MQ Explorer

1. In the **MQ Explorer – Navigator** view, right-click **Authentication Information** and click **New > CRL LDAP Authentication Information**
  - A. Enter a name for the CRL LDAP object
  - B. Enter the LDAP server name as either the network name or the IP address
  - C. If the server requires login details, provide a user ID and password
2. In the **MQ Explorer – Navigator** view, right-click **Namelists** and click **New > Namelist**
  - A. Enter a name for the namelist
  - B. Add the name of the CRL LDAP object (from Step 1A) to the list
3. On the queue manager **SSL** properties page
  - A. Enable **Check certificates received by this queue manager for revocation Lists**
  - B. Type the name of the namelist (from Step 2A) in the **Revocation Namelist** field

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-45. Accessing CRLs and ARLs by using IBM MQ Explorer

You can identify the LDAP CRL and ARL information by using MQSC or IBM MQ Explorer. This figure lists the steps for configuring the access to CRLs and ARLs by using IBM MQ Explorer.

## Possible authentication failures

- A certificate was found in a CRL or ARL
- An OCSP responder identified a certificate as “revoked” or “unknown”
- A matching CA root certificate does not exist or the certificate chain is incomplete

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-46. Possible authentication failures

Authentication failures during the TLS handshake can occur for a number of reasons.

- A CA can revoke a certificate that is no longer trusted by publishing it in a CRL or ARL.
- An OCSP responder can return a response of “revoked”, which indicates that a certificate is no longer valid. A response of “unknown” indicates that OCSP does not have revocation data for that certificate.
- Each digital certificate from a CA also provides a root certificate that contains the public key for the CA. The CA signs the root certificates. If the key repository on the computer that is authenticating the certificate does not contain a valid root certificate for the CA that sent the incoming user certificate, authentication fails.

Other possible reasons for authentication errors include the following reasons:

- A certificate expired or is not active
- A certificate is corrupted
- The TLS client does not have a certificate

## Using secret key reset

- IBM MQ supports the resetting of secret keys on queue managers and clients when a specified number of encrypted bytes of data flowed across the channel
- If channel heartbeats are enabled, the secret key is reset before data is sent or received following a channel heartbeat
- Set a specified number of bytes  

```
ALTER QMGR SSLRKEYC(999 999 999)
```
- Display channel status shows:
  - Number of times the key was reset
  - Time and date of the last reset

```
DISPLAY CHSTATUS(*) SSLRKEYS SSLKEYDA SSLKEYTI
```

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-47. Using secret key reset

A TLS channel periodically renegotiates its secret key if secret key reset is enabled. This renegotiation takes place after a specified amount of data is moved across the channel. It also takes place before data is sent across a channel that was idle for a time and sent heartbeat flows.

For a queue manager, use the command ALTER QMGR with the parameter SSLRKEYC to set the values that are used during key renegotiation.

Fields in the channel status show when the last reset was completed and how many resets were completed for the life of the channel instance.

By default, MQI clients do not renegotiate the secret key. You can make an MQI client renegotiate the key in any of three ways. In the following list, the methods are shown in order of priority. If you specify multiple values, the highest priority value is used.

1. Use the KeyResetCount field in the MQSCO structure on an MQCONN call.
2. Use the environment variable MQSSLRESET.
3. Set the SSLKeyResetCount attribute in the MQI client configuration file.

## Refreshing TLS on IBM MQ

- Cached views of key repository
- Changes require refresh of cached views
  - New certificates
  - Deleted certificates
  - Updated certificates that replace expiring certificates
- MQSC command **REFRESH SECURITY TYPE(SSL)**
  - Stops all TLS channels
  - New cached views of the key repository are created
  - Restarts all sender TLS channels
  - Restarts receiver channels when partner tries again
  - Updates new key repository and LDAP CRL and ARL locations

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-48. Refreshing TLS on IBM MQ*

The TLS environment that is set up to run TLS channels in a channel process has a cached view of the key repository that is made at initialization time.

If you change your key repository for any reason, you must refresh the cached view so that the TLS channels use start the new certificates.

To refresh this cached view of the TLS environment without disrupting any channels that are not enabled for TLS, use the REFRESH SECURITY TYPE(SSL) command. The REFRESH command stops all the TLS channels on the queue manager. New cached views of the key repository are made and all the sending type channels are started again. Receiving type channels are restarted when the partner tries to connect again.

When you enter the REFRESH SECURITY TYPE(SSL) MQSC command, all running TLS channels are stopped and restarted. Sometimes TLS channels can take a long time to shut down. IBM MQ sets a time limit of 10 minutes for a TLS refresh to complete.

Also, you can use this command to recognize other changes, such as new key repository locations, or new LDAP CRL and ARL locations.

## Security administration tasks

- Creating certificates and certificate requests
  - Different tools are used on various operating systems
- Storing certificates, public keys, and private keys in key database files
- Managing binary certificates cross-platform
  - DER, CER, BER encodings, such as PKCS #7 DER encoded X.509 certificate
  - PKCS #12 DER encoded X.509 certificate (password protected)
- Managing text certificates cross-platform
  - Must be transferred with text conversion, for example, ASCII to EBCDIC
  - Privacy Enhanced Mail (PEM) encoded X.509 certificate
  - Base64 encoded certificate

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-49. Security administration tasks

Security implementation requires some administration.

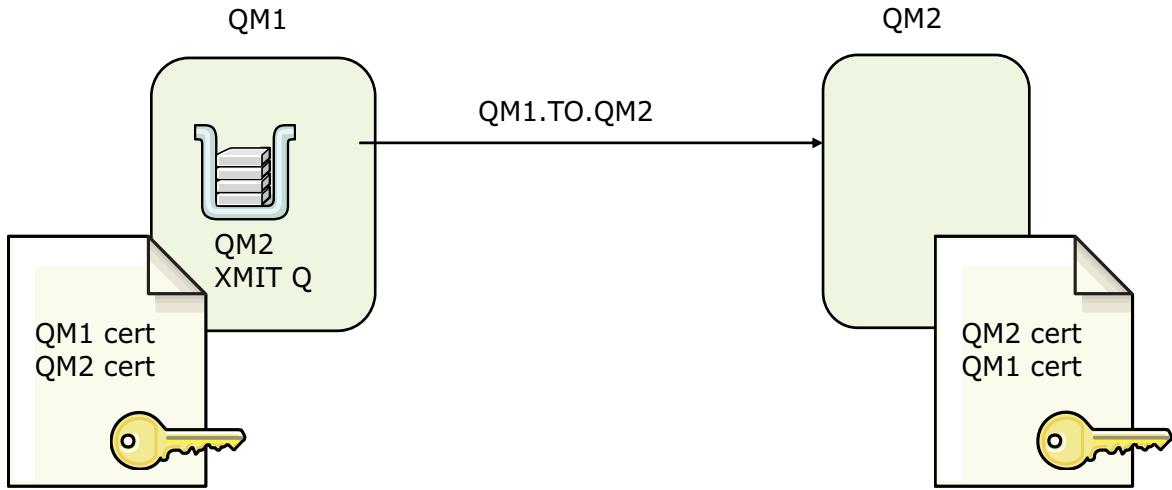
For example, security implementation requires the administration of key repositories and digital certificates. Certificate authority certificates are stored in the key repository so that they can be used to validate certificates that are received from the partner system.

Certificates can be created on the system. They can also come from outside the system, in which case they must be imported onto the system. Several standard formats exist. Some of these formats are binary and must be transported in their exact binary format. In contrast, text formats must be transported as text. If they are transported between an ASCII and EBCDIC system, the data must be converted from ASCII to EBCDIC.

PKCS #12 files contain a personal certificate and optionally, its private key and CA certificates for its signing CAs.

## TLS implementation scenario (1 of 3)

- Two queue managers, QM1 and QM2, need to communicate securely and require mutual authentication
  - Channels are configured and verified without security
  - Configure and test the secure communication by using self-signed certificates



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-50. TLS implementation scenario (1 of 3)

This scenario summarizes the TLS administration tasks by using a TLS implementation example. In this scenario, two queue managers need to communicate securely and require mutual authentication. Self-signed certificates are used to test the implementation of the security requirements.

In this scenario, the key repository for queue manager QM1 contains the certificate for QM1 and the public certificate from QM2. The key repository for queue manager QM2 contains the certificate for QM2 and the public certificate from QM1.

## TLS implementation scenario (2 of 3)

1. Prepare the key repository on each queue manager
2. Create a self-signed certificate for each queue manager
3. Extract a copy of each certificate
4. Transfer the public part of the QM1 certificate to QM2 and the public part of the QM2 certificate to QM1 by using a utility such as FTP
5. Add the partner certificate to the key repository for each queue manager
6. Stop the sender channel
7. On QM1, modify the sender channel for TLS and refresh security

```
ALTER CHANNEL (QM1.TO.QM2) CHLTYPE (SDR) +
SSLCIPH (TLS_RSA_WITH_AES_256_GCM_SHA384) +
DESCR ('Sender using TLS')
REFRESH SECURITY TYPE (SSL)
```

Figure 2-51. TLS implementation scenario (2 of 3)

This figure lists the first seven steps for implementing TLS for this scenario. The detailed information for completing each of these steps is provided in previous pages on this unit.

## TLS implementation scenario (3 of 3)

8. On QM2, modify the receiver channel and refresh security

```
ALTER CHANNEL (QM1.TO.QM2) CHLTYPE (RCVR) +
SSL_CIPHER(TLS_RSA_WITH_AES_256_GCM_SHA384) +
SSLCAUTH (REQUIRED) +
DESCR ('Receiver using TLS')
REFRESH SECURITY TYPE(SSL)
```

The channel uses the same cipher specification that is specified on QM1

9. Start the channel

This figure lists the final two steps for implementing security for this scenario.

## Security problems and solutions

- Eavesdropping
  - Symmetric key cryptography
- Tampering
  - Hash function

- Impersonation
  - Digital certificate
  - Asymmetric keys

- CRLs, ARLs, and OCSP



### IBM MQ solution

- `SSLCIPH()`
- `SSLRKEYC()`

- `SSLKEYR()`
- `SSLPEER()`
- `SSLCAUTH()`

- `SSLCRILNL()`

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

Figure 2-53. Security problems and solutions

Three main security problems, eavesdropping, tampering, and impersonation were described in this unit.

- For eavesdropping, you can use symmetric key cryptography.
- For tampering, you can use the hash function.
- For impersonation, you can use digital certificates, asymmetric keys, and certificate revocation lists.

IBM MQ uses all of these techniques to provide solutions to security problems. You can specify the symmetric key cryptography algorithm and the hash function to use by providing IBM MQ with a cipher specification.

Digital certificates and public keys are found in a key repository, which can be specified to IBM MQ. You can check to make sure that you are talking to the partner you expect. You can choose to authenticate both ends of the connection or only the TLS server end of the connection. Also, you can use CRLs, ARLs, or an OCSP responder to verify the certificate.

## IBM MQ Advanced Message Security

- Provides a high level of protection for sensitive data that flows through the IBM MQ network, while not impacting the end applications
- Secures sensitive or high-value transactions that IBM MQ processes
- Detects and removes rogue or unauthorized messages before a receiving application processes the messages
- Verifies that messages were not modified while in transit from queue to queue
- Protects the data not only as it flows across the network but also when it is put on a queue
- Secures existing proprietary and customer-written applications for IBM MQ
- IBM MQ Advanced license includes entitlement for IBM MQ Advanced Message Security

[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-54. IBM MQ Advanced Message Security*

The focus of this unit is channel-level security. For message-level security, you can use IBM MQ Advanced Message Security to provide the following functions:

- Protect message data while the message is in the queue and as it flows across the network.
- Attach digital signatures to individual messages so that you can verify whether someone tampered with them. You can also identify and trace messages to their point of origin.
- Encrypt messages to shield them from disclosure to unauthorized parties and centrally define and enforce security policies by using a web-based interface.
- Provide security-specific, fine-grained auditing records to document adherence to security policy, thus protecting highly sensitive transactions records.
- Help accelerate new application deployment by reducing development time and costs by providing application level security without requiring complex security coding.

IBM MQ Advanced Message Security is a separately licensed component of IBM MQ. Entitlement to IBM MQ Advanced Message Security is included with the IBM MQ Advanced license.

## Unit summary

- Describe the certificate infrastructure that is supported in IBM MQ  
Manage certificates with IBM Key Management
- Manage certificates with IBM MQ Key Management
- Describe cipher specifications and their support in IBM MQ
- Use certificate revocation lists or Online Certificate Status Protocol (OCSP) to validate currency of certificates
- Use TLS to secure IBM MQ channel communications

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

*Figure 2-55. Unit summary*

## Review questions

1. SSLPEER is an optional parameter that:
  - A. Defines whether the channel needs to receive and authenticate TLS certificates from a TLS client.
  - B. Defines a single cipher specification for a TLS connection.
  - C. Is used to check the DN of the certificate from the peer queue manager or client at the other end of an IBM MQ channel.
2. On UNIX and Linux the digital certificates have labels that are used to:
  - A. Find the correct certificate.
  - B. Allow indexing.
  - C. Provide a description of the certificate.
  - D. Associate a certificate with a queue manager.
3. True or False: IBM Key Management can be used to sign personal certificates as a CA.



Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-56. Review questions

Write your answers here:

- 1.
- 2.
3. .

## Review answers

1. SSLPEER is an optional parameter that:
  - A. Defines whether the channel needs to receive and authenticate TLS certificates from a TLS client.
  - B. Defines a single cipher specification for a TLS connection.
  - C. Is used to check the DN of the certificate from the peer queue manager or client at the other end of an IBM MQ channel.

The answer is C.
  
2. On UNIX and Linux the digital certificates have labels that are used to:
  - A. Find the correct certificate.
  - B. Allow indexing.
  - C. Provide a description of the certificate.
  - D. Associate a certificate with a queue manager.

The answer is D.
  
3. True or False: IBM Key Management can be used to sign personal certificates as a CA.  

The answer is False. IBM Key Management can be used to create keys and manage SSL certificates. It cannot act as a CA.

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

Figure 2-57. Review answers



## Exercise: Securing channels with TLS

Securing IBM MQ channels with TLS

© Copyright IBM Corporation 2017

*Figure 2-58. Exercise: Securing channels with TLS*

In this exercise, you define and start SSL channels between IBM MQ queue managers, and between an IBM MQ client and an IBM MQ server.

## Exercise objectives

- Use the certificate management utility IBM Key Management to create a certificate request
- Secure channels by using TLS on the channel



[Securing IBM MQ channels with TLS](#)

© Copyright IBM Corporation 2017

*Figure 2-59. Exercise objectives*

For detailed instructions, see the Course Exercises Guide.

---

# Unit 3. Authenticating channels and connections

## Estimated time

02:00

## Overview

In this unit, you learn how to use channel authentication to control the access that is granted to connecting systems at a channel level. You learn how to modify the queue manager to use the local operating system or an LDAP server to authenticate user IDs and passwords of clients or applications that are requesting access to IBM MQ resources. The unit also describes channel exit programs and administration.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Determine the current level of authentication that is enabled on a queue manager and a connection
- Add authentication to a channel
- Add authentication to a connection
- Identify and fix channel authentication and connection authentication problems
- Implement a channel exit program for securing messaging channels

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-1. Unit objectives*

## 3.1. Authentication and authorization overview

## Authentication

- Establishing the identity of the user
- Basic authentication: Asking for a user ID and password
- Multifactor authentication can be a combination of:
  - Something that you know, such as a password
  - Something that you have, such as a certificate or hardware token
  - Something that you are, such as a fingerprint or retinal scan
- Provides some level of assurance that the identity that is presented is genuine
- Complexity of authentication typically depends on the importance of keeping data protected

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-2. Authentication*

Authentication is the process of establishing the identity of the user. The user in this case might be an application or another queue manager.

Basic authentication is a user ID and password. Extra authentication provides some level of assurance that the identity that is presented is genuine.

## Authorization

- Control access to resources on a per-ID, per-role, or per-profile basis
- Authorization response is an absolute concept where either a "Yes" or "No" answer is returned for any access request
- A resource can support many different functions such as CONNECT, OPEN, and CLOSE, and many different options such as INPUT, OUTPUT, BROWSE, and INQUIRE
- Answers the question “Does user X have access type Y to resource Z?”

Authorization is the control of access to resources based on an ID, user role, or user profile.

## Identification and authentication in IBM MQ

- You can implement identification and authentication by using:
  - Message context information
  - *Mutual authentication* by the MCA at each end of a channel when a message channel starts
  - Connection authentication
- IBM MQ requires a user ID to make authorization decisions
- IBM MQ needs to understand where the user ID comes from
- What leads to security exposure?
  - Not understanding how IBM MQ determines the user ID
  - Thinking that you understand it
  - Incorrect settings in your definitions

Figure 3-4. Identification and authentication in IBM MQ

In IBM MQ, you can implement identification and authentication by using message context information. You can also implement mutual authentication by the MCA at each end of a channel when a message channel starts.

## Approach to planning security

- Secure administrative access
  - If no meaningful authentication exists, then security can be circumvented
- Any more advanced security hardening depends on securing administrative access
  - If administrative access is not locked down, no accountability exists and security controls can easily be bypassed
- Resulting security model provides three types of access:
  - Administrator
  - Remote queue manager (for queue manager to queue manager access)
  - Client channel user (client or application)

Figure 3-5. Approach to planning security

When planning or reviewing IBM MQ security, you first need to secure administrative access. Securing administrative access means making sure that nobody can programmatically assert a user ID with IBM MQ administration privileges. Any security plan must consider all types of access to data and IBM MQ objects.

## Secure connections in IBM MQ Explorer

- **IBM MQ Explorer Preferences**
  - Default values for client-connection security exits, TLS default values, and user ID and password
  - Passwords that IBM MQ Explorer uses
- **Add Remote Queue Manager** wizard for each remote queue manager connection
- Channel authentication precedence mapping

Authenticating channels and connections

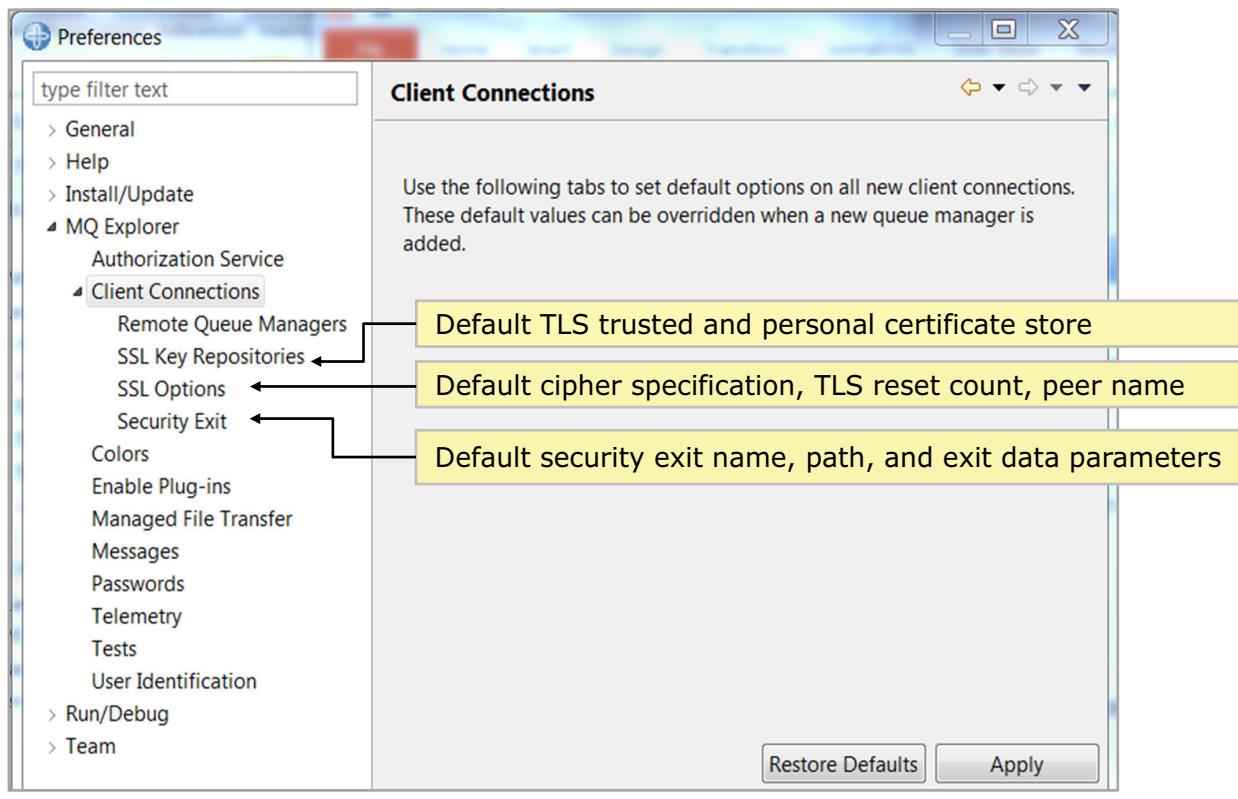
© Copyright IBM Corporation 2017

*Figure 3-6. Secure connections in IBM MQ Explorer*

The security provisions in IBM MQ include securing channels with TLS and controlling access to IBM MQ objects. You can manage both TLS security and object authorities in IBM MQ Explorer.

IBM MQ secures the connection to a queue manager with a user ID and password. You can also configure different security options and channel authentication (CHLAUTH) mapping for each queue manager.

## Setting client connection default values



Authenticating channels and connections

© Copyright IBM Corporation 2017

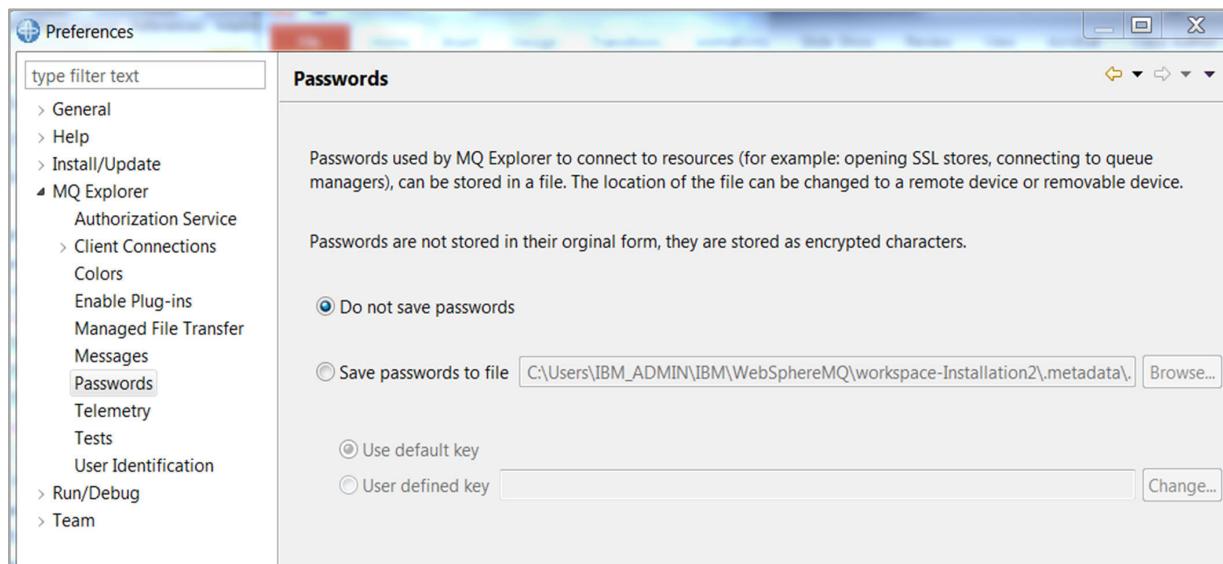
Figure 3-7. Setting client connection default values

To set the default values for client connections in IBM MQ Explorer, click **Window > Preferences** from main menu. Alternatively, you can right-click **IBM MQ** in the **MQ Explorer - Navigator** view and then click **Preferences**.

In the left pane of **Preferences** window, expand **MQ Explorer** and then expand **Client Connections**.



## Setting passwords that IBM MQ Explorer uses



Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-8. Setting passwords that IBM MQ Explorer uses*

Passwords that IBM MQ Explorer uses to connect to resources, such as opening TLS stores or connecting to queue managers, can be stored in a file.

To set default preferences for passwords in IBM MQ Explorer, click **Window > Preferences**. In the left pane of the **Preferences** window, expand **MQ Explorer** and then click **Passwords**.

When using the option **Use default key**, the password store opens automatically when it is needed. The option **User defined key** means that the password store must be accessed with a specific password. In this case, the password window is always displayed when the password store needs to be opened for the first time after IBM MQ Explorer is started.

## Security for remote queue manager connection

- **Add Remote Queue Manager** wizard security options
  - Security exit name, path, and exit data
  - User ID and password for client connections
  - TLS trusted certificate store, and personal certificate store
  - Cipher specification, TLS reset count, and peer name

Authenticating channels and connections

© Copyright IBM Corporation 2017

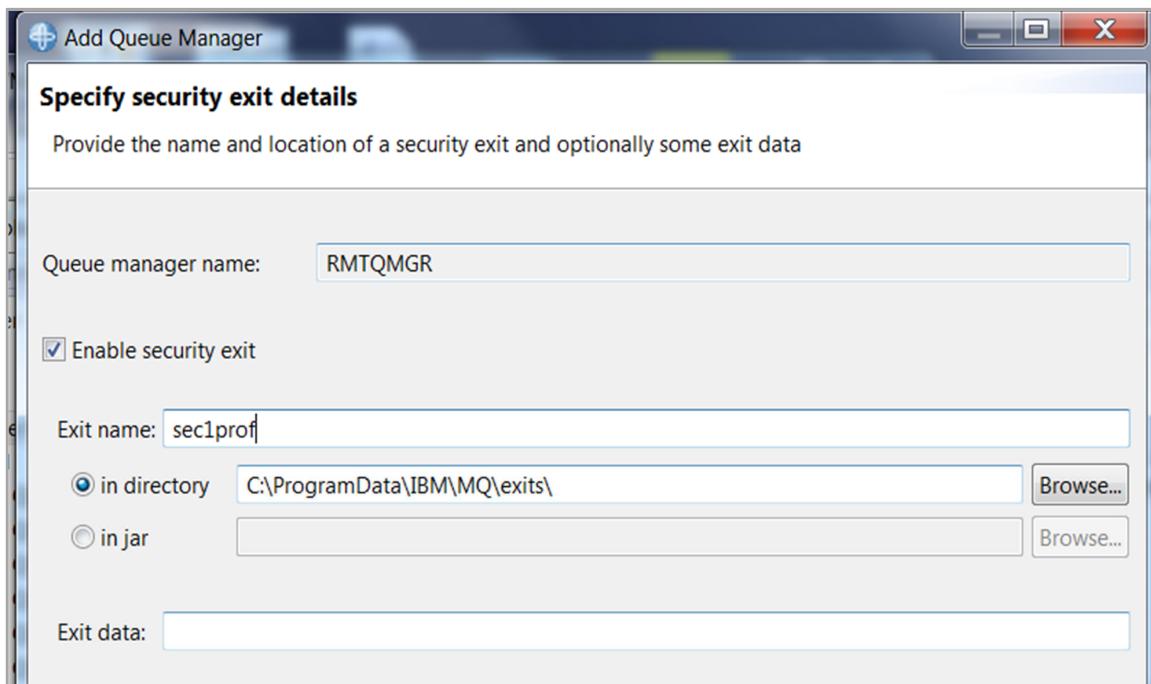
*Figure 3-9. Security for remote queue manager connection*

The **Add Remote Queue Manager** wizard in IBM MQ Explorer allows you to define specific security connection information for each queue manager.

In the wizard, you define the basic connection information such as queue manager name, TCP/IP address, and port. You can also define security information for the remote queue manager connection.



## Security settings for remote queue managers (1 of 4)



Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-10. Security settings for remote queue managers (1 of 4)*

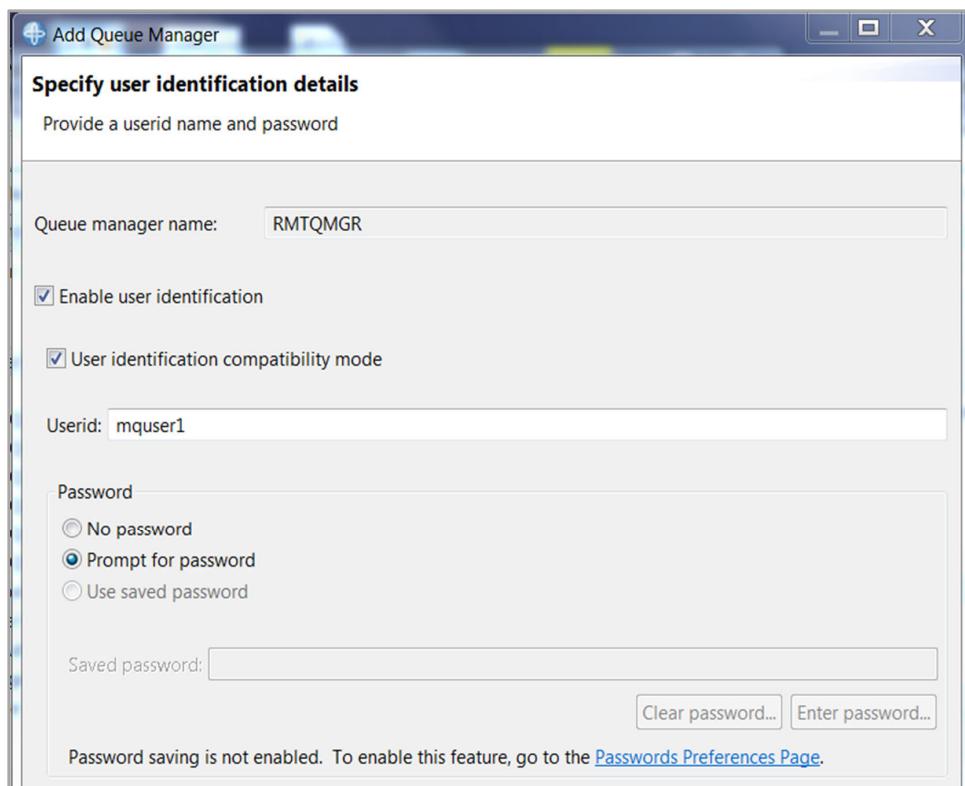
The first two pages in the **Add Remote Queue Manager** wizard define the basic connection information. The next four pages define the security specifications.

To define a connection to a remote queue manager, right-click the **Queue Managers** folder in the **MQ Explorer - Navigator** view and then click **Add Remote Queue Manager**.

The main purpose of this wizard is to connect to a remote queue manager that is enabled for remote administration. The security pages that are displayed in the figure and the next three pages show four of the six pages that are configured during the connection.

The security exit details page that is shown in the figure, is used to enable a security exit. Select **Enable security exit** to provide the name and location of a security exit. You can also choose to add some exit data.

## Security settings for remote queue managers (2 of 4)



Authenticating channels and connections

© Copyright IBM Corporation 2017

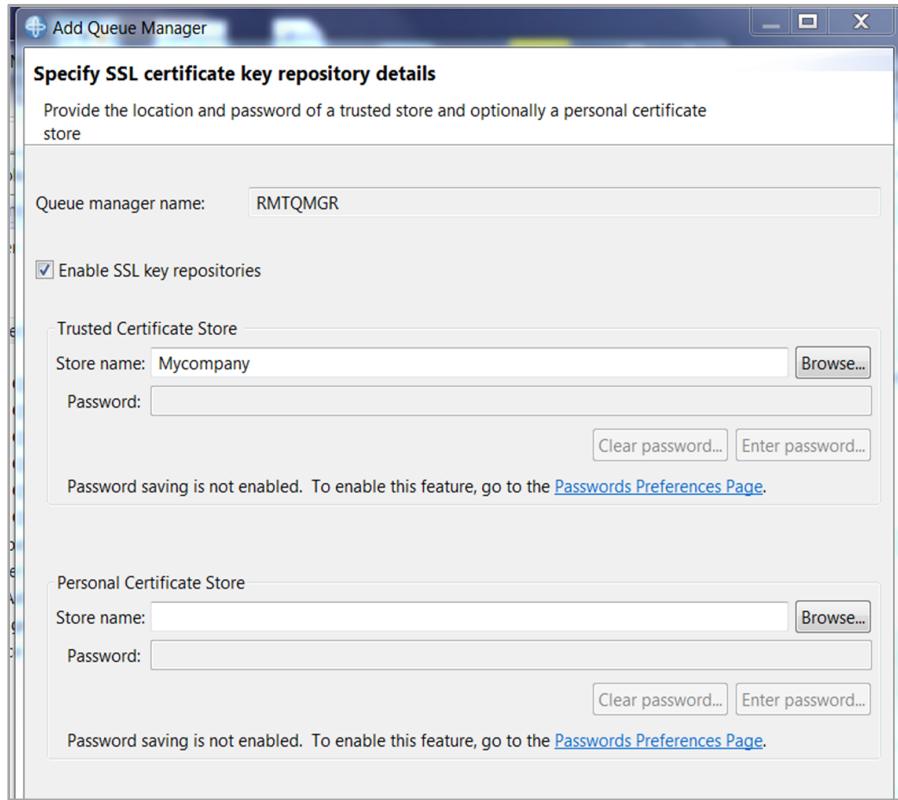
Figure 3-11. Security settings for remote queue managers (2 of 4)

In the next page in the **Add Remote Queue Manager** wizard, you can enable user identification.

Select **Enable user identification** to provide a user ID name and password.



## Security settings for remote queue managers (3 of 4)



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-12. Security settings for remote queue managers (3 of 4)

On the next page in the **Add Remote Queue Manager** wizard, you can provide TLS certificate key repository details.

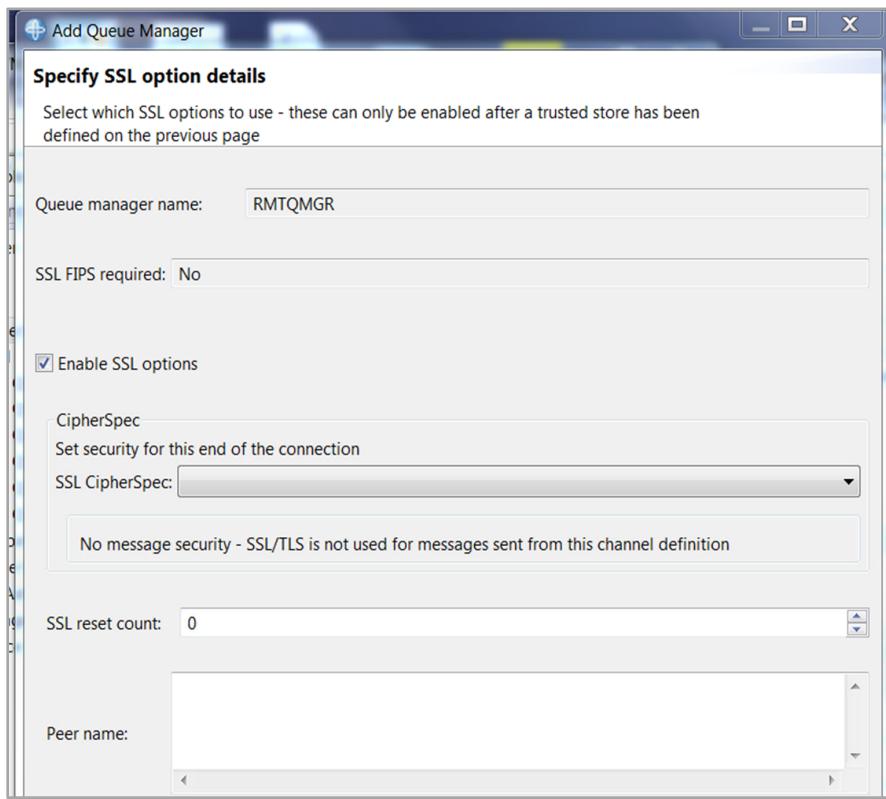
Select **Enable SSL key repositories** to provide the location and password of a trusted store. You can also choose to add the details of a personal certificate store. TLS security configuration was described in detail in a previous unit.



### Note

In this version of IBM MQ, all SSL cipher specifications are deprecated so you must use TLS for security. You enable and configure TLS by using the SSL options in the IBM MQ Explorer and commands options in MQSC.

## Security settings for remote queue managers (4 of 4)



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-13. Security settings for remote queue managers (4 of 4)

On the next page in the **Add Remote Queue Manager** wizard, you can enable the TLS options.



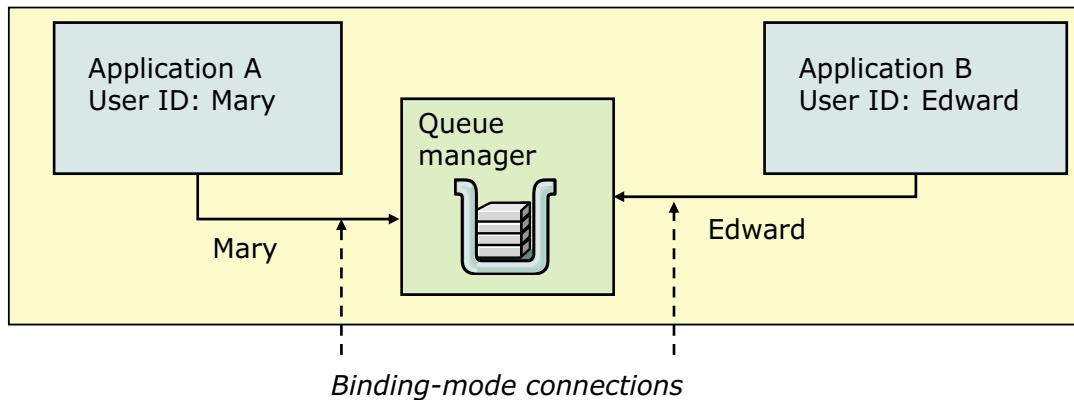
### Note

TLS options can be enabled only after a trusted store is defined in the wizard.

Select the **Enable SSL options** check box to identify the TLS cipher specification and configure the other TLS options.

## User ID when using bindings mode

- Applications connecting to queue manager in bindings mode
  - Application process user ID is assigned through an operating system process, such as user logon
  - User ID flows to the queue manager
- For local bindings-mode connections, the IBM MQ security model default is to “deny all”



Authenticating channels and connections

© Copyright IBM Corporation 2017

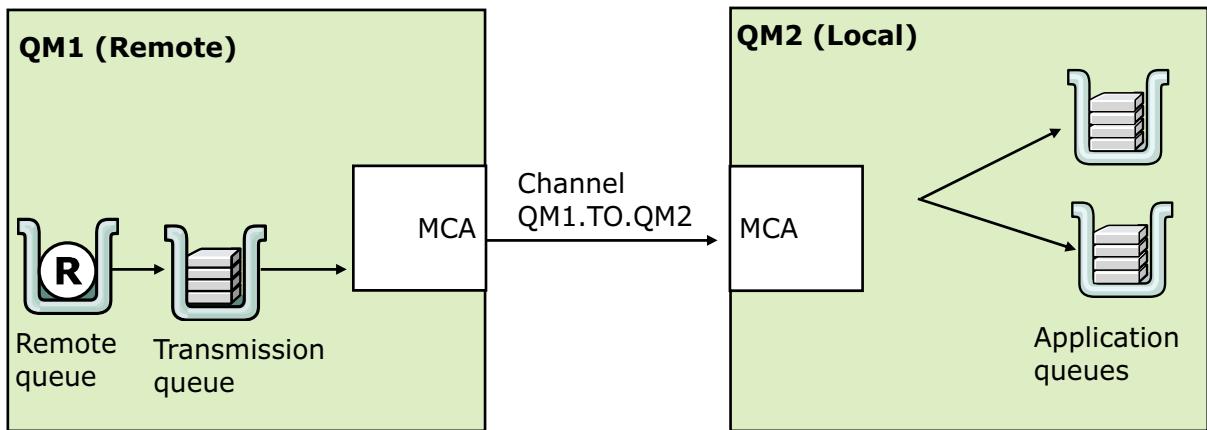
Figure 3-14. User ID when using bindings mode

When an application runs on the same server as the queue manager, it can connect in bindings mode. Bindings mode means that the application process communicates with the queue manager process by using memory. No channels are used when bindings mode is used. The application process has a user ID associated with it. When it connects to the queue manager, the user ID under which the application process runs is also the user ID that the queue manager uses to make authorization decisions.

In this example, the operating system authenticated the user ID. The queue manager trusts the identity that the operating system presents.

## Queue manager to queue manager

- MCA at each end of the channel between queue managers
  - User ID used is that of the MCA
  - By default, MCA puts messages with full administrative authority
  - Process can be overridden by setting the **MCAUSER** attribute on the channel



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-15. Queue manager to queue manager

When the local queue manager receives messages from other queue managers, the user ID of the MCA is used to make authorization decisions, such as whether the message can be written to a queue. If no MCAUSER value is set in the receiving channel definition, then the MCA puts messages with administrative authority.

These rules apply only if PUTAUT(CTX) is not set on the channel. If PUTAUT(CTX) is set on the channel, the user ID from the context information associated with the message is used as an alternative user ID. IBM MQ uses this user ID to make authorization decisions.

Use of PUTAUT(CTX) implies that messages that arrive over that channel can obtain full administrative authority.

## Why using TLS can lead to a false sense of security

- IBM MQ can use TLS over all channel types
- TLS provides assurance that someone who starts an MQI channel possesses a valid certificate
- TLS optionally encrypts the data so that data cannot be viewed in transit
- TLS starts and ends with the MCA
  - Authentication that TLS provides does not extend to the API calls
  - After the connection is established, everything outside the TLS exchange (the connection request, the API calls) works exactly as it does without TLS
- TLS does not affect authorization behavior

*Figure 3-16. Why using TLS can lead to a false sense of security*

IBM MQ allows you to specify the use of TLS over channels. Sometimes administrators might think their IBM MQ environment is secure because they enabled the use of TLS on all channels, which leads to a false sense of security. TLS, when properly configured, limits connections to legitimate remote nodes. For MCA channels (such as RQSTR, RCVR, CLUSRCVR), TLS does not change the fact that they run with full administrative authority by default.

TLS does not limit the queues onto which an MCA channel can put messages.

TLS does not prevent the legitimate user of an MQI (SVRCONN) channel from asserting any arbitrary user ID. Therefore, TLS does not establish a context that extends to the API on which the OAM can make authorization decisions.

The use of TLS begins and ends with the connection.

The MCA does not directly use any information that is available from the context of the TLS session.

TLS does not affect anything at the API layer. Therefore, TLS allows you to authenticate the remote node. However, you still do not have a trusted identity for API calls.

## IBM MQ Advanced Message Security

- Provides the flexibility to add application-level data protection and remote security policy administration, which includes:
  - Signing each message with a unique private key that is associated with the sending application
  - Encrypting individual messages under unique keys, thus helping to remove the threat of compromising the encryption key through repetitive use
- Provides the remote administration of security policies on queue managers and on individual queues
- Protects message data while the message is in queue and as it flows across the network
- Attaches digital signatures to individual messages so you can:
  - Verify whether anyone tampers with them
  - Identify and trace messages to their point of origin

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-17. IBM MQ Advanced Message Security

IBM MQ Advanced Message Security extends IBM MQ security.

IBM MQ Advanced Message Security supports the following features for both IBM MQ servers and clients:

- Application-level, end-to-end data protection for your point-to-point messaging infrastructure, by using either encryption, or digital signing of messages
- Comprehensive security without writing complex security code or modifying or recompiling existing applications
- Use of Public Key Infrastructure (PKI) technology to provide authentication, authorization, confidentiality, and data integrity services for messages
- Administration of security policies for mainframe and distributed servers

## 3.2. Object authorizations review

## Object authority manager (OAM)

- OAM is started by default when the queue manager starts
- When an MQI request is made or a command is entered, the OAM checks the authorization of the entity that is associated with the operation to see whether it can:
  - Complete the requested operation
  - Access the specified queue manager resources
- IBM MQ Explorer, `setmqaut` commands, and MQSC SET AUTHREC commands define authorization rules in the OAM
- Specify authority profiles in the `setmqaut` command:
  - SETALL, ALLMQI, ALLADMIN, ALL, and PASSALL
  - These authority profiles confer administrative privileges, which should not be granted routinely
  - Can use discrete privileges, such as put, get, and inquire

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-18. Object authority manager (OAM)*

The authorization service component that is supplied with the IBM MQ products is called the Object Authority Manager (OAM).

By default, the OAM is active and works with the control commands `dsprmqaut` (display authority), `dmpmqaut` (memory dump authority), and `setmqaut` (set or reset authority).

The OAM works with the entity of a principal or group:

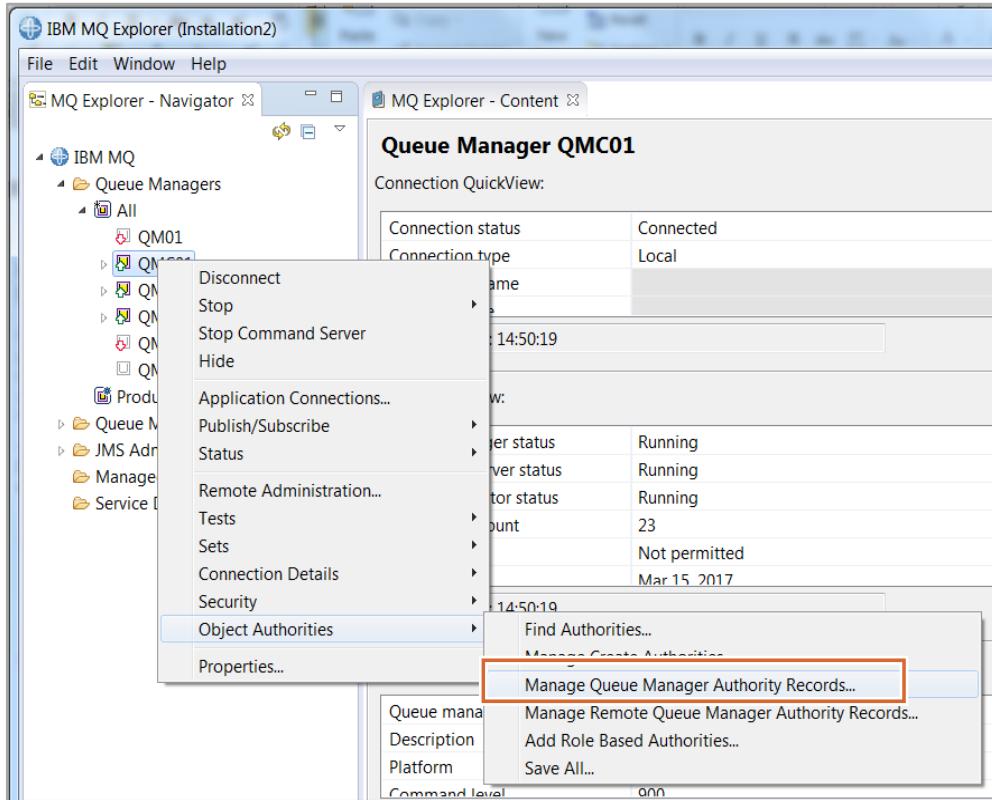
- On UNIX and Linux systems, a principal is a user ID, or an ID associated with an application program that runs on behalf of a user; a group is a system-defined collection of principals.
- On Windows systems, a principal is a Windows user ID, or an ID associated with an application program that runs on behalf of a user; a group is a Windows group.

Authorizations can be granted or revoked at the principal or group level.

The OAM and object authority configuration was described in detail in the prerequisite courses WM153, *IBM MQ V9 System Administration*.



## Managing object authorities in IBM MQ Explorer (1 of 2)



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-19. Managing object authorities in IBM MQ Explorer (1 of 2)

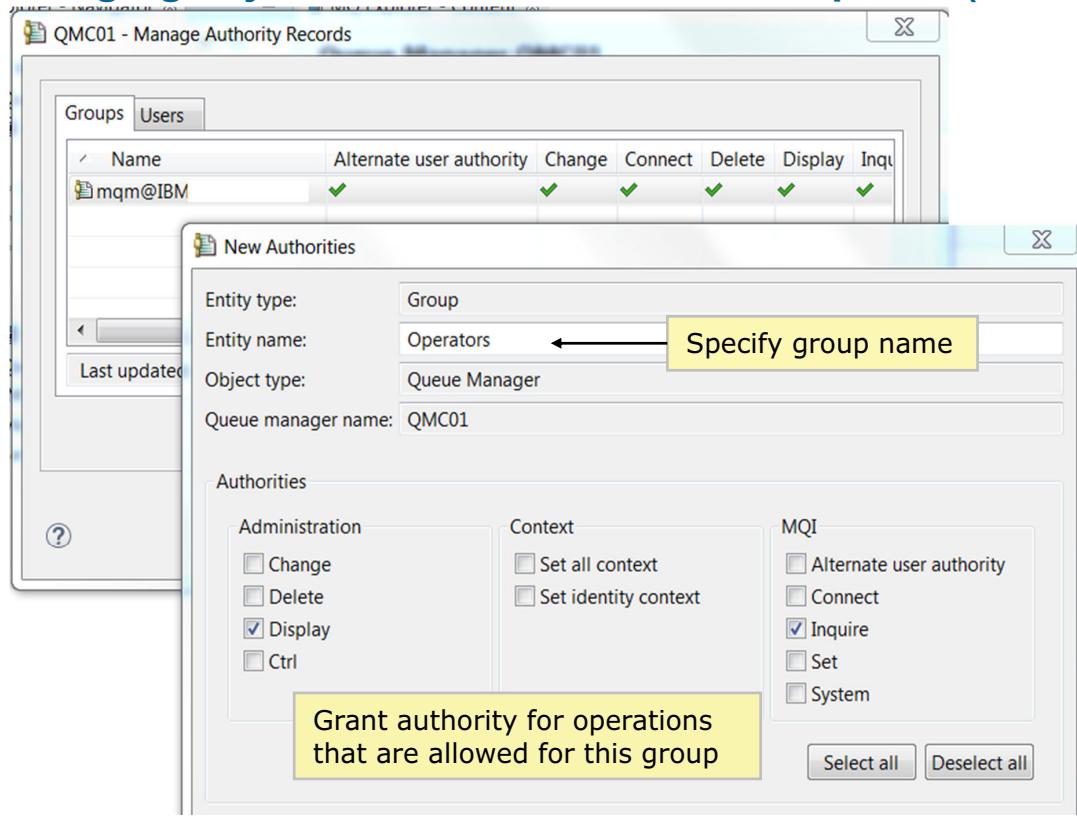
You can use IBM MQ Explorer to manage authorities through other installable authorization services.

To manage object authorities in IBM MQ Explorer, right-click the queue manager or queue in the **MQ Explorer - Navigator** view and then click **Object Authorities > Manage Authority Records**.

The example in the figure shows the option for managing queue manager authorities.



## Managing object authorities in IBM MQ Explorer (2 of 2)



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-20. Managing object authorities in IBM MQ Explorer (2 of 2)

In this example, an authority record is added to the queue manager QM01 for the group that is named “Operators”.

To add a group authority:

1. Right-click the queue manager in the **MQ Explorer - Navigator** view and then click **Object Authorities > Manage Authority Records**.
2. Click the **Groups** tab and then click **New**.
3. Select the authorities and then click **OK**.

## Managing authority records in MQSC

- **SET AUTHREC** to set authority records that are associated with a profile name

Example:

```
SET AUTHREC PROFILE(APP.QUEUE) OBJTYPE(QUEUE) +
GROUP('appuser') AUTHADD(INQ, DSP, BROWSE, PUT, GET)
```

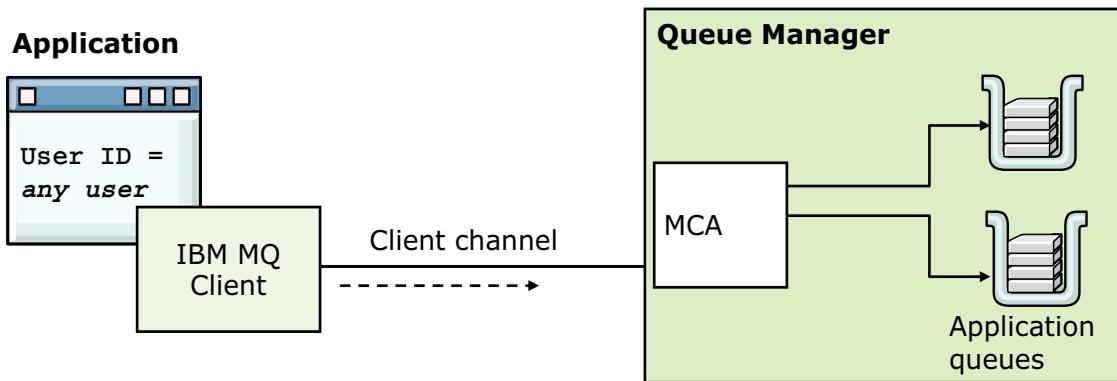
- **DISPLAY AUTHREC** to display the authority records associated with a profile name
- **DELETE AUTHREC** to delete authority records that are associated with a profile name

Figure 3-21. Managing authority records in MQSC

You can also manage authority records by using the SET AUTHREC, DISPLAY AUTHREC, and DELETE AUTHREC commands.

## Application to queue manager

- Application can be:
  - A long-running batch type process
  - Interactive users
- If MCAUSER is not set in SVRCONN channel definition, the application can claim to be any user ID



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-22. Application to queue manager

When an application connects to a queue manager over a client channel, the default behavior for the IBM MQ client is to get the logged on user ID. The client then passes the user ID to the queue manager, where it is set in the MCAUSER field. This behavior can be overridden with application code or channel security exits on the client-side to present any arbitrary user ID. So, it is never safe to trust the ID presented over a server-connection (SVRCONN) channel.

## Blank MCAUSER: Main cause of security exposures

- Any inbound channel definition with a blank MCAUSER value implicitly allows:
  - User impersonation
  - Administrative authority
- To check your system:
  1. In MQSC, enter: DISPLAY CHANNEL (\*) MCAUSER
  2. If any inbound channel has MCAUSER( ), then you have a security exposure

**Note:** This security exposure applies to any channels that are created automatically

Figure 3-23. Blank MCAUSER: Main cause of security exposures

If you have a channel with a blank MCAUSER value, an unauthorized user can access your queue manager with full administrative authority. If someone used this channel, you would not be able to determine who it was.

Use the DISPLAY CHANNEL command to check for a blank MCAUSER on a channel.

Secure the SYSTEM.DEF and SYSTEM.AUTO channel definitions. IBM MQ administrators sometimes incorrectly assume that the SYSTEM.AUTO.\* definitions are not usable by IBM MQ if the CHAD (automatic channel definition) attribute of the queue manager is set to DISABLED but it is not the case.

## MCAUSER remedy

- Provision two low-privileged user IDs and corresponding private groups
  - One for MCA channels
  - One for MQI (SVRCONN) channels
- In the channel definition:
  - Set **MCAUSER** to the user ID depending on channel type
  - Set **PUTAUT** to the value of **DEF**
- Authorize the private group to the application queues
  - Equivalent to using a blank MCAUSER in that only one authorization profile is used, but it is a low-privilege profile
- If clustering is used, also authorize the user ID to the **SYSTEM.CLUSTER.COMMAND.QUEUE**
- Do not authorize this user ID to any transmit queues
  - If multihop routing is required, use remote queue definitions

[Authenticating channels and connections](#)

© Copyright IBM Corporation 2017

*Figure 3-24. MCAUSER remedy*

The figure lists the configuration options that you can implement to ensure that an MCAUSER is not blank.

An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both MCAUSER on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records take precedence. The MCAUSER on the channel definition is only used if the channel authentication record uses USERSRC(CHANNEL).

## Secure remote administration

- Setting MCAUSER might mean you cannot use remote IBM MQ administration
  - IBM MQ administrators must log on to the server locally
  - IBM MQ provides ways to securely allow secure remote administration
- Client-channel security exit
  - Provides a way to authenticate the user at the client end
  - Passes this user ID to the server end
- Use of TLS and client certificate:
  - Can use TLS authentication to secure client channels
  - Configure it to allow only administrators to connect

[Authenticating channels and connections](#)

© Copyright IBM Corporation 2017

*Figure 3-25. Secure remote administration*

While setting MCAUSER can prevent someone from inappropriately obtaining administration access to a queue manager, the downside is that it also prevents legitimate access. The IBM MQ administrator now needs to log on to the server on which the queue manager is running to administer it. IBM MQ administration might not be much of a problem if you have a few queue managers only. But if there are many queue managers, or if they are clustered, then not being able to use a tool like IBM MQ Explorer to manage multiple queue managers creates inefficiencies. So, it is still necessary to provide secure remote administration.

To provide secure remote administration, you need to configure the environment so that you can be sure that only those people who need this capability can obtain it.

One approach is to develop a channel security exit for use on client channels. A security exit is able to set a user ID on the MCAUSER field. The exit on the client-side would need to gather the user's ID so that you can be sure that the user is the one you expect. It would then pass this user ID to the exit on the queue manager end, where it would be set in the MCAUSER field. The security exits at each end of the client channel can communicate with each other. Coding channel security exits is a nontrivial task. Also, if the client-side security exit is passing credentials, such as a user ID and password, to the server-side security exit, then use SSL to encrypt this communication.

Another approach can be to use TLS and client certificates. You can define a CLNTCONN and SVRCONN definition that specifies the use of client certificates. By ensuring that only your IBM MQ

administrators have access to these client certificates, you can control access. A typical use of a server-side exit is to map the TLS certificate to a user ID and set the MCAUSER with the authenticated ID.

Without SSLPEER, any certificate that a trusted signer issues is accepted. Set SSLPEER on the channel and delete unneeded certificate authorities from the truststore. If you use IBM Key Management to create a keystore, it contains well-known CA certificates. Delete the well-known CA certificates if you are not planning to use them.

## Authorizing groups and principals

- On UNIX and Linux
  - You can authorize access to groups
  - If enabled on the queue manager, you can authorize access to principals (users)
- On Windows
  - You can authorize access to groups and principals (users)
  - Fully qualify principals with the domain to avoid ambiguity

Example: `authorize -p user@domain`
- Decide what actions are necessary to apply the appropriate level of security for the groups or principals to the IBM MQ components
  - Who needs access to the queue manager?
  - Do these users or groups need full administrative access or partial administrative access on a subset of queue manager resources?
  - Do these users or groups need read-only access to all queue manager resources?

*Figure 3-26. Authorizing groups and principals*

You can authorize access to IBM MQ resources by groups or principals. Before you configure authorization, determine whether access should be limited to a user or a group.



### Note

The ability to authorize principals on UNIX or Linux must be explicitly configured on the queue manager.

To enable principal authorization when you create the queue manager, add the `-oa user` option to the `crtmqm` command. If you do not set this parameter, group authorization is used.

You can change the authorization model later by setting the **SecurityPolicy** parameter in the **Service** stanza of the `mq.ini` file.

## Interactive users with OAM

- User IDs and authorization policies that are associated with interactive users can be dynamic
  - Ongoing administration can be extensive and expensive
- Do not routinely grant CREATE, DELETE, CHANGE, CONTROL, and CONTROLX
  - These authorizations confer administrative authority to create, delete, or change objects and to start or stop channels or services

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-27. Interactive users with OAM*

It might be better to enable authorizations for groups than by principals to reduce the amount of administration. It is important to ensure that all users in a group should have access to resources.

Use care when giving principals or groups access to create, delete, change, or control IBM MQ resources.

## Considerations for applications with OAM

- Applications are generally fairly static and relatively small in number
- Do not make authorization too restrictive with OAM
  - Understand the access the application really requires
- Most applications need only PUT or GET/BROWSE and sometimes INQUIRE
- JMS applications always need INQUIRE
- Assign applications their own specific exception queue
  - For dead messages, for example
  - However, they are still allowed PUT access (but not GET access) to the dead-letter queue

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-28. Considerations for applications with OAM*

This figure lists some considerations when configuring IBM MQ resource access for applications.

## OAM and queue manager to queue manager

- Set MCAUSER on an inbound MCA channel (RCVR, RQSTR, or CLUSRCVR) to a low-privileged ID to restrict that channel so it can access only specific queues
- In addition to **+put** and **+inq**, the ID needs **+setall** authority
  - An administrative right, but the channel agent is trusted code
  - Do not allow other users in this group and disable the account
- Do not authorize access to the command queue
- Consider restricting access to the dead-letter queue and to any transmission queues
- Business requirements might determine whether you need to use TLS between queue managers
  - Consider many factors such as physical security, network interception points, and disaster recovery requirements

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-29. OAM and queue manager to queue manager

This figure lists some tips for configuring access for queue manager to queue manager authorization.

If TLS is not used, some other method of authentication, such as using security exits, is required. Physical security and static IP addresses make IP filtering potentially useful for these connections.

## A note about transmission queues

- Authorization to access a transmission queue gives the user PUT authority to any queue that is accessible from that transmission queue
- For the cluster transmission queue, the user can PUT to any queue in the cluster, whether it is advertised to the cluster or not
  - Restrict access to a transmission queue at the receiver side with a low-privileged MCAUSER user ID
  - Restrict access to a transmission queue at the sender side with aliases or by using remote queue definitions

Figure 3-30. A note about transmission queues

Remember to restrict access to the transmission queues. Authorization to a transmission queue gives the user PUT authority to any queue that is accessible from the transmission queue.

Allowing a PUT to a transmission queue can open the network to security attacks.

## PUTAUT: Put Authority for channels

- When PUTAUT(CTX) is enabled, the alternative user ID is used from the context information that is associated with the message
- Because PUTAUT(CTX) requires all queue managers to be in the same authentication domain, it might not be practical to use PUT authority in a mixed operating system environment and is therefore seldom used
- Alternative to authorizing the transmission queue requires a remote queue or alias queue for every authorized remote destination
  - OAM privileges are then granted on the locally defined remote queue and alias queue objects

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-31. PUTAUT: Put Authority for channels*

The administrator must also consider whether to allow PUT authority for channels.

## Security guidelines

- When sending and receiving messages with an organization external to your company:
  - Use a dedicated gateway queue manager
  - Do not have channels that are connected directly to internal queue managers
- Do not allow an IBM MQ client to come through a firewall to an internal queue manager
- Do not allow IBM MQ clustering through a firewall
- Run all IBM MQ layered services such as dead-letter queue handler, trigger monitor, and HTTP bridge as low-privileged applications
  - Do not put the user accounts to run IBM MQ layered services in the “mqm” group
  - If you start services as IBM MQ defined service, use a mechanism that starts them with a low-privileged ID

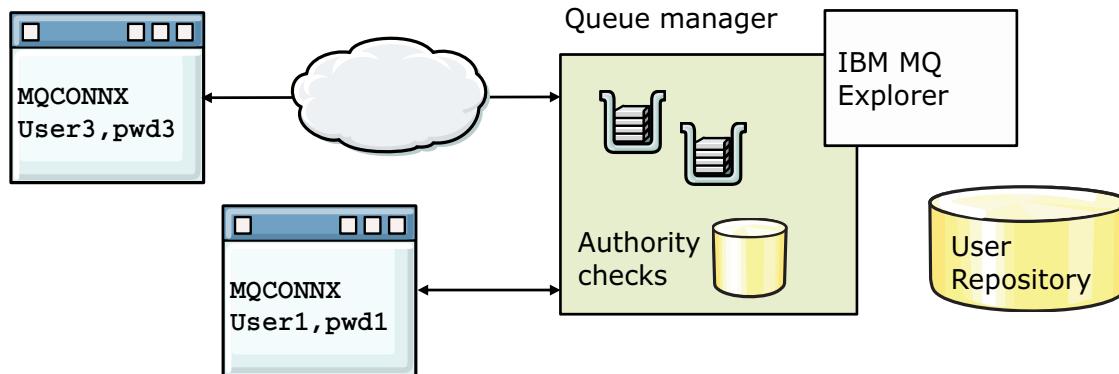
Figure 3-32. Security guidelines

This figure lists some guidelines for helping to keep your IBM MQ network secure.

### 3.3. Connection authentication

## Connection authentication

- Ability for an application to provide a user ID and password
  - Client
  - Local bindings
- Enabled by default in new IBM MQ V9 queue managers
- Some configuration is required in the queue manager
- Requires a user repository that knows whether the user ID and password are a valid combination



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-33. Connection authentication

This figure shows the general IBM MQ network that is referenced in this topic for connection authentication.

On the left side of the network are applications that are making connections. The top application is connecting as a client. The bottom application is connecting by using local bindings. These applications might be using different APIs to connect to the queue manager, but they all can provide a user ID and a password. The user ID that the application is running under might be different from the user ID provided by the application with its password. The user ID of each application is shown on the figure.

In the middle of the network is the queue manager. The queue manager is responsible for managing resources and the checking of authority to those resources. There are many resources in IBM MQ that an application might require authority to, but this example uses a queue that must be opened for output.

On the right of the figure is a representation of a user repository that contains the user IDs and passwords.

## Enabling connection authentication on a queue manager

- Set the CONNAUTH attribute on the queue manager to the name of an authentication information object
- Define the authentication information object with the DEFINE AUTHINFO command
- Use the REFRESH SECURITY TYPE(CONNAUTH) command to refresh the cached view of the configuration for connection authentication

*Figure 3-34. Enabling connection authentication on a queue manager*

The first step is to enable connection authentication on the queue manager.

On the queue manager definition, the CONNAUTH (connection authentication) attribute points to an authentication information object. The authentication information object is defined by using the DEFINE AUTHINFO command.

After configuration is complete, the final step is to refresh the security cache for connection authentication.

## Connection authentication configuration

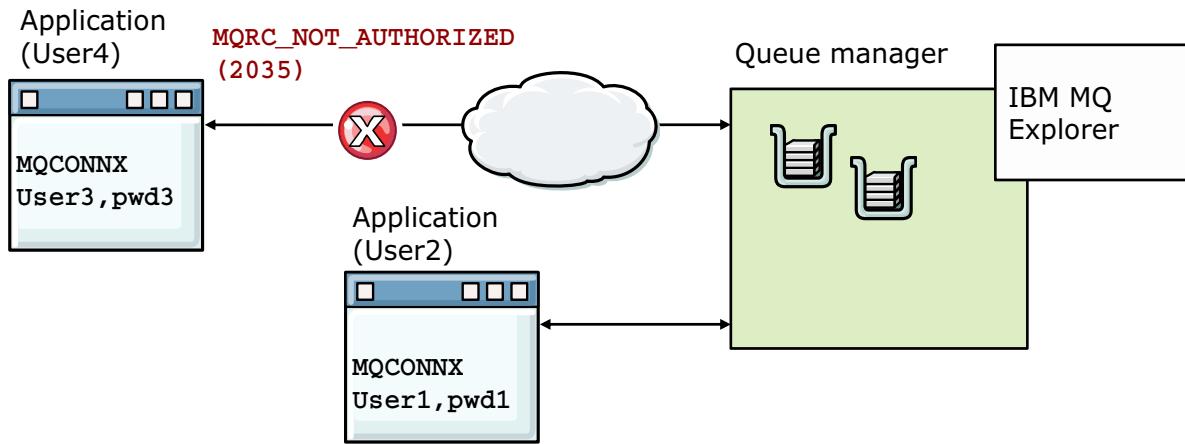
```

ALTER QMGR CONNAUTH(USE.PW)

DEFINE AUTHINFO(USE.PW) AUTHTYPE(xxxxxxxx)
FAILDLAY(1) CHCKLOCL(OPTIONAL)
CHCKCLNT(REQUIRED)

REFRESH SECURITY TYPE(CONNAUTH)

```



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-35. Connection authentication configuration

The figure shows the three main commands that configure connection authentication.

- The ALTER QMGR command modifies the queue manager to refer to a connection authentication information object that is named USE.PW.
- The DEFINE AUTHINFO command defines the USE.PW connection authentication information object.
- The REFRESH command updates the connection authentication and the cache so that the queue manager recognizes the changes.

The AUTHINFO command contains two parameters for user ID and password checking on a connection: CHCKLOCL (check local connections) and CHCKCLNT (check client connections).

Both of these parameters have the same set of attributes, which provides for a strictness of checking. You can set these fields to the following values:

- NONE disables connection authentication. When password checking is turned off with the NONE option, invalid passwords are not detected.
- OPTIONAL validates a user ID and password when the application provides them but a user ID and password are not mandatory.
- REQUIRED requires that all applications provide a user ID and password.

- REQDADM requires a valid user ID and password from a privileged user and is optional for nonprivileged users.

Any application that does not supply a user ID and password when required to, or supplies an incorrect combination even when it is optional receives an MQRC\_NOT\_AUTHORIZED error.

Any failed authentications are held for the number of seconds that are specified in FAILDLAY field before the error is returned to the application. This attribute provides some protection against a “busy loop” from an application that is repeatedly connecting.

In the example, the local application connection succeeds because CHCKLOCL is set to OPTIONAL.

## Connection authentication error notification

- Application: MQRC\_NOT\_AUTHORIZED (2035)
- Administrator: Error message
- Monitoring tool
  - “Not Authorized” event message (Type 1 – Connect)
  - Connection not authorized: MQRQ\_CONN\_NOT\_AUTHORIZED
  - User ID and password not authorized: MQRQ\_CSP\_NOT\_AUTHORIZED
  - Field in the “connect” event: MQCACF\_CSP\_USER\_IDENTIFIER
- To enable authorization events: `ALTER QMGR AUTHOREV (ENABLED)`

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-36. Connection authentication error notification

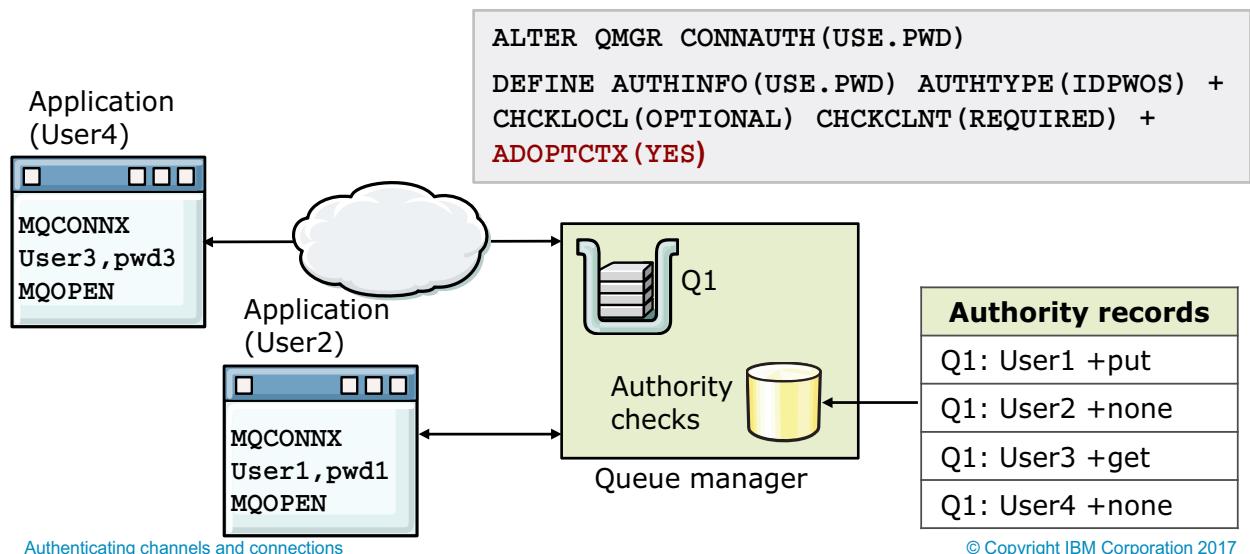
When an application provides a user ID and password that fails the authentication check, the application receives the standard IBM MQ security error, 2035 – MQRC\_NOT\_AUTHORIZED. The administrator can view this error in the error log and can see that the application was rejected because the user ID and password check failed.

A monitoring tool can also be notified of this failure by checking the `SYSTEM.ADMIN.QMGR.EVENT` queue if authority events are enabled. The generated event is a *Not Authorized* event, which is a Type 1, Connect event. This type of event includes the `MQCSP` field that indicates that a user ID is provided so two user IDs are provided in the message: the user ID that the application is running as and the user ID that the application presented for user ID and password checking. The event message does not provide the password.

To enable authority events, enter the MQSC command `ALTER QMGR AUTHOREV (ENABLED)`

## Connection authentication and authorization

- When **ADOPTCTX = NO**, user ID provided by the application is authenticated but authorization uses the user ID that the application is running under
- When **ADOPTCTX = YES**, user ID that the application provides in the MQSCP structure of the MQCONN call is used for authentication
  - To adopt another security context, also set **ChauthEarlyAdopt** to Y in **Channels** stanza of **qm.ini**



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-37. Connection authentication and authorization

On the previous figure, you saw how you can configure the queue manager to mandate user IDs and passwords from specific applications. You should also recognize that the user ID that the application is running under might not be the same user ID that the application presents.

So what is the relationship of these user IDs to the ones that are used for the authorization checks when the application, for example, opens a queue for output? The ADOPTCTX attribute on the authentication information object controls the relationship.

If you set ADOPTCTX(NO), the applications provide a user ID and password for the purposes of authenticating them at connection time, but continue to use the user ID that they are running under for authorization checks. The option might be useful when upgrading, or beneficial for client connections because authorization checks are done by using an assigned MCAUSER based on IP address or SSL/TLS certificate information.

The MQCSP structure on an MQCONN call enables the authorization service to authenticate a user ID and password. When you use the ADOPTCTX(YES) parameter on an authentication information object, the security context is set as the user ID that is presented in the MQCSP structure, when validated by a password. In this case, another security context cannot be adopted, unless you set the **ChauthEarlyAdopt** to Y in the **Channels** stanza of **qm.ini** file.

---

**1+1=2 Example**

The default authentication information object is set to ADOPTCTX(YES), and the user “fred” is logged in. The following two CHLAUTH rules are configured:

```
SET CHLAUTH ('MY.CHLAUTH') TYPE (ADDRESSMAP) DESCRIPTOR ('Block all access by default') ADDRESS ('*') USERSRC (NOACCESS) ACTION (REPLACE)
```

```
SET CHLAUTH ('MY.CHLAUTH') TYPE (USERMAP) DESCRIPTOR ('Allow user bob and force CONNAUTH') CLNTUSER ('bob') CHCKCLNT (REQUIRED) USERSRC (CHANNEL)
```

The following command is entered, with the intention of authenticating the command as the adopted security context of the user “bob”:

```
runmqsc -c -u bob QMGR
```

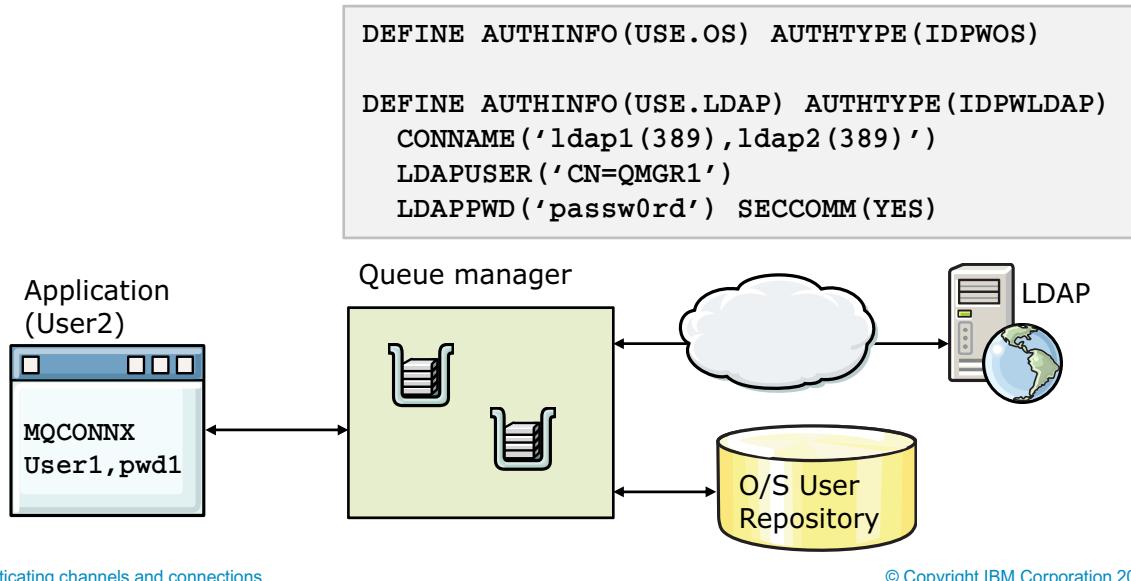
The queue manager uses the security context of “fred”, not “bob”, and the connection fails.

To use the security context of “bob”, the **ChlauthEarlyAdopt** parameter in the **Channels** stanza of the `qm.ini` file must be set to **Y**.

---

## Connection authentication user repositories

- 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



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-38. Connection authentication user repositories

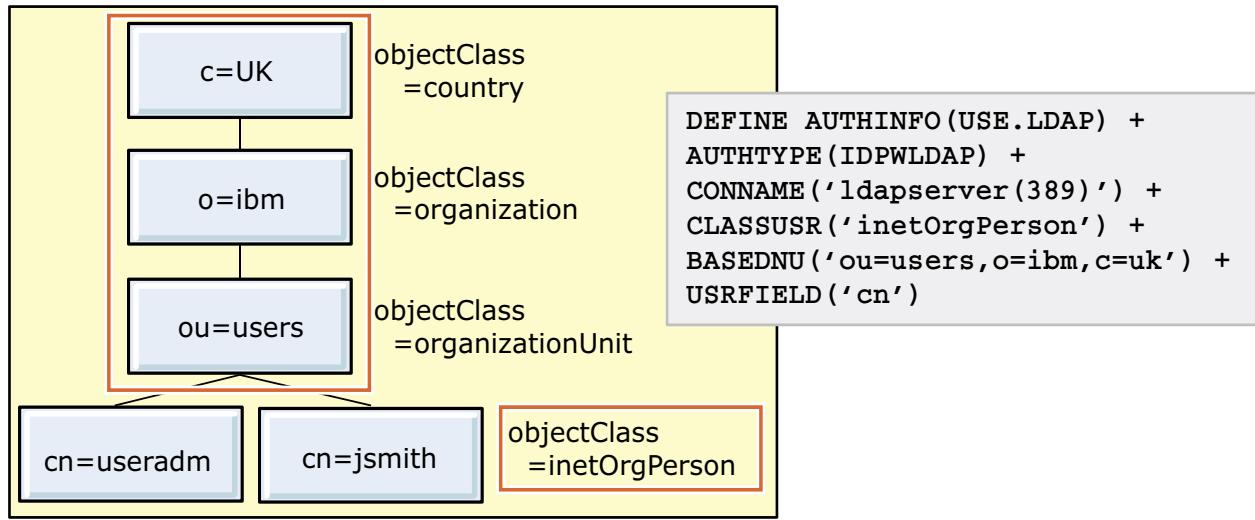
This figure describes the object types of the authentication information object.

The authentication information objects are defined on the AUTHTYPE field of the DEFINE AUTHINFO command.

- AUTHTYPE(IDPWOS) indicates that the queue manager uses the local operating system to authenticate the user ID and password.
- AUTHTYPE(IDPWLDAP) indicates that the queue manager uses an LDAP server to authenticate the user ID and password.

Only one type of authentication can be chosen for the queue manager.

## LDAP user repository



<b>Application provides</b>	<b>USRFLD</b>	<b>BASEDNU</b>
cn=useradm,ou=users,o=ibm,c=uk		
cn=useradm		Adds ou=users,o=ibm,c=uk
useradm	Adds cn=	Adds ou=users,o=ibm,c=uk

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-39. LDAP user repository

When using an LDAP user repository, the queue manager needs to know the location of the LDAP, and other information.

User ID records defined in an LDAP server use a hierarchical structure to uniquely identify them. An application can connect to the queue manager and present its user ID as the fully-qualified hierarchical user ID. However, this fully-qualified hierarchical user ID contains much information, and it would be simpler to configure the queue manager to a more generic information. For example, it would be simpler to have the queue manager assume that all user IDs that are presented are found in a specific area of the LDAP server and then add the qualification. In the DEFINE AUTHINFO command, the BASEDNU attribute identifies the area in the LDAP hierarchy where all the user IDs are found. So, the queue manager adds the BASEDNU value to the user ID presented by an application to fully qualify it before looking it up in the LDAP server.

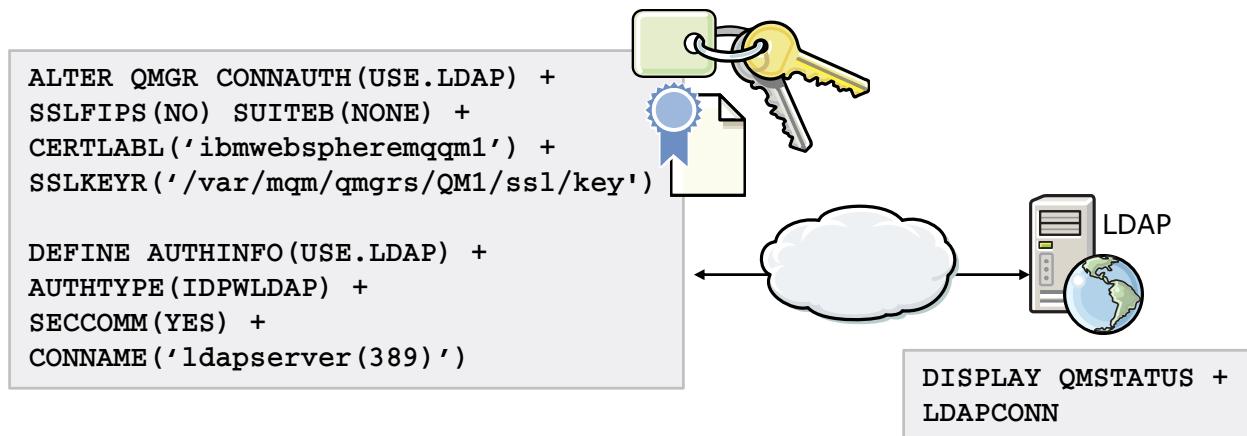
Also, an application might present the user ID but not the LDAP attribute name, such as "CN=". If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, the USRFIELD field in the DEFINE AUTHINFO command identifies the field in the LDAP user record that is used to interpret the provided user ID.

Any user ID that is presented to a queue manager without an equals sign (=) has the attribute and the equal sign added as a prefix and the BASEDNU value added as a suffix before looking it up in the LDAP server. This process can be beneficial when moving from operating system user IDs to

LDAP user IDs because the application might present the same string in both cases, which avoids any change to the application.

## Secure connection to an LDAP server

- Enable SECCOMM on AUTHINFO command for TLS communication
- Queue manager attributes SSLFIPS and SUITEB restrict the set of cipher specs
- OCSP servers that are named in the AIA certificate extensions check certificate revocation
- Use DISPLAY QMSTATUS command to checks status of the connection from the queue manager to the LDAP server



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-40. Secure connection to an LDAP server

You might want to secure your connection between the LDAP server and the queue manager. Unlike channels, there is no SSLCIPH parameter to enable the use of SSL/TLS for the communication with the LDAP server. So, in this case, IBM MQ acts as a client to the LDAP server. This topology requires that much of the configuration is done on the LDAP server.

Some parameters in IBM MQ (shown on the figure) define the connection between the queue manager and the LDAP server.

- The SECCOMM attribute on the DEFINE AUTHINFO command enables SSL/TLS communication.
- The queue manager attributes SSLFIPS and SUITEB restrict the set of cipher specifications that are chosen.

The NSA Suite B standard restricts the set of enabled cryptographic algorithms to provide an assured level of security. IBM MQ can be configured to operate in compliance with the NSA Suite B standard on Windows, UNIX, and Linux.

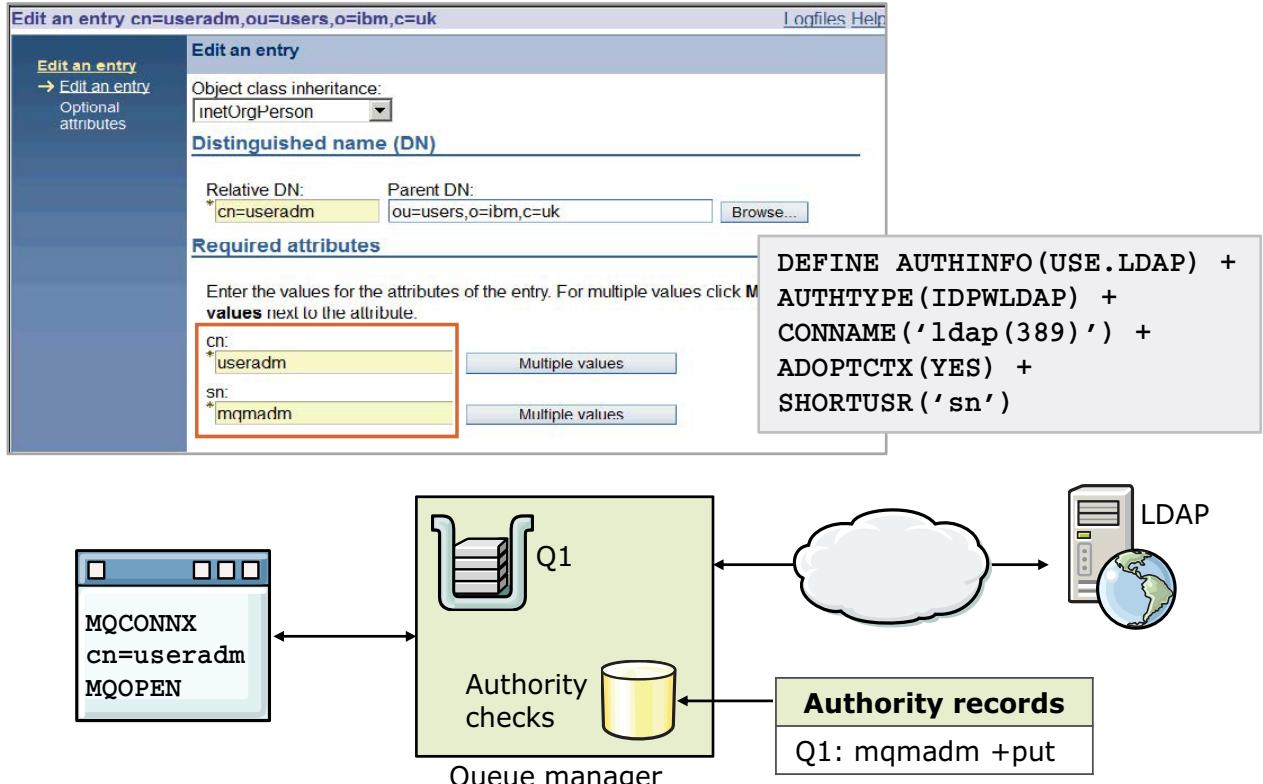
The Federal Information Processing Standards (FIPS) Cryptomodule Validation Program of the US National Institute of Standards and Technology and the cryptographic functions can be used in IBM MQ on TLS channels, for Windows, UNIX, Linux, and z/OS.

- The certificate that identifies the queue manager to the LDAP server is the queue manager certificate that the string `ibmwebspheremq<qmgr-name>` or the CERTLBL attribute identifies.
- Certificate revocation is checked by using the OCSP servers that are named in the AIA certificate extensions. This option can be enabled setting the SSL stanza attribute **OCSPCheckExtensions** in the `qm.ini` file.

The connection to an LDAP server is made as a network connection. The status of this connection from the queue manager to the LDAP server is shown by entering the DISPLAY QMSTATUS command.



## Connection authentication and authorization: LDAP



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-41. Connection authentication and authorization: LDAP

Previously in this topic, you learned about using the ADOPTCTX attribute to adopt the authenticated user ID as the context for the connection. So how does this work if you are using LDAP as the user repository but the operating system user ID is used for authorization?

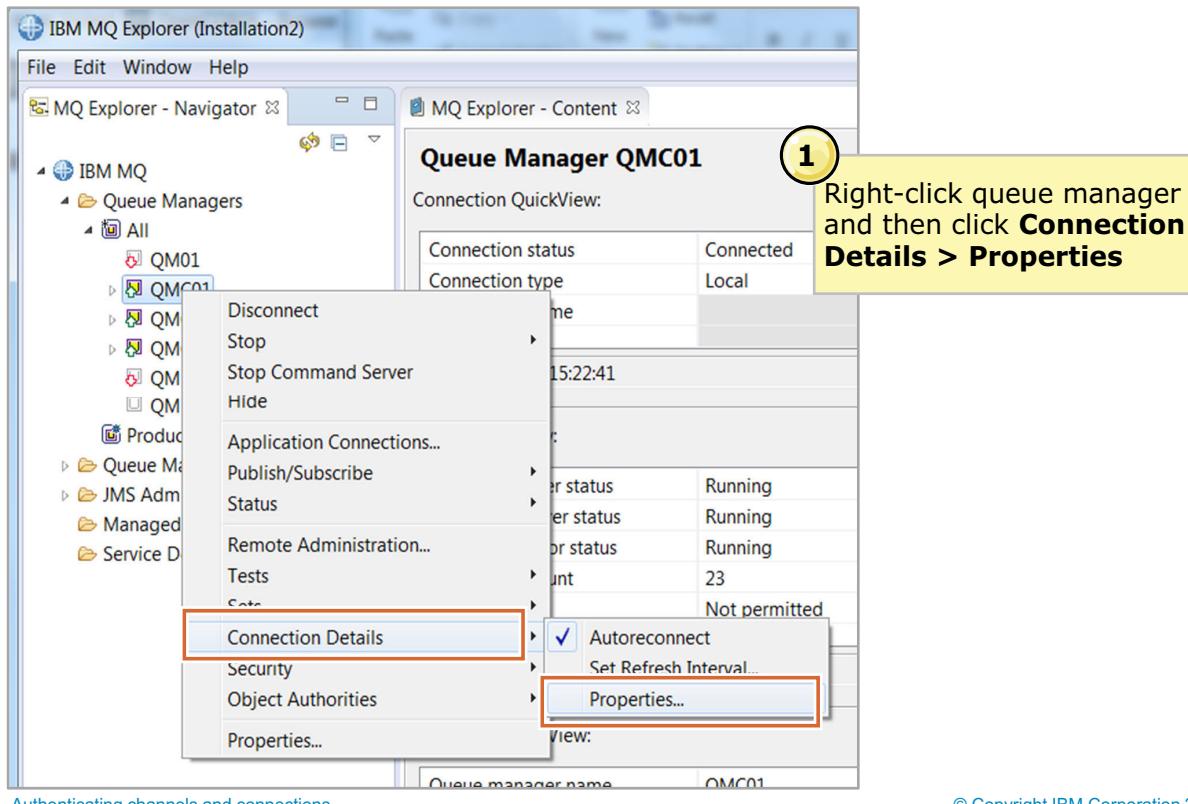
You need to get a user to represent the LDAP user that was presented, as an operating system user ID. This user can be found from the LDAP user record. The user can be any field that is defined in the user record, such as the short name field (`sn=`), which is a mandatory part of the definition of the `inetOrgPerson` class. Alternatively, the user might be in the user ID (`uid=`) field.

The queue manager then uses the LDAP user information to determine the operating system user ID to use as the context for this connection.

You configure the LDAP user by using the SHORTUSR option on the DEFINE AUTHINFO command.

In the example in the figure, the application connects with `cn=useradm`. The SHORTUSR option identifies `sn` as the location for the user name. The LDAP server looks up the record with `cn=useradm` and returns the short name of `mqmadm`. This user name is the user ID that is used for authority checks by the queue manager.

## User ID and password for IBM MQ Explorer (1 of 2)



Authenticating channels and connections

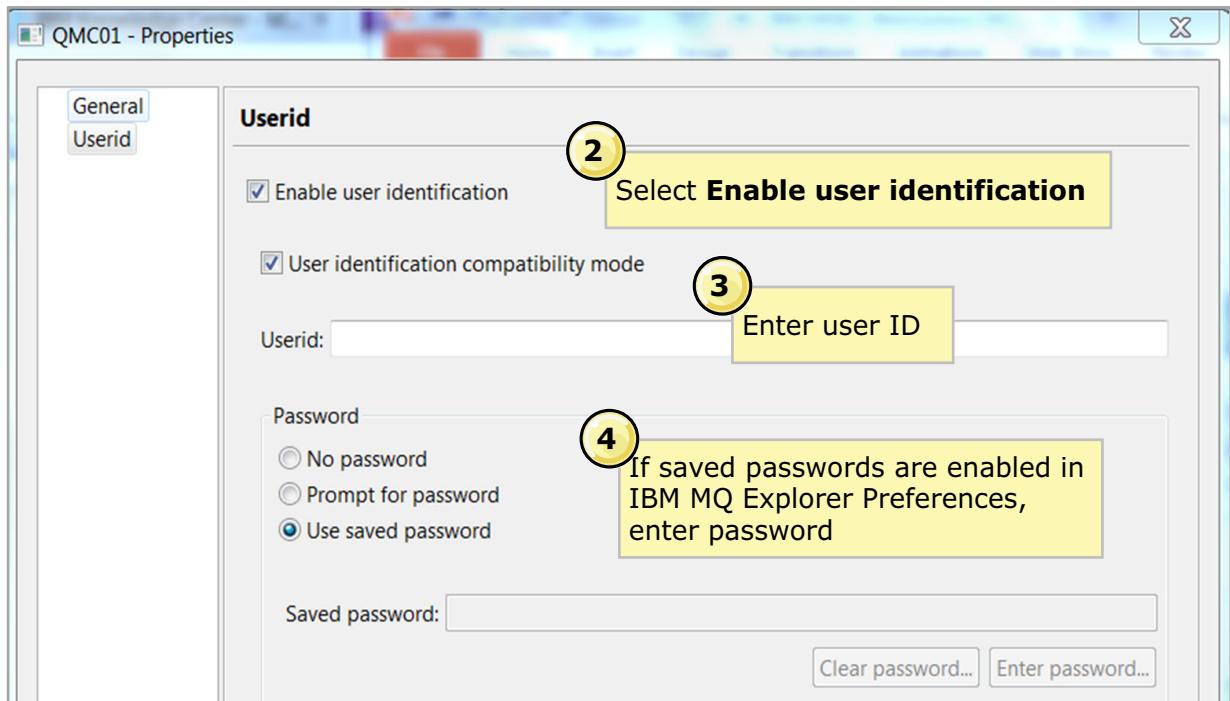
© Copyright IBM Corporation 2017

Figure 3-42. User ID and password for IBM MQ Explorer (1 of 2)

IBM MQ Explorer can be configured with a user ID and password that it uses to connect to a local or client connection to a queue manager.

1. Right-click the queue manager in the **MQ Explorer - Navigator** view and then click **Connection Details > Properties**.

## User ID and password for IBM MQ Explorer (2 of 2)



Authenticating channels and connections

© Copyright IBM Corporation 2017

2. On the **Userid** page, click **Enable user identification**.
  3. Enter the user ID.
  4. IBM MQ Explorer has a password cache that must be enabled to use passwords. If saved passwords are enabled, enter the password.
- If you did enable saved passwords, a link appears on the **Userid** page to help you set up the cache.

## User IDs and passwords in IBM MQ programs

- Some commands allow a user ID and prompt for a password: **dmpmqcfg**, **dspmqrte**, **runmqtrm**, **runmqtmc**, **runmqsc**, and **runmqd1q**
- Some IBM MQ sample programs allow a user ID and prompt for a password
  - Use the environment variable **MQSAMP\_USER\_ID** for the user ID  
Example: `set MQSAMP_USER_ID=j smith`

Sample name	Description
<b>amqspput / amqspputc</b>	Put one or more messages to a queue
<b>amqsget / amsgetc</b>	Get messages from a queue
<b>amqsbcg / amqsbcgc</b>	Read and output the message descriptor fields, any other message properties, and the message content of all the messages on a queue

Figure 3-43. User IDs and passwords in IBM MQ programs

Some of the IBM MQ sample programs allow a user ID and password so that you can test your applications with authentication enabled. Set the sample program user ID by using the **MQSAMP\_USER\_ID** environment variable. When you run the sample program, it prompts you for a password.

For a complete list of the sample programs that support this feature, see the IBM Knowledge Center.

## Connection authentication summary

- Application can provide a user ID and password in the MQCONN call or by using a security exit
- Queue manager checks password against operating system or LDAP

Examples:

```
ALTER QMGR CONNAUTH('CHECK.PWD')
DEFINE AUTHINFO('CHECK.PWD')
AUTHTYPE(IDPWOS|IDPWLDAP)
CHKLOCL(NONE|OPTIONAL|REQUIRED|REQDADM)
CHKCLNT(NONE|OPTIONAL|REQUIRED|REQDADM)
ADOPTCTX(YES)
plus LDAP attributes
```

- Refresh security to update connection authentication

```
REFRESH SECURITY TYPE(CONNAUTH)
```

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-44. Connection authentication summary

This figure summarizes connection authentication.

## 3.4. Channel authentication

## Channel authentication rules

- Set rules to control how inbound connections are treated
  - Inbound clients
  - Inbound queue manager to queue manager channels
  - Other rogue connections that are causing FDCs
- Rules can be set to:
  - Allow a connection
  - Allow a connection and assign an MCAUSER
  - Block a connection
  - Ban privileged access
  - Provide multiple positive or negative TLS peer name matching
- Rules can use any of the following identifying characteristics of the inbound connection
  - IP address or host name
  - TLS subject's Distinguished Name
  - Client asserted user ID
  - Remote queue manager name

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-45. Channel authentication rules

Channel authentication records allow you to define rules for handling inbound connections to the queue manager. Inbound connections include client channels and queue manager to queue manager channels.

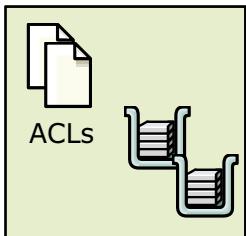
These rules can specify whether connections are allowed or blocked. If the connection is allowed, the rules can provide a user ID for the channel or use the channel user ID from the client or a user ID that was defined on the channel definition.

These rules can be used to do the following actions:

- Set up appropriate identities for channels to use when they run against the queue manager
- Block unwanted connections
- Ban privileged users

## Channel access blocking points

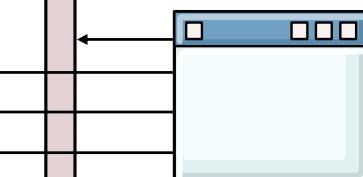
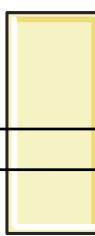
Queue manager



Channel  
blocking/mapping



Listener  
blocking



IP  
firewall

### Channel blocking/mapping

- Rules to block channels, map channels to MCAUSER, and allow channels
- Runs before security exit
- Final check for user ID
  - After security exit runs and final MCAUSER is assigned
  - Ban privileged users with \*MQADMIN

### Listener blocking

- *Not a replacement for an IP firewall*
- Blocked before any data is read from the socket
- Simplistic avoidance of “denial of service” attack
- “Ping” attacks, if blocked, do not raise an alert

Authenticating channels and connections

© Copyright IBM Corporation 2017

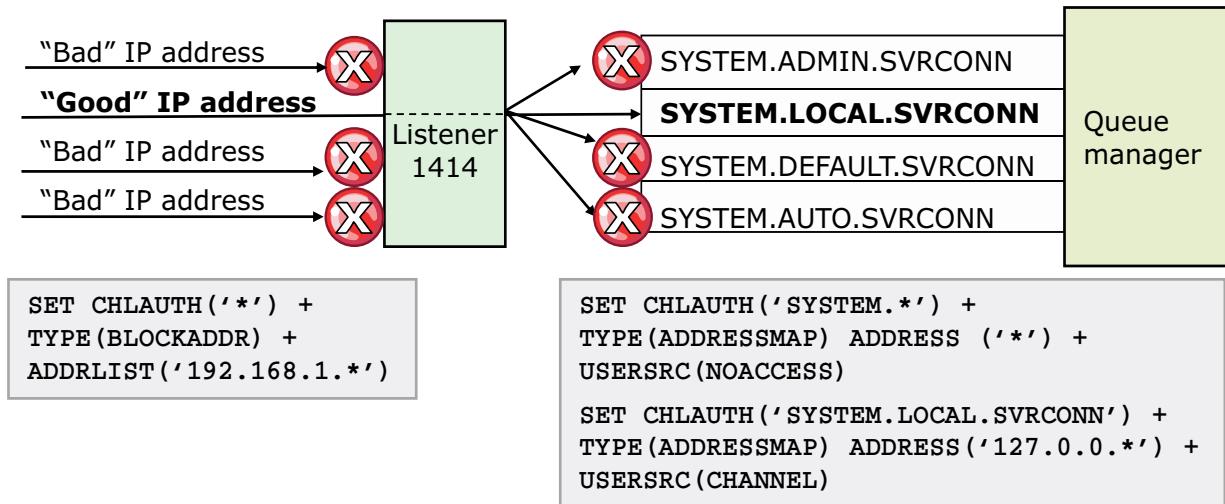
Figure 3-46. Channel access blocking points

This figure shows blocking points that an inbound connection must get through before it can access a queue: the IP firewall, the listener, and the channel.

The IP firewall is included in this set of blocking points. Do not ignore it as a blocking point. IBM MQ security features do not supersede the functions that the IP firewall provides.

The users that are considered as privileged users is different depending on the operating system on which the queue manager runs. There is a special value ‘\*MQADMIN’ that is defined to mean “any user that is privileged on this operating system”. This special value can be used in the rules that check against the final user ID that the channel uses in TYPE(USERLIST) rules to ban any connection that is about to run as a privileged user. This application identifies any blank user IDs that come from a client, for example.

## Channel authentication example



- 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

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-47. Channel authentication example

The figure shows how you can filter connections by using channel authentication rules.

The SET CHLAUTH command on the left in the figure specifies TYPE(BLOCKADDR). It is a global blocking rule that occurs at the listener before the channel name is known. The channel name in the CHLAUTH attribute must be "\*" when the TYPE is BLOCKADDR. The address list (ADDRLIST) is the address from which to refuse 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, the rule blocks all IP addresses that start with 192.168.1.

The per-channel filtering rules on the right have the TYPE attribute set to ADDRESSMAP. These rules map IP addresses to MCAUSER values. An ADDRESS must accompany the ADDRESSMAP parameter.

The first rule restricts access to all SYSTEM channels from any IP address because the USERSRC parameter is set to NOACCESS. The NOACCESS option means that inbound connections that match this rule do not have access to the queue manager and the channel ends immediately.

The second rule gives the local host (127.0.0.1) access to the channel that IBM MQ Explorer uses (SYSTEM.LOCAL.SVRCONN) because the USERSRC parameter is set to CHANNEL. The CHANNEL option means that inbound connections that match this rule use the flowed user ID or any user ID that is defined on the channel object in the MCAUSER field.

## Precedence order

- Rules that use channel authentication records follow a precedence order so that it is clear which rule is used when an inbound connection matches multiple rules

Order	Identity mechanism	Notes
0	Channel name	
1	TLS Distinguished Name	
2=	Client asserted user ID	Several different user IDs can be running on the same IP address.
2=	Queue manager name	Several different queue managers can be running on the same IP address.
4	IP address	
5	Host name	One IP address can have multiple host names.

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-48. Precedence order

Channel authentication rules follow a precedence order so that it is clear which rule is used when an inbound connection can match multiple rules.

A channel authentication rule that explicitly matches the channel name takes priority over a channel authentication record that matches the channel name by using a wildcard.

A channel authentication rule that uses a TLS DN takes priority over a rule that uses a user ID, queue manager name, IP address, or host name.

A channel authentication rule that uses a user ID or queue manager name takes priority over a rule that uses an IP address or host name.

Host names are less specific than an IP address because a single IP address can have multiple host names. If you have an IP address rule and a host name rule that can both match an inbound connection, then the IP address rule is used because it is considered to be more specific.

## Precedence order example

```
DISPLAY CHLAUTH(APPL1.*)
CHLAUTH(APPL1.*)
TYPE(SSLPEERMAP)
SSLPEER('O="IBM UK"') MCAUSER(UKUSER)

CHLAUTH(APPL1.*)
TYPE(USERMAP)
CLNTUSER('jsmith') MCAUSER(SMITH)

CHLAUTH(APPL1.*)
TYPE(ADDRESSMAP)
ADDRESS('9.180.165.163') MCAUSER(SMITH)

CHLAUTH(APPL1.*)
TYPE(ADDRESSMAP)
ADDRESS('* .ibm.com') MCAUSER(IBMUSER)
```

Example input

Chl: APPL1.SVRCONN  
 DN: CN=SMITH.O=IBM UK  
 UID: jsmith  
 IP: 9.180.165.163

*Figure 3-49. Precedence order example*

In this example, you can see that example input record matches more than one rule.

The example input matches all rules at the channel level (APPL1.\*) so the next order of precedence is the DN, which takes precedence over the user ID and IP address. So in this example, the first rule applies and the MCAUSER is set to UKUSER.

## Using IP addresses in channel authentication rules

- Initial listener blocking list
  - List, range, or pattern of IP addresses
  - Does not replace IP firewall
  - Should be used sparingly
- Channel-based blocking of single IP address, range, or pattern
- Channel allowed, based on single IP address, range, or pattern
- Further qualified rule that includes IP address on another rule type
  - Works with SSLPEER, QMNAME, and CLNTUSER

```
SET CHLAUTH('*') TYPE(BLOCKADDR) +
ADDRLIST('9.20.*', '192.168.2.10')
```

```
SET CHLAUTH('APPL1.*') +
TYPE(ADDRESSMAP) ADDRESS('9.20.*') +
USERSRC(NOACCESS)
```

```
SET CHLAUTH('*.*.SVRCONN') +
TYPE(ADDRESSMAP) +
ADDRESS('9.20-21.*') MCAUSER(HUSER)
```

```
SET CHLAUTH('*') TYPE(SSLPEERMAP) +
SSLPEER('CN="Jon Smith"') +
ADDRESS('9.20.*') MCAUSER(JSMITH)
```

*Figure 3-50. Using IP addresses in channel authentication rules*

IP addresses can be used in channel authentication rules in four different ways.

The initial check that the listener makes for banned IP addresses is created with TYPE(BLOCKADDR) in the rule (the first example in the figure). Use this type of rule sparingly. It is intended as an IBM MQ administrator control to temporarily configure banned IP addresses until the IP firewall is updated to restrict access to the network.

After the initial listener check, the mapping rules are applied. You can ban a specific IP address from a channel by using USERSRC(NOACCESS) on a mapping rule (shown in the second example).

You can also map a channel to use a particular MCAUSER or to flow through its client-side credentials if it comes from a particular IP address (shown in the third example).

Finally, IP address restrictors such as SSLPEER, QMNAME, and CLNTUSER can be added to any of the other types of mapping rules (shown in the fourth example).

## Using host names in channel authentication rules

- Initial listener blocking list
  - Host names are not allowed
- Channel-based blocking of host names with single host name or pattern
- Channel allowed, based on host names with single host name or pattern
- Further qualified rule that includes host name on another rule type
  - Works with SSLPEER, QMNAME, and CLNTUSER

SET CHLAUTH('') TYPE(BLOCKADDR) +  
ADDRLIST()

SET CHLAUTH('APPL1.\*') +  
TYPE(ADDRESSMAP) +  
ADDRESS('\*.**ibm.com**') +  
USERSRC(NOACCESS)

SET CHLAUTH('\*.**SVRCONN**') +  
TYPE(ADDRESSMAP) +  
ADDRESS('mach123.**ibm.com**') +  
MCAUSER(HUSER)

SET CHLAUTH('\*') TYPE(SSLPEERMAPP) +  
SSLPEER('CN="Jon Smith"') +  
ADDRESS('s\*.ibm.\*') MCAUSER(JSMITH)

Figure 3-51. Using host names in channel authentication rules

You can substitute host names for IP addresses in most channel authentication rules. The exception is on rules with TYPE(BLOCKADDR). This type of rules allows IP addresses only.

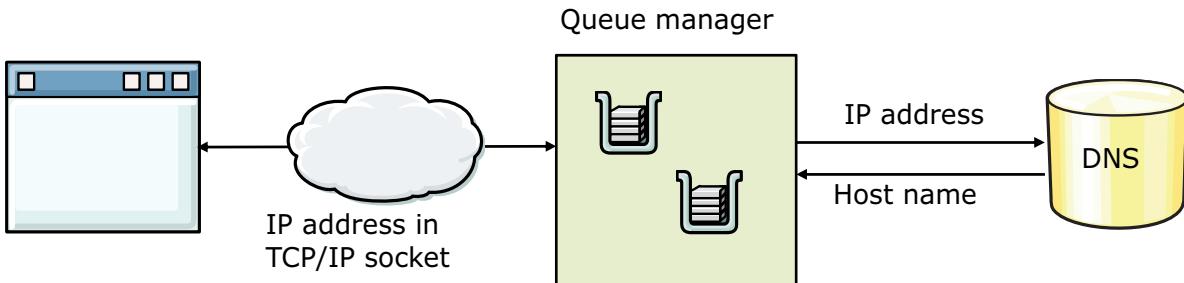
If you want to block specific hosts by using channel authentication, use a TYPE(ADDRESSMAP) rule with host names and USERSRC(NOACCESS), as shown in the second example.

As shown in the other examples, mapping rules, and address restrictors also allow host names instead of IP addresses.

Similar to IP addresses and channel names, host names also support pattern matching wildcard characters. Host names use wildcard string matching, such as **\*.ibm.com**, which matches any host name that ends with **ibm.com**.

## Getting the host name

- Reverse look up of host name from Domain Name Server (DNS) based on IP address
- If host name is used in channel authentication rules:
  - Queue manager must be able to contact DNS
  - DNS must be able to resolve the IP addresses of senders or clients
- To disable reverse lookup, specify: `ALTER QMGR REVDNS (DISABLED)`



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-52. Getting the host name

To process channel authentication rules that contain host names, IBM MQ must be able to get the host name that represents the IP address of the socket. The host name is not sent to IBM MQ by the channel or by TCP/IP. IBM MQ gets the IP address from the socket. IBM MQ gets the other attributes that channel authentication records use from the various internal flows across the socket.

To get the host name, IBM MQ interrogates the Domain Name Server (DNS) to determine which host name matches the IP address in the socket. For this action to succeed, the queue manager must be able to use the DNS. The queue name already uses DNS if host names are specified in CONNAME attributes. Also, the DNS must be able to reverse look-up the IP address and find a host name for IBM MQ, which might not be the case.



### Important

If host names are used in channel authentication rules, verify that all the sender channel or client application IP addresses are currently available in the DNS so that the IP address in the socket can be matched to a host name.

Some administrators are concerned about the potential security hazards of using host names than others. When REVDNS(ENABLED) is specified on the queue manager, the reverse look-up of the

IP address to retrieve the host name is done only when necessary. If you do not use host names in channel authentication rules, then the only time a reverse look-up is done is when writing an error message that contains that information.

## Connection authentication configuration granularity

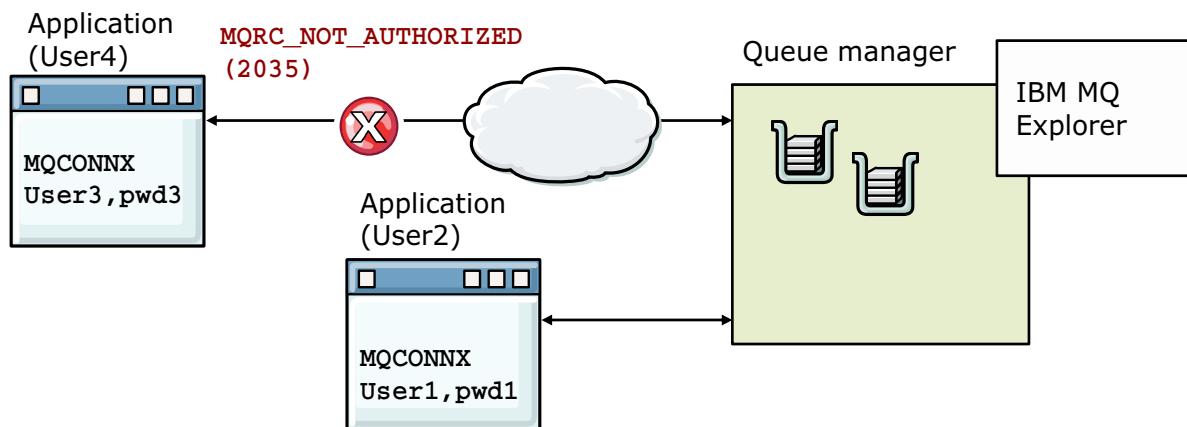
```

DEFINE AUTHINFO(USE.PW) AUTHTYPE(IDPWOS) +
CHKCLNT(OPTIONAL)

SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') +
USERSRC(CHANNEL) CHKCLNT(REQUIRED)

SET CHLAUTH('*') TYPE(SSLPEERMAP) +
SSLPEER('CN=*') USERSRC(CHANNEL) +
CHKCLNT(ASQMGR)

```



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-53. Connection authentication configuration granularity

In addition to the two fields that enable connection authentication (CHKCLNT and CHCKLOCL), you can use channel authentication rules (CHLAUTH) to enable authentication for all client and locally bound applications.

In a channel authentication rule, you set the CHCKCLNT field so that more specific configuration can be made for client connections. For example, you can set CHCKCLNT to OPTIONAL on the AUTHINFO command as the default. You can then make authentication more stringent for certain channels by setting CHCKCLNT(REQUIRED) or CHCKCLNT(REQDADM) on the CHLAUTH rule.

By default, channel authentication rules are run with CHCKCLNT(ASQMGR) so that it defaults to using the authorization set at the queue manager level.

Channel authentication is described in more detail later in this unit.

## Diagnosing host name lookup failures

- IBM MQ returns error message when a channel is blocked
  - Message contains the IP address and host name (if available)
- Use the information in the error message in DISPLAY CHLAUTH MATCH(RUNCHECK) command to get the authentication record that matched the inbound channel at run time

Example message:

```
AMQ9777: Channel was blocked
EXPLANATION:
The inbound channel 'SYSTEM.DEF.SVRCONN' was blocked from address
'smith.ibm.com(9.1.1.1)' because the active values of the channel matched
a record configured with USERSRC(NOACCESS). The active values of the
channel were 'CLNTUSER(hughson) ADDRESS(smith.ibm.com,
smith.hursley.ibm.com)'.
```

*Figure 3-54. Diagnosing host name lookup failures*

If it can be resolved, the error message contains the socket IP address and the host name when a channel is blocked and the channel authentication rule contains a host name. The figure includes an example an error message.

You can the DISPLAY CHLAUTH MATCH(RUNCHECK) command to find the authentication record that matched the inbound channel at run time.

## Using MATCH(RUNCHECK) example

```
DISPLAY CHLAUTH(SYSTEM.ADMIN.SVRCONN) MATCH(RUNCHECK) +
SSLPEER('CN="John Smith", O="IBM UK"') +
CLNTUSER('jsmith') ADDRESS('9.1.1.1')

returns ===>

CHLAUTH(SYSTEM.ADMIN.SVRCONN)
TYPE(ADDRESSMAP)
ADDRESS('*.*.ibm.com') MCAUSER(JSMITH)
```

Example input

Chl: SYSTEM.ADMIN.SVRCONN  
 DN: CN=John Smith.O=IBM UK  
 UID: jsmith  
 IP: 9.1.1.1

Figure 3-55. Using MATCH(RUNCHECK) example

The DISPLAY CHLAUTH MATCH(RUNCHECK) command returns the channel authentication rules that matched a specific inbound channel at run time. The specific inbound channel is described by providing values that are not generic, such as the channel name or IP address that appears in the error message.

You cannot use the host name in the MATCH(RUNCHECK) command because this command needs the IP address to find the host name. The queue manager then calls the DNS, as it would if the real inbound connection appeared, and finds the host name. The queue manager then runs the match against the rules.

If the queue manager is configured to use REVDNS(DISABLED) and you have channel authentication rules that use host names, a warning message appears. So, the DISPLAY CHLAUTH MATCH(RUNCHECK) command can help you to determine whether the reverse look-up for particular IP addresses is likely to work.

## More security options

- Dynamic mapping of MCAUSER
- User ID blocking with CHLAUTH BLOCKUSER
- Filtering by IP address

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-56. More security options*

More security options that you might want to consider when creating your channel authentication rules include the following options:

- Dynamic mapping of the MCAUSER user ID
- Using the CLAUTH BLOCKUSER command to block access based on the user ID
- Filtering by IP address

## Dynamic mapping of MCAUSER

- Configuration that is based on various validation criteria
- Allows fewer channels to support same or finer granularity than a security exit
- Uses standard IBM MQ tools for configuration and management

Authenticating channels and connections

© Copyright IBM Corporation 2017

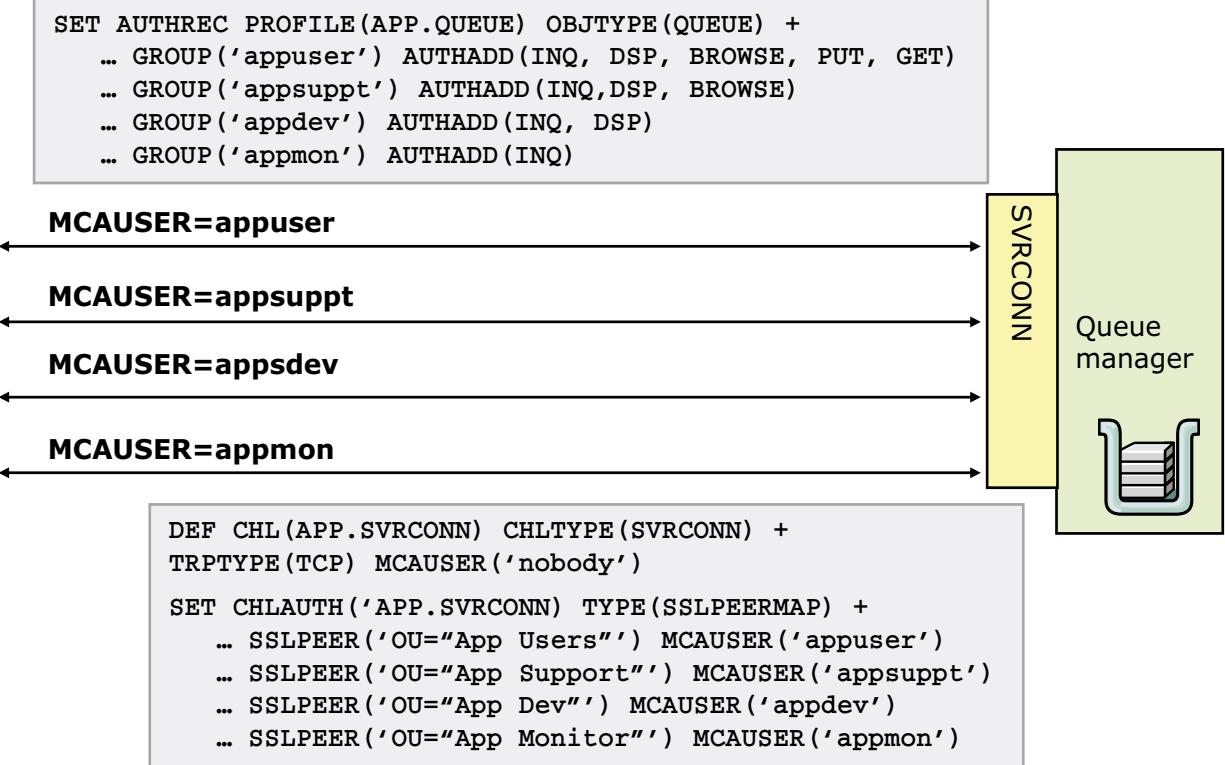
*Figure 3-57. Dynamic mapping of MCAUSER*

An alternative way of providing a user ID for a channel is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials.

If both the MCAUSER on the channel is set and channel authentication rules apply to the same channel, the channel authentication records take precedence. The MCAUSER on the channel definition is only used if the channel authentication record uses USERSRC(CHANNEL).

You can use a channel authentication record to change the MCAUSER attribute of a server-connection channel, according to the original user ID received from a client. You can also use a channel authentication record to set the MCAUSER attribute of a channel, according to the queue manager from which the channel is connecting.

## Dynamic MCAUSER mapping example



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-58. Dynamic MCAUSER mapping example

The figure provides an example of dynamic MCAUSER mapping.

The SET AUTHREC commands set authority records that are associated with a queue profile that is named APP.QUEUE. The authority profile assigns inquire, display, browse, put, or get authority for the groups that are named “appuser”, “appsuppt”, “appdev”, and “appmon” as required. For example, the group that is named “appuser” is given the authority to inquire, display, browse, put, and get. The group that is named “appmon” is given inquire authority only.

The DEFINE CHANNEL command sets the MCAUSER to a nonblank value (“nobody”) so that the MCA uses a user identifier for authorization to access IBM MQ resources.

The SET CLHAUTH commands set the channel authentication record for the APP.SVRCONN channel and creates a peer map. The peer map maps TLS DNs to MCAUSER values. A peer filter that compares the DN of the certificate from the peer queue manager or client at the other end of the channel must accompany the SSLPEERMAP parameter. In the example, the organizational unit (OU) determines the MCAUSER.

Any connection requests that do not match one of the CHLAUTH records are refused. The result is that a single channel definition serves four different security roles for the same application.

The DEF CHL, SET CHLAUTH, and SET AUTHREC definition are all managed by using standard IBM MQ administration tools.

## User ID blocking with CHLAUTH BLOCKUSER

- Rules take effect after all other rules and exits are processed and the final value for MCAUSER is determined
- \*MQADMIN represents administrative users as defined for the local operating system
  - Simplifies restricting access to administrators
- Blank user ID resolved to the ID of the MCA
- Rules are hierarchical and the most specific one matches
- Can implement a limited deny/allow policy by altering list of blocked names at different levels

### Examples:

```
SET CHLAUTH(*) TYPE(BLOCKUSER) USERLIST('nobody, *MQADMIN')
SET CHLAUTH(SYSTEM.ADMIN.*) TYPE(BLOCKUSER) USERLIST('nobody')
```

*Figure 3-59. User ID blocking with CHLAUTH BLOCKUSER*

The BLOCKUSER parameter on the channel authentication rules prevents a specified user or users from connecting. A USERLIST must accompany the BLOCKUSER parameter.

In the user list, the string \*MQADMIN represents administrative users as defined for the local operating system.

Rules in the user list are hierarchical; the most specific rule determines the MCAUSER value.

The figure shows two examples of the SET CHLAUTH TYPE(BLOCKUSER) command.

- The first rule blocks administrative users and the MCAUSER “nobody”, which prevents someone from creating a user ID “nobody” and putting it into an authorized group.
- The second rule restricts access for SYSTEM.ADMIN channels to administrators. It is assumed here that some other CHLAUTH rule such as an SSLPEERMAP or an exit validated the connection from administrator.

## Filtering by IP address

- Ability to 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
  - Take precedence over per-channel rules

Authenticating channels and connections

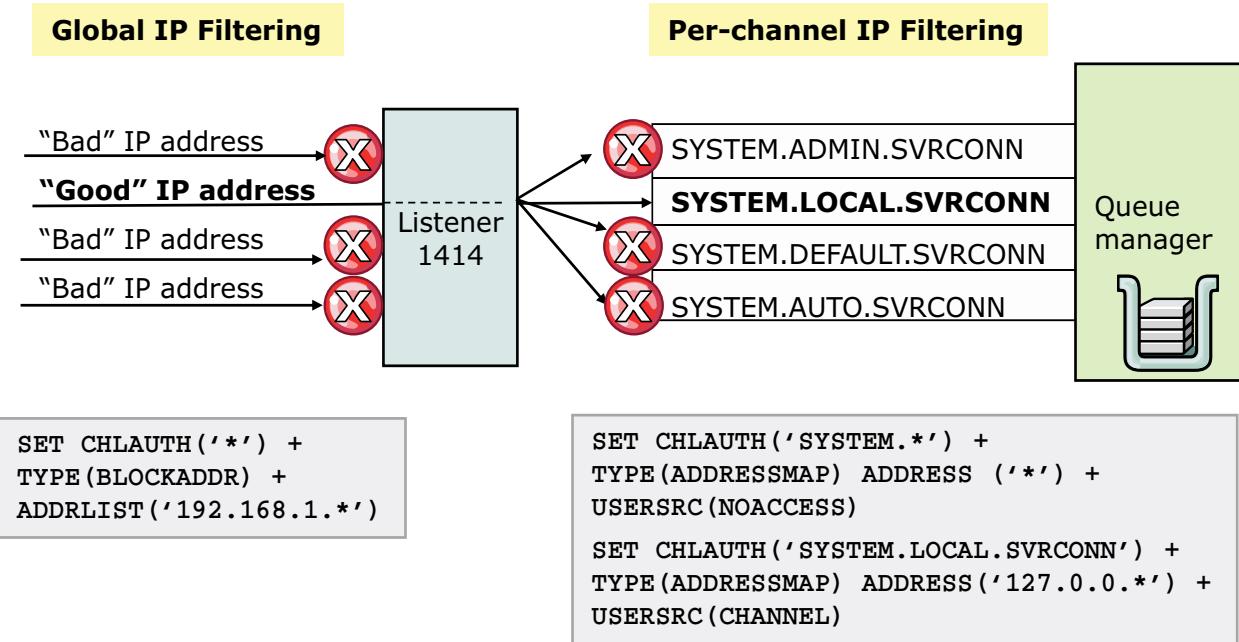
© Copyright IBM Corporation 2017

Figure 3-60. Filtering by IP address

You can filter connection requests based on the IP address of the requester.

Setting TYPE (ADDRESSMAP) in the SET CHLAUTH command maps IP addresses to MCAUSER values. The ADDRESS parameter must accompany the ADDRESSMAP parameter.

## Filtering by IP address: Example



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-61. Filtering by IP address: Example

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 the 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 channels from any IP address because the USERSRC parameter is set to NOACCESS. The NOACCESS option means that inbound connections that match this mapping have no access to the queue manager and the channel ends immediately.

The second command gives the local host (127.0.0) access to the channel IBM 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.

This example is the same example that was shown earlier in this topic to illustrate channel authentication rules.

## Filtering by IP address: Guidelines

- Provide a list of authorized addresses instead of trying to identify all possible unauthorized addresses
- Use CHLAUTH TYPE(BLOCKADDR) to:
  - Temporarily handle transient problems such as a runaway client
  - Handle specific issues such as port scanners, which cause IBM MQ to generate FDC files
- CHLAUTH TYPE(ADDRESSMAP) rules are hierarchical
  - With all other parameters equal, the most specific matching profile name takes precedence
  - Start with “deny-all” policy rule that is followed by specific access rules

Figure 3-62. Filtering by IP address: Guidelines

The figure lists some guidelines for implementing channel authorization that is based on the IP address.

In general, it might be more concise to list the allowed IP addresses in the address list than trying to identify all the possible unauthorized addresses.

## Step-by-step guide to a basic secure setup

- Steps to provide basic security hardening that locks down administrative access
  1. Set MCAUSER
  2. Provision user IDs and groups
  3. Set channels that are not SYSTEM channels
  4. Authorize groups
  5. Set up strong authentication
- Extra configuration is required to provide granularity between different user groups and roles

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-63. Step-by-step guide to a basic secure setup*

The next five pages provide a step-by-step guide to implementing a basic secure setup for an IBM MQ environment. More security is required to provide granularity between different user groups and roles.

## Step 1: Set MCAUSER

- Set MCAUSER to value of `nobody` in:
  - SYSTEM.ADMIN.SVRCCONN
  - SYSTEM.AUTO.\* channels
  - Default channels
- The string `nobody` is important because it has a specific meaning to IBM MQ and to some operating systems

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-64. Step 1: Set MCAUSER

Setting the MCAUSER to `nobody` is a method for using a string for which there is no matching account name. Because it was used by convention for many years, most administrators would not create an account that is called `nobody` on any operating system and they would notice if one appeared. In addition, some varieties of UNIX do not allow an account by that name.

IBM MQ verifies that the string does not match an actual account. It might match an account with no privileges, but then you must make sure that someone does not put the account into a privileged access group like “mqm”.

## Step 2: Provision user IDs and groups

- Provision two low-privileged user IDs and their corresponding private groups
  - One is used for MCA channels  
Example: `mqmmca`
  - One is used for MQI (SVRCONN) channels  
Example: `mqmmqi`
- Configure user IDs so that they cannot be used for logon

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-65. Step 2: Provision user IDs and groups

It is important to note here that the “mqmmca” user ID and its associated group are used to control access on MCA channels, which are channels between queue managers.

The “mqmmqi” user ID and its associated group are used to control access on the MCI channels, which are client channels.

## Step 3: Set channels that are not SYSTEM channels

- Set any channels that are not SYSTEM channels of type RCVR, RQSTR, and CLUSRCVR to use `MCAUSER ('mqmmca')`
- Set any channels that are not SYSTEM channels of type SVRCONN to use `MCAUSER ('mqmmqi')`

Figure 3-66. Step 3: Set channels that are not SYSTEM channels

The SYSTEM channel definitions are created automatically when a queue manager is created. Recall that in step 1, these channels have an MCAUSER value that is set to “nobody.” In the channel definitions that are created later, the MCAUSER field must be set to one of the user IDs that were created according to the instructions for Step 2.

## Step 4: Authorize groups

- Authorize MCA channels group to +put, +inq, and +setall on all the queues it needs
  - Typically all or most of the application queues
  - Does not include command, transmission, or initiation queues
- Authorize MQI channels group to +get, +browse, +put, +inq, and +disp as needed for application use
  - Do not authorize this group to create queues or have PUT access to transmission, initiation, and command queues

Figure 3-67. Step 4: Authorize groups

The next step is to use the `setmqaut` command or IBM MQ Explorer to set the group authorizations on the queues and the MQI channels.

## Step 5: Set up strong authentication

- Expand the basic security configuration
  - Filter by IP address
  - Add user ID blocking
  - Configure dynamic mapping of MCAUSER
- Set up strong authentication to:
  - Allow for full remote administration by IBM MQ administrators
  - Control remote access for non-administrative interactive users and applications

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-68. Step 5: Set up strong authentication*

The final step is to strengthen the security by using channel authentication rules to add IP address or host name filtering, user ID blocking, and dynamic of the MCAUSER.

## 3.5. Channel exits

## Channel exit programs

- Called at defined places in the processing sequence of an MCA
  - Users and vendors can write their own channel exit programs
  - Some channel exit programs are supplied by IBM
- Can be on sending MCA, receiving MCA, or both
- In most cases, programs are written in C
  - On Windows, the exit must be a .dll file
  - Can use exits that are written in Java, C, or C++ when IBM MQ classes for JMS applications use channel security and send and receive exits on the MQI channel that starts when the application connects to a queue manager
- Example application:  
If channel authentication records are not suitable, you can use channel exits for added security

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-69. Channel exit programs

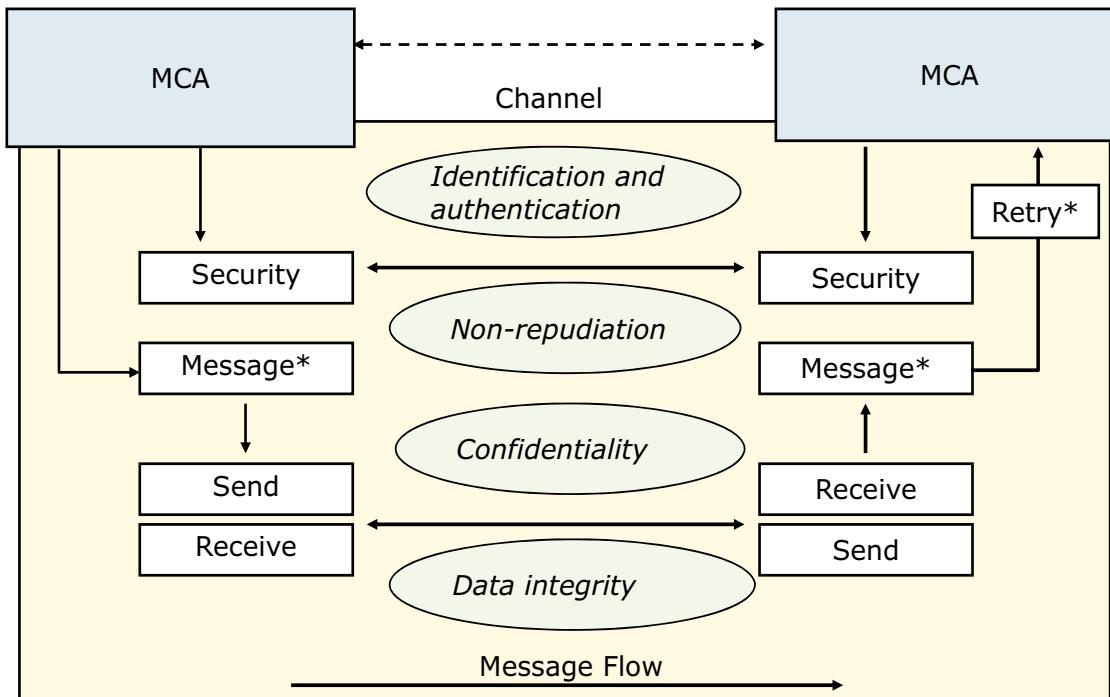
IBM MQ has many options for providing channel security. If the channel security that the IBM MQ provides is not adequate, there are several exit points where you can tailor the behavior of an MCA to your requirements.

All the channel exits normally must be supplied in pairs. That means that if the exit is activated on one side of the channel, it also must be activated on the other side.

Exits are written in C. In Windows, the exit must be a .dll file.

The IBM MQ classes for JMS applications can use channel security and send and receive exits on the MQI channel that starts when the application connects to a queue manager. The application in this case can use exits that are written in Java, C, or C++.

## Channel exits for link level security



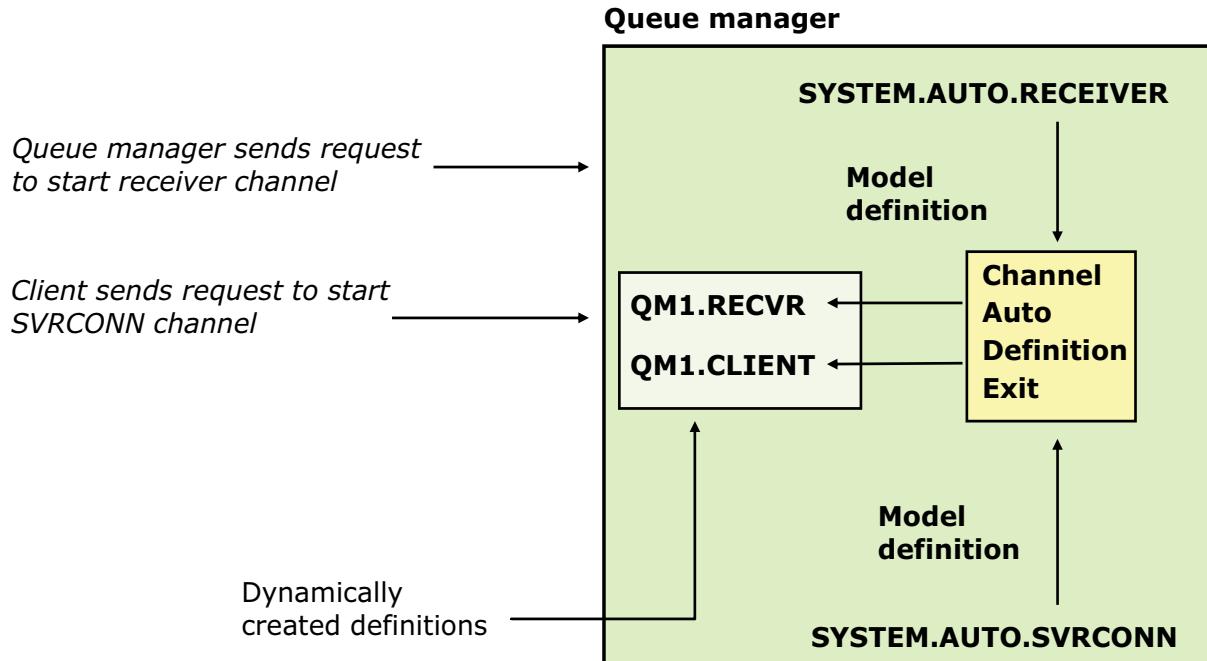
\* Not available on client and server connection

Figure 3-70. Channel exits for link level security

There are several types of channel exit program, but only four have a role in providing link level security:

- **Security exit** is called during the channel setup. It allows for the addition of code that can check the security credentials of the partner MCA. This process is known as authentication. Normally, the receiving side starts the security exit data flow, thus forcing the requester side to act appropriately, or the channel is not allowed to start. Security data flows run before any messages are transmitted.
- **Message exit** is called on the sending side after a message is read from the transmission queue and on the receiving side before a message is put onto a destination queue. The message exit receives the entire message, including the IBM MQ message descriptor (MQMD) structure, and it can change the message and its descriptor, if required. The typical purpose of the message exit is the encryption and decryption of data or the logging of messages to ensure that a particular message was sent or received.
- **Send and receive exits** are called before a buffer of data is transmitted over the network or after one is received from the network. These exits can be called multiple times for a single message. Their typical purpose is data compression and decompression.
- **Retry exit** is called when a received message cannot be delivered to the target queue that the transmission queue header specifies, which includes reasons that are of a temporary nature.

## Channel auto-definition exit



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-71. Channel auto-definition exit

The channel auto-definition exit is called when a request is received to start a cluster-sender channel. It is also called when starting a cluster-receiver channel.

You can use the channel auto-definition exit to modify the supplied default definition for an automatically defined receiver or server-connection channel, SYSTEM.AUTO.RECEIVER, or SYSTEM.AUTO.SVRCONN. The exit might change most of these parameters.

The channel auto-definition exit is the property of the queue manager, not the individual channel. It must be named in the queue manager definition. The regular command is DEFINE or ALTER QMGR CHADEXIT(*programName*).

## Channel exits on message channels

Channel type	Message exit	Message retry exit	Receive exit	Security exit	Send exit	Auto-definition exit
Sender	Yes		Yes	Yes	Yes	
Server	Yes		Yes	Yes	Yes	
Receiver	Yes	Yes	Yes	Yes	Yes	Yes
Requester	Yes	Yes	Yes	Yes	Yes	Yes
Cluster-sender	Yes		Yes	Yes	Yes	
Cluster-receiver	Yes	Yes	Yes	Yes	Yes	Yes
Client-connection			Yes	Yes	Yes	
Server-connection			Yes	Yes	Yes	Yes

Authenticating channels and connections

© Copyright IBM Corporation 2017

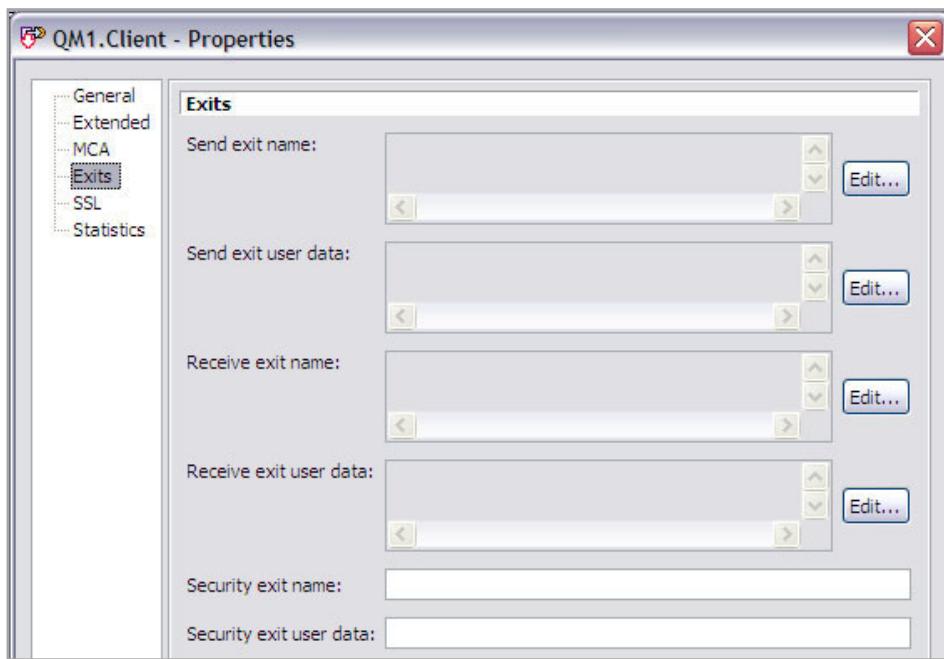
Figure 3-72. Channel exits on message channels

The figure identifies the IBM MQ exits that are available for each of the channel types.

Channel exits include message exits, message-retry exits, receive exits, security exits, send exits, and auto-definition exits.

Channel types include sender channels, server channels, receiver channels, requester channels, cluster-sender channels, cluster-receiver channels, client-connection channels, and server-connection channels.

## Channel exit configuration



```
ALTER CHANNEL('QM1.CLIENT') CHLTYPE(SVRCONN) +
SCYEXIT(MySecExit) SCYDATA(MAXIMUM)
```

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-73. Channel exit configuration*

For IBM MQ on UNIX, Linux, and Windows, you can specify the channel exit program names as properties of a channel definition that are set when you define or alter the channel.

The figure shows how IBM MQ Explorer or MQSC can be used to define or modify the channel to implement a security exit or message exit.

The security exit can specify a single program. The message exit and the send and receive exits can specify the name of more than one exit program. When multiple programs are specified, a comma separates each program name. However, the total number of characters that are specified must not exceed 999.

For the security exit, a text string of up to 32 characters in length can be specified. This string is passed to the exit program when it starts. For the types of exit programs, you can specify data for more than one exit program by separating multiple strings with a comma. The total length of the field must not exceed 999.

## Location of exit modules

- Default exit paths
  - Windows: <install\_location>\exits
  - UNIX and Linux: /var/mqm/exits (32-bit channel exits)  
/var/mqm/exits64 (64-bit channel exits)
- If required, default exit paths can be changed
  - On the server, paths are in the **ExitPath** stanza the **qm.ini** file
  - On a client, paths are in the **ClientExitPath** stanza in the IBM MQ client configuration file **client.ini**

Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-74. Location of exit modules

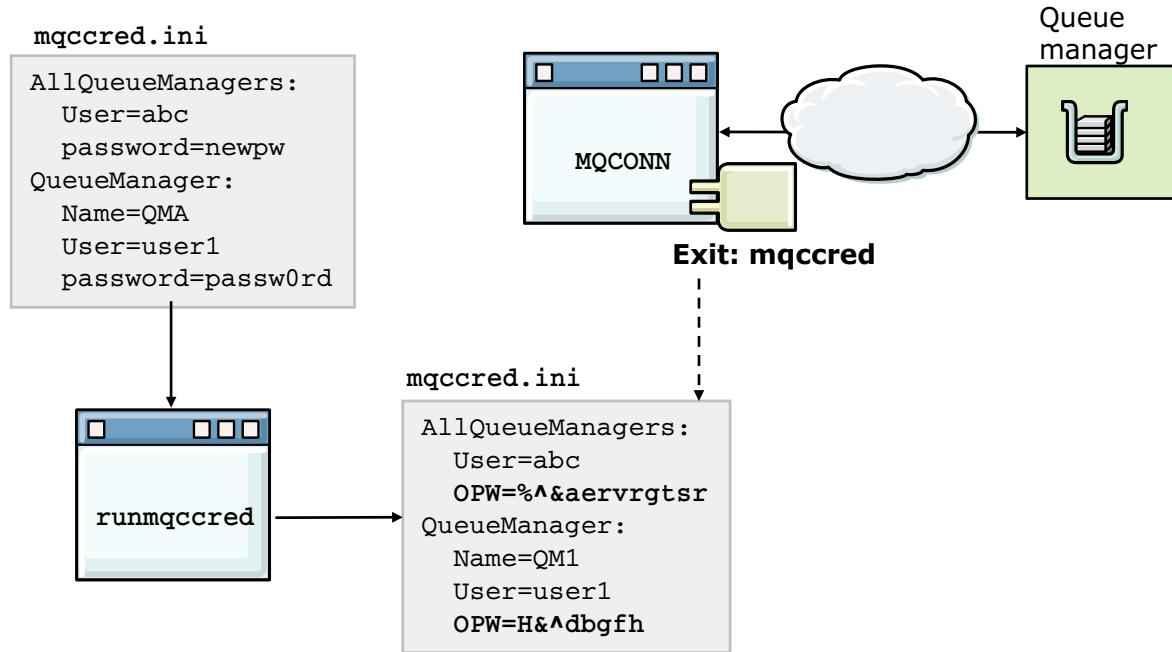
On UNIX, Linux and Windows systems, a client configuration file is added to the system during installation of the MQI client. A default path for location of the channel exits on the client is defined in this file, by using the stanza:

```
ClientExitPath:  
ExitsDefaultPath=string  
ExitsDefaultPath64=string
```

Where, **string** is a file location in a format appropriate to the operating system.

## Client-side security exit

- IBM MQ provides a client-side security exit for setting the user ID and password for IBM MQ user authentication



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-75. Client-side security exit

IBM MQ provides a client-side security exit to set the user ID and password. You might want to use this exit instead of changing an application.

The exit runs at the client-connection end of the channel and gets the user ID and the password from a file. The operating system controls the permissions to this file. If the exit discovers that the file permissions are too open, it causes a failure so that it is evident that this file is not protected.

IBM MQ provides a tool that obfuscates the passwords in this file from casual browsers. The algorithm for this obfuscation is not published, and neither is the source of the exit.

## Unit summary

- Determine the current level of authentication that is enabled on a queue manager and a connection
- Add authentication to a channel
- Add authentication to a connection
- Identify and fix channel authentication and connection authentication problems
- Implement a channel exit program for securing messaging channels

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-76. Unit summary*

## Review questions

1. True or False: Authorization is the process of determining the user ID of the user.
2. True or False: If you have MCAUSER set to blank on any inbound channel definitions, then you have a serious security exposure.
3. After entering the following command, what access does the “payroll” group have to the queue:  
`setmqaut -m DEMO -n DEMO.QUEUE -t queue -g payroll +put`
  - A. No access.
  - B. PUT access.
  - C. Full access.
  - D. PUT access plus any other access that a previous `setmqaut` command granted.



Authenticating channels and connections

© Copyright IBM Corporation 2017

Figure 3-77. Review questions

Write your answers here:

- 1.
- 2.
- 3.

## Review answers

- True or False: Authorization is the process of determining the user ID of the user.

**The answer is False. Authorization is the control access to resources on a per-ID, per-role, or per-profile basis**



- True or False: If you have MCAUSER set to blank on any inbound channel definitions, then you have a serious security exposure.

**The answer is True.**

- After entering the following command, what access does the “payroll” group have to the queue:

`setmqaut -m DEMO -n DEMO.QUEUE -t queue -g payroll +put`

- A. No access.
- B. PUT access.
- C. Full access.
- D. PUT access plus any other access that a previous `setmqaut` command granted.

**The answer is D.**

## Exercise: Implementing connection authentication

Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-79. Exercise: Implementing connection authentication*

In this exercise, you modify an IBM MQ network to add connection authentication security.

## Exercise objectives

- Check locally bound connections
- Check client connections
- Configure the authentication failure delay



Authenticating channels and connections

© Copyright IBM Corporation 2017

*Figure 3-80. Exercise objectives*

For detailed instructions, see the Course Exercises Guide.

---

# Unit 4. Implementing workload management in an IBM MQ cluster

## Estimated time

01:00

## Overview

In this unit, you learn how to use the IBM MQ cluster configuration options to balance and manage the workload.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Describe how queue manager clusters assist with workload management
- Describe the attributes that affect the workload balancing algorithm
- Use multiple transmission queues to separate the workload

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

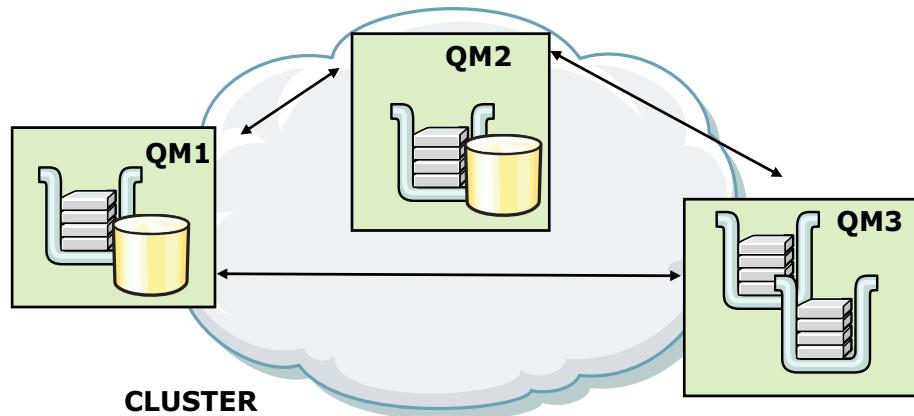
*Figure 4-1. Unit objectives*

## 4.1. Queue manager cluster concepts review

This topic reviews the concepts that you learned about IBM MQ clusters in the prerequisite courses for this course.

## What is a queue manager cluster?

- Group of queue managers that make the queues that they host available to other queue managers in the cluster without the need for explicit channel definitions, remote-queue definitions, or transmission queues for each destination



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

*Figure 4-2. What is a queue manager cluster?*

A cluster is a collection of queue managers that can be on different servers and operating systems, and typically serve a common application.

Every queue manager can make the queues that they host available to every other queue manager in the cluster, without the need for (remote) queue definitions.

Cluster-specific objects remove the need for explicit channel definitions and transmission queues for each destination queue manager.

The queue managers in a cluster often assume the role of a client or a server. The servers host the queues that are available to the members of the cluster and applications that process these messages and generate responses. The clients PUT messages to the server queues and might receive response messages.

Queue managers in a cluster normally communicate directly with each other, although typically, many of the client systems never need to communicate with other client systems.

## Benefits of clustering

- Improved performance and distributed workload of message traffic
  - If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to
  - IBM MQ uses a cluster workload management algorithm and cluster workload-specific attributes to determine the optimal queue manager
- Greater resilience to system failures through redundancy
- Simplified administration and flexible connectivity
  - Not necessary to explicitly define channels, remote-queue definitions, or transmission queues for every destination in the cluster

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

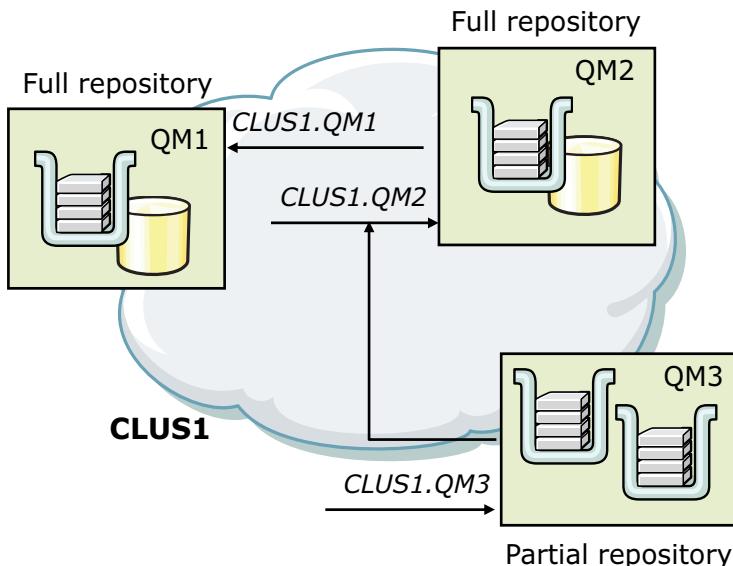
Figure 4-3. Benefits of clustering

Clusters simplify the administration of IBM MQ networks, which usually require many object definitions for channels, transmission queues, and remote queues.

Clusters can be used to distribute the workload of message traffic across queues and queue managers in the cluster. Such distribution allows the message workload of a single queue to be distributed across equivalent instances of that queue that is on multiple queue managers.

The distribution of the workload can be used to achieve greater resilience to system failures, and to improve the scaling performance of active message flows in a system.

## Overview of cluster components



- *Full repository queue manager* hosts a complete set of information about every queue manager in the cluster
- *Partial repository queue manager* contains information about those queue managers with which it needs to exchange messages
- *Cluster queues*
- *Cluster channels*
  - Cluster-receiver
  - Cluster-sender

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-4. Overview of cluster components

A cluster can contain two or more queue managers. The example cluster in the figure contains three queue managers in a cluster that is named CLUS1: QM1, QM2, and QM3.

Two of queue managers, QM1 and QM2, are configured as full repository queue managers. Full repository queue managers contain information about all the queue managers in the cluster. The repositories are represented in the figure by the shaded cylinders. The other queue manager in this cluster is a partial repository queue manager. A partial repository queue manager holds records for local objects and remote objects that are used locally.

All the queue managers in the cluster can host local queues that are accessible to any other queue manager in the cluster. These queues are called cluster queues. As with distributed queuing, an application uses an MQPUT call to put a message on a cluster queue at any queue manager. An application uses the MQGET call to retrieve messages from a cluster queue on the local queue manager.

Each queue manager has a cluster-receiver channel for the receiving end of a channel. Include the name of the cluster and the name of the queue manager that is the destination of that channel in each channel name. A cluster-receiver channel is similar to a receiver channel used in distributed queuing, but in addition to carrying messages, this channel also carries information about the cluster.

Each queue manager also has a cluster-sender channel definition for the sending end of a channel. A cluster-sender channel is similar to a sender channel used in distributed queuing, but in addition to carrying messages, this channel also carries information about the cluster.

## Considerations for a full repository

- Ensure that full repositories are highly available
  - Avoid single point of failure
  - Define two full repositories to improve availability
  - Locate on highly available servers
- Full repositories must be fully interconnected
  - Define a cluster-receiver channel at each queue manager
  - Define one cluster-sender channel between the full repository queue managers

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

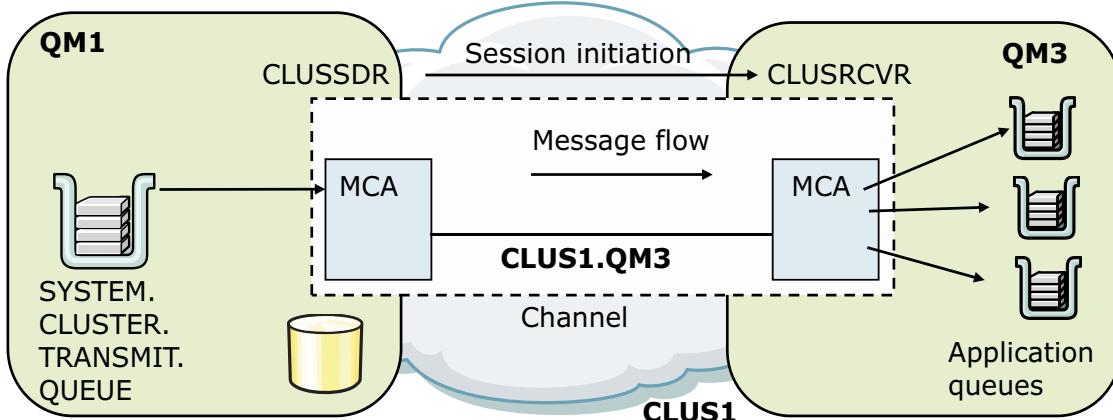
*Figure 4-5. Considerations for a full repository*

It is best to have two full repositories in a cluster so that the cluster has a backup repository. You interconnect the full repository queue managers by defining cluster-sender channels between them.

When a queue manager sends out information about itself or requests information about another queue manager, the information or request is sent to the full repositories. A full repository that is named on a cluster-sender channel handles the request whenever possible, but if the chosen full repository is not available another full repository is used. When the first full repository becomes available again, it collects the most recent information from the other full repository so that they contain the same information.

## Cluster channels

- Cluster-receiver (CLUSRCVR)
  - Define one for each cluster queue manager
  - Enables queue managers to automatically define corresponding cluster-sender channels
- Cluster-sender (CLUSDR)
  - Explicitly define one between the full repositories and one from each partial repository to one full repository
  - Other cluster-sender channels are defined automatically



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-6. Cluster channels

In a cluster, each queue manager has a cluster-receiver channel on which it can receive messages and information about the cluster. You must manually define a cluster-receiver channel on each queue manager in the cluster.

In a cluster, each queue manager has a cluster-sender channel on which it sends cluster information to one of the full repository queue managers. Queue managers can also send messages to other queue managers on cluster-sender channels. Manually define a cluster-sender channel between the full repository queue managers and from each partial repository queue manager to one of the full repository queue managers. The other cluster-sender channels are defined automatically.

## Steps to set up a basic cluster

1. Decide how many queue managers to define (organization of the cluster and naming conventions)
2. Establish which queue managers are to hold full repositories
3. Alter the full repository queue manager definitions to add the repository information
4. Define the cluster-receiver (CLUSRCVR) channels
5. Define the cluster-sender (CLUSSDR) channels
6. Define the cluster queues
7. Verify and test the cluster

MQSC commands  
or  
IBM MQ Explorer Create Cluster wizard

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-7. Steps to set up a basic cluster

The figure lists the basic steps for setting up a cluster.

You can also use MQSC or one of the wizards that is supplied with IBM MQ Explorer to create a cluster.

To use IBM MQ Explorer, right-click the **Queue Manager Clusters** folder, click **New > Queue Manager Cluster**, and follow the instructions in the wizard.

## Controlling clusters with MQSC commands

- **DISPLAY CLUSQMGR**      Display cluster information
- **SUSPEND QMGR**      Remove a queue manager from a cluster
- **RESUME QMGR**      Reinstate a queue manager to a cluster
- **REFRESH CLUSTER**
  - Discard locally held data about a cluster
  - Objects are rebuilt according to REPOS parm; enforces a cold start of the queue manager
- **RESET CLUSTER**
  - Sent by a repository queue manager
  - Forcibly removes a queue manager from a cluster
  - Only used in exceptional circumstances

[Implementing workload management in an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

*Figure 4-8. Controlling clusters with MQSC commands*

The figure lists the commands that can be used to control clusters:

### DISPLAY CLUSQMGR

If you enter this command from a queue manager with a full repository, the information that is returned pertains to every queue manager in the cluster. If you enter this command from a partial repository queue manager, the information that is returned pertains only to the queue managers in which it has an interest.

### SUSPEND QMGR and RESUME QMGR

Use the SUSPEND QMGR command and RESUME QMGR command to remove a queue manager from a cluster temporarily and then to reinstate it by using the RESUME command. The SUSPEND command does not completely stop messages from being sent to a queue manager. If the suspended queue manager is the only queue manager with an available copy of a queue, the messages are sent to the queue on the suspended queue manager.

### REFRESH CLUSTER

You can enter REFRESH CLUSTER(\*) to refresh the queue manager in all of the clusters it is a member of. If used with REPOS(YES), this command forces the queue manager to restart its search for full repositories from the information in the local CLUSSDR definitions. The search

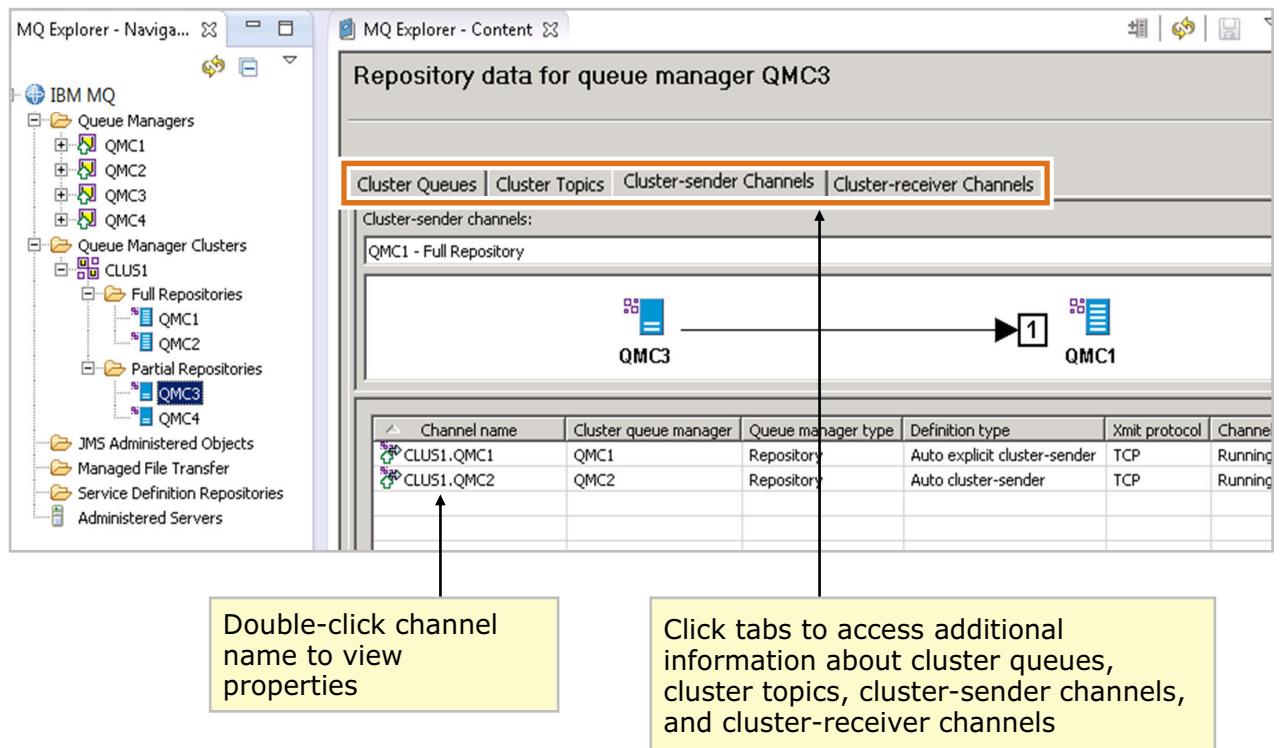
information in the local cluster-sender channel definitions. The search occurs even if the cluster-sender channel connects the queue manager to several clusters.

#### RESET CLUSTER

In an emergency where a queue manager is temporarily damaged, you might want to inform the rest of the cluster before the other queue managers try to send it messages. The RESET CLUSTER command can be used to remove the damaged queue manager. The RESET CLUSTER command can also be used to remove a queue manager that should not belong to a cluster, perhaps because of a security problem. If necessary, you can use the REFRESH CLUSTER command to reverse the effect of RESET CLUSTER and put the queue manager back into the cluster.

# IBM Training

## Monitoring clusters with IBM MQ Explorer



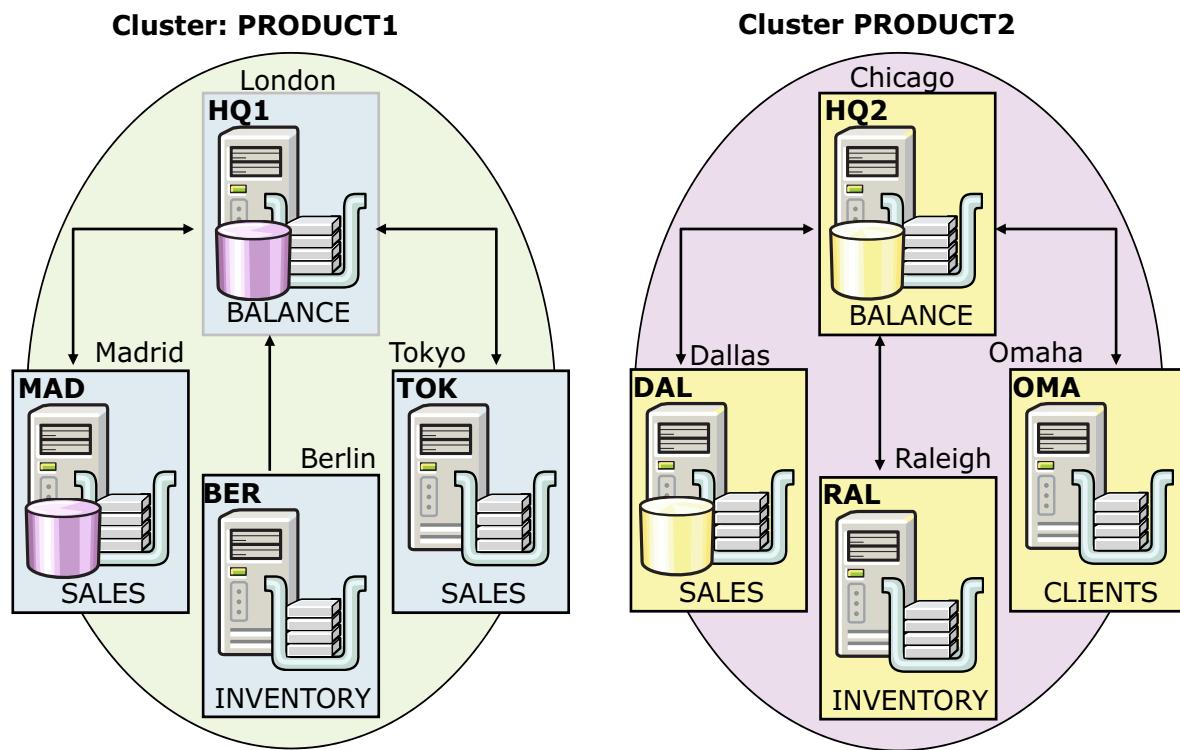
Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-9. Monitoring clusters with IBM MQ Explorer

When a cluster queue manager is selected, tabs in the IBM MQ Explorer **Content** view can be selected to access additional information about the cluster. Double-click an object, such as the channel name, to view its properties.

## Independent clusters



Implementing workload management in an IBM MQ cluster

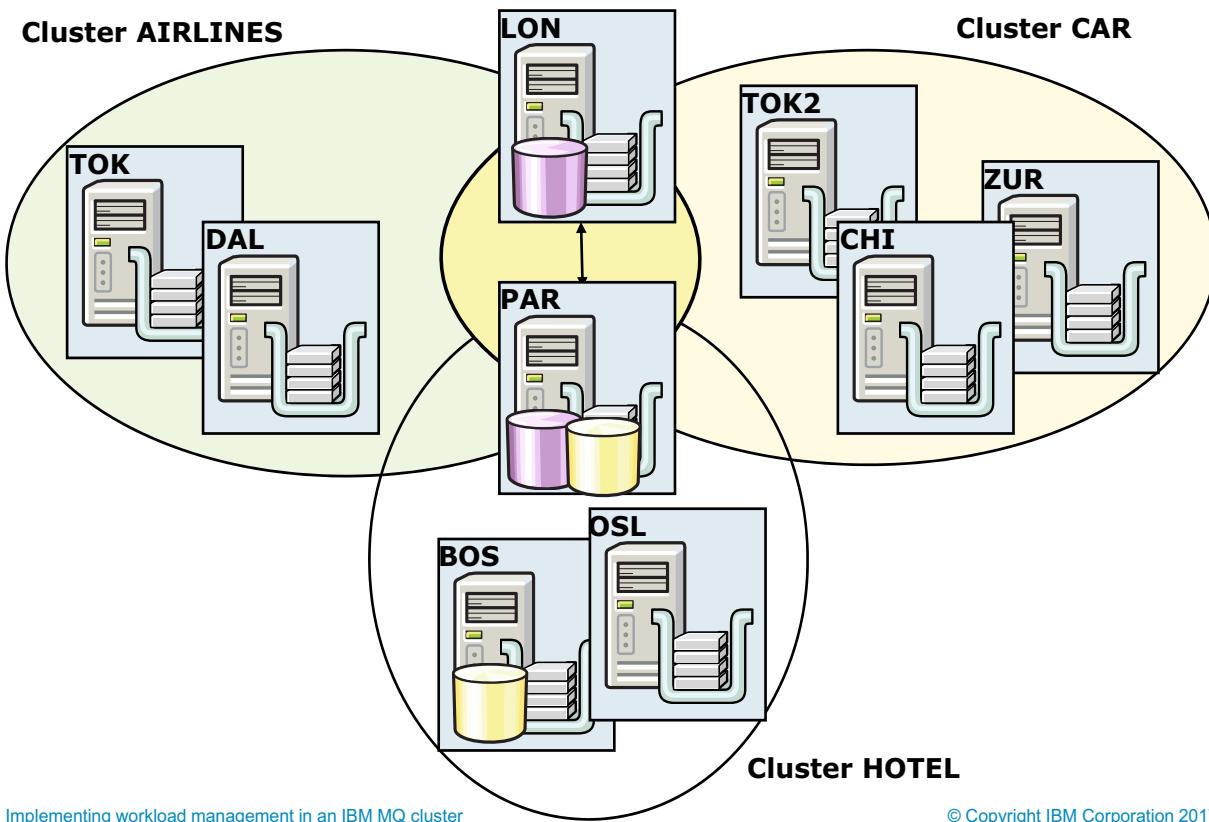
© Copyright IBM Corporation 2017

Figure 4-10. Independent clusters

Imagine a large organization that is composed of many companies with each one acting in different business areas. You might choose to group the queue managers into a separate cluster for each company. Each company can manage, see, and handle just its own data and definitions and can implement a different security policy.

The figure shows two clusters with the same queue names. The clusters can have similar configurations but the messages are processed separately and they can have different security policies.

## Overlapping clusters



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-11. Overlapping clusters

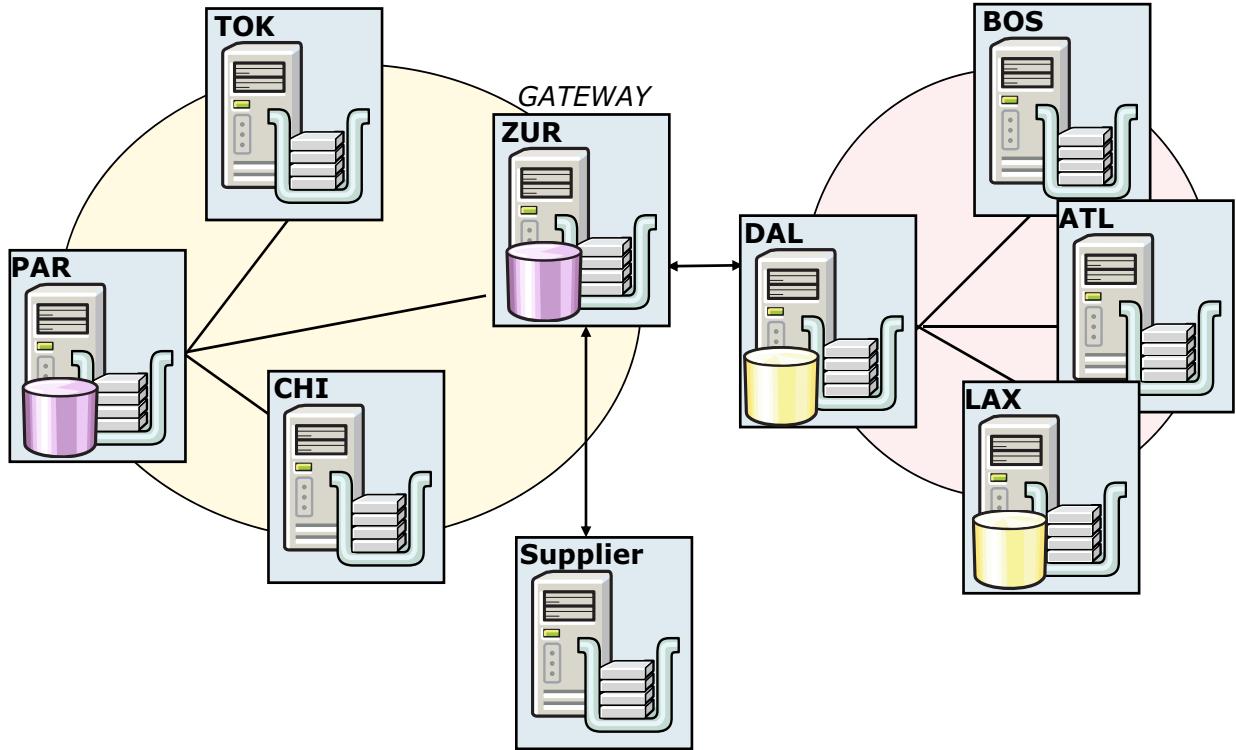
You can group all your queue managers into multiple smaller clusters with one or more queue managers in each acting as a bridge, overlapping another cluster.

Overlapping clusters provide visibility of IBM MQ definitions that are made in the bridge queue manager to the overlapped clusters.

You might overlap clusters for a number of reasons:

- To restrict the visibility of the queues and queue manager names across the clusters
- To create classes of service
- To allow different applications to be administered separately
- To allow different organizations to have their own administration

## Cluster gateways



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-12. Cluster gateways

An application on a queue manager inside a cluster might need to communicate with a queue manager that is outside of the cluster. A good example is when the destination queue manager is in a different enterprise. In such cases, to communicate with a queue manager outside the cluster, one or more queue managers inside the cluster must act as a gateway.

## Cluster troubleshooting

- Check the channels
  - All cluster channels are paired
  - Every queue manager in the cluster has a manually defined cluster-receiver channel and the channel is running
  - Channels are running: `DISPLAY CHSTATUS (*)`
  - Every full repository has a cluster-sender channel to every other full repository and the channel is running
- Check the queue managers
  - All queue managers in the cluster are aware of all the full repositories: `DISPLAY CLUSQMGR (*) QMTYPE`
  - Intended full repository queue managers are defined as full repositories and are in the correct cluster: `DISPLAY QMGR REPOS REPOSNL`
- Check the queues
  - Messages are not building up on transmission queues
  - Messages are not building up on system queues

Figure 4-13. Cluster troubleshooting

This figure summarizes cluster troubleshooting tips that help you to detect and resolve problems when you use queue manager clusters.

## 4.2. Workload management in queue manager clusters

## Workload balancing in queue manager clusters

- If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to
- IBM MQ uses the cluster workload management algorithm and cluster workload-specific attributes to determine the best queue manager
- Administrator can specify cluster workload channel attributes on the cluster-receiver channels at the target queue managers

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-14. Workload balancing in queue manager clusters

With an IBM MQ cluster, more than one queue manager can host an occurrence of the same queue. So, two or more queue managers can be clones of each other, capable of running the same applications and with local definitions of the same queues.

You might use this technique to implement queues to increase the capacity available to process messages. You might also use this technique to implement queues to introduce an ability to fail over work from one server to another and improve availability of the service.

This architecture can also be used to spread the workload between queue managers, if applications allow it to do so. A built-in workload management algorithm determines the remote queue manager when multiple choices exist, which are based on availability and channel priorities. A local occurrence takes precedence by default.

## Cluster workload management options

- If cluster contains more than one instance of the same queue, IBM MQ uses the cluster workload management algorithm and cluster workload-specific attributes to determine target queue
  - Application binding options
  - Use-queue
  - Channel rank and queue rank
  - Channel and queue priority
  - Most recently used
  - Channel weighting
  - Multiple transmission queues

Implementing workload management in an IBM MQ cluster

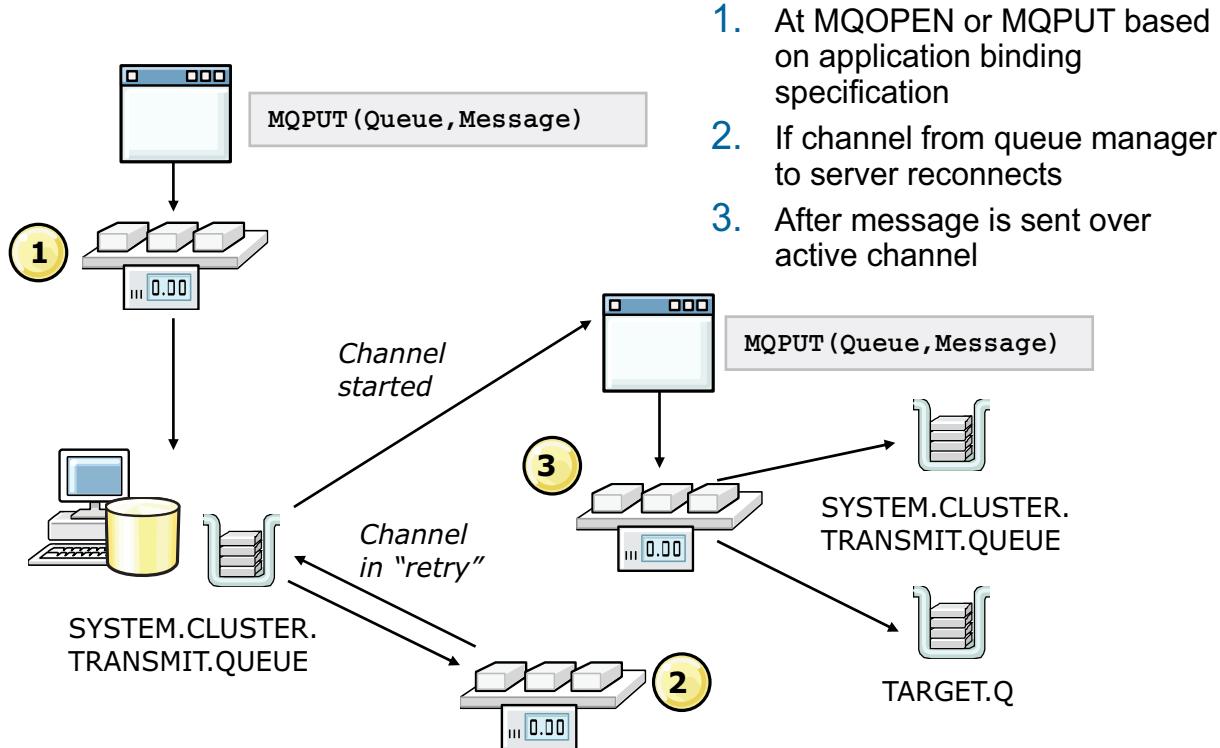
© Copyright IBM Corporation 2017

*Figure 4-15. Cluster workload management options*

Soft-coded cluster workload management features are required to remove the need for cluster workload exits in some situations, allow more flexible interconnected clusters, and provide greater scalability.

The figure lists the cluster workload parameters that can be specified on channel and queue definitions. Each of these parameters is described in detail in this topic.

## When workload balancing can occur



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-16. When workload balancing can occur

In a typical queue manager cluster, a message is put to a queue manager in the cluster. In some cases, the message is destined for a back-remote queue manager that is also in the cluster. In this configuration, workload balancing can occur in three points.

1. Workload balancing can occur at either MQOPEN or MQPUT, depending on the bind options that are associated with the message affinity.
  - If “bind on open” is specified, then all messages go to the same destination. After the destination is chosen at MQOPEN time, no more workload balancing occurs on the message.
  - If “bind not fixed” is specified, the messages can be sent to any of the available destinations. The decision where to send the message is made at MQPUT time. However, the destination can change while the message is in transit.

In either case, the message is put on the local cluster queue instance if one is available on the application’s queue manager. If a local cluster queue instance is not available, the message is put on the **SYSTEM.CLUSTER.TRANSMIT.QUEUE**.

2. Workload balancing can occur if a channel from a queue manager to a back-end server is attempting to connect again. In this case, any “bind not fixed” messages that are waiting to be sent go through workload balancing again to see whether a different destination is available.

3. After a message is transported over an active, started channel, the channel calls MQPUT to put the message to the target queue and the workload algorithm is called again.

## Application binding overview

- When multiple queues with the same name are advertised on a queue manager cluster, applications can choose one of the following options on MQOPEN call:
  - Send all messages from this application to a single queue instance
  - Request that a 'group' of messages go to the same queue instance
  - Allow workload management algorithm to select the most suitable destination on a per message basis
  - Use queue binding option instead of specifying binding in the MQOPEN call

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-17. Application binding overview

An option on the MQOPEN call, allows the application to specify that the target queue manager must be fixed if a cluster contains multiple instances of the same queue. That is, all messages that are put to the queue that specifies the object handle that the MQOPEN call returns must be directed to the same queue manager, by using the same route.

## Application binding options

- Bind on open
  - Messages are bound to a destination chosen at MQOPEN
  - All messages that are put by using open handle are bound to the same destination
- Bind not fixed
  - Each message is bound to a destination at MQPUT
  - Workload balancing is done on every message
  - Does not create affinities
- Bind on group
  - Messages are bound to a destination chosen at MQOPEN
  - All messages in a message group that are put to a queue by using MQPUT are allocated to the same destination instance
  - Ensures that messages in a group are sent to the same destination
- Bind as queue definition
  - Queue DEFBIND attribute specifies the binding option instead of the application

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

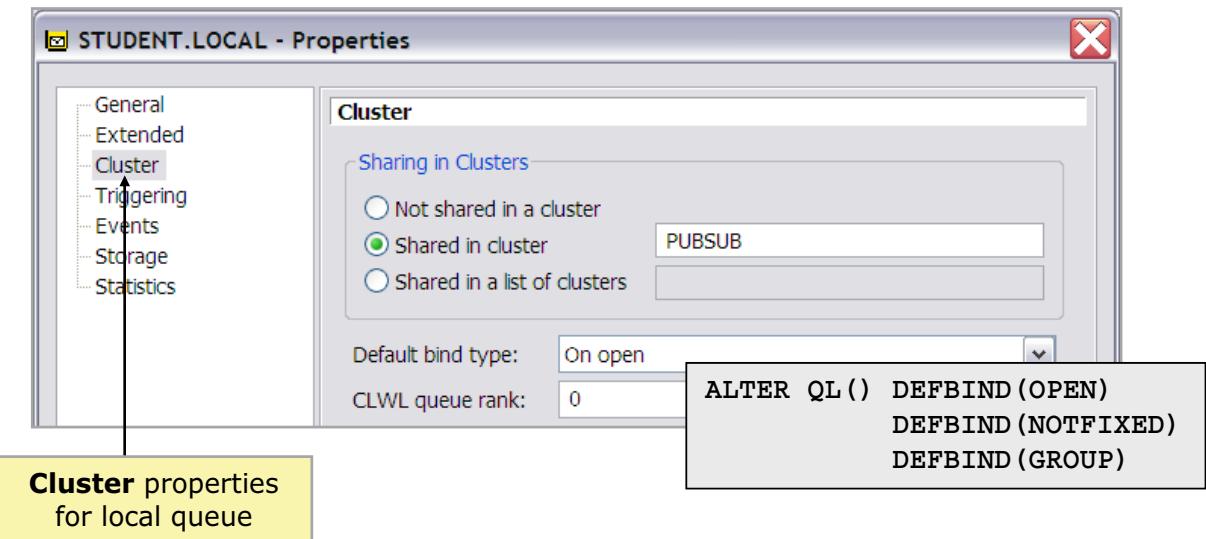
Figure 4-18. Application binding options

If it is not necessary for the application to force all the messages to be written to the same destination, specify BIND\_NOT\_FIXED on the MQOPEN call. This configuration selects a destination at MQPUT time, that is, on a message-by-message basis.

If the application does not specify BIND\_ON\_OPEN, BIND\_NOT\_FIXED, or BIND\_ON\_GROUP, the default option is BIND\_AS\_Q\_DEF. Using BIND\_AS\_Q\_DEF takes the binding that is used for the queue handle from the DEFBIND queue attribute.



## Setting default bind type for a queue



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-19. Setting default bind type for a queue

The figure shows how to configure the cluster workload Default bind type (DEFBIND) attribute for a queue in the queue.

## Cluster workload management attributes

- Queue manager
  - Use-queue (CLWLUSEQ)
  - Most recently used channels (CLWLMRUC)
- Queues
  - Priority (CLWLPRTY)
  - Rank (CLWLRANK)
  - Use-queue (CLWLUSEQ)
- Channels
  - Priority (CLWLPRTY)
  - Rank (CLWLRANK)
  - Weight (CLWLWGHT)
  - Cluster-receiver channel priority (NETPRTY)

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-20. Cluster workload management attributes

This figure lists the attributes that are used when IBM MQ determines the final destination for messages that are put onto cluster queues.

The settings that are applied to the attributes for queues, queue managers, and channels influence the rules that are used in the cluster workload management algorithm.

## Cluster workload management algorithm: Summary

1. Queue manager builds a list of possible destinations
2. If queue is specified, eliminate the following queues:
  - Queues with PUT(DISABLED)
  - Remote instances of queues that do not share a cluster with the local queue manager
  - Remote CLUSRCVR channels that are not in the same cluster as the queue
3. If queue manager name is specified, eliminate the following queues:
  - Remote instances of queues that do not share a cluster with the local queue manager
  - Remote CLUSRCVR channels that are not in the same cluster as the queue or topic
4. If available, use the local queue instance unless overridden by use-queue
5. Evaluate channel rank
6. Evaluate channel status
7. Evaluate cluster-receiver channel priority
8. Evaluate channel priority
9. Evaluate queue priority
10. Evaluate most recently used channels
11. Evaluate least recently used with channel weight

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

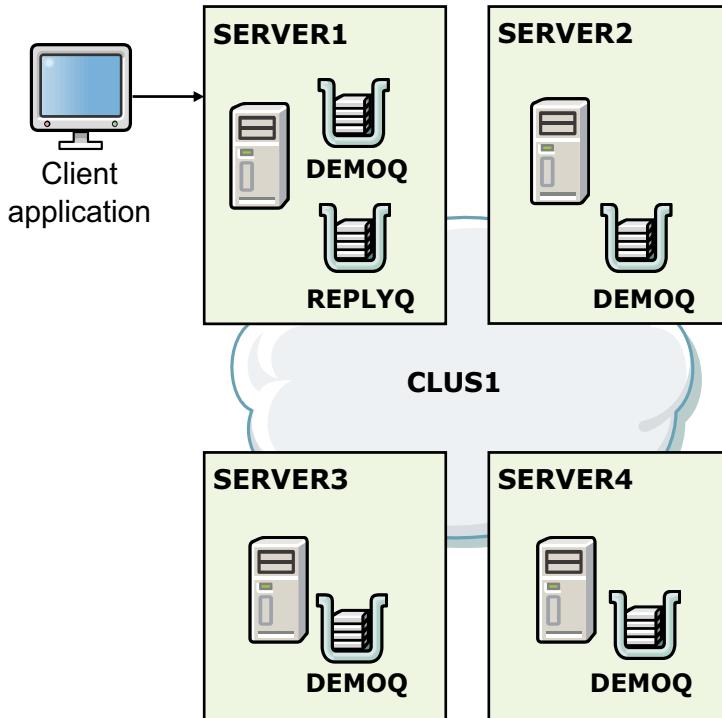
Figure 4-21. Cluster workload management algorithm: Summary

Every time a message is put on a queue, the cluster workload management algorithm runs once.

This figure summarizes the workload management algorithm.

The distribution of user messages is not always exact. The administration and maintenance of the cluster causes messages to flow down channels, which can result in an apparent uneven distribution of user messages. For this reason, do not rely on the exact distribution of messages during workload balancing.

## Queue manager cluster scenario



- Four queue managers, each hosting the local cluster queue DEMOQ
- Client application is connected to SERVER1
  - MQPUT request to DEMOQ
  - MQGET reply from REPLYQ
- Server application is connected to each queue manager
  - MQGET request from DEMOQ
  - MQPUT reply to REPLYQ

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-22. Queue manager cluster scenario

In this scenario, the cluster contains four queue managers that are on different servers. The client application is connected to SERVER1. It puts request messages to DEMOQ (a cluster queue that is all four queue managers host).

When a message arrives on a DEMOQ, a server application reads it and then puts a reply message to the reply-to queue REPLYQ on SERVER1.

## Use-queue basics

- Allows remote queues to be chosen when a local queue exists
- On local queue, applies when an application or a channel that is not a cluster channel puts the message

```
DEFINE QL(CLUS.Q1) CLUSTER(CLUS1) CLWLUSEQ( )
```

- LOCAL = If a local queue exists, choose it
- ANY = Choose either local or remote queues
- QMGR = Use the use-queue value from the queue manager

- On queue manager when **CLWLUSEQ** queue attribute is set to **QMGR**, specifies whether a local instance of a queue is given preference as a destination over other instances of the queue in a cluster

```
ALTER QMGR CLWLUSEQ( )
```

- LOCAL = If the queue specifies CLWLUSEQ (QMGR) and a local queue exists, choose it
- ANY = If the queue specifies CLWLUSEQ (QMGR), choose either local or remote queues

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

*Figure 4-23. Use-queue basics*

The use-queue attribute, CLWLUSEQ, specifies the behavior of an MQPUT operation when the cluster contains a local instance and at least one remote instance of a cluster queue. The exception is in cases where the MQPUT originates from a cluster channel.

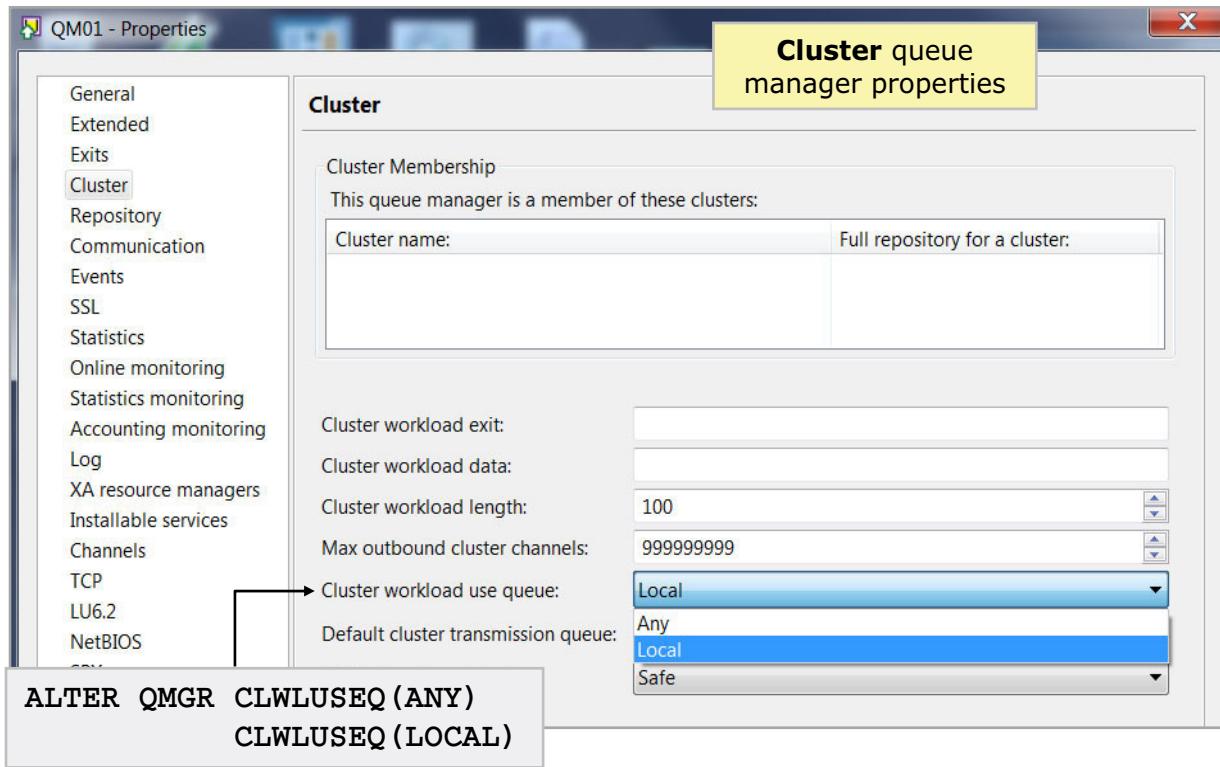
The valid options for the use-queue attribute on a queue are LOCAL, ANY, and QMGR.

- If you specify QMGR for the attribute value on the queue, the CLWLUSEQ parameter of the queue manager definition determines the behavior. This parameter is valid only for local queues.
- If you specify ANY, the queue manager treats the local queue as another instance of the cluster queue for the purposes of workload distribution.
- If you specify LOCAL, the local queue is the only target of the MQPUT operation.

The valid options for the use-queue attribute on a queue manager are LOCAL and ANY.

# IBM Training

## Configuring CLWLUSEQ on a queue manager



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

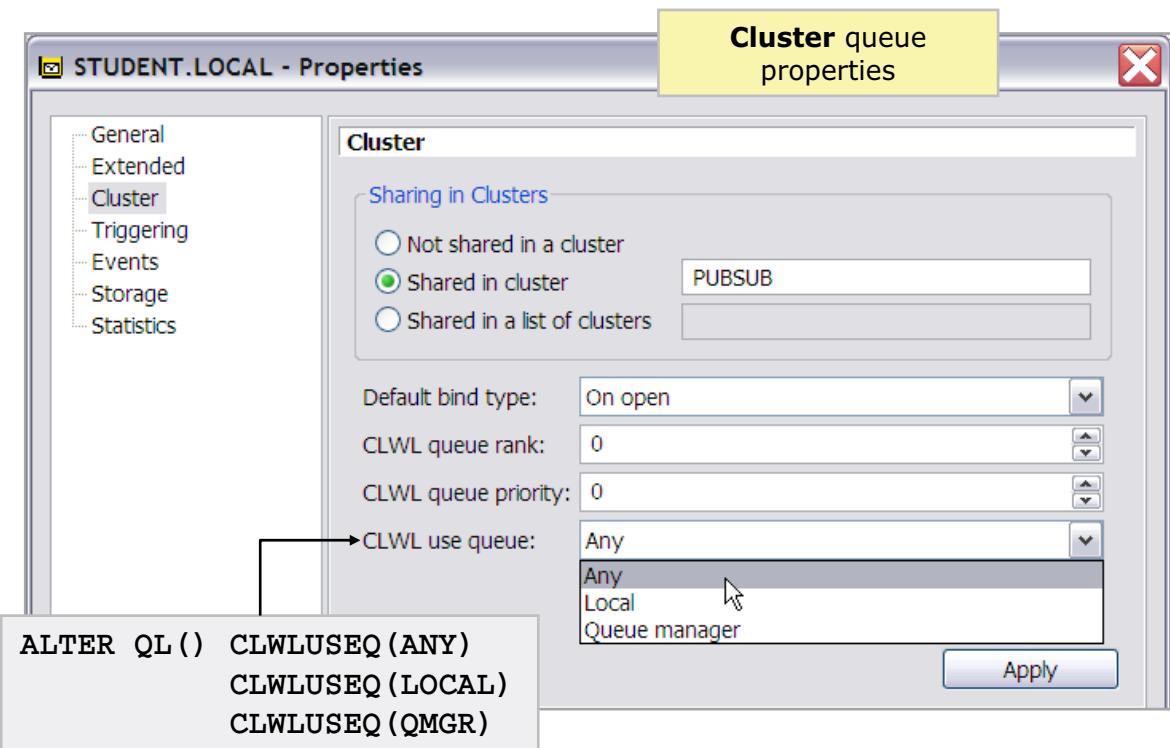
Figure 4-24. Configuring CLWLUSEQ on a queue manager

The figure shows how to configure the cluster workload use-queue attribute for a queue manager in the queue manager **Cluster** properties.

- Setting the **CLWL use queue** property to **Any** is equivalent to specifying CLWLUSEQ(ANY) in an MQSC command.
- Setting the **CLWL use queue** property to **Local** is equivalent to specifying CLWLUSEQ(LOCAL) in an MQSC command.



## Configuring CLWLUSEQ on a queue



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-25. Configuring CLWLUSEQ on a queue

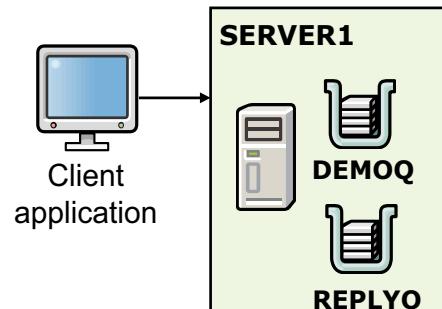
The figure shows how to configure the cluster workload use-queue attribute for a queue in the queue **Cluster** properties.

- Setting the **CLWL use queue** property to **Any** is equivalent to specifying CLWLUSEQ(ANY) in an MQSC command.
- Setting the **CLWL use queue** property to **Local** is equivalent to specifying CLWLUSEQ(LOCAL) in an MQSC command.
- Setting the **CLWL use queue** property to **Queue manager** is equivalent to specifying CLWLUSEQ(QMGR) in an MQSC command.

## Use-queue example

- Start with the use-queue defaults on SERVER1
  - Queue manager CLWLUSEQ (LOCAL)
  - Queue CLWLUSEQ (QMGR)
- Consider the following definition changes:

```
ALTER QMGR CLWLUSEQ (ANY)
ALTER QL (DEMOQ) CLWLUSEQ (LOCAL)
ALTER QL (DEMOQ) CLWLUSEQ (ANY)
```



*With default values, all messages from the client application are delivered to SERVER1*

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-26. Use-queue example

The examples for each workload feature that is described in the rest of this unit starts with all cluster workload attributes reset to the initial values, with one exception.

By default, the queue CLWLUSEQ value is set to QMGR and the queue manager CLWLUSEQ value is set to LOCAL so that all request messages are delivered to the local DEMOQ on SERVER1.

- Changing the queue manager value to ANY means that request messages are delivered to any of the four queue managers because the queue value is set to QMGR.
- Changing the queue value to LOCAL means that the queue manager value is ignored, so all request messages are delivered to SERVER1.
- Changing the queue value to ANY means that the queue manager value is ignored, and all request messages are delivered to any of the four queue managers.

## Rank basics

- Channels and queues with the highest rank are chosen preferentially over channels and queues with lower ranks
  - Range is 0 – 9
  - Default is 0
- Channel rank is checked before queue rank

Examples:

```
DEFINE CHL(CLUS1.SERVER1) CHLTYPE(CLUSRCVR) CLUSTER(CLUS1) ...
CLWLRank( )

DEFINE QL(DEMOQ) CLUSTER(CLUS1) CLWLRank( )
```

*Figure 4-27. Rank basics*

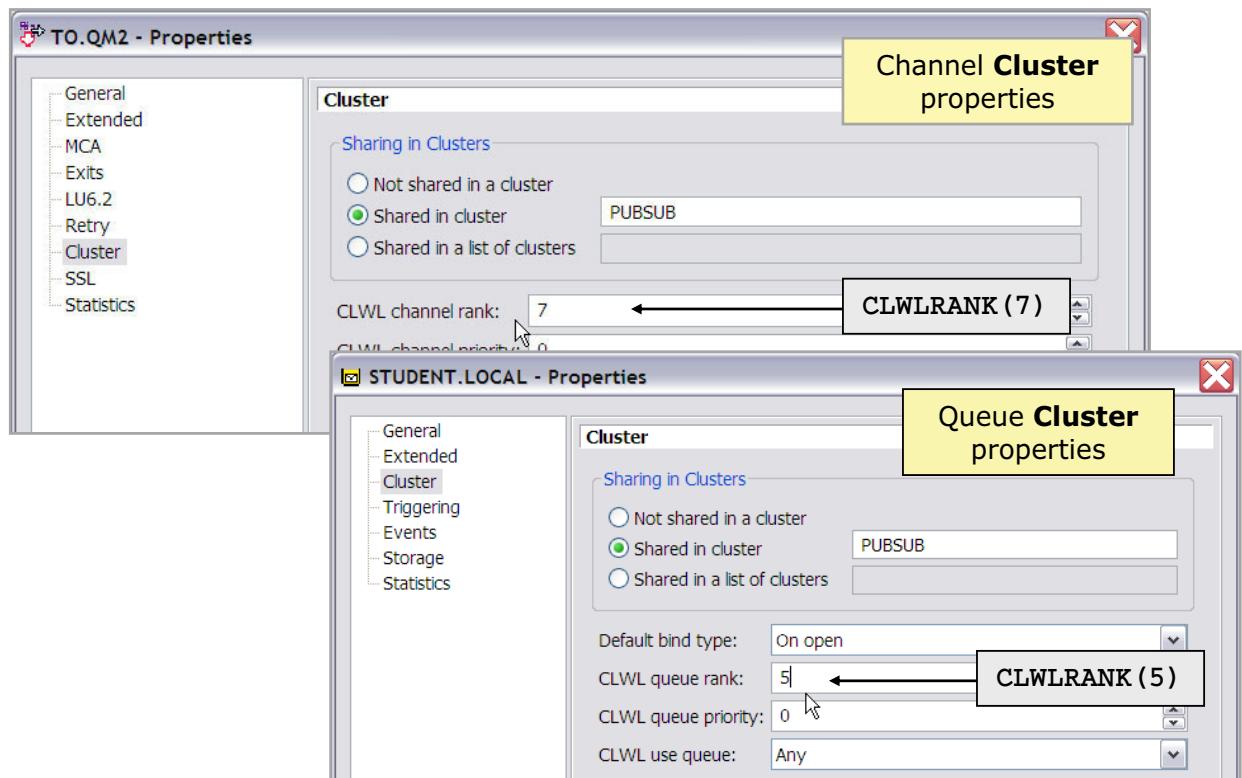
This cluster workload rank attribute (CLWLRank) specifies the rank of the queue for the purposes of cluster workload distribution. Rank can be specified on channels and queues.

The cluster workload rank parameter is valid only for local, remote, and alias queues. The value must be in the range of 0-9, where 0 is the lowest rank and 9 is the highest.

Use the channel cluster workload rank attribute on cluster-receiver channel definitions only.

IBM Training IBM

## Configuring CLWLRANK



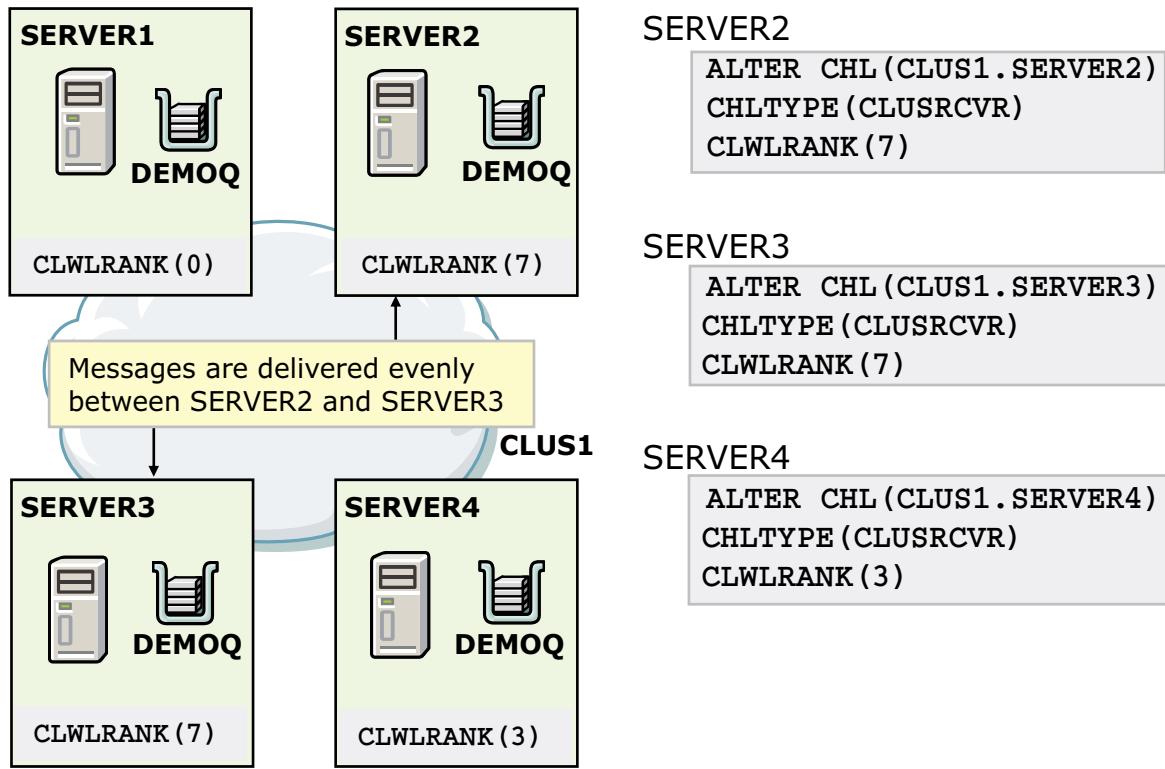
Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-28. Configuring CLWLRANK

The figure shows how to configure the cluster workload rank attribute (CLWLRANK) for a channel in the IBM MQ Explorer channel **Cluster** properties.

## Channel rank example



Implementing workload management in an IBM MQ cluster

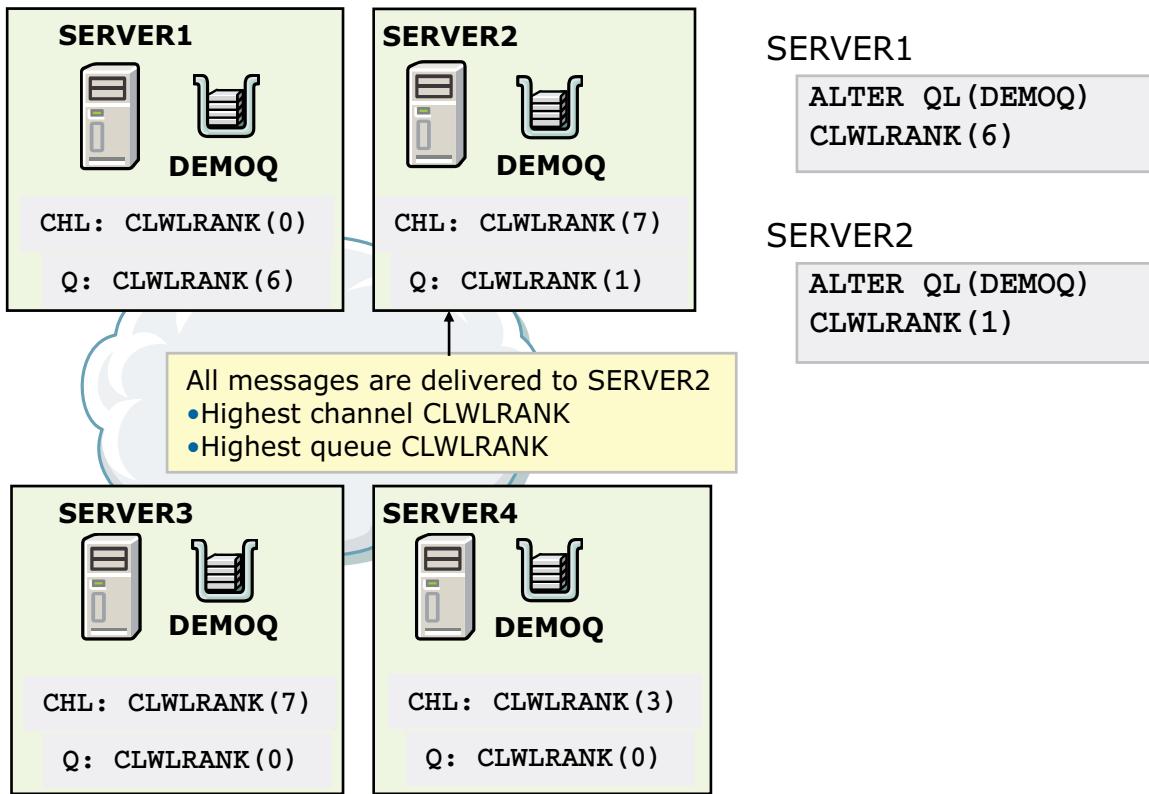
© Copyright IBM Corporation 2017

Figure 4-29. Channel rank example

In this example, the cluster workload channel ranks for SERVER2 and SERVER3 are set higher than SERVER4. SERVER1 has the lowest rank because the default channel rank is zero. After the ranks for channels on SERVER2, SERVER3, and SERVER4 are altered, the messages are distributed equally between the most highly ranked destinations (SERVER2 and SERVER3).

This example assumes that the SERVER1 queue manager cluster workload use-queue (CLWLUSEQ) value is set to "Any". Messages are delivered to all four queue managers, despite the existence of a local instance of DEMOQ on SERVER1.

## Channel and queue rank example



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-30. Channel and queue rank example

After the ranks for queues on SERVER1 and SERVER2 are changed in this example, all the messages are delivered to SERVER2 because the cluster workload management algorithm checks channel ranks before it checks queue rank.

The channel rank check leaves SERVER2 and SERVER3 as valid destinations. Because the queue rank for DEMOQ on SERVER2 is higher than the rank on SERVER3, the messages are delivered to SERVER2. Channel rank is more powerful than queue rank, so the most highly ranked queue (on SERVER1) is not chosen.



### Note

In the figure, the cluster workload queue rank is shown below the cluster workload channel rank. For example, on SERVER1 CLWLRANK for the channel is 0, and for the queue it is 6.

The destinations with the highest rank are chosen regardless of the channel status to that destination. Ranking can lead to messages that build up on the SYSTEM.CLUSTER.TRANSMIT.QUEUE.

## Channel and queue priority basics

- Channels and queues with the highest priorities are chosen preferentially over channels and queues with lower priorities
  - Range is 0 – 9
  - Default is 0
- Channel priority is checked before queue priority
- Rank is checked before channel status, and priority is checked afterward

Examples:

```
DEFINE CHL(CLUS1.SERVER1) CHLTYPE(CLUSRCVR) CLUSTER(CLUS1) ...
      CLWLPRTY( )  
  
DEFINE QL(DEMOQ) CLUSTER(CLUS1) CLWLPRTY( )
```

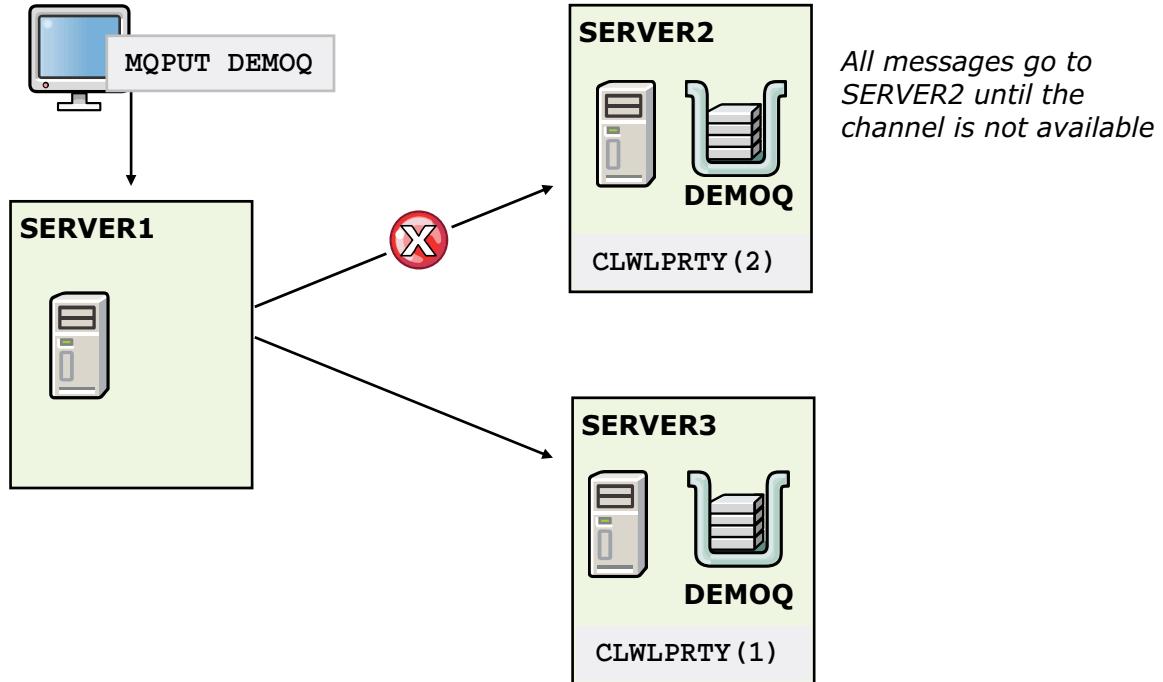
Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-31. Channel and queue priority basics

The cluster workload priority attribute (CLWLPRTY) can be specified on queues and channels. The channels and queues with the highest priorities are chosen over channels and queues with lower priorities.

## Using CLWLPRTY to identify primary and backup servers



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

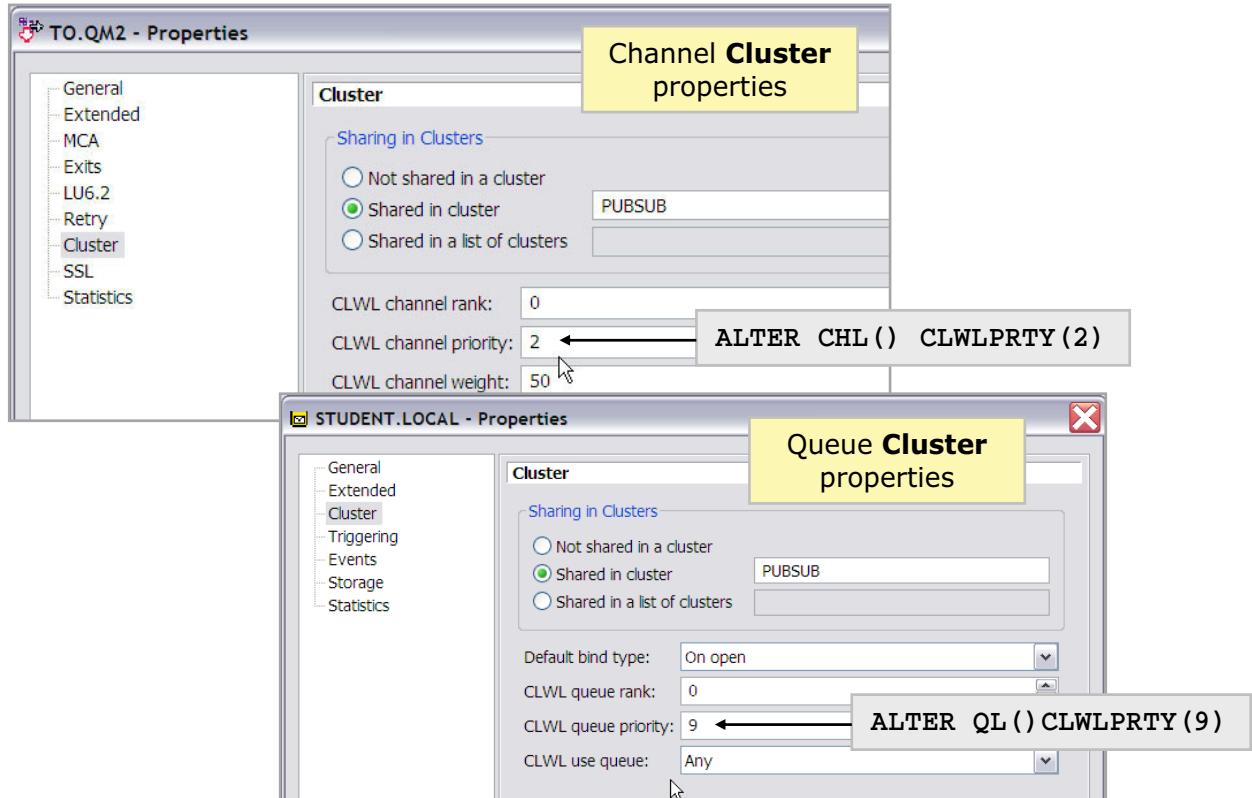
Figure 4-32. Using CLWLPRTY to identify primary and backup servers

Cluster workload priority can be used to identify primary and backup servers. Messages are delivered to primary (high priority) servers until they become unavailable, at which point messages are then workload-balanced to the backup (low priority) servers.

While both channels from SERVER1 are running, all messages put to DEMOQ by the application connected to SERVER1 are delivered to SERVER2 because SERVER2 has a higher channel priority.

When the channel from SERVER1 to SERVER2 stops (as shown with the red "X"), messages that are then put to DEMOQ by the application that is connected to SERVER1 are delivered to SERVER3.

## Configuring channel and queue priority



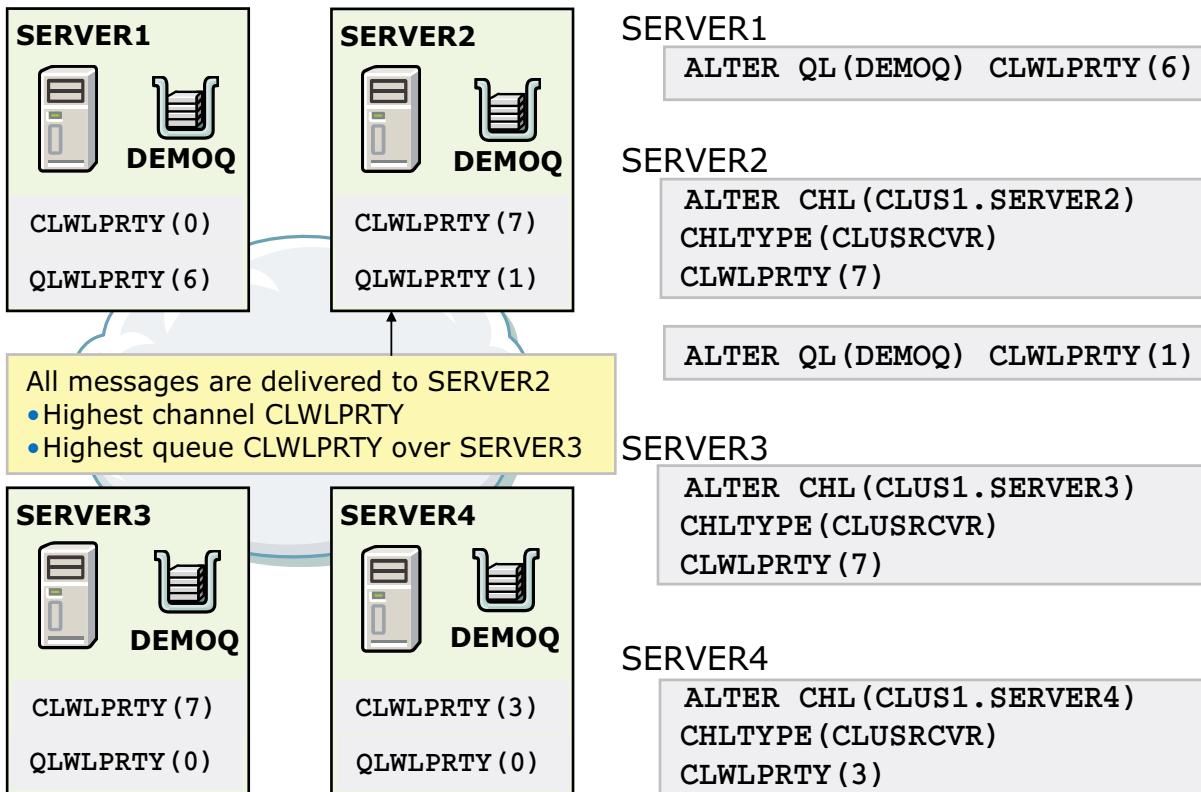
Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-33. Configuring channel and queue priority

The figure shows how to configure the cluster workload priority attribute (CLWLPRTY) for a channel and for a queue.

## Priority example (1 of 2)



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-34. Priority example (1 of 2)

The channel priority check leaves SERVER2 and SERVER3 as valid destinations. Because the queue priority for DEMOQ on SERVER2 is higher than the queue priority on SERVER3, the messages are delivered to SERVER2. This scenario assumes that the channel status to all destinations is equally preferential (in this case, either running or inactive).

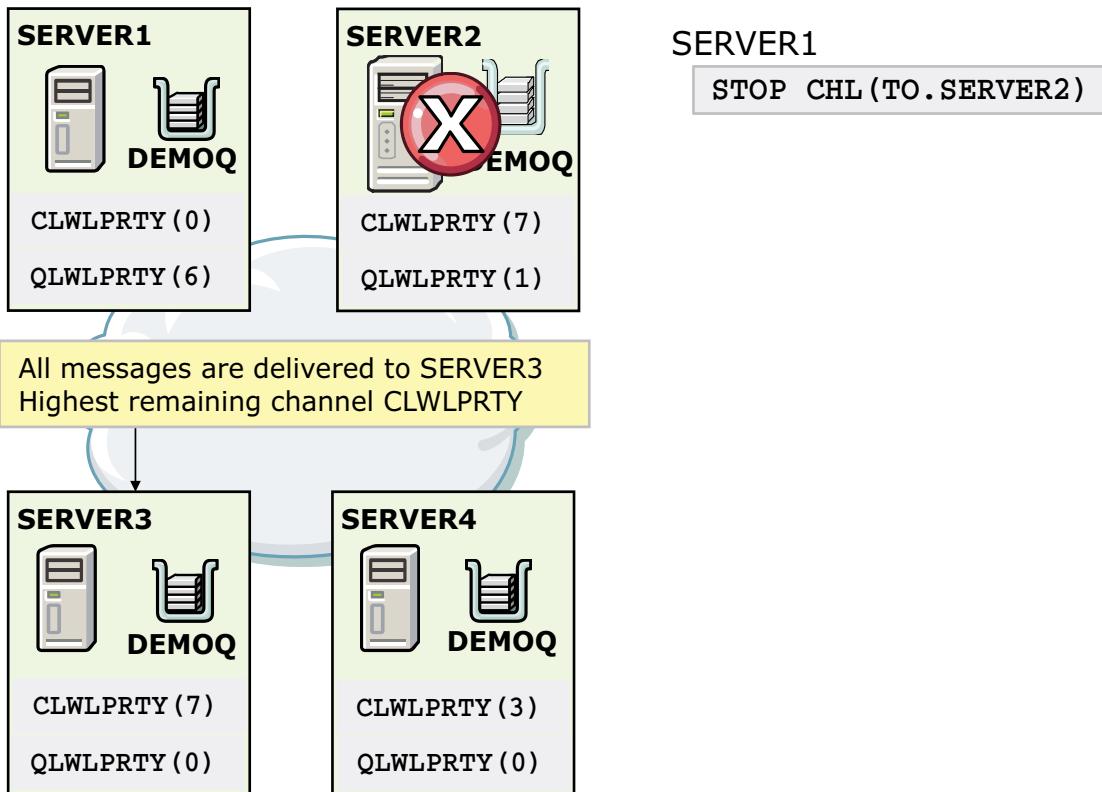


### Note

Because channel priority is more powerful than queue priority, the queue with the highest priority (on SERVER1) is not chosen.



## Priority example (2 of 2)



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-35. Priority example (2 of 2)

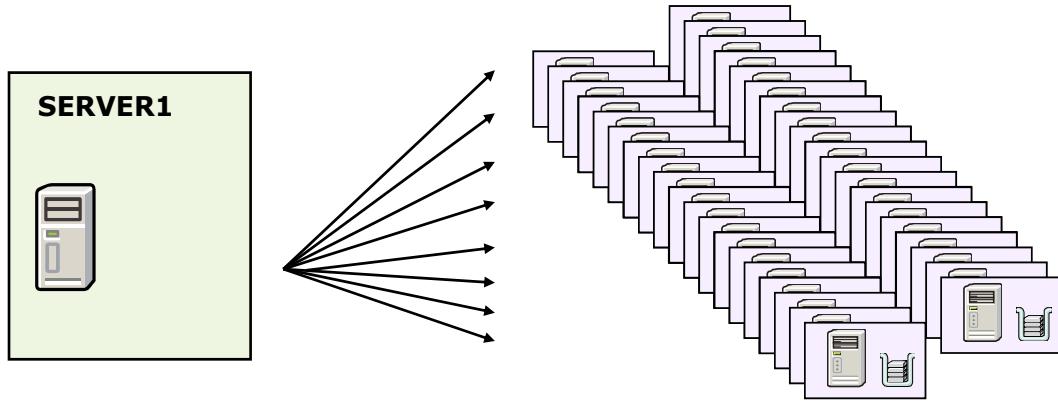
Workload priority is checked after channel status in the cluster workload management algorithm. Stopping the channel from SERVER1 to SERVER2 leaves the following channel status for each destination:

- SERVER1: Inactive
- SERVER2: Stopped
- SERVER3: Running
- SERVER4: Running

The cluster workload management algorithm removes SERVER2 from the list of valid destinations, so the priority check is completed only on SERVER1, SERVER3, and SERVER4. Out of the remaining three queue managers, the one with the highest priority is SERVER3, so all messages are delivered to SERVER3.

## Most recently used channels

- **Problem:** How can you limit the number of active outbound channels in large clusters?
  - Hundreds of queue managers in the cluster means hundreds of outbound channels exist
  - Active channel limits can limit cluster size



- **Solution:** Use the cluster workload most recently used channels feature

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-36. Most recently used channels

The most recently used channels attribute can be used to limit the number of active outbound channels in large clusters. The cluster workload attribute is CLWLMRUC.

## Most recently used channels basics

- If more than one destination is valid, limit the round robin to the “n” most recently used channel destinations
- MQSC: **ALTER QMGR CLWLMRUC( )**
  - Range: 1 – 999999999
  - Default: 999999999
- Example:

From the default CLWLMRUC(999999999), reduce the MRU value to restrict the number of destinations in the round robin

```
ALTER QMGR CLWLMRUC(2)
```

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-37. Most recently used channels basics

When the most recently used (MRU) value is lower than the number of valid destinations available to the round robin, all but the MRU channels are removed from the round robin.

If the CLWLMRUC attribute is set to 2 on a queue manager, messages are delivered to only two of the four destinations.

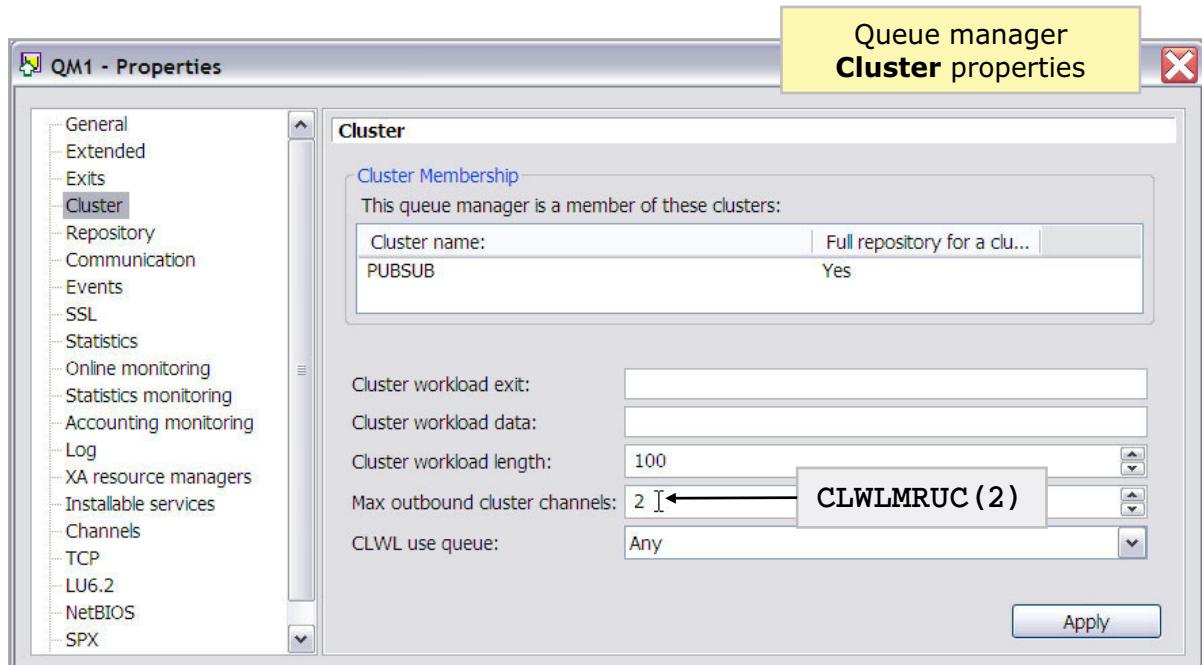


### Note

The MRU channels cannot be reliably identified externally, although this information is available to cluster workload exits.

When IBM MQ uses MRU, the number of active channels a server can run do not limit the cluster size.

## Configuring CLWLMRUC on a queue manager



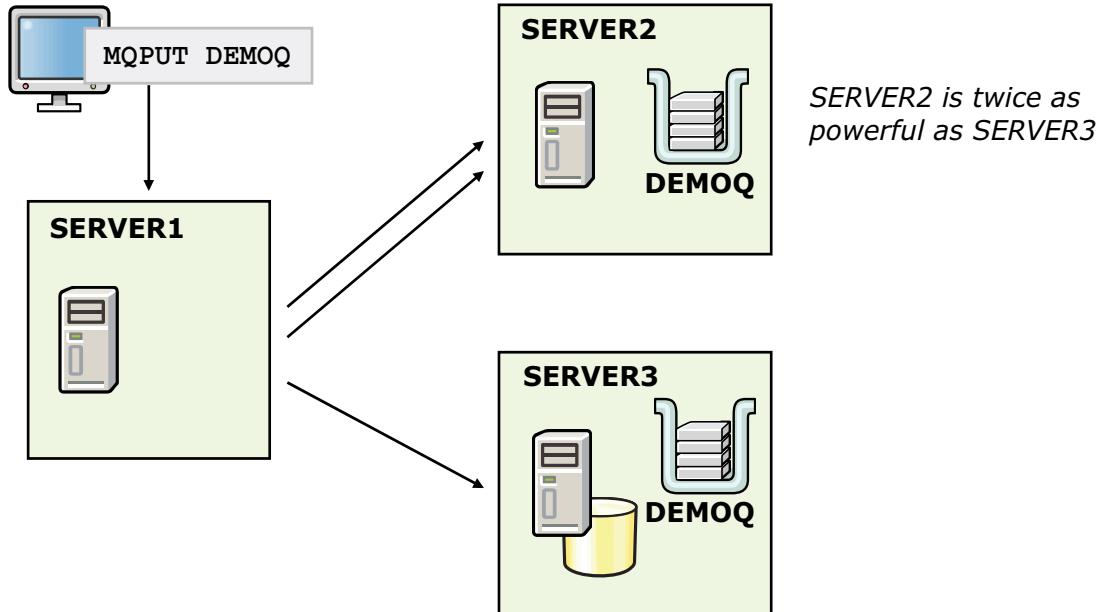
Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-38. Configuring CLWLMRUC on a queue manager

The figure shows how to configure the CLWLMRUC attribute for a queue manager.

## Using processing power to control the workload



How can you send SERVER2 twice as many messages?  
Use **CLWLWGHT** attribute

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-39. Using processing power to control the workload

This figure shows that one of the servers in the cluster has greater processing power, and the user wants to send a greater workload to the more powerful server.

You can use the cluster workload weight attribute to send more workload to a specific server and queue manager in the cluster.

## Channel weight basics

- If more than one destination is valid, the round robin algorithm sends messages in numbers proportional to their channel weights
- Attribute **CLWLWGHT**
  - Range: 1 – 99
  - Default: 50

Example:

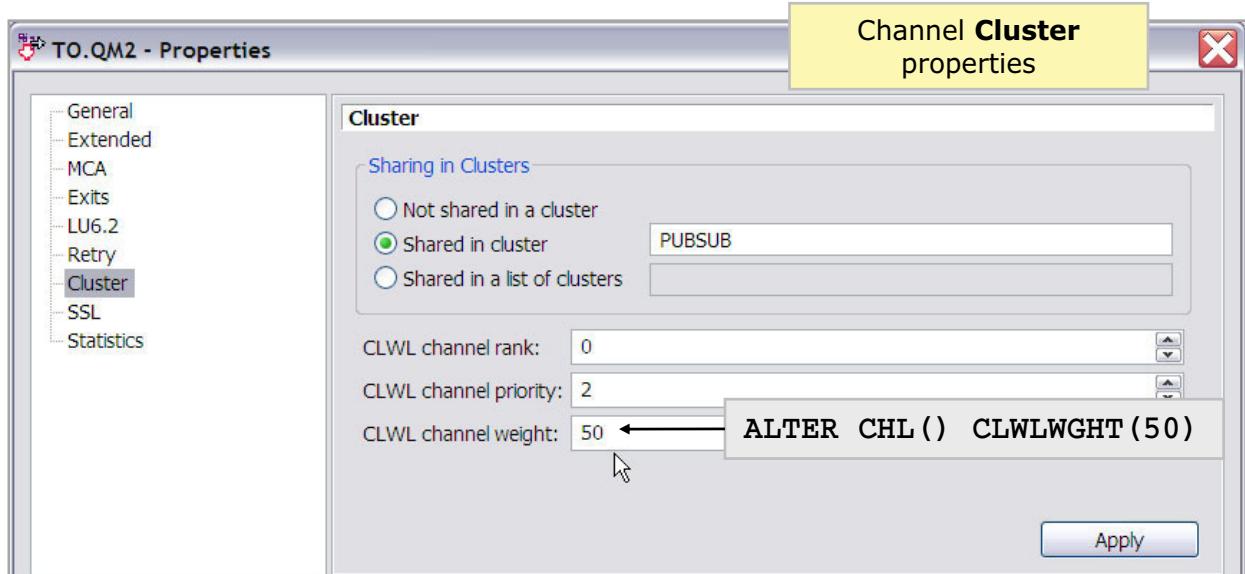
```
DEFINE CHL(CLUS1.SERVER1) CHLTYPE(CLUSRCVR) CLUSTER(CLUS1) ...
      CLWLWGHT( )
```

Figure 4-40. Channel weight basics

The cluster workload channel weight (CLWLWGHT) attribute causes the round robin algorithm to send messages in numbers proportional to their channel weights.

IBM Training 

## Configuring CLWLWGHT



Implementing workload management in an IBM MQ cluster

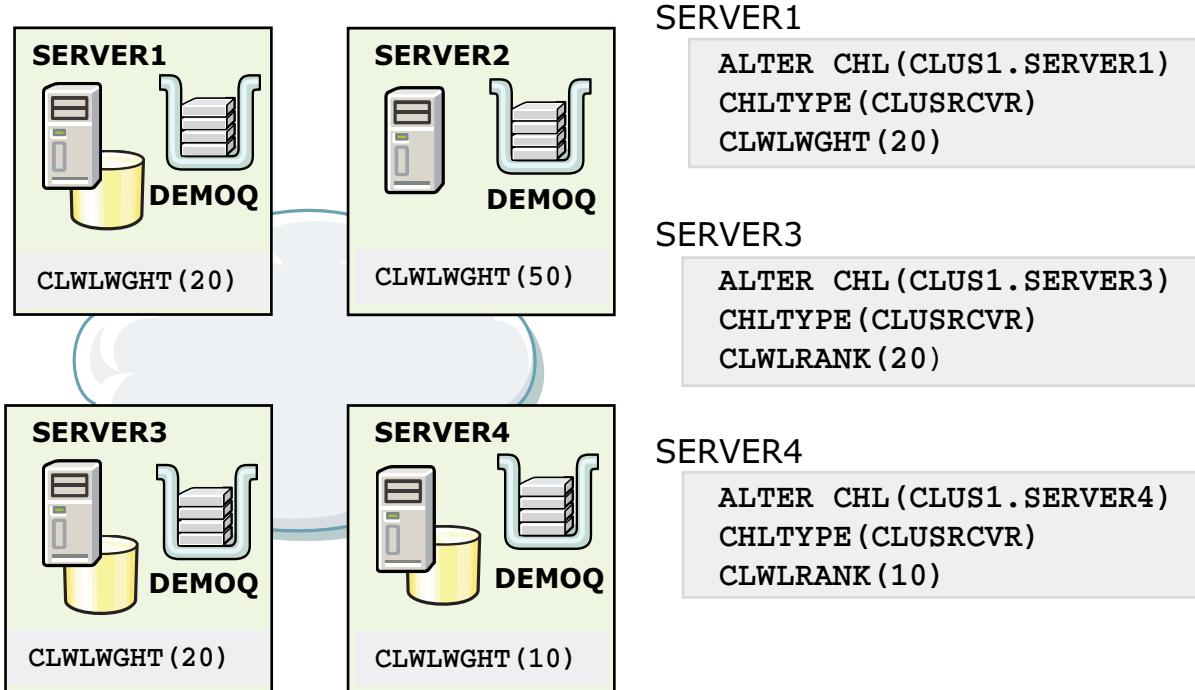
© Copyright IBM Corporation 2017

Figure 4-41. Configuring CLWLWGHT

The figure shows how to configure the CLWLWGHT attribute for a channel.



## Channel weight example



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-42. Channel weight example

By using cluster workload weight, the cluster workload management algorithm can favor more powerful servers.

In this example, the approximate percentages of messages that are distributed to each queue manager are as follows:

- SERVER1: 20%
- SERVER2: 50%
- SERVER3: 20%
- SERVER4: 10%

## Cluster workload management algorithm: Summary

1. Queue manager builds a list of possible destinations
2. If queue is specified, eliminate the following queues:
  - Queues with PUT(DISABLED)
  - Remote instances of queues that do not share a cluster with the local queue manager
  - Remote CLUSRCVR channels that are not in the same cluster as the queue
3. If queue manager name is specified, eliminate the following queues:
  - Remote instances of queues that do not share a cluster with the local queue manager
  - Remote CLUSRCVR channels that are not in the same cluster as the queue or topic
4. If available, use the local queue instance unless overridden by use-queue
5. Evaluate channel rank
6. Evaluate channel status
7. Evaluate cluster-receiver channel priority
8. Evaluate channel priority
9. Evaluate queue priority
10. Evaluate most recently used channels
11. Evaluate least recently used with channel weight

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-43. Cluster workload management algorithm: Summary

This figure summarizes the workload management algorithm.

## Multiple cluster transmission queues

- Uses:
  - Separate message traffic
  - Manage messages
  - Monitoring
- Configured on the *sending queue manager*, not the owners of the cluster-receiver channel definitions
- Configuration option 1: Change the queue manager to automatically assign a different cluster transmission queue to each cluster-sender channel
- Configuration option 2: Manually configure cluster transmission queues

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

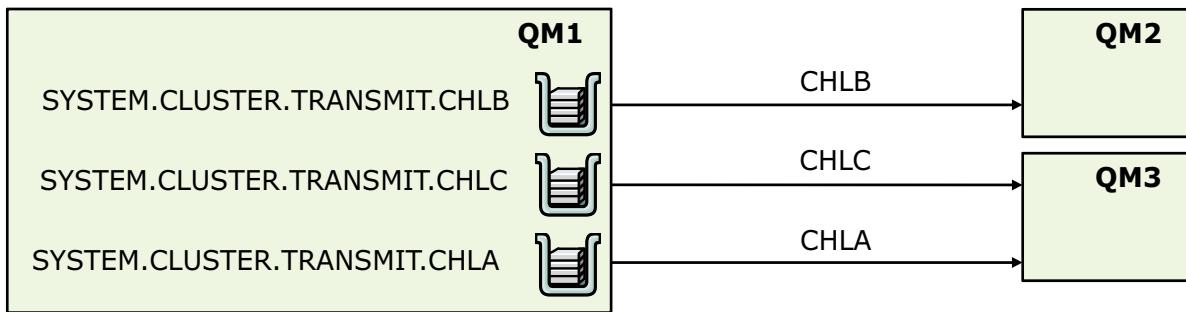
Figure 4-44. Multiple cluster transmission queues

You can change the queue manager DEFCLXQ attribute to assign a different cluster transmission queue to each cluster-sender channel. Messages that the cluster-sender channel forwards are placed on separate cluster transmission queues.

You can also configure cluster transmission queues manually by setting the queue attribute CLCHNAME. You can decide which cluster-sender channels share which transmission queues, which have separate transmission queues, and, which use the cluster transmission queue, or queues.

## Automatic cluster transmission queues

- Modify queue manager switch to automatically create a dynamic transmission queue for each cluster-sender channel:  
`ALTER QMGR DEFCLXQ (CHANNEL)`
- Creates dynamic queues that are based on model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL`
- Uses well-known queue names:  
`SYSTEM.CLUSTER.TRANSMIT.<CHL-NAME>`



[Implementing workload management in an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 4-45. Automatic cluster transmission queues

The default for the queue manager is use a single cluster transmission queue (DEFCLXQ(SCTQ)). You can modify the queue manager to automatically create a separate transmission queue for each cluster-sender channel.

## Manual cluster transmission queues

- Administrator defines a transmission queue and configures which cluster-sender channels use the transmission queue

Example

```
DEFINE QLOCAL(CLUS1.XMITQ) CLCHNAME(CLUS1.* ) USAGE(XMITQ)
```

- CLCHNAME can be a pattern and include wildcard characters to select multiple channels

- Any cluster sender channel that is not covered by a manual transmission queue, defaults to the transmission queue behavior that is specified in the queue manager DEFCLXQ attribute

Figure 4-46. Manual cluster transmission queues

You can manually define multiple transmission queues by manually modifying the local queue definition.

## Unit summary

- Describe how queue manager clusters assist with workload management
- Describe the attributes that affect the workload balancing algorithm
- Use multiple transmission queues to separate the workload

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

*Figure 4-47. Unit summary*

## Review questions

1. True or False: The SYSTEM.CLUSTER.COMMAND.QUEUE holds inbound and outbound administrative messages.
2. True or False: Remote queue definitions are not required when using clusters.
3. True or False: The most recently used attribute can be specified on both a queue manager and a channel definition.
4. True or False: A cluster workload algorithm uses workload balancing attributes and rules to select the final destination for messages that are put onto cluster queues.



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-48. Review questions

Write your answers here:

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

## Review answers

1. True or False: The SYSTEM.CLUSTER.COMMAND.QUEUE holds inbound and outbound administrative messages.  
**The answer is False. The SYSTEM.CLUSTER.COMMAND.QUEUE holds inbound administrative messages only.**
2. True or False: Remote queue definitions are not required when using clusters.  
**The answer is True.**
3. True or False: The most recently used attribute can be specified on both a queue manager and a channel definition.  
**The answer is False. The most recently used attribute is only specified on a queue manager definition.**
4. True or False: A cluster workload algorithm uses workload balancing attributes and rules to select the final destination for messages that are put onto cluster queues.  
**The answer is True.**



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 4-49. Review answers

## Exercise: Implementing workload management in a cluster

Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

*Figure 4-50. Exercise: Implementing workload management in a cluster*

In this exercise, you create a cluster of four queue managers. You then use the cluster mechanism to send messages between queues on all queue managers in the cluster.

## Exercise objectives

- Create a queue manager cluster
- Use channel and queue attributes in various combinations to alter the workload distribution in a cluster



Implementing workload management in an IBM MQ cluster

© Copyright IBM Corporation 2017

*Figure 4-51. Exercise objectives*

For detailed instructions, see the Course Exercises Guide.

---

# Unit 5. More troubleshooting tools and techniques

## Estimated time

02:00

## Overview

In this unit, you learn more about the IBM MQ tools and techniques for identifying and handling problems. Topics in this unit include event monitoring, message monitoring, and dead-letter queue message handling. The unit also provides some tips and techniques for disaster recovery.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Use IBM MQ events to detect problems in your queue manager network
- Uses trace-route messages to determine the last known location of a message and determine configuration problems with a queue manager
- Use the IBM MQ unload and load utility to copy messages to or from a file
- Implement a dead-letter queue handler to automate the handling of messages on the dead-letter queue
- Recover IBM MQ if a failure occurs

More troubleshooting tools and techniques

© Copyright IBM Corporation 2017

Figure 5-1. Unit objectives

## 5.1. Event and activity monitoring

## Instrumentation event monitoring

- An *instrumentation event* is a logical combination of events that a queue manager or channel instance detects and causes the queue manager or channel instance to put a special *event message* on an event queue.
- Use to:
  - Detect problems in your queue manager network
  - Generate an audit trail
  - React to queue manager state changes
- Event queues are either local queues, alias queues, or local definitions of remote queues
  - Centralize monitoring activities by using local definitions of remote queue
- Requires an MQI application program that:
  - Gets the message from the queue
  - Processes the message to extract the event data

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-2. *Instrumentation event monitoring*

Event monitoring is the process of detecting occurrences of instrumentation events in a queue manager network. IBM MQ instrumentation events provide information about errors, warnings, and other significant occurrences in a queue manager. Such an event causes the queue manager or channel instance to put a special message, called an event message, on an event queue.

Applications that use events to monitor queue managers assume that you configured channels between the queue managers in your network. It is also assumed that you implemented the required data conversions. The normal rules of data conversion apply. For example, if you are monitoring events on a queue manager on UNIX from a queue manager on z/OS, ensure that you convert EBCDIC to ASCII.

## Examples of conditions that cause events

- Threshold limit for the number of messages on a queue is reached
- Channel instance is started or stopped
- Queue manager becomes active, or is requested to stop
- Application tries to open a queue that specifies a user ID that is not authorized
- Objects are created, deleted, changed, or refreshed
- MQSC command runs successfully
- Queue manager starts writing to a new log extent
- Message is put on the dead-letter queue

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-3. Examples of conditions that cause events*

This figure lists examples of some conditions that cause events in IBM MQ.

## Event queues

SYSTEM.ADMIN.QMGR.EVENT	Queue manager events
SYSTEM.ADMIN.CHANNEL.EVENT	Channel events
SYSTEM.ADMIN.PERFM.EVENT	Performance events
SYSTEM.ADMIN.CONFIG.EVENT	Configuration events
SYSTEM.ADMIN.COMMAND.EVENT	Command events
SYSTEM.ADMIN.LOGGER.EVENT	Logger events

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-4. Event queues*

By incorporating instrumentation events into your own system management application, you can monitor the activities across many queue managers, across many different nodes, and for multiple IBM MQ applications. When an event occurs, the queue manager puts an event message on the appropriate SYSTEM.ADMIN event queue, if defined.

Each type of event is written to a specific queue. For example, queue manager events are written, by default to the SYSTEM.ADMIN.QMGR.EVENT queue.

## Queue manager events

- Enable and disable events with MQSC or IBM MQ Explorer by specifying the appropriate values for queue manager, queue, or both, depending on the type of event
  - Authority exceptions
  - MQPUT or MQGET inhibits
  - Local queue errors, such as unknown queue or alias queue error
  - Remote queue or queue manager problems
  - Queue manager start or queue manager stop events
  - Performance events
  - Command events
  - Channel events
  - Channel auto-definition events
  - SSL events
  - Logger events
  - Configuration events

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-5. Queue manager events

Queue manager events are related to the use of resources within queue managers. For example, a queue manager event is generated if an application tries to put a message on a queue that does not exist. The figure lists the types of queue manager events.

- Authority events report an authorization, such as an application that tries to open a queue for which it does not have the required authority.
- Inhibit events indicate that an MQPUT or MQGET operation was attempted against a queue where the queue is inhibited for puts or gets, or against a topic where the topic is inhibited.
- Local events indicate that an application or the queue manager cannot access a local queue or other local object.
- Remote events indicate that an application or the queue manager cannot access a remote queue on another queue manager.
- Start and stop events indicate that a queue manager started or was requested to stop or quiesce.

## Controlling queue manager events

Queue manager events set on queue manager **Properties > Events** page

Events		
Authority events:	Disabled	AUTHOREV
Inhibit events:	Disabled	INHIBTEV
Local events:	Disabled	LOCALEV
Remote events:	Disabled	REMOTEEV
Start and stop events:	Enabled	STRSTPEV
Performance events:	Disabled	PERFMEV
Command events:	Disabled	CMDEV
Channel events:	Disabled	CHLEV
Channel auto definition events:	Disabled	CHADEV
SSL events:	Disabled	SSLEV
Logger events:	Logger Events are only valid in the Queue Manager Properties page	LOGGEREV
Configuration events:	Disabled	CONFIGEV

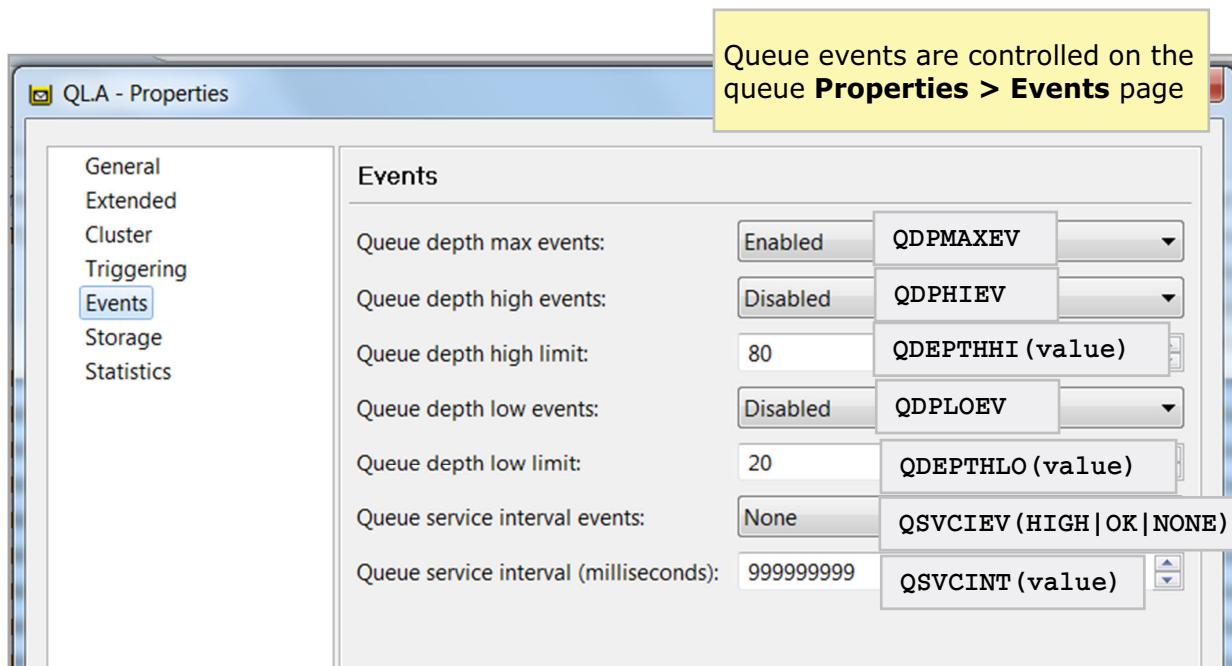
[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-6. Controlling queue manager events

You can control queue manager event generation in the IBM MQ Explorer queue manager **Events** properties or by using the comparable event attribute on the ALTER QMGR command.

## Controlling queue events



[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-7. Controlling queue events

You can enable queue events such as queue depth high events and queue depth low events, in the queue **Events** properties in IBM MQ Explorer or by using the comparable attribute in MQSC.

A **Queue depth high** event occurs when a message that is put on the queue causes the queue depth to be greater than or equal to the **Queue depth high limit** (QDEPTHHI) value. A **Queue depth high** event automatically enables both a **Queue depth low** event and a **Queue full** event on the same queue.

A **Queue depth low** event occurs when a message that is taken off the queue causes the queue depth to be less than or equal to the value determined by the **Queue depth low limit** (QDEPTHLO). A **Queue depth low** event automatically enables both a **Queue depth high** event and a **Queue full** event on the same queue.

Queue service interval events indicate whether an operation was done on a queue within a user-defined time interval that is called the service interval. Depending on your installation, you can use queue service interval events to monitor whether messages are being taken off queues quickly enough.

For a detailed description of all events, see the “Enabling queue depth events” topic in the IBM Knowledge Center.

## Configuration events

- Configuration events notify you about changes to the attributes of an object:
  - Create object events
  - Change object events
  - Delete object events
  - Refresh object events
- Event data contains:
  - Origin information: Queue manager, user ID, and the application
  - Context information: Replica of the context information in the message data
  - Object identity: Name, type, and disposition of the object
  - Object attributes: Values of all the attributes in the object
- “Change object” event generates two messages:
  - One message has the information before the change
  - One message has the information after the change

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-8. Configuration events*

Configuration events are generated when a configuration event is requested explicitly, or automatically when an object is created, modified, or deleted.

A configuration event message contains information about the attributes of an object. For example, a configuration event message is generated if a namelist object is created, and contains information about the attributes of the namelist object.

The event messages for configuration events are put on the SYSTEM.ADMIN.CONFIG.EVENT queue.

## Configuration event examples

Configuration event message that is generated when CONFIGEV(ENABLED)	Configuration event message is not generated
DEFINE/ALTER/DELETE AUTHINFO DEFINE/ALTER/DELETE CFSTRUCT DEFINE/ALTER/DELETE CHANNEL DEFINE/ALTER/DELETE NAMELIST DEFINE/ALTER/DELETE PROCESS DEFINE/ALTER/DELETE QLOCAL DEFINE/ALTER/DELETE QMODEL/QALIAS/QREMOTE DEFINE/ALTER/DELETE STGCLASS DEFINE/ALTER/DELETE TOPIC DEFINE MAXSMSGS SET CHLAUTH ALTER QMGR MQSET call, other than for a temporary dynamic queue	<ul style="list-style-type: none"> <li>• When a command or an MQSET call fails</li> <li>• When a queue manager encounters an error when it tries to put a configuration event on the event queue</li> <li>• For a temporary dynamic queue</li> <li>• When internal changes are made to the TRIGGER queue attribute</li> <li>• For the configuration event queue SYSTEM.ADMIN.CONFIG.EVENT, except by the <b>REFRESH QMGR</b> command</li> <li>• For <b>REFRESH/RESET CLUSTER</b> and <b>RESUME/SUSPEND QMGR</b> commands that cause clustering changes</li> <li>• When creating or deleting a queue manager</li> </ul>

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-9. Configuration event examples*

This figure lists the commands that generate configuration events when configuration events are enabled. For example, entering a DEFINE AUTHINFO command generates an event message.

This figure also lists some events that do not generate event messages. For example, entering a command to create a queue manager does not generate an event message.

## Logger events

- Notifications that a queue manager started writing to a new log extent
  - Name of the current log extent
  - Name of the earliest log extent that is needed for restart recovery
  - Name of the earliest log extent that is needed for media recovery
  - Directory in which the log extents are located
- Every logger event message is put on the logger event queue SYSTEM.ADMIN.LOGGER.EVENT
- For linear logs only
- Sample program to monitor log events:  
`<mqdir>\tools\c\samples\amqslog0.c`

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-10. Logger events*

Logger events are reported when a queue manager that uses linear logging starts writing log records to a new log extent.

A logger event message contains information that specifies the log extents that the queue manager requires to restart the queue manager, or for media recovery.

The event messages for logger events are put on the SYSTEM.ADMIN.LOGGER.EVENT queue.

IBM MQ provides a sample program to monitor log events.

## Logger event examples

Logger event message is generated	Logger event message is not generated
<ul style="list-style-type: none"> <li>• When the queue manager starts writing to a new log extent</li> <li>• When the queue manager starts</li> <li>• When the LOGGEREV queue manager attribute is changed from DISABLED to ENABLED</li> </ul>	<ul style="list-style-type: none"> <li>• When a queue manager is configured to use circular logging (LOGGEREV = DISABLED and cannot be altered)</li> <li>• When a queue manager encounters an error when it tries to put a logger event on the event queue (the action that caused the event completes, but no event message is generated)</li> </ul>

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-11. Logger event examples*

This figure lists some events that generate logger event messages. For example, an event message is generated when the LOGGEREV queue attribute is set to ENABLED and the queue manager starts writing to a new log extent.

The figure also lists some events that do not generate logger event messages. For example, a logger event message is not generated when the queue manager is configured to use circular logging.

## SupportPac MH05: WebSphere MQ Events Display Tool

- Provides a simple command tool (`xmqdspev`) to display IBM MQ events that are generated on the SYSTEM.ADMIN.\*.EVENT queues
- Output can be redirected to a file
- Current revision level is IBM MQ V8.0

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-12. SupportPac MH05: WebSphere MQ Events Display Tool*

SupportPac MH05, IBM WebSphere MQ Events Display Tool, is a command tool that you can use to display IBM MQ events that are generated on SYSTEM.ADMIN.\*.EVENT queues. As of the time of this printing of this course, the revision level for the SupportPac was IBM MQ V8.0.

## Sample output from MH05 SupportPac

```
C:\MQ\>xmqdsspev -m QMC01 -q SYSTEM.ADMIN.QMGR.EVENT
Connected to queue manager 'QMC01' (Platform=Windows 2010)
Processing EVENT queue 'SYSTEM.ADMIN.QMGR.EVENT'. . .

ReasonCode: 2222
EventName: Queue Manager Active - MQRC_Q_MGR_ACTIVE (2222,X'8AE')
Event Type: Start and Stop
Description: This condition is detected when a queue manager becomes
active.
QMngrName: QMC01

ReasonCode: 2085
EventName: Unknown Object Name - MQRC_UNKOWN_OBJECT_NAME (2085,X'825')
EventType: Local
Description: On an MQOPEN or MQPUT1 call, the ObjectQMngrName field in
the object descriptor MQOD is set to one of the following.
. .
QMngrName: QMC01
ApplType: 11 (MQAT_WINDOWS)
ApplName: C:\MQ\amqsput.exe
QName: UKNOWN_QUEUE

2 event message(s) processed.
2 event message(s) displayed.
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-13. Sample output from MH05 SupportPac

This figure shows an example of using the MH05 SupportPac to get event messages from the SYSTEM.ADMIN.QMGR.EVENT queue. The event messages include a reason code, event name, event type, description, and other information about the event.

## 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 in PCF format
  - Activity records are written to well-defined topics
- 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

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-14. Application activity trace*

## Tracing application activity

- Messages written to SYSTEM.ADMIN TRACE.ACTIVITY.QUEUE
- Behavior is specified in activity trace configuration file (`mqat.ini`)
  - Can be changed without queue manager restart
  - Configurable detail level can include partial, full, or both partial and full message payload
  - Frequency options for tuning
- Negatively affects performance
  - Reduce the effect on performance by modifying the **ActivityCount** and **ActivityInterval** parameters

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-15. Tracing application activity*

Application activity trace produces detailed information about the behavior of applications that are connected to a queue manager. It traces the behavior of an application and provides a detailed view of the parameters that are used by an application as it interacts with IBM MQ resources. It also shows the sequence of MQI calls that an application sends. Activity records are written to the system queue SYSTEM.ADMIN TRACE.ACTIVITY.QUEUE. You configure the activity trace option in the `mqat.ini` file.

Messages are normally written when the application disconnects from the queue manager. For long running applications, intermediate messages are written when any of the following conditions occur:

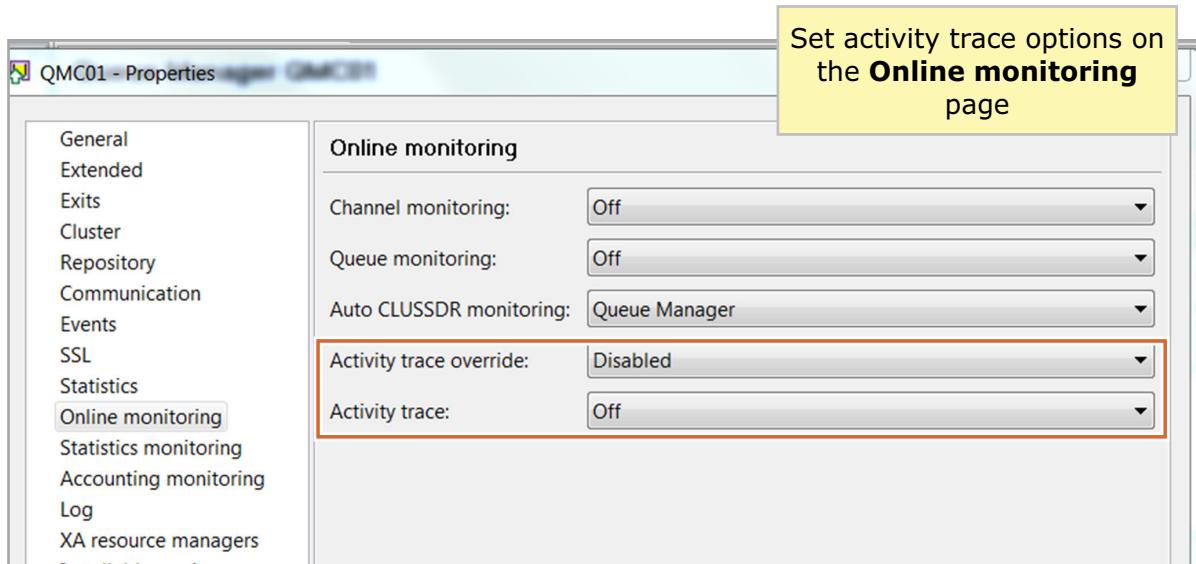
- If the lifetime of the connection exceeds a user-defined timeout (**ActivityInterval** parameter)
- After a user-defined number of operations (**ActivityCount** parameter)
- When the size of the record approaches the largest message that is allowed on the trace queue

When the timeout expires or MQI count is reached, the activity data that is accumulated so far is written as a message and the activity data is reset.

Activity trace negatively affects performance. You can reduce the effect of an activity trace on performance by modifying the **ActivityCount** and **ActivityInterval** parameters.



## Enabling activity trace in IBM MQ Explorer



[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-16. Enabling activity trace in IBM MQ Explorer

You can enable an application activity trace in IBM MQ Explorer:

1. Right-click the queue manager in the **IBM MQ Explorer - Navigator** view and then click **Properties**.
2. On the **Online Monitoring** properties page, set the **Activity trace** option.

## Choosing the application activity trace level

- Configurable options in the `mqat.ini` file
  - Restrict the applications that the trace runs against
  - Include message data in the trace message
  - Time interval between trace messages
  - Batch multiple trace records into a single message.
  - Level of detail that is included in the trace message

Example `mqat.ini`:

```
AllActivityTrace:          # Global settings stanza
ActivityInterval=1         # Time interval between trace messages
                           # Values: 0-99999999 (0=off), default=0
ActivityCount=100          # 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
StopOnGetTraceMsg=ON        # Stop trace on get of activity trace message
                           # Values: ON | OFF # Default: ON
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-17. Choosing the application activity trace level

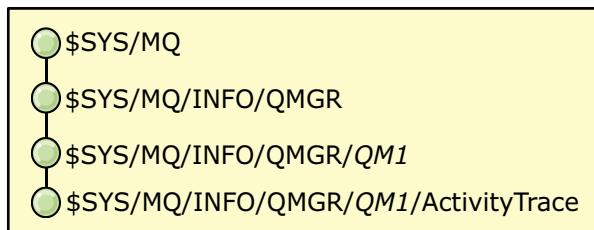
The activity trace behavior is configured by using the `mqat.ini` configuration file. The `mqat.ini` file is in the queue manager data directory. Users that are running traced applications require “read” permissions to the `mqat.ini` file.

The `mqat.ini` file includes the following configurable options:

- Identify the application that you want to trace by specifying the `ApplName` in the `ApplicationTrace` stanza.
- Specify the amount of message data to include in the application activity trace message.
- Configure the time interval between trace messages and batch multiple trace records into a single message. With this capability, you can modify the number of messages that are generated in cases where performance is a priority. The default is to generate a record for every MQI call that is made to the queue manager. Each application activity trace message contains 100 trace records.
- Specify the level of detail that is included in the application activity trace message. Depending on the granularity you need, you can set the trace level to LOW, MEDIUM, or HIGH with the `TraceLevel` option.

## Subscribing to activity trace data

- IBM MQ publishes application activity trace data from the queue manager to \$SYS/MQ topic branch
  - Access to \$SYS/MQ is restricted to administrators by default
  - Others can subscribe to a subset of the data
- Subscription format:  
`$SYS/MQ/INFO/QMGR/QmgrName/ActivityTrace/ResourceType/  
ResourceIdentity`
- Types of traced resource (*ResourceType*)
  - **ApplName**: Any connection from a matching application name
  - **ChannelName**: Either any connection on a matching SVRCONN channel or any PUTs or GETs on a queue manager channel
  - **ConnectionId**: ID from the “application connections” in IBM MQ Explorer, or the CONN value concatenated with the EXTCNN value from `DISPLAY CONN`



[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-18. Subscribing to activity trace data

In IBM MQ V9, you can subscribe to queue manager ActivityTrace topics to get activity trace information for applications, channels, and connection IDs.

## Example activity trace topic strings

- Topic string for an application that is named **amqspput**:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/App1Name/amqspputc.exe
```

- Topic string for a channel:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ChannelName/  
SYSTEM.DEF.SVRCONN
```

- Topic string for a specific existing connection:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ConnectionId/  
414D5143514D475231202020202020206B576B5420000701
```

- Topic string to trace all channels on queue manager QMGR1:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ChannelName/#
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-19. Example activity trace topic strings*

This figure provides some examples of activity trace subscription strings.

## Formatting application activity trace output

- Write your own application
- Use IBM MQ `amqsact` sample program to format application activity trace messages
  - Default mode reads messages from `SYSTEM.DATA TRACE.ACTIVITY.QUEUE`
  - Dynamic mode subscribes to the system topic
 

<code>-m QMgrName</code>	Override default queue name
<code>-q QName</code>	Subscribe to event topic
<code>-t TopicString</code>	Browse messages
<code>-b</code>	Verbose output
<code>-v</code>	Number of messages to display
<code>-d depth</code>	Time to wait (in seconds)
<code>-w timeout</code>	Start time of message to process
<code>-s startTime</code>	End time of message to process
<code>-e endTime</code>	

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-20. Formatting application activity trace output

IBM MQ supports three options for formatting the application activity trace messages:

- You can use the `amqsact` sample program that is provided with IBM MQ. The source code is also provided, so you can easily see how it works and can extend and modify the behavior to meet your requirements.
- You can use SupportPac MSOP: WebSphere MQ Explorer Extended Management Plug-ins. This SupportPac provides an **Application Activity Trace** viewer. The viewer formats application activity trace messages and also sorts the trace messages by application. With the sort option you can easily view the flow of messages by application. After you install the SupportPac, you can format application activity trace messages by right-clicking on `SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE` in the content window and then clicking **Format Activity Records**.
- You can write your own application.

This figure lists the command options that you can use with `amqsact` sample program.

## Activity trace example

```
$ amqsact -m GATEWAY1 -v > out.txt
MQI Operation: 6
Operation Id: MQXF_PUT
ApplicationTid: 12451
OperationDate: '2017-04-09'
OperationTime: '01:39:48'
High Res Time: 1397003988665548
Completion Code: MQCC_OK
Reason Code: 0
...
Object_name: 'SENDINGAPP.REPLY'
Object_Q_mgr_name: 'GATEWAY1'
Resolved_Q_Name: 'SENDINGAPP.REPLY'
Resolved_Q_mgr: 'GATEWAY1'
Resolved_local_Q_name: 'SENDINGAPP.REPLY'
Resolved_local_Q_mgr: 'GATEWAY1'
...
Msg_id:
00000000: 414D 5120 4741 5445 5741 5931 2020 2020 'AMQ GATEWAY1 '
00000010: 0207 4453 2007 2603 '..DS .& .
Correl_id:
00000000: 414D 5120 4741 5445 5741 5931 2020 2020 'AMQ GATEWAY1 '
00000010: 0207 4453 2007 2203 '..DS .. '.
...
```

Check queue name resolution to find out why messages are going to the wrong place

Track individual messages and request/reply scenarios with Msg\_id and Correl\_id

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

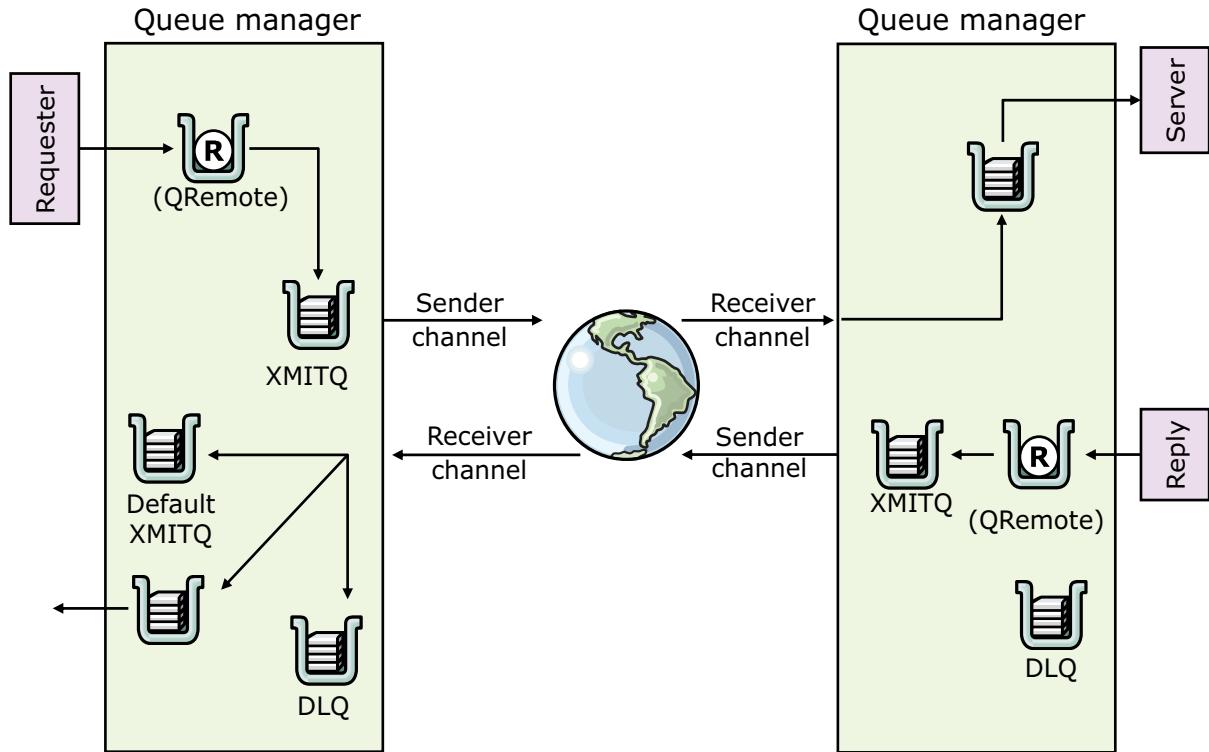
Figure 5-21. Activity trace example

This figure is an excerpt of an application activity trace record that was read by using the `amqsact` sample program.

In the activity report, you can check the queue name resolution to find out why messages are going to the wrong place. You can also track individual messages and request/reply scenarios with the message ID and correlation ID.

## 5.2. Message monitoring

## Where is the message?



More troubleshooting tools and techniques

© Copyright IBM Corporation 2017

Figure 5-22. Where is the message?

The figure shows a typical IBM MQ application with multiple queue managers and queues. A message might end up in many places after a successful MQPUT, particularly when it must cross systems.

IBM MQ does not lose messages by design, but messages can be lost for various reasons.

## Message monitoring

- Record activity information for a message as it is routed through a queue manager network
- Determine the routes that messages take through a queue manager network
- Use the IBM MQ trace-route messages to record activity information for a message
- Use the IBM MQ display route application (`dspmqrte`) to work with trace-route messages and activity information that is related to a trace-route message

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-23. Message monitoring*

This figure lists some of the options that are available in IBM MQ for monitoring messages.

## Recording message activity

- Requesting application must explicitly specify the report option
- An activity report contains information about the activity that was done on the message
  - Determine the last known location of a message
  - Determine configuration issues with a queue manager network



Do not enable all messages in a queue manager network for activity recording to avoid the increase of network traffic to an unacceptable level

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-24. Recording message activity

An activity report contains information about the activity on the message.

- An activity report can be used to determine the last known location of a message. Activity reports that are generated for the message as it was routed through a queue manager network can be studied to determine the last known location of the message.
- An activity report can be used to determine configuration issues with a queue manager network. By studying the activity reports that are related to messages that are enabled for activity recording, it can become apparent that they did not take the expected route. One reason a message might not take the expected route is a stopped channel, which forces the message to take an alternative route.

Messages and queue managers explicitly need the report option that is specified for the requesting applications to generate activity reports on behalf of the message as it is routed through a queue manager network.

You can use activity recording with trace-route messages by using the IBM MQ display route application.



### Attention

Avoid enabling all messages in a queue manager network for activity recording. Messages that are enabled for activity recording can have many activity reports that are generated on their behalf. If every message in a queue manager network is enabled for activity recording, the queue manager network traffic can increase to an unacceptable level.

## Activate message activity recording

- Request activity reports for a message in the application message descriptor
  - Specify MQRO\_ACTIVITY in the **Report** field
  - Specify the name of a reply-to queue in the **ReplyToQ** field

- Enable or disable the queue manager for activity recording in MQSC

**ALTER QMGR ACTIVREC()**

- Set to MSG to deliver activity reports to the reply-to queue specified in the message descriptor (default)
- Set to QUEUE to deliver activity reports to SYSTEM.ADMIN.ACTIVITY.QUEUE
- Set to DISABLED to disable queue manager activity recording

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-25. Activate message activity recording

To generate activity reports for a message as it is routed through a queue manager, complete the following actions.

1. Define the message to request activity reports.
2. Enable the queue manager for activity recording.

The figure describes how to complete these steps.

Ensure that your application uses the same algorithm as MCAs use to determine whether to generate an activity report for a message.

- Verify that the message requested that activity reports are generated.
- Verify that the queue manager that contains the message is enabled for activity recording.
- Put the activity report on the queue that the ACTIVREC queue manager attribute identifies.

If you do not want activity reports to be generated for a message as it is routed through a queue manager, disable the queue manager for activity recording.

## Trace-route messaging

- Dedicated trace-route messages can be put into the network and specify the following information:
  - Destination of the trace-route message
  - How the trace-route message mimics another message
  - How the trace-route message should be handled as it is routed through a queue manager network
  - Whether activity recording or trace-route messaging is used to record activity information
- Activity data is either reported or accumulated depending on message and queue manager settings
- Trace-route message is discarded just before it reaches its final destination queue

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-26. Trace-route messaging*

IBM MQ can inject trace messages in the message flow that are hidden from applications. These trace messages generate activity reports as they pass through an IBM MQ network.

You can use the IBM MQ tool `dsprmqrte` to view the information about the message routes so that you can see the route that missing messages might take. Trace-route messaging can help you find a problem by identifying the point that the trace message goes to the wrong location or where the message is discarded or lost.

You can also use trace-route messaging to test connectivity through the IBM MQ network and to test cluster workload balancing.

## Enabling queue manager for trace-route messaging

```
ALTER QMGR ROUTEREC()
```

- MSG
  - Applications within the scope of the queue manager can write activity information to the trace-route message
  - Trace-route reply message is generated, and delivered to the reply-to queue that is specified in the message descriptor of the trace-route message
- QUEUE
  - Applications within the scope of the queue manager can write activity information to the trace-route message
  - A trace-route reply message is generated and delivered to SYSTEM.ADMIN TRACE.ROUTE.QUEUE
- DISABLED
  - Queue manager is not enabled for trace-route messaging



When you modify the ROUTEREC queue manager attribute, a running MCA does not detect the change until the channel restarts

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-27. Enabling queue manager for trace-route messaging

To control trace-route messaging, use the MQSC command ALTER QMGR and specify an option for the ROUTEREC attribute. The figure lists the options for the ROUTEREC attribute.

## Display route application (dspmqrt)

- Generates trace-route messages
- Gathers responses
- Interpret and display results
- User added properties are also displayed

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-28. Display route application (dspmqrt)*

You can use the IBM MQ display route application (`dspmqrt`) to work with trace-route messages and activity information that is related to a trace-route message.

The display route application generates and puts a trace-route message into a queue manager network. As the trace-route message travels through the queue manager network, activity information is recorded. When the trace-route message reaches its target queue, the display route application collects and displays the activity information.

You can specify the characteristics of the trace-route messages.

- Specify the destination of the trace-route message.
- Specify how the trace-route message mimics another message.
- Specify how to handle the trace-route message as it is routed through a queue manager network.
- Specify whether activity recording or trace-route messaging are used to record activity information.

If the display route application puts a trace-route message into a queue manager network, the route information can be ordered and displayed immediately after the related activity information is

returned. Alternatively, the IBM MQ display route application can be used to order, and display activity information that is related to a trace-route message that was previously generated.

## Display route application parameters

Parameter categories	Parameters	
Queue manager connection	<code>-m <i>QMgrName</i></code>	Name of the queue manager to which the display route application connects
Target for trace-route message	<code>-q <i>TargetQName</i></code>	Name of the target queue
	<code>-qm <i>TargetQMgr</i></code>	Target queue manager
	<code>-o</code>	Target destination is not bound to a specific destination
Trace-route message properties	<code>-l <i>Persistence</i></code>	Persistence of the trace-route message
	<code>-p <i>Priority</i></code>	Priority of the trace-route message
	<code>-xs <i>Expiry</i></code>	Expiry time in seconds
	<code>-ac</code>	Accumulate activity information in the trace-route message
	<code>-t <i>details</i></code>	Specifies the activities that are recorded
	<code>-v <i>DisplayOption</i></code>	Select the amount of information to display

For a list of all parameters, see the IBM Knowledge Center for IBM MQ

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-29. Display route application parameters

This figure is a partial list of the parameters that can be used with the display route application. For a complete list of all parameters and their options, see the IBM Knowledge Center.

## dspmq rte sample output (1 of 2)

```
dspmq rte -m QM1 -q QRMT1 -v outline
AMQ8653: DSPMQ RTE command started with options '-m QM1 -q QRMT1 -v
outline'.
AMQ8659: DSPMQ RTE command successfully put a message on queue 'QM1', queue
manager 'QM1'.
AMQ8674: DSPMQ RTE command is now waiting for information to display.
-----
Activity:
ApplName: 'MQ\bin64\dspmq rte.exe'
Operation:
  OperationType: Put
  QMgrName: 'QM1'
  QName: 'QRMT1'
  ResolvedQName: 'QM1'
  RemoteQName: 'QL.A'
  RemoteQMGrName: 'QM1'
-----
Activity:
ApplName: 'MQ\bin64\runmqch1.exe'
Operation:
  OperationType: Get
  QMgrName: 'QM1'
  QName: 'QM1'
  ResolvedQName: 'QM1'
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-30. *dspmq rte sample output (1 of 2)*

This figure is the first of two that provides an example of the display trace route application.

In this example, the command specifies the queue (-q) and queue manager (-m). It also specifies outline information only (-v outline) in the results. An outline includes the application name, the type of each operation, and any operation-specific parameters.

The first activity in this example shows trace-route application (`dspmq rte`) putting the trace-route message on the queue that is named `fred` on queue manager `AJGMQ1`. It also shows that the local queue was resolved to a remote queue.

The second activity shows the sender-channel process (`runmqch1`) taking the message off the transmission queue and sending the message to the queue manager `AJGMQ2`.

## dspmqrte sample output (2 of 2)

```

Operation:
  OperationType: Send
  QMgrName: 'QMR1'
  RemoteQMGRName: 'QM1'
  ChannelName: 'QMR1.QM1'
  ChannelType: Sender
  XmitQName: 'QM1'

-----
Activity:
  ApplName: 'Sphere MQ\bin64\amqrmpa.exe'

Operation:
  OperationType: Receive
  QMgrName: 'QM1'
  RemoteQMGRName: 'QMR1'
  ChannelName: 'QMR1.QM1'
  ChannelType: Receiver

Operation:
  OperationType: Discard
  QMgrName: 'QM1'
  QName: 'QL.A'
  Feedback: NotDelivered

-----
AMQ8652: DSPMQRTE command has finished.

```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-31. *dspmqrte sample output (2 of 2)*

The third activity shows the channel-receiver process (`amqrmpa`) receiving the message off the channel AJGMQ1.TO.AJGMQ2 and attempting to deliver the message.

The last operation shows `Discard`, which is not unexpected because the trace-route message is typically discarded at the end of the route. However, the last operation also shows a feedback message of “UnknownObjectName”, which means that the application did not find the output queue.



### Note

You can specify whether the trace-route message is delivered to the target queue by setting `-d yes`. If you do not specify this parameter, the trace-route message is discarded at the end of the route.

## Hints and tips for using trace route

- Missing information does not always indicate a problem
  - Queue manager attribute on one computer might be set to QUEUE, rather than MSG
  - Path to reply queue might not be defined or available
- If user cannot PUT to reply queue activity, information seems to be missing
  - Disabling PASS\_DISCARD\_AND\_EXPIRY should send responses to the dead-letter queue if not otherwise delivered
  - If using `dspmqrte`, raise the verbosity level (`-v all`)
- If the last operation is a PUT to a transmission queue, then a channel might be down
- Default `dspqmrte` options should show the last operation as DISCARD
- Do not put trace-route messages to distribution lists

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-32. Hints and tips for using trace route

The figure lists some hints and tips for using trace-route messaging.

Missing information does not necessarily mean that a problem exists. It might mean that a previous version queue manager, or the trace-route-recording attribute of the queue manager is not set to message, or perhaps the path to the reply-to queue is not available.

You can change the amount of information that is displayed by using the `-v` option. If you need more information, rerun the application with the `-v all` option to display all information.

By default, the application should discard the trace-route message as the last operation. The message might be discarded for other reasons such as a missing or PUT disabled queue. Always check the **Feedback** field on the discard operation for the reason why the trace-route message is discarded.

Trace-route messages do not work with distribution lists; activity reports do work with distribution lists.

## Comparing activity recording and trace route

Benefit	Activity recording	Trace-route messaging
Can determine the last known location of a message	Yes	Yes
Can determine configuration problems with a queue manager network	Yes	Yes
Any message can request it (not restricted to trace-route messages)	Yes	No
Message data is not modified	Yes	No
Message is processed normally	Yes	No
Activity information can be accumulated in the message data	No	Yes
Optional message delivery to target queue	No	Yes
If a message is caught in an infinite loop, it can be detected and handled	No	Yes
Activity information can be put in order reliably	No	Yes
Application is provided to display the activity information	No	Yes

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-33. Comparing activity recording and trace route

Activity recording and trace-route messaging both have uses for locating problems in the IBM MQ network.

Both applications can be used to determine the last known location of a message, and they both can be used to check the configuration of queue manager networks.

Activity recording can be used with real messages. It is an option that is set on the input message. The report information that activity recording returns represents the real path of the message through the IBM MQ network. This reporting comes with a small processing impact but formatting this data requires custom code.

Trace-route messaging provides an application for formatting the data, but the trace-route message approximates the path of the application message. Trace-route messaging supports the accumulation of activity reports within the body of the trace-route message itself. So, the return data paths from an intermediate queue manager are not required as they are with activity reporting. When the time comes to report trace-route message data, it is automatically presented in the correct order.

## Exercise: Tracing message routes

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-34. Exercise: Tracing message routes*

In this exercise, you use a network of two queue managers that are connected with channels in a linear fashion, with remote queue definitions that pass a message from one queue manager to another and back. You use the display route application to trace the message and observe the results under different scenarios.

## Exercise objectives

- Use the IBM MQ display route application to determine the route that a message took through a queue manager network



[More troubleshooting tools and techniques](#)

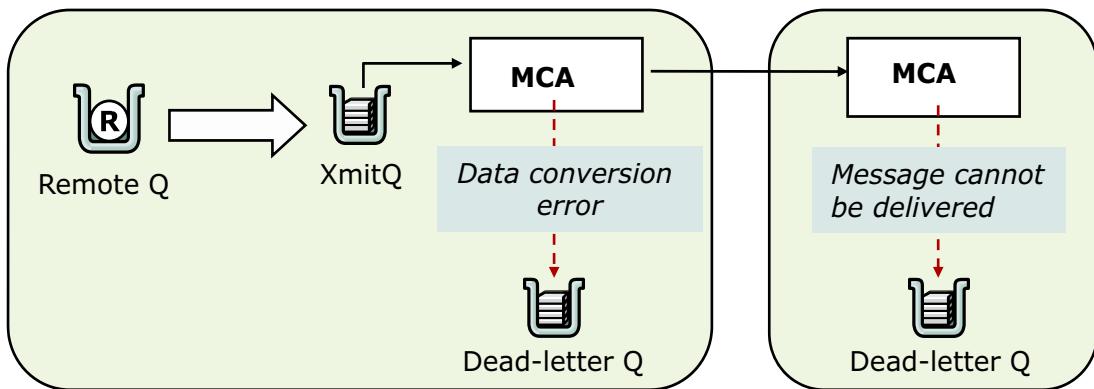
© Copyright IBM Corporation 2017

*Figure 5-35. Exercise objectives*

For detailed instructions, see the Course Exercise Guide.

## Dead-letter queues

- Messages are put on dead-letter queue when they cannot be delivered or returned
- Must be a local queue
- Maximum message length (MAXMSGL) attribute must be set to accommodate the largest messages that the queue manager must handle **plus** the size of the dead-letter header (MQDLH)
- USEDLQ channel attribute determines whether the dead-letter queue is used when messages cannot be delivered



More troubleshooting tools and techniques

© Copyright IBM Corporation 2017

Figure 5-36. Dead-letter queues

When a queue manager cannot deliver a message, it attempts to put the message on a local queue that is designated as its dead-letter queue. Define this queue when the queue manager is installed. Every queue manager has a SYSTEM.DEAD.LETTER.QUEUE that is created when the queue manager is created that can be designated as the dead-letter queue.

When the queue manager puts a message on this queue, it adds a header to the message. This header includes the name of the target queue and the reason that the message was put on the dead-letter queue. It must be removed and the problem must be resolved before the message is put on the intended queue.

When you configure the dead-letter queue properties, allow for the largest possible message size plus the dead-letter header when you specify the maximum message length.

## Using dead-letter queues

- Create a dead-letter queue on all queue managers
  - Use message-retry on message channels for transient conditions
  - Consider "return to sender" in the application

```
MQRO_PASS_MSG_ID +
MQRO_PASS_CORREL_ID +
MQRO_EXCEPTION_WITH_FULL_DATA +
MQRO_DISCARD_MSG
```

- Do not allow an application dead-letter queue to become full
- Can use the IBM MQ dead-letter queue handler to monitor and handle messages on a dead-letter queue

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-37. Using dead-letter queues*

The figure describes a strategy for using dead-letter queues.

Create a dead-letter queue on all queue managers. Creating a dead-letter queue is a two-step process:

1. Identify the dead-letter queue on the queue manager properties.
2. Create the dead-letter queue if it does not exist.

You can use the combination of report options that implements "return to sender" as an alternative to putting a message on the dead-letter queue. Using the "return to sender" option might mean that some messages are not put on the dead-letter queue.

A dead-letter queue must not be allowed to fill up, so it is important to monitor the depth of this queue.

Use the dead-letter queue handler that the arrival of a message on the dead-letter queue triggers. If no reasonable option to try again exists, forward the message to an application dead-letter queue.

## IBM MQ dead-letter queue handler

```
runmqd1q < Rules_Table
```

- Rules table contains a set of rules
  - Each rule consists of pattern matching keywords and an action
  - Specify queue name and queue manager name in first entry in rules table or as options in the command line
  - For each message on the dead-letter queue, each rule whose pattern matches the message is attempted in turn
  - A message can be tried again, forwarded, discarded, or ignored
  - A message can be forwarded with, or without, the dead-letter header
- Can have multiple instances, each with a different rules table
- For the list of pattern matching keywords, see the IBM Knowledge Center

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-38. IBM MQ dead-letter queue handler

The IBM MQ dead-letter queue handler provides an automated way of monitoring and handling messages on the dead-letter queue.

The dead-letter queue handler is started by using the `runmqd1q` control command. It can have a queue name and a queue manager name as parameters; it assumes the dead-letter queue of the default queue manager. A rules table, which is read from the standard input, drives the handler. There must be at least one rule in the table.

A rule can match the destination queue name, the message type, the contents of the **Feedback** field, and other message properties. A rule action can indicate that the dead-letter queue handler try again to put a message on its destination queue. It can also forward the message to another queue, or discard it, or leave it on the dead-letter queue.

It is possible to have multiple instances of the dead-letter queue handler that run concurrently against the same queue, each with a different rules table. However, it is more typical to have a one-to-one relationship between a queue and a dead-letter queue handler.

## Dead-letter queue handler sample

```

* Control data entry
* -----
* If no queue manager name is supplied as an explicit
* parameter to runmqd1q, use the default queue manager for
* the machine. If no queue name is supplied as an explicit
* parameter to runmqd1q, use the DLQ defined for the local
* queue manager.

inputqm(' ') inputq(' ')

* Rules
* -----
DESTQM(QMC01) ACTION(FWD) FWDQ(&DESTQ) FWDQM(QM17A)

MSGTYPE(MQMT_REPORT) FEEDBACK(MQFB_EXPIRATION) +
ACTION(DISCARD)

REASON(MQRC_Q_FULL) ACTION(RETRY)

DESTQ(XYZ*) ACTION(FWD) FWDQ(XYZ_DEADQ) FWDQM(' ')

```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-39. Dead-letter queue handler sample

The figure shows examples of dead-letter queue handler rules. In the rules, an action keyword describes how a matching message is processed follows a pattern matching keyword. DESTQM (destination queue manager) and REASON (reason code that describes why the message was put to the dead-letter queue) are example of pattern matching keywords.

The first rule takes messages that are intended for queue manager QMC01 (DESTQM) and forwards those messages. The messages are forwarded to the queue that is identified in the dead-letter header destination queue field (&DESTQ) on the queue manager that is named QM17A (FWDQM).

The second rule discards any message on the dead-letter queue that matches the pattern that is defined in the rule. In the rule, the message is discarded when the message type field contains MQMT\_REPORT and the FEEDBACK parameter indicates that the message was discarded because it was not removed from the destination queue before its expiry time elapsed (MQFB\_EXPIRATION).

The third rule attempts to forward the message to its destination queue if a message is placed on the dead-letter queue because its destination queue is full.

The fourth rule forwards any messages that were destined for queues with queue names that start with XYZ to the queue that is named XYZ\_DEADQ on the local queue manager as identified by FWDQM('').

## Using a dead-letter queue handler

- Trigger when message arrives on the dead-letter queue
- Possibly attempt further attempts to try again
- Do not let the dead-letter queue become full

ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)

- If unsuccessful, forward to application dead-letter queue associated with the following information:
  - Destination queue
  - Application that the *PutAppName* field specifies in the message descriptor

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-40. Using a dead-letter queue handler

You can set up the dead-letter queue handler to trigger automatically when a message arrives on the dead-letter queue.

In the dead-letter queue handler rules, you can specify that IBM MQ try to process the message again. You can also specify an interval at which the dead-letter queue handler attempts to reprocess messages that might not be processed at the first attempt.

With the FWDQ keyword in a rule, you can forward messages to another queue, such as an application-specific dead-letter queue.

## 5.3. Disaster recovery

## What is disaster recovery?

- The process, policies, and procedures that are related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster
- Focus is on the technology systems that support business functions
  - Getting applications to run after a major failure or loss
  - Business and often by regulators drive requirements

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-41. What is disaster recovery?*

Disaster recovery is the process, policies, and procedures that are related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster.

Disaster recovery is a subset of business continuity. Business continuity involves planning for keeping all aspects of a business that must continue to function in the midst of disruptive events. Disaster recovery focuses on the systems that support business functions.

Getting applications to run after a major (often whole-site) failure or loss is not about high availability although often the two are related and share design and implementation choices. High availability is about keeping systems active, while disaster recovery is about recovering when high availability fails.

Business and often regulators drive the requirements for disaster recovery.

## What to do if a queue manager stops unexpectedly

- Do not restart a queue manager until you know why it failed
- Take a backup of the queue manager data, log files, and any error logs, FDCs, and memory dumps
- Check IBM Support site for “known problems” and fixes

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-42. What to do if a queue manager stops unexpectedly*

A problem that you might encounter when you manage a queue manager network is a queue manager that stops running. The tendency is to try to restart the queue manager immediately after a failure; however, attempting to restart a queue manager without understanding the reason why it stopped might compound the problem. Do not restart a failed queue manager until you know why it failed.

Before you try a restart, ensure that you take a backup of the queue manager data, log files, and any error logs, FDC files, and memory dumps. Also, check the IBM Support website for known problems and software updates.

## What is the point of logging?

- A log record is written for each persistent update
  - Optimization to minimize serialization points
  - The log is always more up-to-date than the actual data
- Log is a sequential file
  - Sequential I/O is much quicker than random
  - Single point of writing rather than to individual object files
- Log and actual data are reconciled during queue manager start
- Point of consistency
  - Log control file `amqhlctl.1fh` in the log directory
  - Checkpoint `amqalchk.fil` in the queue manager directory

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-43. What is the point of logging?

Each update to persistent data is written to disk at least once. The first copy is a log record. The second copy can be the actual modified data on disk if the effect on the persistent data goes unchanged for a long time. This process might sound like unnecessary management but using the log has advantages.

- Log records are always written to the end of the log, whereas the updates to the data on disk are more or less random by comparison. The disk head is much more likely to be in the right place when it writes to the log, especially if the log has a dedicated disk drive.
- Special care is taken when IBM MQ writes log records to cope with power failures. This process is fairly simple with a sequential file.
- The log makes it easy to track the operations that make up transactions.

Writing the data twice does not mean that IBM MQ waits for the disk twice. In fact, message operations under sync point do not result in synchronous I/O until commit or rollback.

Non-persistent messages, even those messages that are written to disk, do not cause log records to be written.

- The log record describes the update in enough detail for the update to be re-created.
- The log records are written by using a protocol that is called *write-ahead logging*.

- The log record that describes an operation is guaranteed to arrive on disk before the data is updated.
- The log is never less up-to-date than the data.
- The contents of the log records can be used to update the data.

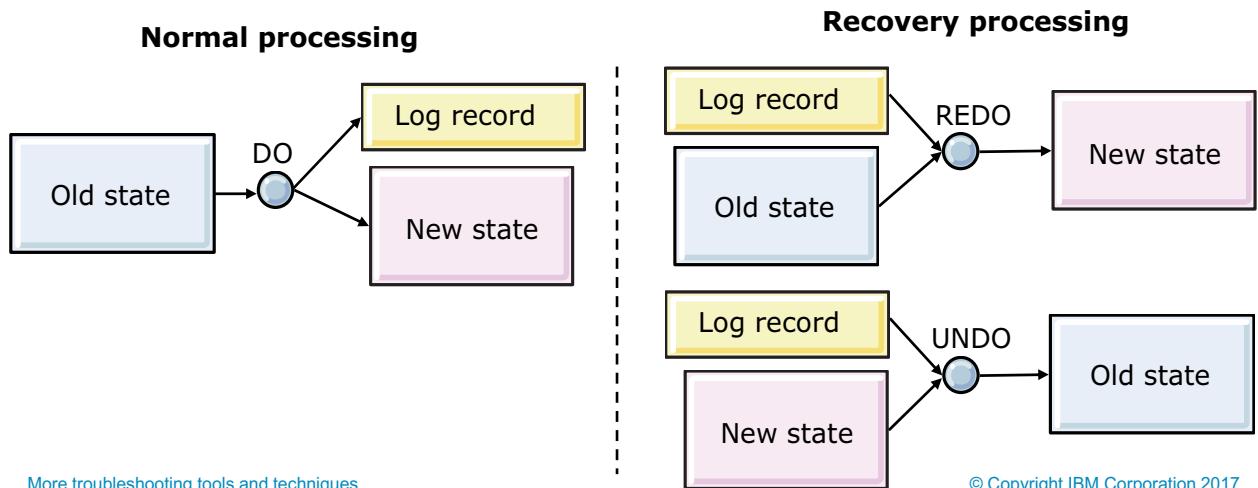
Periodically the log and data are synchronized for consistency. This point of consistency is called a *checkpoint*. At the end of a checkpoint, the queue files can be brought as up to date as the log at the start of the checkpoint if the queue manager was recovered.

During normal running, checkpoints are taken every 30 minutes provided there are at least 100 log record. A checkpoint is also taken when 10000 log records are written.

The log and data are reconciled when the queue manager is started. This process is called *restart recovery*. When the queue manager starts, displayed messages show the phases of reconciliation.

## DO, REDO, and UNDO operations

- DO: Log record contains an encapsulated version of the operation
- REDO: Contents of the log record and the old copy of the resources that the operation affects can be used to re-create the updated state of the resources from the last checkpoint
- UNDO: After the REDO, the contents of the log record and the updated copy of the resources that the operations affect can be used to re-create the state of the resources as they were before the operations



More troubleshooting tools and techniques

Figure 5-44. DO, REDO, and UNDO operations

© Copyright IBM Corporation 2017

IBM MQ uses a programming style that is known as DO-REDO-UNDO, which means that all operations on recoverable data are split into three operations.

- **DO:** During normal processing, each operation on recoverable data is completed and an associated log record is generated. The log record contains an encapsulated version of the operation.
- **REDO:** During recovery operations, resource managers might need to reapply changes that were originally made. The contents of the log record and the old copy of the resources that the operation affects can be used to re-create the updated state of the resources from the last checkpoint.
- **UNDO:** After the REDO phase certain operations must be undone, such as partial transactions. The contents of the log record and the updated copy of the resources that the operation affects can be used to re-create the state of the resources as they were before the operations were completed.

The DO-REDO-UNDO programming style is commonly used for resource and transaction managers. It relies that the correct information is logged. It also relies on the availability of programs that can complete the operations independently of the original application.

**Important**

During restart, the log must contain all the information necessary to allow the resource to be recovered without the intervention of any code other than the resource manager.

## Phases of restart recovery

- Replay and analysis phase
  - All log records after the last checkpoint are redone
  - Checkpoint is the last known point of consistency between the log and the object files
  - Transaction table is rebuilt during this phase
  - If the queue manager stopped cleanly (no in-flight transactions), the processing that is required is to replay the most recent checkpoint
- Indoubt phase
  - A list of transactions in-flight exists at the time that the queue manager ended
  - Scan backwards through the log from the last record the transaction wrote and to identify in-flight activities when the time the queue manager stopped
- Undo phase
  - For any transactions not recovered, roll back
  - Write Compensation Log Records (CLR)

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-45. Phases of restart recovery

Each time that you restart a queue manager, it goes through three phases of recovery to restore the queue manager to a consistent state.

### Phase 1: Replay and Analysis

During this first phase, all log records after the last checkpoint are redone. This process is known as repeated history.

Also, the transaction table is rebuilt. Any transactions that are mentioned in log records are added to a list as candidates for rollback in the final phase of restart. The state of all of the transactions that are found is also maintained during this phase. When the log records for the end of a transaction are found, the transaction is removed from the list.

If the queue manager stopped cleanly (no in-flight transactions), the only processing that is required is to replay the most recent checkpoint. In this case, there is no work to do in the next two phases.

### Phase 2: Indoubt

In this phase, IBM MQ has a list of transactions that were in-flight at the time that the queue manager ended. For each transaction, IBM MQ scans backward through the log from the last record that the transaction wrote. It follows the links between records in the same transaction and determines what the transaction was doing when the queue manager stopped.

### Phase 3: Undo

Any transactions in the list that are not prepared are rolled back. Rollback involves writing special “undo” log records called Compensation Log Records (CLRs). A CLR contains an image that corresponds to the “before image” of the log record it is undoing.

After the CLR is written, the Indoubt phase ignores the operation that it describes if the restart is interrupted.

At the end of restart, IBM MQ might have some prepared transactions. The Indoubt phase creates a reconstructed list of operations that make up the transaction so that the transactions can be committed or rolled back when the transaction manager calls them.

## First manual restart of the queue manager

- Clean the IPC (shared memory and semaphores) at the request of IBM Support only
- Start the queue manager as simply as possible  
Example: `strmqm -ns QM1`
- Monitor the restart, look for FDCs
  - If queue manager is OK after restart, then end the queue manager and restart normally
  - If queue manager fails to start, option to escalate to “cold” start

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-46. First manual restart of the queue manager*

After you backed up the log and identified the cause of the program, restart the queue manager. Clean the IPC (shared memory and semaphores) only if IBM Support requests it.

Start the queue manager as simply as possible. For example, start the queue manager with the `-ns` option on the `strmqm` command: `strmqm -ns QM1`

The `-ns` option prevents any of the following processes from starting automatically when the queue manager starts:

- Channel initiator
- Command server
- Listeners
- Services

When you restart the queue manager, monitor the restart for any errors. If the queue manager starts without any errors, then stop the queue manager and restart it following standard procedures.

If the restart fails, the next options include trying a “cold” start or rebuilding the queue manager.

## Cold start of IBM MQ

- “Cold start” is a technique to restart without needing logs
  - Replace the “bad” logs with “good” logs by creating a dummy queue manager and by using its logs instead
- Typical reason: Hardware (most likely disk) failure or logs deleted by mistake
- Symptoms: Cannot start queue manager because logs are not available or files are corrupt

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-47. Cold start of IBM MQ

It might be necessary to *cold start* the IBM MQ queue manager, such as is a hardware failure occurs or where an administrator mistakenly deleted the logs. A cold start is a technique that starts a queue manager without logs.

## Cold start considerations

- Disadvantages of a cold start
  - In-flight transactions are not automatically rolled-back
  - In-doubt transactions are forgotten
  - Cannot recover messages from the logs
  - Possible loss of messages
  - Possible duplication of already-processed messages
- Is this queue manager part of a cluster?
- Is this queue manager under the control of a high-availability cluster?

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-48. Cold start considerations*

A cold start affects message integrity in the following ways:

- In-flight transactions are not automatically rolled back
- In-doubt transactions are forgotten
- Messages cannot be recovered from the logs
- Messages might be lost
- Messages that were already processed might be duplicated

Another consideration is the relationship of this queue manager to other queue managers. For example, is this queue manager part of a cluster.

## Cold start procedure (1 of 2)

1. Create a temporary queue manager that is EXACTLY like the one that failed
  - Use queue manager initialization file (`qm.ini`) to get parameters for the `crtmqm` command

Example `qm.ini` stanza

Log :

```
LogPrimaryFiles=10
LogSecondaryFiles=10
LogFilePages=65535
LogType=CIRCULAR
```

2. Enter the `crtmqm` command to create temporary queue manager
  - Make sure that enough space exists for the new log files in the directory

Example:

```
crtmqm -lc -lf 65535 -lp 10 -ls 10 -ld /tmp/mqlogs TEMP.QMGR
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

Figure 5-49. Cold start procedure (1 of 2)

This figure lists the first steps in a queue manager cold start.

The name that you use of the queue manager in a cold start is not important. The goal of the cold start is to create log files that you can use to restore the queue manager that failed.

The create queue manager command in the figure, creates a queue manager with circular logging (-lc), 65535 log file pages (-lf), 10 primary log file (-lp), and 10 secondary log files (-ls). The log files are created in the `/tmp/mqlogs` directory (-ld). The options that you use in the create queue manager command are based on the log file options in the queue manager initialization file for the failed queue manager (shown in Step 1 in the example).

## Cold start procedure (2 of 2)

### 3. Replace old logs and amqlctl.1fh with the new ones

- If messages are persistent, data in the queues are preserved
- Object definitions are preserved

Example:

```
cd /var/mqm/log  
mv QM1 QM1.SAVE  
mv /tmp/mqlogs/TEMP!QMGR QM1
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-50. Cold start procedure (2 of 2)*

The final step in the cold start procedure is to replace the old log files and the log control file (**amqlctl.1fh**) with the new files.

In the example, the existing queue manager files are saved before the new files replace them.

Using this technique for a cold start of a queue manager, object definitions and persistent messages are preserved.

## Rebuilding a queue manager

- Create a replacement queue manager with same object definitions
- Make sure that you:
  - Have a backup of the definitions
  - Know which version of IBM MQ is installed
  - Have the security configuration
  - Identify any customization requirements in the `qm.ini` file
- Replacement queue manager has a new QMID
  - IBM MQ Explorer saves QMID in its list of known queue managers
  - You can connect to queue manager, but requires confirmation that the new QMID is expected
- For clustered queue manager, enter RESET CLUSTER at full repository to remove the old QMID before bringing the replacement online

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-51. Rebuilding a queue manager*

The last resort for recovering from a queue manager failure is to rebuild the queue manager. Before you rebuild the queue manager, make sure that you have a backup of the definitions. You must also verify the version of IBM MQ that is installed.

## Copying or moving contents of a queue to a file

- Use `dmpmqmsg` utility to copy or move the contents of a queue, or its messages, to a file
- Possible uses:
  - Saving the messages that are on a queue for archiving
  - Reloading a queue with messages
  - Removing old messages from a queue
  - 'Replaying' test messages from a stored location



The file has a specific format that the utility understands. Do not change the format of this file.

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-52. Copying or moving contents of a queue to a file*

With the `dmpmqmsg` utility, you can copy or move the contents of a queue, or its messages, to a file. This file can be saved for backup and used later to reload the messages onto the same queue or another queue.

## Using the dmpmqmsg utility

- Save the messages that are on a queue, into a file

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile
```

- Load a queue from a file:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile
```

- Copy the messages from one queue to another queue

```
dmpmqmsg -m QM1 -i Q1 -o Q2
```

- Move the messages from one queue to another queue

```
dmpmqmsg -m QM1 -I Q1 -o Q2
```

- Display the ages of messages currently on a queue

```
dmpmqmsg -m QM1 -i Q1 -f stdout -dT
```

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-53. Using the dmpmqmsg utility*

This figure lists examples of the `dmpmqmsg` utility for saving, loading, copying, moving, and displaying information about messages.

For a complete description of the command options, see the “`dmpmqmsg`” topic in the IBM Knowledge Center.

## Planning for recovery

- Write a disaster recovery plan
- Identify the information that you need to recover IBM MQ
- Identify the relationship and impact with other resources and applications
  - Applications must be able to connect and target the *exact* same IBM MQ resources
  - Every existing, in-flight, message must be processed while in disaster recovery mode
- Test the disaster recovery plan frequently
- Might need different plans for different disaster scenarios

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-54. Planning for recovery*

The key component to disaster recovery is a disaster recovery plan. Ensure that this plan documents the configuration of every component and the steps to recover from a disaster.

The person that uses this plan in a real emergency might not have all of the prerequisite skills so provide detailed, step-by-step procedures.

Test the plan frequently. Each time that you test the plan, you are likely to discover a step or task that you forgot.

Update the plan any time you add or update applications, hardware, or software.

## Unit summary

- Use IBM MQ events to detect problems in your queue manager network
- Uses trace-route messages to determine the last known location of a message and determine configuration problems with a queue manager
- Use the IBM MQ unload and load utility to copy messages to or from a file
- Implement a dead-letter queue handler to automate the handling of messages on the dead-letter queue
- Recover IBM MQ if a failure occurs

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-55. Unit summary*

## Review questions

1. True or False: An application that connects to IBM MQ must handle activity reporting.
2. Which troubleshooting tool can you use to detect and handle a message that is caught in an infinite loop?
  - A. Activity recording
  - B. Trace-route messaging
  - C. Dead-letter queue handler
3. The first step to restarting a queue manager that fails is to clean the shared memory and semaphores.



[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-56. Review questions*

Write your answers here:

- 1.
- 2.
- 3.

## Review answers

1. True or False: An application that connects to IBM MQ must handle activity reporting.  
The answer is False. Activity reporting is an optional feature; is it not required for an application to support activity reporting.
  
2. Which troubleshooting tool can you use to detect and handle a message that is caught in an infinite loop?
  - A. Activity recording
  - B. Trace-route messaging
  - C. Dead-letter queue handlerThe answer is B.
  
3. True or False: The first step to restarting a queue manager that fails is to clean the shared memory and semaphores.  
The answer is False. Clean the IPC (shared memory and semaphores) at the request of IBM Support only.



More troubleshooting tools and techniques

© Copyright IBM Corporation 2017

Figure 5-57. Review answers

## Exercise: Handling messages on the dead-letter queue

[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-58. Exercise: Handling messages on the dead-letter queue*

In this exercise, you configure IBM MQ to automatically handle messages that arrive on the dead-letter queue by using the dead-letter queue handler.

## Exercise objectives

- Configure the queue manager to use a dead-letter queue
- Handle dead-letter messages



[More troubleshooting tools and techniques](#)

© Copyright IBM Corporation 2017

*Figure 5-59. Exercise objectives*

For detailed instructions, see the Course Exercises Guide.

# Unit 6. High availability

## Estimated time

01:00

## Overview

In this unit, you learn about the IBM MQ high availability solutions.

## How you will check your progress

- Review questions

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Plan for using high availability systems with IBM MQ
- Configure and manage a multi-instance queue manager

High availability

© Copyright IBM Corporation 2017

Figure 6-1. Unit objectives

## 6.1. Planning for high availability

## High availability objectives

- Achieve 24x7 processing of all messages
- Avoid application awareness of availability solutions

Availability %	Downtime per year
99	3.65 days
99.9	8.76 hours
99.99	52.6 minutes
99.999	5.26 minutes
99.9999	30.00 seconds

[High availability](#)

© Copyright IBM Corporation 2017

*Figure 6-2. High availability objectives*

The objective of high availability is to achieve 24x7 processing of all messages. The availability percentage determines the downtime per year.

The goal is for applications to process messages continually, regardless of any failures in any components. Service Level Agreements (SLAs) define the level of availability that applications and services provide.

## Potential outage types

- 80% scheduled downtime:
  - New software release
  - Upgrades
  - Maintenance
- 20% unscheduled downtime:
  - 40% operator error
  - 40% application error
  - 20% other (network failure, disk failure, power outage)

High availability

© Copyright IBM Corporation 2017

*Figure 6-3. Potential outage types*

In general, 80% of system downtime is for scheduled tasks such as installing new software and system maintenance. The other 20% of system downtime is unscheduled. Operator errors, applications, and other failures in the hardware or network can cause unscheduled downtime.

High availability (HA) solutions can increase availability given scheduled or unscheduled downtime.

## Some terms

- Fault tolerance and continuous availability: Handle almost any type of failure without interruption to service
- High availability: Service can be interrupted but reappears quickly with no manual intervention
- Disaster recovery: Recovery from large system or site loss
- Scalability: The ability to add extra servers to achieve extra throughput
- Workload balancing: The ability to spread the load across multiple servers

[High availability](#)

© Copyright IBM Corporation 2017

*Figure 6-4. Some terms*

With fault tolerance and continuous availability, a system can cope with most types of failure. No downtime is apparent to the applications and no connections are lost.

With high availability, some limited downtime is permissible. The service might be interrupted, but reappears quickly with no manual intervention.

Disaster recovery is the process, policies, and procedures that are related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster.

Scalability is the ability to add extra servers and get higher throughput. Scalability is often achieved by running cloned servers.

Workload balancing spreads the load where the network contains multiple potential servers. Most workload balancing algorithms (for example round-robin, send to least-loaded) are aimed at maximizing throughput. Workload balancing can give the appearance of high availability by spreading the load to available servers. If one of the servers stops, then no new work is given to it and client applications do not notice the failure. The surviving servers have a higher workload until the stopped server is restarted.

## Single points of failure

- With no redundancy or fault tolerance, a failure of *any* component can lead to loss of availability
- Every component is critical
  - Power supply, processor, memory
  - Disk controllers, disks, network adapters, cables
- Various techniques exist to help tolerate failures
  - Dual power supplies and uninterruptible power supplies
  - RAID for disk resiliency and recovery
  - Fault tolerant architectures for processor and memory
- High availability clustering
  - Used with some of the techniques to help tolerate failures
  - Cost effective means of avoiding single points of failure

High availability

© Copyright IBM Corporation 2017

Figure 6-5. Single points of failure

A single point of failure is any component whose failure results in the end of availability of the system as a whole. Most systems contain many such single points of failure, which include the points that are listed in the figure.

Many products and techniques are available to reduce the likelihood of complete system failure due to the failure of a single component. Most solutions involve hardware duplication such as dual power supplies and even uninterruptible power supplies. Redundant Array Of Independent Disks (RAID) provides similar redundancy but spread across many devices.

## Failover strategies

- Cold standby
  - Primary component runs actively while secondary or backup component stays dormant
  - When the primary component fails, secondary component is activated to assume an active role
- Warm standby
  - Primary component runs actively while secondary or backup component runs without active participation in workload management
  - Secondary component receives frequent data updates from the primary component
  - When the primary component fails, the secondary component assumes an active role
- Hot standby
  - Primary and secondary components run as single, unified system
  - When the primary component fails, the secondary component continues without interruption to users

High availability

© Copyright IBM Corporation 2017

Figure 6-6. Failover strategies

In a cold standby failover strategy, the primary component runs actively while the secondary or backup component stays dormant. When the primary component fails, the secondary component is activated to assume an active role. The interruption is visible to users.

In a warm standby failover strategy, the primary component runs actively with the secondary or backup component runs without active participation in workload management. The secondary component receives frequent data updates from the primary component, which means that times might occur when both components are not synchronized. When the primary component fails, the secondary component assumes an active role. Because the secondary component was running passively with partial data, the failover is faster than a cold standby, with minimal interruption to users.

In a hot standby failover strategy, the primary and secondary components that are run as a single, unified system. Active data replication happens between the primary and secondary components. When the primary component fails, the secondary component continues to function without interruption to the users.

## IBM MQ high availability technologies

- Queue manager clusters
- Multi-instance queue managers
- Client reconnection
- Support for networked storage
- Support for high availability (HA) clusters
  - On UNIX and Linux: PowerHA for AIX, Veritas Cluster Server, HP Serviceguard, or a Red Hat Enterprise Linux cluster with Red Hat Cluster Suite
  - On Windows, Microsoft Cluster Service (MSCS)

High availability

© Copyright IBM Corporation 2017

Figure 6-7. IBM MQ high availability technologies

IBM MQ has many features for providing and supporting high availability technologies if you want to operate your IBM MQ queue managers in a high availability configuration.

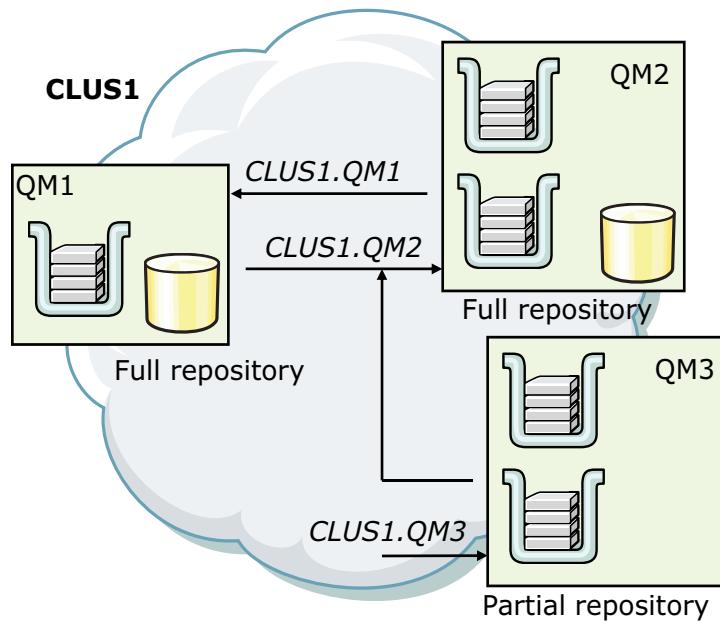
The basic support that IBM MQ provides for high availability includes queue manager clusters, and support for networked storage.

You can expand this basic high availability support by setting up your queue managers to work in an HA cluster with an external HA coordinator, such as PowerHA for AIX or the Microsoft Cluster Service on Windows.

If you do not want to add or support an external HA coordinator, you can create multiple instances of a queue manager.

IBM MQ also supports automatic client reconnection, which is an important part of a highly available IBM MQ network.

## IBM MQ clusters as a high availability strategy



High availability

© Copyright IBM Corporation 2017

*Figure 6-8. IBM MQ clusters as a high availability strategy*

IBM MQ clusters are part of high availability solution. The key features of IBM MQ clusters are:

- Automatic definition of channels
- Automatic discovery of target queues
- Automatic rerouting of messages when a target computer is unavailable
- Workload balancing, with an exit capability to override default choices

Each queue manager is an independent entity, but they share configuration and topology information by using a repository queue manager that uses messages and queues for its integrity.

This solution works for new messages. However, if a remote computer fails then any messages that are on that queue manager (old messages), or that are still in-doubt on the sending computer, are not available.

In a scenario where different queue managers or servers can process a message, ensure that equivalent services are available. Equivalent services must be available so that the requesting application does not need to know that the service moves around.

## IBM MQ clustering high availability considerations

- Sharing cluster queues on multiple queue managers prevents a queue from a single point of failure
- Cluster workload algorithm automatically routes traffic away from failed queue managers
- Ensure that full repositories are highly available
  - Full repositories propagate all definition changes
  - Define at least two full repositories for all clusters regardless of size
  - Must be fully interconnected

[High availability](#)

© Copyright IBM Corporation 2017

*Figure 6-9. IBM MQ clustering high availability considerations*

Full repositories must be fully connected with each other by using manually defined cluster-sender channels and cluster-receiver channels.

Ensure that there are always at least two full repositories in the cluster so that if one full repository fails, the cluster can still operate. If only one full repository exists and it loses its information about the cluster, then manual intervention on all queue managers within the cluster is required to get the cluster to work again. If the network contains two or more full repositories, the information is always published to and subscribed from two full repositories; the failed full repository can be recovered with minimal effort.

Create full repositories on servers that are reliable and highly available. If no full repositories are available in the cluster for a short time, it does not affect application messages that are being sent by using the clustered queues and channels. It does mean that the clustered queue managers do not find out about administrative changes in the cluster until the full repositories are active again.

For most clusters, two full repositories are adequate. It is possible to have more than two full repositories, which can be useful for placing the full repositories in different geographical locations, so that partial repositories use local full repositories.

## Support for networked storage

- Support for queue manager data in networked storage
  - Network-attached storage (NAS) so that data is available to multiple computers concurrently
  - Storage area network (SAN) to access a dedicated network that provides access to consolidated, block level storage
  - Added protection against starting concurrent instances of a queue manager that use the same queue manager data
- Some clients have a “no local disk” policy for queue manager data

High availability

© Copyright IBM Corporation 2017

*Figure 6-10. Support for networked storage*

Queue manager data can be kept in networked storage by using network-attached storage (NAS) or a storage area network (SAN) so that data is available to multiple servers concurrently.

Networked storage also provides added protection against starting two instances of the same queue manager by using the same queue manager data.

Some clients have a “no local disk” policy for queue manager data, so support for networked storage enables this topology.

## Client configuration for availability

- Use wildcards in queue manager names in CCDT
  - Gets weighted distribution of connections
  - Selects a “random” queue manager from an equivalent set
- Use multiple addresses in a CONNAME
- Use automatic client reconnection
- Use IP routers to select address from a list that is based on workload or anything else that is known to the router

High availability

© Copyright IBM Corporation 2017

*Figure 6-11. Client configuration for availability*

IBM MQ also supports various options for configuring clients for high availability.

In a highly available implementation, clients must be able to connect to the active queue manager and update the connection information when the active queue manager changes.

With IBM MQ, you can use wildcard characters in queue manager names in the CCDT, which sets up a choice of connections.

You can also configure a list of IP addresses in the CONNAME attribute when you configure the channel. The connection tries the IP addresses in the order that it appears in the list until a connection is established.

IBM MQ also supports options for automatically reconnecting the client after a dropped connection.

An alternative to providing a solution in IBM MQ, is to use an IP router that selects an address from a list that is based on workload or other information.

## Automatic client reconnection

- For a non-JMS client:
  - Set reconnection variable (**DefRecon**) in the `mqclient.ini` file
  - Use a CCDT to connect to a queue manager
  - If the client connects to a multi-instance queue manager, provide the network addresses of the active and standby queue manager instances in the CCDT
- For a JMS client, set the reconnection options in the connection factory configuration
- In an application, set MQCNO options on MQCONN call to `MQCNO_RECONNECT` or `MQCNO_RECONNECT_Q_MGR`

[High availability](#)

© Copyright IBM Corporation 2017

*Figure 6-12. Automatic client reconnection*

To configure automatic client reconnection for a non-JMS client:

- Set the **DefRecon** variable in the `mqclient.ini` file.
- Use a CCDT to connect to a queue manager.
- If the client is to connect to a multi-instance queue manager, provide the network addresses of the active and standby queue manager instances in the CCDT.

For a JMS client, set the reconnection options in the connection factory configuration. When it runs inside the EJB container of a Java EE server, message-driven beans (MDBs) can reconnect to IBM MQ by using the reconnect mechanism that is provided by activation specifications of the IBM MQ resource adapter (or listener ports if it runs in IBM Application Server). However, if the application is not an MDB (or is running in the web container) the application must implement its own reconnect logic because automatic client reconnect is not supported in this scenario. The IBM MQ resource adapter provides this reconnect ability for the delivery of messages to message driven beans, but other Java EE elements such as servlets must implement their own reconnection.

For all applications, you must also modify the MQCONN call from the client application to send the `MQCNO_RECONNECT` or the `MQCNO_RECONNECT_Q_MGR` option.

## Failover and IBM MQ

- Failover is the automatic switching of availability of a service (queue manager)
- Traditionally provided by an HA cluster and requires:
  - Data that is accessible on all servers
  - Equivalent or compatible servers
  - Common software levels
  - Sufficient capacity to handle workload after failure
  - Start processing of queue manager that follows the failure
- IBM MQ offers two ways of configuring for failover:
  - Multi-instance queue managers
  - HA clusters

High availability

© Copyright IBM Corporation 2017

Figure 6-13. Failover and IBM MQ

Failover is the automatic switching of availability of a service. For IBM MQ, the service is a queue manager.

A high availability (HA) cluster coordinator traditionally handles failover. It also requires strict adherence to hardware and software configuration.

- Data must be accessible on all servers.
- Ensure that all servers are at the same software levels.
- Ensure that servers have sufficient capacity to handle the workload after a failure. The workload might be rebalanced after failover, which requires spare capacity.
- A way to control the start processing of a queue manager after the failure must exist.

IBM MQ supports two methods of configuring for failover:

- Multi-instance queue managers
- HA clusters with an external HA coordinator

## Failover considerations

- Data access
  - Shared disks (not concurrent but “switchable”)
  - Alternatively mirrored data (must be true, synchronized mirror)
- Network connectivity and IP address take-over
- Node equivalence
  - Common hardware
  - Common software (levels, paths)
- Performance and capacity sufficient to handle maximum workload
- Take-over time
  - Time to notice the failure
  - Time that it takes to establish the environment before activating the service
  - Time that it takes to restart the queue manager
- Loss of nonpersistent messages and nondurable subscriptions

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-14. Failover considerations

Failover considerations include the following requirements.

- **Data access.** All systems must have access to the queue manager data. Normally shared disks provide data access. This configuration does not necessarily mean concurrent access, but near-concurrent access by using switching is acceptable.
- **Network connectivity.** The ability to switch an IP address between nodes is essential for take-over.
- **Node equivalence.** Ensure that all nodes have similar hardware and software, including things like patch levels and paths. Node equivalence also eases maintenance of the nodes.
- **Performance and capacity.** Ensure that each node can handle the full workload of all queue managers for which it is able to take-over. Failure to account for the full workload can cause significant degradation during a take-over.
- **Take-over time.** Be aware of long running transactions that can result in long restart times.

Failover involves a queue manager restart so non-persistent messages and nondurable subscriptions are discarded. For the fastest restart time, avoid long running transactions, if possible.

## 6.2. HA clusters

## IBM MQ in HA clusters

- Queue manager data and logs are placed on a shared disk and the disk is switched between servers during failover
- Queue manager has its own “service” IP address
  - IP address is switched between servers during failover
  - Queue manager’s IP address stays the same after failover
- Channel state is hardened
  - Sender channels are automatically restarted (if triggered)
  - Virtual IP address must be used on all incoming sender channels
  - Requester channels are not automatically restarted
- Some services might require manual restart:
  - Trigger monitors
  - Command server
  - Applications
- Effectively a normal queue manager restart
  - Non-persistent messages are lost
- Long running transactions slow down restart

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-15. IBM MQ in HA clusters

In an HA cluster, the queue manager data and logs are on a separate shared disk.

In an HA cluster, each queue manager has its own IP address, which is switched between the active and standby servers during failover.

In an HA cluster, channel state is hardened, which means that state of the channels before the takeover is preserved across the takeover. Sender channels are automatically restarted with the correct trigger specification. For the takeover to not be apparent to incoming sender channels and clients, they must use the virtual IP address in the CONNAME parameter of their channels.

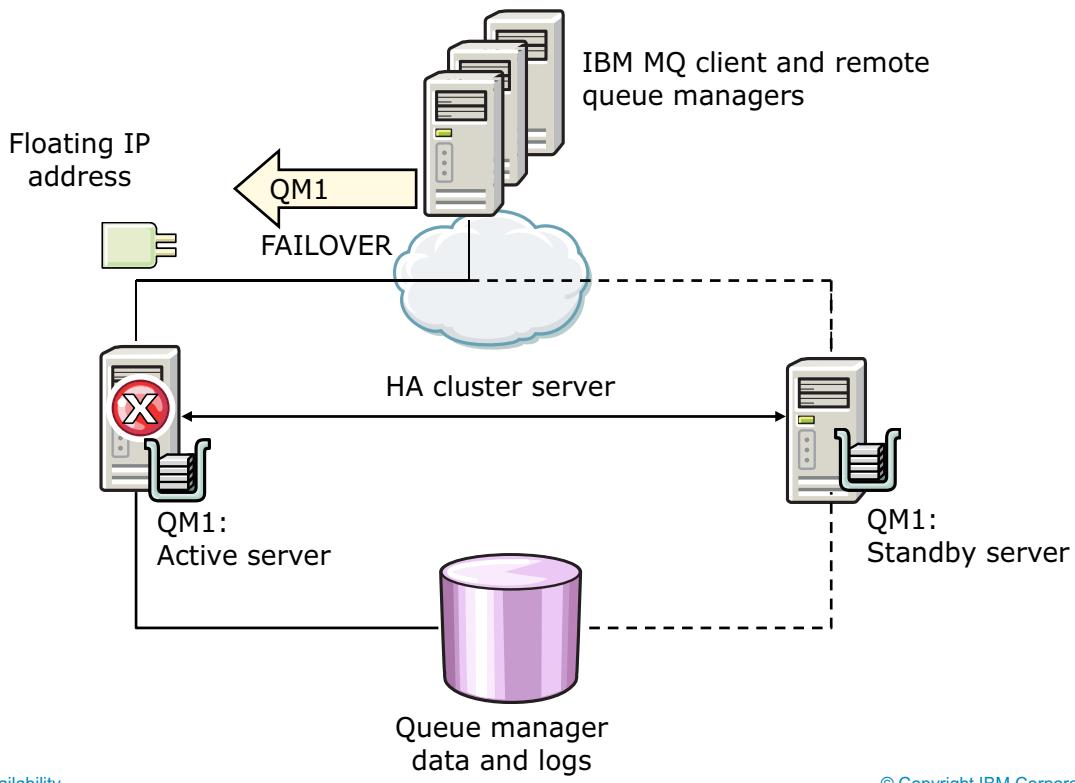
Requester channels are not automatically restarted, but it is not likely that too many requester channels exist on a server.

In an HA cluster, some services like trigger monitors, command servers, and applications must be manually restarted, or have custom scripts and automation to achieve automatic restart. Some of these services depend on the options that are specified with the `crtmqm` command.

In an HA cluster, a queue manager takeover is effectively a queue manager restart, so all non-persistent messages are lost.

Long running transactions are not conducive to quick takeover restarts.

## Standby failover



High availability

© Copyright IBM Corporation 2017

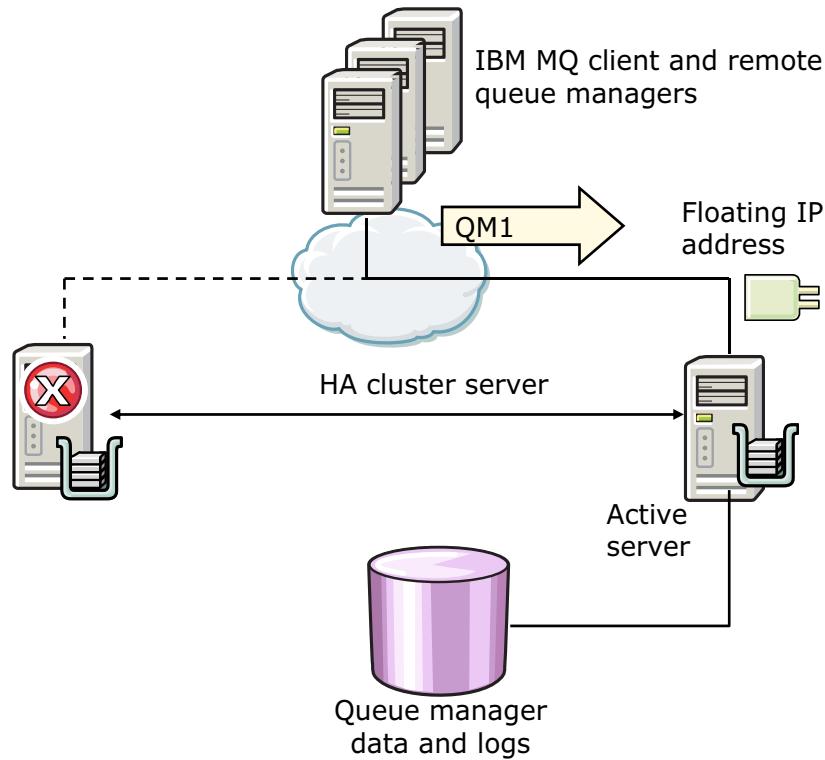
Figure 6-16. Standby failover

A standby failover configuration includes an active queue manager on one server and a standby queue manager on another server. Both servers have access to a shared disk for queue manager data and logs. If the active server fails, the standby node initiates a failover to the standby queue manager.

In this example, the active server failed. The standby node detects the failure, typically over a private network connection. The standby node takes control of the shared disks and the IP address.

This example shows a shared disk. Alternatively, disk mirroring can be used instead of shared disk. An advantage of disk mirroring is geographical separation, but latency limits the distance that can be achieved. For reliability, any transaction logs must be the same, which means any synchronous disk writes must also be sent before it is confirmed.

## Standby failover complete



High availability

© Copyright IBM Corporation 2017

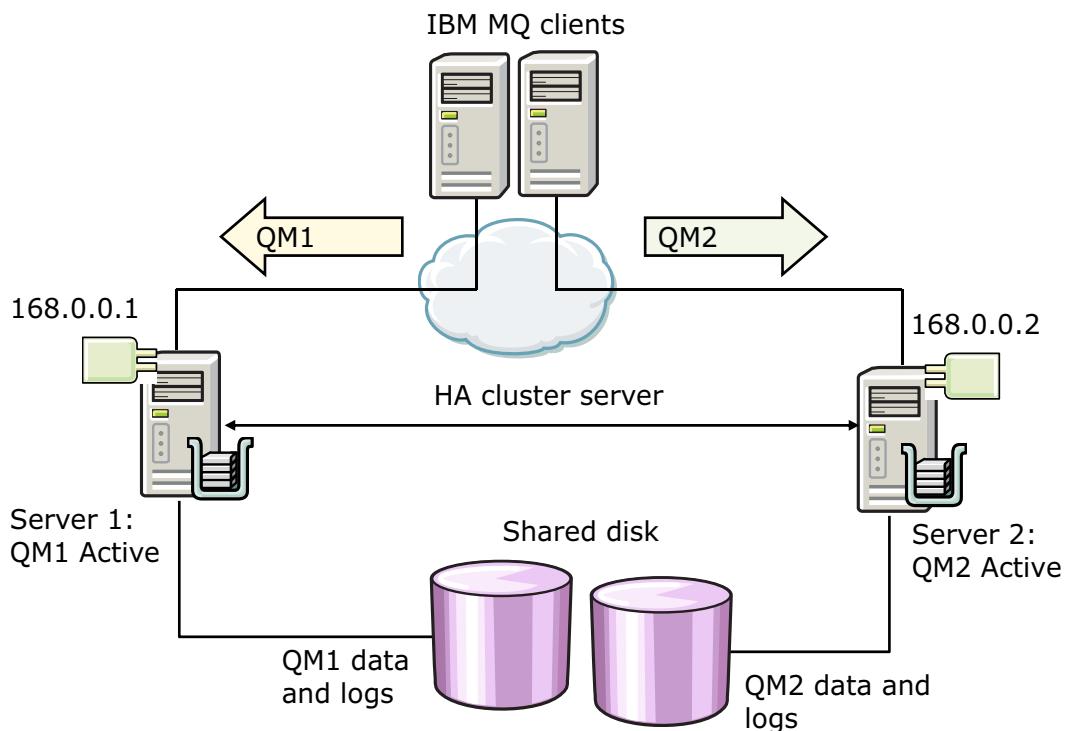
Figure 6-17. Standby failover complete

When the failover is complete, the standby server is designated as the active server. It has control of the queue manager data and logs.

The virtual IP address now directs incoming traffic to the new, active server.

When the original active server is restored, a process that is known as *fail-back* is initiated to return service to the original active server. This process is essentially the reverse of the failover process.

## Active-active configuration



High availability

© Copyright IBM Corporation 2017

Figure 6-18. Active-active configuration

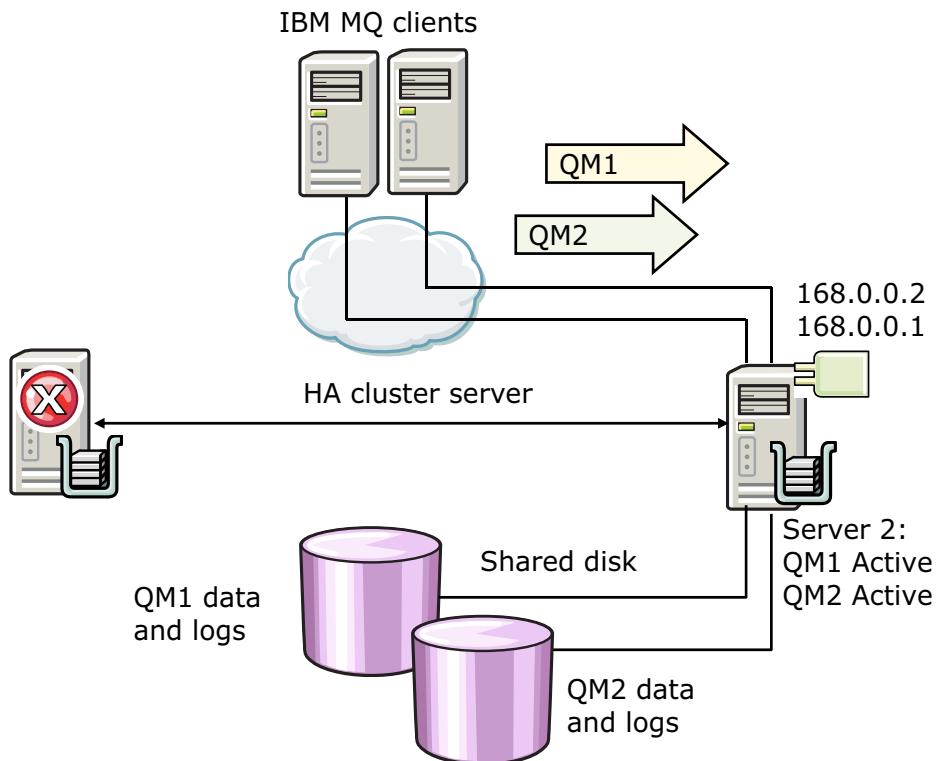
An active-active configuration is a more advanced configuration in which all nodes process some messages and a node can take over critical work from another node if it fails. A “one-sided” standby configuration, a standby node does some additional, noncritical, and nonmovable work.

This configuration is similar to a standby configuration but this configuration has a standby node that completes the noncritical work. In an active-active configuration, all nodes complete highly available (movable) work. This type of cluster configuration is called active-active to indicate that all nodes are actively processing critical workload. The figure shows an example of an active-active configuration.

With an active-active configuration, it is important to consider the peak load that can be placed on any node that can take over the work of other nodes. Such a node must possess sufficient capacity to maintain an acceptable level of performance.

Each queue manager must have sufficient capacity to handle the work of both queue managers or such a setup does not prove acceptable.

## Active-active takeover



High availability

© Copyright IBM Corporation 2017

Figure 6-19. Active-active takeover

The active-active configuration is also sometimes called a *mutual takeover* system.

In normal operation, both servers are running independent queue managers. If one of the systems fails, this configuration can switch the failed queue manager to the working computer. To applications outside the HA cluster, two queue managers still exist. The throughput of each queue manager might degrade, depending on how heavily they are loaded and how much spare capacity existed, but at least the work is still getting done.

In an active-active configuration, the IP address that is associated with each queue manager is switched. You also need to keep a port number reserved for each queue manager and have listeners that are defined correctly.

With this setup, ensure that the HA cluster provides a fail-back capability so that the queue manager can be sent back to its original node when the failure is corrected. It would be advisable to monitor the workload and initiating fail-back only when it does not disrupt too much work.

Many variations of cold and hot standby are possible; for example, you can have three nodes to host two queue managers. The availability of these options depends on your HA cluster software.

## Benefits of using IBM MQ with HA clusters

- Coordination of multiple resources such as application server and database
- Consist of more than two servers
- Failover more than once without operator intervention
- Queue manager is defined to the HA cluster as a resource that depends on the shared disk and the IP address
  - During failover, the HA cluster switches the disk, takes over the IP address, and starts the queue manager
- Likely to be more resilient in cases of IBM MQ and operating system defects

High availability

© Copyright IBM Corporation 2017

Figure 6-20. Benefits of using IBM MQ with HA clusters

Using IBM MQ with an HA cluster coordinator provides many benefits.

First, the HA coordinator can coordinate multiple resources, not just IBM MQ.

Also, an HA cluster configuration supports more than one active queue manager at a time, so both queue manager can actively process messages.

The HA coordinator also handles the failover process without any manual intervention.

## Implementing HA clusters on UNIX

1. Configure the shared disks
  - HA cluster requires data files and log files in common, named, remote file systems on a shared disk
2. Create the queue manager for use in the HA cluster
3. Add the queue manager configuration information to the other nodes in the HA cluster
4. Start the queue manager under the control of the HA cluster by using a shell script or external application

High availability

© Copyright IBM Corporation 2017

*Figure 6-21. Implementing HA clusters on UNIX*

This figure lists the steps for implementing an HA cluster on UNIX. Each of these steps is described in detail on the next pages.

## Configuring the shared disk

1. Decide on names of the mount points for the queue manager file systems.  
Example: /MQHA/QM1HA/data for data files  
/MQHA/QM1HA/log for log files
2. Create a volume or disk group to contain the queue manager data and log files.
3. Create the file systems for the queue manager data and log files in the volume group.
4. For each node, create the mount points for the file systems and verify that the file systems can be mounted

High availability

© Copyright IBM Corporation 2017

Figure 6-22. Configuring the shared disk

A queue manager in an HA cluster requires data files and log files in common named remote file systems on a shared disk.

The example in the figure assumes that the data and log directories for the queue manager are both on the shared disk, which is mounted at /MQHA/QM1HA. This disk is switched between the nodes of the HA cluster when failover occurs so that the data is available wherever the queue manager is restarted.

## Creating the queue manager for the HA cluster

1. Mount the queue manager's file systems on the node
2. Create the queue manager and identify log and data file directories  
Example:  
`crtmqm -md /MQHA/QM1HA/data -ld /MQHA/QM1HA/log QM1HA`
3. Start the queue manager manually by using the `strmqm` command
4. Complete the queue manager configuration
  - Create queues and channels
  - Start a listener automatically when the queue manager starts
5. Stop the queue manager by using the `endmqm` command
6. Use the `dspmqinf` command to display the `addmqinf` command that you use later to add the queue manager instance to the other nodes  
Example: `dspmqinf -o command QM1HA`
7. Unmount the queue manager file systems

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-23. Creating the queue manager for the HA cluster

The next step is to create the queue manager on one of the nodes.

To create a queue manager for use in an HA cluster, select one of the nodes in the cluster on which to create the queue manager. On this node, complete the steps that are listed in the figure.

## Adding queue manager to other nodes in HA cluster

1. Mount the queue manager file systems
2. Add the queue manager configuration information to the node
  - Option 1: Edit the `/var/mqm/mqs.ini` file
  - Option 2: Enter the `addmqinf` command

Example:

```
addmqinf -s QueueManager -v Name=QM1HA  
          -v Directory=QM1HA\ -v Prefix=/var/mqm  
          -v DataPath=/MQHA/QM1HA/data/QM1HA
```

3. Start and stop the queue manager to verify the configuration
4. Unmount the queue manager file systems

High availability

© Copyright IBM Corporation 2017

Figure 6-24. Adding queue manager to other nodes in HA cluster

After you create the queue manager on one of the nodes, you must add the queue manager configuration to the other nodes in the HA cluster.

## Starting the queue manager under HA control

```
#!/bin/ksh
# The script must be run by the mqm user.
# The only argument is the queue manager name.
QM=$1
if [ -z "$QM" ]
then
    echo "ERROR! No queue manager name supplied"
    exit 1
fi
# End any queue manager processes which might be running.
srchstr="( | -m) $QM *.*$"
for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
amqzlsa0 runmqchi runmqlsr amqcrsta amqrmmfa amqrmpa \
amqzfuma amqzmuf0 amqzmur0 amqzmgr0
do
    ps -ef | tr "\t" " " | grep $process | grep -v grep | \
    egrep "$srchstr" | awk '{print $2}' | \
    xargs kill -9 > /dev/null 2>&1
done
# It is now safe to start the queue manager.
# The strmqm command does not use the -x flag.
strmqm ${QM}
```

Sample shell  
script

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-25. Starting the queue manager under HA control

The queue manager is represented in the HA cluster as a resource. The HA cluster must be able to start and stop the queue manager. In most cases, you can use a shell script to start the queue manager.

The shell script in this figure is an example of starting a queue manager without making any assumptions about the current state of the queue manager. This script uses an abrupt method of ending any processes that belong to the queue manager.

Make these scripts available at the same location on all nodes in the cluster, either by using a network file system or by copying them to each of the local disks.

## IBM PowerHA cluster

- IBM high availability solution for Power Systems that run AIX
- Supports multiple applications that run over a number of nodes with shared or concurrent access to the data on shared disk storage
- Supports active-standby and active-active configurations

High availability

© Copyright IBM Corporation 2017

Figure 6-26. IBM PowerHA cluster

PowerHA for AIX is an application that makes system fault resilient and reduces downtime of applications.

PowerHA uses networks to detect and diagnose failures, and provides clients with highly available access to applications.

On PowerHA for AIX, the unit of failover is called a *resource group*.

## Microsoft Cluster Service (MSCS)

- Support integrated with IBM MQ
- Capabilities:
  - Active-active configuration
  - Unit of failure is queue manager (with one or more queue managers per MSCS group)
  - Automatic queue manager registration on standby node
  - Security and registration synchronization
  - Supports IBM MQ custom services
- Utility programs:
  - Register the resource type                    `haregtyp /r`
  - Unregister the resource type                `haregtyp /u`
  - Remove queue manager from node        `hadltmqm /m QmgrName`
  - Check and save setup details            `amqmsysn`
  - Move a queue manager to MSCS storage:  
`hamvmqm /m QmgrName /dd "DataDirectory" /ld "LogDirectory"`

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-27. Microsoft Cluster Service (MSCS)

The Microsoft Cluster Service (MSCS) is an HA clustering solution. Support for MSCS is integrated with IBM MQ to provide active-active configurations.

The MSCS unit of failure is the queue manager in the MSCS group. For clustering to work with IBM MQ, you need an identical queue manager on node B for each one on node A. However, you do not need to explicitly create the second one. You can create or prepare a queue manager on one node, and then duplicate it on another node.

Security synchronization is managed by adding the servers to a domain or by setting up a subdomain. MSCS also synchronizes the registries of the nodes.

MSCS also supports IBM MQ custom services like listeners by leaving the listener to autostart dependent on the queue manager start. This way, when MSCS starts the queue manager, dependent services like the listener are automatically restarted at the right time.

The utility programs that are listed on the figure are utilities that are supplied with IBM MQ to support running IBM MQ with MSCS.

For more information about configuring MSCS and IBM MQ for an HA cluster, see the IBM Knowledge Center.

## 6.3. Multi-instance queue managers

## Multi-instance queue managers

- Basic failover support without HA cluster
- Two instances of the same queue manager on different servers
  - Active instance owns the queue manager files and accepts connections from applications
  - Standby instance monitors the active instance
  - If active instance fails, standby instance restarts queue manager and becomes active
- One set of queue manager data that is kept in networked storage

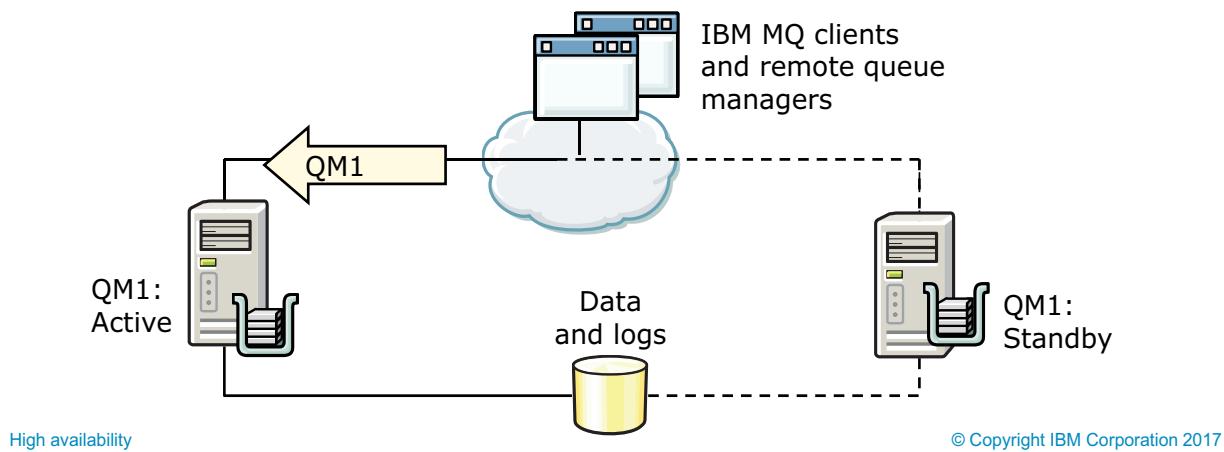


Figure 6-28. Multi-instance queue managers

Multi-instance queue managers provide basic failover support without an HA cluster. In this implementation for high availability, two instances of the same queue manager exist on different servers. One is the active instance and the other is the standby instance.

- The active instance controls the queue manager's files and accepts connections from applications.
- The standby instance monitors the active instance. If the active instance fails, the standby restarts the queue manager and becomes active.

Each instance shares a single set of data files that are held in the networked storage.

If a multi-instance queue manager instance is already active on a *different* server, the new instance becomes a standby, allowing it to take over from the active queue manager instance. While it is in standby, the queue manager cannot accept local or remote connections.

## Creating multi-instance queue managers

1. Set up shared file systems for queue manager data and logs
2. Create the queue manager on server 1 with pointers to shared stored for data and logs

Example:

```
crtmqm -md /shared/qmdata -ld /shared/qmlog QM1
```

3. Add configuration information for the queue manager to server 2

Example:

```
addmqinf -v Name=QM1 -v Directory=QM1 -v Prefix=/var/mqm  
-v DataPath=/shared/qmdata/QM1
```

4. Start the (active) queue manager on server 1.

Example:

```
strmqm -x QM1
```

5. Start the (standby) queue manager on server 2.

Example:

```
strmqm -x QM1
```

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-29. Creating multi-instance queue managers

This figure lists the steps for creating a multi-instance queue manager for high availability.

The `-x` option on the start queue manager command, starts an instance of a multi-instance queue manager on the local server, making it highly available. If an instance of the queue manager is not already running elsewhere, the queue manager starts and the instance becomes active. The active instance is ready to accept local and remote connections to the queue manager on the local server.



### Windows

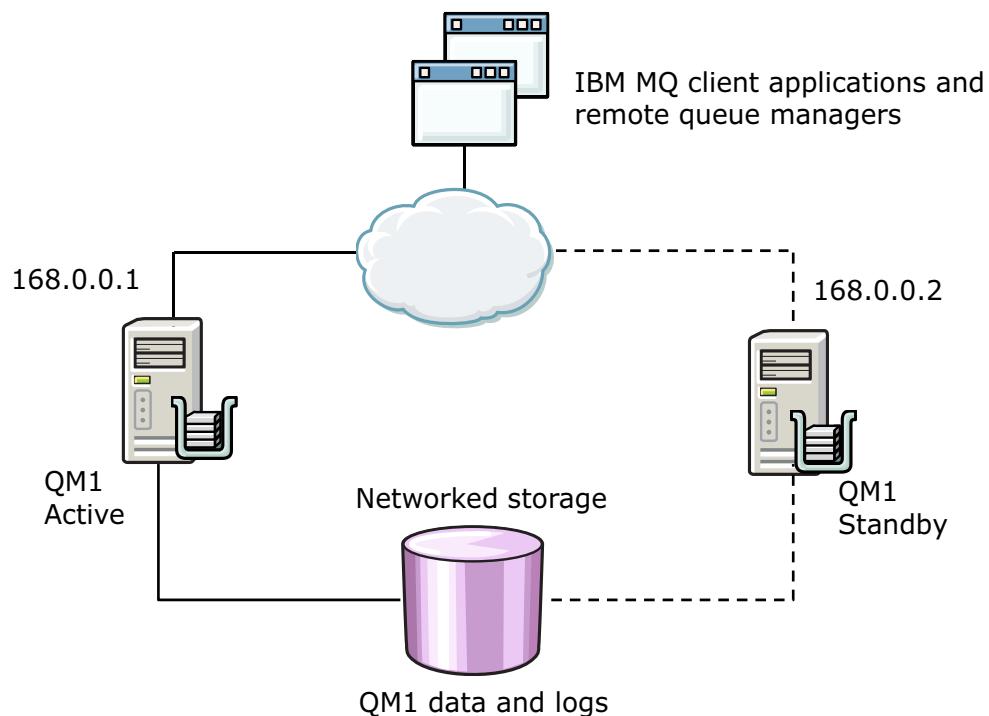
All instances must run on domain controllers.



### Important

Do not start a second instance of a queue manager on the *same* server.

## Standby configuration



[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-30. Standby configuration

With multi-instance queue managers, the active queue manager owns the queue manager data.

In this configuration, the active queue manager provides processing. The standby queue manager is idle and does not process messages.

This type of a configuration requires a high degree of hardware redundancy. To economize on hardware, you can extend this configuration to multiple worker nodes with a single standby so that the standby can take over the work of any other worker node. This configuration is still known as a standby configuration and sometimes as an “N+1” configuration.

This figure shows a typical standby configuration. IBM MQ clients and other remote queue managers all attempt to connect to a queue manager by using a single queue manager name and IP address. However, its IP address is a virtual IP address. Connections to this IP address are directed to a secondary IP address, which is the real address of the current active queue manager server. One node runs an application or service, while another standby node waits and monitors the active node. This configuration is sometimes called a cold standby.

## Standby failover

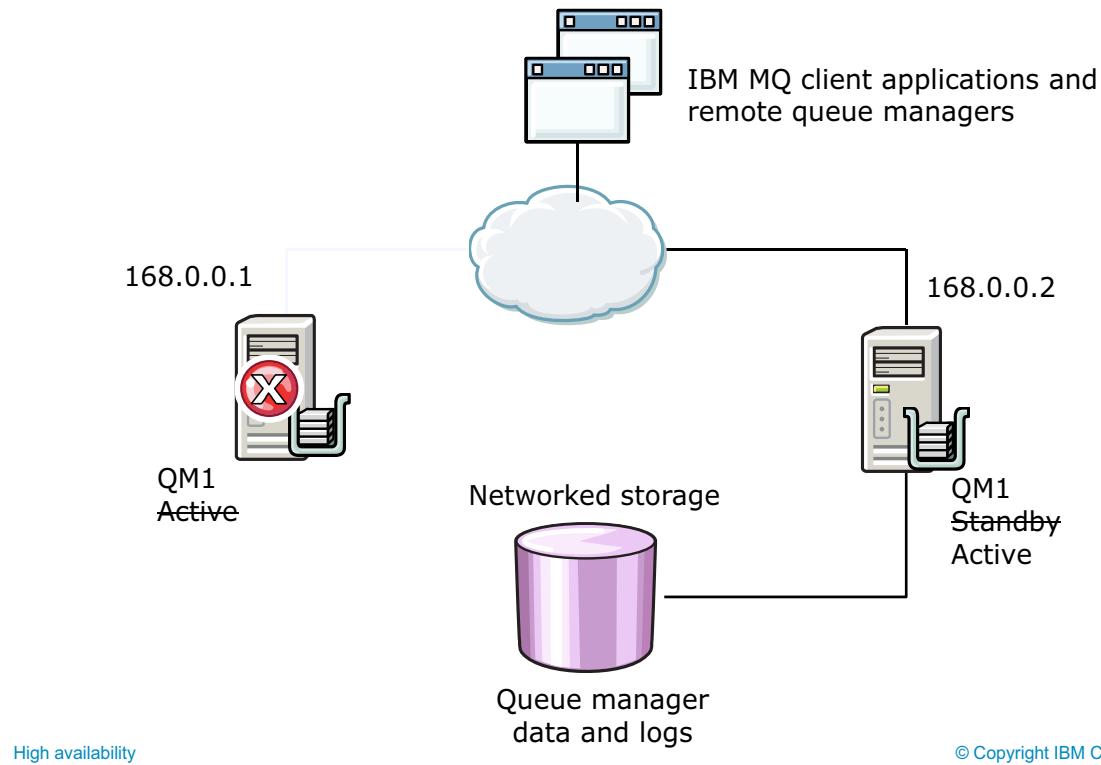


Figure 6-31. Standby failover

During the failure, client connections are broken. When the standby queue manager becomes active, the client connections activate on the new queue manager. The active queue manager now owns the queue manager data.

## Handling multiple IP addresses in multi-instance

- IP address of the queue manager changes active queue manager moves during failover
- IBM MQ channel configuration needs way to select address
- Multiple IP address options:
  - Connection name syntax is extended to a comma-separated list  
Example: `CONNNAME ('168.0.0.1,168.0.0.2')`
  - Use external IP address translation or intelligent router

High availability

© Copyright IBM Corporation 2017

Figure 6-32. Handling multiple IP addresses in multi-instance

The IP address of the queue manager changes when it moves from the active instance to the standby instance, so the IBM MQ channel configuration needs a way to select an address.

One way to identify multiple IP addresses is by providing the IP addresses in a comma-separated list in the CONNAME attribute when you configure the connection.

## Multi-instance queue manager administration

- IBM MQ is *not* an HA cluster coordinator
  - Coordination of other resources requires an HA cluster
  - Queue manager services can be automatically started, but with limited control
- System administrator is responsible for restarting another standby instance when failover occurs
- All queue manager administration is on the active instance
  - Use `dspmq -x` command to identify active and standby instances

Example:

```
$ hostname
  server2
$ dspmq -x
QMNAME (QM1)           STATUS (Running as standby)
INSTANCE (server1)     MODE (Active)
INSTANCE (server2)     MODE (Standby)
```

[High availability](#)

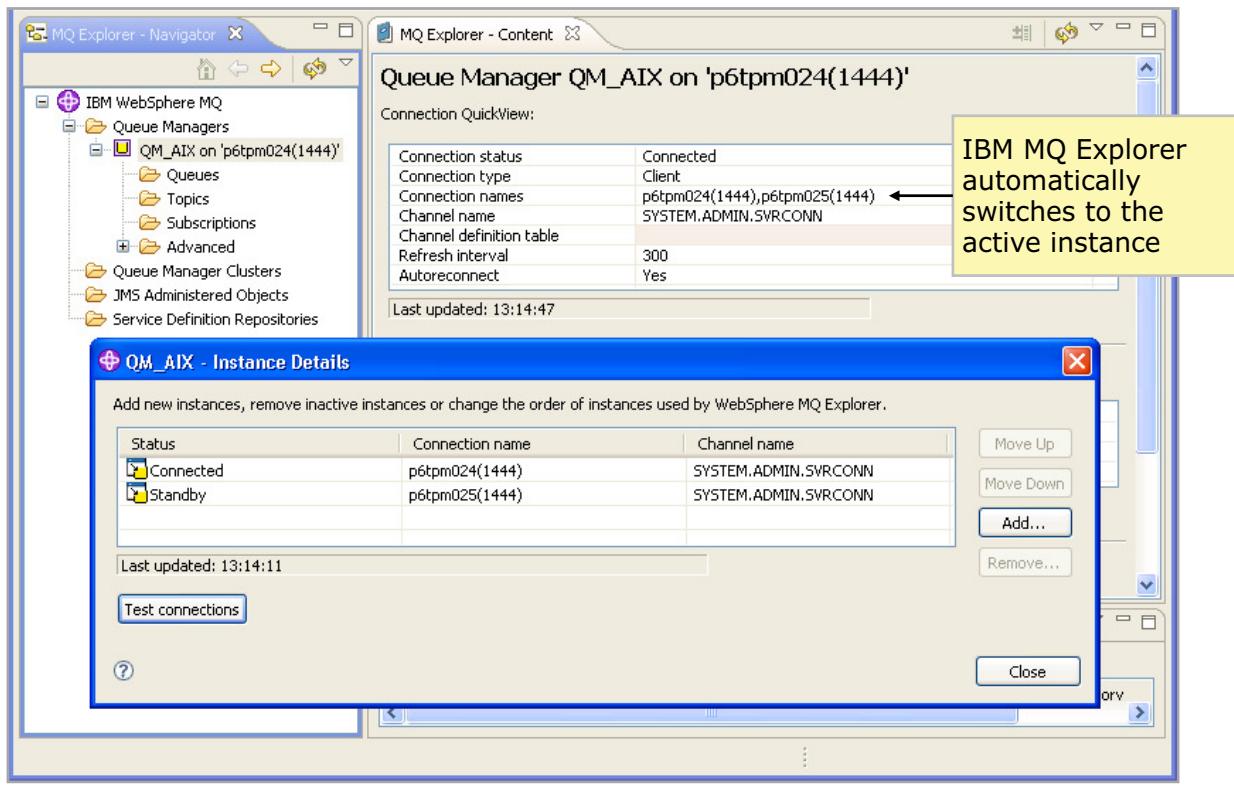
© Copyright IBM Corporation 2017

Figure 6-33. Multi-instance queue manager administration

IBM MQ is not an HA cluster coordinator, so the system administrator must complete many of the tasks that an HA cluster coordinator handles. For example, in a multi-instance queue manager configuration, the system administrator is responsible for restarting a standby instance when failover occurs.

All queue manager administration is done on the active instance. To determine the active instance, enter the `dspmq -x` command, as shown in the figure.

## Managing multi-instance queue manager in IBM MQ Explorer



High availability

© Copyright IBM Corporation 2017

Figure 6-34. Managing multi-instance queue manager in IBM MQ Explorer

You can manage multi-instance queue managers in IBM MQ Explorer.

When you have a multi-instance queue manager, both instances are shown in the **Connection name** row on the queue manager content view. You can double-click the **Connection name** row to display the instance details.

## Comparing multi-instance queue managers and HA clusters

### Multi-instance queue manager

- Advantages
  - Basic failover support that is integrated into IBM MQ
  - Faster failover than HA cluster
  - Simple configuration and operation
  - Integration with IBM MQ Explorer
- Limitations
  - Requires highly available, high-performance networked storage
  - More complex network configuration because queue manager changes IP address when it fails over

### HA cluster

- Advantages
  - Can coordinate multiple resources
  - More flexible configuration options
  - Can fail over multiple times without operator intervention
  - Takeover of queue manager's IP address as part of the failover
- Limitations
  - Requires external product and skills
  - Requires disks that can be switched between the nodes of the cluster
  - Configuration of HA clusters is relatively complex
  - Failover can be slow
  - Unnecessary failovers can occur

[High availability](#)

© Copyright IBM Corporation 2017

Figure 6-35. Comparing multi-instance queue managers and HA clusters

This figure highlights the advantages and limitations of the multi-instance queue managers and HA cluster solutions for high availability.

Multi-instance queue managers solution provides a high availability solution with basic failover support. It does require highly available, high-performance network storage, and extra configuration for handling IP addresses.

The HA cluster solution, requires a separate HA cluster coordinator such as PowerHA on AIX. The advantage of the HA cluster solution is that it can coordinate with multiple resources. It also supports more configuration options, such as an active-active configuration where both queue managers are actively processing messages. With its use of shared, highly available disk, means that all persistent messages are available after a failure.

## Unit summary

- Plan for using high availability systems with IBM MQ
- Configure and manage a multi-instance queue manager

High availability

© Copyright IBM Corporation 2017

*Figure 6-36. Unit summary*

## Review questions

1. True or False: All HA solutions employ a floating IP address to affect transparent failover.
2. True or False: Failover is effectively a queue manager restart, so all non-persistent messages are lost.



[High availability](#)

© Copyright IBM Corporation 2017

*Figure 6-37. Review questions*

Write your answers here:

1.

2.

## Review answers

1. True or False: All HA solutions employ a floating IP address to affect transparent failover.  
The answer is True.
  
2. True or False: Failover is effectively a queue manager restart, so all non-persistent messages are lost.  
The answer is True. However, due to transparency of the failover, applications might fail in unexpected ways.



High availability

© Copyright IBM Corporation 2017

Figure 6-38. Review answers

---

# Unit 7. Introduction to distributed publish/subscribe

## Estimated time

01:30

## Overview

In this unit, you learn about the publish/subscribe support in IBM MQ. The unit describes how to use IBM MQ commands and IBM MQ Explorer to define and manage publications and subscriptions.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Describe publish/subscribe in IBM MQ
- Explain distributed publish/subscribe topologies
- Manage publish/subscribe topics, subscriptions, and topologies
- Compare publish/subscribe topologies

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-1. Unit objectives*

## 7.1. Publish/subscribe overview

## What is publish/subscribe?

- Application model in which the publisher of information is decoupled from the subscriber of that information
- Publishers
  - Providers of information (topics)
  - Do not need to know subscribers
- Subscribers
  - Consumers of information
  - Do not need to know providers
- New publishers and subscribers can be added without disruption

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

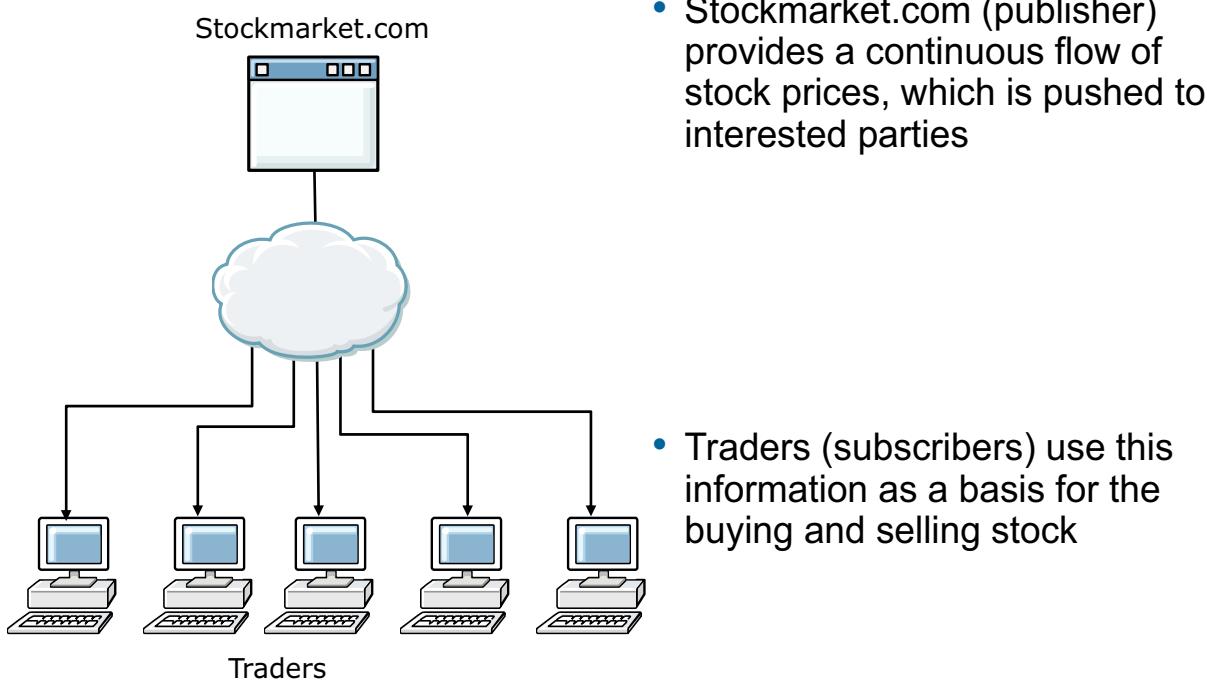
Figure 7-2. What is publish/subscribe?

Publish/subscribe is the mechanism by which subscribers can receive information, in the form of messages, from publishers.

Publisher applications of data messages do not need to know the identity or location of the subscriber applications that receive the messages. Similarly, the subscribing applications do not need to know the identity or location of the publishing applications that provide their information. In this sense, the providers and consumers are said to be *loosely coupled*.

- Publishers supply information about a subject, without needing to know anything about the applications that are interested in that information. Publishers generate this information in the form of messages, called publications that they want to publish and define the topic of these messages.
- Subscribers create subscriptions that describe the topic that the subscriber is interested in. So, the subscription determines which publications are forwarded to the subscriber. Subscribers can make multiple subscriptions and can receive information from many different publishers.

## Publish/subscribe example



[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-3. Publish/subscribe example*

The most-commonly quoted example of a publish/subscribe system is one that provides stock-market information.

In this example, the stock market application at Stockmarket.com (publishes) a continuous flow of information that contains the most recent stock prices. Traders require the current stock price information so that they can conduct trades.

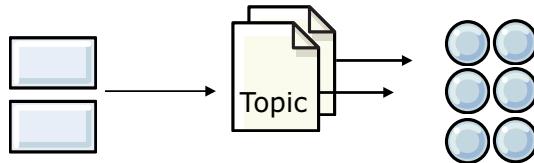
Traders register their interest in (subscribe to) particular stocks and receive updates as prices change. Traders can be added and removed without disruption to the providers of the information who have no knowledge of who is receiving their information.

The terms "push" and "pull" are also becoming increasingly popular when one describes the flow of information between applications. If you concentrate on this example, traders receive new information from the stock-feed as soon as a stock price changes. In this sense, the information can be thought of as being pushed directly to them. This pushing of information from provider to consumer is one of the major differentiators between publish/subscribe and more conventional systems. The stock market example might also be designed so that updated stock prices only flowed to the traders when they are requested, or pulled from a central repository (server) of all stock prices. In such a system, the emphasis would instead be on the traders to request a refresh of their stock prices on a continual basis. IBM MQ supports both modes of operation.

## Publish/subscribe patterns

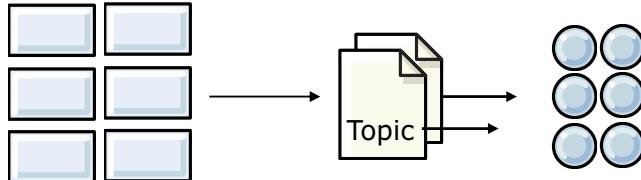
### Few-to-many:

- Research
- News tickers



### Many-to-many:

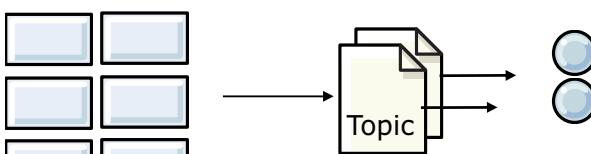
- Prices
- Quotations



### Many-to-few:

- Orders

Key



[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-4. Publish/subscribe patterns

Publishers of information are unaware of subscribers if they publish information even if no subscribing applications require it. Publishing and subscribing are dynamic processes. New subscribers and new publishers can be added to the system without disruption.

There are different types of publish/subscribe patterns, as shown in the figure: few-to-many, many-to-many, and many-to-few.

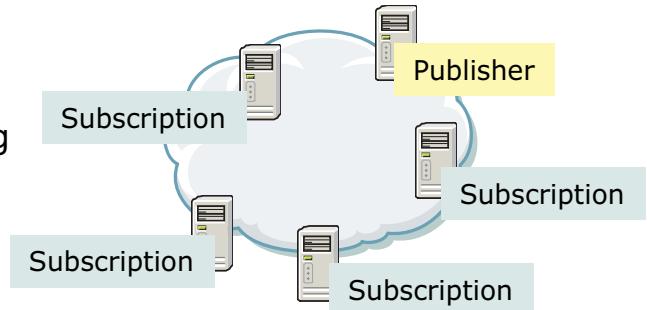
Regarding a topic, or piece of information, all combinations of publishers/subscribers are possible, that is:

- A single or multiple publishing applications can provide information about each topic
- One or more subscribing applications can receive and process the information

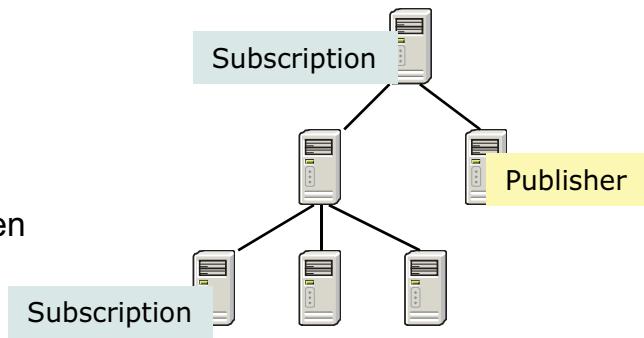
The number of publishers and subscribers to which a single topic connects depends upon the type of information that is flowing between them.

## Distributed publish/subscribe topologies

- Publish/subscribe clusters
  - Many-to-many connectivity
  - Direct routing or topic host routing many-to-many connectivity
  - Based on IBM MQ queue manager cluster



- Hierarchies
  - Indirect many-to-many connectivity
  - Direct one-to-many connectivity
  - Parent-child relationship between queue managers



*Figure 7-5. Distributed publish/subscribe topologies*

Queue managers can communicate with other queue managers in your IBM MQ publish/subscribe system so that subscribers can subscribe to one queue manager and receive messages that were initially published to another queue manager.

The figure shows two main topologies for managing distributed publish/subscribe. The first topology is based on IBM MQ clusters. The second topology uses the more traditional publish/subscribe hierarchies.

## Publications, subscriptions, and topics

- Subscribers make subscriptions with the queue manager to register their interest in information for specific topics
  - Use MQSUB
  - Create subscriptions on behalf of a third party with `DEFINE SUB`
  - See who is subscribing to topics with `DISPLAY SBSTATUS`
- Publishers provide information about specific topics by sending publications to the queue manager by using MQPUT
- Topic names provide an administrative control point for the topic tree
  - Configuration attributes
  - Security profiles
  - Topic tree isolation

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-6. Publications, subscriptions, and topics

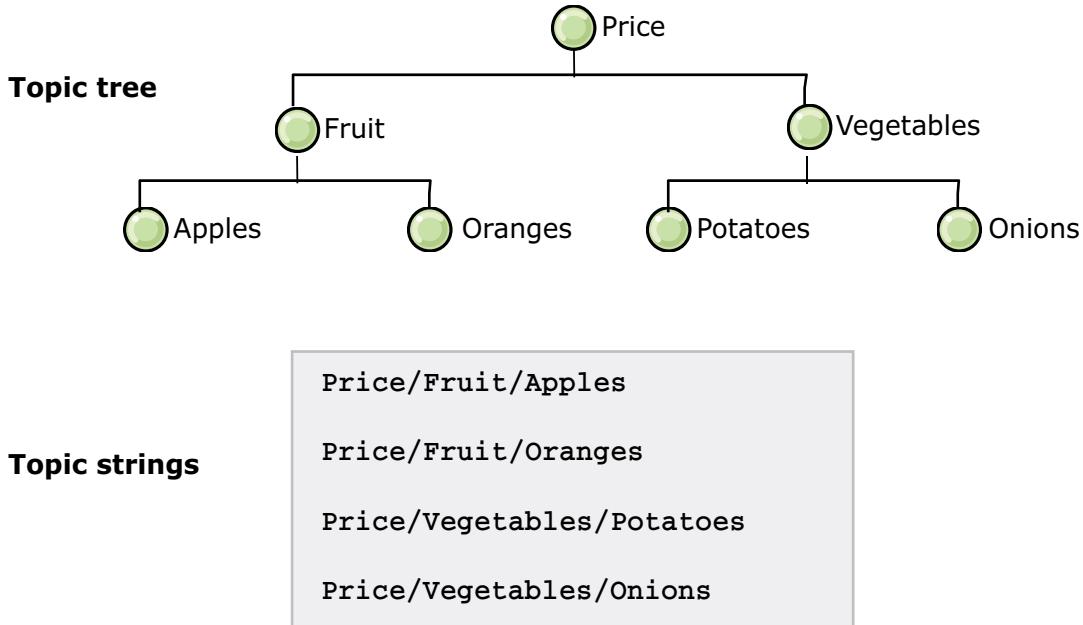
Publish/subscribe is a messaging pattern where publishers, do not program the messages to be sent directly to specific subscribers. Instead, published messages are characterized into topics without knowledge of what, if any, subscribers there might be.

Similarly, subscribers express interest in one or more classes, and receive messages that are of interest only, without knowledge of what, if any, publishers there are.

A subscriber application specifies the topic that it is interested in receiving information about by specifying it on the MQSUB call. A subscriber can make multiple subscriptions to the queue manager.

A publisher publishes its information by putting a message to a *topic*. It is the job of the queue manager, or queue manager network if multiple queue managers are connected, to ensure that all subscribing applications with matching subscriptions to the topic receive the publisher message.

## Topic strings and topic tree



[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-7. Topic strings and topic tree

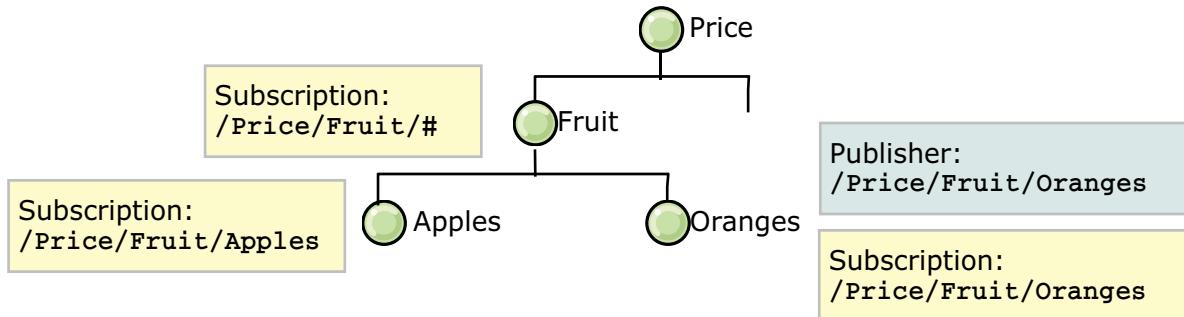
A *topic string* is divided into parts as delimited by the '/' character. Each part is represented as a node in IBM MQ topic tree. As IBM MQ becomes aware of new topic strings, topics are automatically created in the topic tree.

Topic strings can be any characters that you choose. Add structure to your topic strings by using the '/' character, which creates a topic tree with a hierarchical structure, as the example on this figure shows. Avoid some special characters in your topic strings. These characters have special significance: #, +, \*, and ?.

Each topic that you define is an element, or topic, in the topic tree. The topic tree can be empty to start or it can contain topics that were previously defined by using IBM MQ commands. You can define a new topic either by using the create topic commands or by specifying the topic for the first time in a publication or subscription.

You do not need to define any topic objects to use publish/subscribe with IBM MQ. You might want to define topic objects if you need to configure the topic tree to use nondefault attributes. For example, if you want to apply different security profiles to parts of your topic tree, or if you want to isolate your applications from administrative changes to the topic tree you would need topic objects.

## Matching publications to subscriptions



- Subscriptions are attached to matching nodes in the topic tree
- Publications identify the relevant topic
- A copy of the publication is delivered to the queue identified by each matching subscription
- Subscriptions with wildcards at the topic level can receive messages from multiple topic strings

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-8. Matching publications to subscriptions

IBM MQ publish/subscribe is a subject-based publish/subscribe system. A publisher creates a message, and publishes it with a topic string. To receive publications, a subscriber creates a subscription with a pattern matching topic string to select publication topics. The queue manager delivers publications to subscribers with subscriptions that match the publication topic, and are authorized to receive the publications.

## Summary of IBM MQ publish/subscribe

- Decouples publishers from subscribers
  - Queue manager holds a view of all the topic strings in a hierarchical construct that is known as the topic tree
  - MQI applications can programmatically interface with the topic as a subscriber by using MQSUB and as a publisher by using MQOPEN and MQPUT
- Uses MQSC and IBM MQ Explorer to:
  - Enable publish/subscribe in the queue manager
  - Define queue manager publish/subscribe properties
  - Define and monitor topics and subscriptions
- Supports distributed publish/subscribe topologies

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-9. Summary of IBM MQ publish/subscribe*

IBM MQ publish/subscribe support decouples publishers from subscribers.

You can use MQSC and IBM MQ Explorer to define and monitor topics and subscriptions.

IBM MQ supports clustered and hierarchical topologies for distributed publish/subscribe.

## IBM MQ publish/subscribe terminology

- Publisher
  - Application that puts messages to a topic
  - Can open a topic object or a topic string
- Publication: A message that is put to a topic
- Subscription
  - Artifact on a queue manager that describes destinations for copies of messages that are published for a particular topic string
  - Identifies output queue for subscription messages
- Subscriber: An application that uses messages from a subscription

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-10. IBM MQ publish/subscribe terminology*

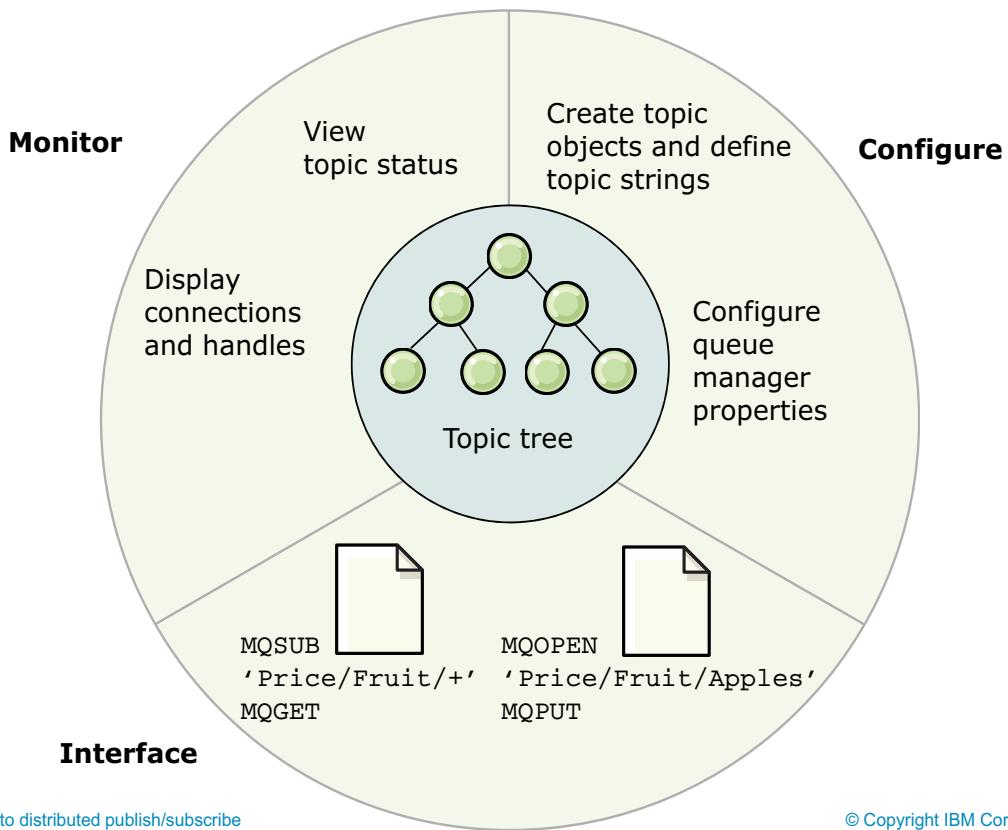
In IBM MQ, a publisher is an application that puts publication messages to a topic queue.

A subscription is an artifact on a queue manager that describes the destinations for copies of publication messages.

A subscriber is an application that gets messages from a subscription.

# IBM Training

## Publish/subscribe in IBM MQ



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-11. Publish/subscribe in IBM MQ

In the IBM MQ publish/subscribe model, the only thing that connects publishing and subscribing applications is the topic that the publisher associates with the information. Publishers and subscribers need to agree on the topic to connect to one another.

Published information is sent in an IBM MQ message, and the topic identifies the subject of the information. The publisher specifies the topic when it publishes the information, and the subscriber specifies the topics about which it wants to receive publications. The subscriber is sent information about only those topics it subscribes to.

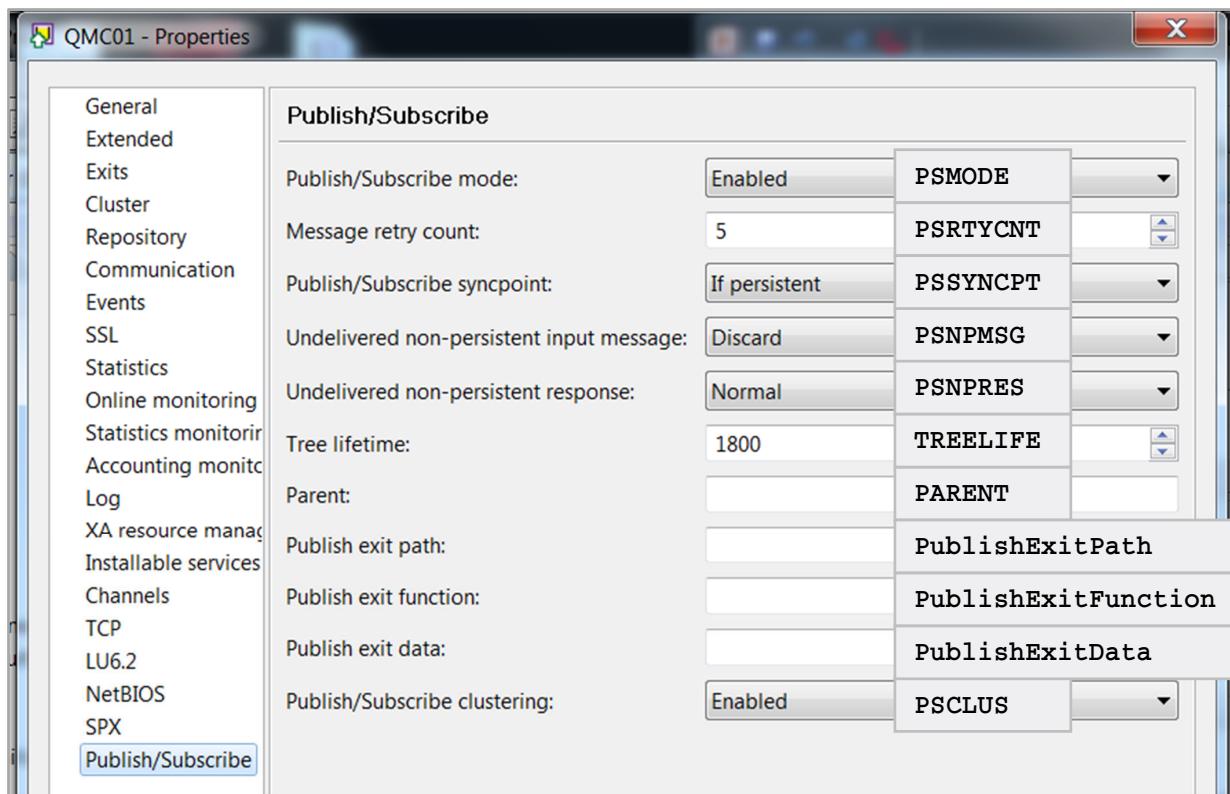
The queue manager holds a view of all the topic strings that are in a topic tree. The topic tree is the central control point for all publish/subscribe. You can configure the behavior of the topic tree by defining topic objects and changing attributes on them.

You can programmatically interface with the topic tree as a subscriber by using MQSUB and as a publisher by using MQOPEN and MQPUT.

You can monitor the use of your topic tree from an application by using the topic status command and the commands to display connections and their handles.



## Publish/subscribe queue manager properties



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-12. Publish/subscribe queue manager properties

Some publish/subscribe properties are configured at the queue manager level. This figure shows the queue manager **Publish/Subscribe** properties in IBM MQ Explorer. This figure also shows the comparable MQSC ALTER QMGR attribute or for the **Publish exit** properties, the queue manager configuration file attributes.

- **Publish/Subscribe mode** is used to enable publish/subscribe. When set to COMPAT, this option also enables publish/subscribe co-existence with previous versions of WebSphere MQ.
- **Message retry count** is the number of times that the channel tries to connect to the remote queue manager before it decides that it cannot deliver the message to the remote queue.
- **Publish/Subscribe sync point** defines whether messages are processed under sync point.
- **Undelivered non-persistent input message** defines what the publish/subscribe engine does with non-persistent input messages that are not delivered.
- **Undelivered non-persistent response** defines what the publish/subscribe engine does with non-persistent responses that are not delivered.
- **Tree lifetime** determines how long the queue manager waits before you remove that node when this nonadministrative node no longer has any active subscriptions.

- **Parent** is the name of the parent queue manager to which the local queue manager is to connect as its child in a hierarchy. Channels must exist in both directions, between the parent queue manager and the child queue manager. This property is described in more detail later in this unit.
- **Publish/subscribe** clustering controls whether this queue manager participates in publish subscribe activity across any clusters in which it is a member. This property is described in more detail later in this unit.

## 7.2. Managing topics

## Defining the topic tree

- Make it extendable
- Avoid excessive wide or deep trees
- Use structure where appropriate
- Limit the topic tree to valid subscription content
- Avoid a rapidly changing set of topics

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-13. Defining the topic tree

There is no standard way to design a topic tree. The tree design depends on the specific data.

A well-structured topic hierarchy should be intuitive to the user, but also consider the later design considerations. The general recommendation is to describe the high level, broad information, then break down this information into finer detail.

Build expandability into the design. Consider the use of high-level topics such as '/global' and '/local' to make their scope clear.

It is possible to combine a topic hierarchy with message properties and selection strings to best fit the usage pattern. For example, do not encode the price into the topic for the rare occasion where a subscriber wants to know when it reaches £100, add it as a *message property* and allow a selector.

- Selectors can affect performance
- Selectors are not flowed around a multiple queue manager publish/subscribe topology
- Selector parsing is only done on the queue managers where the subscriptions exist

## Why worry about the size and shape of the topic tree?

- To simplify administration
- IBM MQ can efficiently scale to tens or hundreds of thousands of topics but certain topic designs can cause problems
  - Broad topic trees put stress on certain topics
  - Deep topic trees add unnecessary topics and levels
  - Many topics can put a strain on memory resources of the system

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-14. Why worry about the size and shape of the topic tree?*

The size and shape of the topic tree can make administration harder or easier. The fewer topic objects and topics there are, the less there is to manage. The more topics, the more memory that is used by the queue manager.

IBM MQ can efficiently scale to tens or hundreds of thousands of topics but certain topic designs can cause problems and affect the user.

When an application opens a topic string, the topic string is looked up in a hash table. The more topic strings, the bigger the hash table. If the topic string is not found, the topic string is stripped down to its '/' delimited parts and every new topic that is referenced in the topic string is dynamically created. Each new node is linked to its parent.

For every topic that is created, that branch of the topic tree is walked backwards to discover the node's configuration and the set of wildcard subscriptions that might match it.

When an application publishes a message on a topic string, a decision is made whether to deliver a copy of the message or not. If the subscription has a selector, the message properties are parsed to assess its suitability. For example, if there is a single topic with a thousand subscriptions, all with selectors, each publication results in a thousand checks. However, if there are 100 topics, each with 10 subscriptions, all with selectors, each publication results in 10 checks. Both would result in the same number of publications, the latter with a lot less work.

Periodically the topics are scanned to discover if they are still necessary. An unnecessary topic is a leaf topic with no publishers or subscriptions currently directly attached. An unnecessary topic is also a topic that was not used for a specific time (minimum 30 minutes by default). The TREELIFE queue manager attribute specifies the time.

Unnecessary topics are deleted and unlinked from their parent. This scanning is necessary to keep memory use under control.

## Defining a topic object with MQSC

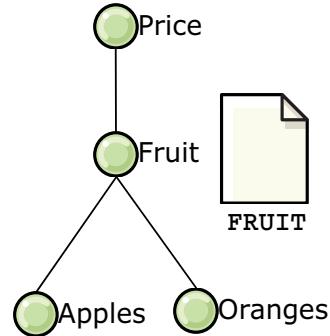
Starting MQSC for queue manager QMC1.

```
DEFINE TOPIC(FRUIT) TOPICSTR('Price/Fruit')
DEFPSIST(YES)
```

```
DISPLAY TOPIC(FRUIT)
```

AMQ8633: Display topic details.

TOPIC(FRUIT)	TYPE(LOCAL)
TOPICSTR(Price/Fruit)	DESCR( )
CLUSTER( )	DURSUB(ASPARENT)
PUB(ASPARENT)	SUB(ASPARENT)
DEFPSIST(ASPARENT)	DEFPRTY(ASPARENT)
DEFPRESP(YES)	ALTDATE(2014-02-26)
ALTTIME(15.05.22)	PMMSGDLV(ASPARENT)
NPMMSGDLV(ASPARENT)	PUBSCOPE(ASPARENT)
SUBSCOPE(ASPARENT)	PROXYSUB(FIRSTUSE)
WILDCARD(PASSTHRU)	MDURMDL( )
MNDURMDL( )	



[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-15. Defining a topic object with MQSC

The topic object can be defined by using IBM MQ Explorer or MQSC. The example shows how to define the topic with the DEFINE TOPIC command and then display the topic attributes with the DISPLAY TOPIC command. The TOPIC object type has DEFINE, ALTER, DELETE, and DISPLAY commands.



### Note

You cannot alter the object name (TOPICSTR) attribute with the ALTER command. You must delete and then redefine an object if the topic name changes.

## Topic object attributes

- TOPICSTR identifies path
  - Do not put the root node, indicated by a forward slash (/), into a cluster
  - Split the tree into /global and /local
- PUB and SUB control whether messages can be published or subscriptions can be made on the topic
- PUBSCOPE and SUBSCOPE determine whether this queue manager propagates publications to other queue managers as part of a hierarchy or as part of a publish/subscribe cluster
- PROXYSUB controls whether a proxy subscription can be sent for this topic to directly connected queue managers, even if no local subscriptions exist
- When an attribute has the value ASPARENT, the value is taken from the setting of the first parent administrative node that is found in the topic tree

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

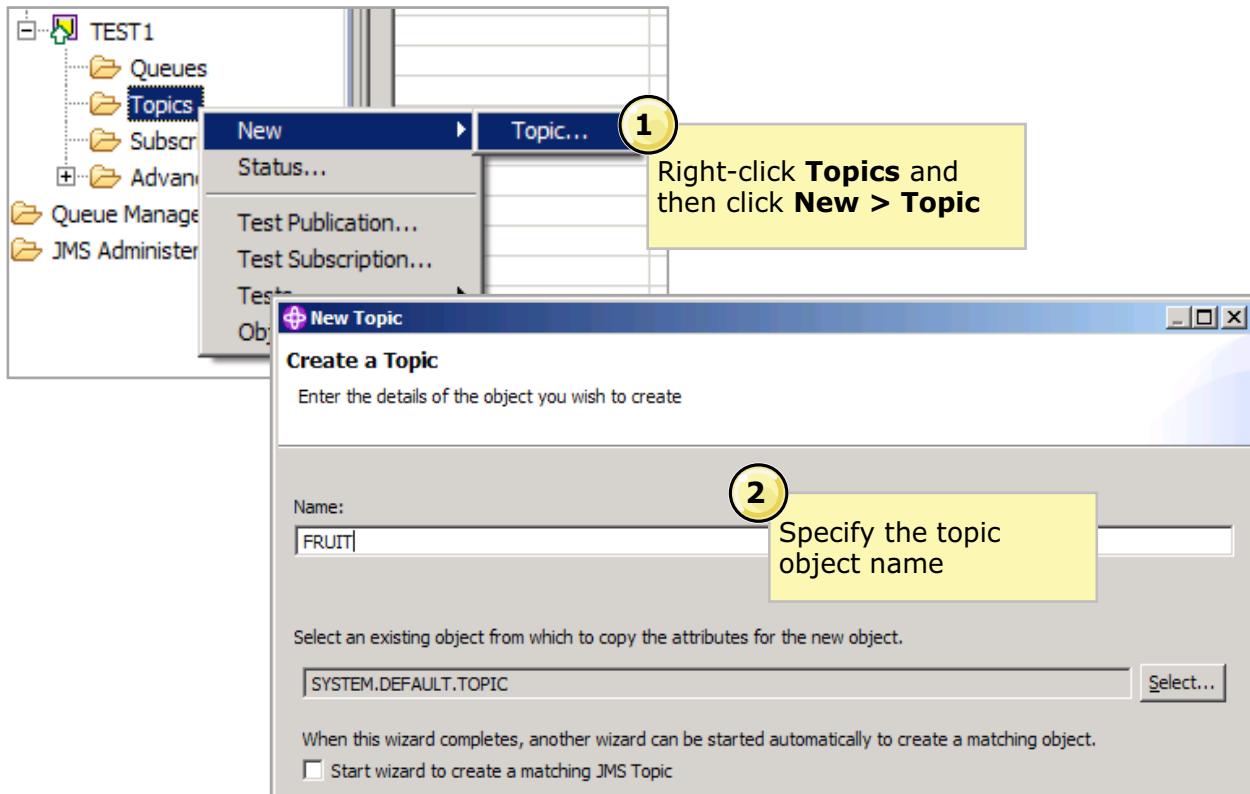
Figure 7-16. Topic object attributes

You can use the DISPLAY TOPIC command to display the topic attributes. More properties that are not described in the figure are ASPARENT, DURSUB, MDURMDL, and MNDURML.

- The ASPARENT value is based on the setting of the closest parent administrative topic object in the topic tree. Each topic can inherit attributes from the topic above it. Topic objects that use the ASPARENT value inherit their values from the topic above it. If a particular topic has no parent topic object, or those parent objects also have ASPARENT, any remaining ASPARENT attributes are inherited from the SYSTEM.BASE.TOPIC object.
- The DURSUB attribute specifies whether applications can make durable subscriptions on this topic. Durable subscriptions are described in more detail later in this unit.
- The MDURMDL and MNDURMDL attributes specify the name of the model queue that durable (and nondurable) subscriptions use to request that the queue manager manages the destination of its publications.



## Defining a topic object in IBM MQ Explorer (1 of 2)



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

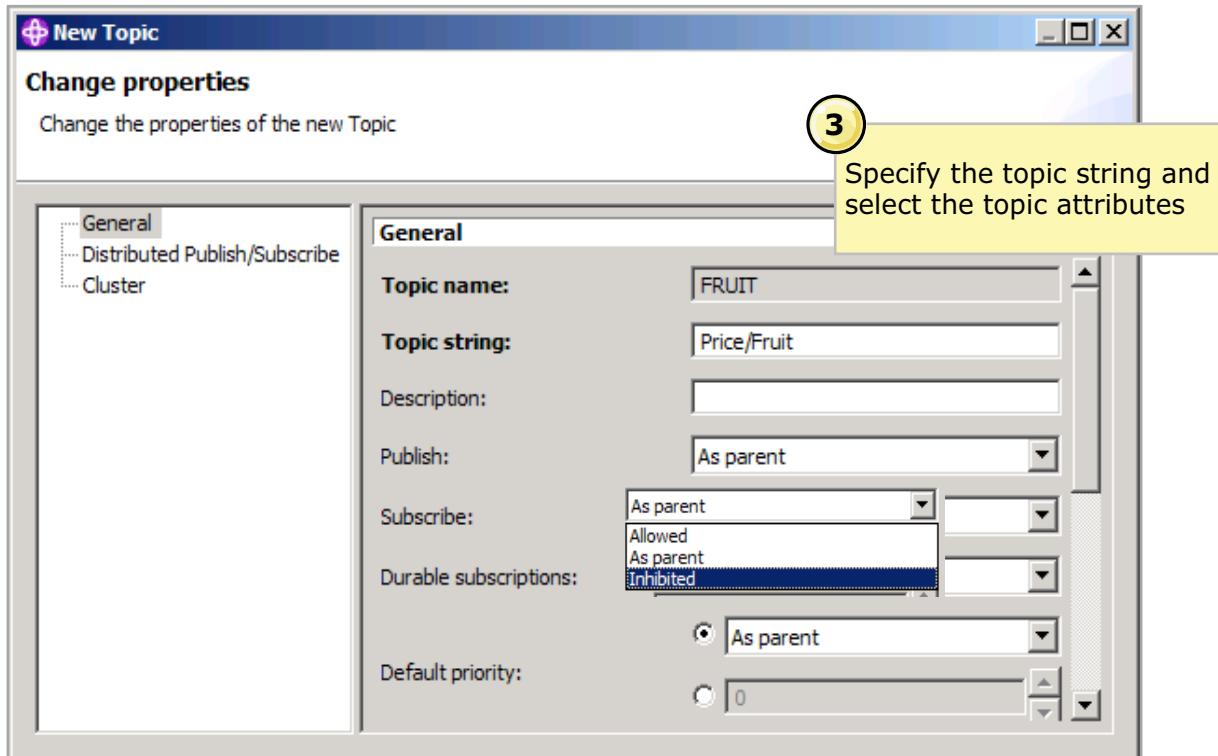
Figure 7-17. Defining a topic object in IBM MQ Explorer (1 of 2)

A topic can be defined in IBM MQ Explorer.

To define a topic in IBM MQ Explorer:

1. In the IBM MQ Explorer Navigator under the queue manager, right-click **Topics** and then click **New > Topic**.
2. Specify a topic object name.

## Defining a topic object in IBM MQ Explorer (2 of 2)



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-18. Defining a topic object in IBM MQ Explorer (2 of 2)

3. On the **General** properties, specify the topic string, select the topic attributes, and then click **Finish**.

In addition to the topic name, topic string, and description, you can also set other topic attributes.

- The **Publish** attribute controls whether messages can be published to the topic.
- The **Subscribe** attribute controls whether messages can subscribe to the topic.
- The **Durable subscriptions** attribute controls whether the topic allows durable subscriptions.
- The **Default priority** attribute is the default priority of messages that are published to the topic.
- The **Default persistence** attribute is the persistence of a new topic.

## Managing topic object definitions

- **DISPLAY TOPIC(\*) TOPICSTR**
  - Show all locally defined topic objects on this queue manager, which includes the topic string that they are associated with
- **DISPLAY TOPIC(*TopicObject*)**
  - Show all configured attributes of the topic object
  - Identify attributes that are inherited (**ASPARENT**)
- **DISPLAY TOPIC(\*) TYPE(CLUSTER) TOPICSTR**
  - When using clustered topics, show all the clustered topic objects that are known by this queue manager

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

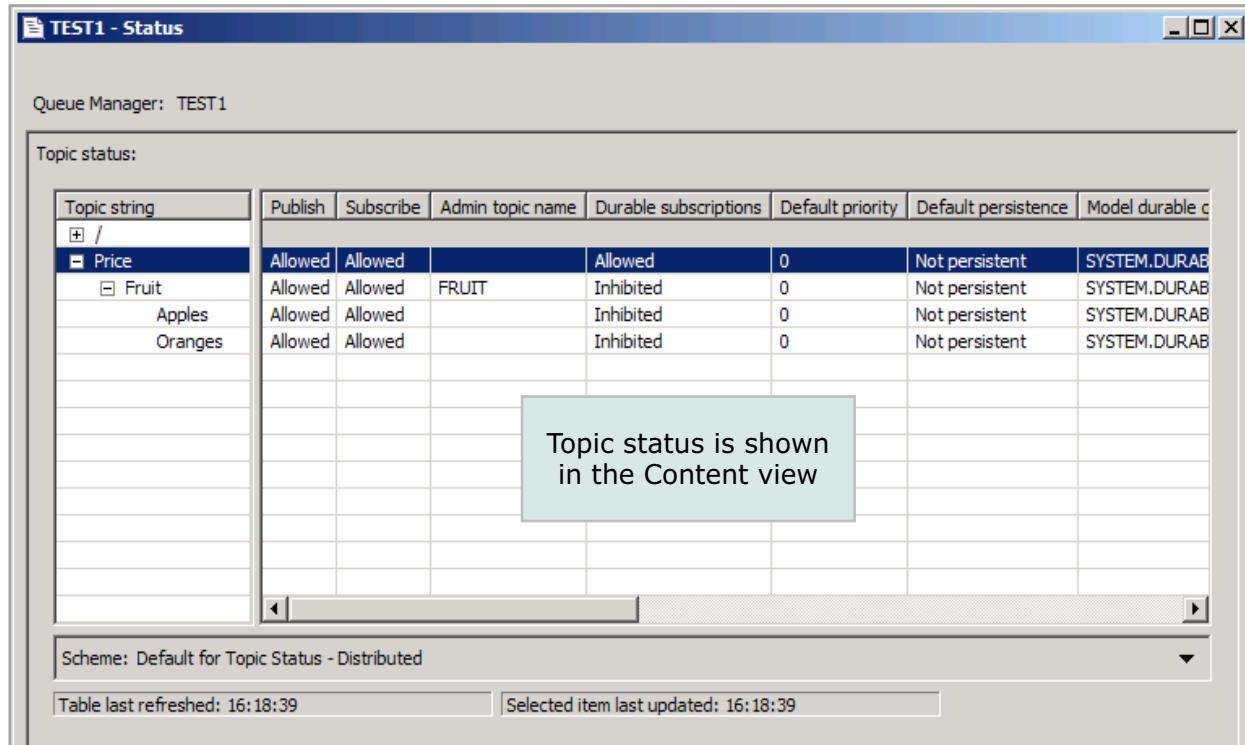
Figure 7-19. Managing topic object definitions

This figure lists some other commands that you can use to manage topic definitions.

Clustered topics are described in detail later in this unit.

IBM Training

## Getting topic status in IBM MQ Explorer



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-20. Getting topic status in IBM MQ Explorer

To display the topic status in IBM MQ Explorer, right-click the queue manager **Topics** folder in the navigator and then click **Status**.

The topic status is shown in the IBM MQ Explorer **Content** view. The topic status shows the properties of each topic and the relationship between topics and subtopics.

## Getting topic status with MQSC

- **DISPLAY PUBSUB ALL**
  - Returns TPCOUNT, which indicates the total number of topics currently in the topic tree of this queue manager
  - Can be used to see whether the tree is continually growing and possibly introducing a memory usage risk
- **DISPLAY TPSTATUS ('#')**
  - Returns an entry for every node in the topic tree
- **DISPLAY TPSTATUS ('#') TOPICSTR WHERE(SUBCOUNT GT 0)**
  - Returns those topics with one or more registered subscriptions
- **DISPLAY TPSTATUS ('*TopicString*')**
  - Shows an individual topic and all resolved attributes of that node
  - Returns status for this topic, such as the number of registered subscriptions (SUBCOUNT) and currently connected publishers (PUBCOUNT)

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-21. Getting topic status with MQSC

This figure lists some other commands that can be used to monitor topic status.

DISPLAY PUBSUB ALL returns TPCOUNT, gives an idea of the total number of topics currently in the topic tree of this queue manager. This command can be used to see whether the tree is continually growing (introducing a memory usage risk) or stable. If it is growing, it might indicate a problem in the topic tree design, such as too many unique topic strings. Investigate the TREELIFE attribute of the queue manager.

DISPLAY TPSTATUS('#') returns an entry for every topic in the topic tree. The use of quoted topic string wildcard notation ('#') is not the standard MQSC wildcard character.

DISPLAY TPSTATUS('#') TOPICSTR WHERE(SUBCOUNT GT 0) returns those topics with one or more subscriptions that are registered against it.

DISPLAY TPSTATUS('TopicString') displays an individual topic with its topic string value. This command returns all resolved attributes of that node and replaces any inheritable values with what they resolve to in the topic tree (you do not see ASPARENT in this output). It also returns the status for this topic, such as the number of registered subscriptions (SUBCOUNT) and currently connected publishers (PUBCOUNT).

## Display topic status example

```

Starting MQSC for queue manager TEST1.

DIS TPSTATUS('Price/Fruit/+') TYPE(PUB) all

AMQ8754: Display topic status details.
    TOPICSTR(Price/Fruit/Oranges)          LPUBDATE(2014-02-26)
    LPUBTIME(16:50:44)
    ACTCONN(414D51435445535431202020202020832AC44720005E02)
    NUMPUBS (3)
AMQ8754: Display topic status details.
    TOPICSTR(Price/Fruit/Apples)          LPUBDATE(2014-02-26)
    LPUBTIME(16:50:37)
    ACTCONN(414D5143544553543120202020202020832AC44720007601)
    NUMPUBS (1)

```

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

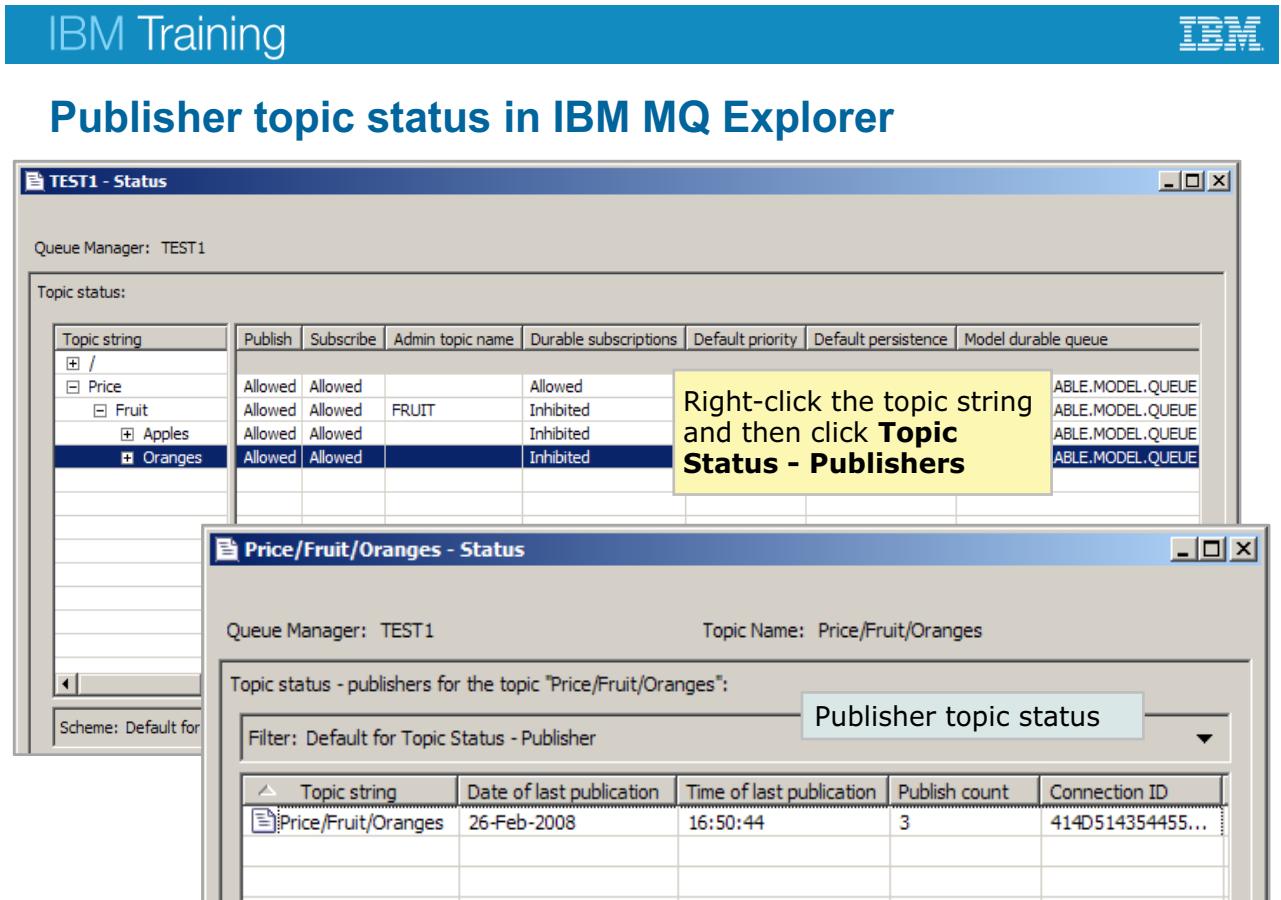
*Figure 7-22. Display topic status example*

The DISPLAY TPSTATUS command by default assumes a TYPE(TOPIC). If TYPE(PUB) is specified, then the command displays status information that relates to applications that have topics open for publish.

With DISPLAY TPSTATUS TYPE(PUB) command, you can see the details of the current publishers on the topic string.

The parameters of interest are:

- **ACTCONN** is the active connection ID (**CONNID**) that is associated with the handle that has this topic open for publish.
- **NUMPUBS** is the number times this publisher published. This value records the actual number of publishes, not the total number of messages that are published to all subscribers.



Introduction to distributed publish/subscribe

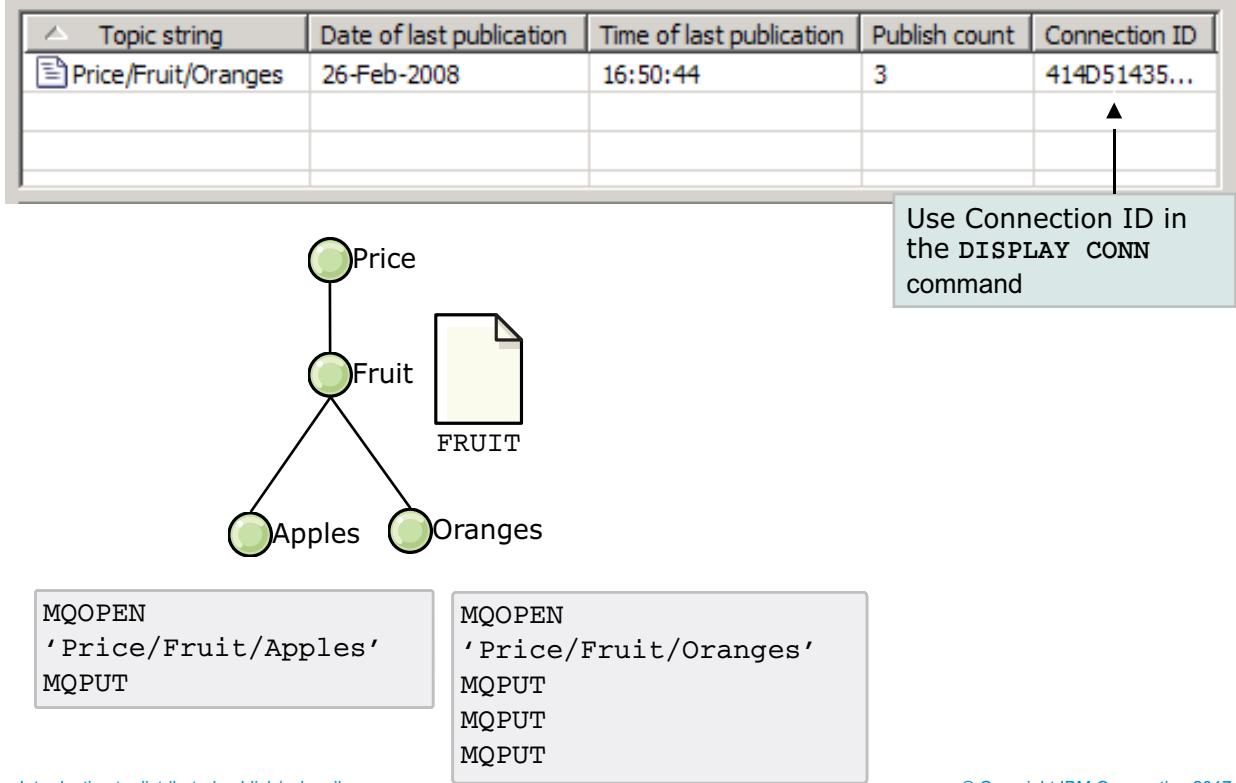
© Copyright IBM Corporation 2017

Figure 7-23. Publisher topic status in IBM MQ Explorer

The publisher topic status contains the time and date of last publication, publish count, and connection ID.

To display the publisher topic status, right-click the topic string and then click **Topic Status - Publishers**.

## Finding publishers (1 of 2)



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-24. Finding publishers (1 of 2)

The topic status of the publisher contains information about the topic such as the time and date of the last publication, the publish count, and the connection ID.

The topic status of the publisher can be accessed in IBM MQ Explorer or from a command.

The example shows the publisher topic status for the topic 'Price/Fruit/Oranges'.

## Finding publishers (2 of 2)

```

DIS TPSTATUS('Price/Fruit/+') TYPE(PUB) all
AMQ8754: Display topic status details.
    TOPICSTR(Price/Fruit/Oranges)          LPUBDATE(2014-02-26)
    LPUBTIME(16:50:44)
    ACTCONN(414D51435445535431202020202020832AC44720005E02)
    NUMPUBS(3)

AMQ8754: Display topic status details.
    TOPICSTR(Price/Fruit/Apples)          LPUBDATE(2014-02-26)
    LPUBTIME(16:50:37)
    ACTCONN(414D5143544553543120202020202020832AC44720007601)
    NUMPUBS(1)

DIS CONN(832AC44720007601) TYPE(ALL)
AMQ8276: Display Connection details.
    CONN(832AC44720007601)
    EXTCONN(414D5143544553543120202020202020)
    TYPE(CONN)
    APPLTAG(D:\q.exe)                  APPLTYPE(USER)
    USERID(jsmith)
    OBJNAME( )                         OBJTYPE(TOPIC)
    OPENOPTS(MQOO_OUTPUT,MQOO_FAIL_IF_QUIESCING)
    TOPICSTR(Price/Fruit/Apples)

```

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-25. Finding publishers (2 of 2)

As shown in the figure, the ACTCONN attribute contains the currently active connection ID associated with the handle that has this topic open for publish.

Further connection details can be displayed by using the DISPLAY CONN command with the handle that is specified in the ACTCONN attribute of the topic status details.

## Retained publications

- When a message is published to a topic string, it is delivered to each matching subscription that is registered
  - Subscriptions that are created after that point do not receive the message, only newly published ones
- A publisher can specify to retain a message so that it can be sent to future subscribers
- Only one publication can be retained for each topic string, so the existing retained publication is deleted when a new retained publication arrives
- When a subscription is created, any matching retained message is delivered to it

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-26. Retained publications*

By default, after a publication is sent to all interested subscribers it is discarded. However, a publisher can specify that a copy of a publication is retained so that it can be sent to future subscribers who register an interest in the topic. Retained publications can be useful if information on a topic is infrequently published, and a new subscriber needs to see the most recent update as soon as they are created.

The queue manager stores retained publication messages on a queue. The queue manager can retain only one publication for each topic, so the existing retained publication of a topic is deleted when a new retained publication arrives at the queue manager.

The same recommendation of not building a deep application queue applies equally here. Calculate the total number of topic strings where publications would be retained when you consider the use of retained publications.

If the application connects intermittently, or if regular updates are not required, the application can choose to get retained publications only when requested (MQI only).

## 7.3. Managing subscriptions

## Subscription management

- Application creates and deletes subscriptions

Examples:

- MQI uses MQSUB to create and MQCLOSE to delete
- JMS uses `TopicSession.createDurableSubscriber()` and `TopicSession.unsubscribe()`

- IBM MQ administrator creates and deletes subscriptions

MQSC examples:

```
DEFINE SUB('SUB1') DEST(Q1) TOPICSTR('/Price/Fruit/Apples')
DELETE SUB('SUB1')
```

- Applications can either use the publish/subscribe APIs to access subscriptions or access the subscription queue by using point-to-point APIs

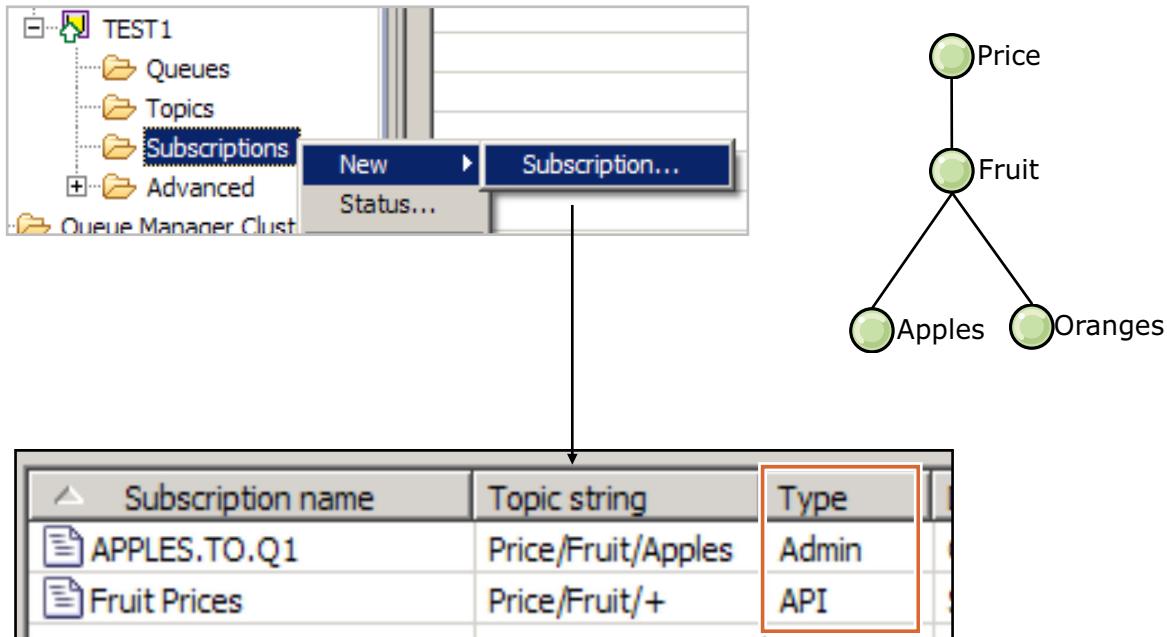
*Figure 7-27. Subscription management*

There are two ways to create and manage subscriptions:

- An application can use an API to create and delete subscriptions
- An IBM MQ administrator can use MQSC or IBM MQ Explorer to create and delete subscriptions

# IBM Training

## Creating administrative subscriptions in IBM MQ Explorer



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-28. Creating administrative subscriptions in IBM MQ Explorer

You do not have to create subscriptions by coding applications to use MQSUB, you can create them administratively.

The figure shows the properties of a subscription in IBM MQ Explorer. The **Type** of subscription indicates how the subscription was created. The subscription types are:

- **API** indicates that the subscription was created with an MQSUB API request.
- **Admin** indicates that the subscription was created with a DEFINE SUB command or in IBM MQ Explorer. **Admin** is also used to indicate that a subscription was modified with an administrative command.
- **Proxy** indicates that the subscription that is created internally for routing publications through a queue manager network.

## Subscription lifetime

- Durable subscriptions
  - The lifetime of the subscription is independent of any application
  - Explicit creation and deletion of the subscription is required
  - Every durable subscription must be uniquely named within a queue manager
  - If no applications read from the queue, depth of the queue that is named in the durable subscription continues to increase
  - Subscriptions can be set to expire
- Non-durable subscriptions
  - The creating application bounds the lifetime of the subscription
  - Subscriptions are automatically deleted when the application closes the subscription or disconnects
  - Application can create, administrator cannot

	Durable	Non-durable
Administrator creates	Yes	No
Application creates	Yes	Yes

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-29. Subscription lifetime

Subscriptions can be durable or nondurable. Durable subscriptions can receive messages that are sent while the subscribers are not active. Durable subscriptions provide the flexibility and reliability of queues but still allow clients to send messages to many recipients.

A durable subscription is a subscription that an application or administrator creates that is independent of any application.

An application can create a nondurable subscription that is automatically deleted when the application closes. An administrator cannot create a nondurable subscription.

A *durable subscription* saves messages for an inactive subscriber and delivers the saved messages when the subscriber reconnects. In this way, a subscriber does not lose any messages even though it disconnected. If you need to disallow the creation of durable subscriptions for one half of the topic tree, you can create one `TOPIC` object at the highest point where you need this behavior to start. The nodes in the topic tree below that point inherit the behavior without the need for any further `TOPIC` object definitions.

## Resolving ASARENT for durable subscriptions

**Topic status:**

Topic string	Allowed	Allowed		Allowed
Price	Allowed	Allowed	FRUIT	Inhibited
Fruit	Allowed	Allowed		Inhibited
Apples	Allowed	Allowed		Inhibited
Oranges	Allowed	Allowed		Inhibited

**IBM MQ Explorer**

**MQSC**

```
DISPLAY TPSTATUS('Price/Fruit')
AMQ8754: Display topic status details.
  TOPICSTR(Price/Fruit)          ADMIN(FRUIT)
  MDURMDL(SYSTEM.DURABLE.MODEL.QUEUE)
  MNDURMDL(SYSTEM.NDURABLE.MODEL.QUEUE)
  DEFPSIST(NO)                  DEFPRTY(0)
  DEFPRESP(SYNC)                DURSUB(NO)
  PUB(ENABLED)                   SUB(ENABLED)
  PMSGDLV(ALLDU)
  RETAINED(NO)
  SUBCOUNT(0)
  SUBSCOPE(ALL)
```

```

graph TD
    Price((Price)) --- Fruit((Fruit))
    Fruit --- Apples((Apples))
    Fruit --- Oranges((Oranges))
    FRUIT[FRUIT]
  
```

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-30. Resolving ASARENT for durable subscriptions

Topic status can be accessed in IBM MQ Explorer or with the DISPLAY TPSTATUS command.

The DISPLAY TPSTATUS command displays the status of one or more topics in a topic tree. It also resolves the ASARENT attribute to the inherited value. For example, because durable subscriptions are inhibited in the 'Fruit' topic, the subtopics of 'Apples' and 'Oranges' also have durable subscriptions inhibited.

As shown in the figure, the DISPLAY TPSTATUS command requires a topic string value to determine which topics the command returns.

## Managing subscription queues

- Subscription maps a topic to a queue
- Queue relationship is either explicit or implicit
- Managed subscription queue
  - Creation of the subscription automatically creates and deletes a queue for the use of queuing any matching publications
  - Deleting the subscription automatically deletes the managed queue
- Unmanaged subscription queue
  - Administrator must provide name and location of an existing queue when the subscription is created
  - Queue is not deleted when the subscription is deleted
  - Multiple subscriptions can point to the same unmanaged queue
  - Unmanaged subscriptions can be created administratively or created programmatically by using the MQI interface

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-31. Managing subscription queues*

Every subscription is mapped to a subscriber queue. When the local queue manager receives a publication, it scans the information to determine whether a subscription matches the publication's topic and selection string. For each matching subscription, the queue manager directs the publication to the subscriber's subscriber queue.

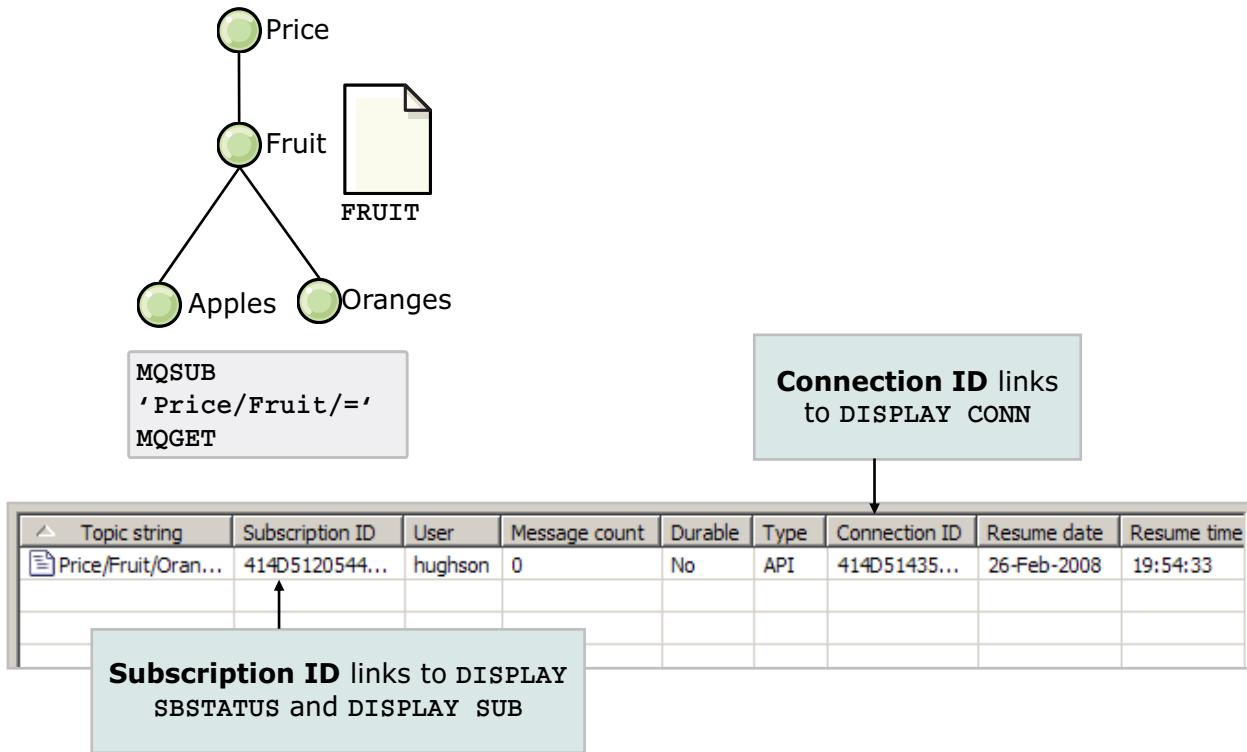
When you create a subscription, you can choose to use managed queuing. If you use managed queuing, a subscription queue is automatically created when you create a subscription. The durability of the subscription determines whether the managed queue remains when the subscribing application's connection to the queue manager is broken.

Using managed queues means that you do not have to worry about creating queues to receive publications. Any unconsumed publications are removed from subscriber queues automatically if a nondurable subscription connection is closed.

With unmanaged queues, it is the administrator's or application's responsibility to create and manage the queues.



## Monitoring application subscriptions in IBM MQ Explorer



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-32. Monitoring application subscriptions in IBM MQ Explorer

You can use IBM MQ Explorer or MQSC to monitor application subscriptions. This figure shows the subscription status view in IBM MQ Explorer for the topic ‘Price/Fruit/Orange’.

To show the subscription status in IBM MQ Explorer, right-click the **Subscriptions** folder under the queue manager in the navigator and then click **Status**.

Subscription properties include the following fields:

- **Subscription ID** is a unique identifier for this subscription that the queue manager assigns.
- **User** is the identifier that is associated with the subscription.
- **Durable** is a subscription parameter. When **Durable** is set to **Yes**, the subscriptions are not deleted when the creating application closes its subscription handle.
- The active **Connection ID** that opened this subscription.
- The number of messages that were put to the destination this subscription specifies since it was created, or since the queue manager was restarted, whichever is more recent.
- The date and time of the most recent MQSUB that connected to this subscription.

## Monitoring subscriptions with MQSC (1 of 2)

- **DISPLAY PUBSUB ALL**
  - Returns SUBCOUNT, which is the number of subscribers for the topic string, including durable subscribers who are not currently connected
- **DISPLAY SUB(\*) TOPICSTR**
  - List all the subscriptions, including the topic string
- **DISPLAY SUB (*SubName*)**
  - Display the configuration of a subscription
- **DISPLAY SBSTATUS (*SubName*)**
  - Display the status of a subscription
  - Includes the last time that the subscription was resumed (opened) and the approximate number of messages that matched this subscription since it was created (or since a queue manager restart, if later)

Figure 7-33. Monitoring subscriptions with MQSC (1 of 2)

This figure and the next list some useful commands for monitoring subscriptions.

## Monitoring subscriptions with MQSC (2 of 2)



## Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

*Figure 7-34. Monitoring subscriptions with MQSC (2 of 2)*

This figure lists more commands that you can use to monitor subscriptions.

## Display subscription example

```
DISPLAY SUB (SUB1)
 1: DISPLAY SUB (SUB1)
AMQ8096: MQ subscription inquired.
  SUBID (414D51205445535431202020202020832AC44720013D07)
  SUB (SUB1)                      TOPICSTR (/Price/Fruit/Apples)
  TOPICOBJ ( )
  DEST (SYSTEM.MANAGED.DURABLE.533D180705230020)
  DESTQMGR (QMC1)                  PUBAPPID ( )
  SELECTOR ( )                     SELTYPE (NONE)
  USERDATA ( )
  PUBACCT (16010515000000DEA960DF65174E4B97C192FFE80300000)
  DESTCORL (414D5120514D4752132020202022007183D532000230)
  DESTCLAS (MANAGED)              DURABLE (YES)
  EXPIRY (UNLIMITED)              PSPROP (MSGPROP)
  PUBPRTY (ASPUB)                 REQONLY (NO)
  SUBSCOPE (ALL)                  SUBLVEL (1)
  SUBTYPE (ALL)                   VARUSER (ANY)
  WSCHEMA (TOPIC)                 SUBUSER (xxxx)
  CRDATE (2014-04-03)             CRTIME (09:19:15)
  ALTDATE (2014-04-03)            ALTTIME (09:19:15)
```

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-35. Display subscription example*

The DISPLAY SUB command shows the more static attributes of the subscription, whether it was created by using a DEFINE SUB command or an MQSUB call in an application.

## Display subscription status example

```
DISPLAY SBSTATUS (SUB1)
2: DISPLAY SBSTATUS (SUB1)
AMQ8099: MQ subscription status inquired
SUB (SUB1)
SUBID (414D51205445535431202020202020832AC44720013D07)
SUBUSER (xxxx)             RESMDATE (2014-02-26)
RESMTIME (19:10:07)        LMSGDATE (2014-02-26)
LMSGTIME (19:10:28)
ACTCONN (414D5143544553543120202020202020832AC44720013D05)
DURABLE (YES)              MCASTREL( , )
NUMMSGS (5)                SUBTYPE (API)
TOPICSTR (/Price/Fruit/Apples)
```

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-36. Display subscription status example

The DISPLAY SBSTATUS command shows the runtime status of a subscription. The figure shows an example of the DISPLAY SBSTATUS command and the subscription status details.

## Display connection status for the subscription

```
DIS CONN(832AC44720013D05) TYPE(ALL)
AMQ8276: Display Connection details
CONN(832AC44720013D05)
EXTCONN(414D51435445535431202020202020)
TYPE(CONN)
APPLTAG(D:\q.exe) APPLTYPE(USER)
USERID(xxxx)
OBJNAME(Q1) OBJTYPE(QUEUE)
OPENOPTS(MQOO_INPUT_SHARED,MQOO_FAIL_IF_QUIESCING)
OBJNAME( ) OBJTYPE(TOPIC)
DEST(Q1) DESTQMGR(QM1)
SUBNAME(Fruit Prices)
SUBID(414D51205445535431202020202020832AC44720016403)
TOPICSTR(Price/Fruit/Apples)
```

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

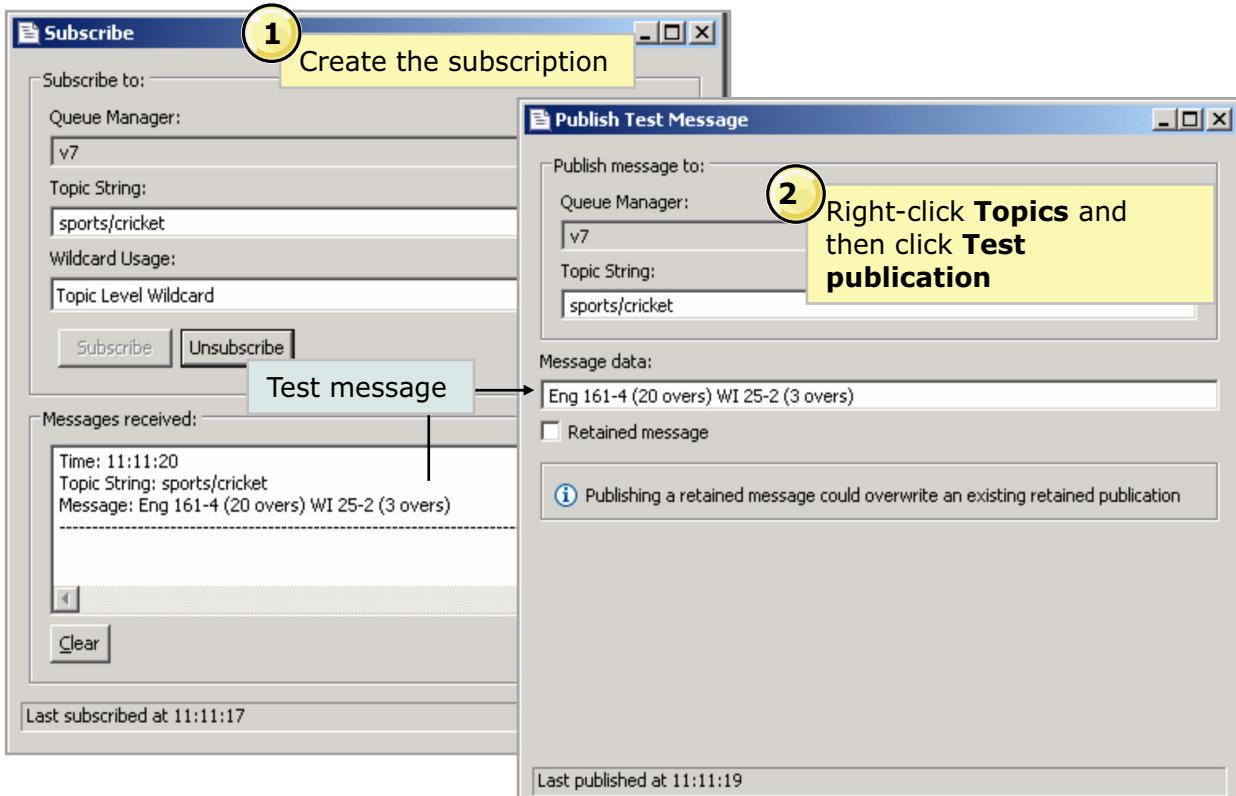
*Figure 7-37. Display connection status for the subscription*

The ACTCONN subscription status parameter contains the connection ID. You can use the connection ID in the DISPLAY CONN command to display information about the active connection and the OPEN options. The figure shows an example of the DISPLAY CONN command and the connection details.

- **OBJNAME** is the name of an object that the connection has open.
- **OBJTYPE** is the type of the object that the connection has open. If this handle is that of a subscription to a topic, then the **SUBID** parameter identifies the subscription. You can then use the **DISPLAY SUB** command to find all the details about the subscription.
- **OPENOPTS** identifies the open options currently in force for the connection for the object.

# IBM Training

## Testing publish/subscribe in IBM MQ Explorer (1 of 2)



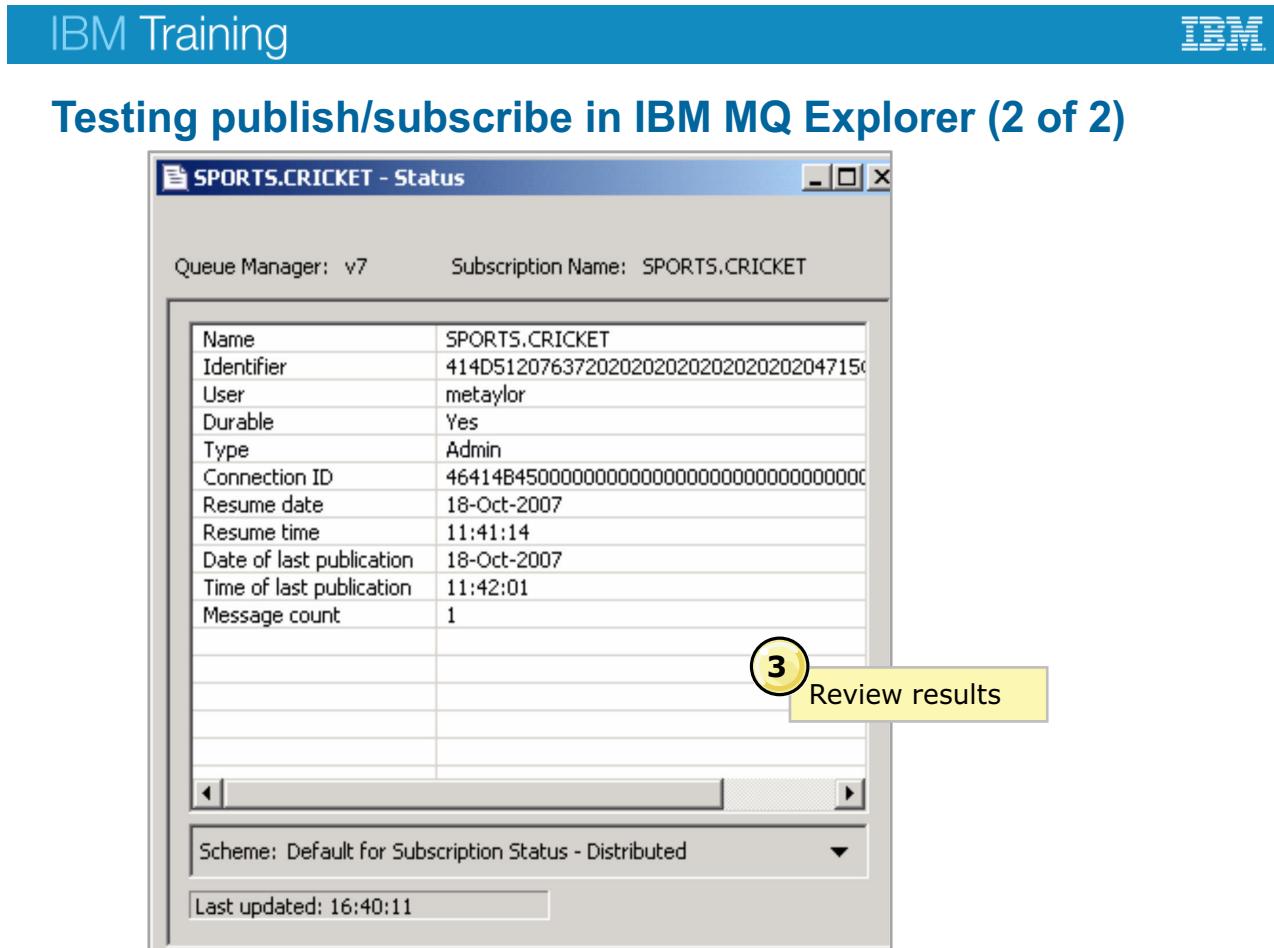
Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-38. Testing publish/subscribe in IBM MQ Explorer (1 of 2)

You can publish and subscribe to test publications (messages) in IBM MQ Explorer to check that your network and topics work as intended.

1. Create the subscription and subscribe to the topic that you want to test by right-clicking the **Topics** folder, and then clicking **Test Subscription**. The Subscribe application opens.
2. Publish a message to the same topic:
  - a. Right-click the **Topics** folder, and then click **Test Publication**. The Publish Test Message application opens.
  - b. In the **Topic String** field, type the name of the topic. You or another publisher can already be registered to publish on the topic, or you can enter a new topic name. When you publish the message, you are automatically registered as a publisher on the topic.
  - c. In the **Message data** field, type a message to send in the publication.
  - d. Click **Publish message** to send the message.



## Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

*Figure 7-39. Testing publish/subscribe in IBM MQ Explorer (2 of 2)*

3. If the subscriber receives the message (the publication), the subscription status is shown for your review.

## 7.4. Security overview

## Publish/subscribe security

- Based on defined topic objects
- Checked from the bottom up for authorization in the topics tree
- When publishing, the MQOPEN checks the topic that you want to publish
- When subscribing, a check is carried out when an MQSUB to the specified topic is received
- Subscriber must have authority to PUT to the destination queue for the subscription (actual ‘publish’ check)

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-40. Publish/subscribe security*

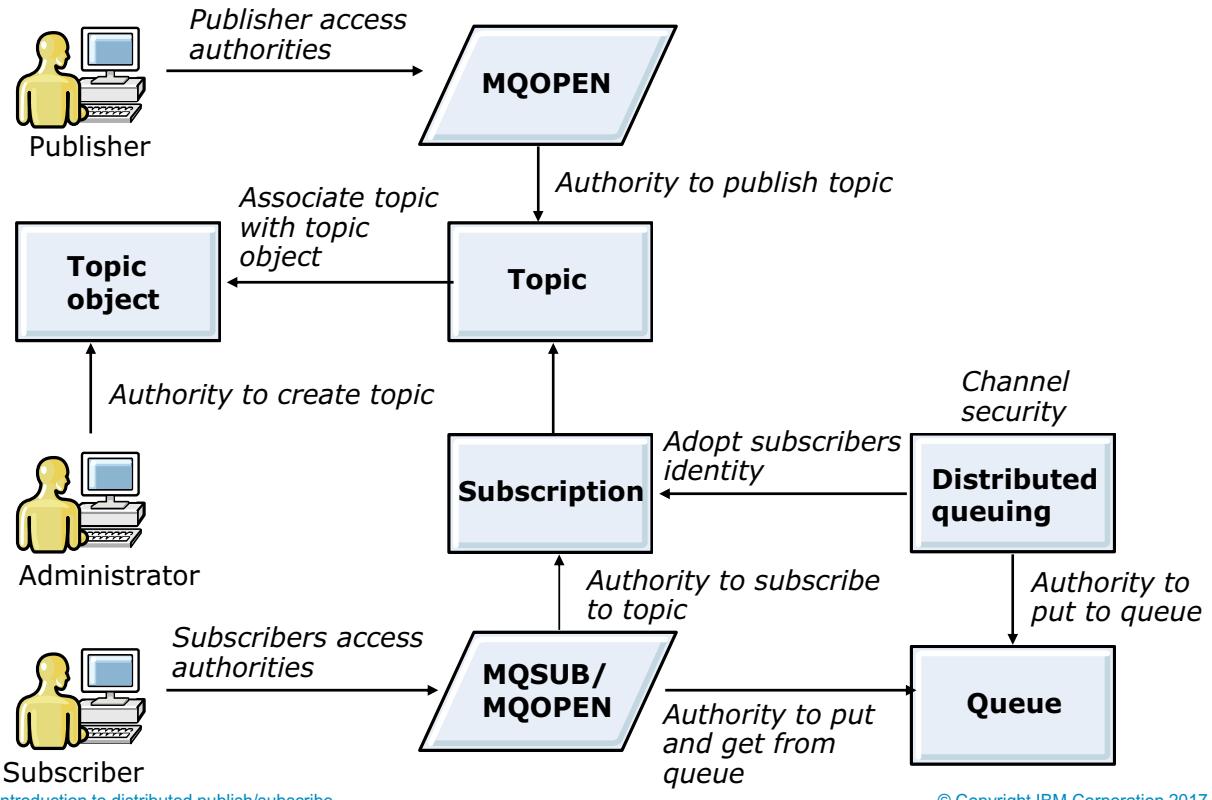
Many components are involved in publishing and subscribing to a topic.

Topic strings identify topics, and are typically organized into trees. You need to associate a topic with a topic object to control access to the topic. The security attributes are associated with the appropriate administration topic in the topic tree. When an authority check is made for a particular user ID during a subscribe or publish operation, the authority that is granted is based on the security attributes of the associated topic.

Your permission to publish or subscribe to a topic is checked on the local queue manager by using local identities and authorizations. Authorization does not depend on whether the topic is defined or not, nor where it is defined. So, you need to authorize topics on every queue manager in a cluster when clustered topics are used.

A subscription contains information about the identity of the subscriber and the identity of the destination queue on which you must place the publications. It also contains information about how the publication is placed on the destination queue. You can also restrict subscriptions to use by an individual subscriber. You can also control what information about the subscriber the queue manager uses when publications are placed on to the destination queue.

## Publish/subscribe security relationships



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-41. Publish/subscribe security relationships

Some of the security relationships between the publish/subscribe components are illustrated in the figure.

The destination queue is an important queue to secure. It is local to the subscriber, and publications that matched the subscription that is placed on it. You need to consider access to the destination queue from two perspectives:

1. Putting a publication on to the destination queue.
2. Getting the publication off the destination queue.

Distributed publish/subscribe internal messages such as proxy subscriptions and publications are put to distributed publish/subscribe system queues (such as SYSTEM.INTER.QMGR.CONTROL) by the receiving channel by using normal channel security rules.

## Topic security

- Access control can be set for a defined topic object (*not* a topic string)
- Authority checks on topic objects in the tree
  - Go up the tree, just like attributes
  - Keep checking until an authorization is found or reach the end of the tree
- Authority check on the destination queue for the subscription
- Pick a suitable layer of the topic hierarchy and set access control at this layer
- Use caution when adding more access control at higher levels in the tree as it can cause confusion and grant wide authorizations

[Introduction to distributed publish/subscribe](#)

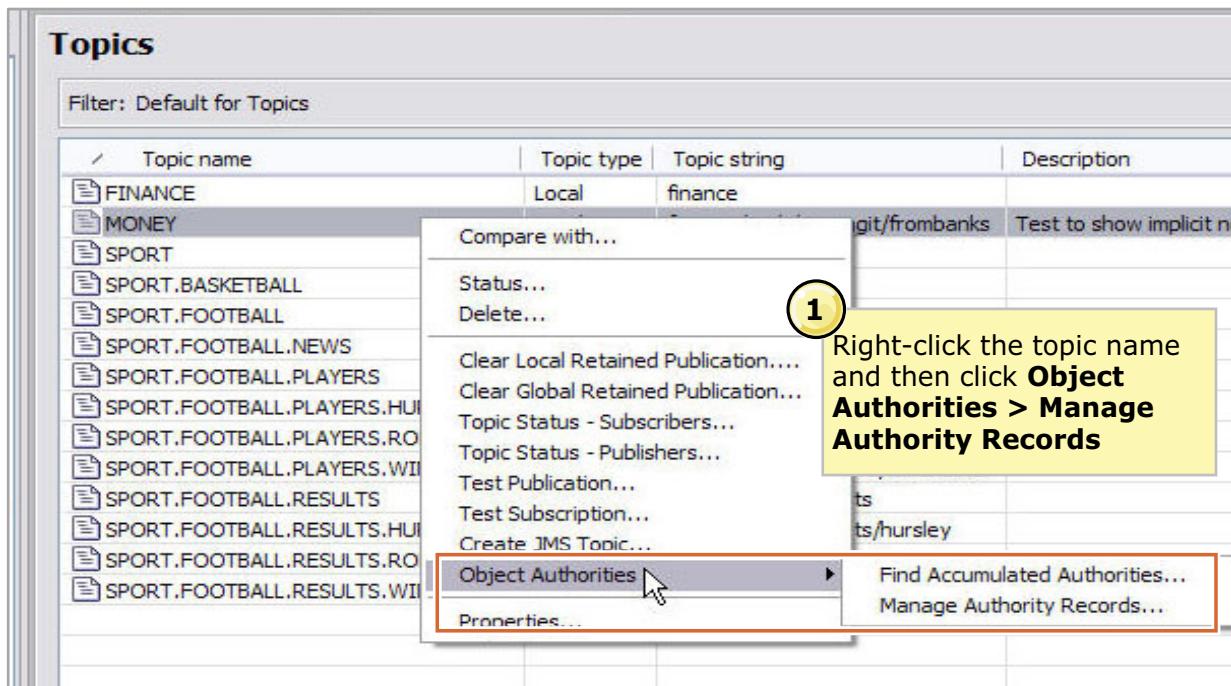
© Copyright IBM Corporation 2017

*Figure 7-42. Topic security*

This figure summarizes topic security.

# IBM Training

## Controlling topic security with IBM MQ Explorer (1 of 3)



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

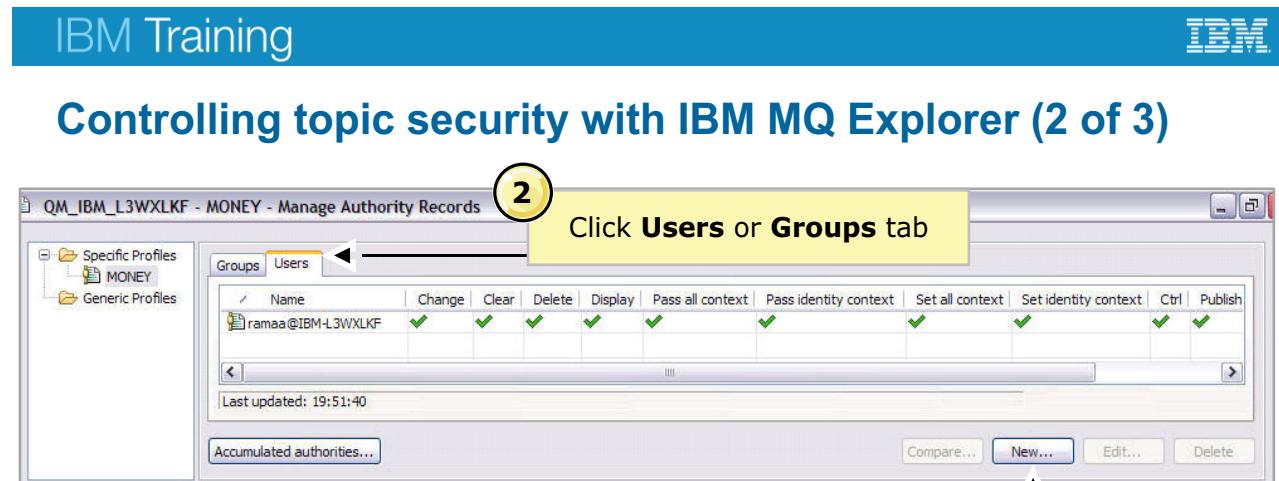
Figure 7-43. Controlling topic security with IBM MQ Explorer (1 of 3)

You can apply authorization to topic objects, just like queues and other objects.

You can control who has access to a topic, and for what purpose, in IBM MQ Explorer or by using the command `setmqaut` with a list of administrative topic objects.

This example shows you how to assign “publish” authority to a user MQUSER for the topic MONEY.

1. In the **Topics** view in IBM MQ Explorer, right-click the topic and then click **Object Authorities > Manage Authority Records**.



[Introduction to distributed publish/subscribe](#)

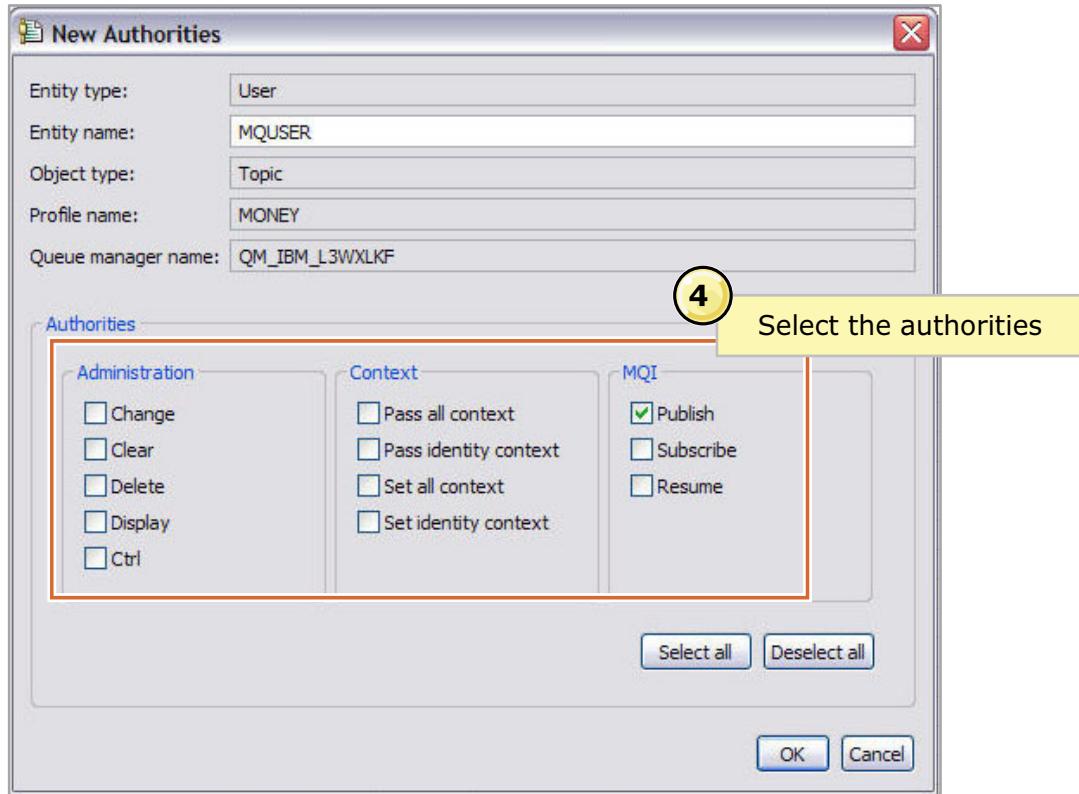
© Copyright IBM Corporation 2017

Figure 7-44. Controlling topic security with IBM MQ Explorer (2 of 3)

2. On the left pane, expand the **Specific Profiles** option, which shows the profile for the topic. Select the profile to show the authorities for this topic. By default, groups and users that have access to this topic are shown.  
Click the **Users** tab to show the users who have access to the topic or the **Groups** tab to show the groups that access to the topic.
3. Click **New** to define a new authority record or select a user or group name and then click **Edit** to modify an existing authority record.



## Controlling topic security with IBM MQ Explorer (3 of 3)



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-45. Controlling topic security with IBM MQ Explorer (3 of 3)

4. Three authorization operations can apply only to topics: **Publish**, **Subscribe**, and **Resume**.

Under the **MQI** column, select **Publish** and then click **OK**.

## Controlling security topic with setmqaut

- Control authorities for publish, subscribe, and resume

Examples:

Allow the “users” group to subscribe to SPORT:

```
setmqaut -m QM1 -n SPORT -t topic -g users +sub
```

Allow the “journalist” group to publish to SPORT:

```
setmqaut -m QM1 -n SPORT -t topic -g journalist +pub +sub
```

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-46. Controlling security topic with setmqaut

The **setmqaut** command can be used to control “publish” and “subscribe” authority by specifying the **+pub** (publish) and **+sub** (subscribe) attributes.



### Note

The **-pub** and **-sub** attributes do not block the publication. They clear the authority on that topic. The OAM looks higher in the tree for a **+pub** and **+sub** authority.

## Basic IBM MQ security on publish/subscribe queues

- IBM MQ publish/subscribe uses standard queues, and when subscribing must have access to the reply-to queue
- When using managed subscriptions, need access to:
  - SYSTEM.NDURABLE.MODEL.QUEUE
  - SYSTEM.DURABLE.MODEL.QUEUE
- Requires connect access to the queue manager

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-47. Basic IBM MQ security on publish/subscribe queues

The application or user that sends the MQSUB request must have the authority to put messages to the destination queue. The authority check follows the existing rules for queue security checking.

For IBM MQ publish/subscribe operations, authorize the user to have the appropriate access to the destination queue (reply-to queue).

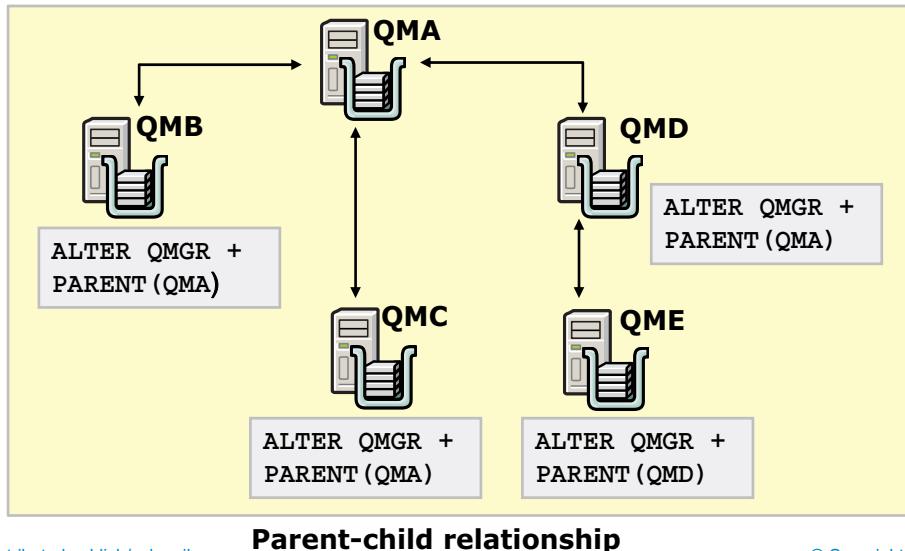
When you resume or alert a subscription, ensure that there is put, get, and browse authority on the destination queue.

The system-managed operations use the SYSTEM.NDURABLE.MODEL.QUEUE and SYSTEM.DURABLE.MODEL.QUEUE; the user needs access to these queues too.

## 7.5. Publish/subscribe topologies

## Publish/subscribe hierarchies

- Manually defined relationship between each queue manager
- Messages flow by using those relationships
- Each queue manager must have publish/subscribe mode (PSMODE) enabled
- Use queue manager PARENT attribute to define relationships



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-48. Publish/subscribe hierarchies

A hierarchy manually defines the relationship between each queue manager. The queue manager relationships control the flow of messages.

By default, publications flow between queue managers from publishers to subscribers in a hierarchy.

When you implement a publish/subscribe hierarchy, consider the scalability of the *higher* nodes in the hierarchy, especially the root node as more publication traffic is expected to flow through these nodes.

The figure shows an example of publish/subscribe hierarchy. The PARENT queue manager identifies the relationship to the other queue managers in the hierarchy.

Channels must exist for the hierarchy to function.

- Transmit queues must have the same name as the remote queue manager.
- Each queue manager must be enabled for publish/subscribe with this command: **ALTER QMGR PSMODE(ENABLED)**

## Publish/subscribe hierarchy configuration

- Every queue manager defines its *parent* in the hierarchy
- Each pair of directly related queue managers must be able to connect to each other by using IBM MQ channels
- By default all subscription knowledge is shared across the hierarchy
  - No requirement to define administered topic objects or modify applications
  - Controlled by *subscription scope* and *publication scope*
- After parent relationship, related queue managers should show each other when displaying PUBSUB status
- Informational messages are written to the queue manager logs when a relationship is established

Example:

AMQ5964: Pub/sub hierarchy connected.  
EXPLANATION: A pub/sub hierarchy connection has been established with parent or child queue manager 'QMA'

Figure 7-49. Publish/subscribe hierarchy configuration

This figure lists some guidelines for defining a publish/subscribe hierarchy.

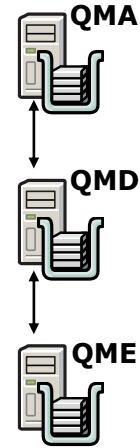
## Display publish/subscribe hierarchy status

```
DIS PUBSUB [TYPE(LOCAL | CHILD | PARENT | ALL)]
```

- Display publish/subscribe status information for a queue manager
  - LOCAL: Display the publish/subscribe status for this queue manager
  - CHILD: Display the publish/subscribe status for child connections
  - PARENT: Display the publish/subscribe status for the parent connection
  - ALL: Display the publish/subscribe status for this queue manager and for parent and child hierarchical connections

Example:

```
runmqsc QMD
Starting MQSC for queue manager QMD.
DISPLAY PUBSUB
1: DISPLAY PUBSUB
AMQ8723: Display pub/sub status details
QMNAME (QMD)  TYPE (LOCAL)
AMQ8723: Display pub/sub status details
QMNAME (QME)  TYPE (CHILD)
AMQ8723: Display pub/sub status details
QMNAME (QMA)  TYPE (PARENT)
```



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-50. Display publish/subscribe hierarchy status

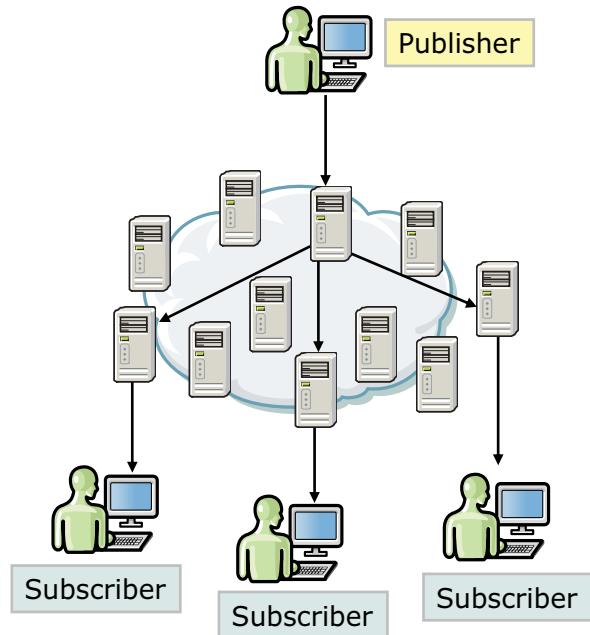
You can use the MQSC DISPLAY PUBSUB command to display publish/subscribe status information for a queue manager.

The TYPE attribute in the results indicates the hierarchy relationship to the other queue managers.

Entering the DISPLAY PUBSUB without attributes displays a summary of the relationships. In the example, the queue manager that the DISPLAY PUBSUB command is run against (QMD) has a child queue manager (QME) and a parent queue manager (QMA).

## Publish/subscribe clusters

- Any queue manager can publish and subscribe to topics
- Published messages can go directly between queue managers or can be routed to specific queue manager
- Use **ALTER QMGR PSCLUS()** command to control whether this queue manager participates in publish/subscribe activity across any clusters in which it is a member



[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-51. Publish/subscribe clusters

Using clusters in a publish/subscribe topology provide the following benefits.

- Messages that are destined for a specific queue manager in the same cluster can be transported directly to that queue manager and do not need to pass through an intermediate queue manager.
- There is no single point of failure in this topology. If one queue manager is not available, publications and subscriptions can still flow through the rest of the publish/subscribe system because each queue manager is directly connected with each other.
- If your clients are geographically dispersed, set up a cluster in each location, and connect the clusters (by joining a single queue manager in each cluster) to optimize the flow of publications and subscriptions.
- You can group clients according to the topics to which they publish and subscribe.
- A subscribing application can connect to its nearest queue manager to improve its own performance.
- The number of clients per queue manager can be reduced by adding more queue managers to the cluster to share workload, which makes a publish/subscribe cluster topology highly scalable.

If you have several queue managers in your publish/subscribe system, many channels are required to connect these queue managers together. However, the connections between queue managers can be created automatically to reduce the administrative work load.

Use the PSCLUS attribute on the ALTER QMGR command to control whether a queue manager participates in publish subscribe activity across any clusters in which it is a member. No clustered topic objects can exist in any cluster when you change it from ENABLED to DISABLED.

## Cluster topics

- Publish/subscribe in a cluster requires a *clustered* administered topic definition on *one* of the queue managers in the cluster

Example: `DEFINE TOPIC(FRUIT) TOPICSTR('/Price/Fruit') + CLUSTER(CLUS1)`

- Publish/subscribe cluster exists when one or more cluster topics exist
  - Cluster topic is propagated to all queue managers in the cluster with matching subscriptions
  - Channels are automatically defined between all queue managers in the cluster
- Every queue manager in the cluster automatically learns about the clustered topic and shares subscription knowledge for any topic strings within that branch of the topic tree

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-52. Cluster topics

A publish/subscribe cluster is created when a clustered topic is defined. This definition is shared with all members of the cluster. So publications on the clustered topic are shared with all members of the cluster.

Publications do not flow between queue managers in a cluster unless the branch of the topic tree is clustered by setting the CLUSTER attribute of an administered topic object.

Like traditional clusters, publish/subscribe clusters are designed for a many-many queue manager connectivity. In traditional clusters, cluster objects are automatically defined based on usage, such as being put to a queue. The usage model is based on a putter that uses a set of cluster queues (which are not necessarily defined on all queue managers in the cluster). Therefore, in traditional clusters, it is unlikely that all queue managers are connected to all other queue managers by automatically defined channels.

In publish/subscribe clusters, cluster objects are automatically defined before usage at the time the first cluster topic is defined in the cluster because the usage model is different from traditional clusters. Channels are required from any queue manager through which a subscriber for a cluster topic is connected to all the other queue managers. The channels are required so that a proxy subscription can be fanned out to all the queue managers. Channels are also required back from any queue manager in which a publisher (for a cluster topic) is connected to those queue managers that have a connected subscriber. Therefore, in publish/subscribe clusters, it is much more likely

that all of the queue managers are connected to all of the other queue managers by automatically defined channels. It is for this reason that cluster objects are automatically defined before usage in publish/subscribe clusters.

To define a cluster topic, you must provide a topic object name, topic string, and cluster name. Make the queue manager on which the topic is defined a member of the specified cluster.

When displaying cluster topics, two types of objects are available.

- **Local:** Directly administrable objects
- **Cluster:** Cluster cache records, which are based on local objects

## Cluster topic example

Define cluster topic:

```
DEFINE TOPIC(SPORTS) TOPICSTR('/global/sports/football') +
CLUSTER(SPORTS)
```

Display cluster topic:

DISPLAY TOPIC(FOOTBALL)	Local objects only
DISPLAY TOPIC(FOOTBALL) TYPE(LOCAL)	Local and cluster objects
DISPLAY TOPIC(FOOTBALL) TYPE(ALL)	Cluster objects only
DISPLAY TOPIC(FOOTBALL) TYPE(CLUSTER)	
DISPLAY TOPIC(FOOTBALL) CLUSINFO	
DISPLAY TCLUSTER(FOOTBALL)	
DISPLAY TOPIC(FOOTBALL) TYPE(ALL) CLUSINFO	

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-53. Cluster topic example*

The figure shows how to define a cluster topic with the DEFINE TOPIC command.

It also shows some examples for displaying the cluster topic by using the DISPLAY TOPIC and DISPLAY TCLUSTER commands.

For example, to display only cluster objects enter: DISPLAY TOPIC(TopicName) TYPE(CLUSTER)

## Display cluster topic example

```
DISPLAY TOPIC(*) TYPE(CLUSTER) TOPICSTR CLUSQMGR CLROUTE
CLSTATE
AMQ9633: Display topic details
    TOPIC(FRUIT)           TYPE(CLUSTER)
    TOPICSTR(/global/sports) CLUSQMGR(QMGR2)
    CLROUTE(DIRECT)        CLSTATE(ACTIVE)
```

If the cluster topic is not visible on a queue manager, check that the cluster channels to and from the full repositories are functioning correctly and that no errors are reported on those full repositories

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-54. Display cluster topic example*

The figure shows an example of the DISPLAY TOPIC TYPE(CLUSTER) command.

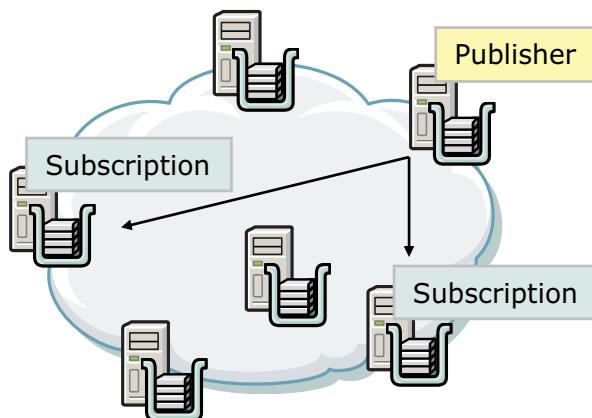
The CLROUTE attribute identifies the publish/subscribe cluster routing as direct or topic host. Cluster routing options are described in more detail next.

The CLSTATE attribute provides feedback on the current state of this cluster object. Any state other than ACTIVE implies a problem that should be investigated.

## Cluster modes for routing publications

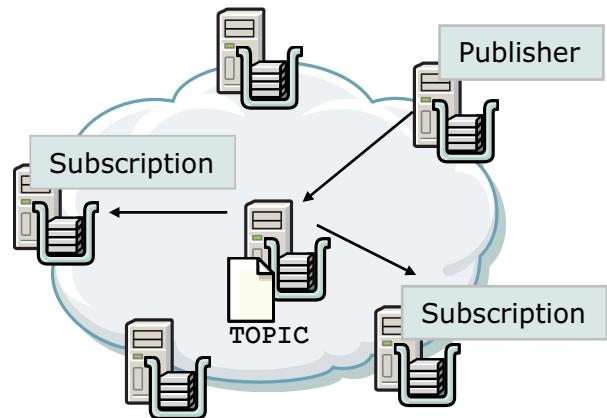
### Direct routing

- Publications are sent directly from the publisher queue manager to every queue manager in the cluster with a matching subscription
- Default mode



### Topic host routing

- Publications are routed through the queue manager where the clustered topic definitions are defined
- Multiple queue managers can host the same clustered topic



[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-55. Cluster modes for routing publications

IBM MQ supports two types of routing for publish/subscribe clusters.

In a direct routing cluster, all queue managers in the cluster are aware of every other queue manager and sends publications directly to other queue managers in the cluster with matching subscriptions. Direct routing is the default.

In a cluster that uses topic host routing, only the queue managers where the topic object is defined are aware of the other queue managers in the cluster. Publications are forwarded to these topic host queue managers, which then forward the publication to the queue managers with matching subscriptions.

## Cluster topic routing

- Cluster topics can be defined on any queue manager in the cluster, which includes full or partial repository queue managers
- Identify topic message routing with CLROUTE
  - For direct routing (the default): CLROUTE (DIRECT)
  - For topic host routing: CLROUTE (TOPICHOST)
- Cluster topics can be defined on more than one queue manager
  - Make duplicate topic definitions identical
  - If a mismatch in the topic definitions is found, the conflict is reported
  - Typically used with topic host routing only

Example:

```
On QM1: DEF TOPIC(SPORTS) TOPICSTR('/Sport/Scores') +
          CLUSTER(DEMO) CLROUTE(TOPICHOST)
On QM2: DEF TOPIC(SPORTS) TOPICSTR('/Sport/Scores') +
          CLUSTER(DEMO) CLROUTE(TOPICHOST)
```

*Figure 7-56. Cluster topic routing*

A cluster topic host is a queue manager in which a clustered topic object is defined. Clustered topic objects can be defined on any queue manager in the publish/subscribe cluster. When at least one clustered topic exists within a cluster, the cluster is a publish/subscribe cluster.

To guard against a failure in a topic host routing cluster, you can identically define all clustered topic objects on two highly available queue managers. For example, take a scenario in which the single topic host of a clustered topic object is lost due to disk failure. Any cluster topic cache records that are based on the clustered topic object and that exist in the other queue managers' cluster cache are usable within the cluster for up to 30 days or until the cache is refreshed. The clustered topic object can be redefined on a functioning queue manager. If a new object is not defined, all members of the cluster report, up to 27 days after the host queue manager failure, that an expected object update was not received.

It is not necessary for full repositories and topic hosts to overlap, or be separated. In publish/subscribe clusters that have just two highly available servers among many servers, define both of the highly available servers as full repositories and cluster topic hosts. In publish/subscribe clusters with many highly available servers, define full repositories and cluster topic hosts on separate highly available servers. This configuration allows the operation and maintenance of one function without affecting the operation of the other functions.

Duplicating a topic on another queue manager is typically used for backup support when the publish/subscribe network used topic host routing. Direct routing makes the topic available to all queue managers in the cluster, so it is not necessary to define the same cluster topic.

## Routing behavior for publish/subscribe clusters

- Setting the CLROUTE attribute at a point in the topic tree causes the entire branch beneath it to route topics by using that method
- All clustered definitions of the same named topic object in a cluster must have the same CLROUTE setting
- Verify the CLROUTE attribute for all topics on all hosts in the cluster with the MQSC command: `DISPLAY TCLUSTER (*) CLROUTE`
- After a topic object is clustered, you cannot change the value of the CLROUTE property unless you first remove the topic from the cluster (set CLUSTER to ' ' on the topic object)
- To stop a given queue manager from acting as a topic host for a cluster topic, either delete the topic object, or use the PUB(DISABLED) attribute to quiesce message traffic for the topic

## Topic host routing requirements

- Publications are forwarded to these topic host queue managers that then forward publications to queue managers with matching subscriptions
- Only the queue managers where the topic object is defined are aware of all other queue managers
- Some queue managers must be at IBM MQ V8 or higher:
  - Topic hosting queue managers
  - Full repository queue managers
  - Queue managers where subscriptions exist
  - Queue managers where publishers connect
- Any queue managers that are not IBM MQ V8 or higher are not aware of the topic host route and behave as if it was not defined in the cluster

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-58. Topic host routing requirements

To use topic host routing in a cluster, the following queue managers must be at IBM MQ V8 or above:

- The topic host queue managers
- The full repository queue managers
- The queue managers where subscriptions exist
- The queue managers where publishers connect

Any queue managers that are not on IBM MQ V8 do not recognize the topic host routed topic definition and continue to behave as if it was not defined in the cluster.

## Subscription propagation

- Subscription for a topic string is created on a queue manager
- Other queue managers register *proxy subscriptions* to indicate interest in the topic on behalf of applications that subscribed locally
- When a message is published to the topic string, a copy of the message is sent over IBM MQ channels to each queue manager with a proxy subscription
- Receiving queue manager processes the message and gives a copy to each subscription it holds (including any proxies)

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-59. Subscription propagation

Proxy subscriptions are a special type of subscription that tells one queue manager that another queue manager needs a copy of any matching publications.

Proxy subscriptions can be viewed on a queue manager by using MQSC or IBM MQ Explorer. This proxy subscription shows which queue managers receive a publication for a topic.

If a queue manager has multiple subscriptions to the same topic string, a single proxy subscription is generated.

Proxy subscriptions do not include any selectors set on the subscription.

## Viewing proxy subscriptions

- Proxy subscriptions can be viewed on a queue manager by using MQSC or IBM MQ Explorer

Example MQSC command:

```
DISPLAY SUB(*) SUBTYPE(PROXY) TOPICSTR DESTQMGR
AMQ8096: MQ subscription inquired
    SUBID(414D5120514D5752312020202021123C5320000E2)
        SUB(SYSTEM.PROXY.QMGR2 CLUS1 /X/Y)      TOPICSTR(/X/Y)
        DESTQMGR(QMGR2)                          SUBTYPE(PROXY)
AMQ8096: MQ subscription inquired
    SUBID(414D5120514D5752312020202021123C5320000E5)
        SUB(SYSTEM.PROXY.QMGR2 CLUS1 /X/Z)      TOPICSTR(/X/Z)
        DESTQMGR(QMGR2)                          SUBTYPE(PROXY) )
```

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-60. Viewing proxy subscriptions

You view proxy subscriptions on a queue manager by using MQSC or IBM MQ Explorer. The figure provides an example of the MQSC command and the command results.

## 7.6. Comparing topologies

## Proxy subscription in a hierarchy

- Proxy subscriptions are sent from a queue manager to every **directly** connected queue manager in the hierarchy
- Each directly connected queue manager sends proxy subscriptions to other relations until the queue manager in the hierarchy knows about the topic string
- Publications are sent down the path of proxy subscriptions

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

*Figure 7-61. Proxy subscription in a hierarchy*

In a hierarchy, proxy subscriptions are sent from a queue manager to every directly connected queue manager in the hierarchy.

## Proxy subscriptions in a cluster

- Direct routed cluster topics
  - Proxy subscriptions are sent from a subscribing queue manager to **every** other queue manager in the cluster
  - Every queue manager is aware of every other queue manager in the cluster
  - Any publication from a queue manager is sent directly to those queue managers that sent proxy subscriptions
- Topic host routed cluster topic
  - Proxy subscriptions are sent from a subscribing queue manager to **only** those queue managers that host a definition of the clustered topic
  - Only the topic hosts are aware of every other queue manager in the cluster
  - Any publication from a queue manager is **always** sent to one of the topic hosts, who then forwards the message to any subscribing queue managers

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-62. Proxy subscriptions in a cluster*

In a direct routed publish/subscribe cluster, proxy subscriptions are sent from a subscribing queue manager to every other queue manager in the cluster.

In a topic host routed publish/subscribe cluster, proxy subscriptions are sent from a subscribing to only those queue managers that host a definition of the clustered topic only.

In a direct routed publish/subscribe cluster, every queue manager knows about every other queue manager in the cluster. In a topic host routed publish/subscribe cluster, only the topic hosts are aware of every other queue manager in the cluster.

## Scaling

- Direct routed clusters
  - Share subscription knowledge across all queue managers
  - Result in the establishment of many queue manager channels, even to queue managers with no publishers or subscribers
- Topic routing clusters
  - Requires few channel connections
  - Hosting different topics on different queue managers can increase scaling
- Hierarchies
  - Channel connectivity is restricted
  - Share subscription knowledge across all queue managers

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-63. Scaling*

This figure compares the scalability for each distributed publish/subscribe topology.

## Publication flows

- Direct routed clusters
  - Ensure that publications take the shortest path between publishers and subscribers
- Topic host routed clusters
  - Can introduce an extra hop between publishers and subscribers
  - Requires careful placement of subscribers so that publishers can achieve the same single-hop route that direct clusters provide
- Hierarchies
  - Can have multiple hops between publishers and subscribers

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-64. Publication flows

This figure compares how the supported publish/subscribe topologies handle publication flows. Direct routed clusters ensure that the publications take the shortest path between publishers and subscribers. The other topologies can introduce one or more extra hops between publishers and subscribers.

## Configuration

- Direct routed clusters
  - Allow queue managers to join and leave without much consideration after the overall plan is defined
- Topic host routed clusters
  - Allow queue managers to join and leave without much consideration after the overall plan is defined
  - Careful planning of topic host routers is required
- Hierarchies
  - Requires detailed planning and configuration of queue managers
  - Harder to alter the hierarchical structure

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-65. Configuration*

This figure compares how configuration differs for the supported publish/subscribe topologies.

Direct routed clusters require the least amount of configuration. Queue manager join and leave the clusters without affecting the network.

Topic host routed clusters require some planning and configuration of topic host routers. Care must be taken when you remove queue managers from this type of topology.

Hierarchies require the most configuration and planning. It is also harder to add or remove queue managers from a publish/subscribe hierarchy without affecting other queue managers in the topology.

## Availability

- Direct routed cluster
  - Queue manager that is not available affects the subscribers and publishers on that queue manager only
- Topic host routed cluster
  - If a queue manager that is not hosts the only definition of a topic, that queue manager can affect other queue managers
  - Hosting different topics on different queue managers can increase availability
- Hierarchies
  - Queue manager that is not available can cut off many other queue managers

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-66. Availability*

This figure compares the effects of an unavailable queue manager in the different publish/subscribe topologies.

## Comparison summary

- Scaling
  - Topic host routed clusters provide the best options for scaling
- Availability
  - Clusters can be configured to avoid single points of failures, unlike hierarchies
- Publication performance
  - All topologies use similar queue manager techniques for transferring messages
  - All topologies can be designed to minimize routes between queue managers
- Configuration
  - Clustering is the most dynamic and simplest solution, especially where clusters are already in use

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-67. Comparison summary*

How do the three topologies compare?

Publish/subscribe clusters are highly available because the cluster is fully connected. Publication delivery is also fast due to direct connections. They are flexible, scalable, and have simpler administration. Topic host routed clusters provide the best options for scaling.

Hierarchies create proxy subscriptions when each queue manager is sent a proxy subscription. A hierarchy is scalable but the reorganization of the hierarchy can be labor-intensive. Availability is impacted if intermediate nodes are not running.

## 7.7. Troubleshooting

## Tips

- Start small
- Do not put the root node, indicated by a forward slash (/), into a cluster
  - Make global topics obvious, such as /global or /cluster
- Use care when designing deep hierarchies
  - Shallow hierarchy with a highly available parent works well
- Monitor the depth of transmission queues
  - SYSTEM.CLUSTER.TRANSMIT.QUEUE
  - Hierarchy transmit queues
- Manage subscriptions
  - Do you need durable subscribers?
  - Did you stop subscribers before stopping the queue manager?
- Be careful when mixing traditional clusters
  - Large clusters and cluster topic objects require many channels

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

Figure 7-68. Tips

Start small with your choice of distributed clustering. It is easy to add nodes to the distribution topology when you use clustering or a hierarchy.

Do not put the topic root node forward slash (/) into a cluster. Create some *base* or *global* topic strings like /global or /cluster.

Be careful of deep hierarchies. Leaf node publications must traverse the tree to the destination. It is better to have a shallow hierarchy with highly available parent nodes.

Monitor the depth of transmission queues. Unavailable queue managers or channels can result in a significant buildup of undelivered publications that wait on the transmission queues. For clustered topologies, message buildup is apparent for SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Stop subscriptions before you stop a queue manager to prevent a buildup on the transmission queues.

Use durable subscriptions when necessary only.

Be careful when you use existing clusters for publish/subscribe clustering. Publish/subscribe clusters must be fully connected. Many channels are automatically defined.

## Problem determination (1 of 2)

- Default behavior in IBM MQ depends on the persistence of the message and the durability of the subscription
  - For persistent messages (**PMSGDLV=ALLDUR**), publish fails if it cannot be delivered to one or more **durable** subscriptions; failures to any **non-durable** subscriptions are *acceptable*
  - For non-persistent messages (**NPMSGDLV=ALLAVAIL**), any failures to subscriptions are acceptable, the publish succeeds (even if no subscribers can receive the message)
- If a dead-letter queue is configured, a failure to put a publication to a subscription queue results in an attempt to put it to the dead-letter queue
  - Put to dead-letter queue is classified as a success
  - Controlled by using the **USEDLQ** option on a topic definition

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-69. Problem determination (1 of 2)

The default behavior for publish/subscribe depends on the persistence of the message and the durability of the subscription. This behavior can be changed through configuration of the PMSGDLV and NPMSGDLV attributes of a topic.

If a dead-letter queue is configured, a failure to put a publication to a subscription queue results in an attempt to put it to the dead-letter queue.

## Problem determination (2 of 2)

- Double check your configuration
- When in a cluster, check each cluster knowledge of the queue manager
- When in a hierarchy, check the queue manager relationships
- Check the channels
- Check the proxy subscriptions
- Check the topic status
- Watch for publication movement
- Look for a buildup of messages on a queue

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-70. Problem determination (2 of 2)*

Double check your configuration. Topic objects inherit behavior from higher topics, which can affect how lower topics behave.

When in a cluster, check each queue manager's cluster knowledge. One option for checking the queue manager's cluster knowledge is by using the DISPLAY TOPIC(\*) TYPE(CLUSTER) command.

When in a hierarchy, check the queue manager relationships by using the DISPLAY PUBSUB command or IBM MQ Explorer.

Check your channels. If the channels to and from the queue managers are not running, publication traffic, proxy subscriptions, topic configuration (clusters), or relationships (hierarchies) do not flow.

Check the proxy subscriptions on each queue manager by using the DISPLAY SUB(\*) SUBTYPE(PROXY) command.

Check the topic status. Use the DISPLAY TPSTATUS( ) command to show the behavior of a topic string.

Watch for publication movement.

- Check the NUMMSGS value on the proxy subscriptions by using DISPLAY SBSTATUS to see whether publications were sent to the originator of the proxy subscription.

- Check the MSGS value on the channels to see whether they are flowing by using the DISPLAY CHSTATUS command.

Look for a build-up of messages. If there are problems, messages can build up on system queues. A build-up can occur when the system produces messages too quickly or an error occurs. Investigate to see whether messages are still being processed, but slowly, or no messages are processed.

Queues that should be empty when the publish/subscribe is running correctly are:

- SYSTEM.CLUSTER.TRANSMIT.QUEUE (or any other transmission queue involved): All queue manager outbound messages to other queue managers
- SYSTEM.BROKER.CONTROL.QUEUE: Requests to register/deregister subscriptions

If you are using publish/subscribe clusters, the following queues should also be empty:

- SYSTEM.INTER.QMGR.PUBS: Publications arriving from remote queue managers
- SYSTEM.INTER.QMGR.FANREQ: Requests to send proxy subscriptions
- SYSTEM.INTER.QMGR.CONTROL: Requests from other queue managers to register proxy subscriptions

If you are using publish/subscribe hierarchies, the following queues should also be empty:

- SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS: Requests from other queue managers in the hierarchy
- SYSTEM.BROKER.DEFAULT.STREAM: Publications arriving from remote queue managers (and local ‘queued’ publish/subscribe applications)

## Loop detection

- Loops in the publish/subscribe network are not good
  - Subscribers receive multiple copies of the same publication
  - It is detrimental to other workloads
  - Takes up system resources such as the processor, network, and IBM MQ processes
- Messages are fingerprinted
  - Outbound of publish/subscribe cluster with cluster name
  - On each hierarchy queue manager with queue manager name

[Introduction to distributed publish/subscribe](#)

© Copyright IBM Corporation 2017

*Figure 7-71. Loop detection*

When you create a cluster, it is possible to create a loop that causes messages to cycle forever within the network.

As publications move around a publish/subscribe topology, each queue manager adds a unique fingerprint to the message header. Whenever a publish/subscribe queue manager receives a publication from another publish/subscribe queue manager, the fingerprints that are held in the message header are checked. If its own fingerprint is already present, the publication fully circulated around a loop, so the queue manager discards the message, and adds an entry to the error log.

In a distributed publish/subscribe network, it is important that publications and proxy subscriptions cannot loop. Looping results in a flooded network with connected subscribers that receive multiple copies of the same original publication.

## Stopping a queue manager

- Disconnect subscribers before stopping a queue manager or channel initiator
  - When stopping a queue manager, local subscribers are disconnected, but they might not notify other queue managers in the publish/subscribe cluster or hierarchy
- If subscribers are not disconnected first
  - Durable subscribers: Transmission queue build-up is required
  - Nondurable subscribers: Transmission queue build-up might occur
- Resynchronize on start
- Try to avoid backlog of messages on SYSTEM.CLUSTER.TRANSMIT.QUEUE and hierarchy transmission queues

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-72. Stopping a queue manager

Shutting down a queue manager at a busy moment can result in a buildup of messages on the various transmission queues.

To avoid a buildup of messages, disconnect any subscribers before you shut down a queue manager. Dropping the subscriptions results in the removal of proxy subscriptions on remote queue managers and the prevention of message buildup on transmission queues.

Durable subscriptions are not dropped.

## Unit summary

- Describe publish/subscribe in IBM MQ
- Explain distributed publish/subscribe topologies
- Manage publish/subscribe topics, subscriptions, and topologies
- Compare publish/subscribe topologies

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

*Figure 7-73. Unit summary*

## Review questions

1. True or False: A subscriber is an application that can make only one subscription at a time to a topic on a particular queue manager.
2. True or False: All values of a topic object definition can be altered by using the ALTER TOPIC command.
3. What is the purpose of a proxy subscription?
  - A. It is an alias of another subscription.
  - B. It is a subscription that is made on behalf of another application.
  - C. It is a subscription that is sent to members of a cluster or hierarchy that indicates a local subscription at a remote queue manager.



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-74. Review questions

Write your answers here:

- 1.
- 2.
- 3.

## Review answers

- True or False: A subscriber is an application that can make only one subscription at a time to a topic on a particular queue manager.

The answer is False. An application can make multiple subscriptions to the queue manager for different topics.



- True or False: All values of a topic object definition can be altered by using the ALTER TOPIC command.

The answer is False. The TOPICSTR value cannot be altered. If this value needs to be changed, the topic object must be deleted and redefined.

- What is the purpose of a proxy subscription?

- A. It is an alias of another subscription.
- B. It is a subscription that is made on behalf of another application.
- C. It is a subscription that is sent to members of a cluster or hierarchy that indicates a local subscription at a remote queue manager.

The answer is C.

## Exercise: Configuring distributed publish/subscribe

Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

*Figure 7-76. Exercise: Configuring distributed publish/subscribe*

In this exercise, you define and test an IBM MQ publish/subscribe network by using a direct cluster and a topic host cluster. You also use the IBM MQ sample programs and IBM MQ Explorer to test the cluster and the IBM MQ display route command to show the message route through the publish/subscribe cluster.

## Exercise objectives

- Define a direct route publish/subscribe cluster
- Define a topic host route publish/subscribe cluster
- Test the publish/subscribe cluster
- Use the IBM MQ display route (`dspmqrte`) command to verify the route that the message takes through the publish/subscribe cluster



Introduction to distributed publish/subscribe

© Copyright IBM Corporation 2017

Figure 7-77. Exercise objectives

For detailed instructions, see the Course Exercises Guide.

# Unit 8. Supporting JMS with IBM MQ

## Estimated time

00:30

## Overview

In this unit, you learn about IBM MQ support for Java Message Service (JMS).

## How you will check your progress

- Review questions

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- Describe IBM MQ as a JMS provider
- Manage JMS resources in IBM MQ Explorer

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

*Figure 8-1. Unit objectives*

## 8.1. IBM MQ and JMS

## What is JMS?

- JMS is a messaging API for Java applications
- JMS V2.0 is current standard
- Part of Java Platform, Enterprise Edition (Java EE)
- Allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous
- Supports point-to-point and publish/subscribe messaging styles

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-2. What is JMS?

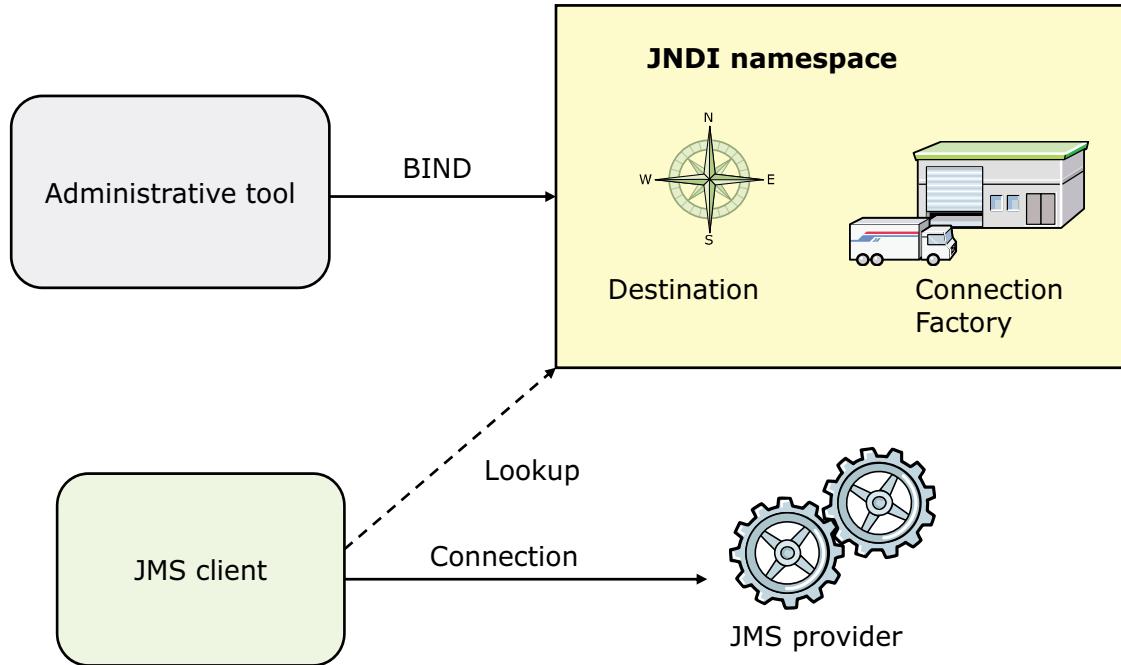
JMS is a messaging API specification. It defines how a Java application accesses messaging resources with the Java Platform, Enterprise Edition framework. The current version of the JMS standard is V2.0 (at time of publication of this course).

Like all Java specifications, operating system neutrality is a key design focus. Object code portability flows from this neutrality.

JMS allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous. In computing and systems design, a loosely coupled system is one in which each of its components has, or uses little or no knowledge of the definitions of other separate components.

JMS supports both the point-to-point and publish/subscribe messaging styles.

## JMS elements



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-3. JMS elements

A JMS provider is an implementation of the JMS interface for message-oriented middleware, such as IBM MQ. Providers are implemented as either a Java JMS implementation or an adapter to message-oriented middleware that is not Java.

A JMS client is an application or process that produces and receives messages.

Java Naming and Directory Interface (JNDI) is a Java API for a directory service that allows Java software clients to discover and look up data and objects by using a name. The JNDI namespace stores the JMS destination and connection factory administrative objects.

A JMS client must be able to look up the JMS provider-specific connection and location data in a way that is independent of the JMS provider itself. The JNDI namespace provides an implementation neutral mapping facility for this purpose. There is no set format for the JNDI namespace. It is not necessarily a database, although it might be. Some typical implementations include LDAP repositories down to plain text files. More often than not, the format of the JNDI repository is specific to the Java Platform, Enterprise Edition implementation.

In this figure, a JMS client consults the JNDI namespace to retrieve an object that creates a connection to the JMS provider.

The retrieved destination object provides a connection to the queue or topic.

## JMS administered objects

- Connection factory
  - Set of connection configuration parameters that a client uses to create a connection with a JMS provider
  - On IBM MQ, contains queue manager connection information
- Destination
  - Object that a client uses to specify the destination of messages it is sending and the source of messages it receives
  - On IBM MQ, identifies queue or topic information

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-4. JMS administered objects

JMS has two administered objects: the connection factory and the destination.

The JMS connection factory is the object that a client uses to create a connection with a JMS provider. In terms of IBM MQ, this object holds queue manager connection information.

The JMS destination specifies the source or destination of messages. For IBM MQ as a JMS provider, this object holds queue or topic information. These objects are stored in a central repository that is accessed by using the JNDI.

The JMS specification expects connection factory and destination objects to be administered objects. An administrator creates and maintains administered objects in a central repository, and a JMS application retrieves these objects by using the JNDI.

## JMS providers

- IBM MQ
- IBM Application Server: Service Integration Bus
- Open source: Apache ActiveMQ
- Just about any other Java EE provider

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

*Figure 8-5. JMS providers*

There are many JMS providers, including IBM MQ. Every Java Platform, Enterprise Edition implementation must have a JMS provider, so many JMS providers exist.

IBM Application Server provides Service Integration Bus (SIB) to provide JMS support. IBM Application Server also provides interfaces to external and generic JMS providers.

## IBM MQ classes for Java or JMS

IBM MQ classes for Java	JMS
<ul style="list-style-type: none"> <li>• Encapsulate MQI</li> <li>• Easy to implement for anyone familiar with MQI in other languages</li> <li>• Can use the full features of IBM MQ</li> <li>• Similar object model to other object-oriented language interfaces like C++ and .NET</li> <li>• Built with Java 7</li> </ul>	<ul style="list-style-type: none"> <li>• Implement Oracle JMS interfaces</li> <li>• Easy for anyone with JMS skills</li> <li>• Part of Java EE</li> <li>• Can use message-driven beans</li> <li>• Can use other JMS providers</li> <li>• Bridge applications between JMS providers</li> </ul>

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-6. IBM MQ classes for Java or JMS

The figure lists some of the considerations when you choose between IBM MQ classes for Java and JMS.

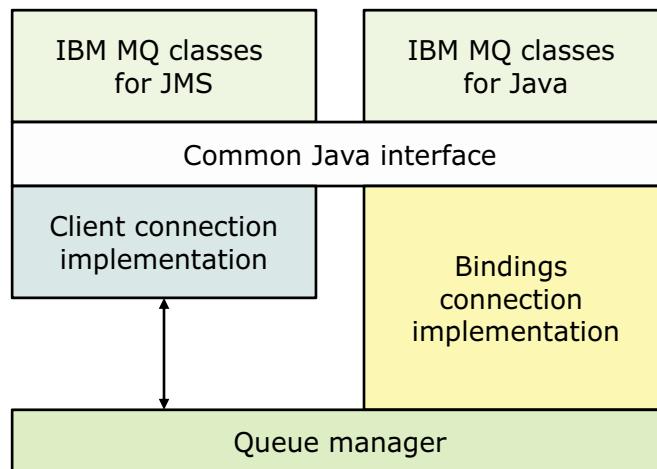
The IBM MQ classes for Java provide a close correlation to the underlying MQI. Programmers familiar with the MQI should have little trouble adapting that knowledge to the IBM MQ classes for Java. The full function of IBM MQ can be used.

Some of the JMS considerations are already described. Message-driven beans (MDB) are a Java Platform, Enterprise Edition way to drive code when a message arrives on a destination.

Bridging JMS providers of any heritage is easy with a JMS implementation.

## IBM MQ classes for JMS and IBM MQ classes for Java

- IBM MQ classes for Java and IBM MQ classes for JMS are peers that use a common Java interface to the MQI
  - More scope for optimizing performance
  - Setting fields or calling methods in the MQEnvironment class does not affect runtime behavior of code that is written by using IBM MQ classes for JMS



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-7. IBM MQ classes for JMS and IBM MQ classes for Java

As the figure shows, IBM MQ classes for Java and IBM MQ classes for JMS are peers that use a common Java interface to the MQI. This architecture allows for more flexibility when optimizing for performance.

The MQEnvironment contains static fields that control the environment in which an IBM MQ queue manager object (and its corresponding connection to IBM MQ) is constructed. Setting fields or calling methods in the MQEnvironment class does not affect the runtime behavior of code that is written by using IBM MQ classes for JMS.

## JMS message properties

- When an IBM MQ classes for JMS application sends a JMS message, the JMS message is mapped to an IBM MQ message
  - Some JMS header fields and properties are mapped into fields in the MQMD
  - Some JMS header fields and properties are mapped into fields in the MQRFH2 header
- When an application calls MQGET to receive a message from an IBM MQ classes for JMS application, the application can choose to receive the message in one of the following ways:
  - Message contains message descriptor, an MQRFH2 header that contains data that is derived from JMS header fields and properties, and the application data
  - Message contains message descriptor, application data, and a set of message properties

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-8. JMS message properties

A JMS message consists of a set of header fields, a set of properties, and a body that contains the application data. As a minimum, an IBM MQ message consists of a message descriptor and the application data.

When an IBM MQ JMS application sends a JMS message, IBM MQ maps the JMS message into an IBM MQ message. Some of the JMS header fields and properties are mapped into fields in the message descriptor. Other JMS header fields are mapped into fields in an extra IBM MQ header that is called an MQRFH2 header. When an IBM MQ JMS application receives a JMS message, IBM MQ does the reverse mapping.

## Support for JMS 2.0 in IBM MQ

- Delayed delivery
  - Defer message delivery by specifying a delivery delay when a message is sent so that the JMS provider does not deliver the message until after the specified delivery delay elapses
- Shared subscriptions
  - Share messages from a topic subscription among multiple consumers
  - Each message from the subscription is delivered to only one of the consumers on that subscription
- Asynchronous send operation
  - Applications can send messages asynchronously
  - Control is immediately returned to the sending application without waiting for a reply from the server
- Any code that is written to be used with IBM MQ must use the `jms.jar` file that is supplied with IBM MQ

[Supporting JMS with IBM MQ](#)

© Copyright IBM Corporation 2017

*Figure 8-9. Support for JMS 2.0 in IBM MQ*

IBM MQ supports the JMS 2.0 version of the JMS standard. This implementation offers all the features of the classic API but requires fewer interfaces and is simpler to use.

In IBM MQ, you can defer message delivery by specifying a delivery delay when a message is sent so that the JMS provider does not deliver the message until after the specified delivery delay elapses.

A shared subscription is used to share messages from a topic subscription among multiple consumers. Each message from the subscription is delivered to only one of the consumers on that subscription.

Applications can send messages asynchronously. When a message is sent asynchronously, control is immediately returned to the sending application without waiting for a reply from the server. This process enables the sending application to do something else.

Any code that is written to be used with IBM MQ must use the `jms.jar` file that is supplied with the IBM MQ.

## 8.2. Managing JMS resources with IBM MQ Explorer

## Configuring JMS administered objects in IBM MQ

- Before an application can retrieve administered objects from a JNDI namespace, an administrator must first create the administered objects
  - Connect to the JNDI namespace and add an initial context
  - Create and configure connection factory and destination objects that are stored in the JNDI namespace
- Use IBM MQ Explorer to create and maintain administered objects in a JNDI namespace from IBM MQ

Supporting JMS with IBM MQ

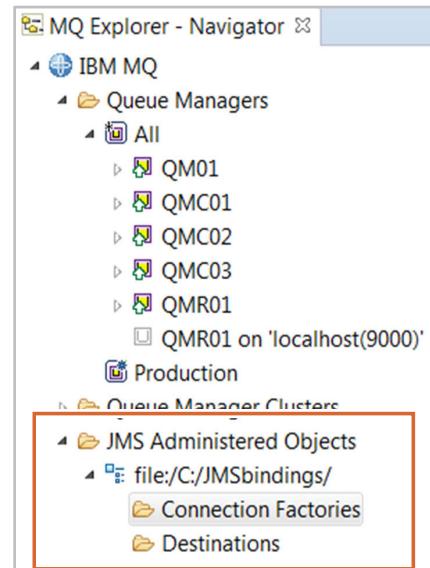
© Copyright IBM Corporation 2017

*Figure 8-10. Configuring JMS administered objects in IBM MQ*

The administered objects are stored in a naming and directory service that IBM MQ Explorer accesses by using the JNDI API. The administered objects are stored in locations on the naming and directory service that are known as the JNDI namespaces. Different JNDI service providers can be used as the naming and directory service, including LDAP and local or remote file systems.



- **JMS Administered Objects** folder contains initial contexts
  - An initial context must be added to IBM MQ Explorer before you can administer the JMS administered objects
  - You can add more initial contexts and you can remove initial contexts when you no longer want to use IBM MQ Explorer
  
- After you add the initial context to IBM MQ Explorer, you can create:
  - Connection factory objects
  - Destination objects
  - Subcontexts in the JNDI namespace



[Supporting JMS with IBM MQ](#)

© Copyright IBM Corporation 2017

Figure 8-11. IBM MQ Explorer and JMS

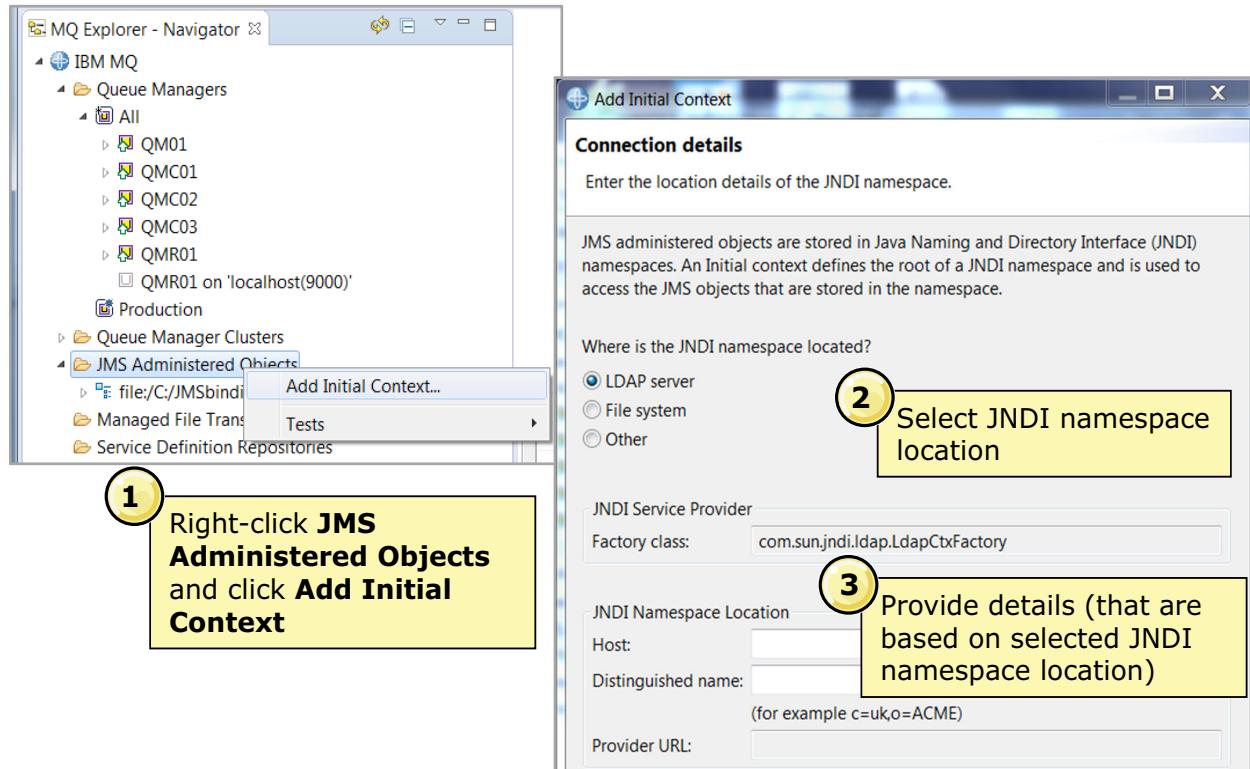
You can define connection factory objects, destination objects, and subcontexts in IBM MQ Explorer.

An initial context defines the root of a JNDI namespace in which JMS administered objects are stored.

A subcontext is a subdivision of a JNDI namespace and can contain connection factories and destinations and other subcontexts. A subcontext is not an object. It is an extension of the naming convention for the objects in the subcontext. You can create multiple subcontexts in a single context.



## Adding an initial context



Supporting JMS with IBM MQ

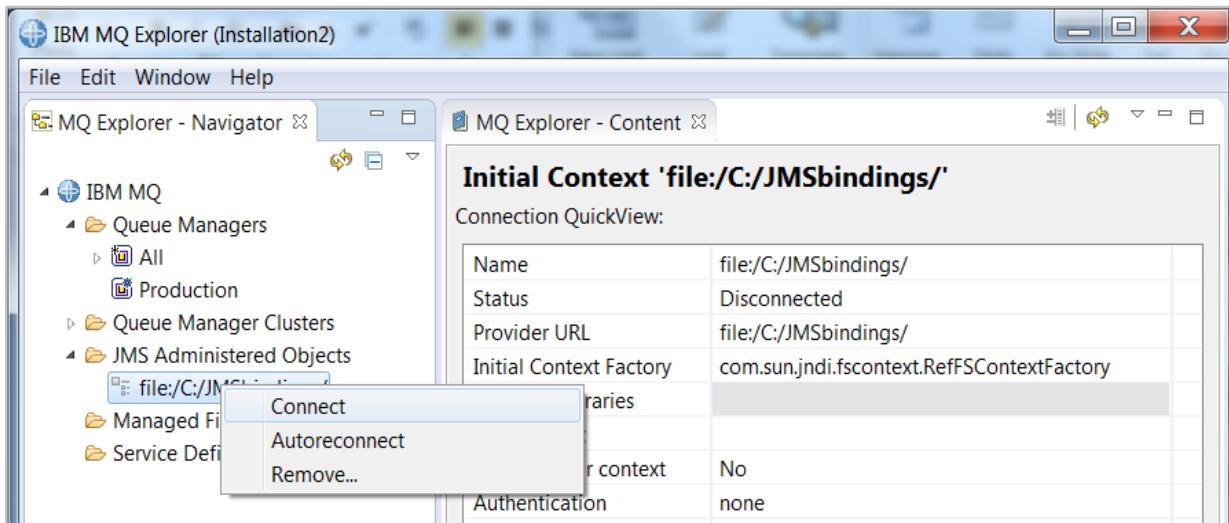
© Copyright IBM Corporation 2017

Figure 8-12. Adding an initial context

The first step in IBM MQ Explorer is to identify the **Initial Context** that identifies type and location of the JNDI repository. After a connection is established, the IBM MQ Explorer can be used to complete most JMS administration tasks.

1. Right-click the **JMS Administered Objects** folder in the **MQ Explorer - Navigator** view and then click **Add Initial Context**.
2. Select the location of the JNDI namespace in the naming and directory service.
  - Click **LDAP server** if the JNDI namespace is on an LDAP server. You must know the host name of the LDAP server and the distinguished name of the location of the JNDI namespace.
  - Click **File system** if the JNDI namespace is on a file system. You must know the path of the location of the JNDI namespace on the file system.
  - Click **Other** if the JNDI namespace is located somewhere else. You must know the name and location of the initial context factory class of the JNDI service provider and the URL of the location of the JNDI namespace.
3. Provide the details for the JNDI namespace.

## Connecting the initial context



- Initial context must be connected to create and administer connection factories and destinations

Supporting JMS with IBM MQ

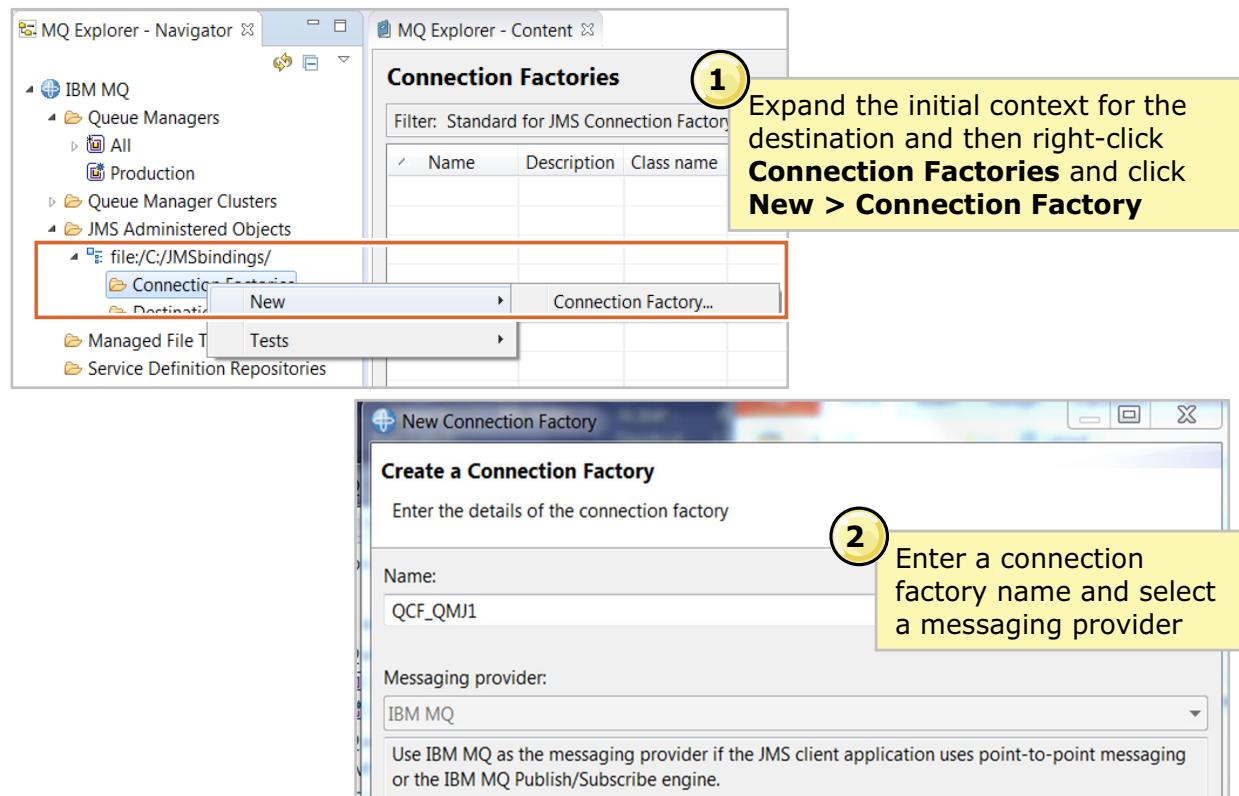
© Copyright IBM Corporation 2017

*Figure 8-13. Connecting the initial context*

You must connect to the initial context from IBM MQ Explorer before you can create other JMS administered objects.

Right-click the initial context under the **JMS Administered Objects** in the **MQ Explorer - Navigator** view and then click **Connect**.

## Creating a connection factory (1 of 3)



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

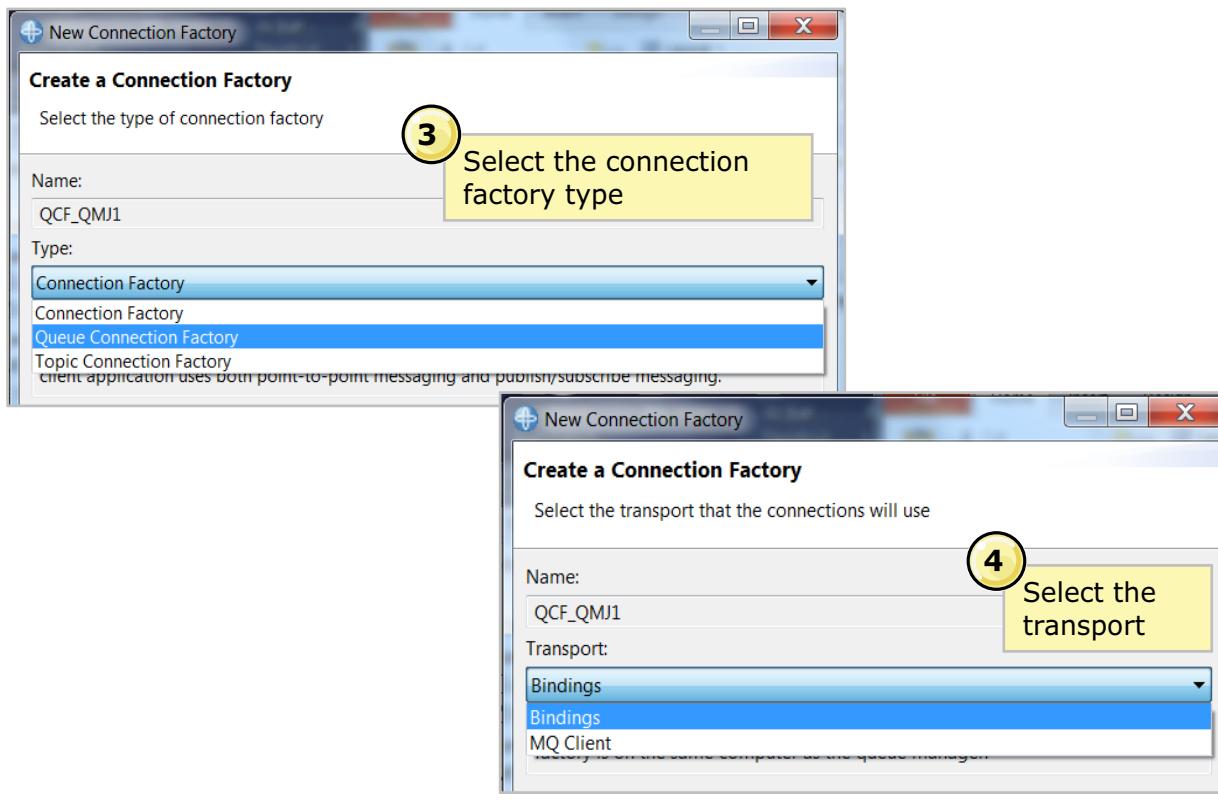
Figure 8-14. Creating a connection factory (1 of 3)

To create a connection factory in IBM MQ Explorer:

1. Right-click **Connection Factories** under **JMS Administered Objects** in the **MQ Explorer - Navigator** view and then click **New > Connection Factory**.
2. Enter a name for the connection factory and select the messaging provider.

A JMS client (that is, a Java application that uses the JMS API) uses connection factories to create connections to the JMS provider, which is a messaging provider such as IBM MQ. When you define a connection factory, you select the messaging provider that is used as the JMS provider. If you want to change the JMS provider, you must create a connection factory for the new JMS provider.

## Creating a connection factory (2 of 3)



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-15. Creating a connection factory (2 of 3)

3. Select the connection factory type. IBM MQ Explorer supports the following connection factory types.

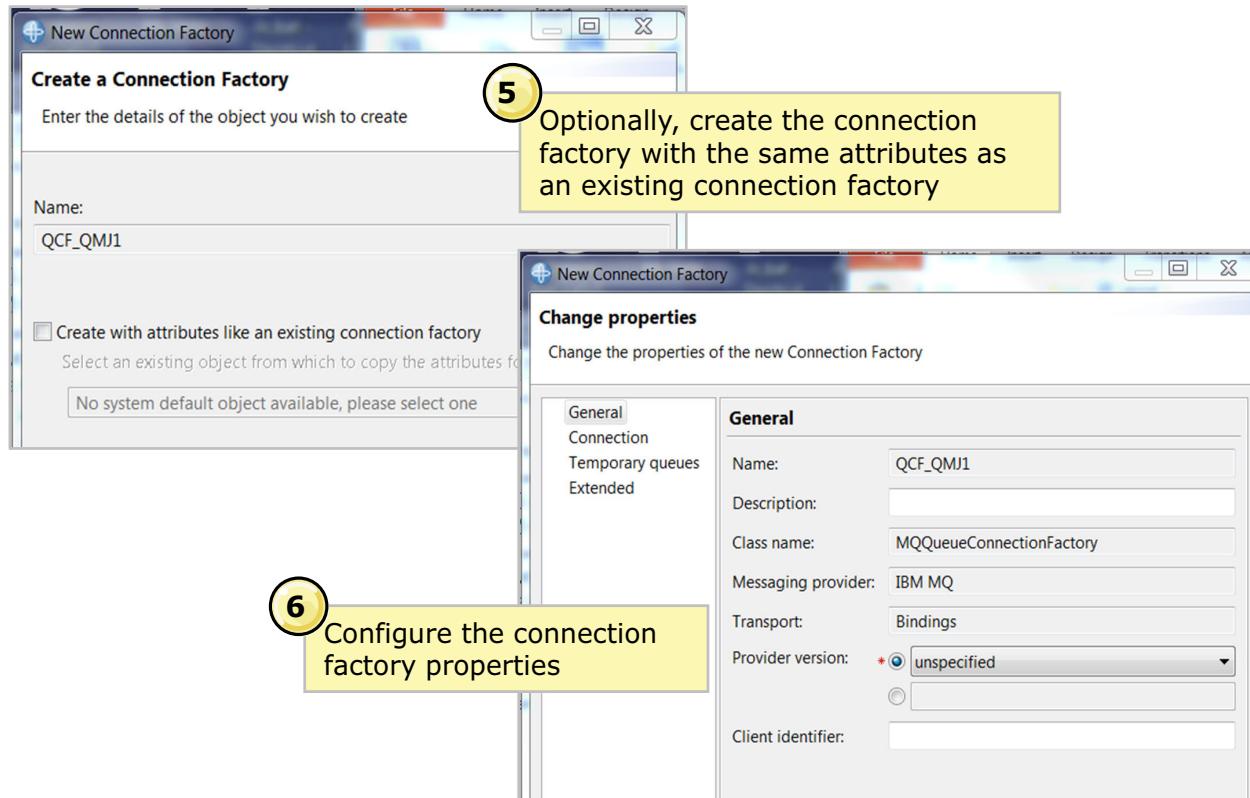
- **Connection Factory:** The JMS application uses both point-to-point messaging and publish/subscribe messaging.
- **Queue Connection Factory:** The JMS application uses only point-to-point messaging.
- **Topic Connection Factory:** The JMS application uses publish/subscribe messaging only.

4. Select the connection factory transport type that is used by the connections that the connection factory creates:

If the JMS client that is using the connection factory is on a different computer from the queue manager, click **MQ Client**. This option means that the connection uses TCP/IP. If you select **MQ Client** and you selected **Support XA transactions** on the previous page of the wizard, you must install the Java Extended Transaction Support component of IBM MQ.

If the JMS application that is using the connection factory is running on the same computer as the queue manager, you can select **MQ Client** or **Bindings**. **Bindings** means that the JMS client connects directly to the queue manager.

## Creating a connection factory (3 of 3)



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

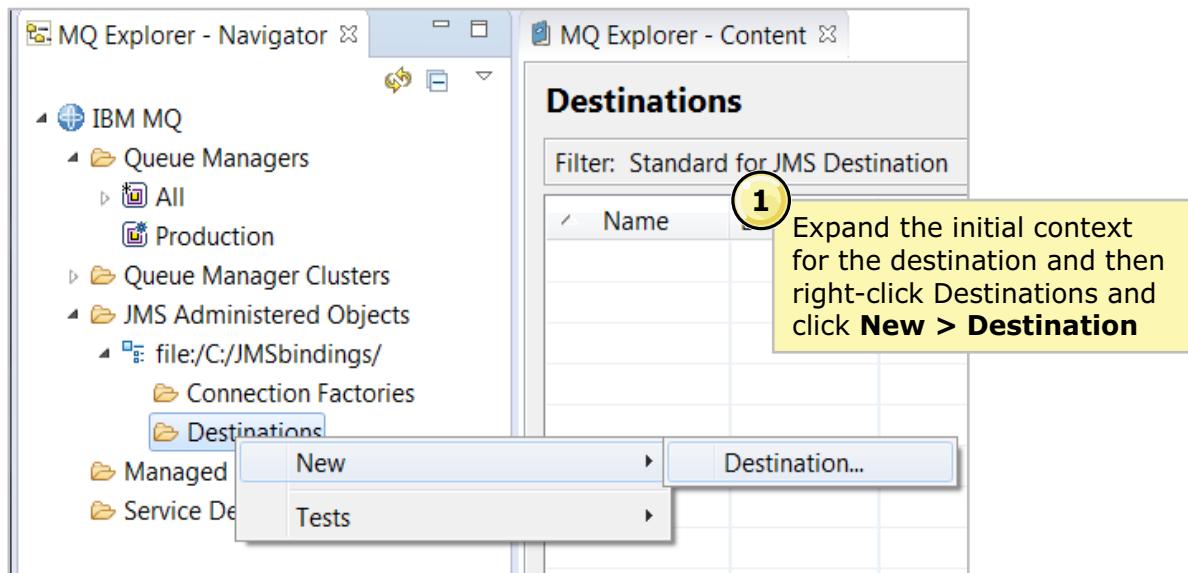
Figure 8-16. Creating a connection factory (3 of 3)

5. If you want to create the connection factory by using the properties of an existing connection factory, click **Create with attributes like an existing connection factory**. Select the connection factory to use for reference.
6. If you did not create the connection factory by using a reference to another connection factory, configure the connection properties.

The available properties depend on the options that you selected on the previous pages. The context help in IBM MQ Explorer contains a detailed description of connection factory properties.



## Creating a JMS destination (1 of 3)



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-17. Creating a JMS destination (1 of 3)

A JMS client uses a destination object to specify both the target of the messages it produces and the source of messages it receives. Destination objects can represent queues (for point-to-point messaging) or topics (for publish/subscribe messaging).

To create a destination in IBM MQ Explorer:

1. Right-click **Destinations** and then click **New > Destination**.

# IBM Training

## Creating a destination (2 of 3)

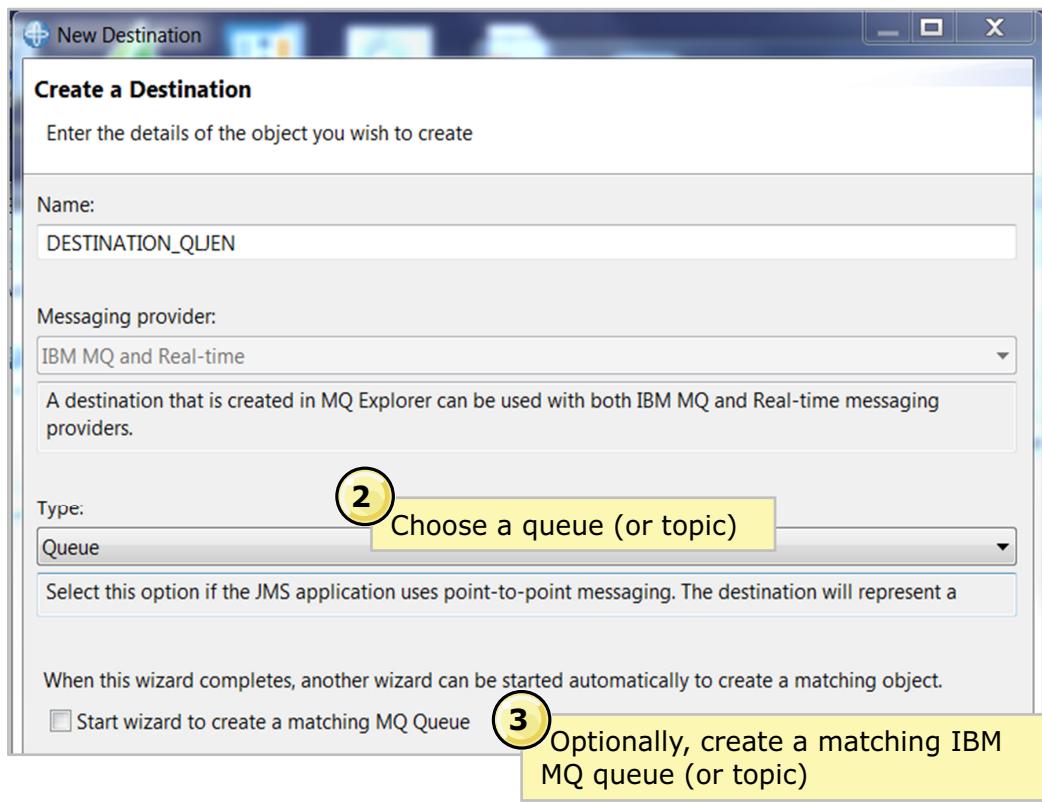
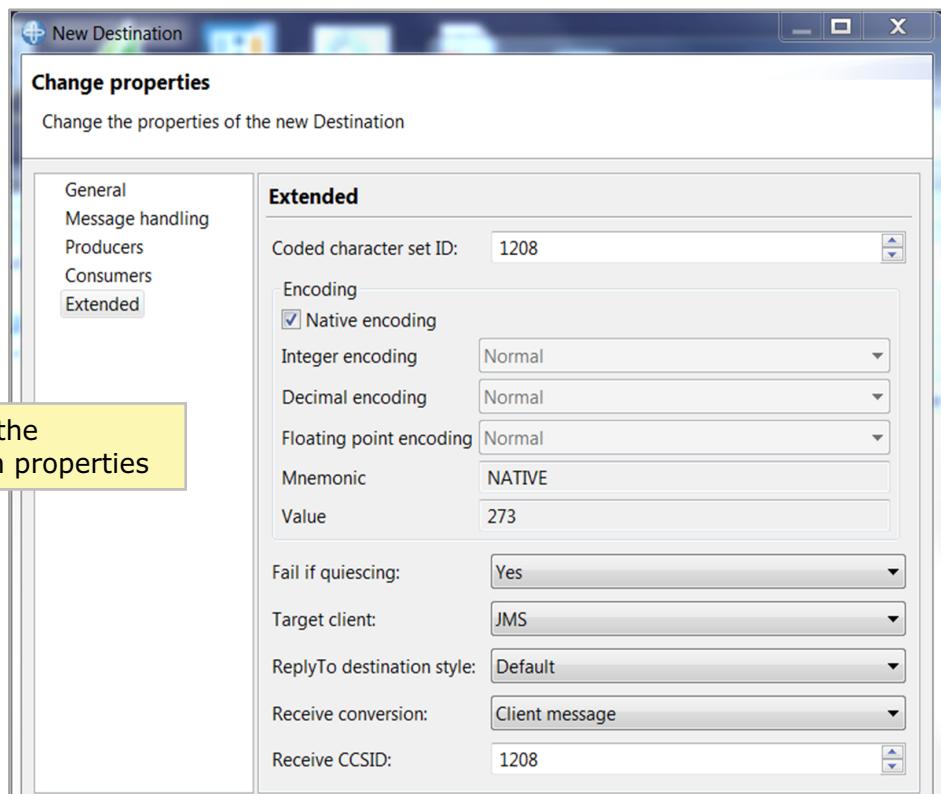


Figure 8-18. Creating a destination (2 of 3)

2. Select the type of destination that you want to create.
  - If you are using point-to-point messaging, click **Queue**.
  - If you are using publish/subscribe messaging, click **Topic**.
3. Optionally, select the option to start a wizard to creating a matching IBM MQ queue or topic.



## Creating a destination (3 of 3)



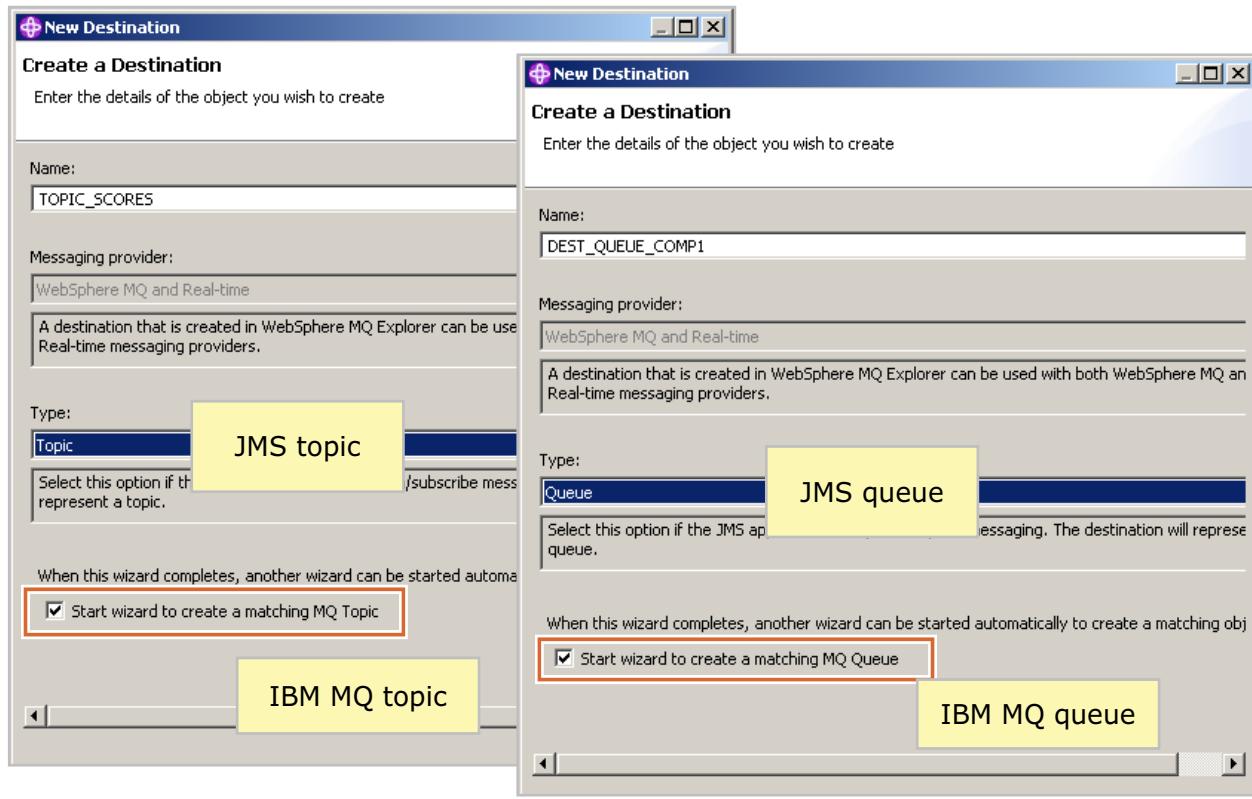
Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-19. Creating a destination (3 of 3)

4. Configure the destination properties. You can use the IBM MQ Explorer context-sensitive help to get a description of each property.

## Mapping between IBM MQ and JMS objects (1 of 2)



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

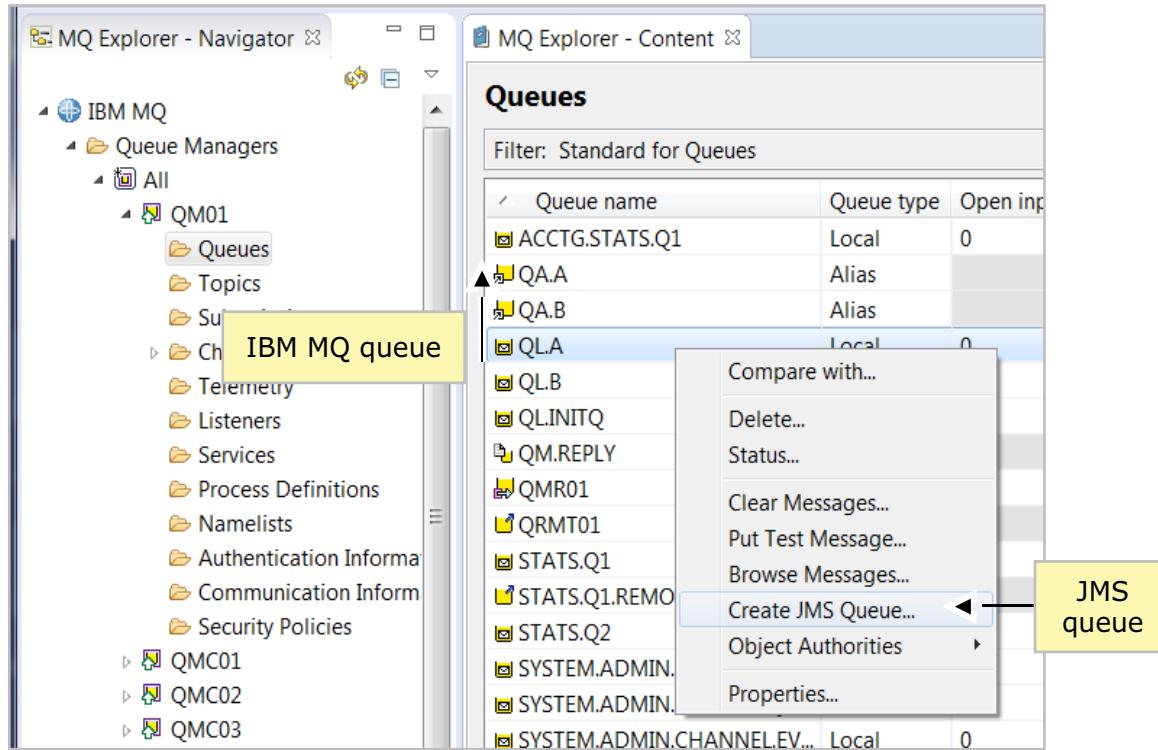
Figure 8-20. Mapping between IBM MQ and JMS objects (1 of 2)

You can map between IBM MQ and JMS objects in IBM MQ Explorer.

You can create the mapped object during the creation of the destination or after you finish creating the destinations. For creating MQ objects, this feature is available in the **New Destination** wizard.

- Create a JMS topic and then create an MQ topic simultaneously when the type **Topic** is selected.
- Create a JMQ queue and then create an MQ queue simultaneously when the type **Queue** is selected.

## Mapping between IBM MQ and JMS objects (2 of 2)



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-21. Mapping between IBM MQ and JMS objects (2 of 2)

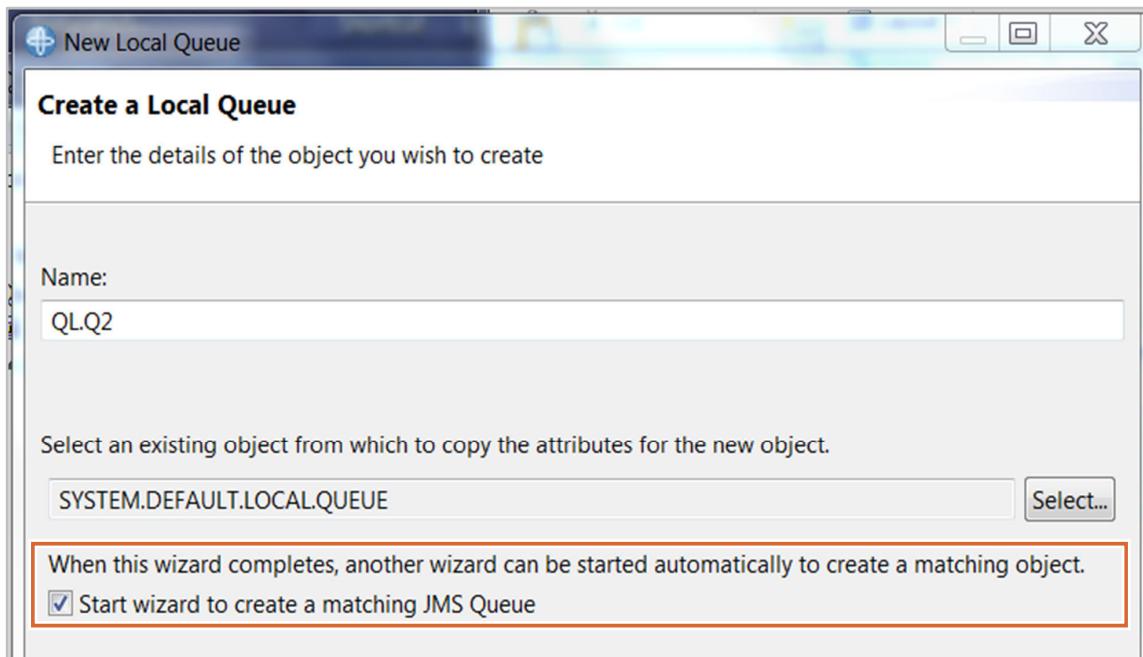
You can create a JMS administered object from an MQ object in IBM MQ Explorer:

1. In the **MQ Explorer - Navigator** view, expand the queue manager that hosts the MQ object (either a queue or topic). Click the **Queues** or **Topics** folder to list the objects in the **Content** view.
2. In the **Content** view, right-click the object, and then click **Create JMS Queue** or **Create JMS Topic**. The **New Destination** wizard opens.
3. In the wizard, click **Select**. Then, select the JMS context in which you want to create the JMS object. The JMS context name is displayed in the **JMS Context** field of the wizard.
4. Step through the wizard to define the new JMS object, then click **Finish**.

The JMS administered object is created and displayed under the appropriate JMS context in the **MQ Explorer - Navigator** view.

# IBM Training

## JMS object creation wizards



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-22. JMS object creation wizards

IBM MQ includes integrated JMS definition wizards to assist with the creation of JMS administered objects for newly created queues and topics and existing queues and topics. The figure shows the options to create a matching JMS queue when you are creating a local queue in IBM MQ Explorer.

## IBM MQ classes for JMS problem determination

- An application can use a provided class to:
  - Start and stop tracing
  - Specify the required level of detail in a trace
  - Customize trace output
- Logging
  - Log file contains messages about errors in plain text
  - An application can use a provided class to specify the location of the log file and its maximum size
- If a failure occurs, an FFST report that contains information that IBM Support can use to diagnose the problem is generated in an FDC file
- An application can use a provided class to query the version of IBM MQ classes for JMS
- Exception messages provide information about the causes of errors and the actions that are required to correct errors

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-23. IBM MQ classes for JMS problem determination

IBM MQ classes for JMS contain a class that an application can use to control tracing. An application can start and stop tracing, specify the required level of detail in a trace, and customize trace output in various ways.

IBM MQ classes for JMS maintain a log file, which contains messages about errors that you need to correct. The messages are written in plain text.

IBM MQ classes for JMS contain a class that an application can use to specify the location of the log file and its maximum size.

If a serious failure occurs, IBM MQ classes for JMS generate an FFST report in an FDC file. The FFST report contains information that IBM Service can use to diagnose the problem.

IBM MQ classes for JMS contain a class that an application can use to query the version of IBM MQ classes for JMS.

Exception messages provide information about the causes of errors and the actions that are required to correct errors.

## Unit summary

- Describe IBM MQ as a JMS provider
- Manage JMS resources in IBM MQ Explorer

Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

*Figure 8-24. Unit summary*

## Review questions

1. Which two objects are JMS administered objects:
  - A. Queue
  - B. Destination
  - C. Collection Factory
  - D. TARGCLIENT
2. True or False: The JNDI namespace can be a database.



Supporting JMS with IBM MQ

© Copyright IBM Corporation 2017

Figure 8-25. Review questions

Write your answers here:

- 1.
- 2.

## Review answers

1. Which two objects are JMS administered objects:

- A. [Queue](#)
- B. [Destination](#)
- C. Collection Factory
- D. TARGCLIENT

The answer is [A](#) and [B](#).



2. [True](#) or False: The JNDI namespace can be a database.

The answer is [True](#).

---

# Unit 9. Introduction to the IBM MQ Console

## Estimated time

01:00

## Overview

The IBM MQ Console is a web-based user interface that can be used to administer IBM MQ. In this unit, you learn how to use the IBM MQ Console to complete common administration tasks.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for IBM MQ V9

## Unit objectives

- List the steps for implementing the IBM MQ Console
- Describe how to use widgets and dashboard layouts to manage IBM MQ objects and authority records, and monitor system resource usage

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

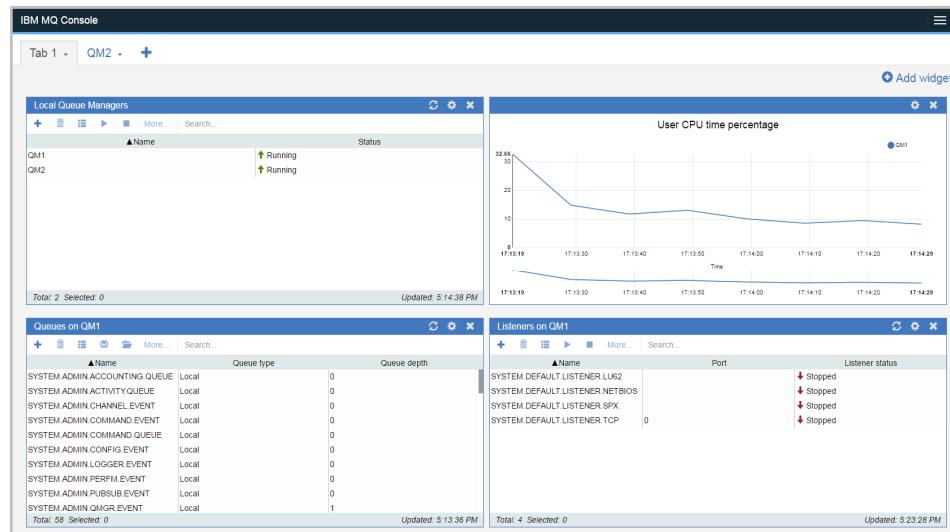
*Figure 9-1. Unit objectives*

## 9.1. IBM MQ Console overview



## The IBM MQ Console

- Web-based administration interface
  - Alternative to IBM MQ Explorer
  - Available from any device with a web browser
  - Consistent with IBM MQ Appliance web-based administration interface
- Added to IBM MQ in Version 9.0.1 Continuous Delivery release



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-2. The IBM MQ Console

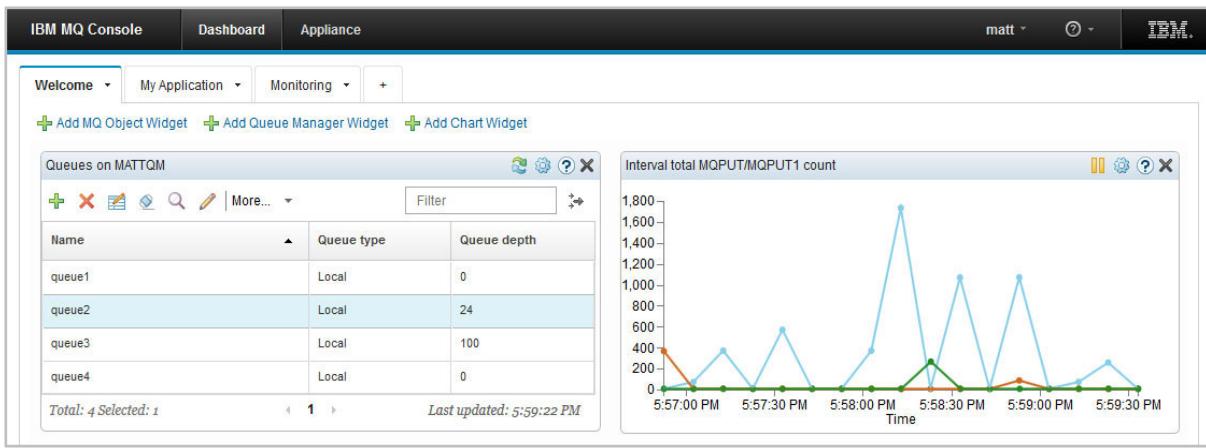
From the IBM MQ Version 9.0.1 Continuous Delivery release, you can use a web-based user interface to administer IBM MQ. The IBM MQ Console runs in a browser and provides control over queue managers and IBM MQ objects.

The IBM MQ Console is similar to the web-based administration interface that administrators can use to manage and monitor the IBM MQ Appliance.



## IBM MQ Console dashboard

- Personalized dashboard of widgets that show queue managers and IBM MQ objects
- Create multiple widgets of the same or different types and position them in the dashboard to fit your requirements
- Import and export dashboards to share with co-workers
- Supports multiple tabs for managing views



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-3. IBM MQ Console dashboard

In the IBM MQ Console, you can add views, referred to as *widgets*, to show and administer IBM MQ objects. You can also add chart widgets to show monitoring statistics for the IBM MQ network.

A dashboard is a container in the IBM MQ Console in which widgets are shown. The dashboard can contain a number of tabs, and each tab can hold a number of widgets, in a grid arrangement. You could, for example, create a tab for each business application that uses MQ, with widgets showing the objects relevant to each application. Or you might have a tab focused purely on monitoring, with charts showing the levels of various MQ components.

You can save a dashboard layout by exporting it from the IBM MQ Console. You can import a saved dashboard layout into the IBM MQ Console.

You can create multiple dashboard tabs to show different selections of information.

## Advantages of the IBM MQ Console

- Avoids installing and maintaining software packages locally
- Available across a wide variety of operating systems and devices
- Offers both configuration and monitoring capabilities
- Provides a customized experience for each user



The current version of the IBM MQ Console does not provide all of the configuration functions of IBM MQ Explorer

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-4. Advantages of the IBM MQ Console

The IBM MQ Console provides some advantages over the IBM MQ Explorer.

The IBM MQ Console requires a web-browser only.

With the IBM MQ Console, you can add a Charts widget to your dashboard and then configure it to monitor a particular aspect of resource usage in a graphical representation instead of a text file that might be difficult to interpret.



### Important

The current version of the IBM MQ Console does not support all the configuration options that IBM MQ Explorer and MQSC support. It might be necessary to use the IBM MQ Explorer and MQSC to complete some administration tasks.

## Administration with the IBM MQ Console

- Configure queues, topics, listeners, channels, authentication information, and subscriptions
  - View, create, and delete
  - Modify some properties
- Create, delete, start, stop, and view status of queue managers
- Put, get, and browse messages
- Create, start, stop, and ping channels

Name	Running TCP listener ports	Status	High Availability
AWebUIQM	1616	Running	
AWebUIQM2	1666	Running	
MATTQM		Stopped	

Total: 3 Selected: 1      Last updated: 6:11:13 PM

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-5. Administration with the IBM MQ Console

In the IBM MQ Console, authorized users can configure and view the status of queues, topics, listeners, authentication information, and subscriptions.

You can use the local queue manager widget in the IBM MQ Console to create, configure, and control local queue managers.

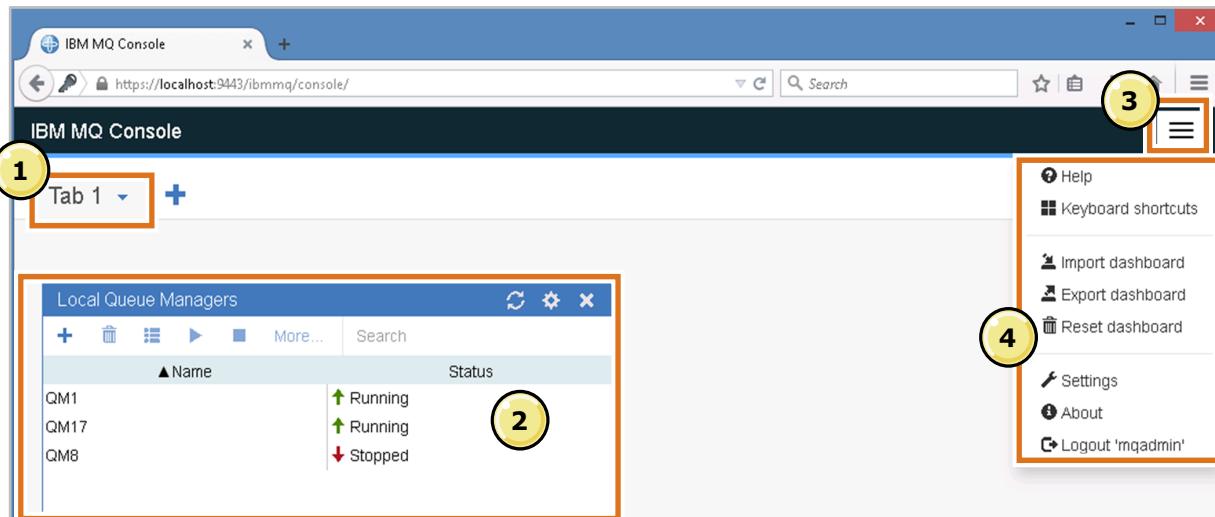
You can use the queues widget in the IBM MQ Console to create and delete local, remote, and alias queues. You can also use the queues widget to put, get, and browse messages.

You use the channels widget in the IBM MQ Console to create and delete queue manager channels and client connection channels. You can also use the channels widget to start, stop, and ping channels.



## Dashboard components

1. Default dashboard includes a single tab
2. Example widget is included in the initial dashboard by default
3. Click **Dashboard Settings** icon to display Dashboard menu
4. Dashboard menu provides access to utility and help functions



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-6. Dashboard components

Each dashboard contains the following components:

- One or more tabs
- On each tab, one or more widgets that display IBM MQ objects or monitoring charts
- A dashboard menu

The default dashboard layout includes a single tab. You can rename the tab to make it more descriptive. You can also add tabs to a dashboard.

The default dashboard includes a Local Queue Managers widget. You can add more widgets to the dashboard as required.

Click the **Dashboard Settings** icon to display the Dashboard menu.

From the Dashboard menu you can view the keyboard shortcuts, enable IBM MQ Console tracing, import and export dashboards, access the product documentation, view information about the IBM MQ Console, and log out of the IBM MQ Console.

## IBM MQ Console security

- User can be configured to use the IBM MQ authorization model when authenticating actions against a queue manager
- Read-only user can access IBM MQ Console to view but cannot delete, create, stop, or change configuration

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

*Figure 9-7. IBM MQ Console security*

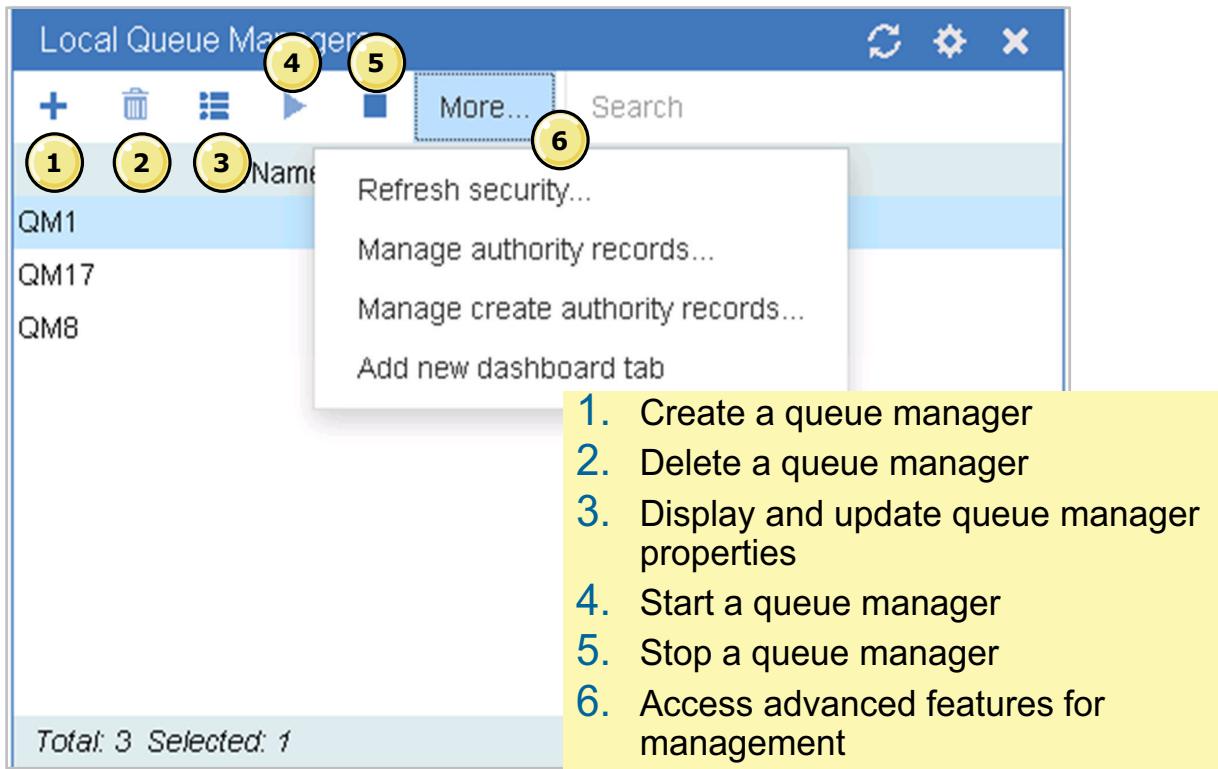
To use of the IBM MQ Console, users need to authenticate against a user registry that is defined to the IBM MQ Console web server.

Authenticated users must be a member of one of the groups that authorizes access to the capabilities of the IBM MQ Console.

Users with read-only access can access the IBM MQ Console to view IBM MQ objects and charts. Users with read-only access cannot change the configuration of IBM MQ objects.



## Managing queue managers



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-8. Managing queue managers

You can use a queue manager widget to manage queue managers.

Click the icons on the queue manager widget to create and delete queue managers, display and update queue manager properties, and start and stop a queue manager.

You can use the option on the **More** menu to refresh security, manage authority records, and add new dashboard tabs.

The local queue manager widget lists the local queue managers that are added to the IBM MQ installation from which the IBM MQ Console is running. Queue managers that are associated with different installations of IBM MQ on the same system are not listed. You can select individual queue managers from the list to work with.



## Queue manager status

Properties for 'MQCONLAB'

General	<b>Queue manager name:</b>	MQCONLAB
Extended	<b>Queue manager status:</b>	Running
Cluster	<b>Command server status:</b>	Running
Repository	<b>Channel initiator status:</b>	Running
Communication	<b>Connection count:</b>	27
Events	<b>Current log extent name:</b>	
SSL	<b>Restart recovery log extent name:</b>	
Statistics	<b>Media recovery log extent name:</b>	
Online monitoring	<b>LDAP connection status:</b>	Inactive
Statistics monitoring	<b>Log path:</b>	C:\ProgramData\IBM\MQ\log\MQCONLAB\activ
Accounting monitoring	<b>Standby:</b>	Not permitted
Publish/Subscribe	<b>Start date:</b>	April 20, 2017
Status	<b>Start time:</b>	7:29:26 AM
	<b>Installation name:</b>	Installation1
	<b>Installation description:</b>	
	<b>Installation path:</b>	C:\Program Files\IBM\MQ

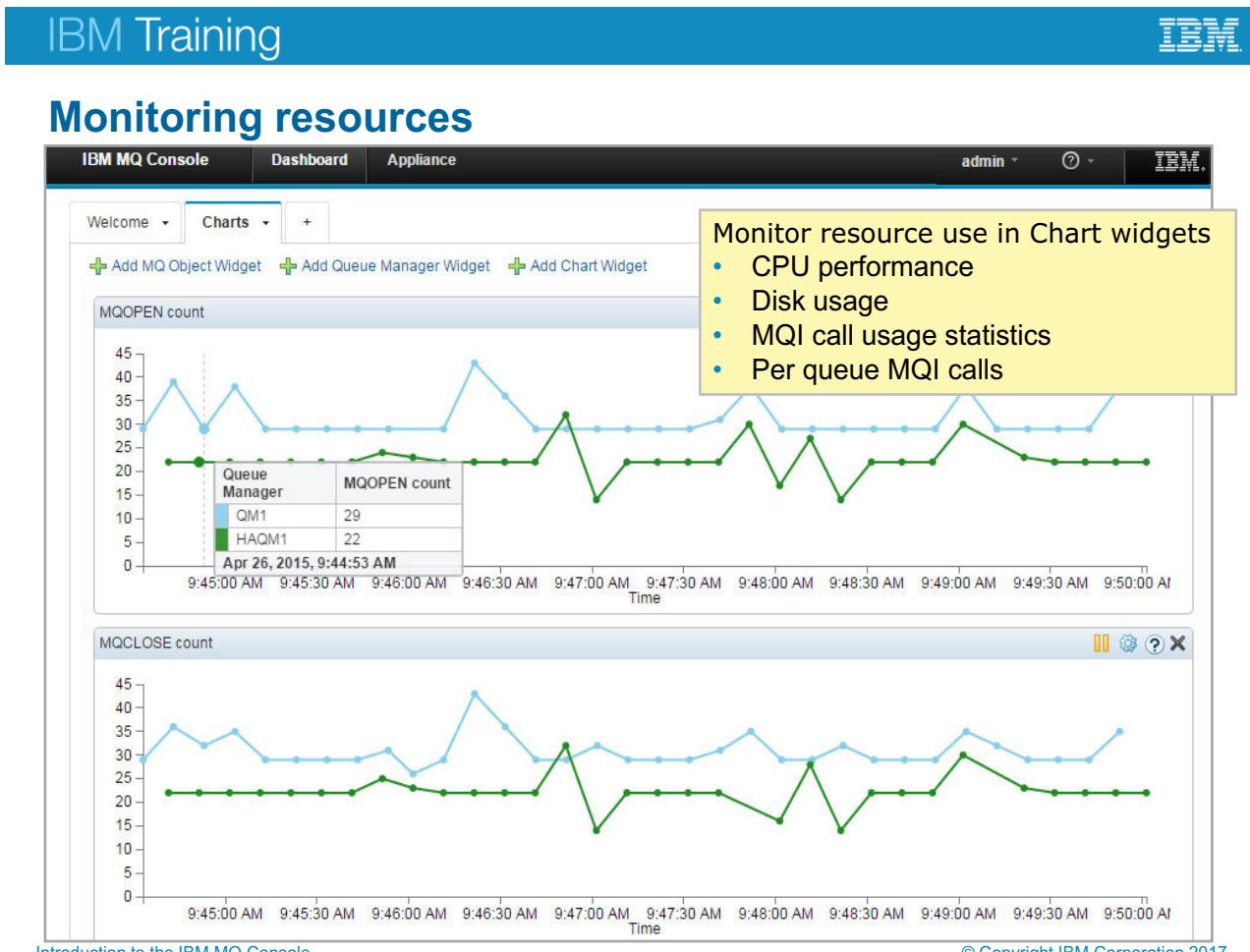
**Status** properties page shows the status of queue manager

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-9. Queue manager status

You can view the queue manager status on the **Status** page of the queue manager properties.



*Figure 9-10. Monitoring resources*

You can use an IBM MQ Console chart widget to view monitoring data for a queue manager.

There are three types of resources that can be monitored:

- **Platform central processing** monitors CPU usage.
- **Platform persistent data stores** shows the use of disk resources.
- **API usage statistics** monitors MQI calls. It shows the same type of monitoring statistics as available with IBM MQ Explorer or by using MQSC options.

You can add chart widgets that can show multiple queue managers.

As shown in the example, you can hover the cursor over a collection point on the graph to display the values at that collection point.

## IBM MQ Console and the administrative REST API

- From IBM MQ Version 9.0.1, you can use the administrative REST API to request information about queue managers and installations
  - Call HTTP methods on URLs that represent the various IBM MQ objects, such as queue managers or queues
  - Information is sent and received in the JSON format
- Step 1: Construct the URL

Example:

To request all attributes of queue manager QM1, create the following URL:

```
https://localhost:9443/ibmmq/rest/v1/qmgr/QM1?attributes=*
```

- Step 2: Call the relevant HTTP method on the URL
  - Specify any optional JSON payload
  - Provide security credentials to authenticate

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-11. IBM MQ Console and the administrative REST API

From IBM MQ Version 9.0.1, you can use the administrative REST API to request information about queue managers and installations.

You can use the administrative REST API to administer IBM MQ objects, such as queue managers and queues. Information is sent to, and received from, the administrative REST API in the JSON format.

In IBM MQ Version 9.0.1, the administrative REST API is not integrated with IBM MQ security. The administrative REST API is disabled by default. You must manually enable the administrative REST API before you can use it.

From IBM MQ Version 9.0.2, the administrative REST API is integrated with IBM MQ security and the administrative REST API is enabled by default. However, you must configure security before you can use the administrative REST API.

When you use the administrative REST API, you invoke HTTP methods on URLs that represent the various IBM MQ objects, such as queue managers or queues. The HTTP method, for example POST, represents the type of action to be performed on the object that is represented by the URL.

## 9.2. IBM MQ Console setup

## Getting started

1. Install IBM MQ Console component
  - On Linux and UNIX, install the **MQSeriesWeb** component
  - On Windows, install the **Web Administration** feature
2. Configure basic security for users and groups to access the IBM MQ Console
3. (Optional) Enable remote connections to the IBM MQ Console web server
4. As a privileged user, start the IBM MQ Console web server
5. Connect to the IBM MQ Console from a web browser
6. Log in to IBM MQ Console

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-12. Getting started

This figure summarizes the steps for setting up and running the IBM MQ Console.

Before you can start the IBM MQ Console you must install the correct components, configure security, and start the IBM MQ Console web server.

On Linux and UNIX, install the **MQSeriesWeb** component.

On Windows, install the **Web Administration** feature.

## Configuring users and roles

- Users must authenticate against a user registry that is defined to the IBM MQ Console web server
- By default, the user registry does not contain any users
  - Add users by editing the **mqwebuser.xml** file
- Copy registry file to  
*MQDATA\_PATH/web/installations/installationName/servers/mqweb*

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-13. Configuring users and roles

You must configure basic security to allow users and groups to access the IBM MQ Console.

First, copy one of the sample registry files from the

*MQ\_INSTALLATION\_PATH/web/mq/samp/configuration* directory to the IBM MQ Console web server directory for the installation. On UNIX, Linux, and Windows the installation directory is *MQ\_DATA\_DIRECTORY/web/installations/installationName/servers/mqweb*. Then, rename the sample XML file to **mqwebuser.xml**.

## Sample user registry files

- Can use sample registry files in *MQINSTALLATION\_PATH/web/mq/samp/configuration* to authenticate and authorize users of the IBM MQ Console
  - **no\_security.xml** disables security, which includes the ability to access the IBM MQ Console by using HTTPS
  - **basic\_registry.xml** defines a basic registry of users and groups
  - **ldap\_registry.xml** defines a connection to an LDAP registry from which user and group information is retrieved
- Replace contents of **mqwebuser.xml** file in *MQDATA\_PATH/web/installations/installationName/servers/mqweb*

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-14. Sample user registry files

The IBM MQ installation includes three sample registry files for the IBM MQ Console.

You can use one of the sample files or customize one of the sample files. Before you start the IBM MQ web server, replace the `mqwebuser.xml` file in the IBM MQ Console web server installation directory with your registry file.

## IBM MQ Console roles

- **MQWebAdmin** user or group can do all operations, and operates under the security context of the operating system user ID that started the IBM MQ Console web server
- **MQWebAdminRO** gives read only access to display and inquire operations on IBM MQ objects such as queues and channels and browse messages on queues
- **MQWebUser** can start, stop, define, set, display, and inquire IBM MQ objects

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-15. IBM MQ Console roles

This figure summarizes the IBM MQ Console roles.

## Sample basic\_registry.xml file (1 of 2)

```

<enterpriseApplication id="com.ibm.mq.console">
    <application-bnd>
        <security-role name="MQWebAdmin">
            <group name="MQWebUI" realm="defaultRealm"/>
        </security-role>
        <security-role name="MQWebAdminRO">
            <user name="reader" realm="defaultRealm"/>
        </security-role>
        <security-role name="MQWebUser">
            <user name="guest" realm="defaultRealm"/>
        </security-role>
    </application-bnd>
</enterpriseApplication>

```

Group **MQWebUI** is granted access with the role **MQWebAdmin**

User **reader** is granted access with the role **MQWebAdminRO**

User **guest** is granted access with the role **MQWebUser**

[Introduction to the IBM MQ Console](#)

© Copyright IBM Corporation 2017

Figure 9-16. Sample basic\_registry.xml file (1 of 2)

The `basic_registry.xml` file can be used to get started with IBM MQ Console.

The `security-role` sections define the users and groups that are members of each IBM MQ Console role.

## Sample basic\_registry.xml file (2 of 2)

```
<basicRegistry id="basic" realm="defaultRealm">

    <!-- This sample defines two users with unencoded passwords -->
    <!-- and a group, these are used by the role mappings above -->

    <user name="mqadmin" password="mqadmin"/>

    <user name="mqreader" password="mqreader"/>

    <group name="MQWebUI">
        <member name="mqadmin"/>
    </group>

</basicRegistry>
```

**basicRegistry** section  
defines users and groups

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-17. Sample basic\_registry.xml file (2 of 2)

The `basicRegistry` section in the `basic_registry.xml` file defines the users and the groups that are referenced in the security-roles sections.

The `basic_registry.xml` file defines the following users and groups:

- An administrative user that is named “mqadmin” with a password of “mqadmin”
- A read-only user that is named “mqreader” with a password of “mqreader”

The `basicRegistry` section defines the user “mqadmin” as a member of the **MQWebUI** group. In the security-role that was shown on the previous page, the **MQWebUI** group is defined with a security role of **MQWebAdmin**.

## Commands to control IBM MQ Console web server

- Start the IBM MQ Console web server: `strmqweb`
- Stop IBM MQ Console web server: `endmqweb`
- Display the status of IBM MQ Console web server and the URL: `dspmqweb`

Example output:

```
Server mqweb is running.  
URLs:  
https://localhost:9443/ibmmq/console/
```

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-18. Commands to control IBM MQ Console web server

To start the IBM MQ Console web server, type `strmqweb`. The user that starts the IBM MQ Console web server must be an IBM MQ privileged user. A privileged user is a user with full administrative authorities for IBM MQ.

You can stop the IBM MQ Console web server by typing `endmqweb`.

The IBM MQ Console web server must be running to use the IBM MQ Console or the administrative REST API. You can use the `dspmqweb` command to view the status of the IBM MQ Console web server. If the server is running, then the URLs that are used by the IBM MQ Console and administrative REST API are displayed.

## Working with queue managers

- Local queue manager widget automatically lists the local queue managers that are added to the IBM MQ installation from which the IBM MQ Console is running
- Create, start, stop, and delete local queue managers
- View properties of a local queue manager
- Refresh security on a local queue manager

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

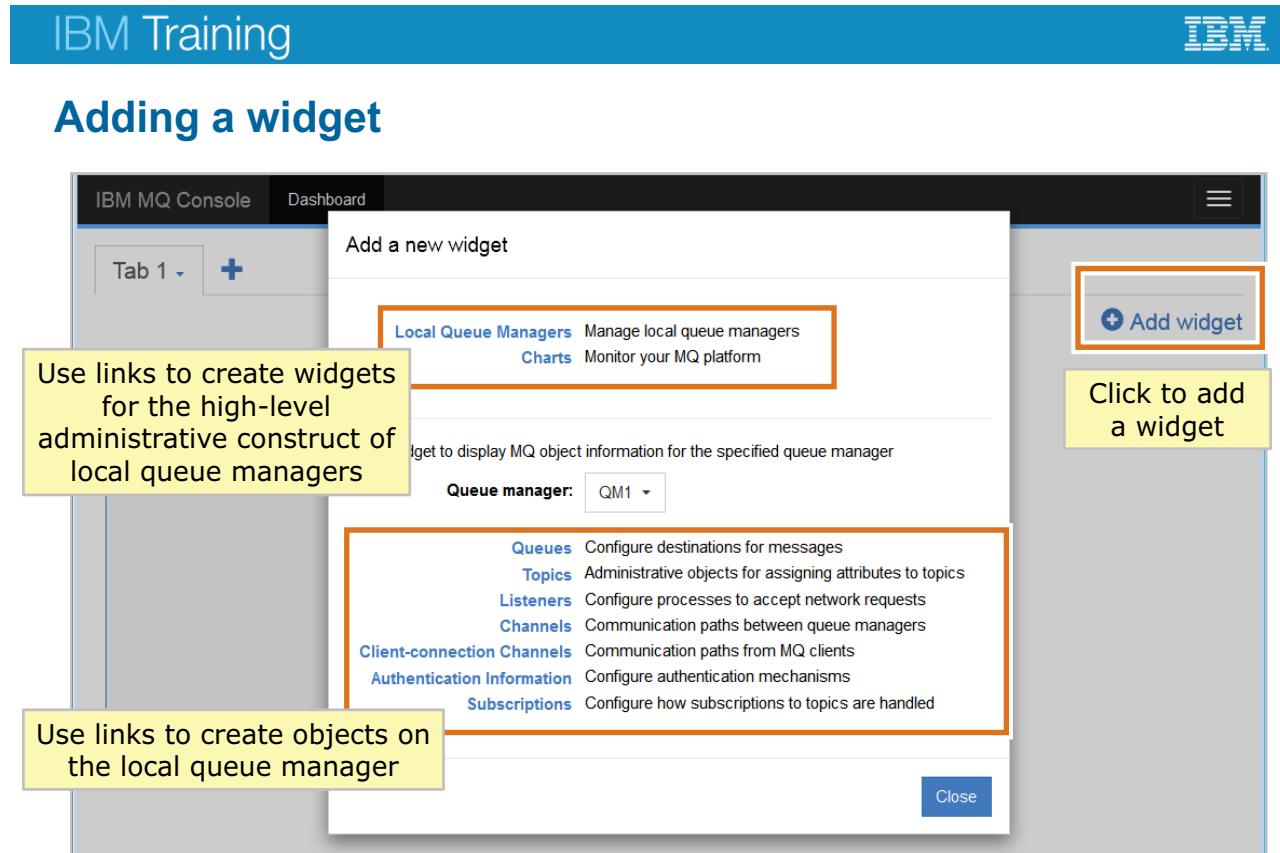
Figure 9-19. Working with queue managers

You can use a queue manager widget to manage queue managers.

In the queue managers to create and delete queue managers, display and update queue manager properties, and start and stop a queue manager.

You can use the option on the **More** menu to refresh security, manage authority records, and add new dashboard tabs.

The local queue manager widget lists the local queue managers that are added to the IBM MQ installation from which the IBM MQ Console is running. Queue managers that are associated with different installations of IBM MQ on the same system are not listed. You can select individual queue managers from the list to work with.



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-20. Adding a widget

Each IBM MQ object widget contains objects that are associated with a specific queue manager. You can add the following types of IBM MQ object widgets to your dashboard:

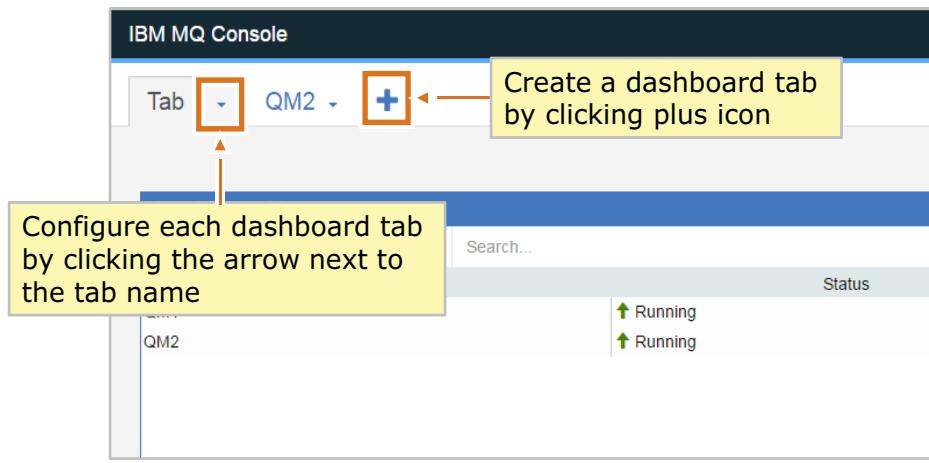
- Queues
- Topics
- Listeners
- Channels
- Client-connection channels
- Authentication information
- Subscriptions

To create an IBM MQ object widget:

1. Click **Add widget**.
2. Select the appropriate queue manager from the list.
3. Click the name of the type of object widget that you want to create.

## Configuring the dashboard layout

- Create and delete dashboard tabs
- Configure dashboard tabs
  - Change the tab name
  - Add a description
  - Configure the number of columns
- Configure the layout of the widgets within a dashboard tab by dragging and dropping the widgets



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-21. Configuring the dashboard layout

A dashboard is a container in the IBM MQ Console in which widgets are shown.

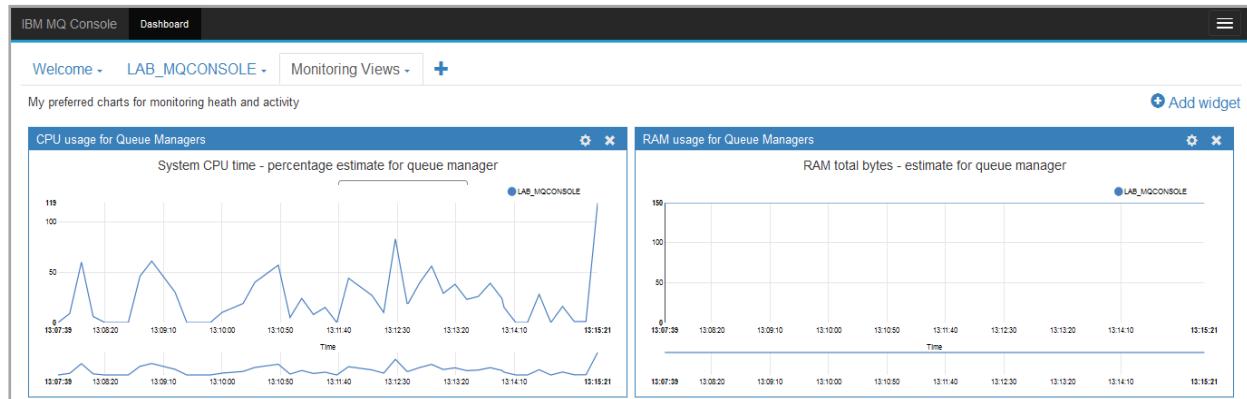
You can configure each dashboard tab by clicking the arrow next to the tab name. You can change the tab name, and add a description for the tab. You can also configure the number of columns that the tab uses.

You can configure the layout of the widgets within a dashboard tab by dragging and dropping the widgets.



## Monitoring system resource usage

- Configure Chart widget to monitor a particular aspect of resource usage
  - Can create many instances of the Chart widget to display different data
  - Chart X-axis displays the timeline
  - Chart Y-axis displays units appropriate to the resource and is dynamically resized to accommodate the data that is returned



- For complete list of resource monitoring classes and resource types, see "Monitoring system resource usage" in IBM Knowledge Center:  
[https://www.ibm.com/support/knowledgecenter/SSFKSJ\\_9.0.0/com.ibm.mq.mqc.doc/q127750.htm](https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.mqc.doc/q127750.htm)

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

*Figure 9-22. Monitoring system resource usage*

You use the Charts widget in the IBM MQ Console to view monitoring data for queue managers.

You add a Charts widget to your dashboard and then configure it to monitor a particular aspect of resource usage. You can create many instances of the Charts widget to display different data.

The data is displayed in a chart format.

- The X-axis of the chart displays a timeline.
- The Y-axis displays units appropriate to the resource that you are viewing. The Y-axis is dynamically resized to accommodate the data that is returned.

Data is collected at 10-second intervals.

You must have at least one queue manager that is running before you can configure a chart widget.

## More information on the IBM MQ Console

- IBM developerWorks MQdev Blog
  - “The IBM MQ Console – a first look”  
[https://www.ibm.com/developerworks/community/blogs/messaging/entry/ibm\\_mq\\_console?lang=en](https://www.ibm.com/developerworks/community/blogs/messaging/entry/ibm_mq_console?lang=en)
  - “What’s new for the IBM MQ Console in 9.0.1”  
[https://www.ibm.com/developerworks/community/blogs/messaging/entry/Whats\\_new\\_for\\_the\\_IBM\\_MQ\\_Console\\_in\\_9\\_0\\_1?lang=en](https://www.ibm.com/developerworks/community/blogs/messaging/entry/Whats_new_for_the_IBM_MQ_Console_in_9_0_1?lang=en)
- YouTube Videos
  - Introduction to IBM MQ Console:  
<https://www.youtube.com/watch?v=DyoUnWbt5ek>

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

*Figure 9-23. More information on the IBM MQ Console*

The IBM MQ Console is described in the IBM Knowledge Center for IBM MQ.

This figure provides additional references for more information about the IBM MQ Console.

## Unit summary

- List the steps for implementing the IBM MQ Console
- Describe how to use widgets and dashboard layouts to manage IBM MQ objects and authority records, and monitor system resource usage

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

*Figure 9-24. Unit summary*

## Review questions

1. True or False: The IBM MQ Console supports all the configuration functions that the IBM MQ Explorer supports.
2. True or False: A member of the **MQWebUser** security group can start, stop, define, set, display, and inquire IBM MQ objects.



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-25. Review questions

Write your answers here:

- 1.
- 2.

## Review answers

1. True or False: The IBM MQ Console supports all the configuration functions that the IBM MQ Explorer supports.  
**The answer is False.** You might need to supplement the IBM MQ Console with the IBM MQ Explorer or MQSC to complete some tasks.
  
2. True or False: A member of the **MQWebUser** security group can start, stop, define, set, display, and inquire IBM MQ objects.  
**The answer is True.**



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

Figure 9-26. Review answers

## Exercise: Getting started with the IBM MQ Console

Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

*Figure 9-27. Exercise: Getting started with the IBM MQ Console*

In this exercise, you set up and use the IBM MQ Console for basic administration of IBM MQ objects. You also monitor system resources and configure dashboard layouts.

## Exercise objectives

- Configure basic security to allow users and groups to access the IBM MQ Console
- Start the IBM MQ Console
- Manage local queue managers
- Monitor system resources
- Configure dashboard layouts



Introduction to the IBM MQ Console

© Copyright IBM Corporation 2017

*Figure 9-28. Exercise objectives*

For detailed exercise instructions, see the Course Exercises Guide.

---

# Unit 10. Course summary

## Estimated time

00:30

## Overview

This unit summarizes the course and provides information for future study.

## Unit objectives

- 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

[Course summary](#)

© Copyright IBM Corporation 2017

*Figure 10-1. Unit objectives*

## Course objectives

- Use conversation sharing, read-ahead, and asynchronous put to improve the performance of MQI client connections
- Use Transport Layer Security (TLS) to secure TCP/IP channels
- Authenticate IBM MQ channels, connections, and users
- Manage the workload in an IBM MQ queue manager cluster
- Implement IBM MQ high availability
- Monitor application activity, events, and messages
- Use the IBM MQ dead-letter queue message handler to manage a dead-letter queue
- Administer distributed publish/subscribe networks
- Use the IBM MQ Console to administer IBM MQ objects and resource usage
- Administer Java Message Service (JMS) in MQ

[Course summary](#)

© Copyright IBM Corporation 2017

*Figure 10-2. Course objectives*

## To learn more on the subject

- IBM Training website:  
[www.ibm.com/training](http://www.ibm.com/training)
- IBM Knowledge Center for IBM MQ V9.0:  
[www.ibm.com/support/knowledgecenter/SSFKSJ\\_9.0.0/com.ibm.mq.helphome.v90.doc/WelcomePagev9r0.htm](http://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.helphome.v90.doc/WelcomePagev9r0.htm)

Course summary

© Copyright IBM Corporation 2017

Figure 10-3. To learn more on the subject

## Enhance your learning with IBM resources

*Keep your IBM Cloud skills up-to-date*

- IBM offers resources for:
  - Product information
  - Training and certification
  - Documentation
  - Support
  - Technical information



- To learn more, see the IBM Cloud Education Resource Guide:
  - [www.ibm.biz/CloudEduResources](http://www.ibm.biz/CloudEduResources)

Course summary

© Copyright IBM Corporation 2017

Figure 10-4. Enhance your learning with IBM resources

## Unit summary

- 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

[Course summary](#)

© Copyright IBM Corporation 2017

*Figure 10-5. Unit summary*

## Course completion

**You have completed this course:**

IBM MQ V9 Advanced System Administration (Distributed)

**Any questions?**



[Course summary](#)

© Copyright IBM Corporation 2016

*Figure 10-6. Course completion*

# Appendix A. List of abbreviations

<b>AES</b>	Advanced encryption standard
<b>AIA</b>	AuthorityInfoAccess
<b>API</b>	Application programming interface
<b>ARL</b>	Authority revocation list
<b>ASCII</b>	American Standard Code for Information Interchange
<b>BER</b>	Basic encoding rules
<b>CA</b>	Certificate authority
<b>CC</b>	completion code
<b>CCDT</b>	Channel definition table
<b>CER</b>	Canonical encoding rules
<b>CN</b>	Common name
<b>CLR</b>	Compensation Log Records
<b>CPU</b>	Central processing unit
<b>CRL</b>	Certificate revocation list
<b>DER</b>	Distinguished encoding rules
<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>DLQ</b>	dead-letter queue
<b>DN</b>	Distinguished name
<b>DNS</b>	Domain Name Server
<b>EBCDIC</b>	Extended binary coded decimal interchange code
<b>EJB</b>	Enterprise Java Bean
<b>FDC</b>	Failure Data Capture
<b>FFST</b>	First Failure Support Technology
<b>FIPS</b>	Federal Information Processing Standards
<b>HA</b>	High availability
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IBM</b>	International Business Machines Corporation
<b>I/O</b>	input/output
<b>IP</b>	Internet Protocol
<b>IPC</b>	Interprocess communication

<b>Java EE</b>	Java Platform, Enterprise Edition
<b>JCEKS</b>	Java cryptography extended key store
<b>JKS</b>	Java key store
<b>JMS</b>	Java Message Service
<b>JNDI</b>	Java Naming and Directory Interface
<b>LDAP</b>	Lightweight Directory Access Protocol
<b>MAC</b>	Message Authentication Code
<b>MDB</b>	message driven bean
<b>MQI</b>	IBM MQ Message queue interface
<b>MQMD</b>	MQ message descriptor
<b>MQSC</b>	IBM MQ script command
<b>MRU</b>	Most recently used
<b>MSCS</b>	Microsoft Cluster Service
<b>NAS</b>	network-attached storage
<b>OAM</b>	Object Authority Manager
<b>OCSP</b>	Online Certificate Status Protocol
<b>OU</b>	Organization unit
<b>PCF</b>	Programmable command format
<b>PEM</b>	Privacy Enhanced Mail
<b>PIN</b>	Personal identification number
<b>PKCS</b>	Public Key Cryptography Standards
<b>PKI</b>	Public Key Infrastructure
<b>RC</b>	reason code
<b>RAID</b>	Redundant array of independent disks
<b>RAM</b>	random access memory
<b>REST</b>	Representational State Transfer
<b>SAN</b>	storage area network
<b>SIB</b>	Service Integration Bus
<b>SLA</b>	Service level agreements
<b>SNI</b>	Server name indication
<b>SSL</b>	Secure socket layer
<b>TCP</b>	Transmission Control Protocol
<b>TLS</b>	Transport layer security
<b>URL</b>	Uniform Resource Locator



IBM Training



© Copyright International Business Machines Corporation 2017.