

Course Guide

Designing, Implementing, and Managing IBM MQ V9 Clusters

Course code WM253 ERC 1.0



June 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.

© 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

Trademarks	ix
Course description.....	x
Agenda	xii
Unit 1. IBM MQ baseline.....	1-1
How to check online for course material updates	1-2
Unit objectives	1-3
IBM MQ components and baseline	1-4
Queue manager	1-5
Messages	1-6
Queues	1-7
QLOCAL partial attributes and defaults	1-9
Some local queues are designated for a special purpose	1-10
Queue types used for different cluster routing scenarios	1-11
Basic Message Queue Interface (MQI) calls	1-12
Channels	1-13
IBM MQ clients	1-14
Using the transmission queue to trigger start a channel	1-16
Queue name resolution	1-17
Sender-receiver channel pair without local remote queue	1-18
Sender-receiver channel pair with local remote queue	1-19
Where to look for errors	1-20
One-way objects with remote queue definition checkpoint	1-21
Typical queue manager point-to-point channel definitions	1-22
IBM MQ clusters	1-23
Messaging styles	1-25
What are shared queues?	1-26
Unit summary	1-28
Review questions	1-29
Review answers (1 of 2)	1-30
Review answers (2 of 2)	1-31
Exercise: Configuring and reviewing base IBM MQ resources	1-32
Exercise objectives	1-33
Unit 2. Before you start	2-1
Unit objectives	2-2
Everyone impacts the cluster	2-3
IBM MQ administrative tasks	2-5
Before you start	2-7
Distributed queuing versus clustered configuration (1 of 3)	2-8
Distributed queuing versus clustered configuration (2 of 3)	2-9
Distributed queuing versus clustered configuration (3 of 3)	2-10
IBM MQ objectives for highly available architectures	2-12
Applications and workload balancing	2-14
IBM MQ administration settings and workload balancing	2-16
Clusters and IBM MQ administration	2-17
Unit summary	2-18
Review questions	2-19

Review answers (1 of 2)	2-20
Review answers (2 of 2)	2-21
Unit 3. Understanding and implementing an IBM MQ cluster	3-1
Unit objectives	3-2
IBM MQ clusters	3-3
IBM MQ cluster components	3-4
The SYSTEM.CLUSTER queues	3-6
How long is cluster information kept in a repository?	3-8
Basic queue manager cluster	3-10
Queue manager cluster definition	3-11
MQ0A definition check: What happens in the cluster (1 of 2)	3-12
MQ0A definition check: What happens in the cluster (2 of 2)	3-14
MQ00 definition check: What happens in the cluster (1 of 2)	3-15
MQ00 definition check: What happens in the cluster (2 of 2)	3-16
MQ03 definition check: Errors, TMPQMGR, TMPUUID entries	3-17
MQ03 definition check: Log messages	3-19
Cluster queues	3-21
Cluster transmission queue options	3-23
Message trajectory in a basic cluster	3-25
Routing messages to and from clusters	3-27
Workload balancing from outside a cluster: Cluster gateway	3-28
Definitions to MQ00 gateway from an external queue manager	3-30
Cluster administrative options	3-31
Unit summary	3-32
Review questions (1 of 2)	3-33
Review questions (2 of 2)	3-34
Review answers (1 of 2)	3-35
Review answers (2 of 2)	3-36
Exercise: Implementing and verifying a cluster, cluster queues, and a cluster gateway	3-37
Exercise objectives	3-38
Unit 4. Cluster administration tasks and commands	4-1
Unit objectives	4-2
IBM MQ and IBM MQ cluster administration contrasts	4-3
SENDER-RECEIVER distributed message channels	4-4
CLUSSDRx-CLUSRCVR clustered message channels	4-5
IBM MQ cluster administration commands	4-7
DISPLAY CLUSQMGR	4-9
DISPLAY CLUSQMGR	4-10
DISPLAY QCLUSTER	4-11
DIS QCLUSTER caveats	4-12
SUSPEND QMGR command	4-13
SUSPEND QMGR command with MODE(FORCE)	4-14
RESUME QMGR command	4-15
Cluster-specific procedures from IBM Knowledge Center (partial)	4-16
Add queue manager that uses separate transmission queue	4-17
Removing a cluster queue: Remove WM253.in from MQ03	4-19
Moving a repository to a different queue manager	4-20
Moving repositories (1 of 3)	4-21
Moving repositories (2 of 3)	4-22
Moving repositories (3 of 3)	4-23
Moving a full repository queue manager to a different host	4-24
Considerations when removing a queue manager from a cluster	4-25
Removing MQ03 from the cluster: Basic scenario (1 of 2)	4-26
Removing MQ03 from the cluster: Basic scenario (2 of 2)	4-27

RESET CLUSTER	4-28
Removing MQ0X from the cluster: Alternative scenario (1 of 2)	4-29
Removing MQ0x from the cluster: Alternative scenario (2 of 2)	4-30
REFRESH CLUSTER	4-31
REFRESH CLUSTER command	4-32
REFRESH CLUSTER command phase	4-34
Incorporating a queue-sharing group into an existing cluster	4-35
Recap of resources to administer your cluster environment	4-37
Monitor SYSTEM.CLUSTER and application queue instances	4-39
Using IBM MQ Explorer to work with clusters	4-41
Display cluster status	4-42
Suspend the cluster membership of a queue manager	4-43
Resume the cluster membership of a queue manager	4-44
Unit summary	4-45
Review questions	4-46
Review answers (1 of 2)	4-47
Review answers (2 of 2)	4-48
Exercise: Working with cluster administration tasks	4-49
Exercise objectives	4-50
Unit 5. IBM MQ security and clusters	5-1
Unit objectives (1 of 2)	5-2
Unit objectives (2 of 2)	5-3
Implementing IBM MQ security	5-4
Terminology baseline: Security areas	5-5
Connection authentication	5-7
CONNAUTH settings new queue manager initial default	5-9
CHCKCLNT and CHCKLOCL values (1 of 3)	5-10
CHCKCLNT and CHCKLOCL values (2 of 3)	5-11
CHCKCLNT and CHCKLOCL values (3 of 3)	5-12
Channel authentication rules	5-13
Channel blocking points	5-14
MCAUSER basics for client channels	5-15
CHLAUTH rule types	5-17
Which rule to use: ADDRESSMAP or BLOCKADDR	5-19
CHLAUTH rules using IP addresses and host names	5-20
CHLAUTH actions	5-21
Replacing and removing CHLAUTH rules	5-23
Generic IP addresses	5-25
CHLAUTH rule precedence	5-26
CHLAUTH initial settings	5-27
The back-stop rule and warning mode	5-28
Allow access to administrative users	5-29
CHLAUTH best practices (1 of 2)	5-30
CHLAUTH best practices (2 of 2)	5-32
Security considerations for IBM MQ clusters	5-33
Troubleshooting queue manager default settings (1 of 5)	5-35
Troubleshooting queue manager default settings (2 of 5)	5-36
Troubleshooting queue manager default settings (3 of 5)	5-37
Troubleshooting queue manager default settings (4 of 5)	5-38
Troubleshooting queue manager default settings (5 of 5)	5-39
Other considerations when working with IBM MQ security	5-40
An introduction to IBM MQ object authorizations	5-41
Components used to control access to an object	5-43
dspmqaut and setmqaut examples	5-45
Mixed commands in a secured queue manager (1 of 3)	5-47

Mixed commands in a secured queue manager (2 of 3)	5-48
Mixed commands in a secured queue manager (3 of 3)	5-49
Troubleshooting an object authorization problem (1 of 2)	5-50
Troubleshooting an object authorization problem (2 of 2)	5-51
Problems and troubleshooting recap	5-52
Search string to additional IBM MQ security reference	5-54
Unit summary (1 of 2)	5-55
Unit summary (2 of 2)	5-56
Review questions (1 of 2)	5-57
Review questions (2 of 2)	5-58
Review answers (1 of 2)	5-59
Review answers (2 of 2)	5-60
Exercise: Working with IBM MQ security	5-61
Exercise objectives (1 of 2)	5-62
Exercise objectives (2 of 2)	5-63
Unit 6. Influencing workload balancing behavior	6-1
Unit objectives	6-2
Before you begin	6-3
Workload balancing in clusters	6-4
When workload balancing occurs	6-5
How message affinity affects workload balancing	6-7
Workload: Many queue instances	6-8
Workload: A local queue instance	6-9
Cluster workload (CLWL) management options	6-10
CLWL use-queue basics	6-11
CLWL use-queue any	6-12
Specifying CLWL use-queue on a queue manager	6-13
Specifying CLWL use-queue on a queue	6-14
CLWL use-queue example	6-15
CLWL rank basics	6-16
Specifying CLWL channel rank	6-17
Specifying CLWL queue rank	6-18
CLWL channel rank example	6-19
CLWL queue rank example	6-20
NETPRTY channel attribute	6-21
CLWL priority basics	6-22
Specifying CLWL channel priority	6-23
Specifying CLWL queue priority	6-24
CLWL priority example (1 of 2)	6-25
CLWL priority example (2 of 2)	6-26
Most recently used channels	6-27
Most recently used channels basics	6-28
CLWL most recently used channel	6-29
Specifying CLWL most recently used channel	6-30
CLWL channel weight basics	6-31
Clustering weighted round-robin methodology	6-32
Specifying CLWL channel weight	6-33
Cluster workload management attributes	6-34
Cluster workload management algorithm: Summary (1 of 2)	6-35
Cluster workload management algorithm: Summary (2 of 2)	6-36
Cluster Queue Monitoring sample program (amqsc1m)	6-37
Cluster Queue Monitoring sample program logic	6-38
Message affinities	6-39
Handling message affinities	6-40
Workload balancing BIND options	6-41

BIND_ON_OPEN	6-42
BIND_NOT_FIXED	6-43
Customizing workload balancing	6-44
Cluster workload exits	6-45
Writing cluster workload exits	6-46
Unit summary	6-47
Review questions	6-48
Review answers	6-49
Exercise: Working with workload balancing options	6-50
Exercise objectives	6-51
Unit 7. Publish/subscribe and clusters	7-1
Unit objectives	7-2
Point-to-point and publish/subscribe	7-3
Distributed publish/subscribe patterns	7-4
Publish/subscribe basic components	7-5
Current publish/subscribe functionality: A little history	7-6
Publish/subscribe terminology baseline (1 of 2)	7-7
Publish/subscribe terminology baseline (2 of 2)	7-8
Publish/subscribe functionality	7-9
Topic tree, topic strings, topic nodes, and topic objects	7-10
Topic objects	7-11
Topic object attributes: DISPLAY TOPIC view	7-12
Topic object attributes: DISPLAY TPSTATUS view	7-13
Topic alias	7-14
Topic administration options	7-15
Topic status with IBM MQ Explorer	7-16
Subscriptions: The three aspects (1 of 2)	7-17
Subscriptions: The three aspects (2 of 2)	7-18
Subscription commands: DIS SUB(VEGET*) ALL	7-19
Subscription commands: DISPLAY PUBSUB ALL	7-20
Subscription commands: DISPLAY SUB(*) TOPICSTR DEST	7-21
Subscription related command: DISPLAY QLOCAL	7-22
Publications	7-23
Retained publications	7-24
Publish/subscribe testing in IBM MQ Explorer	7-25
Publish/subscribe lifecycle descriptions	7-26
Publish/subscribe lifecycles: Managed non-durable subscriber	7-27
Distributed publish/subscribe	7-28
Publish/subscribe clusters	7-29
Cluster topics	7-31
Cluster topic example	7-33
Display cluster topic example	7-34
Cluster modes for routing publications	7-35
Cluster topic routing	7-36
Routing behavior for publish/subscribe clusters	7-38
Topic host routing requirements	7-39
Scaling	7-40
Publication flows	7-41
Configuration	7-42
Availability	7-43
Comparison summary	7-44
Problem determination (1 of 2)	7-45
Problem determination (2 of 2)	7-46
Loop detection	7-48
Stopping a queue manager	7-49

Unit summary	7-50
Review questions	7-51
Review answers (1 of 2)	7-52
Review answers (2 of 2)	7-53
Exercise: Configuring a publish/subscribe cluster	7-54
Exercise objectives	7-55
Unit 8. Cluster design considerations	8-1
Unit objectives	8-2
Before you start, consider your environment	8-3
Reasons for dividing large systems	8-5
Combining related applications	8-7
Isolate data and users	8-8
Reduce resources	8-9
Independent administration	8-10
Different environments	8-11
Why overlapping clusters?	8-12
Classes of service and security	8-13
Design considerations for multiple clusters	8-15
Overlapping cluster considerations	8-16
Bridge queue managers	8-17
Bridge queue manager configuration	8-18
Bridges: Joining more than one cluster	8-19
Bridges: Two bridge queue managers	8-20
Put across clusters with a bridge queue manager (1 of 2)	8-21
Put across clusters with a bridge queue manager (2 of 2)	8-22
Gateway queue managers	8-23
Cluster gateway with paired queue managers	8-24
Gateways to and from clusters	8-25
Connected gateway queue managers	8-26
Put outside the cluster	8-27
Put outside the cluster by using a queue manager alias	8-28
Put outside the cluster by using a clustered QREMOTE	8-29
Put into the cluster by using QREMOTE	8-30
Put into the cluster by using a queue manager alias	8-31
Use workload management when putting to a cluster	8-32
Reply to a queue manager outside the cluster	8-33
Reply from a queue manager outside the cluster	8-34
Clustering channel rank between clusters	8-35
Unit summary	8-36
Review questions	8-37
Review answers	8-38
Exercise: Configuring an overlapping cluster	8-39
Exercise objectives	8-40
Unit 9. Course summary	9-1
Unit objectives	9-2
Course objectives (1 of 2)	9-3
Course objectives (2 of 2)	9-4
To learn more on the subject	9-5
Enhance your learning with IBM resources	9-6
Unit summary	9-7
Course completion	9-8
Appendix A. List of abbreviations	A-1

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:

CICS®
Notes®
z/OS®

DB2®
RACF®

IMS™
WebSphere®

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

Windows is a trademark 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.

VMware is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

PAR® is a trademark or registered trademark of Merge Healthcare, an IBM Company.

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

Course description

Designing, Implementing, and Managing IBM MQ V9 Clusters

Duration: 2.5 days

Purpose

This course prepares you to plan, design, configure, and administer IBM MQ clusters.

After a review of IBM MQ, the course explains the similarities and differences between administration of a cluster and administration of a distributed message IBM MQ environment. You learn cluster-specific commands and procedures, and explore the role of a cluster in a highly available IBM MQ infrastructure. You learn how to do a detailed verification of a new cluster configuration, review and identify all resulting components, and troubleshoot problems.

You also learn about cluster administration, workload balancing, security, and use of publish/subscribe clusters. You learn how to recognize when a problem is due to connection authentication, channel authentication, or object authorizations. You also learn about design options such as overlapping clusters and use of clusters with z/OS queue-sharing groups. Many of the considerations in this course are based on actual engagement experiences.

Audience

This course is designed for IBM MQ administrators, architects, application developers, and other professionals who need to understand the design considerations, architectural role, and implementation of IBM MQ clusters.

Prerequisites

Before taking this course, you should have experience with IBM MQ or complete one of the following courses:

- *IBM MQ V9 System Administration (using Windows for labs)* (WM153G)
- *IBM MQ V9 System Administration (using Linux for labs)* (WM154G)
- *IBM MQ V8 System Administration for z/OS* (WM302G)

You should also have working knowledge of the Windows operating system to run the lab exercises.

Objectives

- Describe the basic IBM MQ components
- Identify which IBM MQ objects are used to impact routing in a cluster environment

- Identify who in your organization can impact the health of a cluster and the need for adequate communication
- Describe the correct role of a cluster in a highly available IBM MQ infrastructure
- Describe the differences and similarities between administering clustered and non-clustered IBM MQ environments
- Describe how to configure, verify, and troubleshoot an IBM MQ cluster
- Identify the various channels that are present in a cluster environment and how each is created
- Describe how to use separate transmission queues in a clustered queue manager
- Explain how to remove a queue manager from a cluster on a permanent or temporary basis
- Explain IBM MQ connection authentication
- Explain IBM MQ channel authentication
- Describe IBM MQ object authorizations
- Explain how to troubleshoot security challenges in a cluster
- List ways to influence workload balancing in a cluster
- Describe the history and basic components of IBM MQ publish/subscribe
- Explain the considerations and details of implementing publish/subscribe in an IBM MQ clustered environment
- Describe cluster design architectural considerations
- Summarize the benefits of design and configuration simplicity in a cluster implementation
- Explain how to configure overlapping clusters

Agenda



Note

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

Day 1

- (00:15) Course introduction
- (01:30) Unit 1. IBM MQ baseline
- (01:00) Exercise 1. Configuring and reviewing base IBM MQ resources
- (00:30) Unit 2. Before you start
- (01:30) Unit 3. Understanding and implementing an IBM MQ cluster
- (01:30) Exercise 2. Implementing and verifying a cluster, cluster queues, and a cluster gateway

Day 2

- (01:00) Unit 4. Cluster administration tasks and commands
- (01:30) Exercise 3. Working with cluster administration tasks
- (01:00) Unit 5. IBM MQ security and clusters
- (02:00) Exercise 4. Working with IBM MQ security
- (01:00) Unit 6. Influencing workload balancing behavior
- (01:30) Exercise 5. Working with workload balancing options

Day 3

- (01:00) Unit 7. Publish/subscribe and clusters
- (01:00) Exercise 6. Configuring a publish/subscribe cluster
- (00:30) Unit 8. Cluster design considerations
- (01:00) Exercise 7. Configuring an overlapping cluster
- (00:15) Unit 9. Course summary

Unit 1. IBM MQ baseline

Estimated time

01:30

Overview

This unit provides a review of IBM MQ topics. Where appropriate, it describes how administration of a cluster resembles regular unclustered IBM MQ administration. The unit also establishes a base for later topics, and identifies areas that might require extra considerations for clusters.

How you will check your progress

- Checkpoint
- Machine exercises

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
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 exists, this page is where it is found.
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
--------------	--------------	------------------------	-------

IBM MQ baseline

© Copyright IBM Corporation 2017

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

Unit objectives

- Summarize the IBM MQ components
- Describe the role of a queue manager
- Describe an IBM MQ message
- Describe queues and identify what types of queues hold messages
- Describe the IBM MQ procedural application programming interface
- Describe precedence between queue and API attributes
- Distinguish among the various types of IBM MQ channels
- Explain how a channel is started by using triggering
- Describe the trajectory of a message that uses distributed message channels
- Identify IBM MQ troubleshooting resources
- Describe queue name resolution in a distributed message channel environment
- Distinguish between the point-to-point and publish/subscribe messaging styles
- Explain the impact of IBM MQ design and development considerations
- Introduce an IBM MQ cluster
- Explain IBM MQ Shared Queues

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-2. Unit objectives

IBM MQ components and baseline

- The first part of this unit focuses on the basic IBM MQ components
 - Messages
 - Queues
 - Queue manager
 - Channels
 - Message queuing (application) interface

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-3. IBM MQ components and baseline

This unit sets the base for your work with IBM MQ clusters by reviewing IBM MQ concepts that you build on to implement a cluster.

The queue manager, or messaging server, owns IBM MQ resources and controls processes such as defining objects, trigger starting a channel or process, creating event messages, expiring messages, and message distribution by using clustering.

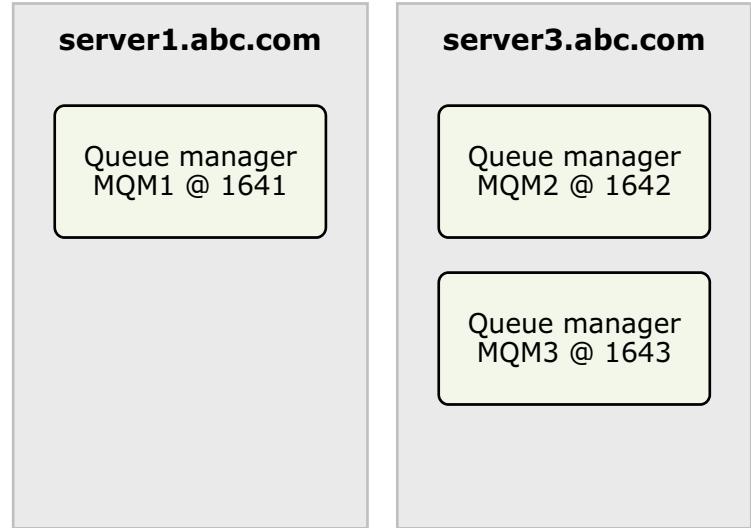
Your business data travels as messages that the queue managers exchange.

Queues are where messages are kept.

Channels move messages from one queue manager to another queue manager.

Queue manager

- System program that owns the resources it services
- Provides services that are needed for messaging
- A server can host more than one queue manager
- Queue managers that share a server require different TCP/IP port numbers



IBM MQ baseline

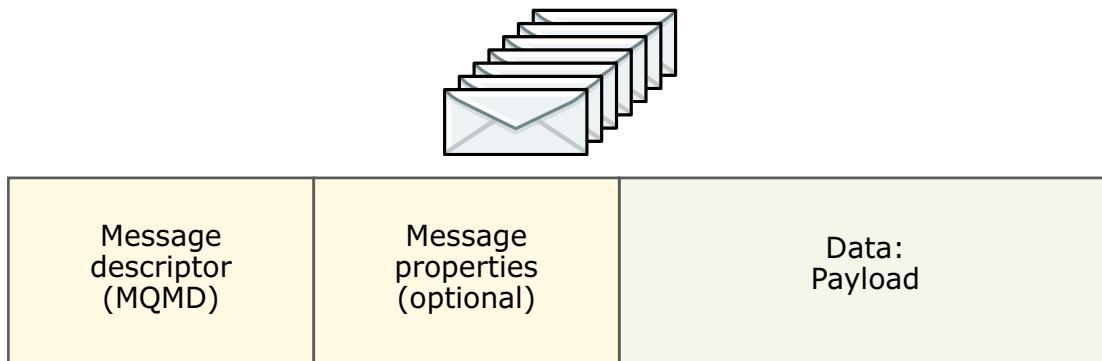
© Copyright IBM Corporation 2017

Figure 1-4. Queue manager

When you install IBM MQ server software, you need to create one or more queue managers. A queue manager owns resources that are defined to it, which you use to exchange messages. When you have more than one queue manager in the same host, the queue managers must have a different port number. The default IBM MQ port is 1414.

Messages

- Messages contain the business data or payload
- Messages are a distinct string of bytes that are exchanged between two programs
- All messages contain a IBM MQ Message Descriptor (MQMD) with control information
- Optionally, applications can add message properties to a message



IBM MQ baseline

© Copyright IBM Corporation 2017

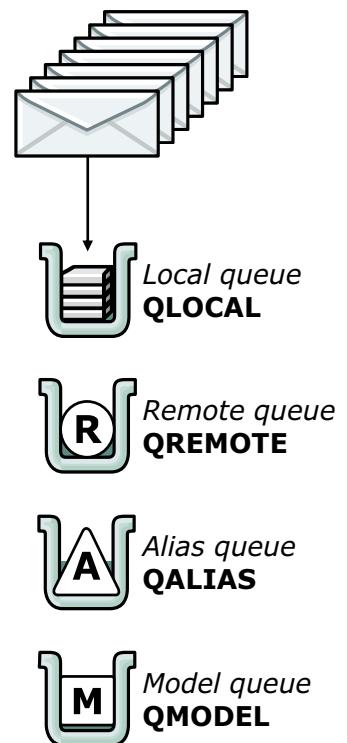
Figure 1-5. Messages

Messages contain the message descriptor, or MQMD structure, and the message data or payload. Messages might contain other headers. The message descriptor contains many attributes, or properties, and other details that IBM MQ uses to process the messages. When developers write an IBM MQ application, they use preset initial default values to initialize the MQMD structure.

A message can optionally have properties. Message properties to allow an application to select messages to process, or to retrieve information about a message.

Queues

- A queue is a defined destination for messages
- Four types of queues can be created: QLOCAL, QREMOTE, QALIAS, QMODEL
- Some local queues are designated for special purposes in a queue manager
- A remote queue or QREMOTE is a pointer to a local queue in a remote queue manager
- An alias queue or QALIAS is a pointer to a local queue or a locally owned remote queue
- A model queue or QMODEL is a template to create a dynamic local queue



Only local queues that are defined as a QLOCAL queue type can hold messages

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-6. Queues

Queues are where applications send messages to and get messages from.

When you start developing IBM MQ code, you might spend some time determining “Where is my message?” Queues have different types. What is key to remember, is:

- That **only local queues of type QLOCAL at definition time, hold messages**. Keeping this concept in mind is helpful when looking for a message.
- The **type of queue** counts when the queue manager is doing queue name resolution, that is, when it determines which is the target queue to send a message.

Queues other than QLOCAL queue types are pointers or templates to other queues:

- Remote queues point to a transmit queue, and a remote queue manager and target queue. QREMOTE is the queue type only. You can also think of these queues as “local QREMOTES”.
- Alias queues, or QALIAS, point to another local queue or a topic. You revisit these queues in a later unit.

Model queues, or QMODEL, can be thought of as a template to create queues. The QLOCAL type queues, which are created when an application uses a model queue, hold messages. The model queue itself does not hold messages. Since these local queues are created in your code, they are referred to as dynamic queues. You work to create these dynamic queues by using a model queue in a later exercise.

- In a cluster, there exists a SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE, which is used as a model for transmission queues that get defined in a cluster.
- You might be familiar with “shared queues”. Shared queues are available exclusively in the z/OS platform. You learn about shared queues later in this unit.

QLOCAL partial attributes and defaults

- When defining a QLOCAL, the only required parameter is the queue name
- When an attribute is not specified in the definition, it is given a default value
- The DEFBIND attribute of a queue can influence workload balancing behavior for clusters
 - DEFBIND(OPEN) causes an affinity
 - DEFBIND(NOTFIXED) is preferred
- The persistence attribute (DEFPSIST) defaults to non-persistent
 - The persistence attribute determines whether a message survives restarts of the queue
 - Standards on how and when to use persistence should be established
- A transmission queue is a QLOCAL with its USAGE attribute set to XMITQ

```

DIS Q(BILL_IN)
QUEUE(BILL_IN)
TYPE(QLOCAL)
CRDATE(2014-02-26)
CURDEPTH(1)
DEFBIND(OPEN)
DEFPSIST(NO)
INITQ( )
NOTRIGGER
PROCESS( )
QDEPTHHI(80)
TRIGDATA( )
TRIGDPTH(1)
TRIGTYPE(FIRST)
MAXMSGI(4194304)
USAGE(NORMAL)

```

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-7. QLOCAL partial attributes and defaults

Defining a queue local is simple. To define a queue local without any special requirements, you issue a DEFINE QLOCAL followed by the queue name, and the queue is created; but many attribute values are defaulted. As you do more work with IBM MQ, you learn the rest of the attributes and their importance.

Of particular interest for cluster is the DEFBIND attribute. When DEFBIND is set to OPEN, it creates message affinities that interfere with the cluster workload balancing. You learn more about DEFBIND later in this course.

The queue manager sets some of the queue attributes; for instance, CURDEPTH, which displays how many messages are in the queue, or CRDATE, which displays the date that the queue was created. An application might specify attributes that can supersede most of the attributes that are specified or defaulted in a queue definition. If it is imperative to use a specific attribute, it is a good practice to have the application set it rather than default to the attribute in the queue definition.

A message's persistence determines whether the message survives restarts of the queue manager. Whether the message is critical to the application, for instance, is not a simple query; it should be made persistent. However, if a message does not need to be persistent, it is more efficient to leave it as a non-persistent message. Persistent messages place a higher burden on system resources.

Some local queues are designated for a special purpose

- Transmit or transmission queue
 - Local queue with its usage attribute set to XMITQ in the queue definition
 - If an application puts a message to a remote queue, it goes to a transmit queue
 - Channels use transmit queues to send messages to the remote queue manager
- Initiation queues
 - Identified as an initiation queue in a definition of another local queue
 - Associated with triggering
- Dead-letter queue
 - Local queue that is identified to the queue manager as its dead-letter queue to hold undeliverable messages
 - Usually the SYSTEM.DEAD.LETTER queue is designated as the dead-letter queue; however, a different local queue can be defined, such as BILLING.DLQ



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-8. Some local queues are designated for a special purpose

The basic queue types were discussed in the last slide. Several other queues are named after their purpose, but these queues are also local queues, or QLOCALS.

Transmission queues, initiation queues, and the dead-letter queue are all queues of type QLOCAL.

Transmission queues can be a deciding factor for queue name resolution.

Initiation queues are used for the purposes of triggering processes. A process can be an application, and can also be the start of a channel.

The dead-letter queue is used for undeliverable messages. A dead-letter queue includes a header, which contains the reason that the message was placed in the queue.

Queue types that are used for different message routing scenarios

- Queue manager alias
 - Uses the QREMOTE queue object definition with partial fields that are defined to map queue manager name
 - Example:

```
DEFINE QREMOTE(GATEWAY0) RQMNAME(' ') RNAME(' ') XMITQ(' ')
```

- Remote queue
 - Is used to map to other queue names in a remote IBM MQ server
 - Example:

```
DEFINE QREMOTE(PAYMENTS) RQMNAME(GATEWAY0) RNAME(CASHIN) +
XMITQ(MQ00)
```

- Queue alias
 - Is used to map to other queue or topic names and also for security purposes
 - Example:

```
DEFINE QALIAS(SPORT.SALES.IN) TARGET(PAYMENTS)
```

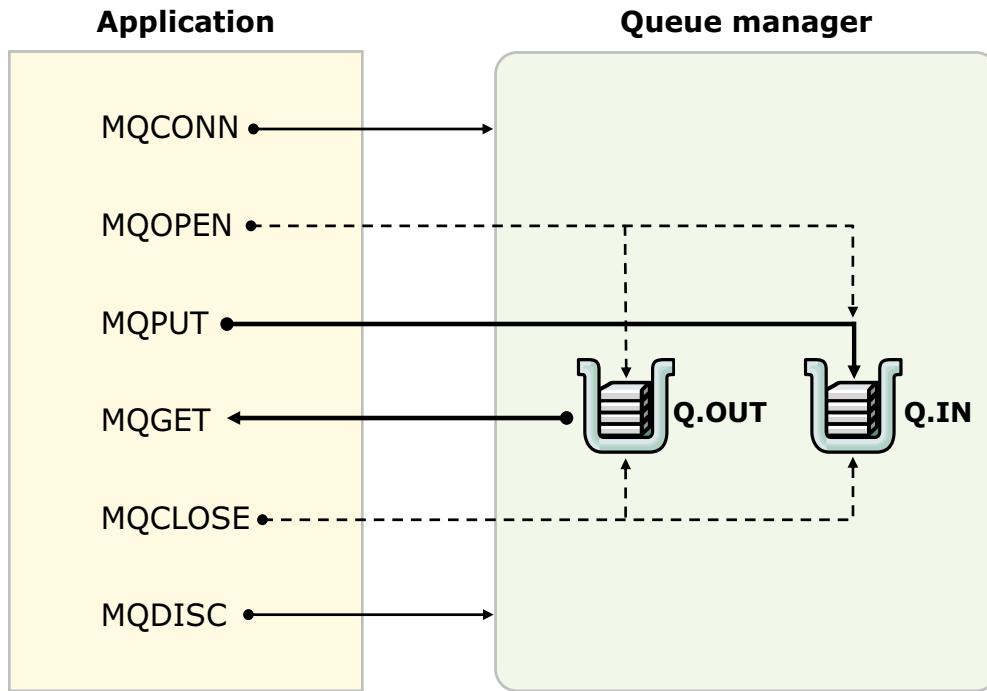
Figure 1-9. Queue types used for different cluster routing scenarios

In your IBM MQ administration work, you use a QREMOTE object to define a queue manager alias. The queue manager alias mitigates routing messages between two or more queue managers. The queue manager alias is also used for routing messages when you use a cluster.

You can use the QREMOTE in a normal context with all the parameters specified. You see how the definition that is shown in the example is used in a later unit.

You are also familiar with the QALIAS object, which can be used to point to a queue, or also to a topic. Topics are used for publish/subscribe, which is detailed in a later unit.

Basic Message Queue Interface (MQI) calls



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-10. Basic Message Queue Interface (MQI) calls

You learned about queues, messages, and channels. Message data is placed and retrieved from queues by using applications that are written with the IBM MQ function calls.

Applications connect to a queue manager and open a queue. If the MQCONNECT and MQOPEN calls are successful, then messages can be put to, or retrieved from, a queue.

The MQOPEN provides access to the queue. Each queue needs its own MQOPEN call, and its own MQCLOSE call.

The attributes that are used in the MQOPEN are a critical consideration in a cluster environment. The application attributes override the queue definition attributes. Applications that use BIND on OPEN in the MQOPEN call create affinities. If many messages are expected to be placed in the queue, a loop of puts can accomplish this purpose. It would be more efficient than connecting and opening the queue for every put.

Applications can also open a topic and “publish” to this topic. You work with the concept of opening a topic in a later exercise. The figure in the slide is a high-level representation. The MQOPEN broken arrow to each queue denotes that each queue has an MQOPEN call, and each queue has an MQCLOSE call.

The slide shows the sequence in which the calls need to be coded. Each queue requires its own MQOPEN, MQCLOSE, and related declarations.

Channels

- Channels are the combination of MCAs and network
- Usually the sending MCA, the network connection, and the receiving MCA
- Different channel categories:
 - **Message channels:** Distributed or clustered
 - **Client (or MQI) channels**

AMQ8414: Display Channel details.
 CHANNEL (MQM1.MQM3)
 CHLTYPE (SDR)
 CONNAME (server3.abc.com(1643))
 TRPTYPE (TCP)
 XMITQ (MQM3)



The term "MQI" has two uses in IBM MQ



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-11. Channels

Channels are the processes that move messages across queue managers or servers with an IBM MQ client installation. Channels have different categories:

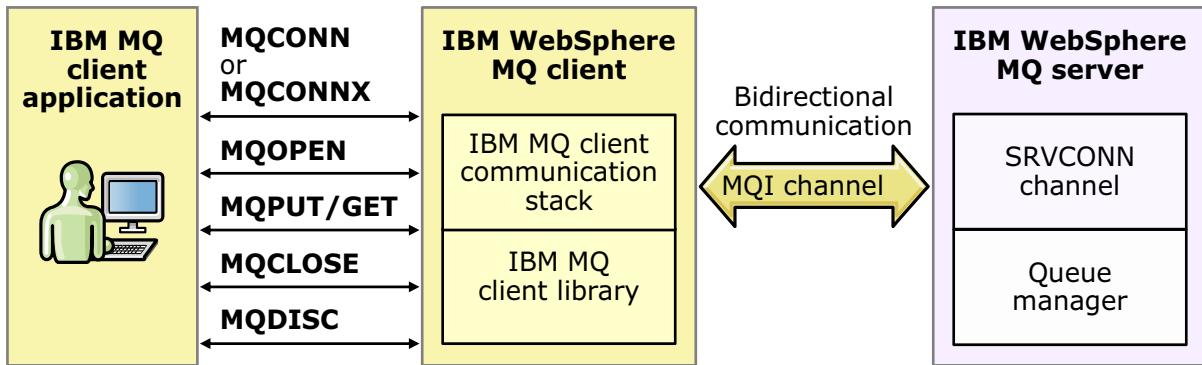
- Message channels: These message channels can be distributed, or as you learn later in this course, can be clustered.
- Client, or MQI channels.

Within distributed channels are different types of channel pairs. Sender-receiver channels are the most prevalent type of distributed message channel pairs. Usually, one main difference between different channel pairs is the way that the channels are started.

Clustered channels are detailed in a later unit. IBM MQ has two base products: the IBM MQ server and an IBM MQ client. An IBM MQ client installation does not host queues or queue managers. An IBM MQ client is a set of libraries that allows an application to be written by using the IBM MQ API, also referred to as the MQI, that can connect to a queue manager.

An IBM MQ client channel, or MQI channel, is a connection from an IBM MQ client installation to a server connection, or SVRCONN channel, in a queue manager that is configured in an IBM MQ server installation. An IBM MQ client application can connect to an IBM MQ cluster queue manager in a similar way that a non-cluster full queue manager server can connect to a cluster queue manager. The main difference is that the client uses client connections to an SVRCONN channel in the queue manager side.

IBM MQ clients



- An IBM MQ client has IBM MQ client libraries that are installed
- No queue managers or queues are defined
- Uses the same MQI calls as the IBM MQ server
- Flow of messages in an MQI channel is bidirectional
- Transactional capabilities

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-12. IBM MQ clients

An IBM MQ client is an installation option that contains a different set of IBM MQ libraries that run in client mode. The IBM MQ client does not have queues or queue managers. An IBM MQ client application and a server queue manager communicate with each other by using an MQI channel. An MQI channel is a bidirectional connection. The IBM MQ client side has different options to connect to the queue manager SVRCONN channel.

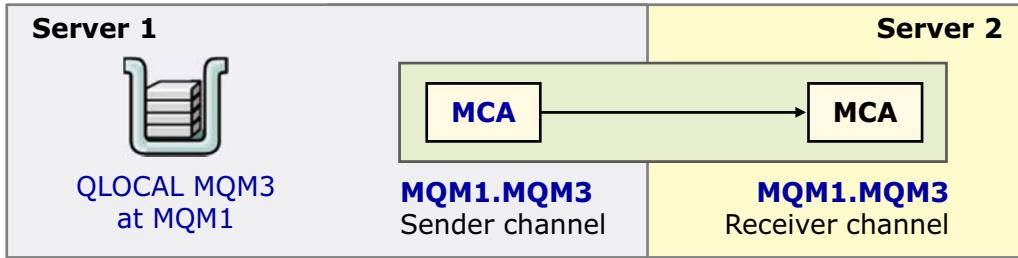
An MQI channel starts when the client application issues an MQCONN or MQCONNXX call to connect to the queue manager. It ends when the client application issues an MQDISC call to disconnect from the queue manager.

Do not confuse the term client as used in the general architectural mode, with an IBM MQ client. An application that is compiled with IBM MQ server libraries might serve as a client application, but uses distributed channel pairs. An IBM MQ client is compiled with IBM MQ client libraries, uses MQI channels, and must connect to a queue manager SVRCONN channel.

In addition to the traditional IBM MQ client installation is an IBM MQ redistributable client. A redistributable client is not installed, but extracted to a selected directory. A redistributable client facilitates packaging your IBM MQ client applications and moving them to another server without any IBM MQ installed. The one requirement for a redistributable client is to have the correct runtime libraries. A redistributable client that is installed in a server without IBM MQ is referred to as a “non-installed” client. The redistributable can also be installed in the same host as an IBM MQ server and extracted to

its own directory. A redistributable client that is colocated with an IBM MQ server is referred to as a “relocatable” redistributable client. The redistributable client has utilities to set up its own environment, and it runs independently and is isolated from the IBM MQ server installation.

Using the transmission queue to trigger start a channel



- Configure the transmission queue by observing the following steps:
 - Ensure that the channel initiator task is running
 - Use SYSTEM.CHANNEL.INIT.QUEUE as the initiation queue
 - Set up the queue to TRIGGER
 - TRIGDPTH and TRIGTYPE can be left as default
 - Specify the channel name in the TRIGDATA attribute
- Channel must be started manually
 - One time before it is triggered
 - Any time the channel sender channel is explicitly stopped

```

QUEUE (MQM3)
INITQ (SYSTEM.CHANNEL.INIT.QUEUE)
TRIGGER
TRIGDATA (MQM1 . MQM3)
TRIGDPTH (1)
TRIGTYPE (FIRST)
USAGE (XMITQ)
  
```

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-13. Using the transmission queue to trigger start a channel

When a channel is initially defined, it is in stopped state. You must manually start the channel one time to get it out of stopped state. The channel runs for a predetermined time, then stops.

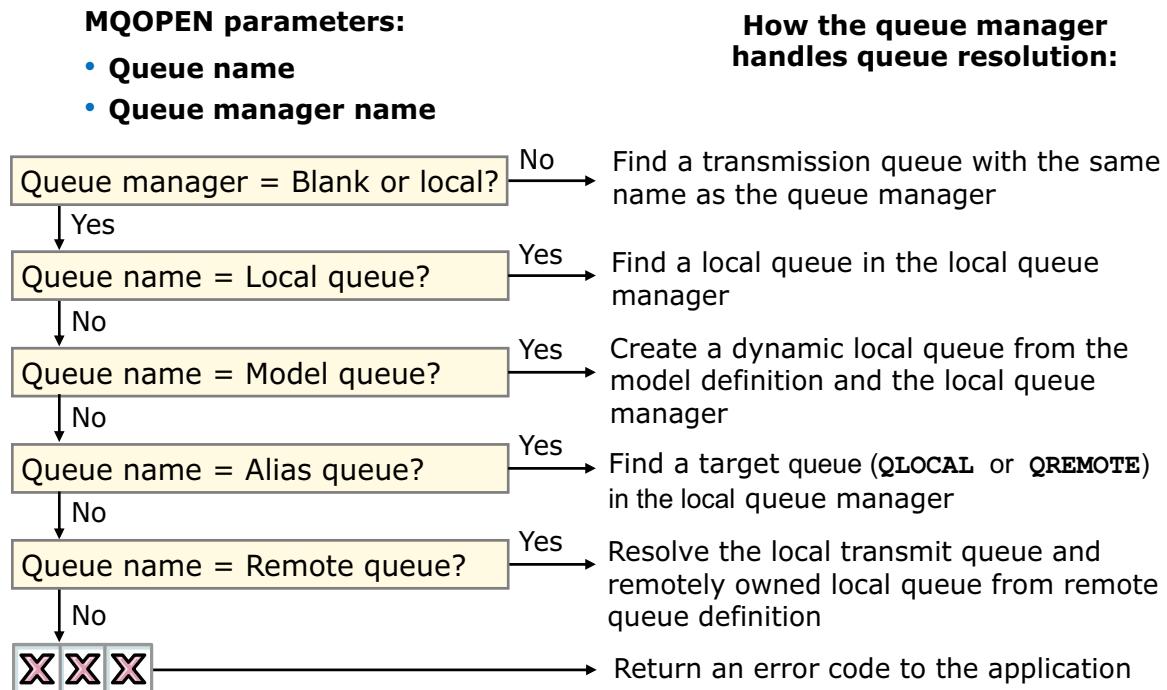
It would be inconvenient to constantly manually start channels. Channels are automatically started by using triggering.

“Triggering a channel” is a misnomer, as what is triggered is the transmission queue. As depicted on this slide, when you trigger start the channel, you need to set the attributes of the associated transmit queue, where you:

- Set the queue to TRIGGER
- Set the initiation queue to SYSTEM.CHANNEL.INITQ
- Can leave the default initial values for TRIGDPTH and TRIGTYPE
- Type the channel name in the TRIGDATA attribute.

Ensure that the channel initiator process is running. You also need to manually start the channel before triggering works. If you must manually stop the channel, you must repeat the manual restart.

Queue name resolution



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-14. Queue name resolution

Different factors determine how the queue manager handles queue name resolution. The diagram summarizes some of the key concepts.

The queue manager name that is passed in the MQOPEN call is checked.

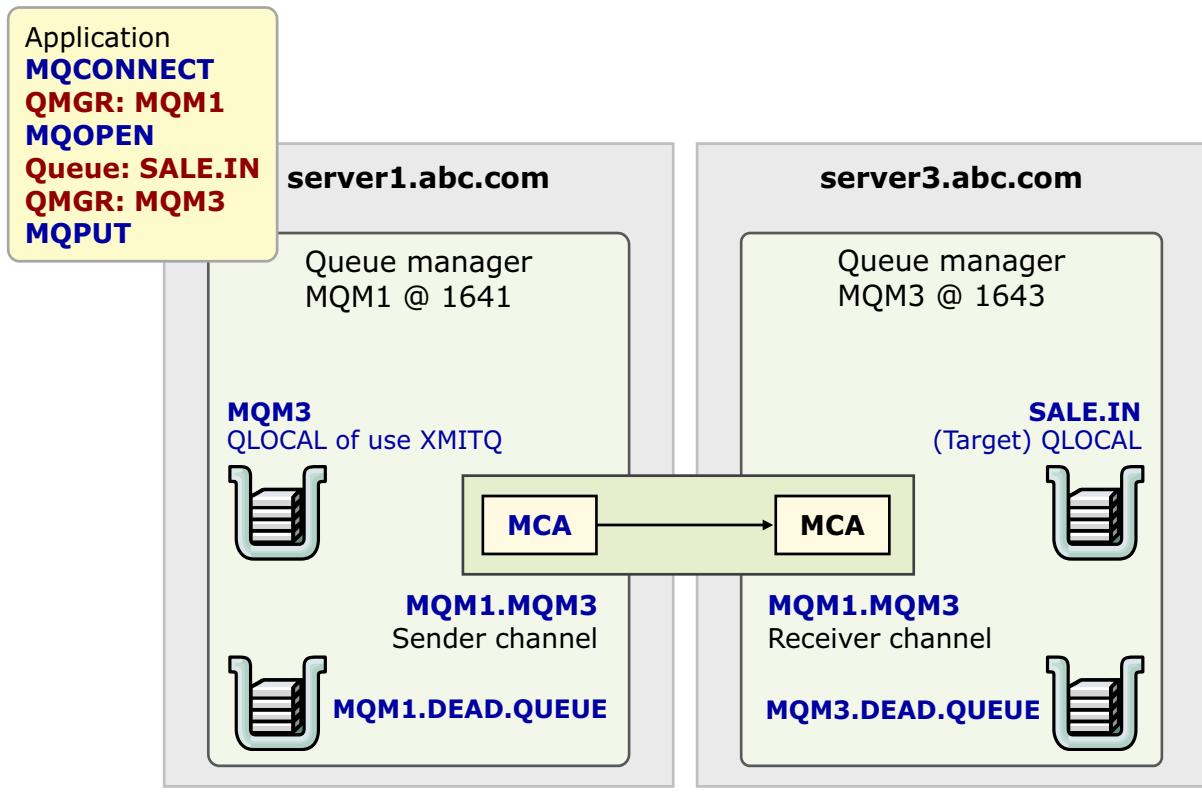
If the name of this queue manager is not omitted, and does not match the name of the local queue manager, queue name resolution looks for a locally defined transmission queue. It looks for one with a name that matches the name of the queue manager that is used in the MQOPEN call.

If the queue manager name was blank, queue resolution checks whether a local queue that matches the name of the queue that is used in the MQOPEN exists. If such a local queue is found, queue resolution ends placing the message in the local queue.

The name of the queue that is used in the MQOPEN might not match the name of a model queue, or the name of an alias queue. In that case, name resolution looks for the name of a remote queue that matches the name that is used in the MQOPEN.

If no match is found, an error is returned to the application, usually a return 2085, unknown object name.

Sender-receiver channel pair without local remote queue



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-15. Sender-receiver channel pair without local remote queue

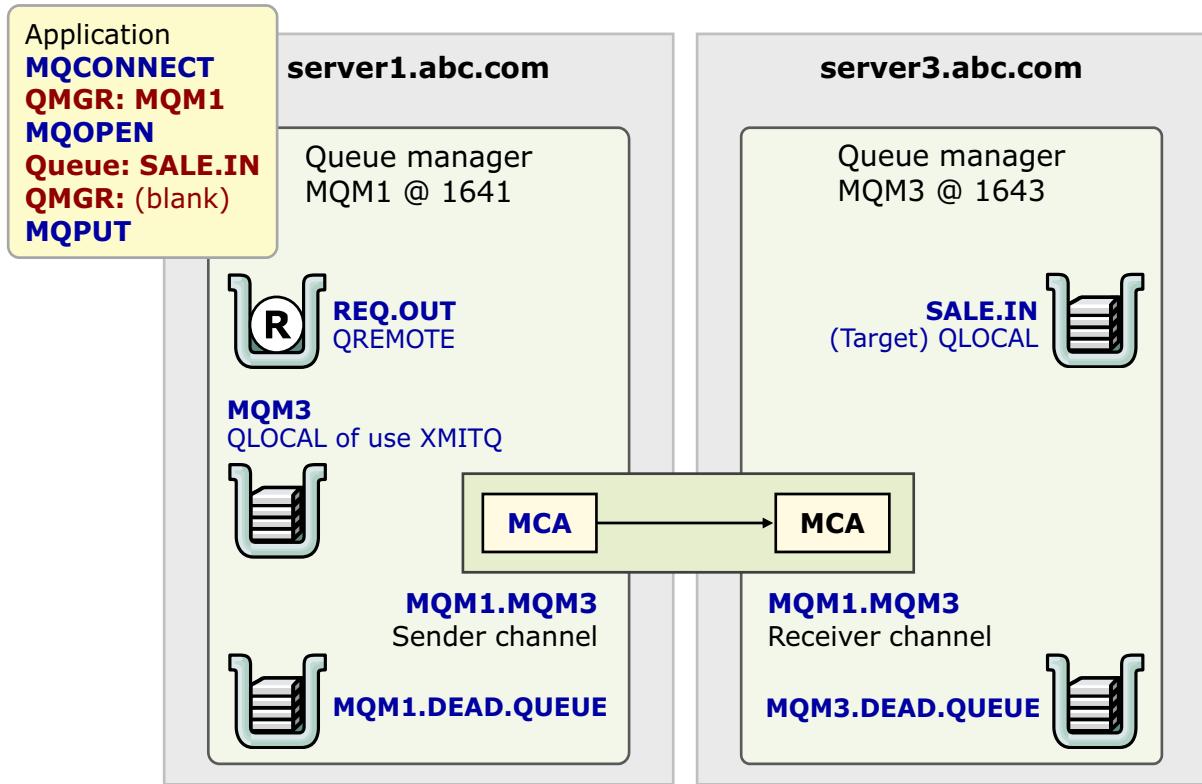
This visual of a sender-receiver channel pair displays some basic concepts. The names of the channels – sender and receiver – are both MQM1.MQM3. If the sender and receiver names are not identical, the channel does not start. The channels might be called anything, such as TRICK.OR.TREAT. If both ends of the channel have the same name, assuming all other connectivity details are correct, they work.

Be careful with the connection name (CONNNAME) attributes. Omitting the port number results in a connection to port 1414 attempted. 1414 is the IBM MQ default port.

This channel takes messages from transmit queue MQM3 to send it to the remote queue manager of the same name. It is a good practice to name the channels to be the same as the “from” and “to” queue managers, in the correct from-to order.

If a second channel pair needs to be defined from and to the same queue managers, the name can always be qualified to differentiate. The from-to naming convention aids with self-documentation of the channel.

Sender-receiver channel pair with local remote queue



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-16. Sender-receiver channel pair with local remote queue

You defined the queues and channels, and sent a message. Now you need to locate the message.

This application for this diagram uses the REQ.OUT QREMOTE in the MQM1 local queue manager to place a message to the SALE.IN remote local queue in the MQM3 remote queue manager. Where can the message be?

QREMOTES, or local remote queues, do not hold messages, so the following list shows the path of the message on its way to the SALE.IN queue:

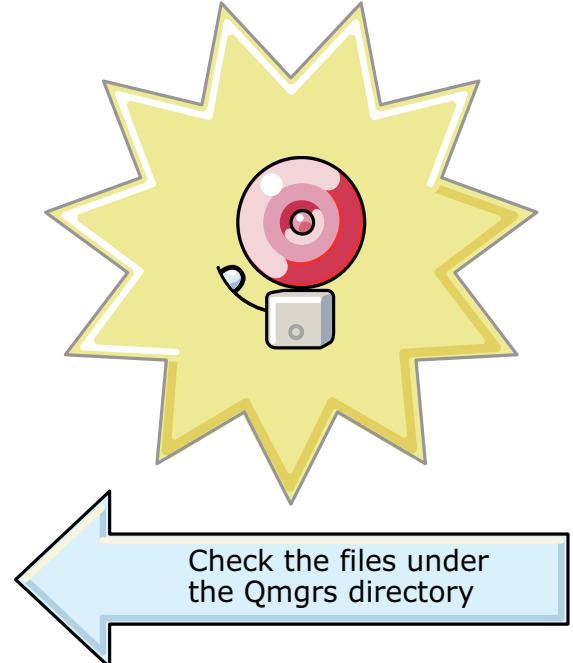
- When an application uses REQ.OUT for the MQPUT, a message goes to the transmission queue associated with the REQ.OUT remote queue, that is, the MQM3 local transmit queue.
- If the channel is triggered or already running, the channel picks up the message and forwards it to the MQM3 queue manager. If the channel is not running, the message is in the MQM3 transmission queue of the MQM1 queue manager.

If the SALE.IN remote local queue is defined and available, the message is placed in its target queue.

When you use clusters, IBM MQ still uses transmission queues and channels, but the amount of work you must do to configure the environment as an administrator is reduced.

Where to look for errors

- Error codes
 - Can be returned to an application or written to a log or queue
 - Can be the result of issuing a command
 - Can arise from a situation that the queue manager detects, such as low on log space or channel starting
- 4-character error codes
 - In distributed platforms, you can use the `mqrc` utility
 - Look up errors in the IBM Knowledge Center
- Log files for queue manager-specific errors are at:
`<install_loc>/qmgrs/QMName/errors`
- MQSC commands
- Dead-letter queue message



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-17. Where to look for errors

If you find a 4-character numeric return code when you use a distributed queue manager, you can use the `mqrc` command followed by the return code to obtain a summary line of the error. For example, you type the command followed by the code in a command prompt window:

```
mqrc 2053
```

Response:

```
2053 0x00000805 MQRC_Q_FULL
```

You can also find error information by using the other resources that are listed in this slide. When using the logs, you need to use the **queue manager logs**, under the `qmgrs` directory. Other error logs that are not under the `qmgrs` directory contain **IBM MQ product errors**, not queue manager-related errors. The IBM MQ product logs do not provide any help in the resolution of channel problems. You must use the queue manager log under the `qmgrs` directory, and then under the queue manager name.

One-way objects with remote queue definition checkpoint

At server1.abc.com

- Create, configure, and start queue manager MQM1
- Set dead-letter queue to SYSTEM.DEAD.LETTER.QUEUE
- Define and start listener
 - Set to automatically start with queue manager at port 1641
- Define QLOCAL MQM3 to use as a transmission queue
- Define QREMOTE REQ.OUT to:
 - Use QLOCAL MQM3 as transmit queue
 - Point to remote QLOCAL SALE.IN in queue manager MQM3 as target
- Define sender channel MQM1.MQM3 to:
 - Point to server server3.abc.com port 1643
 - Use MQM3 as its transmission queue
 - After RECEIVER is defined, start channel

At server3.abc.com

- Create, configure, and start queue manager MQM3
- Set dead-letter queue to SYSTEM.DEAD.LETTER.QUEUE
- Define and start listener
 - Set to automatically start with queue manager at port 1643
- Define QLOCAL SALE.IN
- Define receiver channel MQM1.MQM3

IBM MQ baseline

© Copyright IBM Corporation 2017

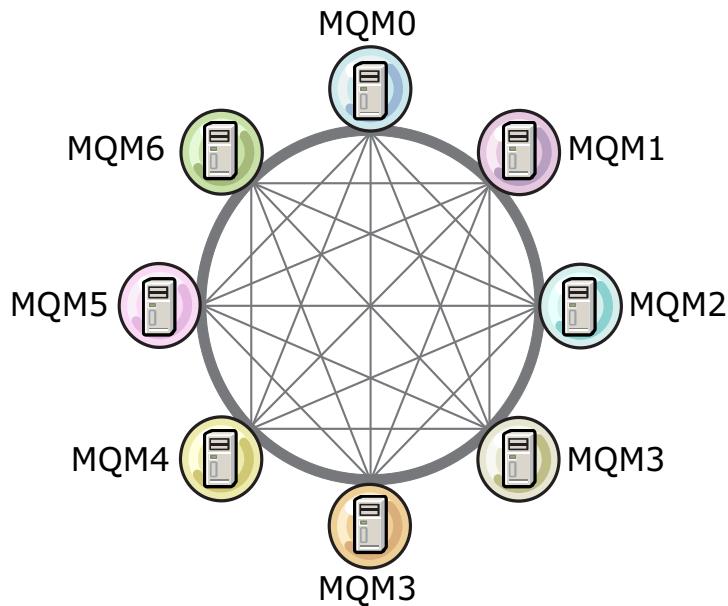
Figure 1-18. One-way objects with remote queue definition checkpoint

This slide summarizes the definitions that are made for the channel pair from MQM1 to MQM3. After you configure and start a queue manager on each side, the definitions that are needed to send a message that uses a remote queue over a sender-receiver channel pair are:

- On the sender side:
 - A local queue of type XMITQ to be used by the QREMOTE and sender channel definitions.
 - Optional: A queue remote. In a coming unit, you learn that a QREMOTE definition is not always needed.
 - A sender channel to the remote queue manager.
- On the receiver, or remote queue manager:
 - A local queue, which is associated with the sender side QREMOTE.
 - A receiver channel, of the same name as the sender channel in the sender side.

It is assumed that when you configure the queue manager on each side, you assign a port, which is used in the sender channel unless it is 1414. It is also assumed that you identify a queue to be used as the dead-letter queue.

Typical queue manager point-to-point channel definitions



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-19. Typical queue manager point-to-point channel definitions

If you need to establish point-to-point connectivity across all your queue managers, you end up with numerous channels that point to and from each queue manager. You might have several channel pairs across the same two queue managers, which perhaps handle messages of different sizes, or different classes of service.

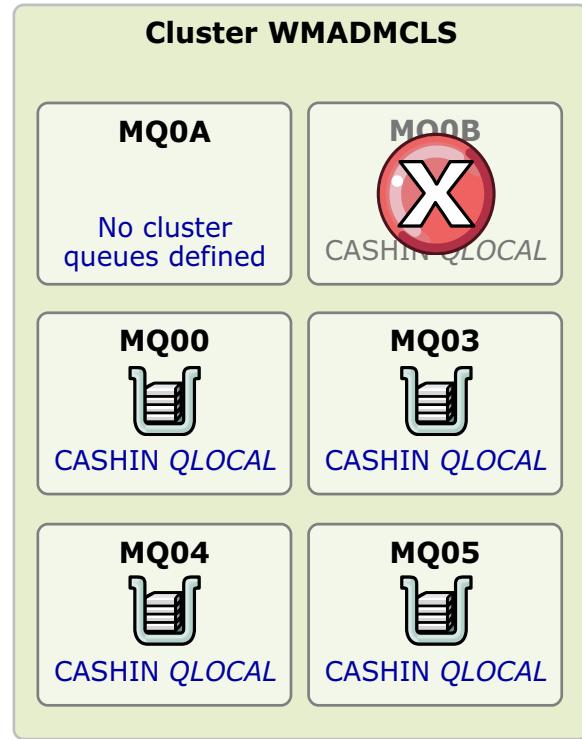
When you looked at the sender-receiver pair, you worked with a sender-receiver distributed message channel pair.

Clustered channels are less labor-intensive for the IBM MQ administrator. Clustered channels pose other considerations to the application developer. These considerations, which involve avoiding the introduction of queue manager affinities, are discussed in a later unit.

You now look at clusters and clustered message channels.

IBM MQ clusters

- Simplified administration
 - Reduce number of remote queues, transmit queues, and channel definitions
- Workload balancing
 - Same queue can be hosted in several queue managers
 - Route around failures
- Scalable
- Contribute to high availability
- MQPUT to a local queue manager if the queue exists locally
 - If the queue is not found locally, an algorithm is used for target queue selection
- MQGET is done to the local queue manager
- Publish/subscribe can use clusters



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-20. IBM MQ clusters

An IBM MQ cluster is a grouping of queue managers that share knowledge of each other's queues and can exchange messages across queue managers without the need to define specific channels between all queue managers. After a queue manager “joins” a cluster, it learns about the queues in other clusters, and how to reach those queues. The information that is needed to reach the clustered queues is kept in a queue manager cluster repository.

A repository can be full or partial. A cluster should have two repositories that are designated as full repositories. These two full repositories should know about each other. The queue managers that hold these full repositories are called the full repository queue managers. Other queue manager members of the cluster hold partial repositories. These partial repositories collect information on a “need to know” basis, and also send any news about objects that are defined to one of the full repository queue managers.

*Two repositories are the preferred number of full repositories, but not a requirement. Each cluster should have **at least** two full repositories, and can have more than two. However, more than two repositories might cause management of the cluster to be more complex.*

If an application needs to put a message to the CASHIN queue from MQ0A, this queue would be available in queue managers MQ00, MQ03, MQ04, and MQ05. IBM MQ recognizes the member queue manager that is not available, and excludes that queue manager from the potential target queue managers.

As you look at how the clusters are defined, the reduction in the number of IBM MQ object definitions that are required becomes apparent.

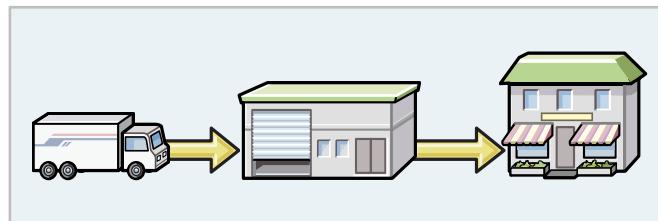
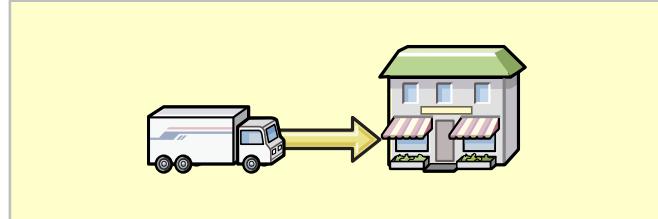
The slide qualifies the statement “contribute to high availability” because a cluster is not a high availability solution. Messages in cluster queues in a failed cluster queue manager are inaccessible unless the queue manager is part of a queue-sharing group or configured for high availability. However, clustering contributes to high availability, scalability, and workload balancing for new incoming requests.

MQGET calls are always done to the local queue manager.

This slide shows some of the advantages of clusters. However, in the next unit, you take a more detailed look at the capabilities that cluster environments provide, and details that you must take into consideration for a successful cluster environment.

Messaging styles

- Point-to-point application
 - Sends messages to a predefined destination
 - Application does not need to know where the destination is because IBM MQ locates the target by using object definitions
- Publish/subscribe application
 - Publishes messages to an interim destination according to a topic
 - Interested recipients subscribe to the topic
 - No explicit connection between publishing and subscribing applications



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-21. Messaging styles

The two messaging styles are point-to-point and publish/subscribe.

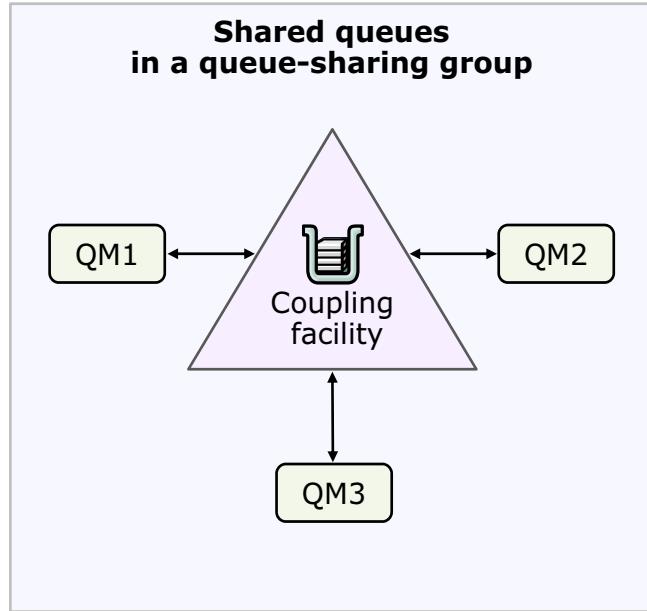
With point-to-point messaging, you know where the message is going. Publish/subscribe messaging style is further decoupled. In publish/subscribe, the producers of messages send or publish messages to an interim place where the consumers or subscribers obtain, or subscribe to a topic. This topic or topic string is associated to the messages.

You can use clusters with point-to-point and publish/subscribe environments.

You look at publish/subscribe in a later unit.

What are shared queues?

- Exclusive to z/OS
- Queues whose messages are kept in a coupling facility
- Queue managers in the same z/OS sysplex access the same shared queues by using connectivity via the coupling facility
- When several z/OS queue managers share queues, they are called a **queue-sharing group**
- Provide high availability in an IBM MQ infrastructure



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-22. What are shared queues?

Shared queues are available on the z/OS platform. Shared queues are **not** to be confused with the SHARE queue attribute, which determines whether more than one application instance can retrieve messages from the queue. Shared queues are not to be confused with clusters. Architecturally, they have significant differences, but they also have similarities.

Shared queues are used in queue-sharing groups. A queue-sharing group uses a z/OS component that is called a coupling facility to store messages. The queue managers in the queue-sharing group all have access to these messages. Unlike clusters, where messages in a failed queue manager are unavailable, if a queue manager fails in a queue-sharing group, the message is still available to the other queue managers members of the group. The message is kept in the coupling facility accessible to all queue managers. If a coupling facility fails, then queues in all queue managers are unavailable. However, coupling facilities mitigate the risk of failure by being configured with redundancy.

Coupling facilities can fill up. However, several techniques can be used to mitigate space consumption in a coupling facility. Queue-sharing groups and coupling facilities are covered in detail in course ***IBM MQ Advanced System Administration for z/OS***.

At the time this course was written, the course code for the z/OS course mentioned was WM312. However, course codes might change with new releases, so it is preferable to search for the course

title and obtain the newest release of the course. Although WM312 is for IBM MQ V8, all material about coupling facility and storage is current.

Unit summary

- Summarize the IBM MQ components
- Describe the role of a queue manager
- Describe an IBM MQ message
- Describe queues and identify what types of queues hold messages
- Describe the IBM MQ procedural application programming interface
- Describe precedence between queue and API attributes
- Distinguish among the various types of IBM MQ channels
- Explain how a channel is started by using triggering
- Describe the trajectory of a message that uses distributed message channels
- Identify IBM MQ troubleshooting resources
- Describe queue name resolution in a distributed message channel environment
- Distinguish between the point-to-point and publish/subscribe messaging styles
- Explain the impact of IBM MQ design and development considerations
- Introduce an IBM MQ cluster
- Explain IBM MQ Shared Queues

IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-23. Unit summary

Review questions

1. Select the correct answers. Which of the following queues can hold messages?
 - A. A remote queue
 - B. The dead-letter queue
 - C. A local queue
 - D. A transmission queue

2. Select the correct answers. You sent a message to a local queue in a remote queue manager. The expected recipient contacts you because no messages are received. How do you determine where the message is?
 - A. Determine whether the messages are in the transmission queue on your queue manager
 - B. Alter the queue definition
 - C. Display the channel status
 - D. Check the corresponding queue manager log for the cause of the problem



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-24. Review questions

Review answers (1 of 2)

1. Select the correct answers. Which of the following queues can hold messages?
 - A. A remote queue
 - B. The dead-letter queue
 - C. A local queue
 - D. A transmission queue

The answer is B, C, and D. The dead-letter queue and a transmission queue are all local queues. A remote queue points to a transmission queue but does not hold messages.



Review answers (2 of 2)

2. Select the correct answers. You sent a message to a local queue in a remote queue manager. The expected recipient contacts you because no messages are received. How do you determine where the message is?
 - A. Determine whether any messages are in the transmission queue on the sending queue manager side
 - B. Alter the queue definition
 - C. Display the channel status
 - D. Check the corresponding queue manager log for the cause of the problem

The correct answers are A, C, and D. If the channel is in stopped status, you might want to manually start the sender channel first. If the channel is inactive but expected to be triggered, check all aspects of the triggering configuration for the sender channel.



Exercise: Configuring and reviewing base IBM MQ resources

© Copyright IBM Corporation 2017

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

Figure 1-27. Exercise: Configuring and reviewing base IBM MQ resources

Exercise objectives

- Define and start IBM MQ queue managers
- Establish two-way distributed message channels between two queue managers
- Examine the channel status
- Locate the IBM MQ queue manager logs and the dead-letter queue
- Describe the queue manage cluster repository process



IBM MQ baseline

© Copyright IBM Corporation 2017

Figure 1-28. Exercise objectives

Unit 2. Before you start

Estimated time

00:30

Overview

This unit provides guidelines that are derived from actual engagement experiences to help with planning, design, implementation, and administration of a successful IBM MQ cluster environment. The unit emphasizes the role of cross organizational communication in the success of the cluster.

How you will check your progress

- Checkpoint

Unit objectives

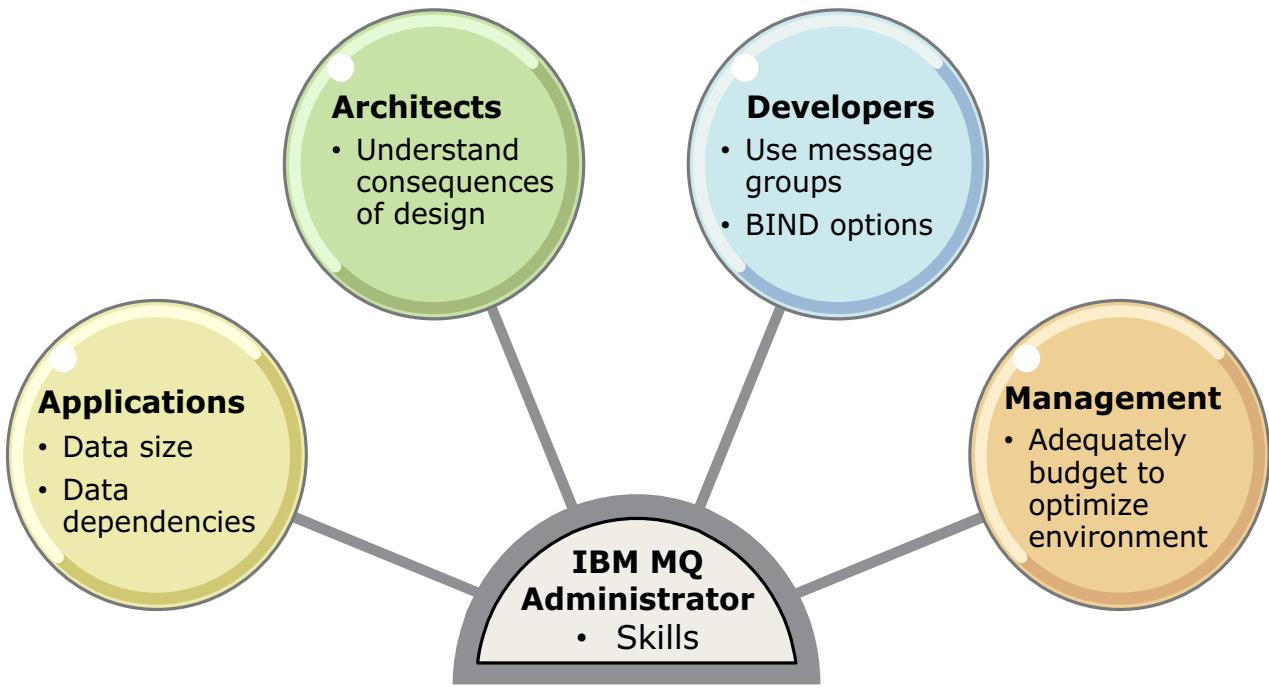
- Identify who in an organization impacts a cluster
- Describe the details that must be considered before designing and implementing a cluster
- Contrast the effort that is required to implement a cluster with distributed administration
- Identify the role of a cluster in a highly available IBM MQ infrastructure
- Describe workload management considerations
- Explain key considerations to observe as an IBM MQ administrator when introducing clusters to the environment

Before you start

© Copyright IBM Corporation 2017

Figure 2-1. Unit objectives

Everyone impacts the cluster



Before you start

© Copyright IBM Corporation 2017

Figure 2-2. Everyone impacts the cluster

In order for a cluster deployment to be successful, adequate communication must exist in the entire organization. Who in an organization has an impact in the cluster? Whether management, architects, or the IBM MQ administrator initiated implementation of the cluster, everyone in the organization has an impact in the cluster.

- Applications that connect to the cluster might have different message sizes. When an application has large messages, or an application segments messages that must arrive as a message group, it introduces affinities to a queue manager that lessen the workload balancing benefits of the cluster. It is important to have good communication with applications to ensure that the queue manager queues and channels can handle the messages. Applications with message affinities are not good candidates for the cluster.
- Organizations often have architects that design interfaces with other applications. However, the IBM MQ expertise of the architects might vary. Often it is the architects that design applications with affinities, such as segmenting messages that need to be retrieved as a group. Applications with segmented messages have affinities that negate the benefits of the cluster.
- Developers write the code and select the MQOPEN options. Developers might be using the option to BIND ON OPEN in the code and create an affinity. When the code standards are not enforced, developers can introduce affinities in the code.

- Management is critical in a successful cluster implementation. Management must ensure that standards are followed when it comes to clusters. It must also budget for dedicated full repository queue manager servers, at least for the production environment. However, it is preferred that at least an interim environment, such as quality assurance, is also configured to mirror production.
- IBM MQ administrators hear about all the problems with the cluster. They are the first to get a call when a problem occurs, or if the workload balancing does not seem to be taking place as expected. However, other than defining the queues with DEFBIND(NOTFIXED), the IBM MQ administrators might have little control over the environment. If IBM MQ administrators customize workload balancing options without adequately running a baseline of the base, unaltered IBM MQ environment, they might cause problems that are difficult to resolve in an outage or period closing situation.

A successful cluster starts with the involvement and communication of everyone in the organization.

IBM MQ administrative tasks

- Install, apply maintenance, and configure IBM MQ infrastructure
- Configure queue managers
- Create queues, clusters, channels, queue-sharing groups, and objects that users request
- Administer publish/subscribe topologies
- Administer security and SSL
- Back up data sets and object definitions
- Participate in performance analysis



- Monitor buffers and data set spaces
- Perform problem determination
- Participate in capacity planning
- Participate in infrastructure planning sessions
- Participate in naming standards

[Before you start](#)

© Copyright IBM Corporation 2017

Figure 2-3. IBM MQ administrative tasks

If you are an IBM MQ administrator, you are aware of your duties. Some of your activities are as follows:

- Apply maintenance. When you apply maintenance to a cluster, you should always keep the full repository queue managers at the higher maintenance level, or highest product level used in your organization. The reason is to ensure that the full repository queue managers can respond to new functionality in cluster queue managers. While the cluster might have mixed level queue managers, the full repositories must be able to handle the newest features available in your environment.
- The main consideration in your work with IBM MQ clusters must be to keep the environment as simple as possible. Most common organizations today see outsourced or partially outsourced environments. With the greater diversity of IBM MQ skills in the administrator community comes the greater need for simplicity. You do not want to have a cluster problem at a critical quarter-end or year-end period, or a system outage that is prolonged because an administrator cannot resolve a problem in a complex cluster. As the administrator, you must voice your concerns.
- Later in this unit you learn about security considerations in clusters, and about publish/subscribe clusters. You must be especially careful to understand the possible traffic implications of having a publish/subscribe cluster, as described in the later unit.

In the first slide of this unit, it mentioned who impacted the clusters. As the administrator with responsibility for the health of the cluster, you must be sure to be involved in all the discussions. Notice the potential issues that are listed on the right column, where a design or decision that jeopardizes the health of the cluster might be made in your absence.



CAUTION

If you are an IBM MQ administrator, you must question and document specifications that might negatively impact the cluster. For example, if an application has large message sizes, you must alter all the supporting SYSTEM and dead-letter queues to allow for the larger messages. In turn, larger messages challenge system resources. If you do not document the concerns, when an outage occurs due to a high volume of large messages, you must handle the problem. As an application architect and designer, you must be aware of the impact of large messages.

The large messages are only an example. Other specification requests might need to be questioned, such as allowing external organizations to join the cluster.

Before you start

- What is the strategic plan for the cluster?
- Is the staff trained to support all aspects of the cluster?
- Is adequate communication available with the IBM MQ administrator staff about standards for use of the cluster?
- Are there any known queue manager affinities in the applications?
- Is adequate communication available with development?
- Is a team identified to baseline workload balancing?
- What are the expected message sizes? If system support objects are clustered, do they need to change sizes?
- Are adequate servers available to hold full cluster repository queue managers and gateways?
- Does the decision include consideration of queue-sharing groups?
- Are adequate failover solutions identified?

Before you start

© Copyright IBM Corporation 2017

Figure 2-4. Before you start

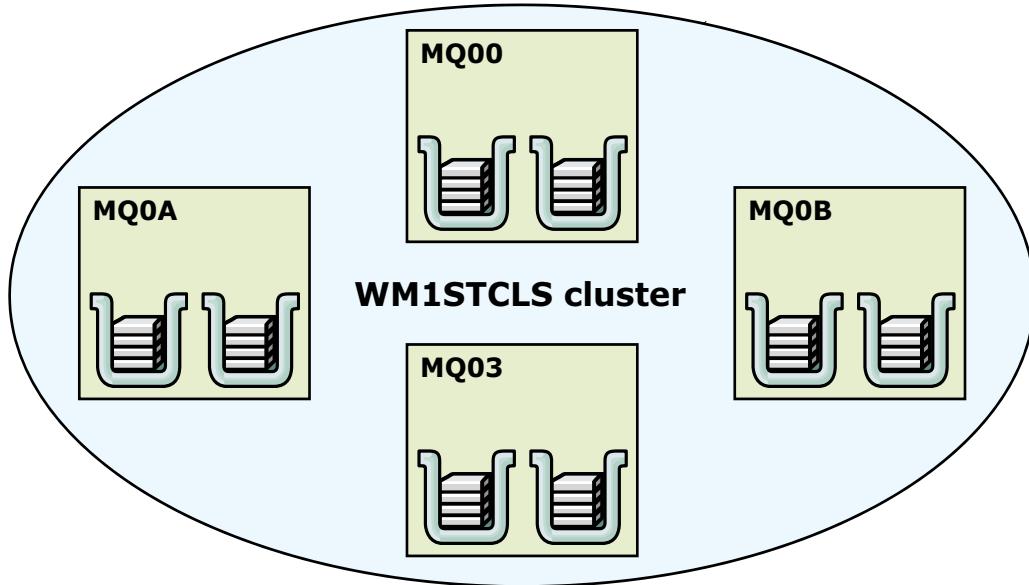
Whether your organization does not yet implement clusters, or already has a cluster environment, you must keep the questions in this slide in mind.

Introduction of a cluster in an organization is an interesting event. After the cluster is introduced, those resources that opposed used of clusters will blame the cluster for any mishap in the environment. It is best to have a documented, pragmatic approach.

- The reason for a cluster might be to save on the administration effort. If the administrative staff is not fluent with cluster work, expect a learning curve period during which the administration effort is more difficult until adequate experience with clusters is acquired.
- Written standards are critical. Among some of the items you must document are:
 - How outside users can connect to the cluster. You do not want to allow outside users to join the cluster, as it can create unexpected problems. In one organization the monitoring system was picking up queue manager outages from an outside company, and sending out pages to the administrative staff.
 - Whether the full repositories must be dedicated, and the maintenance process for the cluster.
 - Application and architecture standards that mitigate the risk of affinities.
 - Failover requirements for cluster queue managers.

Distributed queuing versus clustered configuration (1 of 3)

- Definitions that are required in a network with four queue managers, each with two local queues in the configuration



Before you start

© Copyright IBM Corporation 2017

Figure 2-5. Distributed queuing versus clustered configuration (1 of 3)

This set of three slides shows comparisons on the numbers of definitions that are required for distributed message channels and cluster channels.

Although the definition effort in cluster channels is substantially less, the initial added complexity of managing a cluster must be considered.

Distributed queuing versus clustered configuration (2 of 3)

- Definitions that are required for distributed queuing

Definitions	Number per queue manager	Total number
Sender channel on which to send messages to every other queue manager	3	12
Receiver channel on which to receive messages from every other queue manager	3	12
Transmission queue for a transmission queue to every other queue manager	3	12
Local queue for each local queue	2	8
Remote queue for each remote queue to which this queue manager wants to put messages	6	24

Before you start

© Copyright IBM Corporation 2017

Figure 2-6. Distributed queuing versus clustered configuration (2 of 3)

The figure shows the number of definitions that are required to use distributed queuing with four queue managers and two application queues on each queue manager.

To send and receive messages with the other queue managers in the network, you would need to define three sender channels and three receiver channels on each queue manager, a total of 24 channel definitions.

You would also need to create three transmission queues for each remote queue manager and six remote queues.

The maximum number of definitions can be as many as 17 on each queue manager, which is a total of 68 for this network.

Distributed queuing versus clustered configuration (3 of 3)

When using clusters, you need:

- Just one CLUSSDR and one CLUSRCVR definition at each queue manager
- No separately defined transmission queues
- No remote-queue definitions

Definitions	Number per queue manager	Total number
Cluster-sender channel on which to send messages to a repository queue manager	1	4
Cluster-receiver channel on which to receive messages from other queue managers in the cluster	1	4
Local queue for each local queue	2	8

Figure 2-7. Distributed queuing versus clustered configuration (3 of 3)

If queue managers are grouped in a cluster, the queue managers can make the queues that they host available to every other queue manager in the cluster. Any queue manager can send a message to any other queue manager in the same cluster without explicit channel definitions, remote-queue definitions, or transmission queues for each destination. Every queue manager in a cluster has a single transmission queue from which it can transmit messages to any other queue manager in the cluster.

This figure shows the number of definitions that are required when you use clustering to support the simplified administration example.

Each queue manager in a cluster needs to define only:

- One cluster-receiver channel on which to receive messages
- One cluster-sender channel that points to one of the full repository queue managers

The cluster-sender and cluster-receiver channel definitions are made one time. When the cluster is in place, you can add or remove queue managers (other than the repository queue managers) without any disruption to the other queue managers. This process significantly reduces the number of definitions that are required to set up a network that contains many queue managers.

To set up this cluster of queue managers (with two full repositories), you would need four definitions on each queue manager; a total of 16 definitions. You would also need to alter the queue manager

definitions for two of the queue managers to make them full repository queue managers for the cluster.

The general expectation is that with fewer definitions to make, the possibility for errors should decrease.

IBM MQ objectives for highly available architectures

Objectives	IBM MQ cluster	Failover solution	Shared queues
Immediate availability for new requests			 * If coupling facility and storage are adequately sized
Access to marooned messages			

- A cluster by itself is not a high-availability solution
- Shared queues can be part of a cluster
- **Caveat:**
 - Shared queues depend on the health of the coupling facility
 - Chart assumes that the coupling facility is configured with redundancy

Before you start

© Copyright IBM Corporation 2017

Figure 2-8. IBM MQ objectives for highly available architectures

IBM MQ clusters contribute to high availability by adding scalability for new messages. However, as stated earlier, if a cluster queue manager fails, the messages in the failed queue manager are unreachable, or “marooned”, until a failover solution for the unavailable queue manager takes effect.

It is key to realize that a cluster is one component of a highly available IBM MQ infrastructure. It must be coupled with adequate capacity planning, and a high availability solution that allows queue managers to fail over with shared disk.

Shared queues, available for z/OS queue managers, are a good addition to highly available IBM MQ infrastructures. However, the caveat is to ensure that:

- **Redundancy is built into the coupling facility.** If the coupling facility fails without redundancy, none of the queue managers in the queue-sharing group can access any messages.
- **Adequate capacity planning is done to implement capabilities to prevent the coupling facility from becoming full, which is done in two ways:**
 - By implementing offload of messages to shared message data sets, or SMDS.
 - By allowing overflow of unused messages to a storage class memory, or SCM.

The course that is titled *IBM MQ Advanced System Administration for z/OS* contains two units to help you set up queue-sharing groups and understand the offload and overflow technologies. Since

the course codes change, you can search for the course by name by replacing the “x” with the IBM MQ version. At the time that this course was written, the most recent course is for V8, code WM312. At the time that this course was written, the V8 course *is* applicable to the V9 environment.

Applications and workload balancing

- Use of workload algorithm depends on MQOPEN or MQPUT
 - Queue manager name included in MQOD: Messages go to specified queue manager
 - Queue manager name not included in MQOD: Queue manager decides
- When the queue manager decides:
 - Always put to local queue manager if it hosts the queue (unless CLWLUSEQ is set to ANY)
 - If the clustered queue is not local, a round-robin workload management algorithm is used:

MQOPEN	When cluster workload management algorithm used
MQOO_BIND_ON_OPEN	Workload algorithm is invoked when MQOPEN issued
MQOO_BIND_NOT_FIXED	Workload algorithm is invoked for every MQPUT
MQOO_BIND_ON_GROUP	Workload algorithm is invoked at start of a message group
MQOO_BIND_AS_Q_DEF	Default is to use queue DEFBIND attribute

- Applications with large messages that require message segmentation and use of message groups also create queue manager affinities

Caveat: After the cluster is implemented, baseline the application workload distribution

Before you start

© Copyright IBM Corporation 2017

Figure 2-9. Applications and workload balancing

In a subsequent unit, you learn about settings that influence workload balancing behavior. A good route to take is:

- Baseline the existing environment before introducing any changes
- Keep the environment as simple as possible, and avoid any unneeded settings
- Consider the ability to support the changes in production

Unless compelling reasons exist, a simple approach without altering the default queue manager settings is best when starting to implement a cluster workload balancing solution. Altering the workload balancing attributes without adequate analysis or governance can lead to situations that are complex to adjust or rectify.

A valid reason to alter the defaults might exist, such as having a server with more or less capacity than another server. Even in such a case, a baseline of the message distribution of the existing applications should always be scheduled before changing any settings. It is important to confirm, for example:

- Whether any applications have affinities
- Whether any applications are inadvertently using the MQOO_BIND_ON_OPEN option
- Whether queues are inadvertently using DEFBIND=OPEN and applications are using the MQOO_BIND_AS_Q_DEF option, which would bind messages to the same queue

- And many other factors that might skew the distribution

When conducting a baseline, it is important to note that even if a queue definition has the DEFBIND attribute set to NOTFIXED, if the application uses the MQOO_BIND_ON_OPEN option, it overrides the queue definition. That is, the messages will be bound to the same queue manager until the next MQOPEN.

If an attribute is defined on a QALIAS definition, it overrides attributes that are defined in a clustered QLOCAL queue.

Default behaviors for queue attributes and application development are:

- Queue definition: DEFBIND=OPEN
- Application: MQOO_BIND_AS_Q_DEF

You learn more about affinities in a later unit.

IBM MQ administration settings and workload balancing

- The algorithm is further influenced from specific object attributes and by the use of custom exits

Queues	Channels	Queue managers
Priority: CLWLPRTY	Priority: CLWLPRTY, NETPRTY	CLWSUSEQ
Rank: CLWLRANK	Rank: CLWLRANK	CLWLMRUC
Use remote: CLWLUSEQ	Channel weight: CLWLWGHT	Availability status
PUT enabled or disabled	Channel status	

- Caveats:
 - Baseline the workload distribution before you make any changes
 - Make one change at a time, and rebaseline the distribution

[Before you start](#)

© Copyright IBM Corporation 2017

Figure 2-10. IBM MQ administration settings and workload balancing

The changes that can be made to the cluster workload balancing algorithm at the queue, channel, and queue manager levels are summarized in this slide. These attributes are detailed in a later unit.

As with any other change to the environment, the behavior of the existing applications should be ascertained before making any changes. If changes are made, introduce one change at a time to maintain better control of your test results.

You learn more about object settings that influence workload balancing in a later unit.

Clusters and IBM MQ administration

- Apply maintenance to the full repository queue managers first
- Simplicity is the best approach
 - Design of the cluster
 - Use of extra transmission queues
 - Workload balancing options
- Baseline an application workload distribution baseline before implementing changes
- Avoid allowing outside users to join the cluster
- Ensure that you understand the cluster IBM MQ security considerations

Before you start

© Copyright IBM Corporation 2017

Figure 2-11. Clusters and IBM MQ administration

In this unit, you started by looking at the many different resources that impacted the success of a cluster implementation. When it comes to clusters, adequate communication and involvement at all levels of the organization are key.

This slide summarizes other administrative considerations that are mentioned in this unit. Keep the guidelines in this unit handy when you need to evaluate changes or work with your cluster environment.

Unit summary

- Identify who in an organization impacts a cluster
- Describe the details that must be considered before designing and implementing a cluster
- Contrast the effort that is required to implement a cluster with distributed administration
- Identify the role of a cluster in a highly available IBM MQ infrastructure
- Describe workload management considerations
- Explain key considerations to observe as an IBM MQ administrator when introducing clusters to the environment

Before you start

© Copyright IBM Corporation 2017

Figure 2-12. Unit summary

Review questions

1. True or False: A cluster contributes to high availability by providing scalability for new requests.
2. Shortly after you implement a basic cluster without changing any attributes that influence workload, an application contacts you because the workload is not being evenly distributed. What steps do you take?
 - A. Check your queue definitions for BIND_ON_OPEN attribute
 - B. Look in the IBM MQ log
 - C. Ask the application what attributes it uses in the MQOPEN calls
 - D. Ask the application whether it uses segmented messages that are processed in a group
3. True or False: Use of clusters should always be considered to simplify IBM MQ administration.
4. What is the best general rule to follow when introducing clusters in the environment?



Before you start

© Copyright IBM Corporation 2017

Figure 2-13. Review questions

Review answers (1 of 2)

1. True or False: A cluster contributes to high availability by providing scalability for new requests.
The answer is True. However, a cluster must be coupled with a high-availability solution to access messages that might be marooned in a failed queue manager.
2. Shortly after you implement a basic cluster without changing any attributes that influence workload, an application contacts you because the workload is not being evenly distributed. What steps do you take?
 - A. Check your queue definitions for BIND_ON_OPEN attribute
 - B. Look in the IBM MQ log
 - C. Ask the application what attributes it uses in the MQOPEN calls
 - D. Ask the application whether it uses segmented messages that are processed in a group

The answer is A, B, and C. If you did not change any workload-influencing attributes, you should check the definitions and obtain information from the application.



Before you start

© Copyright IBM Corporation 2017

Figure 2-14. Review answers (1 of 2)

Review answers (2 of 2)

3. True or False: Use of clusters should always be considered to simplify IBM MQ administration.

The answer is False. Clusters might be challenging to the administration staff when first introduced until they are past the learning curve.



4. What is the best general rule to follow when introducing clusters in the environment?

The answer is simplicity.

Before you start

© Copyright IBM Corporation 2017

Figure 2-15. Review answers (2 of 2)

Unit 3. Understanding and implementing an IBM MQ cluster

Estimated time

01:30

Overview

In this unit, you review the components of a cluster. You then learn how to implement a basic cluster, verify its configuration, and resolve any problems. You also learn how to configure a cluster gateway.

How you will check your progress

- Checkpoint
- Machine exercises

Unit objectives

- Review a basic IBM MQ cluster
- Describe the components that support an IBM MQ cluster
- Distinguish between the definitions that are required to set up a cluster and distributed message channel definitions
- Describe how to implement a basic IBM MQ cluster
- Explain how to verify a new cluster configuration
- Describe how to add queues to a cluster
- Describe the trajectory of a message that uses a basic cluster
- Explain how to set up a cluster gateway by configuring a queue manager external to the cluster to route messages to cluster queues
- Summarize the administrative options that can be used to manage a cluster

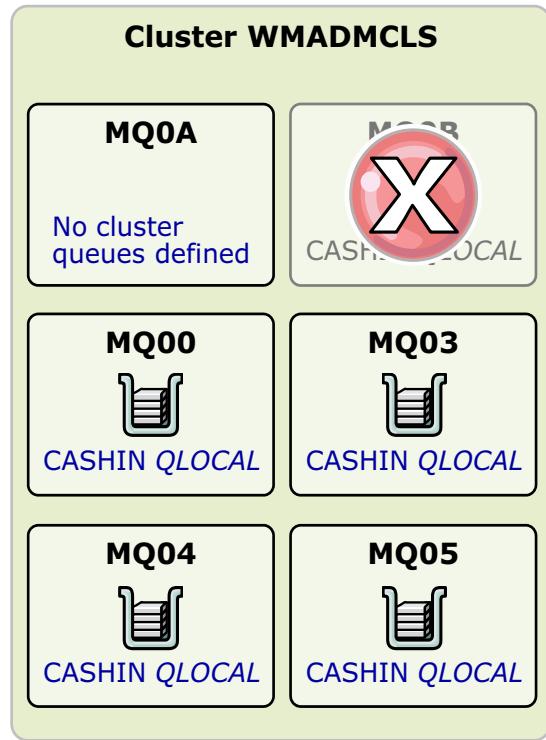
[Understanding and implementing an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 3-1. Unit objectives

IBM MQ clusters

- Simplified administration
 - Reduce number of remote queues, transmit queues, and channel definitions
- Workload balancing
 - Same queue can be hosted in several queue managers
 - Route around failures
- Scalable
- Contribute to high availability
- MQPUT to local queue manager if queue exists locally
 - If the queue is not found locally, an algorithm is used for target queue selection
- MQGET done to local queue manager
- Publish/subscribe can use clusters



Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-2. IBM MQ clusters

This unit starts with a look at the chart that you saw earlier. You learned that a cluster contributes to high availability by providing scalability for new requests. You also learned that a clustered environment requires fewer definitions than distributed message channels. You now look at the components that support clusters.

In this unit, you learn to configure and verify a basic cluster, and also a gateway queue manager. Other configuration steps, such as use of separate transmission queues, are presented in later units.

IBM MQ cluster components

- Repositories
 - Collection of information about the cluster
 - Cluster queue managers can hold a full or a partial repository
- Queue manager SYSTEM.CLUSTER support queues
- CLUSRCVR channels
- CLUSSDR channels
 - Point to one full repository
 - Communicate any cluster-related changes to the full repository
- Dynamic CLUSSDRx channels
 - CLUSSDRA
 - CLUSSDRB
- Transmission queues
 - SYSTEM.CLUSTER.TRANSMIT.QUEUE
 - Alternative cluster transmit queue
- Clustered queues

[Understanding and implementing an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 3-3. IBM MQ cluster components

IBM MQ clusters rely on several components. This slide omits a most important one, which is the cluster or repository manager. In distributed systems, the repository is a process that is called amqrrmfa. In z/OS, the repository manager is part of the channel initiator. The following aspects are important:

- Repositories:
 - The repository is where all information about the cluster is stored.
 - Repositories have two types:
 - A full repository, which is held by a queue manager with the REPOS queue manager attribute set to the cluster name.
 - A partial repository, which is held by a member queue manager. The full repositories are critical. Two queue managers should always be designated as full cluster repositories, and the CLUSSDR channels for these two queue managers should point to each other.
 - Full repositories have information about the entire cluster, while partial repositories keep information about those objects that are of interest to what they need to process.
 - If one full repository is available, it is possible for a cluster to continue functioning. That is, although the optimal case is to have the two full repository queue managers available, it is

not essential. If one queue manager is brought down for maintenance, the cluster is able to operate with the remaining full cluster repository queue manager active.

When a member queue manager defines a queue, it tells the full repository that it connects to, about this new queue.

- SYSTEM.CLUSTER.* support objects. You learn about these objects in a later slide.
- Cluster channels:
 - Understanding the channels is key to learning about clusters. Two channel definitions are cluster-related:
 - CLUSRCVR channels. Unlike distributed RCVR channels, the CLUSRCVR has the connection information. This CLUSRCVR connection information is what cluster member queue managers use to create a dynamic CLUSSDRx channel to the queue manager that owns the CLUSRCVR. The dynamic CLUSSDR channel is of type CLUSSDRA or CLUSSDRB. That is, when using clusters, a member queue manager that just joined a cluster checks the repository for the information on a queue manager that it needs to send messages to. It then uses the connection name in the CLUSRCVR to create a dynamic channel to the required queue manager.
 - CLUSSDR channels. A queue manager that joins the cluster should have one CLUSSDR channel, and only one CLUSSDR channel, which is pointing to **one** of the two full cluster repository queue managers.
 - Thus, each queue manager that joins a cluster has one CLUSRCVR and one CLUSSDR channel, and can create dynamic **CLUSSDRx** channels to other cluster member queue managers.

What is the “x” in **CLUSSDRx**?

- A local queue manager might create a dynamic channel to a remote queue manager, and the CLUSSDR in that local queue manager is **not** pointing to the target remote queue manager. In this case, the CLUSSDR channel is displayed as a **CLUSSDRA**.

When this local queue manager has a CLUSSDR channel that points to the target remote queue manager, the dynamic channel that is created to that remote queue manager where CLUSSDR is pointing displays as **CLUSSDRB**.

The SYSTEM.CLUSTER queues

- On distributed platforms, the set of objects are automatically created when the queue manager is defined
- On z/OS, the default cluster object definitions are in the customization samples

- SYSTEM.CLUSTER.REPOSITORY.QUEUE holds persistent view of repository cache
- SYSTEM.CLUSTER.COMMAND.QUEUE holds inbound administrative messages
- SYSTEM.CLUSTER.TRANSMIT.QUEUE holds outbound administrative and user messages
- SYSTEM.CLUSTER.HISTORY.QUEUE stores the history of cluster state information for service purposes

[Understanding and implementing an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 3-4. The SYSTEM.CLUSTER queues

This slide shows the SYSTEM.CLUSTER queues.

- The SYSTEM.CLUSTER.REPOSITORY queue, as its name implies, keeps the cluster repository. This queue should never be empty for an active cluster.
- The SYSTEM.CLUSTER.COMMAND.QUEUE receives commands and information that the repository manager processes, and it should normally be empty. When messages accumulate in this queue, it is a good indication that the repository manager is not picking them up. Therefore, if the cluster has any problems, you know to check the started channel initiator task of z/OS for any errors. If you are working with a distributed operating system, you check whether the amqrrmfa process is running. Messages that are written to this queue contain updates to the repository data that applies to the local copy of the repository, or requests for repository data.
- The SYSTEM.CLUSTER.TRANSMIT.QUEUE is the transmission queue for all destinations in the cluster.
- The SYSTEM.CLUSTER.HISTORY.QUEUE stores the history of cluster state information for service purposes.

When you work across distributed and z/OS platforms, it is a good habit to check the IBM Knowledge Center for subtle differences in the initial default and use of the cluster system queues.

How long is cluster information kept in a repository?

- Full repository queue managers
 - While it exists on active queue manager
 - Participating queue managers notify repository every 27 days about activity (resources used)
 - Inactive items that are marked for deletion after 30 days; actual deletion another 60 days later
- Partial repository (participating) queue managers
 - While it is actively used
 - Check for usage after 27 days
 - Inactive items are removed after 30 days; stay in the repository for extra 60 days

[Understanding and implementing an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 3-5. How long is cluster information kept in a repository?

In this slide, you learn about the lifecycle of information in cluster repositories. The topic starts with a baseline on the number of repositories to use.

The optimal, suggested use of repositories in a cluster is two. It is possible to have more than two full repositories. However, the number of repositories that are used at a time is two. Use of extra repositories does not improve the health of a cluster, and if one of the active repository queue managers stops and an inactive repository queue manager starts, it might cause confusion. As you see in this chart, information in a full repository is kept for a few days. Thus, if it is necessary to stop one of the two full repository queue managers, a cluster can temporarily run on a single full repository.

A good practice, particularly in busy environments, is to have a dedicated queue manager for each repository, that is, a queue manager that does not handle application work. A dedicated queue manager mitigates problems where the application problems might jeopardize a full repository queue manager.

The information in full or partial repositories is kept for a predetermined time. If the queues used by full and partial repositories do not get used (referenced) within a specific time, the interest is removed from the repository. The amount of time varies depending on the role of the queue manager as a full or partial repository.

Full repository queue managers

When a queue manager sends information about itself, the full repositories store the queue manager information with an expiry of 30 days. All queue managers in the cluster automatically resend all information about themselves after 27 days. If 60 more days elapse without contact from a queue manager, that queue manager's information is removed from all repositories; the queue manager is no longer part of the cluster.

The 90-day time period allows for the possibility that a queue manager might be disconnected due to operational or network problems. If it is still within the 90-day period, it is possible for a queue manager that is removed to rejoin the cluster by reconnecting to the network.

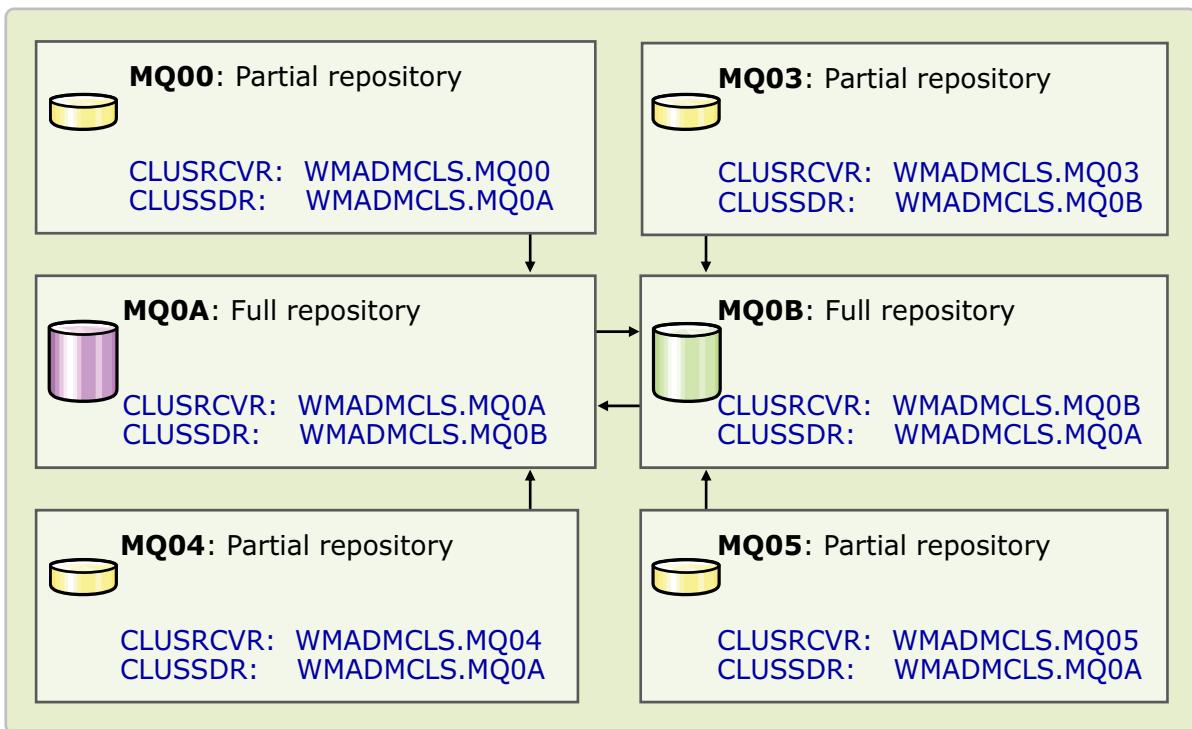
Partial repository queue managers

With a partial repository, a subscription is not reissued after 30 days without reference.

For example, if queue manager QM04 opened the queue that is called Q1 for output, its partial repository gets updated with the appropriate queue and routing information. The full repositories also know that QM04 is interested in the queue called Q1. After 27 days of the initial registration of interest, QM04 checks whether more references were made to Q1 during that time. If more references were made, an automatic update is done, and the information is refreshed from the repository. This process notifies the full repository that QM04 is still interested and ensures that QM04 does not lose its information about Q1.

If no additional references are made to Q1 in the 27 days, IBM MQ does not contact the full repository or refresh the local repository. After 30 days, the definition is allowed to expire from the partial repository. This process helps to keep the amount of information in the partial repository at a minimum and also reduces the number of requests that are made for information about dormant queue managers.

Basic queue manager cluster



Note: Arrows represent the full repository that CLUSSDR channels point to

Figure 3-6. Basic queue manager cluster

Before defining a cluster, it is important to draw a plan and determine what queue managers host the full cluster repositories. For the WMADMCLS cluster, MQ0A and MQ0B are designated as full cluster repositories. As full cluster repositories, MQ0A and MQ0B have:

- The queue manager REPOS attribute set to the cluster name WMADMCLS
- The CLUSRCVR channel that points to their own connection information
- **One** CLUSSDR channel for each, which points to the other

The rest of the queue managers that join the cluster with partial repositories have:

- The CLUSRCVR channel that points to their own connection information
- One CLUSSDR channel that points to **one** of the two full repository queue managers
- No changes to the queue manager object

The naming convention that is used for the channels includes the cluster name as the first node. This naming convention is one of the cluster “best practices.”

Queue manager cluster definition

At MQ0A: Full repository

```
ALTER QMGR REPOS (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0A) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)') cluster (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0B) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)') cluster (WMADMCLS)
```

At MQ0B: Full repository

```
ALTER QMGR REPOS (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0B) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)') cluster (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0A) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)') cluster (WMADMCLS)
```

At MQ00: Partial repository

```
DEFINE CHL (WMADMCLS.MQ00) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1600)') CLUSTER (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0A) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)') CLUSTER (WMADMCLS)
```

At MQ03: Partial repository

```
DEFINE CHL (WMADMCLS.MQ03) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1603)') CLUSTER (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0B) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)') CLUSTER (WMADMCLS)
```

Figure 3-7. Queue manager cluster definition

If you created the cluster in the previous slide, these definitions would be the ones for queue managers MQ0A, MQ0B, MQ00, and MQ03.

Should there be an issue with large messages that come across the cluster, you need three other definitions to add to each queue manager. Alter the message size according to your needs.

- Assuming that `SYSTEM.DEAD.LETTER.QUEUE` is your designated DLQ, and you are not using separate cluster transmit queues:

```
ALTER QMGR MAXMSGL(104857600)
ALTER QL(SYSTEM.DEAD.LETTER.QUEUE) MAXMSGL(104857600)
ALTER QL(SYSTEM.CLUSTER.TRANSMIT.QUEUE) MAXMSGL(104857600)
```

- These additional definitions should not be added unless they are needed because the larger sizes consume more system resources for the channels.

When the cluster definitions are completed, you always need to thoroughly check the results by using the `DIS CLUSQMGR` command. You now walk through the verification or checking process.

MQ0A definition check: What happens in the cluster (1 of 2)

/MQ0A DIS CLUSQMGR(*) ALL (partial display)

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ0A) _____

This definition is for the MQ0A queue manager itself

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0A)

QMID (MQ0A.CDA33A44AE870B2B)

DEFTYPE (CLUSRCVR) _____

CLUSRCVR channel type definition for MQ0A

QMTYPE (REPOS) _____

QMTYPE REPOS: This queue manager (MQ0A) holds a full repository

...

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ0B) _____

The partner queue manager for this MQ0A connection is MQ0B

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0B)

QMID (MQ0B.CDA2735757A9FEA9)

DEFTYPE (CLUSSDRB) _____

A CLUSSDRB is the combination of both MQ0B's CLUSRCVR and MQ0A's CLUSSDR; MQ0A defined CLUSSDR to MQ0B

...

QMTYPE (REPOS) _____

The MQ0B queue manager holds the second full repository

...

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-8. MQ0A definition check: What happens in the cluster (1 of 2)

From one of the full cluster repository queue managers, you issue the `DIS CLUSQMGR(*) ALL` command. You use MQ0A. The output from the command spans two slides. What do you check for?

- Immediately after the cluster is created, you expect to see the cluster channels with a `RUNNING` status. For the MQ0A display, it looks good.
- If any errors have a “`SYSTEM TEMP .xxx`” name as the object, it has a problem. These error entries are usually:
 - `CLUSQMGR (SYSTEM TEMPQMGR.xxxx.xx.xxxx)`
 - `QMID (SYSTEM TEMPUUID.xxxx.xx.xxxx)`
 - No `CLUSSDR` type channels should display. They should all be `CLUSSDRA` or `CLUSSDRB`. A `CLUSSDR` in the `DIS CLUSQMGR` indicates a problem.
- So far all looks well. Also, remember that when an error is made in the connectivity information of the cluster channel, the correction is similar to distributed channels. Therefore, if you see one of these `SYSTEM TEMP .xxxx.xx.xxxx` messages, the first place to check is the `CONNNAME` attribute for the `CLUSRCVR` and the `CLUSSDR`.

Review the notes that are embedded in the slide for the selected `DIS CLUSQMGR` attributes. By the end of the display in this slide, you know that the cluster connection from MQ0A to MQ0B is good and that MQ0A and MQ0B are both full repositories (QMTYPE - REPOS). Also, MQ0B created a dynamic CLUSSDRB channel to MQ0A, which is in RUNNING status. Why is this CLUSSDR channel type from MQ0A to MQ0B a CLUSSDRB?

If you remember the diagram for this cluster, MQ0A chose queue manager MQ0B as its full repository by setting the MQ0A CLUSSDR channel to connect to MQ0B. Therefore, the channel is both dynamic and predefined, and the “B” type is appended to CLUSSDR in the `DIS CLUSQMGR` command.

MQ0A definition check: What happens in the cluster (2 of 2)

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ00)	The partner queue manager for this MQ0A connection is MQ00
CLUSTER (WMADMCLS)	
CHANNEL (WMADMCLS.MQ00)	
QMID (MQ00.CDA33FC1E2935036)	
DEFTYPE (CLUSSDRA)	CLUSSDRAs show a dynamic connection from MQ0A to MQ00's CLUSRCVR; MQ0A has no defined CLUSSDR to MQ0A
...	
QMTYPE (NORMAL)	The MQ00 queue manager holds a partial repository
STATUS (RUNNING)	
VERSION (08000000)	
XMITQ (SYSTEM.CLUSTER.TRANSMIT.QUEUE)	
TRPTYPE (TCP)	
CONNNAME (mvsmc11.ilsvpn.ibm.com(1600))	
...	

/MQ0B DIS CLUSQMGR (*) ALL output is similar to MQ0A and bypassed for brevity

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-9. MQ0A definition check: What happens in the cluster (2 of 2)

You now look at the rest of the DIS CLUSQMGR display from the MQ0A perspective. In this part of the output you see that:

- The information is what cluster queue manager MQ0A learns about the MQ00 cluster queue manager.
- Channel WMADMCLS.MQ00 is a dynamically defined CLUSSDRA type channel from MQ0A to MQ00. This information indicates that queue manager MQ0A learned how to find MQ00 from the information in the repository because the CLUSSDR for MQ0A points to MQ0B.
- Since MQ0A has no CLUSSDR that is defined to MQ00, the DEFTYPE for the dynamic channel that MQ0A creates to MQ00 is a CLUSSDRA type channel.
- The information that MQ0A has about MQ00 also indicates that MQ00 does not hold a full repository, and the QMTYPE shown for MQ00 is NORMAL, not REPOS.

You are finished looking at the output of DIS CLUSQMGR issued from MQ0A. The display from MQ0B is skipped for brevity, as it is similar to MQ0A. It is the view from a full repository queue manager that is pointing to another full repository.

But where is the information for the MQ03 queue manager? You see this information in a later slide.

MQ00 definition check: What happens in the cluster (1 of 2)

```
/MQ00 DIS CLUSQMGR(*) ALL (partial display)
CSQM201I MQ00 CSQMDRTC  DIS
  CLUSQMGR (MQ0A) _____ The partner queue manager for this MQ00
  CLUSTER (WMADMCLS) _____ connection is MQ0A
  CHANNEL (WMADMCLS.MQ0A) _____
  QMID (MQ0A.CDA33A44AE870B2B) _____ A CLUSSDRB is the combination of both
  DEFTYPE (CLUSSDRB) _____ MQ0A's CLUSRCVR and MQ00's CLUSSDR; MQ00 has a defined CLUSSDR to MQ0A
  ...
  QMTYPE (REPOS) _____
  CLUSDATE (2014-09-08) _____
  STATUS (RUNNING) _____
  VERSION (08000000) _____
  XMITQ (SYSTEM.CLUSTER.TRANSMIT.QUEUE) _____
  TRPTYPE (TCP) _____
  CONNAME (mvsmc11.ilsvpn.ibm.com(1621)) _____
```



```
CSQM201I MQ00 CSQMDRTC  DIS
  CLUSQMGR (MQ0B) _____ The partner queue manager for this MQ00
  CLUSTER (WMADMCLS) _____ connection is MQ0B
  CHANNEL (WMADMCLS.MQ0B) _____
  QMID (MQ0B.CDA2735757A9FEA9) _____ CLUSSDRAs show a dynamic connection
  DEFTYPE (CLUSSDRA) _____ from MQ00 to MQ0B's CLUSRCVR; MQ00 has no defined CLUSSDR to MQ0B
  ...
  QMTYPE (REPOS) _____
  STATUS (RUNNING) ... _____ The MQ0B queue manager holds the
                           second full repository
```

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-10. MQ00 definition check: What happens in the cluster (1 of 2)

You now issue `DIS CLUSQMGR(*) ALL` in the MQ00 queue manager. This queue manager has a partial repository, and its CLUSSDR channel points to MQ0A. Currently, you know that:

- The WMADMCLS.MQ0A channel that you see in this image should be DEFTYPE CLUSSDRB. The CLUSSDR channel for MQ00 selected MQ0A as its full cluster repository, so the dynamic channel from MQ00 to MQ0A should be a CLUSSDRB. You can see that this statement checks out.
- Dynamic cluster-sender channels to MQ0B and MQ03 should be type CLUSSDRA because MQ00 does not have any explicitly defined CLUSSDR channels to MQ0B or MQ03. This statement checks out for MQ0B, but where is MQ03?

The CLUSRCVR channel for MQ00 should point to itself. Check the next slide.

MQ00 definition check: What happens in the cluster (2 of 2)

CSQM201I MQ00 CSQMDRTC DIS ...

CLUSQMGR (MQ00)

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ00)

QMID (MQ00.CDA33FC1E2935036)

DEFTYPE (CLUSRCVR)

This definition is for the MQ00 queue manager itself

...

QMTYPE (NORMAL)

CLUSDATE (2014-09-08)

CLUSTIME (09.06.23)

SUSPEND (NO)

VERSION (08000000)

TRPTYPE (TCP)

CONNNAME (mvsnc11.ilsvpn.ibm.com(1600))

...

CLUSRCVR channel type definition for MQ00

The MQ00 queue manager holds a partial repository

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-11. MQ00 definition check: What happens in the cluster (2 of 2)

As expected, the CLUSRCVR for MQ00 shows it pointing to itself. But the MQ03 queue manager has no entries.

You now go to MQ03 and issue the DIS CLUSQMGR(*) ALL command.

MQ03 definition check: Errors, TMPQMGR, TMPUUID entries

```
/MQ00 DIS CLUSQMGR(*) ALL (partial display)
CSQM201I MQ03 CSQMDRTC  DIS CLUSQMGR DETAILS 636
```

CLUSQMGR (MQ03) ————— This definition is for the MQ03 queue manager itself
 CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ03)

QMID (MQ03.CDB68FBADA5634A2)

DEFTYPE (CLUSRCVR)

QMTYPE (NORMAL)

...

CONNNAME (mvsmc11.ilsvpn.ibm.com(1603))

...

```
CSQM201I MQ03 CSQMDRTC  DIS CLUSQMGR DETAILS 637
```

CLUSQMGR (SYSTEM.TEMPQMGR.mvsmc11.ilsvpn.ibm.com(1622))

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0B)

QMID (SYSTEM.TMPUUID.mvsmc11.ilsvpn.ibm.com(1622))

DEFTYPE (CLUSSDR)

QMTYPE (REPOS) ————— Channel DEFTYPE should be a CLUSSDRB to MQ0B

...

STATUS (INACTIVE)

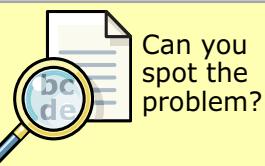
XMITQ (SYSTEM.CLUSTER.TRANSMIT.QUEUE)

TRPTYPE (TCP)

CONNNAME (mvsmc11.ilsvpn.ibm.com(1622))

...

Understanding and implementing an IBM MQ cluster



Always check
new cluster queue
managers before
proceeding

© Copyright IBM Corporation 2017

Figure 3-12. MQ03 definition check: Errors, TMPQMGR, TMPUUID entries



Important

Ensure that you do not have any `SYSTEM.TEMPQMGR.xxx` or `SYSTEM.TMPUUID.xxx` entries in the output of the `DIS` command for any of the queue managers in the cluster. Such entries constitute a configuration error that must be rectified.

You issue the `DIS CLUSQMGR` for MQ03. Note what happens:

- MQ03 is directing its CLUSSDR channel to the MQ0B full repository, but the display for the CLUSSDR* QMTYPE has three obvious problems:
 - CLUSQMGR indicates a `SYSTEM.TEMPQMGR.xxx` name, a red flag.
 - QMID also indicates `SYSTEM.TMPUUID.xxx` name, another red flag.
 - The channel DEFTYPE shows `CLUSDR`, another red flag. It should be a `CLUSSDRB` because of an explicitly defined CLUSSDR to the full repository, MQ0B.

Your first course of action is to check all CLUSSDR and CLUSRCVR connectivity information (contents `CONNNAME` parameter). You check all definitions, but the connection information for the

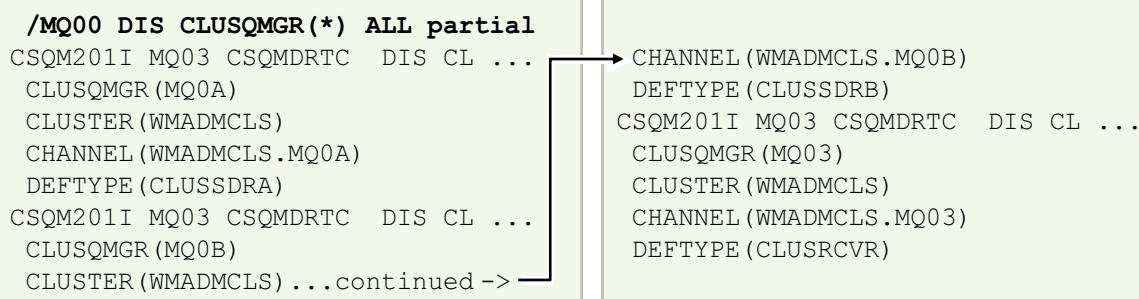
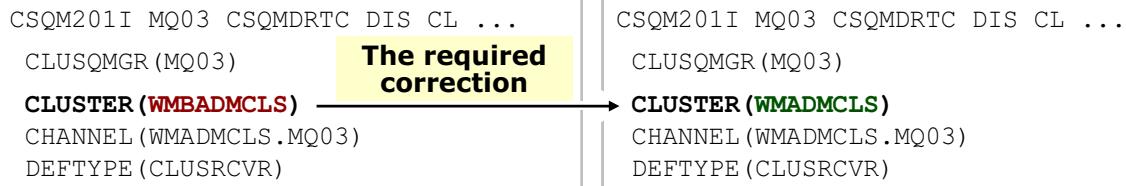
CLUSSDR to MQ0B looks correct, and so does the connection information in the CLUSRCVR channel.

After checking the channel connection attributes, the error was not obvious, so it was decided to refresh the cluster, although the `REFRESH` is not expected to correct the problem. You proceed to issue this command.

The slide that follows shows results from a z/OS system. However, in the distributed environment, you look for equivalent messages in the **queue manager** log. Other than operating system mechanics, the administration of a cluster is almost identical in distributed and z/OS systems.

MQ03 definition check: Log messages

```
From syslog: /MQ03 refresh cluster (WM*)
CSQ9022I MQ03 CSQMRCLU 'REFRESH CLUSTER' NORMAL COMPLETION
+CSQX875I MQ03 CSQXREPO REFRESH CLUSTER processing started for cluster *
+CSQX420I MQ03 CSQXREPO No repositories for cluster WMBADMCLS
+CSQX419I MQ03 CSQXREPO No cluster-receivers for cluster WMADMCLS
+CSQX442I MQ03 CSQXREPO Phase one of REFRESH CLUSTER has completed, 371
```



Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-13. MQ03 definition check: Log messages

After completing due diligence and carefully checking the channel definitions, you now try refresh the partial repository for the MQ03 queue manager by issuing the command `REFRESH CLUSTER(WM*)` without any additional parameters. **As mentioned earlier, the `REFRESH` command is not expected to magically correct the problem, but it might produce extra messages.**

Take a moment to look at the output of the command. You might notice that the command is issued in a z/OS queue manager. However, for a distributed queue manager as used in this course, you see equivalent information in the queue manager log.

It provides two clues:

- First, no repositories were found for cluster **WMBADMCLS**, but this name is not the intended name of the cluster used.
- Second, IBM MQ is reporting no cluster receivers for the **WMADMCLS** cluster, which is the correct cluster name.

These messages bring you to look at the MQ03 CLUSRCVR information, and you now find a typographical error in the CLUSTER name attribute for the MQ03 CLUSRCVR definition. You now alter the CLUSRCVR channel to correct the typographical error on the CLUSTER attribute, and reissue the DIS CLUSQMGR. What are the expected results?

- Dynamic CLUSSDRB channel **WMADMCLS.MQ0B** was created to MQ0B.

- Dynamic CLUSSDRA channels WMADMCLS.MQ0A and WMADMCLS.MQ00 are created to MQ0A and MQ00.

Note: Not all channels were able to be shown on the reduced display.

Usually, the errors that are found during cluster verification consist of errors in connectivity parameters. However, this time an incorrect cluster name in the MQ03 CLUSRCVR definition caused the error.

Cluster queues

At MQ0A:

```
DEFINE QLOCAL(CASHIN) CLUSTER(WMADMCLS) DEFBIND(NOTFIXED)
```

At MQ0B:

```
DEFINE QLOCAL(CASHIN) CLUSTER(WMADMCLS) DEFBIND(NOTFIXED)
```



MQCONNECT
to MQ00
MQOPEN
to CASHIN
MQPUT

```
/MQ0A DIS Q(CASHIN) CURDEPTH
CSQM201I MQ0A CSQMD
QUEUE(CASHIN)
CURDEPTH(15)

/MQ0B DIS Q(CASHIN) CURDEPTH
CSQM201I MQ0B CSQMD
QUEUE(CASHIN)
CURDEPTH(9)
```

/MQ00 DIS QCLUSTER(CA*) ALL (partial)

```
QUEUE(CASHIN)
CLUSTER(WMADMCLS)
DEFBIND(NOTFIXED)
CLWLRANK(0)
CLWLPRTY(0)
CLUSQMGR(MQ0A)
```

```
QMID(MQ0A.CDA33A44AE870B2B)
```

```
QUEUE(CASHIN)
CLUSTER(WMADMCLS)
DEFBIND(NOTFIXED)
CLWLRANK(0)
CLWLPRTY(0)
CLUSQMGR(MQ0B)
```

```
QMID(MQ0B.CDA2735757A9FEA9)
```

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-14. Cluster queues

As part of the cluster setup, you defined some cluster queues. If the applications that consume the messages are available in each queue manager, the same cluster queues can be defined in several cluster member queue managers.



Important

Cluster queues that have the same name must use similar attributes. For example, suppose that the CASHIN clustered queue in queue manager MQ0A has DEFBIND(NOTFIXED), and the CASHIN clustered queue in queue manager MQ0B has DEFBIND(OPEN). If so, you see an error message that informs you of “inconsistent queue definitions” when an attempt is made to display or use the queues.

The default QLOCAL DEFBIND attribute is OPEN, which means that when an application starts putting messages to a queue, they all go to the queue where the first MQOPEN bound these messages.

With no affinities, the DEFBIND can be, and should be, changed to NOTFIXED to allow workload balancing.

The command `DIS QCLUSTER(CAS*)` shows all instances of queues with names that start with `CAS` in the cluster. **However, the individual cluster queue managers with partial repositories might not learn about the newly defined queue until an MQPUT to the queue is attempted.** Partial repositories learn about shared queue definitions on a “need to know” basis.

Notice that the `DIS QCLUSTER` command also indicates the queue manager that each instance of the clustered queue belongs to in the `CLUSQMGR` attribute.

Cluster transmission queue options

1. Use default SYSTEM.CLUSTER.TRANSMIT.QUEUE
2. Automatically associate all cluster sender channels for a queue manager to be associated to a separate transmission queue
 - Set queue manager attribute DEFCLXQ to CHANNEL
 - Queue names are going to be SYSTEM.CLUSTER.TRANSMIT.ChannelName where ChannelName matches the CLUSSDR channel of the queue manager
3. Manually set selected cluster-sender channels to be served by a single transmission queue
 - Create a transmission queue
 - Set the CLCHNAME attribute to the name of the cluster-sender channel
4. Select a group of cluster-sender channels to be served by a single cluster transmission queue
 - Create the transmission queue
 - Set CLCHNAME to a generic name with a wildcard, such as ClusterName.*

[Understanding and implementing an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 3-15. Cluster transmission queue options

For many years, IBM MQ clusters used the queue that is named SYSTEM.CLUSTER.TRANSMIT.QUEUE as the transmission queue to be used by the CLUSSDRA and CLUSSDRB channels of a queue manager.

The ability to use different transmission queues was added to help customers either isolate sensitive messages, or mitigate heavy message loads to use different transmission queues. However, while it is possible to use other transmission queues, without strong justification, the original SYSTEM.CLUSTER.TRANSMIT.QUEUE should be used in favor of simplicity.

The use of separate transmission queues can be configured in two ways:

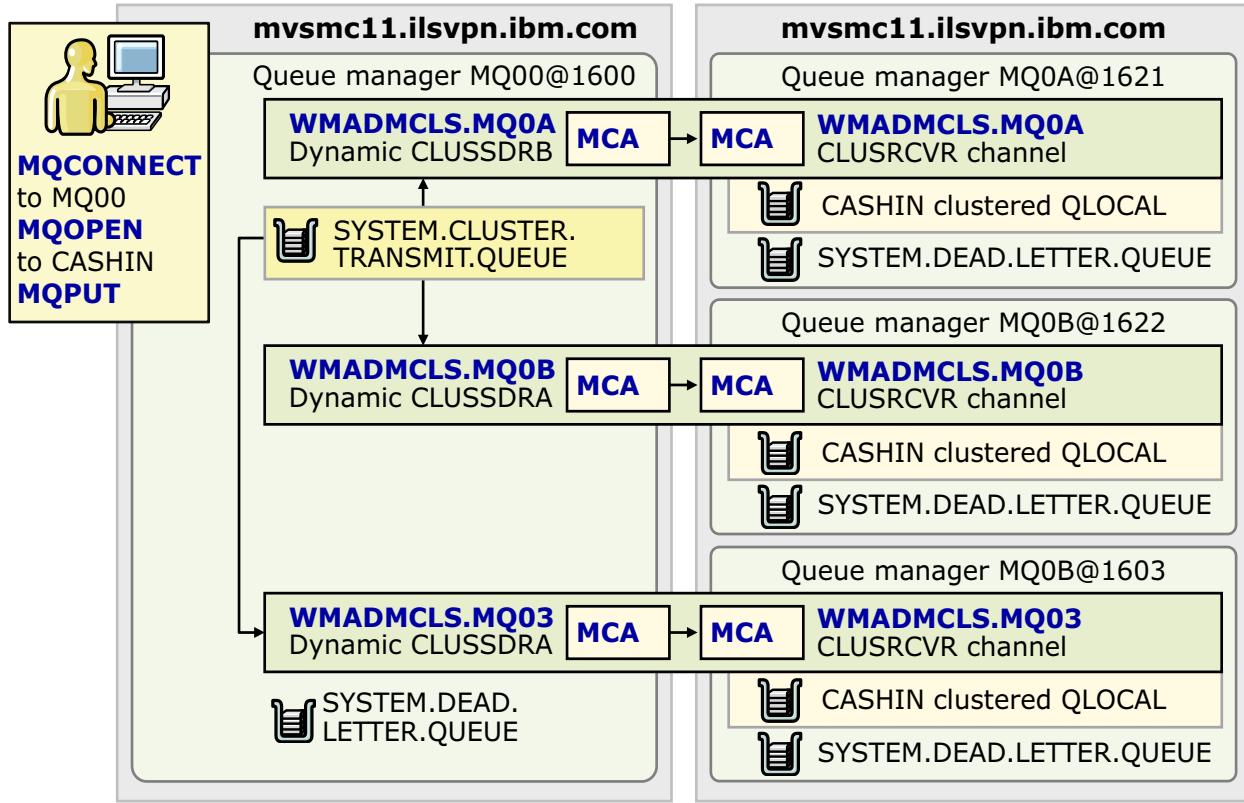
- Change the `DEFCLXQ` queue manager parameter to `DEFCLXQ(CHANNEL)`. In this case, IBM MQ automatically generates the transmission queue. The name of the transmission queue starts with SYSTEM.CLUSTER.TRANSMIT.QUEUE, followed by channel name. For example, if cluster queue manager MQ03 in cluster WMADMCLS creates a dynamic channel to cluster queue manager MQ0B, the transmission queue that is used in queue manager MQ03 for channel WMADMCLS.MQ0B is called `SYSTEM.CLUSTER.TRANSMIT.WMADMCLS.MQ0B`.
- The second option is to manually define a transmission queue, and associate the transmission queue to a cluster channel by changing the queue's `CLCHNAME` parameter. Naming the channel has different specific and generic options.

You can specify the transmission queue changes before or after the channel is active. For example, for a new queue manager, you can change DEFCLXQ before the queue manager joins the cluster. However, if the change is done to an active queue manager, it is optimal to wait until a time that the channel might not be as busy. Use the same consideration if you use the CLCHNAME queue attribute method.

When the queue manager senses a change in transmission queue, it starts “switch” processing to the newly specified transmission queue. In those cases, you might see the channel status display as “switching”.

The z/OS environment has caveats that are documented. If you work with the z/OS platform, ensure that you review the extra z/OS notes for changing the transmission queues in the IBM Knowledge Center.

Message trajectory in a basic cluster



Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-16. Message trajectory in a basic cluster

How does the trajectory of messages that travel across a cluster compare to messages that travel across distributed message channels? What path do cluster messages take?

Other than workload balancing, following the path of a message in a basic cluster queue is similar to working with distributed queuing except that the message could go in some additional cluster queues. In this walk-through, you must make the following assumptions:

- That the queue manager is not set to allow the messages to be placed outside the local version of the CASHIN queue if one is available. This setting is explained in a later slide.
- The MQ00 queue manager does not host the CASHIN queue.
- The queue managers all use the default transmit queue.

An application issues MQOPEN of clustered queue CASHIN for MQ00. The MQ00 queue manager does not host the CASHIN queue. Cluster workload balancing determines the best candidate queue manager that is hosting the CASHIN cluster queue to be the target for the message.

- Assume that the message goes to CASHIN at MQ0A. The message is first placed on SYSTEM.CLUSTER.TRANSMIT.QUEUE at MQ00. MQ00 has its CLUSSDR channel pointed to MQ0A, so a dynamic channel of type CLUSSDRB called WMADMCLS.MQ0A picks up the message. If no problems are encountered, it places this message in the CASHIN queue at MQ0A.

- If a problem occurs with the channel, the message would wait in the SYSTEM.CLUSTER.TRANSMIT.QUEUE at MQ00.
- If a different problem occurs, the message might be forwarded to the dead-letter queue, and identified for either MQ00 or MQ0A, depending on the problem.

This trajectory is similar to tracing the route of a message in a distributed channel. If the workload algorithm selects MQ0B or MQ03 as the target for the message, then a CLUSSDRA instead of a CLUSSDRB dynamic channel carries the message to its target.

If you are using a dedicated transmit queue other than SYSTEM.CLUSTER.TRANSMIT.QUEUE, then instead of looking in SYSTEM.CLUSTER.TRANSMIT.QUEUE you would check in the dedicated named transmit queue. Using different transmit queues is discussed in a later slide.

If it is critical to find which of the clustered queues the message was sent to, the status for the potential candidate cluster queues can be displayed, for instance. In the display, it is assumed that queue monitoring is active for the queue manager and also for the queue.

```
MQ0A DIS QSTATUS(CASHIN) ALL
QSTATUS(CASHIN)
TYPE(QUEUE)
OPPROCS(0)
IPPROCS(0)
CURDEPTH(17)
UNCOM(NO)
MONQ(LOW) <== queue monitoring enabled, but need to check queue manager
QTIME(0,0)
MSGAGE(3138292)
LPUTDATE(2014-10-15) <== date message was last put
LPUTTIME(19.52.35) <== date message was last put
LGETDATE()
```

Since you did get the last put date and time populated in the DIS QSTATUS display, it is obvious that the queue manager was enabled for monitoring information. However, the command to check whether the queue manager is enabled is shown here in case it is needed.

```
MQ0A DIS QMGR MONQ
QMNAME(MQ0A)
MONQn(LOW) <== queue manager is enabled
```

Routing messages to and from clusters

- Configuring request/reply to a cluster
- Configuring request/reply from a cluster
- **Configuring workload balancing from outside a cluster**
- Configuring message paths between clusters
- Queue manager aliases and clusters
- Reply-to queue aliases and clusters
- Queue aliases and clusters

Terminology conflicts are common – always verify with a topology diagram

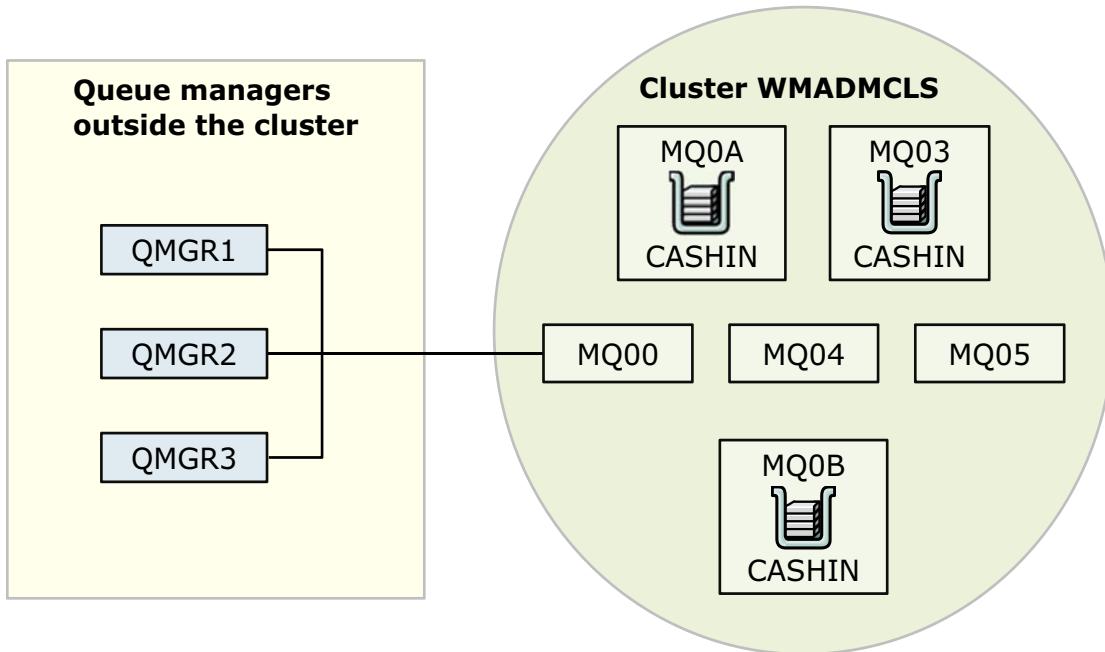
Figure 3-17. Routing messages to and from clusters

In the IBM MQ baseline unit, you reviewed how remote and alias queues definitions are used for various routing and even security purposes. After you complete the cluster configuration, you continue to use the remote and alias definitions to route messages to and from clusters. However, obtaining the correct results might be challenging.

For your IBM MQ configuration, the IBM Knowledge Center documents several examples of how to route messages to and from clusters for different purposes. A partial selection of the configurations that are documented are shown on this slide. Be careful with the terminology that is used. For example, the term “gateway” can be used for different scenarios. Make sure that you and your team use a common topology diagram to ensure a common understanding of the terminology and objectives to be accomplished with the configuration. While you might not share individual attributes, you must have a clear picture of the topology to be implemented.

Numerous routing combinations exist. In this unit, you look at how to implement a cluster gateway that, without other customization, helps leverage the cluster round-robin distribution from outside the cluster.

Workload balancing from outside a cluster: Cluster gateway



Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-18. Workload balancing from outside a cluster: Cluster gateway

Numerous ways are available to design the infrastructure with IBM MQ clusters.

For example, overlapping clusters are a configuration in which two clusters are joined. Overlapping clusters are reviewed in a later unit.

In this slide, you look at a common configuration where a cluster uses a gateway queue manager to interface with external applications. In this topology, the term “gateway” is used to denote a queue manager member of the cluster that is used exclusively for connections with applications that are outside the cluster. With this configuration, the cluster normal round-robin workload balance is influenced from outside the cluster. The gateway queue manager can also hold cluster remote queues to which cluster applications can put messages to be sent from the cluster applications, to outside applications.

WMADMCLS might be the cluster for the ORDERS application. Within the ORDERS application, there might be:

- Redundant servers that host the CASHIN queue for the billing department
- Other servers that deal with fulfillment or shipping, but all interfacing internally

The ORDERS application does not allow users external to the application to join their cluster, so they created MQ00 as a gateway queue manager into the cluster. This configuration is common.

As shown, MQ00 would be a single point of failure. Therefore, it is assumed that MQ00 is set up with a high-availability solution, or it might be part of a queue-sharing group.

Definitions to MQ00 gateway from an external queue manager

At queue manager QMGR1

```
DEFINE QLOCAL(MQ00) +
  USAGE(XMITQ) +
  INITQ(SYSTEM.CHANNEL.INITQ) +
  TRIGGER +
  TRIGDATA(QMGR1.MQ00)

DEFINE QREMOTE(PAYMENTS) +
  RQMNAME(GATEWAY0) +
  RNAME(CASHIN) +
  XMITQ(MQ00)

DEFINE CHANNEL(QMGR1.MQ00) +
  CHLTYPE(SDR) +
  TRPTYPE(TCP)
CONNAME(mvsmc...ibm.com(1600')) +
  XMITQ(MQ00)
```

At queue manager MQ00

```
DEFINE QREMOTE(GATEWAY0) +
  RQMNAME(' ') RNAME(' ') +
  XMITQ(' ')

DEFINE CHANNEL(QMGR1.MQ00) +
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)
```

Local queue CASHIN is defined in other queue manager members of the WMADMCLS cluster

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-19. Definitions to MQ00 gateway from an external queue manager

A partially populated QREMOTE object is used to define a queue manager alias. In queue manager MQ00, the GATEWAY0 object is the queue manager alias definition.

The gateway queue manager, in this case MQ00, does not host any clustered queues. However, it is given a queue manager alias definition, GATEWAY0. This definition helps to resolve queue resolution for inbound messages that are not part of the cluster. It helps them to select an instance of the CASHIN clustered queue from one of the clustered queue managers that host the CASHIN queue.

Remote queue manager QMGR1 is not part of the WMADMCLS cluster.

Without the GATEWAY0 object at the MQ00 queue manager, messages that are arriving to MQ00 go to the MQ00 designated dead-letter queue.

On the QMGR1 side, the QREMOTE definition must have the name of the MQ00 queue manager alias, GATEWAY0, in its RQMNAME attribute. If omitted, messages also end up in the dead-letter queue when they arrive at the MQ00 queue manager.



Cluster administrative options

- ISPF panels and MQSC
- IBM MQ Explorer presents organized display
 - Cluster full and partial repositories
 - QCLUSTER tab
 - CLUSQMGR sender and receiver tabs
 - Publish/subscribe topics

The screenshot shows the IBM WebSphere MQ Explorer interface. The top window title is "IBM WebSphere MQ Explorer (IBMMQV8)". The menu bar includes "File", "Edit", "Window", and "Help". The "MQ Explorer - Navigator" tab is selected. The tree view under "IBM WebSphere MQ" shows "Queue Managers" (with "WM102" and "WMADMCLS" expanded), "Queue Manager Clusters" (with "WM102" and "WMADMCLS" expanded), "Full Repositories" (with "MQ0A" and "MQ0B" listed), and "Partial Repositories" (with "MQ00" and "MQ03" listed). The bottom window title is "MQ Explorer - Content". The content pane displays "Repository data for queue manager MQ00". A tab bar at the top of the content pane includes "Cluster Queues", "Cluster Topics", "Cluster-sender Channels" (which is selected), and "Cluster-receiver Channels". Below the tab bar, the text "Cluster-sender channels:" is followed by "MQ03 - Partial Repository". A diagram shows two nodes: "MQ00" on the left and "MQ03" on the right, connected by a single arrow pointing from MQ00 to MQ03. Below the diagram is a table:

Channel name	Cluster queue manager	Queue manager type	Definition type	Xmit protocol	Channel s
WMADMCLS.MQ03	MQ03	Normal	Auto cluster-sender	TCP	Running
WMADMCLS.MQ0A	MQ0A	Repository	Auto explicit cluster-sender	TCP	Running
WMADMCLS.MQ0B	MQ0B	Repository	Auto cluster-sender	TCP	Running

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-20. Cluster administrative options

IBM MQ clusters can be administered with MQSC commands, with IBM MQ Explorer, or by using the IBM MQ ISPF panels in the z/OS environment. For clusters, IBM MQ Explorer lays out the cluster objects in an organized way.

For this display, you are looking at queue manager MQ00. The navigator menu shows MQ00 as a partial repository. Selecting the MQ00 queue manager brings several tabs:

- The Cluster Queues tab is equivalent to DIS QCLUSTER.
- Cluster topics are discussed in the publish/subscribe unit.
- Cluster-sender channel contains CLUSSDR information equivalent to: DIS CLUSQMGR (*) ALL
- Cluster-receiver channel shows CLUSRCVR information equivalent to: DIS CLUSQMGR(*) ALL

Right-click of the queue manager brings up other commands such as REFRESH and RESET (named REMOVE), and SUSPEND.

Unit summary

- Review a basic IBM MQ cluster
- Describe the components that support an IBM MQ cluster
- Distinguish between the definitions that are required to set up a cluster and distributed message channel definitions
- Describe how to implement a basic IBM MQ cluster
- Explain how to verify a new cluster configuration
- Describe how to add queues to a cluster
- Describe the trajectory of a message that uses a basic cluster
- Explain how to set up a cluster gateway by configuring a queue manager external to the cluster to route messages to cluster queues
- Summarize the administrative options that can be used to manage a cluster

[Understanding and implementing an IBM MQ cluster](#)

© Copyright IBM Corporation 2017

Figure 3-21. Unit summary

Review questions (1 of 2)

1. A cluster is created and a DIS CLUSQMGR(*) issued. If the definitions have no errors, which of the following channel DEFTYPES should appear in the display? Select all that apply for an error-free implementation:
 - A. CLUSSDRA
 - B. CLUSRCVR
 - C. CLUSSDR
 - D. CLUSSDRB

2. True or False: The CLUSRCVR channel does not require a CONNAME attribute.



Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-22. Review questions (1 of 2)

Review questions (2 of 2)

- 
3. A new cluster that is composed of queue managers MQA1, MQA2, MQA3, and MQA4 is defined. DIS CLUSQMGR output from both MQA1 and MQA2, the two full repositories, shows MQA4, but no signs of MQA3. What should the administrator do next?
 - A. Check the cluster channel definitions in the full repository queue managers
 - B. Issue REFRESH CLUSTER commands from one of the full repositories until MQA3 shows up
 - C. Issue REFRESH CLUSTER from MQA3 until the problem is resolved
 - D. Check the MQA3 cluster channel definitions
 4. True or False: Information about newly defined cluster queues is not automatically sent to partial repository queue managers until these queue managers have a “need to know” about the queue.

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-23. Review questions (2 of 2)

Review answers (1 of 2)



1. A cluster is created and a DIS CLUSQMGR(*) issued. If the definitions have no errors, which of the following channel DEFTYPES should appear in the display? Select all that apply for an error-free implementation:

- A. CLUSSDRA
- B. CLUSRCVR
- C. CLUSSDR
- D. CLUSSDRB

The answer is A, B, and D. When CLUSSDR displays for the DIS CLUSQMGR, look for an error. However, CLUSSDR queue types are valid even for a cluster channel when you use the DIS CHLSTATUS command.

2. True or False: The CLUSRCVR channel does not require a CONNAME attribute.

The answer is False. A CONNAME attribute is required.

Review answers (2 of 2)

3. A new cluster that is composed of queue managers MQA1, MQA2, MQA3, and MQA4 is defined. DIS CLUSQMGR output from both MQA1 and MQA2, the two full repositories, shows MQA4, but no signs of MQA3. What should the administrator do next?
 - A. Check the cluster channel definitions in the full repository queue managers
 - B. Issue REFRESH CLUSTER commands from one of the full repositories until MQA3 shows up
 - C. Issue REFRESH CLUSTER from MQA3 until the problem is resolved
 - D. Check the MQA3 cluster channel definitions

The answer is D. You must determine the problem first.
4. True or False: Information about newly defined cluster queues is not automatically sent to partial repository queue managers until these queue managers have a “need to know” about the queue.
 The answer is True. Information about new cluster queues that are defined in a partial repository is not usually relayed until after the first attempt to put a message to the queue.

Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-25. Review answers (2 of 2)



Exercise: Implementing and verifying a cluster, cluster queues, and a cluster gateway

© Copyright IBM Corporation 2017

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

Figure 3-26. Exercise: Implementing and verifying a cluster, cluster queues, and a cluster gateway

Exercise objectives

- Describe the MQSC commands that are used to create an IBM MQ cluster
- Implement an IBM MQ cluster by using MQSC commands
- Use the DIS CLUSQMGR command to verify a new cluster by identifying the channel types, status, and repository type
- Create cluster queues
- Identify problems that are found in the cluster
- Observe the cluster “need-to-know basis” behavior by using the DIS QCLUSTER command from a partial configuration queue manager before any messages are exchanged
- Configure and test a cluster gateway
- Send messages from a queue manager external to the cluster, to a cluster gateway queue manager
- Baseline default message workload distribution



Understanding and implementing an IBM MQ cluster

© Copyright IBM Corporation 2017

Figure 3-27. Exercise objectives

Unit 4. Cluster administration tasks and commands

Estimated time

01:00

Overview

This unit highlights similarities and differences in administering IBM MQ with and without clusters. You learn about the documented processes to accomplish various cluster-specific tasks, and the MQSC commands that are used in the cluster administration processes.

How you will check your progress

- Checkpoint
- Machine exercises

Unit objectives

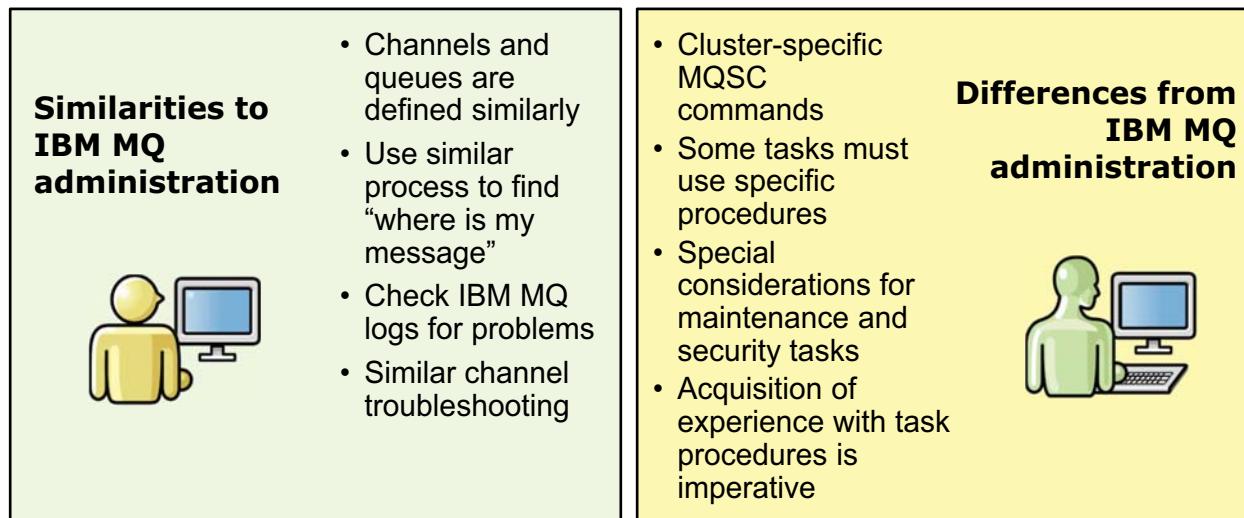
- Summarize the similarities and differences in administering IBM MQ with and without clusters
- Describe the cluster-specific MQSC administration commands
- List the predefined procedures that are available to complete cluster tasks
- Describe how to create a cluster queue manager that uses separate transmission queues
- Describe the process that is used to remove a cluster queue from a queue manager
- Describe the process that is used to remove a cluster member queue manager from the cluster
- Describe the process that is used to move a full repository to a different cluster queue manager
- Explain how to add a queue-sharing group to an existing cluster
- Explain how to use the IBM MQ Explorer equivalent cluster capabilities to complete selected cluster administrative tasks
- Describe how to use selected IBM MQ utilities to work with cluster tasks

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-1. Unit objectives

IBM MQ and IBM MQ cluster administration contrasts



- Understand the appropriate use for IBM MQ cluster-related commands
- Adhere to task-related documented procedures
- Allow the cluster time to learn about changes
- Keep a proactive approach to bridge the learning curve for administration and troubleshooting

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

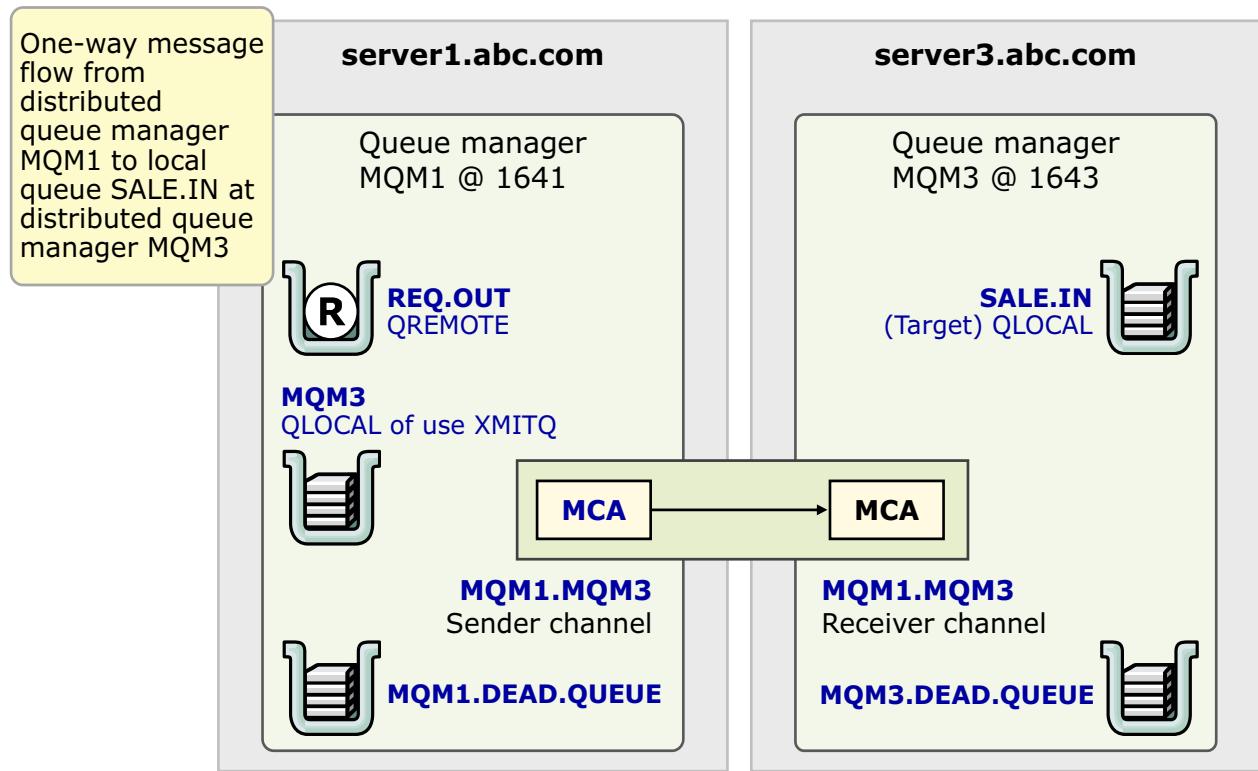
Figure 4-2. IBM MQ and IBM MQ cluster administration contrasts

When you are new to clusters, it is good to look at the similarities between working with non-cluster queue managers, and working with cluster queue managers. You then realize that other than the cluster-specific commands, working with clusters is similar to non-cluster environments. For example:

- You are working with a channel pair. Whether it is a sender-receiver, or a CLUSSDRA-CLUSRCVR connection, you work with a channel pair.
- You check the queue manager logs for any channel problems. You might need to look at the logs for more than one queue manager in the cluster, but the checking of the logs is the same.
- Checking the health of the channels is similar. Although cluster channels have one extra status, which is SWITCHING, the other states are the same. A channel can be running, stopped, or might be in doubt. You follow the same process. A channel that is in SWITCHING state, for IBM MQ V7.5 and later, is changing to a different transmission queue.
- The path that messages follow is the similar. Although queue name resolution has a different selection process, messages still stop by a transmission queue on the way to its target. However, the target might be in different queue managers that host the clustered queue.

When you look at the similarities between distributed and clustered message channels, you realize that other than learning a few cluster-specific processes, your work is not that different.

SENDER-RECEIVER distributed message channels



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

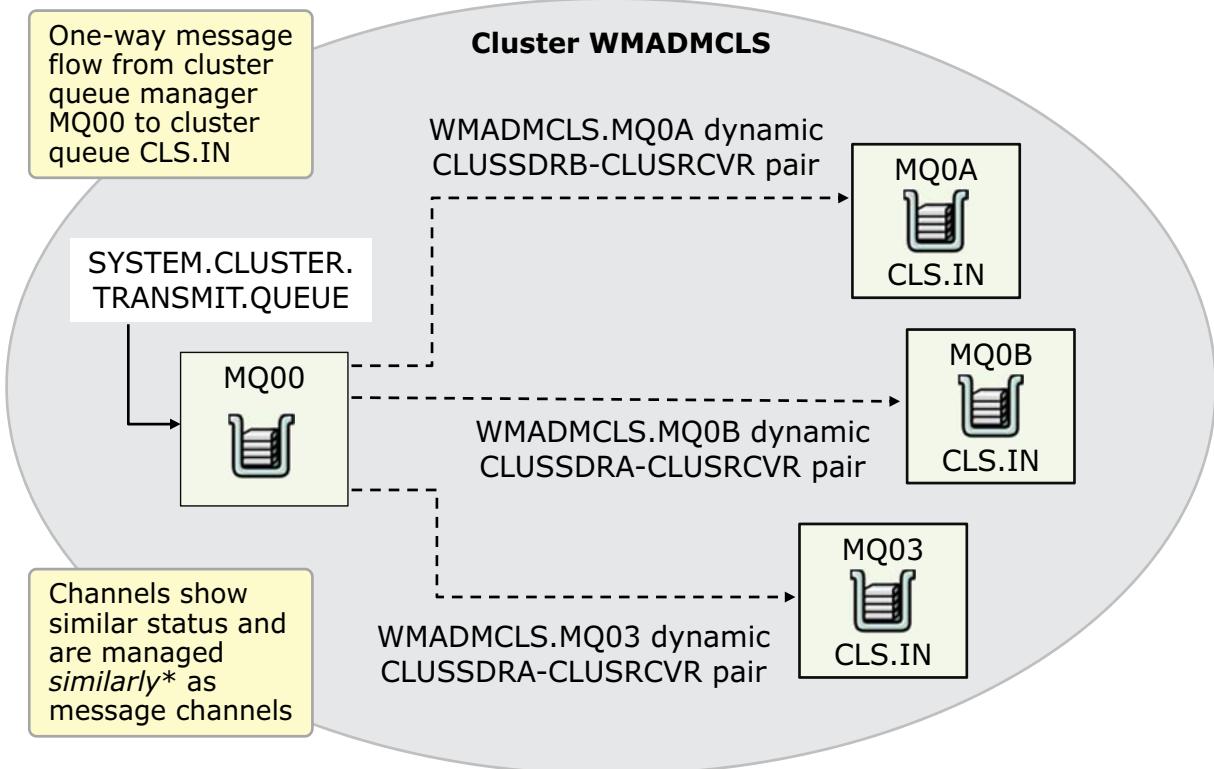
Figure 4-3. SENDER-RECEIVER distributed message channels

In this slide, you look at the basic trajectory that a distributed message takes in reaching its target. The message first goes to the transmission queue of the originating queue manager. A message channel agent retrieves the message from the transmission queue and relays it to the target (or next in line) queue manager. In this figure, the message goes from queue manager MQM1 to queue manager MQM3.

If you need to find your message and the channel is not running, the message is expected to be in the MQM3 transmission queue. However, if it has an error, then depending on the error, the message might be in one of the dead-letter queues for queue managers MQM1 or MQM3.

When you configure the environment, you trigger the channel by setting up the transmission queue to be triggered so that the channel starts automatically upon messages that are arriving at the transmission queue. You then type one message to start the channel the first time to get the channel out of stopped state. The channel then continues to “trigger-start” automatically.

CLUSSDRx-CLUSRCVR clustered message channels



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-4. CLUSSDRx-CLUSRCVR clustered message channels

You now look at a similar trajectory, but for CLUSSDRB-CLUSRCVR and CLUSSDRA-CLUSRCVR channel flow to queue CLS.IN, which is hosted by queue managers MQ0A, MQ0B, and MQ03.

If a message to queue CLS.IN originates from cluster queue manager MQ00, the message first passes by the SYSTEM.CLUSTER.TRANSMIT.QUEUE for queue manager MQ00. If the cluster channel was set up to use a different transmission queue, then the message passes by the transmission queue of queue manager MQ00. If the code includes no message affinities, a group of messages to queue CLS.IN should be sent to each of the hosted instances of queue CLS.IN in queue managers MQ0A, MQ0B, and MQ03.

If messages are not flowing, you can use the `DIS CLUSQMGR` or the `DIS CHS` to determine the channel status. The one extra status you might find in a cluster is **SWITCHING**, which is when you decide to use a different cluster transmission queue. As you get more familiar with clusters, you might find that `DIS CLUSQMGR` provides more information on the channel type, as the difference is subtle. When you use `DIS CLUSQMGR`, the **DEFTYPE** shows you when you are looking at a CLUSSDRA, CLUSSDRB, and CLUSRCVR channel. For a cluster, a CLUSSDR channel in a `DIS CLUSQMGR` display is a problem. When you use `DIS CHS`, the **CLUSSDR CHLTYPE** is acceptable.

You also go to the queue manager log to look for problems. You might want to look at the queue manager log for the expected target queue manager or queue managers first.

If entries in the log indicate a problem, the message might be in the dead-letter queue of any of the four queue managers. However, you are still following a similar process to find your message, and to work with your channels.

As far as managing your channels, your work is similar to your work with distributed message channels.

You now look at cluster-specific processes and commands.

IBM MQ cluster administration commands

Commands	
DIS CLUSQMGR	Shows cluster information and detail on connections
DIS QCLUSTER	Shows cluster queues and pertinent details
SUSPEND QMGR	Notifies cluster member queue managers to stop sending messages to the queue manager that is specified in the command
RESUME QMGR	Informs cluster member queue managers that the queue manager specified in the command is back and available to receive messages
RESET CLUSTER	Forces the removal of a queue manager from a cluster
REFRESH CLUSTER	 <ul style="list-style-type: none"> • Updates the cluster repository • Impacts performance • Refrain from unnecessary use • Read all usage notes

Might be supplemented with other non-cluster-specific MQSC commands

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-5. IBM MQ cluster administration commands

Several commands are used to manage the cluster. Under most circumstances, **DIS CLUSQMGR** and **DIS QCLUSTER** are the most commonly used commands.

Later in this unit you learn about **SUSPEND QMGR** and **RESUME QMGR**. These commands are used mostly when you need to temporarily remove and then restore a queue manager from activity in the cluster under situations such as when you need to apply maintenance.

REFRESH CLUSTER can lead to performance problems, and should not be used unless the administrator is experienced with clusters and reads all the warnings in the documentation. Some forms of this command should never be used from a full repository. Allowing the cluster a few moments to update itself usually is a better option than trying to use **REFRESH CLUSTER**.

For some unknown reason, people who do not have experience with clusters seem to favor use of the **REFRESH CLUSTER** command. **REFRESH CLUSTER** is used to refresh the contents of a partial repository cluster queue manager. It is not used on a full repository queue manager.

The **REFRESH CLUSTER** command is not a magic “fix-all” command that corrects all ills of a cluster channel or configuration. The first detail of the **REFRESH CLUSTER** slide contains an excerpt from the queue manager log where the adverse effects of using **REFRESH CLUSTER** are noted.

The **REFRESH CLUSTER** command is very powerful. Always use it with care. Justifiable uses, along with other details of the **REFRESH CLUSTER** command, are included in the **REFRESH CLUSTER** slide

later in this unit. If a queue manager loses connectivity to either full repository queue manager, then `RESET CLUSTER` can be used to forcibly remove the queue manager from the cluster.

On the `RESET CLUSTER`, in the unlikely event that two queue managers with the same name were defined in the same cluster, it is important to obtain the queue manager ID, or QMID. This ID qualifies the `RESET CLUSTER` command, ensuring that the correct queue manager is removed. More information on the `RESET CLUSTER` slide is presented later in this unit.

DISPLAY CLUSQMGR

- Returns information about all cluster queue managers

CLUSTER	Name of cluster
CHANNEL	Name of channel
CLUSDATE	Date when the object definition was made available
CLUSTIME	Time at which the object definition was made available
QMTYPE	Identifies whether the queue manager is a repository queue manager (REPOS or NORMAL)
QMID	Unique name (generated internally) for each queue manager

- Includes information about cluster channel definition in **DEFTYPE**

CLUSSDR*	Manual definition of a cluster sender channel
CLUSSDRA	IBM MQ automatically defined this channel
CLUSSDRB	Channel was automatically defined <i>and</i> manually defined
CLUSRCVR	Manual definition of a cluster receiver channel



Appearance of a DEFTYPE CLUSSDR channel in the DIS CLUSQMGR output denotes a configuration problem

Figure 4-6. DISPLAY CLUSQMGR

In the cluster configuration unit, you became familiar with the output of the `DISPLAY CLUSQMGR` command. The command provides information useful to verify that your cluster is correctly configured, and to investigate any problems.

You are already familiar with some of the pieces of information that this command generates. However, a DEFTYPE of CLUSSDR, without being followed by an A or a B, is to be considered an error and investigated. The valid DEFTYPE outputs are CLUSSDRA, CLUSSDRB, or CLUSRCVR.

DISPLAY CLUSQMGR

/MQ0A DIS CLUSQMGR(*) ALL (partial display)

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ0A)

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0A)

QMID (MQ0A.CDA33A44AE870B2B)

DEFTYPE (CLUSRCVR)

QMTYPE (REPOS)

...

This definition is for the MQ0A queue manager itself

CLUSRCVR channel type definition for MQ0A

QMTYPE REPOS: This queue manager (MQ0A) holds a full repository

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ0B)

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0B)

QMID (MQ0B.CDA2735757A9FEA9)

DEFTYPE (CLUSSDRB)

...

QMTYPE (REPOS)

STATUS (RUNNING)

...

The partner queue manager for this MQ0A connection is MQ0B

A CLUSSDRB is the combination of both MQ0B's CLUSRCVR and MQ0A's CLUSSDR; MQ0A defined CLUSSDR to MQ0B

The MQ0B queue manager holds the second full repository

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-7. DISPLAY CLUSQMGR

This slide shows some of the DIS CLUSQMGR output that you are already familiar with. One piece of information that possibly did not occur to you earlier is the QMID, which you can also obtain from the DIS QMGR command. In this slide, you see examples of a CLUSRCVR channel and a CLUSSDRB channel.

DISPLAY QCLUSTER

- Displays cluster queues in current queue manager
 - Partial repository cluster queue managers learn about cluster queues on a “need-to-know” basis
 - A queue manager in the cluster might not see new cluster queues until after the first MQPUT is done to the cluster queue

```
dis qcluster(EX3.IN) all
  1 : dis qcluster(EX3.IN) all
AMQ8409: Display Queue details.

  QUEUE (EX3.IN)          TYPE (QCLUSTER)
  ALTDATE (2017-05-04)    ALTTIME (11.01.28)
  CLUSDATE (2017-05-04)   CLUSTER (WMADMCLS)
CLUSQMGR (MQ0B)        CLUSQT (QLOCAL)
  CLUSTIME (11.06.07)     CLWLPRTY (0)
  CLWLRANK (0)            DEFBIND (OPEN)
  DEFPRTY (0)              DEFPSIST (NO)
  DEFPRESP (SYNC)          DESCRIPTOR ( )
  PUT (ENABLED)           QMID (MQ0B_2017-04-27_00.52.52)
```

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-8. DISPLAY QCLUSTER

The DISPLAY QCLUSTER command provides a way to look at clustered queues.

For each clustered queue displayed, the command provides information as to which cluster queue manager owns the instance of the specific cluster queue, and also contains the corresponding host queue manager QMID.

A common mistake is to run this command from a partial repository queue manager before any messages are put to queues that other queue managers in the cluster might host. Then, you might not see the queues that are displayed. The next slide contains extra information.

DIS QCLUSTER caveats

- Queue EX3.IN gets defined in cluster WMADMCLS queue manager members MQ03 and MQ0B
- You might not find newly defined queues until after the first MQPUT
- Immediately after the queues are defined, you use **DIS QCLUSTER** to display the queues from WMADMCLS queue manager MQG1:

```
dis qcluster(EX3.IN) clusqt clusqmgr
  1 : dis qcluster(EX3.IN) clusqt clusqmgr
AMQ8147: IBM MQ object EX3.IN not found.
```



- After an application puts one or more messages via manager MQG1, you repeat the **DIS QCLUSTER** command from queue manager MQG1

```
dis qcluster(EX3.IN) clusqt clusqmgr
  1 : dis qcluster(EX3.IN) clusqt clusqmgr
AMQ8409: Display Queue details.
  QUEUE (EX3.IN)
  CLUSQMGR (MQ0B)
AMQ8409: Display Queue details.
  QUEUE (EX3.IN)
  CLUSQMGR (MQ03)
```

TYPE (QCLUSTER) CLUSQT (QLOCAL)	
TYPE (QCLUSTER) CLUSQT (QLOCAL)	

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-9. DIS QCLUSTER caveats

Partial repository queue managers “learn” about new queue definitions on a “need-to-know” basis. You might run the **DIS QCLUSTER** from a partial repository queue manager that does not host the queue. You see the response “object not found” before any messages are put to a newly defined queue.

Often, administrators new to clusters might interpret the “not found” as an error. Lack of knowledge of a recently defined cluster queue in a partial repository is “working as designed”, “need-to-know” behavior. By using a sample program to put a test message from the same queue manager where you received the “not found” reply, and repeating the **DIS QCLUSTER** command, you see the queues that are displayed.

If you type the **DIS QCLUSTER** command from a full repository, you see the queues before the first MQPUT.

SUSPEND QMGR command



- Informs other members to stop sending messages
- Temporarily inhibits use of the queue manager in the cluster
- Good option when maintenance is applied to a cluster queue manager
- The command is issued for
 - CLUSTER or CLUSNL
 - FACILITY - z/OS DB2 or IMBSBRIDGE
 - LOG – z/OS
- Special z/OS considerations
- The MODE parameter controls how the suspension is processed
 - QUIESCE
 - FORCE

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-10. SUSPEND QMGR command

The SUSPEND QMGR command can take different forms. It can be used on an individual, or on overlapping clusters.

In the z/OS platform, it is also used as SUSPEND QMGR FACILITY to inhibit DB2 and IMS resources, and as SUSPEND QMGR LOG to suspend queue manager logging resources. See the IBM Knowledge Center for extra notes when you use SUSPEND in the z/OS platform.

Two forms or modes are available for inhibiting a queue manager.

- The default mode is QUIESCE. This form advises other queue managers not to send messages to the suspended queue manager. However, it does not inhibit messages from being sent to the suspended queue manager. If a queue is hosted exclusively in the suspended queue managers, messages continue to flow to the queue manager suspended in quiesce mode. However, when a queue is hosted in several cluster queue managers, the non-suspended queue managers are selected to receive the messages.
- MODE(FORCE) keeps a queue manager from receiving messages even when the queue manager is the only host of a clustered queue.

After you issue a SUSPEND QMGR command, you have follow-up considerations.

SUSPEND QMGR command with MODE(FORCE)



- Stops CLUSRCVR channel
- If suspended queue manager is the only host for a specific queue, messages remain in the SYSTEM.CLUSTER.TRANSMIT.QUEUE of the sending cluster queue manager member
- After the queue manager is resumed, it might require a manual start of CLUSSDRA and CLUSSDRB channels from each of the other cluster queue managers

```
suspend qmgr cluster(WMADMCLS) mode(FORCE)
  1 : suspend qmgr cluster(WMADMCLS) mode(FORCE)
AMQ8557: SUSPEND QUEUE MANAGER accepted.
dis clusqmgr(*) deftype status
  2 : dis clusqmgr(*) deftype status
AMQ8441: Display Cluster Queue Manager details.
          CLUSQMGR(MQ03)           CHANNEL(WMADMCLS.MQ03)
          CLUSTER(WMADMCLS)        DEFTYPE(CLUSRCVR)
          STATUS(STOPPED)
  . . . . .
```

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-11. SUSPEND QMGR command with MODE(FORCE)

If you type the DIS CLUSQMGR command to display the channel status after you suspend a queue manager with MODE(FORCE), you observe that the CLUSRCVR channel is stopped.

When the queue manager is resumed, check that the WMADMCLS.MQ03 CLUSSDRA and CLUSSDRB are able to start. While most of the time the channels restart uneventfully, occasionally it might be necessary to manually intervene to restart the CLUSSDRx channels to the resumed queue manager.

RESUME QMGR command



- Reinstates use of a previously suspended queue manager in the cluster
- Might require a manual restart of CLUSSDRA and CLUSSDRB channels to resume the queue manager in each of the cluster queue managers
- The command is issued for
 - CLUSTER or CLUSNL
 - FACILITY - z/OS DB2 or IMBSBRIDGE
 - LOG – z/OS
- Special z/OS considerations

NOTE: Resumed queue manager was suspended with MODE (FORCE)

```
resume qmgr cluster(WMADMCLS)
  1 : resume qmgr cluster(WMADMCLS)
AMQ8556: RESUME QUEUE MANAGER accepted.
dis clusqmgr(*) deftype status
  2 : dis clusqmgr(*) deftype status
AMQ8441: Display Cluster Queue Manager details.
          CHANNEL (WMADMCLS.MQ03)
          CLUSTER (WMADMCLS)
STATUS (INACTIVE)
```

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-12. RESUME QMGR command

The `RESUME QMGR` command returns a suspended queue manager to normal use in the cluster.

If you use the `RESUME QMGR` command on z/OS, you must check other considerations. For example, you cannot include the command in the startup JCL. Refer to the IBM Knowledge Center for notes on uses of `RESUME QMGR` on z/OS.

Cluster-specific procedures from IBM Knowledge Center (partial)

- Converting an existing IBM MQ network into a cluster
- Removing a cluster queue from a queue manager
- Removing a queue manager from a cluster
- Moving a full repository to another queue manager
- Incorporating a queue-sharing group into an existing cluster
- Adding a queue manager to a cluster that uses a separate transmission queue
- Creating two overlapping clusters with a gateway queue manager
- Restoring a queue manager to its pre-cluster state
- Maintaining a queue manager
- Refreshing a cluster queue manager
- Recovering a cluster queue manager



Hint: For complex procedures, start by doing a copy and paste of the IBM Knowledge Center documented procedure, and then replace the cluster and queue manager names with your cluster and queue manager names

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-13. Cluster-specific procedures from IBM Knowledge Center (partial)

When you administer a cluster environment, it is important to complete a set of commands to accomplish specific tasks in the order documented. The IBM Knowledge Center documents the correct sequence of commands to use to accomplish specific tasks. This slide shows a partial list of the documented processes. You look at selected processes in this unit. See the IBM Knowledge Center for other documented tasks that are not listed in this slide, or for details of tasks not feature in this unit.

Some of the most common tasks are selected for inclusion in this unit.

Although a task might appear confusing, a helpful hint is to copy the actual process from the IBM Knowledge Center documentation into a text editor. You can then carefully replace the names of the clusters, queue managers, and queues with the names of your clusters, queue managers, and queues. Then, remove extra text, and you have a procedure ready to be configured, or provided to the IBM MQ team for implementation.

You now look at some of the most common tasks in detail.

Add queue manager that uses separate transmission queue

- This procedure is created from IBM Knowledge Center section *Adding a queue manager to a cluster: Separate transmission queues*
- Added step numbers and other substitutions are displayed in bold red
 - Queue managers PARIS changed to MQ0X, LONDON changed to MQ0A
 - Cluster INVENTORY changed to WMADMCLS

Before you begin

The queue manager is not a member of any clusters.

The cluster exists; there is a full repository to which this queue manager can connect directly and the repository is available.

1. **Create queue manager MQ0X to listen on port 1681**
2. ALTER QMGR DEFCLXQ(CHANNEL)
3. Define a CLUSRCVR channel on queue manager MQ0X.

```
DEFINE CHANNEL(WMADMCLS.MQ0X) CHLTYPE(CLUSRCVR)
  TRPTYPE(TCP) CONNAME('localhost(1681)') CLUSTER(WMADMCLS)
  DESCR('Cluster-receiver channel for queue manager MQ0X')
```
4. Define a CLUSSDR channel on queue manager MQ0X.

```
DEFINE CHANNEL(WMADMCLS.MQ0A) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
  CONNAME('localhost(1657)') CLUSTER(WMADMCLS)
  DESCR('Cluster-sender channel from MQ0X to repository at MQ0A')
```

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-14. Add queue manager that uses separate transmission queue

As you learned earlier in this course, you can configure a cluster queue manager to use separate transmission queues. You do this configuration by either changing the queue manager `DEFCLXQ` attribute to `CHANNEL`, or changing the `CLCHNAME` attribute of the local queue to be used as the transmission queue.

This simple process is used to illustrate how easy it is to copy a more complex procedure from IBM Knowledge Center and create your own document while you are learning to administer the cluster. You might use this document for your own work, or provide it to new administrators as guidance. After you become proficient with cluster administration, you can skip the copy-and-change-procedure hint.

In this slide, the process was copied directly from IBM Knowledge Center. Then, the steps were isolated and numbered, and the queue manager and repository names were changed to the names of the resources used in the slide. The writing in red was added.

For purposes of specifying a separate transmission queue, the change can be done before or after the queue manager joins the cluster. For this example,

- The queue manager `DEFCLXQ` attribute is changed to `CHANNEL before` the queue manager joins the cluster. However, in this case, the cluster channels are not expected to be in `SWITCHING` state at any time because the queue manager was not in the cluster.

- After the queue `DEFCLXQ` attribute of the queue manager is changed, you define the CLUSRCVR and CLUSSDR channels. A partial repository queue manager is assumed.

Removing a cluster queue: Remove WM253.IN from MQ03

- At MQ03, alter the cluster queue so that it does not reference the cluster

```
ALTER QLOCAL(WM253.IN) CLUSTER('')
```

- From a full repository queue manager, check that the queue no longer displays as a cluster queue

```
DIS QCLUSTER(WM253.IN) CLUSQMGR
```

- At MQ03, set the queue to **PUT (DISABLED)**

```
ALTER QLOCAL(WM253.IN) PUT(DISABLED)
```

- At MQ03, verify that the number of processes for this queue is zero

```
DISPLAY Q(WM251.IN) IPPROCS, OPPROCS, and CURDEPTH
```

- From each of the other cluster member queue managers, ensure that none of the channels to MQ03 are INDOUBT

```
DISPLAY CHSTATUS(WMADMCLS.MQ03) INDOUBT
```

- When certain no messages can arrive, delete the queue from MQ03

```
DELETE QL(WM253.IN)
```

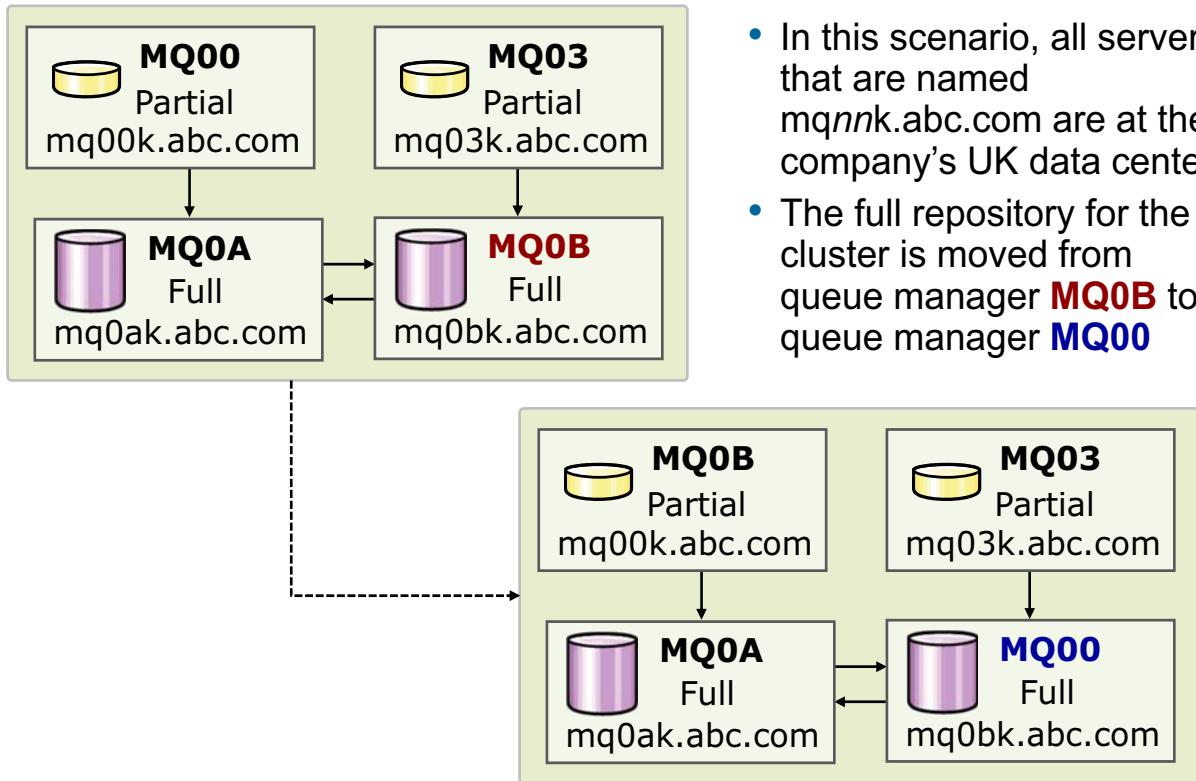
Figure 4-15. Removing a cluster queue: Remove WM253.in from MQ03

Processes to follow can be more or less involved, depending on the complexity of the process and the number of queue managers in the cluster.

In this slide, you look at the process to use when you remove a cluster queue. However, this process might become a section of another process, such as removing a queue manager.

- For this process, you start by removing the cluster name from the queue **CLUSTER** attribute.
- Next, you start a `rwmqsc` session in a full repository queue manager, and ensure that the queue is no longer showing. Ensure that you use the **CLUSQMGR** attribute in the command. If the queue is hosted in other cluster queue managers and you are removing it from one queue manager, you must ensure that you check for the absence of the queue for that queue manager.
- Back at the queue manager from which the cluster queue is removed, disable the queue.
- Then, also from the queue manager from which the cluster queue is removed, check that no processes or messages are in the queue; all counts are expected to be zero.
- Still at the MQ03 queue manager, ensure that the channel status for the **CLUSRCVR** queue is not INDOUBT.
- After you are convinced no other messages are sent to the cluster queue to be deleted from queue manager MQ03, proceed to delete the queue from the queue manager.

Moving a repository to a different queue manager



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-16. Moving a repository to a different queue manager

The move of a repository to a different queue manager might have different requirements.

- In this graphic, the queue manager where the repository is to be moved to is a differently named queue manager. The full repository is moved from queue manager **MQ0B** to queue manager **MQ00**. Such a change might be required because a queue manager with a higher capacity server might be needed.
- It is also possible that the repository is moved to a queue manager of the same name, but in a different server. Such a change might be common in environments where a number of servers change location, for example from the UK to the US.

Moving repositories (1 of 3)

1. Alter MQ00 to make it a full repository queue manager. **From MQ00:**

```
ALTER QMGR REPOS (WMADMCLS)
```

2. Add a **CLUSSDR** channel on MQ00 that points to MQ0A, the other full repository

```
DEFINE CHANNEL (WMADMCLS.MQ0A) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
+ CONNAME(MQ0A.ABC.COM) CLUSTER(WMADMCLS)
```

3. Define a **CLUSSDR** channel on MQ0A that points to MQ00. **From MQ0A:**

```
DEFINE CHANNEL (WMADMCLS.MQ00) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
+ CONNAME(MQ00.ABC.COM) CLUSTER(WMADMCLS)
```

4. Critical checkpoint: Check that queue manager MQ00 now has a full repository. **From MQ00:**

```
DIS QCLUSTER(*) CLUSTER (WMADMCLS) CLUSQMGR
DIS CLUSQMGR(*) CLUSTER (WMADMCLS) DEFTYPE STATUS
```

- Check that the expected cluster queues are visible
- If there are errors in the **DIS CLUSQMGR**, check that the new cluster channels are defined correctly

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-17. Moving repositories (1 of 3)

The first set of steps focuses on building up a full repository in the queue manager that is to become the new full repository, MQ00.

- At the MQ00 queue manager, alter the queue manager REPOS attribute to make MQ00 a full repository.
- Check what queue manager is used by the MQ00 queue manager as a full repository. If MQ00 is not already using queue manager MQ0A as the CLUSSDR full repository, define a new CLUSSDR channel from MQ00 to the full repository MQ0A. In the diagram, the MQ00 CLUSSDR channel is already pointing to queue manager MQ0A, so no change is needed.
- Both full repository queue managers must point to each other. On the MQ0A queue manager, create a CLUSSDR channel to point to MQ00 as the new second full repository.
- Check that MQ00 now has a full repository by issuing the **DIS QCLUSTER** and **DIS CLUSQMGR** commands. Use the “ALL” in the **DIS CLUSQMGR**, or at least request the **DEFTYPE** and **STATUS** attributes to ensure that the command output does not show any errors.
- If any configuration errors are found, correct them before you continue.

Moving repositories (2 of 3)

5. Change queue manager MQ0B to indicate that it is not a full repository. From **MQ0B**:

```
ALTER QMGR REPOS (' ')
```

6. Change the manually defined MQ00 CLUSSDR now obsolete due to the full repository change

- In this case, MQ00 as a partial repository queue manager was already pointing to MQ0A as a full repository; this step can be skipped:

```
STOP CHANNEL (WMADMCLS.MQ0A)
DELETE CHANNEL (WMADMCLS.MQ0A)
START CHANNEL (WMADMCLS.MQ0A)
```

7. Change other manually defined MQ00 CLUSSDR now obsolete due to the full repository change

- Type the sequence of commands on MQ03
- From **MQ03**, WMADMCLS.MQ0B is now used as a CLUSSDRA channel to MQ0B

```
STOP CHANNEL (WMADMCLS.MQ0B)
DELETE CHANNEL (WMADMCLS.MQ0B)
START CHANNEL (WMADMCLS.MQ0B)
```

Figure 4-18. Moving repositories (2 of 3)

- You now move to the old full repository queue manager, MQ0B. “Moving to the old full repository queue manager” implies that a runmqsc session for MQ0B is to be opened. Change the queue manager to indicate that MQ0B is no longer a full repository by setting the queue manager REPOS attribute to blanks.
- Step 6 might appear tricky. However, since MQ00 was already using MQ0A as its full repository, for this case, the step can be skipped.
- In step 7, queue manager MQ03 used to point to MQ0B as its full repository. The MQ03 partial repository queue manager is now going to point to the new MQ00 full repository. You issue the three steps. First, you stop any existing dynamic CLUSSDRB dynamic channels from MQ03 to MQ0B. Next, you delete the explicitly defined WMADMCLS.MQ0B CLUSSDR from MQ00. Last, you start the new CLUSSDRA type channel from MQ00 to MQ0B.
- You repeat step 7 on any other cluster member queue managers that used to point to MQ0B as a full repository. The WMADMCLS cluster that was depicted earlier has four queue managers, so no others to process.

Moving repositories (3 of 3)

8. Replace all other manually defined CLUSSDR channels that point to MQ0B on with CLUSSDR channels that point to either MQ0A or MQ00
 - In the cluster that is depicted, from MQ03:

```
DEFINE CHANNEL (WMADMCLS.MQ00) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
+ CONNAME(MQ00.ABC.COM) CLUSTER(WMADMCLS)
```

9. From MQ03, type DIS CLUSQMGR to ensure that the new explicitly defined CLUSSDR channel, WMADMCLS.MQ00, started as a CLUSSDRB dynamic channel.

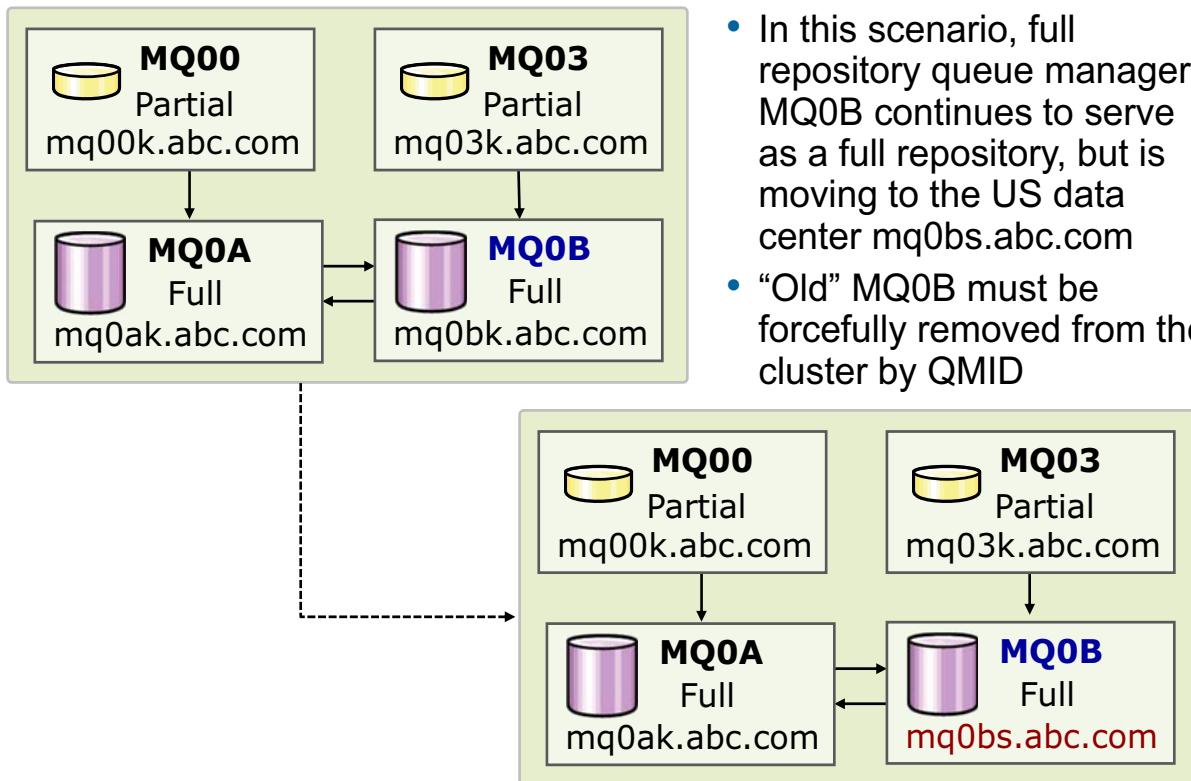
Figure 4-19. Moving repositories (3 of 3)

The cluster that was depicted earlier showed four queue managers. For this cluster, you move the CLUSSDR channel to the new MQ00 full repository. For this reason, you create an explicitly defined CLUSSDR channel on MQ03 to point to MQ00. This channel, WMADMCLS.MQ00, starts as a dynamic CLUSSDRB channel.

If other queue managers pointed to MQ0B as the full repository, you repeat steps 8 and 9. Regarding the cluster as depicted, all four queue managers are already done.

It is also not necessary to move all queue managers with former CLUSSDRB connections to MQ0B to go to MQ00. Some of these CLUSSDR connections might also point to the MQ0A full repository. However, MQ0A must point to the other full repository, which in this case is MQ00.

Moving a full repository queue manager to a different host



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-20. Moving a full repository queue manager to a different host

Another possible scenario occurs when a full repository queue manager name remains the same, but the queue manager itself is moved to a different host. In this case, the process to move the queue manager is a combination of the move process, with eventual removal of the older version.

The same queue manager might be moved to a different host for different reasons, such as:

- An organization decides to use cluster queue managers that are dedicated to the full repository, without application work. This option might be common in organizations where clusters were originally introduced as a prototype, and then the environment grew and might need different standards.
- An organization might have server farms at different locations, and decide to keep one full repository in each location.

The key consideration is that the "old" queue manager eventually needs to be removed by using the QMID to ensure that the correct version of the queue manager is removed from the cluster. Later in this unit, you learn a command to use to forcefully remove a queue manager.

Considerations when removing a queue manager from a cluster

Determine whether the queue manager to be removed represents one of the special situations

- If the queue manager to be removed unable to reach one of the cluster full repositories, follow the process that is documented in slide

Removing MQ03 from the cluster: Alternative scenario

- If the queue manager to be removed hosts a full repository, follow the process that is documented in slide **Moving a repository to a different queue manager** before removing the queue manager
- If the queue manager to be removed hosts any cluster queues, follow the process that is documented in slide **Removing a cluster queue** before removing the queue manager
- If the queue manager to be removed hosts publish/subscribe cluster topics delete the topics, or move the topics to other hosts
- If the queue manager to be removed is a simple case without extra considerations, follow the process that is documented in slides

Removing MQ03 from the cluster: Basic scenario

Figure 4-21. Considerations when removing a queue manager from a cluster

Under most circumstances, the removal of a queue manager from a cluster is accomplished by removing the cluster settings in a predefined order without the need for extra commands. At other times, the removal of the queue manager is more complex, for example:

- If the queue manager lost connectivity with both full repository queue manager, an alternative method to remove the queue manager is followed.
- If the queue manager to be removed hosts a full repository, you must move the full repository to a different queue manager by following the process noted in an earlier slide. This process must be done before the queue manager can be removed.
- If the queue manager hosts any cluster queues, the process to remove cluster queues from the queue manager must be completed.
- If the queue manager to be removed is used for publish/subscribe applications and contains cluster topics, you must remove the topics, or move the topics to other hosts. You learn more about publish/subscribe in a later unit.

If the queue manager to be removed does not fall under any of the circumstances that were described, you can remove the queue manager from the cluster by using the basic scenario.

Removing MQ03 from the cluster: Basic scenario (1 of 2)

From the MQ03 queue manager to be removed from the cluster:

1. If queue the queue manager to be removed is a full repository, alter the queue manager to set the REPOS (or REPOSNL) attributes to blank
2. Confirm whether the changes made propagated around the cluster
 - DIS CLUSQMGR (MQ03) QMTYPE
 - Results are expected to confirm that the QMTYPE is NORMAL
3. Suspend queue manager from the cluster in QUIESCE mode
SUSPEND QMGR CLUSTER (WMADMCLS)
4. Ensure that queue manager MQ03 is suspended before you continue
 - DIS CLUSQMGR (*) SUSPEND
 - Response must contain SUSPEND (YES)
5. Remove the CLUSRCVR definition from the cluster
ALTER CHANNEL (WMADMCLS.MQ03) CHLTYPE (CLUSRCVR) +
CLUSTER (' ')
6. Stop the cluster receiver channel
STOP CHL (WMADMCLS.MQ03)

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-22. Removing MQ03 from the cluster: Basic scenario (1 of 2)

For this process, it is assumed that the queue manager to be removed does not hold a full repository, or fall into any of the scenarios described in the earlier slide. Also, assume that queue manager MQ03 is the queue manager to remove from the cluster.

In this basic process, you remove the cluster settings for the queue manager by following the process that is shown. From the queue manager to be removed:

- Remove the full repository designation from MQ03 by altering the queue manager REPOS or REPOSNL attributes by blanks.
- Display the queue manager by using DIS CLUSQMGR with the QMTYPE property. It is expected to return QMTYPE(NORMAL), which means it is not a repository.
- Suspend the queue manager.
- Remove the CLUSTER value from the CLUSRCVR definition.
- Stop the CLUSRCVR channel.

Removing MQ03 from the cluster: Basic scenario (2 of 2)

From the MQ03 queue manager to be removed from the cluster:

7. Remove the leftover CLUSRCVR channel
`DELETE CHANNEL (WMADMCLS . MQ03)`

From a cluster full repository queue manager:

8. `DIS CLUSQMGR (MQ03)`

Figure 4-23. Removing MQ03 from the cluster: Basic scenario (2 of 2)

After you stop the CLUSRECVR channel, delete the CLUSRCVR channel for the queue manager that is being removed.

You now move to a full cluster repository, and use `DIS CLUSQMGR` with `MQ3` as a parameter for the CLUSQMGR to ensure that the queue manager is no longer found in the cluster.

A



RESET CLUSTER

- Forcibly removes a queue manager from a cluster
- Not used under normal circumstances
- Possible cases that require the **RESET CLUSTER** command
 - A queue manager to be removed from a cluster is unable to reach a full repository
 - A queue manager in the cluster is being moved to a new server, QMID must remove the “old” queue manager
- Parameters
 - ACTION
 - QMNAME or QMID
 - QUEUES
- Use examples:

```
reset cluster(WMADMCLS) QMNAME(MQ0X)
  action(FORCEREMOVE) queues(YES)
```

```
reset cluster(WMADMCLS) QMID('MQ0X_2017-05-03_05.53.55')
  action(FORCEREMOVE) queues(NO)
```

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-24. RESET CLUSTER

If you need to remove a queue manager from a cluster when the queue manager to be removed lost connectivity to both full repositories, you use the **RESET CLUSTER** command. You also need to use **RESET CLUSTER** in the case that a full repository queue manager “moves” to a different server, yet keeps the same queue manager name.

The **RESET CLUSTER** command can be used to remove the queue managers from the cluster either by queue manager name, or by queue manager ID, or QMID. When the QMID option is used, enclose the QMID in single quotation marks, or the command returns a syntax error.

The **QUEUES** option of **RESET CLUSTER** can be used to remove the cluster queues associated with the queue manager. **QUEUES(NO)** is the default. Use of **QUEUES(YES)** removes the cluster queues.

Removing MQ0X from the cluster: Alternative scenario (1 of 2)

- At MQ0X, confirm and stop the cluster channels at queue manager

```
dis clusqmgr(*) deftype status ...
... CLUSQMGR(MQ0A) ... CHANNEL(WMADMCLS.MQ0A) ... DEFTYPE(CLUSSDRB)
... CLUSQMGR(MQ0B) ... CHANNEL(WMADMCLS.MQ0B) ... DEFTYPE(CLUSSDRA)
... CLUSQMGR(MQ0X) ... CHANNEL(WMADMCLS.MQ0X) ... DEFTYPE(CLUSRCVR)

stop chl(WMADMCLS.MQ0X) mode(force)
stop chl(WMADMCLS.MQ0A)
stop chl(WMADMCLS.MQ0B)
```

- At MQ0X, confirm whether all channels stopped

```
dis clusqmgr(*) deftype status
```

- At MQ0X, remove the CLUSSDR channel to the full repository, MQ0A

```
delete chl(WMADMCLS.MQ0A)
```

Figure 4-25. Removing MQ0X from the cluster: Alternative scenario (1 of 2)

This alternative process to remove a queue manager from a cluster is not as kind as the basic process. From the queue manager to be removed, MQ0X in this example, you use `DIS CLUSQMGR` with the `deftype` and `status` parameter to determine all the cluster channels for the queue manager.

You stop the CLUSRCVR type with mode(FORCE), and stop each CLUSSDRA or CLUSSDRB channels without force.

You repeat `DIS CLUSQMGR` again. No channels are expected in the display.

You now delete the CLUSSDR channel from MQ0X to its full repository. The narrative continues in the next page.

Removing MQ0x from the cluster: Alternative scenario (2 of 2)

- At full repository MQ0A, purge queue manager MQ0X from the cluster

```
dis clusqmgr(MQ0X) QMID ... ...
CLUSQMGR(MQ0X) ... ... QMID(MQ0X_2017-05-03_05.53.55)

reset cluster(WMADMCLS) QMID('MQ0X_2017-05-03_05.53.55')
action(forceremove) queues(no)

AMQ8559: RESET CLUSTER accepted.
```

- Confirm that the MQ0A repository does not show queue manager MQ0X

```
dis clusqmgr(MQ0X) ...
AMQ8147: IBM MQ object MQ0X not found.
```

- Repeat the DIS CLUSQMGR(MQ0X) command from each of the other queue managers in the cluster to confirm that MQ0X is not longer found

- Remove the CLUSRCVR channel from MQ0X

```
delete chl(WMADMCLS.MQ0X)
```

[Cluster administration tasks and commands](#)

[© Copyright IBM Corporation 2017](#)

Figure 4-26. Removing MQ0x from the cluster: Alternative scenario (2 of 2)

Next, you again use `DIS CLUSQMGR`, this time with `QMID` to obtain the `QMID` of the queue manager to be removed. You might want to copy the `QMID` so that you can paste it in the `RESET CLUSTER` command, or you might opt to copy the `QMID` from the display. However, whichever method you choose, enclose the `QMID` in single quotation marks in the `RESET CLUSTER` command. An example of the `RESET CLUSTER` command is displayed. The expected response is `RESET CLUSTER accepted`.

You now proceed to one of the full repositories, and repeat the `DIS CLUSQMGR` command for the deleted queue manager, or MQ0X. The expected result is that the object is not found.

Depending on the size of the cluster, it might take longer for the `DIS CLUSQMGR` for the removed queue manager to stop “finding” MQ0X.

You repeat `DIS CLUSQMGR(MQ0X)` from each remaining queue manager in the cluster to confirm that MQ0X is no longer found. Knowledge of MQ0X might take a day or so to be removed from the other cluster member queue managers.

Last, after you no longer find any mention of MQ0X in the other cluster member queue managers, you remove the `CLUSRCVR` type channel from MQ0X.

REFRESH CLUSTER

AMQ9875: REFRESH CLUSTER processing started for cluster.

EXPLANATION: Refresh cluster processing started for cluster(*)
 A REFRESH CLUSTER command has been issued on this queue manager. In phase one this will discard all locally cached information for the cluster and request new information from other members of the cluster when necessary. Phase two processes the information received. **For large cluster configurations this process can take a significant time, especially on full repository queue managers, and during this time applications attempting to access cluster resources may see failures to resolve cluster resources. In addition, cluster configuration changes made on this queue manager may not be processed until the refresh process has completed.**

ACTION: Defer any cluster related work on this queue manager until both phases are complete. **Message AMQ9442 or message AMQ9404 will be issued to this log at the end of phase one. Completion of phase two can be determined when SYSTEM.CLUSTER.COMMAND.QUEUE has reached a consistently empty state.**

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-27. REFRESH CLUSTER

REFRESH CLUSTER is an often misused command whose purpose is to refresh a cluster queue manager's repository information. Users new to clusters tend to use REFRESH CLUSTER as a universal solution when in fact use of the command might worsen a cluster problem by generating significant work in the system. The best testimonials of the consequences of REFRESH CLUSTER are the entries that are made in the queue manager log when the command is requested. You see wording such as:

- . . . for large cluster configurations this process can take a significant time
- . . . during this time, applications that attempt to access cluster resources might see failures to when attempting to resolve the resources
- . . . cluster configuration changes made on this queue manager might not be processed until the refresh process is completed

The IBM Knowledge Center has other warnings for the REFRESH CLUSTER command in the usage notes. In the IBM Knowledge Center you see: "*For large clusters, use of the REFRESH CLUSTER command can be disruptive to the cluster while it is in progress, and again at 27-day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers*". When you issue REFRESH CLUSTER, the command returns right away. However, the return of the response does not signify that the command completed its processing. REFRESH CLUSTER is a significantly disruptive command.

REFRESH CLUSTER command

- Discards locally held information about a cluster that depends on options that are used
- **REFRESH CLUSTER (CLUSTERNAME) REPOS (NO)**
 - Locally defined information is retained
 - Information other than local information is rebuilt from full repositories
 - **AMQ8558: REFRESH CLUSTER accepted** does not mean that the process ended
- **REFRESH CLUSTER (CLUSTERNAME) REPOS (YES)**
 - Queue manager where command runs cannot hold a full repository at the time the command is run
 - Extra considerations for cluster sender channels
- Justified reasons to use **REFRESH CLUSTER**
 - Messages are removed from the SYSTEM.CLUSTER.COMMAND.QUEUE
 - Suggested by IBM Service
 - The CLUSRCVR channels were removed from a cluster, or CONNAME parameter altered on full repository queue managers
 - The same name was used for a CLUSRCVR channel on more than one queue manager in a cluster
 - **RESET CLUSTER ACTION (FORCEREMOVE)** was issued in error
 - Certain queue manager restart errors

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-28. REFRESH CLUSTER command

On distributed platforms, REFRESH CLUSTER has a REPOS option.

- REPOS(NO), which is the default, is the least disruptive of the two REPOS choices.
 - Queue manager keeps knowledge of locally defined queue manager and cluster queues
 - Ensures that information about other queue managers is marked as full repositories
 - All other information is discarded, and refreshed from the other cluster full repositories
 - Channels are not stopped during the process
- REPOS(YES) is the most disruptive of the REPOS options.
 - Includes REPOS(NO) processing
 - Objects that represent the repository queue manager are also refreshed
 - If queue manager is a full repository, it must be temporarily changed to a partial repository before the REFRESH CLUSTER is run
 - Causes the cluster channels to stop and start

In the z/OS environment, a CMDSCOPE, or command scope, attribute influences the REFRESH CLUSTER command when the queue manager is a member of a queue-sharing group.

Regardless of the notoriety of the `REFRESH CLUSTER` command, sometimes its use is valid and justifiable. The justifiable uses for `REFRESH CLUSTER` are listed on the slide.

Always read all the `REFRESH CLUSTER` command notes that are found in the IBM Knowledge Center before you use `REFRESH CLUSTER`.

REFRESH CLUSTER command phase

AMQ9442: Phase one of REFRESH CLUSTER has completed.

EXPLANATION:

Phase one of REFRESH CLUSTER has completed. **The Refresh Cluster(*) command caused 10 objects to be refreshed and republished to 3 queue managers. Applications attempting to access cluster resources may see failures to resolve cluster resources until phase two of REFRESH CLUSTER is complete. Phase two is complete once all new information has been received from other members of the cluster.** Monitor your SYSTEM.CLUSTER.COMMAND.QUEUE to determine when it has reached a consistently empty state to indicate that the refresh process has completed.

ACTION:

None.

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-29. REFRESH CLUSTER command phase

The first slide for the REFRESH CLUSTER topic showed the first of two messages that are generated in the queue manager log when the REFRESH CLUSTER command is issued. The first slide cited message AMQ9442 as an indication that the first phase of the REFRESH CLUSTER processing was completed.

In this slide, you see message AMQ9442. You also see some statistics of the objects that were refreshed.

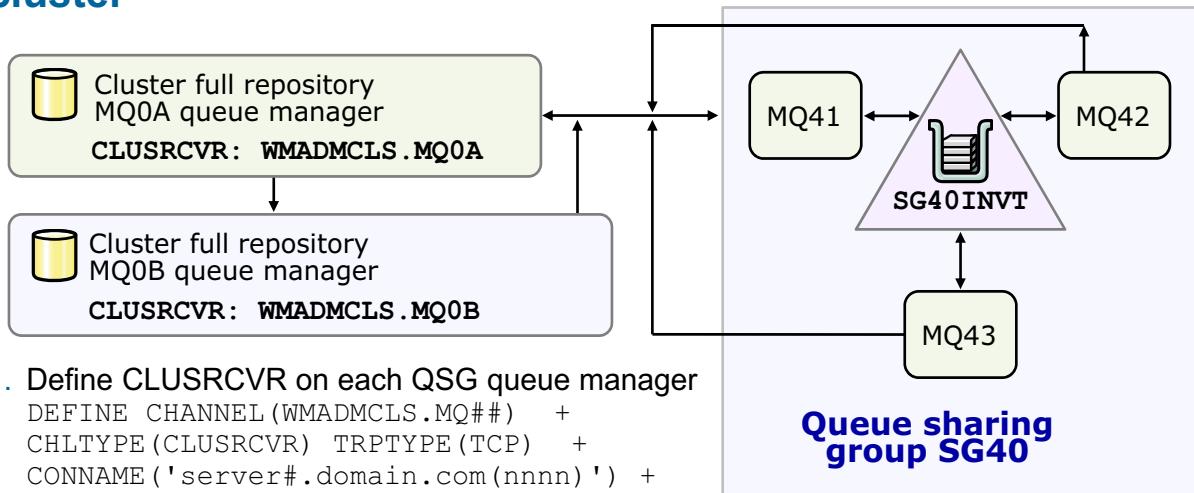
Following the statistics, you again see warnings of problems that might be experienced while the process is running.

The log entry also reiterates how to determine when the REFRESH CLUSTER command completes. REFRESH CLUSTER generates a significant number of requests that go to the SYSTEM.CLUSTER.COMMAND.QUEUE. After the SYSTEM.CLUSTER.COMMAND.QUEUE stabilizes back to an empty state, the REFRESH CLUSTER processing continues.

The SYSTEM.CLUSTER.COMMAND.QUEUE is one of the queues to monitor in a cluster environment. You learn about the other resources to monitor in a later slide in this unit.

IBM Training

Incorporating a queue-sharing group into an existing cluster



1. Define CLUSRCVR on each QSG queue manager

```
DEFINE CHANNEL(WMADMCLS.MQ##) +
CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
CONNNAME('server#.domain.com(nnnn)') +
CLUSTER(WMADMCLS)
```

2. Define **one** CLUSSDR for the QSG: using MQ41

```
DEFINE CHANNEL(WMADMCLS.MQ0A) +
CHLTYPE(CLUSSDR) TPTYPE(TCP) +
CONNNAME('mvs++++.ilsvpn.ibm.com(1621)') +
CLUSTER(WMADMCLS) QSGDISP(GROUP)
```

3. Define **one** local queue on one of the QSG queue managers: using MQ41

```
DEFINE QLOCAL(SG40INV) CLUSTER(WMADMCLS) QSGDISP(SHARED) +
CFSTRUCT(APPL7)
```

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-30. Incorporating a queue-sharing group into an existing cluster

This diagram illustrates the process that is used to incorporate a queue-sharing group into an existing cluster, and create a clustered shared queue. Remember that for a z/OS queue-sharing group, all queue managers see the shared queue, so it is not necessary to define a shared clustered queue in each queue-sharing-group queue manager. This detail might seem odd when you start working with queue-sharing groups.

- Cluster WMADMCLS has two full repository queue managers: MQ0A and MQ0B.
- Queue-sharing group SG40 is composed of three queue managers: MQ41, MQ42, and MQ43.
- A highly available (shared) queue called SG40INV.

The steps to follow are:

1. Select one full repository in cluster WMADMCLS. You select MQ0A.
2. For each queue manager in the queue-sharing group, create one CLUSRCVR channel:
 - a. On MQ41, define CLUSRCVR channel WMADMCLS.MQ41
 - b. On MQ42, define CLUSRCVR channel WMADMCLS.MQ42
 - c. On MQ43, define CLUSRCVR channel WMADMCLS.MQ43

3. Select one queue manager in the queue-sharing group to define the **shared** CLUSSDR channel for the queue-sharing group. In the diagram, channel WMADMCLS.MQ0A is defined by setting the QSGDISP attribute to GROUP.
4. Issue command `DIS CLUSQMGR(*)` from each queue manager in the queue-sharing group to confirm that the queue-sharing group was successfully added to the cluster.

Define queues as required. It is not necessary to define a shared queue more than one time.

Recap of resources to administer your cluster environment

- Use DIS CLUSQMGR
- Review typing of channel names and connectivity information
- Check the queue manager logs
- If queue manager external to cluster use DIS CHSTATUS
- Monitor message counts on cluster transmission queues
- Check the message counts on other SYSTEM.CLUSTER.* queues
- DIS QCLUSTER does not show cluster queues until after first MQPUT to queue if requested from a partial repository queue manager
- Avoid use of the **REFRESH CLUSTER** command

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-31. Recap of resources to administer your cluster environment

Throughout this unit, you learned how a cluster is administered by using some of your existing IBM MQ skills, along with the documented cluster processes and commands. The following list is a summary of the cluster administration resources.

- DIS CLUSQMGR, issued with the DEFTYPE and STATUS attributes, is the best tool to check the health of your queue manager in the cluster, and the best tool to verify a newly configured cluster.
- As in the distributed message channels, you can find problems with channels RETRYING or other errors that are displayed in DIS CLUSQMGR or the queue manager logs. If so, check the typing of the cluster channel names and connectivity information for errors.
- Check the queue manager logs for channel problems. You might need to check the logs for more than one queue manager. Start with the log for the queue manager to which the queue manager that is reporting the error is trying to connect.
- Keep check of the SYSTEM.CLUSTER.* named queues, and the cluster transmission queues. You are going to learn more about monitoring later in this unit.
- You might use DIS QCLUSTER from a cluster queue manager without a full repository, and the queue manager that you type the request from did not try to put a message to the cluster queue. If so, your results are that the queue is not found, which is the intended “need-to-know”

behavior. A partial repository queue manager “learns” about other queues after the first time it needed to find information about the queues by using sample amqsput to send a message to a queue. After queue name resolution had to take place from the queue manager you are working from, DIS QCLUSTER works. Putting a message to a queue, specifically the MQOPEN, initiates queue name resolution. DIS QCLUSTER does not initiate queue name resolution.

- REFRESH CLUSTER does not correct the errors, but might worsen a cluster situation. Limit use of the `REFRESH CLUSTER` command to those situations where its use is justifiable.

Monitor SYSTEM.CLUSTER and application queue instances

- Transmission queues, whether a custom queue is used, or the default SYSTEM.CLUSTER.TRANSMIT.QUEUE
 - Monitor the depth of the queue according to the cluster design
 - If messages build up, you need to identify which channel is backed up
 - From the cluster queue manager with the backed-up transmission queue, use the command in the example to investigate the problem channel
- ```
DIS CHSTATUS (*) where (XQMSGSA GT 1)
```
- SYSTEM.CLUSTER.REPOSITORY.QUEUE
    - Holds the repository information
    - CURDEPTH should be greater than zero
  - SYSTEM.CLUSTER.COMMAND.QUEUE
    - IPPROCS should always be 1
    - CURDEPTH should be zero or decrementing
  - SYSTEM.CLUSTER.HISTORY.QUEUE
  - Application clustered queue instances
    - Ensure that applications are consuming message at each queue instance
    - Check the cluster queue monitoring sample program AMQSCLM

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

*Figure 4-32. Monitor SYSTEM.CLUSTER and application queue instances*

An IBM MQ clustered environment needs to be monitored in a similar way to any other IBM MQ environment, with a few extra considerations.

- Monitor the cluster transmission queues. You need to baseline your expected message volumes to establish a normal depth of messages in the transmission queues. Normally, messages travel through the transmission queues quickly. A large accumulation of messages in a transmission queue indicates a channel problem.
- If a queue manager's cluster transmission queue shows a high message count, you can display the channel status as illustrated on the slide. This display lists the messages to be processed by all channels that use the problem transmission queue. The command helps you identify the channel that is causing the problem.
- The SYSTEM.CLUSTER.REPOSITORY queue is always expected to have messages that contain, as the queue name implies, the full or partial repository information.
- You might see that the SYSTEM.CLUSTER.COMMAND.QUEUE has a higher than usual message count due to a couple of situations:
  - The repository manager might fail. In distributed platforms, the repository manager is process amqrrmfa. If you do not find that this process is running, you need to determine

what brought it down, and restart the queue manager to restart the repository manager. On z/OS, the repository manager runs in the channel initiator started task.

- A REFRESH CLUSTER command might be running. In this case, the emptying of the queue indicates the end of the REFRESH CLUSTER process.
- The SYSTEM.CLUSTER.HISTORY.QUEUE captures a view of a cluster. The information is for the use of IBM support. This queue can be disabled. On distributed systems, the queue is defined automatically. On z/OS platforms, you might have to include the specific queue definition data that was set in your procedure.
- You also need to monitor the clustered application queues. Each server that hosts a clustered queue is supposed to have an application that consumes the messages. If an application stops consuming messages, a buildup of messages can take place. A cluster queue monitoring sample program, which is named `amqsclm`, might be of help in monitoring and redirecting messages in clustered application queues.

You learn more about `amqsclm` in a later unit.

## Using IBM MQ Explorer to work with clusters

- IBM MQ Explorer obtains cluster information from one of the cluster's full repository queue managers
- Can change the information source for IBM MQ Explorer

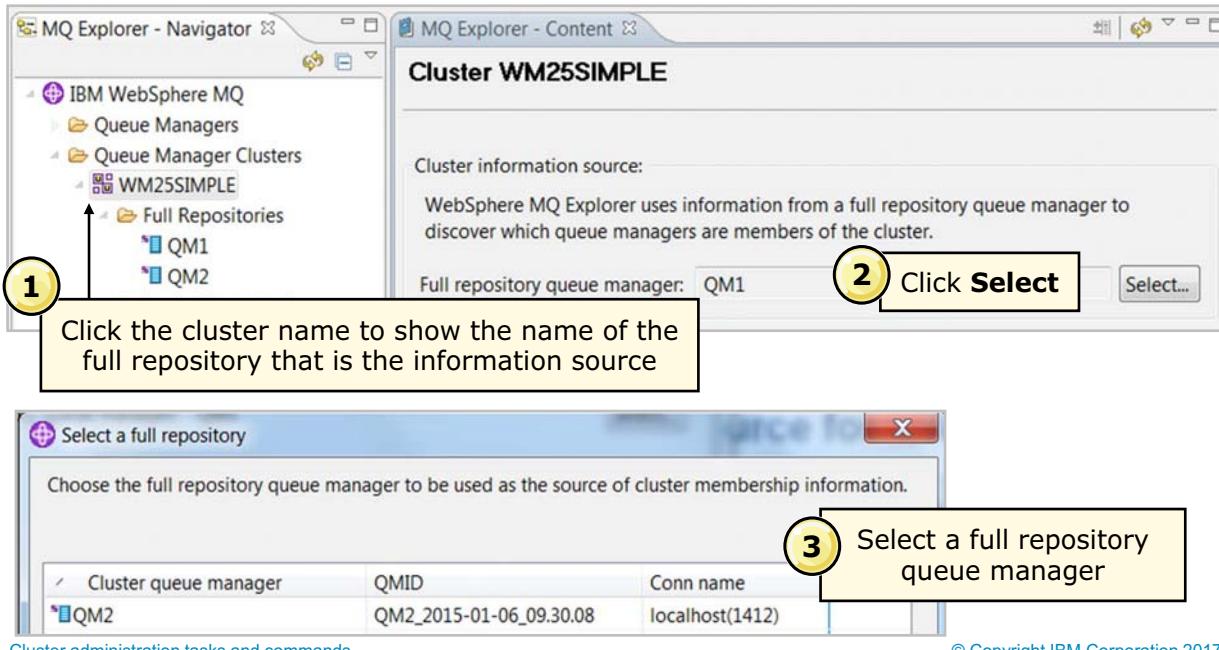


Figure 4-33. Using IBM MQ Explorer to work with clusters

When you open IBM MQ Explorer, you can select the full cluster repository queue manager as indicated by the numbered boxes.

You can also expand the “+” menus for the queue manager clusters, then the “+” for the cluster name (WM25SIMPLE in the diagram), and select one of the two full repositories.

**IBM Training**

**Display cluster status**

Repository data for queue manager QM1

Cluster Queues Cluster Topics Cluster-sender Channels Cluster-receiver Channels

QM2 - Full Repository

QM1 → 1 QM2

| Channel name   | Cluster queue manager | Queue manager type | Definition type              | Xmit protocol | Channel status | Suspend | Cluster name | Mode name |
|----------------|-----------------------|--------------------|------------------------------|---------------|----------------|---------|--------------|-----------|
| WM25SIMPLE.QM2 | QM2                   | Repository         | Auto explicit cluster-sender | TCP           | Running        | No      | WM25SIMPLE   |           |
| WM25SIMPLE.QM3 | QM3                   | Normal             | Auto cluster-sender          | TCP           | Running        | No      | WM25SIMPLE   |           |
| WM25SIMPLE.QM4 | QM4                   | Normal             | Auto cluster-sender          | TCP           | Running        | No      | WM25SIMPLE   |           |

Status summary

Scheme: Standard for Cluster Queue Managers - Distributed

**WM25SIMPLE.QM2 - Channel Status**

Queue Manager: QM1 Channel Name: WM25SIMPLE.QM2

Channel Type: Auto explicit cluster-sender

Filter: Standard for Channel Status

| Channel name   | Channel type   | Channel status | Conn name       | Queue manager name | Remote queue manager | Transmission queue         |
|----------------|----------------|----------------|-----------------|--------------------|----------------------|----------------------------|
| WM25SIMPLE.QM2 | Cluster-sender | Running        | 127.0.0.1(1412) | QM1                | QM2                  | SYSTEM.CLUSTER.TRANSMIT.QU |

Right-click channel and then click **Status** to display channel status

© Copyright IBM Corporation 2017

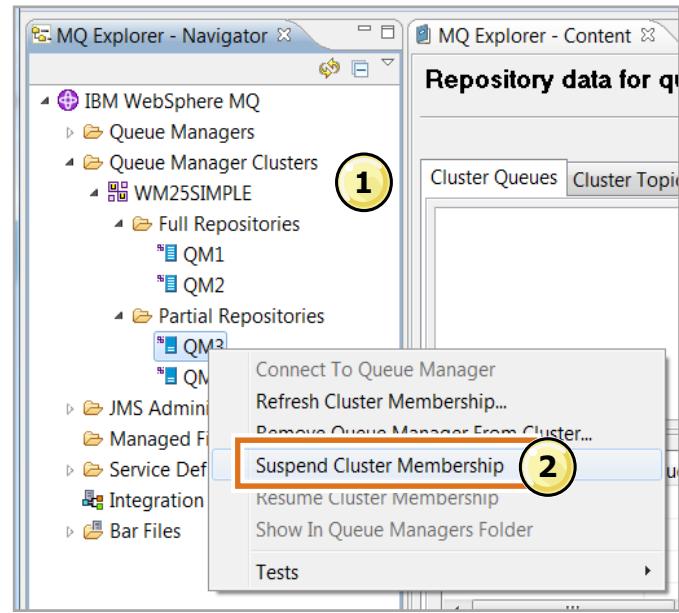
Figure 4-34. Display cluster status

The name of this title is a misnomer. What you look at is the status of the channels in the cluster, not “cluster status”. In the first view of cluster-sender channels, you see the cluster-sender channels that point from one full repository to the other full repository.

In the second list of channels, you see the CLUSSDRA and CLUSSDRB channels.

## Suspend the cluster membership of a queue manager

- Temporarily prevent the queue manager from sharing its cluster queues and exchanging messages by using the cluster
- In the IBM MQ Explorer Navigator view, expand the cluster in the **Queue Manager Clusters** folder
  - Right-click the queue manager, and then click **Suspend Cluster Membership**



Cluster administration tasks and commands

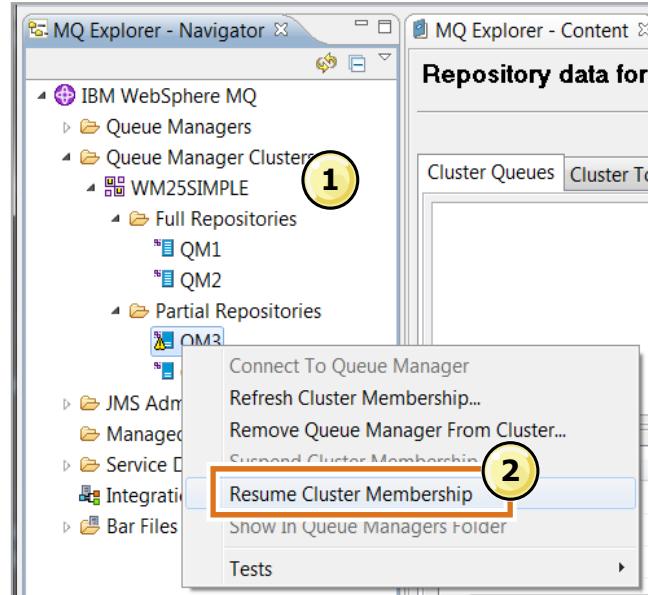
© Copyright IBM Corporation 2017

Figure 4-35. Suspend the cluster membership of a queue manager

This slide depicts how you would suspend a cluster queue manager by using IBM MQ Explorer.

## Resume the cluster membership of a queue manager

- Resume a suspended cluster queue manager without entering the connection details again
  1. In the IBM MQ Explorer Navigator view, expand the cluster in the **Queue Manager Clusters** folder
  2. Right-click the suspended queue manager and then click **Resume Cluster Membership**



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-36. Resume the cluster membership of a queue manager

This slide depicts how you reinstate a suspended queue manager to the cluster by using the `resume queue manager` command.

However, depending on the type of SUSPEND requested, you still need to follow the correct process. You also must determine whether any of the CLUSSDRx channels to the suspended queue manager from other cluster member queue managers might require a restart.

## Unit summary

- Summarize the similarities and differences in administering IBM MQ with and without clusters
- Describe the cluster-specific MQSC administration commands
- List the predefined procedures that are available to complete cluster tasks
- Describe how to create a cluster queue manager that uses separate transmission queues
- Describe the process that is used to remove a cluster queue from a queue manager
- Describe the process that is used to remove a cluster member queue manager from the cluster
- Describe the process that is used to move a full repository to a different cluster queue manager
- Explain how to add a queue-sharing group to an existing cluster
- Explain how to use the IBM MQ Explorer equivalent cluster capabilities to complete selected cluster administrative tasks
- Describe how to use selected IBM MQ utilities to work with cluster tasks

[Cluster administration tasks and commands](#)

© Copyright IBM Corporation 2017

Figure 4-37. Unit summary

## Review questions

1. True or False: The **SUSPEND QMGR** command always inhibits messages from arriving at the suspended queue manager.
2. Select the correct answer or answers. An administrator defines cluster queue SALES.IN in cluster queue managers MQ01 and MQ03. The administrator later uses a runmqsc session on cluster queue manager MQ0A and types a **DIS QCLUSTER (SALES.IN)**, but the queues are not found.
  - A. Administrator must check the MQ01 and MQ03 queue manager logs to look for a cluster error
  - B. Send a test message to queue SALES.IN from queue manager MQ0A and repeat the **DIS QCLUSTER** command
  - C. Type the **REFRESH CLUSTER** command to resolve the problem
  - D. All of the above
3. True or False: A valid use for the **REFRESH CLUSTER** is to reinstate a queue manager to the cluster that is accidentally forced out of the cluster by the **RESET QMGR** command



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-38. Review questions

## Review answers (1 of 2)

1. True or False: The **SUSPEND QMGR** command always inhibits messages from arriving at the suspended queue manager.  
The answer is False. If the **MODE (FORCE)** is not specified, messages can still reach the queue manager.
2. Select the best answer. An administrator defines cluster queue SALES.IN in cluster queue managers MQ01 and MQ03. The administrator later uses a runmqsc session on cluster queue manager MQ00 and types a **DIS QCLUSTER (SALES.IN)**, but the queues are not found.
  - A. Administrator must check the MQ01 and MQ03 queue manager logs to look for a cluster error
  - B. Send a test message to queue SALES.IN from queue manager MQ00 and repeat the **DIS QCLUSTER** command
  - C. Type the **REFRESH CLUSTER** command to resolve the problem
  - D. All of the above

The answer is B. The MQ00 repository learns about the new queues on a “need to know” basis. **REFRESH CLUSTER** does not fix errors and causes significant congestion in the cluster.

Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-39. Review answers (1 of 2)



## Review answers (2 of 2)

3. True or False: A justifiable use for the **REFRESH CLUSTER** command is to reinstate a queue manager to the cluster that was accidentally forced out of the cluster by the **RESET QMGR** command.

The answer is True. If **RESET QMGR** accidentally removes a queue manager, a justified use for the **REFRESH CLUSTER** command is to reinstate the queue manager to the cluster.



## Exercise: Working with cluster administration tasks

© Copyright IBM Corporation 2017  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Figure 4-41. Exercise: Working with cluster administration tasks

## Exercise objectives

- Add a queue manager that uses a separate transmission queue to the cluster
- Inhibit use of a queue manager by using the SUSPEND command
- Contrast the results and possible actions that the various SUSPEND modes require
- Describe how some actions that you take to administer cluster message channels resemble actions that you take to administer distributed message channels
- Restore use of a suspended queue manager by using the RESUME command
- Remove a queue manager from the cluster by using the RESET command
- Use the REFRESH CLUSTER command and review the entries that are generated in the queue manager log for this command
- Describe IBM MQ Explorer cluster administration capabilities



Cluster administration tasks and commands

© Copyright IBM Corporation 2017

Figure 4-42. Exercise objectives

---

# Unit 5. IBM MQ security and clusters

## Estimated time

01:00

## Overview

This unit teaches connection authentication, channel authentication, and object authentication, and the similarities of managing these security mechanisms for a cluster. The unit includes special considerations for the security mechanisms and considerations to observe when introducing SSL/TLS to a cluster environment. You also learn how to troubleshoot connection authentication, channel authentication, and object authorization problems.

## Unit objectives (1 of 2)

- Summarize security areas and terminology
- Describe IBM MQ connection authentication
- Identify the IBM MQ connection authentication initial settings in a new queue manager
- Explain how to modify connection authentication and respond to the new settings
- Describe channel authentication and the types of channel authentication records
- Describe the channel authentication initial default settings and rules in a new queue manager
- Explain channel authentication rule precedence
- Describe the channel authentication back-stop rule
- Explain how to set, change, and remove channel authentication records
- List considerations to observe for a successful channel authentication implementation

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-1. Unit objectives (1 of 2)

## Unit objectives (2 of 2)

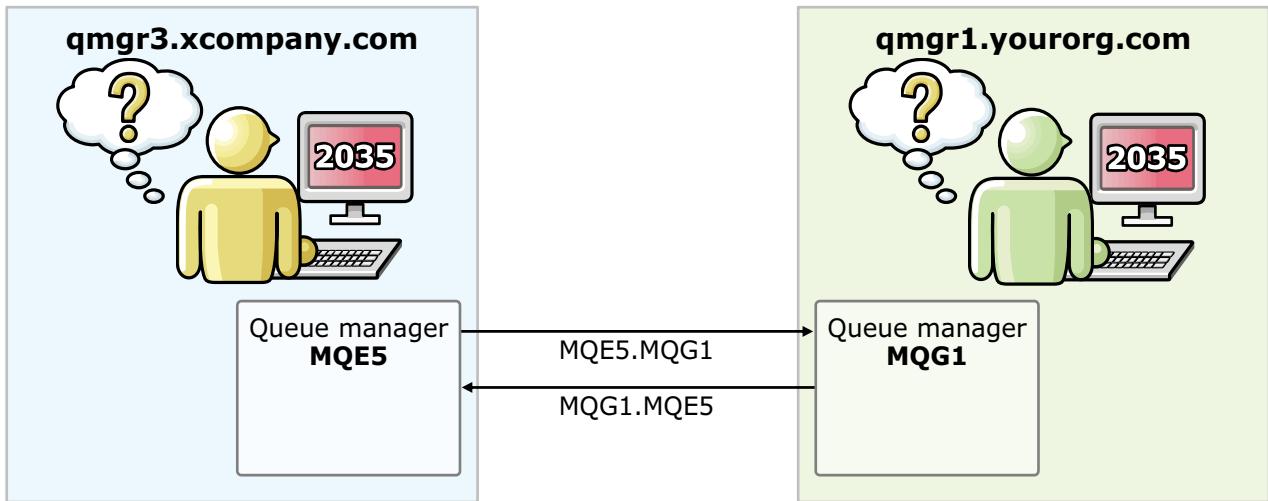
- Describe IBM MQ object authorizations
- Explain how to use the dspmqaut and setmqaut control commands to display and grant object authorizations
- Explain how to use the IBM MQ script commands DISPLAY AUTHREC and SET AUTHREC to display and grant object authorizations
- Differentiate when to use the object authorization control commands, and when to use the object authorization IBM MQ script commands

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-2. Unit objectives (2 of 2)

## Implementing IBM MQ security



- Your configuration work must account for IBM MQ connection authentication, channel authentication, object authorizations, and SSL/TLS
- The 2035 return code can occur repeatedly, but for different reasons
- SSL/TLS presents other considerations

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-3. Implementing IBM MQ security

This unit combines the IBM MQ connectivity unit, and incorporates the IBM MQ security topics of connection, channel authentication, and authorization. The addition of security provides exposure to different challenges you might encounter when you work with a highly secured queue manager.

You start by looking at different variables that can result in security errors in the queue manager connections. You look at a baseline on security terminology to establish what security areas you work with in this unit.

You then learn about channel authentication and the different types of channel authentication rules. **For channel authentication, the words “rule” and “record” are used interchangeably.** You learn what parameters are critical to produce the correct results when you work with the different channel authentication rules. You also look at the default channel authentication settings in the queue manager.

You look at object authorizations, and the two ways to work with them, by using control commands, or by using MQSC commands. You look at how to work with object authorizations with a security restricted queue manager.

Next, you look at the basic parameters that are used for object authorization commands, and how to display and set object authorizations.

## Terminology baseline: Security areas

- **Identification:** Determine the identity of a person who is using a system or resource
- **Authentication:** Proving the person or resources that are being identified are who they claim to be
- **Access control/authorization\***: Limiting access to resources to only those resources that need it
- **Confidentiality:** Protection of sensitive information
- **Data integrity:** Ability to detect tampering with critical data
- **Security auditing:** Ability to determine whether any unauthorized access was done or attempted

**Note:** \*The terms *access control* and *authorization* might be used interchangeably.

Figure 5-4. Terminology baseline: Security areas

This slide looks at different security areas, and highlights the areas on which this unit focuses.

Authentication in IBM MQ relied on identification. Some might argue that if you used SSL/TLS channels, some type of authentication took place. Identification meant that a user or application that connected to IBM MQ, previously cleared security before connecting to IBM MQ. The term “identification” meant that the user was previously identified before connecting to IBM MQ.

Authentication is when you provide credentials and prove who you, or the application is.

Access control or authorization deals with what the user is allowed to do after being authenticated.

Object authorizations provide access control and authorization, by providing granularity of what types of actions can be taken against an object. Object authorizations in IBM MQ respond to questions such as “Can an application connect to a queue manager, open a queue, put, or get a message?”

Confidentiality refers to the ability to encrypt data so it cannot be read. IBM MQ provides the capability to encrypt data that is moving across queue manager by adding SSL/TLS to the channels. However, if data is at rest – that is, in queues – and needs to be encrypted, a product such as IBM MQ Advanced Security or another vendor product needs to be used.

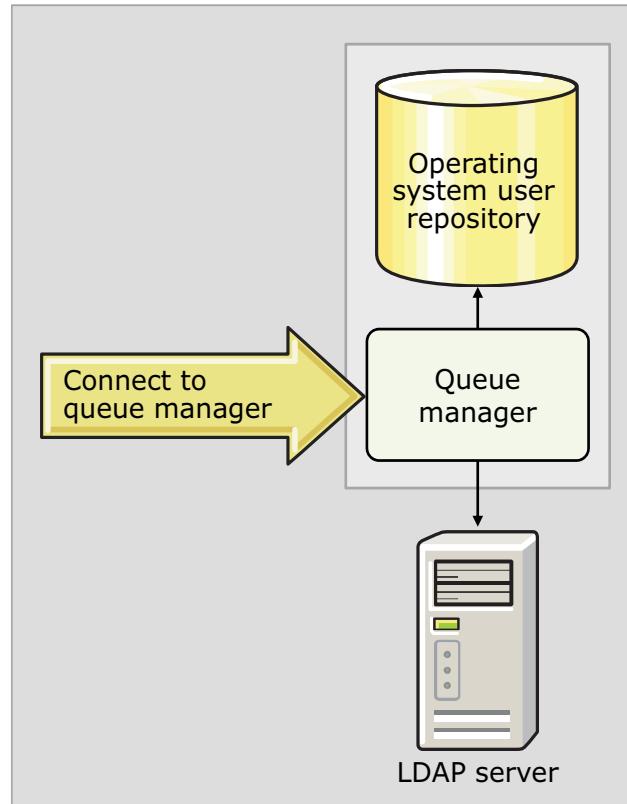
Integrity deals with the ability to detect whether anyone tampered with the data. Integrity is usually done by using algorithms with different checksums before and after receipt of the data, and comparing that the before and after values are equal to prove that no changes took place.

Auditing is having a recorded history of security events so that any unauthorized attempts can be tracked.

In this unit, you work with the authentication and authorization aspects of IBM MQ security.

## Connection authentication

- Queue manager attribute CONNAUTH
- Two new types of authentication objects:
  - OS repository
  - LDAP
- Defaults
  - New queue managers default to using OS system authentication
  - Migrated queue managers have CONNAUTH disabled
- Use of MQSAMP\_USER\_ID environment variable with sample programs sets ID and allows password entry
- On the application side:
  - Code change is required to add security structure
  - Option to use mqccred exit



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-5. Connection authentication

Connection authentication enables applications to provide a user ID and password that the queue manager can validate before an application is allowed to connect. As implemented, connection authentication can use either the system OS or LDAP to store credentials. IBM MQ connection authentication was introduced with IBM MQ V8.0. Both sides of a connection must be at IBM MQ V8.0 or later to use connection authentication.

A new queue manager initially uses type OS connection authentication. With a migrated queue manager, connection authentication is disabled.

After the type of authentication is selected or updated, different options can be set for local or and for client connections. You learn about these options in the next slide.

How can you test connectivity when connection authentication is enabled in a queue manager? After connection authentication is enabled and hardened, you use the same sample programs to test connectivity. When you use amqsput, amqsget, and other sample programs, you use variable MQSAMP\_USER\_ID and provide an ID. With the ID, the sample programs amqsput and amqsget prompt you for the password. Other sample programs also work with the MQSAMP\_USER\_ID variable.

To test a client connection, you use amqscnxc and include the `-u`, or user parameter, and then provide the password when prompted. The amqscnxc sample ends after a connection is attempted.

## CONNAUTH settings new queue manager initial default

```
1 : dis qmgr connauth
AMQ8408: Display Queue Manager details.

QMNAME (MQG1)

CONNAUTH (SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
```

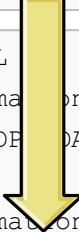
```
2 : dis authinfo (*) CHCKCLNT CHCKLOCL
AMQ8566: Display authentication information details.

AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
.....
AMQ8566: Display authentication information details.

AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
AUTHTYPE (IDPWOS) CHCKCLNT (REQDADM)
CHCKLOCL (OPTIONAL)
AMQ8566: Display authentication information details.

AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.OCSP)
.....
AMQ8566: Display authentication information details.

AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.CRLLDAP)
AUTHTYPE (CRLLDAP)
```



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-6. CONNAUTH settings new queue manager initial default

Connection authentication settings have two parts:

- First is the queue manager CONNAUTH parameter. This parameter identifies the authorization information, or AUTHINFO record that is active for the queue manager. In the display, you see the active AUTHINFO record name ends with “IDPWOS”, meaning “OS defined ID and password”. The slide displays the default CONNAUTH setting that you find in a new queue manager, SYSTEM.DEFAULT.AUTHINFO.IDPWOS.
- Second, the AUTHINFO record has two parameters: CHCKLOCL and CHCKCLNT. These parameters specify the connection authentication requirements for the queue manager. Although the initial settings of CHCKLOCL and CHCKCLNT behave identically, the initial settings might differ according to the operating system.

## CHCKCLNT and CHCKLOCL values (1 of 3)

- **NONE:**
  - Turns off checking
  - Causes failure if CHLAUTH rule with CHCKCLNT is set to REQUIRED
- **OPTIONAL:**  
If an ID is provided, the ID and password must be correct
- **REQUIRED:**  
All applications must provide valid credentials
- **REQADM (distributed)**
  - If it is an administrative ID, credentials are required
  - If it is not an administrative ID, it is treated as OPTIONAL

- **CHCKCLNT=OPTIONAL, amqsputc without user ID**

```
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
target queue is ORDERS.AT.MQ00
xyz
Sample AMQSPUTO end
```

- **CHCKCLNT=OPTIONAL, amqsputc with ID, but a bad password**

```
C:\>set MQSAMP_USER_ID=INGMUSR
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
Enter password: abc123
MQCONNX ended with reason code 2035
```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-7. CHCKCLNT and CHCKLOCL values (1 of 3)

In this slide, you look at the possible values that are used in the CHCKCLNT and CHCKLOCL parameters. Since the settings are similar, the examples are provided for CHCKCLNT.

- NONE, which switches off connection authentication. However, if you later combine connection authentication with a channel authentication rule where CHCKCLNT is set to REQUIRED, the CHLAUTH rule fails.
- OPTIONAL, which is the default for CHCKLOCL connections, means that if a user ID is provided, then a valid password is required. If no user ID is provided, no password is required.
- REQUIRED means that all applications must provide a user ID and password.
- REQADM, which is the default value for the CHCKCLNT parameter for distributed platforms, requires a user ID and password if it detects that the user is an administrative or privileged user.

The examples in this slide show the use of the OPTIONAL setting. These examples use the amqsputc client program, which works like amqsput but for IBM MQ client applications.

The example in the slide shows the behavior of CHCKCLNT=OPTIONAL. If you test the program without providing credentials, the test completes successfully. However, if you provide a user ID but do not provide a valid password, the connection fails with a 2035.

## CHCKCLNT and CHCKLOCL values (2 of 3)

- CHCKCLNT=REQUIRED, amqsputc without credentials

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) +
 CHCKLOCL(OPTIONAL) CHCKCLNT(REQUIRED)
REFRESH SECURITY TYPE(CONNAUTH)
```

```
C:\>set MQSAMP_USER_ID=
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUT0 start
MQCONNX ended with reason code 2035
```

```
+CSQX511I MQ00 CSQXRESP Channel MQ00.CL started 232
connection 10.4.127.184
CSQH045E MQ00 CSQHNSIG MQ00.CL/10.4.127.184 did not provide a password
+CSQX512I MQ00 CSQXRESP Channel MQ00.CL no longer active
```

```
runmqsc MQG1
...
Starting MQSC for queue
manager MQG1.
AMQ8135: Not authorized.
```

```
runmqsc -u administrator MQG1
Enter password:

Starting MQSC for queue
manager MQG1.
```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-8. CHCKCLNT and CHCKLOCL values (2 of 3)

In this second example, CHCKCLNT is set to REQUIRED, but now the `MQSAMP_USER_ID` variable is set to blanks, so it is no longer a valid user ID. As you might note by the presence of the “CSQ-prefixed” messages, the first and third boxes in this slide represent a z/OS system.

In this example, you connect from an IBM MQ client in a Windows server to a z/OS queue manager. When you try to connect to the z/OS queue manager, which has CHCKCLNT(REQUIRED) without a user ID, the connection fails and you see the 2035. In the third box, you see the error as it appears in the z/OS log.

The two boxes across the bottom show how the `runmqsc` utility is treated like another application and also fails without credentials. After connection authentication is hardened, you use `runmqsc` with the `-u` parameter to provide a user ID. Then, you are prompted for the password.

## CHCKCLNT and CHCKLOCL values (3 of 3)

- **CHCKCLNT=REQUIRED** or **REQADM amqsputc** with valid credentials

```
C:\>set MQSAMP_USER_ID=INGMUSR
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
Enter password: VALIDPWD
target queue is ORDERS.AT.MQ00
Check message that the application now works
Sample AMQSPUTO end
```

- **CHCKCLNT=REQUIRED** or **REQADM amqscnxc** with **-u** parameter

```
amqscnxc -x 127.0.0.1(1657) -c WM253.G1.SVRCONN -u administrator MQG1
Sample AMQSCNXC start
Connecting to queue manager MQG1
using the server connection channel WM253.G1.SVRCONN
on connection name 127.0.0.1(1657).
Enter password: websphere
Connection established to queue manager MQG1

Sample AMQSCNXC end
```

Figure 5-9. CHCKCLNT and CHCKLOCL values (3 of 3)

In this last slide, you have tests with the amqsputc, and the amqscnxc. The amqsputc sample, which is the client version of amqsput, works with the `MQSAMP_USER_ID` variable. In the example, the ID is `INGMUSR` set in the variable. You are then prompted for the password.

The amqscnxc sample does not use the `MQSAMP_USER_ID` variable. It takes a `-u` parameter to provide the user ID, and then you are prompted for the password.

The `amqscnxc` command builds the client channel in its code, so it is not necessary to create a client connection from the client side. Provided a server connection channel is in the queue manager side, such as `WM253.G1.SVRCONN`, the `amqscnxc` code takes care the client side of the connection.

## Channel authentication rules

### Channel authentication rules (or records):



A security mechanism that enables setting rules to determine which connections are allowed and which are excluded from accessing the queue manager

- Rules can be set to allow or inhibit connections, privileged users, or a combination of attributes
  - Rules are also referred to as “records”
- Rules can be set to use different details of the incoming connection, such as an IP address or a remote queue manager name
- IBM MQ Managed File Transfer and channel authentication records
  - IBM MQ Managed File Transfer configuration uses IBM MQ infrastructure
  - Create channel authentication records as needed to communicate across the configuration queue manager
  - No separate IBM MQ Managed File Transfer configuration tasks

IBM MQ security and clusters

© Copyright IBM Corporation 2017

*Figure 5-10. Channel authentication rules*

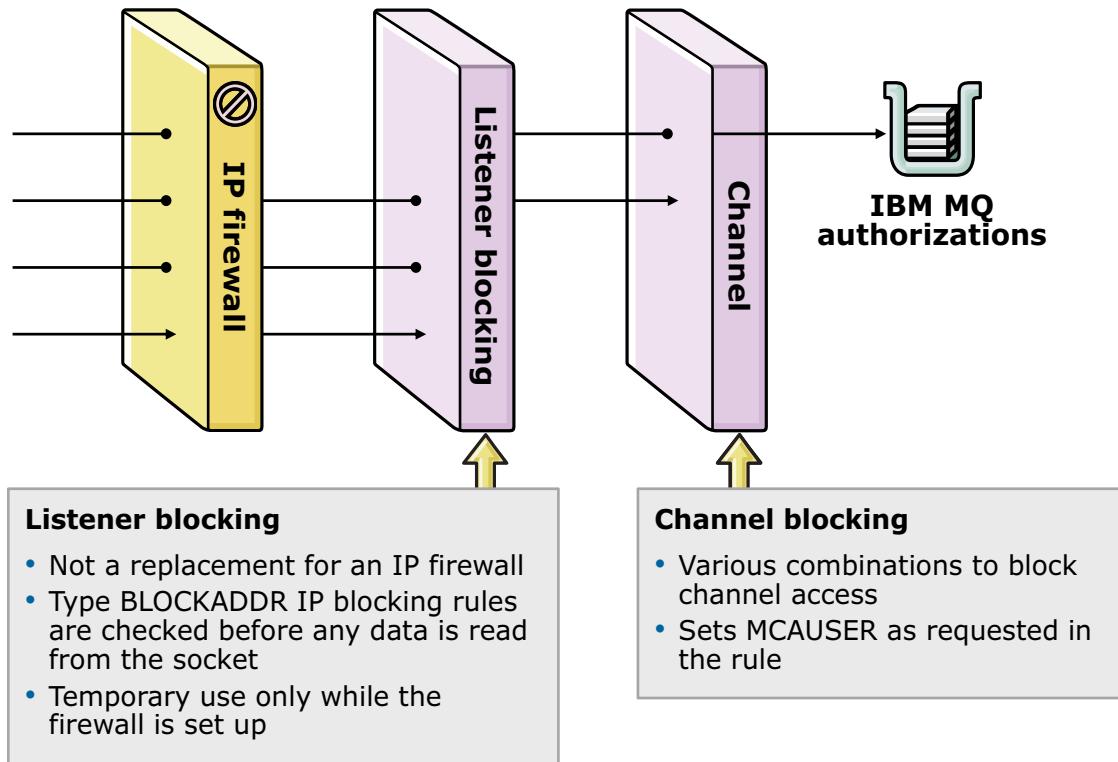
A channel authentication rule is a record set in the queue manager that controls the behavior of one or more channels. You might hear the terms channel authentication rules and channel authentication records used interchangeably. In the queue manager, the use of channel authentication rules is enabled or disabled by using the CHLAUTH queue manager parameter.

Many IBM MQ administrators disable channel authentication rather than being required to set rules. But like with most technologies after you get used to setting the rules, it becomes simpler. After you complete this unit and lab, you are ready to leave CHLAUTH enabled in the queue manager, and set rules as required.

Channel authentication rules have several types.



## Channel blocking points



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-11. Channel blocking points

Before you proceed to look at the different types of channel authentication rules, you must understand at what point in the connection some of these rules are invoked compared to a firewall and object authorizations.

In a typical enterprise configuration, the first block a connection might face is a firewall. A firewall block does not return an IBM MQ code. After the connection is past the firewall, it has one type of channel rule that is called BLOCKADDR, which can be set only to an IP address. This rule is because a BLOCKADDR type rule is invoked before the channel names are known. A BLOCKADDR rule can be used, temporarily, until firewall rules are in place. However, administrators are not to replace firewalls with BLOCKADDR rules.

After the connection passes through the firewall, and a possible BLOCKADDR type rule, the rest of the CHLAUTH rules might be invoked depending on the details of the connection and the name of the channel.

After the channel authorization rule is cleared, the next layer of protection consists of object authorizations. That is, an entity is entitled to open a certain queue, and put messages, or get messages from that queue.

In this slide, you see mention of MCAUSER. Most IBM MQ administrators are aware of the significance of the MCAUSER parameter of a channel. It is important to understand the criticality of this field.

## MCAUSER basics for client channels

The authorizations of the user ID in the MCAUSER parameter of a channel definition must be carefully reviewed to prevent jeopardizing the queue manager

- For a client SVRCONN channel:
  - A user ID set by a security exit might be blocked by a type BLOCKUSER channel authentication rule
  - A user ID set by a channel authentication rule might be overridden by a security exit
  - A user ID set in the channel MCAUSER attribute is used, unless over-ridden by a channel authentication rule or a security exit
  - If no user ID is set through any other means, a user ID is “flowed” from the client server
  - If no user ID “flowed” from the client, and no user ID was set through any other means, the user ID under which the SVRCONN started is used
  - For message channels, see the IBM MQ Knowledge Center

For detailed notes on MCAUSER, see the slide that is titled “Search string to additional IBM MQ security reference” near the end of this unit

*Figure 5-12. MCAUSER basics for client channels*

When you look at channel authentication, you find numerous references to the MCAUSER parameter of the channel definition. This slide presents basic information on how the MCAUSER field is populated in a client, or SVRCONN channel as an introduction to the channel authentication topic. You can find similar information applicable to message channels in the IBM Knowledge Center. The key objective is to highlight the significance of the MCAUSER parameter.

The contents of the MCAUSER parameter of a channel definition, and the related channel object authorizations, are among the most critical security aspects of IBM MQ. Careless setting of this field and related channel object authorizations can lead to significant security breaches, such as intruders who try to access the IBM MQ command queue, or unauthorized access to data.

In the rest of this unit, do not confuse channel authentication with IBM MQ channel object authorizations. Channel authentication is enabled at the queue manager level, and administered at the channel level by using SET CHLAUTH rules. Object authorizations provide specific users or groups authorization to different actions. Channel authentication and object authorizations are separate, but related IBM MQ security mechanisms.

You are urged to research IBM Knowledge Center, and to use a browser with the search string suggested at the end of this unit. A search string is provided instead of a link to mitigate the case when a link might become obsolete. The articles that are found by doing the suggested search

provide a detailed explanation of the importance of this IBM MQ channel parameter, and provides guidance on how to set IBM MQ object authorization on the channels.

You now look at how the MCAUSER parameter might be set for client, or SVRCONN channels. The MCAUSER parameter might, or might not have an originally set user ID. The process to determine the value of the MCAUSER parameter is:

- The first consideration might be the case where a user has a customized security exit. Channels have exit points where users can add a customized security exit. A TYPE(BLOCKUSER) channel authentication rule can override the MCAUSER ID that a security exit supplies. In this case, the type of channel authentication rule is specific to a TYPE(BLOCKUSER) rule, or else the next bullet might sound contradictory.
- The next ID to be used is the ID set by a channel authentication rule, unless a security exit blocks the channel authentication rule ID. Therefore, a security exit can override the actions that a channel authentication rule takes, unless that channel authentication rule is a TYPE(BLOCKUSER) rule.
- A user ID coded in the MCAUSER field of a channel is used, unless security exit or a channel authentication rule overrides this ID.
- If no user ID is set by using any other means, the user ID in use is the user ID “flowed” from the client machine.
- If no user ID is “flowed” from the client machine, then the user ID that started the server connection channel is used.

This slide provided an example of client channels to provide. You can find information on MCAUSER for message channels in section Security for remote messaging in the IBM Knowledge Center.

The processing of the MCAUSER ID can also vary across different operating systems. When a channel authentication rule blocks a user, you see the rejected ID in the entry that is found in the queue manager log for the queue manager that blocked the connection.

Channel authentication rules screen use of channels by asserting different criteria. If used carefully, channel authentication might mitigate certain exposures. If you are not familiar with the MCAUSER ID, you are encouraged to use the references that are provided at the end of this unit, and IBM Knowledge Center. A good starting point in the IBM Knowledge Center is the section titled Planning for your security requirements.

IBM MQ security is a complex topic. Organizations are expected to review their specific security requirements, then establish standards. This policy must outline how to use the different IBM MQ security capabilities, such as SSL/TLS, authentication, and object authorizations, according to organization-specific scenarios.

## CHLAUTH rule types

| CHLAUTH type | Description                                                                                                                    | Required attribute |
|--------------|--------------------------------------------------------------------------------------------------------------------------------|--------------------|
| BLOCKUSER    | Prevents specified users from connecting                                                                                       | USERLIST           |
| BLOCKADDR    | <ul style="list-style-type: none"> <li>Prevents connections from specified IPs</li> <li>Works at the listener level</li> </ul> | ADDRLIST           |
| SSLPEERMAP   | Maps SSL/TLS distinguished names (DNs) to MCAUSER values                                                                       | SSLPEER            |
| ADDRESSMAP   | Maps IP addresses to MCAUSER values                                                                                            | ADDRESS            |
| USERMAP      | Maps user IDs to MCAUSER values                                                                                                | CLNTUSER           |
| QMGRMAP      | Maps remote queue manager names to MCAUSER values                                                                              | QMNAME             |

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-13. CHLAUTH rule types

Channel authentication rules have different types. It is important to understand which rule to use.

- A BLOCKUSER type rule prevents one or more users from connecting to a specific channel or channel profile. The “channel profile” term is used as certain wildcards can be used with the channel name, which can apply to a group of channels. Ironically, you use a BLOCKUSR type rule, but with a non-existent user name, to allow an administrative user to connect. You see this use in a subsequent slide.
- The BLOCKADDR type rule can be used, on a temporary basis, to block connections from a specific IP address or addresses. Because it runs before the channel name is known, it requires the use of an IP, or partial IP address.
- An SSLPEERMAP rule type can map TLS distinguished names to MCAUSER values. This type of rule requires an SSLPEER.
- An ADDRESSMAP type authentication record maps IP addresses to MCAUSER values. This type of rule requires an ADDRESSMAP.
- A USERMAP type rule maps known user IDs to MCAUSER values. This rule requires a CLNTUSER.

- Last is the QMGRMAP rule. This type of rule maps a queue manager name to an MCAUSER ID. This rule requires a QMNAME parameter. The QMGRMAP rule can come in handy when you work with IBM MQ clusters.

## Which rule to use: ADDRESSMAP or BLOCKADDR

- When you use TYPE(ADDRESSMAP), the data already flowed
  - You know the name of the channel
  - Full details of the reason the connection was blocked are available in the log, and in the system events if configured
  - ADDRESSMAP is the type of rule that should be used in most cases
- TYPE(BLOCKADDR) applies much earlier in the process before any data flows
  - The channel name is not yet known
  - Use of BLOCKADDR is to block problem addresses that later get handled with a firewall
  - BLOCKADDR rules should be used on a temporary basis, and should not permanently substitute a firewall
  - Channel name attribute must be the generic name, a single asterisk ( \* )

*Figure 5-14. Which rule to use: ADDRESSMAP or BLOCKADDR*

A frequent question when you work with channel authentication rules is whether to use an ADDRESSMAP or a BLOCKADDR to keep out certain connections.

When an ADDRESSMAP rule executes, data such as the channel name flows, and you can use channel names. An ADDRESSMAP rule provides full details with the reason a connection was blocked in the queue manager log. An ADDRESSMAP is used most of the time.

A BLOCKADDR rule executes before the channel name is known. The BLOCKADDR rule requires a generic channel name. As noted earlier, use the BLOCKADDR rule on a temporary basis, and not as a replacement for a firewall.

## CHLAUTH rules using IP addresses and host names

| Rules                                                                | Using IP address                                                                                               | Using host names                                                                                                                   |
|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Listener blocking list                                               | SET CHLAUTH('*) +<br>TYPE(BLOCKADDR) +<br>ADDRLIST('10.5.*','192.168.2.3') +<br>USERSRC(NOACCESS)              | <b>Not allowed</b>                                                                                                                 |
| Channel-based blocking                                               | SET CHLAUTH('MQ00.CL.*') +<br>TYPE(ADDRESSMAP) +<br>ADDRESS('10.168.*.3') +<br>USERSRC(NOACCESS)               | SET CHLAUTH('MQ00.CL.*') +<br>TYPE(ADDRESSMAP) +<br>ADDRESS('*. <a href="#">ibm.com</a> ') +<br>USERSRC(NOACCESS)                  |
| Channel allowed                                                      | SET CHLAUTH('*. <a href="#">SVRCONN</a> ') +<br>TYPE(ADDRESSMAP) +<br>ADDRESS('10.5.6.*') +<br>MCAUSER(INGUSR) | SET CHLAUTH('*. <a href="#">SVRCONN</a> ') +<br>TYPE(ADDRESSMAP) +<br>ADDRESS('ac. <a href="#">ibm.com</a> ') +<br>MCAUSER(INGUSR) |
| Qualification of another rule type with an IP address or a host name | SET CHLAUTH('*) +<br>TYPE(USERMAP) +<br>CLNTUSER('NOMAD123') +<br>ADDRESS('10.4.*') +<br>MCAUSER('INGUSR')     | SET CHLAUTH('*) +<br>TYPE(USERMAP) +<br>CLNTUSER('NOMAD123') +<br>ADDRESS('a*. <a href="#">ibm.*') +<br/>MCAUSER('INGUSR')</a>     |

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-15. CHLAUTH rules using IP addresses and host names

When channel authentication rules were first made available, it was not possible to use a host name; all the rules required IP addresses. With IBM MQ V8, the ability to use host names along with IP addresses was incorporated to channel authentication rules.

This slide looks at the different rules, and examples of how to specify IP addresses and host names.

As shown on this slide, except for the BLOCKADDR type rules, which use IP addresses exclusively, you can use either IP addresses or host names in all other rules. You see a comparison of the rules in the second and third columns in this slide.

A critical part of the syntax for the rules is the use of single quotation marks around your value specifications. If you do not enclose values within single quotation marks, you do not get a syntax error; however, the rule does not work, and significant time might go into realizing the problem. Single quotation marks are critical to special characters and wildcards, and also the case sensitivity of a value.

Use of single quotation marks in all IPs, hosts, user IDs, and lists, is a critical detail to remember when you create channel authentication rules.

## CHLAUTH actions

- ADD
  - For BLOCKUSER and BLOCKADDR, configuration added to the list
  - If a rule exists for other rule types, command fails
  - ADD is default
- REPLACE
  - Might result in an ADD for SSLPEERMAP, ADDRESSMAP, USERMAP, and QMGRMAP rule types
  - For BLOCKUSER and BLOCKADDR, REPLACE with an empty list works like a REMOVEALL
- REMOVE
  - Deletes a single specified record
  - If the last entry from a list is removed, it works like a REMOVEALL
- REMOVEALL
  - For BLOCKUSER and BLOCKADDR, removes all list members
  - For SSLPEERMAP, ADDRESSMAP, USERMAP, and QMGRMAP it removes all mappings from the channel authorization records

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-16. CHLAUTH actions

Channel authentication rules have an ACTION parameter that specifies what to do with the rule. The default ACTION parameter is `ADD`. Previous examples of rules were expected to be added, so no ACTION parameter was included because `ADD` is the default.

If you create a rule with ACTION type for rule types that require a list of parameters, such as `BLOCKUSER` and `BLOCKADDR`, the extra user or the address is added to the existing rule. However, if you try to add a rule for any other rule types with a pre-existing rule, the rule command fails.

Special care must be taken with `ACTION(REPLACE)` to ensure that it does not result in an `ADD`. When you start working with channel authentication, a good practice to follow is to count the number of rules in the queue manager before you run a rule with `ACTION(REPLACE)`. If an extra rule is there after the `ACTION(REPLACE)` rule succeeds, it means that the change failed, and a new rule was added. A `SET CHLAUTH` command with `ACTION(REPLACE)` must include all other required parameters, depending on the type of rule to ensure that the rule that you intend to change is indeed changed.

The next slide shows some of the parameters that are required, depending on the rule type, to mitigate the possibility that a `REPLACE` does not turn into an `ADD`.

`ACTION(REMOVE)` deletes a specific channel authentication rule. `ACTION(REMOVEALL)` can be used as noted in the slide and works depending on the type of rule. However, `ACTION(REMOVEALL)` must be used with caution to ensure that it does not delete any unintended rules.

Remember to always count the number of rules before you run a `SET CHLAUTH` with `ACTION(REPLACE)`.

## Replacing and removing CHLAUTH rules

| Key fields                                                                                                            |      |                 |                      |              |                    |                      |
|-----------------------------------------------------------------------------------------------------------------------|------|-----------------|----------------------|--------------|--------------------|----------------------|
| Channel name                                                                                                          | Type | Values for type | Address (restrictor) | Source of ID | Mapped ID          | User ID and password |
| SET CHLAUTH('AP1.*')<br>TYPE(SSLPEERMAPP)<br>SSLPEER('CN=OrdApp')<br>ADDRESS('1.2.3.4')                               |      |                 |                      |              | USERSRC(MAP)       | MCAUSER('Ordmgr')    |
| SET CHLAUTH(ADMIN.SVRCONN)<br>TYPE(ADDRESSMAP)<br>ADDRESS('*.*.ibm.com')                                              |      |                 |                      |              | USERSRC(CHANNEL)   | CHKCLNT(REQUIRED)    |
| SET CHLAUTH('AP1.*') TYPE(SSLPEERMAPP) SSLPEER('CN=OrdApp') ADDRESS('1.2.3.4')<br>ACTION(REMOVE)                      |      |                 |                      |              |                    |                      |
| SET CHLAUTH(ADMIN.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*.*.ibm.com')<br>USERSRC(MAP) MCAUSER('IBMUSER') ACTION(REPLACE) |      |                 |                      |              |                    |                      |
| SET CHLAUTH('AP1.*') TYPE(SSLPEERMAPP)<br>ACTION(REMOVEALL)                                                           |      |                 |                      |              | All the key fields |                      |

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-17. Replacing and removing CHLAUTH rules

It might not be obvious, but when you start your work you might confirm that this slide is critical. Your work with channel authentication rules is going to involve a learning curve. Setting a rule to add channel authentication is simple. However, changing and removing existing rules can be challenging. You must include the correct number of parameters in your changes or removals, or an extra rule might be added. For example:

- You might get a “AMQ8885: Parameter not allowed for this action on a channel authentication record” return if you use too many parameters.
- You might get a “AMQ8884: Channel authentication record not found” if you do not use the required number of parameters for a change or removal.
- When you remove a channel authentication record, you expect confirmation that the record was removed. However, instead, the return is “AMQ8877: IBM MQ channel authentication record set”. You confirm that it is removed by displaying the authentication records.

With extra practice and determination, you learn to get past these initial challenges.

This slide shows the parameters that are required, depending on the type of rule, to qualify a specific channel authentication record to be replaced. In general, the more parameters you use to differentiate the specific channel authentication record, the less chance that an ACTION ADD is attempted.

The first example is for an SSLPEERMAP channel for any channel with a name that starts with a first name node of AP1. Notice the use of the single quotation marks. In this command, the exact SSPLEER value and exact address of the original CHLAUTH record are included in the SET CHLAUTH with ACTION(REPLACE) command.

The second example is for a type ADDRESSMAP record. Notice the original specification of the host name, including single quotation marks, '`*.ibm.com`'. In the same record, look at the CHCKCLNT value specified. Do you remember the connection authentication slide that provided the various values for CHCKCLNT and CHCKLOCL? In this channel authentication record, the CHCKCLNT value is REQUIRED. If connection authentication in the queue manager is set to CHCKCLNT value of NONE, this particular channel authentication rule would fail when it is initially invoked. In this case, connection authentication parameters need to agree with channel authentication parameters.

The last example is for an ACTION(REMOVEALL). In this case, all channel authentication records with channel name of "AP1" in the first node of the channel name are deleted, regardless of any other values. This command might remove more rules than you expect. Care must be taken when you use ACTION(REMOVEALL).

Always check the IBM Knowledge Center for key fields that differentiate among specific channel authentication rules. The specifications are also related to the IP version as noted.

Display the channel authentication rules before and after any additions, changes, or removals, to ensure that your results are correct.

## Generic IP addresses

- Single IPv4 address such as 10.15.131.7
- IPv4 address with wildcard, such as:
  - 10.15.131.\*
  - 10.15.\*
  - 10.15.\*.7
  - 10.\*.131
- Single or wildcard IPv6 addresses
- Use of a hyphen to indicate a range in IPv4 or IPv6, such as 10.15.131.1-7
- Combination of hyphen and an asterisk in IPv4 or IPv6 addresses, such as 10.15.1.\*.1-7
- Must be within dot separators
- No asterisk must precede a trailing asterisk
  - 10.\*.131.\* is valid
  - 10.5.\*.\* is invalid

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-18. Generic IP addresses

This slide looks at some of the IP address forms that can be specified.

For example, you can specify an asterisk as a wildcard when you specify an IP address. However, the asterisk must be specified within the dot separators, which means that if you want to specify 10.10.131.7 through 10.19.131.7, you cannot use 10.1\*.131.7.

You can use a mix of an asterisk and a hyphen as shown. However, an asterisk cannot precede a trailing asterisk. In the slide, 10.\*.131.\* is valid, but 10.5.\*.\* is invalid.

You can find more examples of how to use generic IP addresses in the **Generic IP addresses** link that is included in the `SET CHLAUTH` command documentation.

## CHLAUTH rule precedence

| Channel name           | CHLAUTH('MQ00.*') TYPE(SSLPEERMAP)<br>SSLPEER('O="IBM US")<br>MCAUSER(INGMUSR)    |
|------------------------|-----------------------------------------------------------------------------------|
| SSL distinguished name | CHLAUTH('MQ00.*') TYPE(USERMAP)<br>CLNTUSER('NOMAD123')<br>MCAUSER(TSM0000)       |
| Client user ID         | CHLAUTH('MQ00.*') TYPE(ADDRESSMAP)<br>ADDRESS('10.6.187.184')<br>MCAUSER(TSM0000) |
| Queue manager name     | CHLAUTH('MQ00.*') TYPE(ADDRESSMAP)<br>ADDRESS('*.*.ibm.com')<br>MCAUSER(IBMUSER)  |
| IP address             |                                                                                   |
| Host name              |                                                                                   |

- Channel authorization record selection criteria
  - Channel authorization with a full channel name takes precedence over a name that uses a wildcard
  - SSL DN or TLS DN takes precedence over a user ID, a queue manager name, or an IP address
  - Client ID or queue manager name takes precedence over an IP address
- Within each category, the most specific entry takes precedence
- Precedence also within SSL rules

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-19. CHLAUTH rule precedence

Within channel authentication rules, a certain precedence order exists. Two factors determine precedence: the type of rule and the specificity of the parameters that are specified. The following list contains examples of precedence:

- A channel authentication record with a full channel name takes precedence over a channel authentication record that uses wildcard characters in its name.
- SSL or TLS distinguished name records take precedence over user ID, queue manager name, or IP address type records.
- Client and queue manager name type records take precedence over an IP address type record. Records that use host names have the least precedence.

Within each category above, the most specific type takes rule precedence, so records with full channel names take precedence within a record type category.

## CHLAUTH initial settings

```
dis qmgr chlauth
 1 : dis qmgr chlauth
AMQ8408: Display Queue Manager details.
 QMNAME (MQG1) CHLAUTH (ENABLED)
```

```
dis chlauth(*) descr
 2 : dis chlauth(*) descr
AMQ8878: Display channel authentication record details.
 CHLAUTH(SYSTEM.ADMIN.SVRCCONN) TYPE (ADDRESSMAP)
 DESCR(Default rule to allow MQ Explorer access)
 ADDRESS(*) USERSRC (CHANNEL)
AMQ8878: Display channel authentication record details.
 CHLAUTH(SYSTEM.*) TYPE (ADDRESSMAP)
 DESCR(Default rule to disable all SYSTEM channels)
 ADDRESS(*) USERSRC (NOACCESS)
AMQ8878: Display channel authentication record details.
 CHLAUTH(*) TYPE (BLOCKUSER)
 DESCR(Default rule to disallow privileged users)
 USERLIST (*MQADMIN)
```

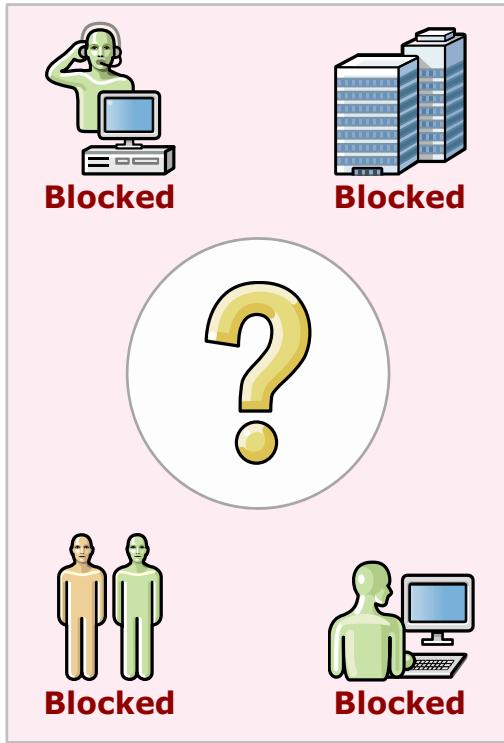
*Figure 5-20. CHLAUTH initial settings*

When you work with a new V7.5 or later queue manager, channel authentication is enabled. You can confirm that it is enabled by displaying the queue manager attribute CHLAUTH, and you see a value of ENABLED. If you display all channel authentication records for a new queue manager, you find three records. You look at the second and third records first.

- The second CHLAUTH record is of type ADDRESSMAP, named 'SYSTEM.\*'. This record protects the queue manager by blocking access to all SYSTEM channels.
- The third CHLAUTH record is of type BLOCKUSER, and uses the all-inclusive generic name of a single asterisk. This rule is intended to block all administrative users. The user ID in your lab exercises is an administrative user. In a later slide, you learn to add a channel authentication rule to allow use of an administrative user, without removing the original default rule.
- The first rule is another ADDRESSMAP rule, but has the more specific name of SYSTEM.ADMIN.SVRCCONN, with an initial intent of allowing access to the client channel that is used for IBM MQ Explorer. Since channel SYSTEM.ADMIN.SVRCCONN is known to hackers and might be subject to an attack, some IBM MQ administrators remove this channel. However, further discussion is out of the scope of this unit. If you refer to the search in the last page of this unit, you learn extra details about the use of this channel.

You work with channel authentication rules in the lab exercise for this unit.

## The back-stop rule and warning mode



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-21. The back-stop rule and warning mode

- Preferred way to implement CHLAUTH is to inhibit access to all users
  - Then, configure rules to allow known users
- Leave the initial CHLAUTH rules in place
- Called the “back-stop” rule:  
`SET CHLAUTH('*') +  
TYPE(ADDRESSMAP) ADDRESS('*') +  
USERSRC(NOACCESS) +  
DESCR('Back-stop rule')`
- WARN attribute
  - Allows connections to continue
  - Captures would-be blocked connection information
  - Default behavior is **no** warning (block)

The suggested method to implement channel authentication is by defining the back-stop rule. The back-stop rule is an ADDRESSMAP type rule, which blocks all connections to the queue manager. After all connections are blocked, you create new channel authentication rules to unblock connections that are entitled to connect to the queue manager. When you set the back-stop rule, you leave the three original rules in place.

When you implement the back-stop rule, any users who previously were accessing the queue manager are no longer able to connect.

One option that can be used with a channel authentication rule is the `WARN` parameter. The default setting of the `WARN` parameter is `NO`, which means that the connection is blocked without warning. If you set the back-stop rule `WARN` parameter to `YES`, you can find out who tried to connect to the queue manager, but the connection is not blocked. One way to find the information is to enable events and check for channel blocked events. In this course, you implement the back-stop rule, but leave the `WARN` parameter set to `NO` to experience the security challenges caused by channel authentication rules.

## Allow access to administrative users

- Problem: Channel TSM0021.XPL is defined with MCAUSER('TSM0021')
  - The channel initiator for the queue manager is running under ID TSM0021
  - After CHLAUTH is enabled, channel TSM0021.XPL is blocked with error message:  
**CSQX776E MQ21 CSQXRESP Channel TSM0021.XPL from 10.61.23.250 has been blocked due to userid, Detail: MCAUSER(TSM0021)CLNTUSER(Administrator)**
- Solution:
  - Do not remove the initial CHLAUTH rule to disallow privileged users
  - Create a BLOCKUSER rule for the channel
  - The name of the channel should be specific
  - Use a bogus user name in the USERLIST parameter  
**SET CHLAUTH (TSM0021.XPL) TYPE (BLOCKUSER) +  
USERLIST ('GTMETHRU')**

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-22. Allow access to administrative users

In earlier slides, you learned that channel authentication rules had a precedence, and rules that used a more specific channel name had precedence over rules with less specificity. When you looked at the three original channel authentication rules, the third BLOCKUSER type rule that was displayed showed that it blocked administrative users from all channels, with the channel name specified as a least specific single '\*'.

You might notice that the example in this slide was taken from a z/OS system by the CSQ message prefix. The message that you obtain in the queue manager log for a distributed platform is similar to the message in this display, with a different message prefix.

In common for all channel authentication-related rejections is the term “blocked”. When you see that a channel was blocked, look for a channel authentication problem.

To bypass the original rule without removing it, you create a more specific BLOCKUSER type rule. In this example, the name of the blocked channel is a specific name TSM0021.SVRCONN, without any wildcard characters. The way to allow an administrative user such as fteamadmin is to block a bogus user that does not need to exist. In this example, the rule is blocking non-existent user GTMETHRU. As the specific TSM0021.SVRCONN BLOCKUSER rule blocks the GTMETHRU user, it allows other administrative users to connect. It is a rather indirect manner to allow an administrative user to connect. This rule takes precedence over the default BLOCKUSER rule because the name of the channel is the specific full name, TSM0021.SVRCONN, without asterisks.

## CHLAUTH best practices (1 of 2)

- Implement the back-stop rule
  - If you need to minimize impact, use warning mode
- Monitor the outcome of the back-stop rule
  - Check the system log for authorization rejections with WARN(NO)
  - Enable channel events and review event messages
  - Blocked connection messages are **not** generated on the system log for rules with WARN(YES)
  - Must enable channel events to view rejected connections
- Add one CHLAUTH rule at a time and test it
- Count the number of rules before and after you set a rule with ACTION(REPLACE)

Figure 5-23. CHLAUTH best practices (1 of 2)

When you start to implement channel authentication rules, observing a few practices adds control and stability to your work. The following list contains some guidelines.

- Leave channel authentication enabled in the queue manager and leave the three original rules in place.
- If you work with multiple environments, such as development, test, and production, start your channel authentication work in the development environment. Then, continue it through test and production, even if you need to adjust a portion of the channel name in each environment.
- It is easier to leave the `WARN` parameter to its default of NO. If you set `WARN` to YES, channels are not blocked, but you might not get error messages in the logs. When you set `WARN` mode to YES, you need to enable channel events, and review the event messages to determine what channels would be blocked. This process involves extra work, but if you cannot block users, it might have to be used.
- Do not add multiple rules at one time. Add one rule at a time and test it.
- If you must add a rule with ACTION(REPLACE), always count the number of rules in the queue manager before and after the rule with ACTION(REPLACE) is run, to ensure that no extra rules were added. Make sure that you understand what parameters to include according to the rule type, to mitigate the addition of a record instead of a change.

- IBM MQ objects have a description, or DESC field, which most users ignore. Channel authentication records are an exception. You might want to make a practice of including information in the description field. This field contains the reason that the particular record was added, or contact information for a resource on the side of the channel that might get blocked.
- Since channel authentication rules are a relatively recent addition to IBM MQ, you might want to ensure that these rules are being backed up. You should not only back up your rules after you create and test them, but ensure that operational scripts to back up queue manager objects include channel authentication rules.

## CHLAUTH best practices (2 of 2)

- Ensure that you understand the attributes that are required with each type of CHLAUTH rule
  - Use correct attributes for all ACTION rules
- Use the `DESCR` attribute of the `CHLAUTH` definition to identify why the rule is set
- After you implement and test your rules, run a backup of the CHLAUTH objects
- Confirm that your organization scheduled definition backup includes CHLAUTH rules

## Security considerations for IBM MQ clusters

- Which queue managers can send messages to your queue manager
- Which users of a remote queue manager can send messages to a queue on your queue manager
- Which queue managers can join a cluster
- Which process to use to force an unwanted queue manager out of a cluster
- SSL/TLS implementation considerations

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-25. Security considerations for IBM MQ clusters

Throughout this course, you learned about the similarities between working with IBM MQ clusters and distributed message channels. To secure a cluster, you use the same IBM MQ security that you use for distributed message channels, with the caveat of extra considerations.

One of the considerations for application availability is to ensure that all queue managers that host a queue and corresponding application in a cluster allow required authorizations to the users of that cluster. However, what users are allowed, and what users are not allowed?

If you secure one queue manager in the cluster, other cluster queue managers cannot communicate with the secured queue manager, so you must apply the security configuration adequately to all members of the cluster.

If your cluster is a basic cluster, management is simpler. However, if your organization allows overlapping clusters, the complexity in security increases. For example, do users of an added cluster have the same rights as users of the original cluster? What standards are in place when a “problem queue manager” is identified? A problem queue manager might be a queue manager that is used by principals with lower security privileges, or a queue manager that sends messages that are too large for the cluster to handle.

SSL/TLS presents a special challenge to clusters. For example, all queue managers that connect to a specific CLUSRCVR must have access to the cipher specifications that the target queue

managers use. Always refer to the IBM Knowledge Center to ensure that you are aware of all the SSL considerations for IBM MQ clusters.

## Troubleshooting queue manager default settings (1 of 5)

- Selected scenario: IBM MQ default client settings
  - Default channel authentication settings  
CHCKLOCL (OPTIONAL) CHCKCLNT (REQADM)
  - Channel authorization rule  
CHLAUTH (\*) TYPE (BLOCKUSER)  
DESCR (Default rule to disallow privileged users)  
USERLIST (\*MQADMIN)
- User ID: Administrator, a user considered a “privileged user”
- Connection test utility amqscnxc  
amqscnxc -x localhost(1657)  
-c MQG1.MIXAPP.SVRCONN MQG1
- Results: **Repeated 2035 errors**
- Recheck the queue manager logs for each new 2035

Figure 5-26. Troubleshooting queue manager default settings (1 of 5)

In this set of five slides, you walk through the challenges that are faced when you test a client connection by using the amqscnxc sample program. You use it in a new queue manager with default connection authentication and channel authentication settings.

In this scenario, the connection authentication settings use the original AUTHINFO record with the CHCKCLNT setting to REQADM, which means required for an administrative user. You are looking at a client connection throughout the five slides, so for now you can ignore the CHCKLOCL value.

The original channel authentication rule to block administrative users from all channels is also active in the queue manager.

The user ID that is running the test in these five slides is the fteadmin, an administrative user.

You test by typing the required fields for the amqscnxc sample, host name and port, channel name, and queue manager name.

This result is the first of several 2035 return codes. In the next two slides, you see the entries in the MQG1 queue manager log.

## Troubleshooting queue manager default settings (2 of 5)

1/27/2017 06:25:18 - Process(3568.22) User(MUSR\_MQADMIN)  
 Program(amqzlaa0.exe) Host(WS2008R2X64) Installation(IBMMQV9)  
 VRMF(9.0.0.0) QMgr(MQG1)

**AMQ5540: Application 'c\Samples\Bin64\amqscnxc.exe' did not supply a user ID and password**

**EXPLANATION:**

**The queue manager is configured to require a user ID and password, but none was supplied.**

**ACTION:** Ensure that the application provides a valid user ID and password, or change the queue manager configuration to OPTIONAL to allow applications to connect which have not supplied a user ID and password.

1/27/2017 06:25:18 - Process(3568.22) User(MUSR\_MQADMIN)  
 Program(amqzlaa0.exe) Host(WS2008R2X64) Installation(IBMMQV9)  
 VRMF(9.0.0.0) QMgr(MQG1)

**AMQ5541: The failed authentication check was caused by the queue manager CONNAUTH CHCKCLNT(REQDADM) configuration.**

**EXPLANATION: The user ID 'administrator' and its password were checked because the user ID is privileged** and the queue manager connection authority (CONNAUTH) configuration refers to an authentication information (AUTHINFO) object named 'SYSTEM.DEFAULT.AUTHINFO.IDPWOS' with CHCKCLNT(REQDADM).

Figure 5-27. Troubleshooting queue manager default settings (2 of 5)

When you look at the MQG1 queue manager log, the earliest record for this error confirms that credentials were required, but none were supplied.

The second queue manager log record provides extra granularity about the setting of the active AUTHINFO record, SYSTEM.DEFAULT.AUTHINFO.IDPWOS, to CHCKCLNT(REQADM).

You continue to the next queue manager log record.

## Troubleshooting queue manager default settings (3 of 5)

1/27/2017 06:25:19 - Process(1752.6) User(MUSR\_MQADMIN)  
 Program(amqrmpa.exe) Host(WS2008R2X64) Installation(IBMMQV9)  
 VRMF(9.0.0.0) QMgr(MQG1)

AMQ9557: Queue Manager User ID initialization failed for  
 'administrator'.

**EXPLANATION:**  
**The call to initialize the User ID 'administrator' failed with CompCode 2 and Reason 2035.**

**ACTION:**  
 Correct the error and try again.

- Solution: Use **-u** ID parameter in **amqscnxc** sample
  - **amqscnxc** prompts for password

```
amqscnxc -x localhost(1657) -c WM253.G1.SVRCONN
-u administrator MQG1
Sample AMQSCNXC start
Connecting to queue manager MQG1
using the server connection channel WM253.G1.SVRCONN
on connection name localhost(1657).
```

Type password when prompted:

```
Enter password: web1sphere
MQCONNX ended with reason code 2035
```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-28. Troubleshooting queue manager default settings (3 of 5)

The last queue manager log entry for this field shows the 2035 error code.

The amqscnxc sample takes a user ID by adding the **-u** user parameter to the other parameters that were specified earlier. The amqscnxc command takes the user ID and then prompts you for the password.

You include the correct user ID in the amqscnxc. When you are prompted for the password, you provide the correct password. Then, you get another 2035. What happened? Did the user ID and password that were provided not work? You return to the MQG1 queue manager log.

## Troubleshooting queue manager default settings (4 of 5)

```
1/27/2017 06:27:38 - Process(1752.7) User(MUSR_MQADMIN)
Program(amqrmpa.exe) Host(WS2008R2X64) Installation(IBMMQV9)
 VRMF(9.0.0.0) QMgr(MQG1)
```

### **AMQ9776: Channel was blocked by userid**

#### **EXPLANATION:**

The inbound channel 'WM253.G1.SVRCONN' **was blocked from address '127.0.0.1'** because the active values of the channel were mapped to a **userid which should be blocked**. The active values of the channel were 'MCAUSER(administrator) CLNTUSER(administrator)'.

#### **ACTION:**

Contact the systems administrator, who should examine the channel authentication records to ensure that the correct settings have been configured. The ALTER QMGR CHLAUTH switch is used to control whether channel authentication records are used. The command DISPLAY CHLAUTH can be used to query the channel authentication records.

Figure 5-29. Troubleshooting queue manager default settings (4 of 5)

You now see a different message. Earlier it was stated that channel authentication errors usually have the word “blocked” as part of the error message.

This particular error message is helpful in that it shows you the name of the channel name and connection information that were blocked. However, the key part is “mapped to a user ID that should be blocked”. The message is unclear that it was blocked because it was an administrative user. However, if you remember the three default channel authentication rules, you can determine that the problem is with the ID of the user that is running this test.

You need to add a channel authentication rule specific to the channel that is named in the error message so that a user administrator can connect even though it is an administrative ID.

## Troubleshooting queue manager default settings (5 of 5)

- Solution: Use `runmqsc` to set channel authentication rule to allow administrative users for channel WM253.G1.SVRCNN

```
SET CHLAUTH(WM253.G1.SVRCNN) TYPE(BLOCKUSER) +
USERLIST('BOGSUSER') +
DESCR('Allow administrator user to connect')
 1 : SET CHLAUTH(WM253.G1.SVRCNN) TYPE(BLOCKUSER)
USERLIST('BOGSUSER')
DESCR('Allow administrator user to connect')
AMQ8877: IBM MQ channel authentication record set.
```

`amqscnxc -x localhost(1657) -c WM253.G1.SVRCNN  
-u administrator MQG1`

Expected results:

Sample AMQSCNXC start  
Connecting to queue manager MQG1  
using the server connection channel WM253.G1.SVRCNN  
on connection name localhost(1657).

Type password when prompted  
Enter password: websphere  
**Connection established to queue manager MQG1**



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-30. Troubleshooting queue manager default settings (5 of 5)

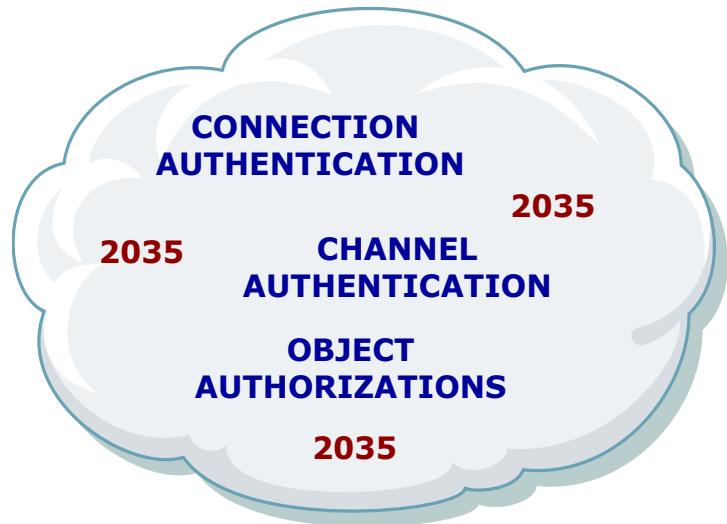
In this slide, you see the channel authentication rule specific to the channel blocked, which blocks a non-existent bogus user. By specifically blocking the bogus user for the selected channel, the administrator user ID is allowed to connect.

The test is now repeated. The addition of credentials cleared the connection authentication requirements. The addition of the channel authentication rule unblocked the connection, and the test completes successfully.

However, if user `administrator` does not have the correct authorizations to the WM253.G1.SVRCNN channel, a third 2035 error code might surface after the connection was unblocked. You learn about object authorizations in a later slide.

## Other considerations when working with IBM MQ security

- Occasionally, an IBM MQ fix pack might introduce slight behavioral changes
- Keep a baseline set of tests for your IBM MQ security configuration and run these tests after you apply any fix packs
- Ensure that channel authentication rules are self-documented, and backed up
- Ensure that object authorizations are backed up



IBM MQ security and clusters

© Copyright IBM Corporation 2017

*Figure 5-31. Other considerations when working with IBM MQ security*

Connection authentication and channel authentication are newer mechanisms than object authorizations in IBM MQ. Occasionally, a fix pack might be released which introduces slight behavior changes in the way the newer mechanisms work.

To mitigate possible problems, always have a strict set of security test cases for your organization. Implement all security hardening, starting with the development environment.

When a new fix pack is introduced, run the set of tests before and after application of the fix pack. Ensure that no behaviors are changed.

If you implement security, starting with the development environment, you can do more testing before you schedule the fixes to be applied to test and production environments.

But you must start with a documented set of test cases, and supplement these test cases with newer scenarios as needed.

Another critical practice is to ensure that all your channel authentication rules and all object authorizations are backed up.

**Plan to always run all security tests after maintenance is applied. Expected behaviors might change after maintenance is applied.**

## An introduction to IBM MQ object authorizations

- IBM MQ authorization addresses different types of access to IBM MQ resources such as:
  - Who can administer IBM MQ
  - Who can connect to the queue manager
  - Who can put or browse a message to or from a queue
- Equivalent authorization mechanisms for different operating systems
  - Object Authority Manager (OAM) for distributed platforms
  - z/OS uses RACF or a compatible external security manager (ESM)
  - IBM i uses Object Authority Manager and IBM i object level security
- Different types of commands can be used to administer authorizations
  - Control commands, such as `setmqaut`
  - IBM MQ script, or MQSC commands such as `SET AUTHREC`
  - Programmable Command Format, or PCF commands
- Control command options when connection authentication is required
  - Temporarily start the secured queue manager with the `-ns` option
  - Use equivalent MQSC commands with `runmqsc`, or use IBM MQ Explorer
  - REFRESH SECURITY must be issued in `runmqsc`, or IBM MQ Explorer



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-32. An introduction to IBM MQ object authorizations

After you get through connection authentication and channel authentication as required, what actions are you allowed to take? Applications, testers, and administrators need to work on different tasks in a queue manager. Compared to object authorizations, connection authentication and channel authentication are newer security mechanisms in IBM MQ.

Initially, customers coded exits for authentication. SSL/TLS was added later for data in transit. Object authorizations are used to disallow certain actions, such as put or get from a queue, or connect to a queue manager when the entity that is attempting the action is not granted the rights.

Object authorizations offer granularity in the actions that a user can take in a queue manager.

Depending on the operating system, different components control object authorizations. For example, distributed platforms use the object authority manager, or OAM. z/OS allows use of an external security manager, or ESM product, which is usually RACF. The final results of object authorizations across different platforms are comparable. This course looks at object authorizations for distributed platforms.

Two types of IBM MQ commands can manage object authorizations in distributed platforms. One type of command is the IBM MQ scripts, or MQSC commands, which are created in an MQSC, or `runmqsc` session. The other type of command is control commands, which are run outside of a `runmqsc` session.

When connection authentication is required, control commands cannot run unless the queue manager is restarted in “ns” mode by using the `-ns` parameter to start the queue manager. When the queue manager is restarted in ns mode, the CONNAUTH setting is ignored. However, the channel initiator, command server, listeners, and other IBM MQ services do not automatically start when `-ns` is specified in the queue manager start command.

You might find it simpler to issue MQSC commands. When you need to use `runmqsc` when connection authentication is required, you include the `-u` followed by the user ID in the `runmqsc` command. You are then prompted for the password before the `runmqsc` session starts.

When connection authentication is required, IBM MQ Explorer needs to be configured to use credentials before it can be used for the secured queue manager.

## Components that are used to control access to an object

- Name of queue manager that owns the object
- Object type and name are specified as a profile
  - Explicit name of an object
  - Generic name, including wildcard characters
- One or more principals and group names
- Authorizations

**See the IBM Knowledge Center for complete lists**

**Object type examples**  
-t parameter

channel - chl

Queue - q

qmqr

topic - top

**Name profile examples**  
-n parameter

**Explicit:** -n MQG1.IN -t q  
refers to queue MQG1.IN

**? single character:** -n MQG1.A?  
refers to objects MQG1.A1, MQG1.AB

**\* single qualifier:** -t chl MQ.\*.CHL  
channels MQ.MQG1.CHL, MQ.USR1.CHL

**\*\* one time qualifier:**  
-t top \*\* refers to all topics  
-t q SYSTEM.FTE.COMMAND.\*\*  
refers to command queues for all agents

**Authorization examples (at end, no parameter)**

connect

browse

get

put

sub

pub

allmqi

inq

crt

none

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-33. Components used to control access to an object

As with other tasks, working with object authorities can be a daunting task until you gain proficiency. Whether you use MQSC or control commands, each call has parameters that are required, which follow the setmqaut or SET AUTHREC command, according to the action taken.

All control commands require the queue manager name parameter, which follows the -m keyword. MQSC commands do not use the queue manager parameter since the runmqsc session already specifies the queue manager in use.

The next parameter is the name of the object, which uses the -n parameter. If the type of object is the queue manager, the name of the object is omitted. You use the command with the type qmqr parameter to grant the authority to connect to the queue manager. When the type of object is other than qmqr, you can use the exact object name, or you can use wildcards to include different object names. When the rule applies to more than one object name, it can be referred to as a “profile”.

Another required parameter is the type of object, which can be -qmqr for the queue manager, or it can identify other objects such as a queue, a channel, or a topic. The possible types of objects are documented in the IBM Knowledge Center.

Also required is the principal or group name parameter to whom the specific authority is granted. You might find it easier to create a “profile” for a specific group of users, and then include the user in the group. You can have more than one user or principal user included in the command, provided

you repeat the `-g` or `-p` for each extra entity. Each command must have at least one group or principal name. A Windows principal name can include a domain-type user ID format.

The last parameter is the actual authorization that is granted. Different objects can be granted different authorizations. However, it would not make much sense to grant connection authority for a queue. Connection rights are granted to a queue manager, as the connection is done to the queue manager.

The slide shows an example of the authorizations that can be granted. For example, you can grant put and get authority, or only put authority. You can grant authority to connect to the queue manager. Or you can grant authority to publish or subscribe. The IBM Knowledge Center documents the authorizations. Some authorizations group several other authorizations. For example, `allmqi` includes all the individual authorizations: `altusr`, `browse`, `connect`, `get`, `inq`, `pub`, `put`, `resume`, `set`, and `sub`.

Refer to IBM Knowledge Center for the list of authorizations and authorization groupings.

You now look at how to put these parameters together in some examples of control commands `dspmqaut` and `setmqaut`.

## dspmqaut and setmqaut examples

1. **dspmqaut -m USR1 -n SYSTEM.FTE.COMMAND.\*\* -t q -p fileusr1**  
AMQ7085: Object SYSTEM.FTE.COMMAND.\*\*, type q not found.
  2. **dspmqaut -m USR1 -n SYSTEM.FTE.COMMAND.USR1AGT1 -t q -p fileusr1**  
Entity fileusr1 has the following authorizations for object SYSTEM.FTE.COMMAND.USR1AGT1:
  3. **setmqaut -m USR1 -n SYSTEM.FTE.COMMAND.\*\* -t q -p fileusr1 +connect +put**  
AMQ7097: You gave an authorization specification that is not valid.
  4. **setmqaut -m USR1 -n SYSTEM.FTE.COMMAND.\*\* -t q -p fileusr1 +put**  
The setmqaut command completed successfully.
  5. **setmqaut -m USR1 -t qmgr -p fileusr1 +connect**  
The setmqaut command completed successfully.
  6. **dspmqaut -m USR1 -n SYSTEM.FTE.COMMAND.\*\* -t q -p fileusr1**  
Entity fileusr1 has the following authorizations for object SYSTEM.FTE.COMMAND.\*\*:  
put
  7. **dspmqaut -m USR1 -n SYSTEM.FTE.COMMAND.USR1AGTS -t q -p fileusr1**  
Entity fileusr1 has the following authorizations for object SYSTEM.FTE.COMMAND.USR1AGTS: put
- ```
C:\>dspmqaut -m USR1 -t qmgr -p fileusr1
Entity fileusr1 has the following authorizations for object USR1:
    connect
```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-34. dspmqaut and setmqaut examples

As is common when you learn a new command, it might take a little familiarization to become used to the output of both control and MQSC commands. The commands in these slides are examples that are used for IBM MQ Managed File Transfer objects, in particular the command queue. IBM MQ Managed File Transfer is taught in course ZM003.

The first command in the slide resulted in an error, which stated that the named object of type q was not found. You know that the command queue exists because you created it before you were able to start the agent. However, what the dspmqaut does not find is a profile for object named SYSTEM.FTE.COMMAND.**, which was previously defined in a setmqaut command.

Notice the user that is specified in the commands in this slide, fileusr1. This user does not have many privileges and is not an administrative user.

The second command is a correction of the first command that uses a fully qualified object name, and this time the dspmqaut process found it. However, user fileusr1 did not have any authorities for this object, so dspmqaut gets ready to find authorizations and displays the object name, but no authorizations were found, so the list is blank.

The third command uses the profile type name for the command queue, and attempts to grant both connect and put authority to the command queue. However, it does not make much sense to grant connect authority to a queue, so the command displays an error message.

You now break up the third command, into commands 4 and 5, and use the right type of object for the correct authority. Now you create the profile SYSTEM.FTE.COMMAND.** to grant put authority to the queue in command 4 with object type -q. And you allow fileuser1 to connect to the queue manager in command 4 by using object type qmgr.

Now in command 6, you can repeat the command as typed in command 1. Since setmqaut created the SYSTEM.FTE.COMMAND.**, it now exists and displays the rights that are granted to the user.

However, queue manager USR1 had more than one agent command queue. If you repeat the dspmqaut for a specific type q object, SYSTEM.FTE.COMMAND.USR1AGTS, that fits the SYSTEM.FTE.COMMAND.**, you can see how by using the profile, you authorized user fileusr1 access to more than one queue with a queue name included in the profile.

Mixed commands in a secured queue manager (1 of 3)

```
dspmqaut -m MQG1 -t qmgr -p fileusr1
AMQ7077: You are not authorized to perform the requested operation.

endmqm -i MQG1
...
IBM MQ queue manager 'MQG1' ended.
```

If “not authorized” due to connection authentication, restart the queue manager with the **-ns** option

```
strmqm -ns MQG1
IBM MQ queue manager 'MQG1' starting.
```

```
...
dspmqaut -m MQG1 -t qmgr -p fileusr1
Entity fileusr1 has the following authorizations for object MQG1:
    connect
dspmqaut -m MQG1 -n MQG1.IN -t q -p fileusr1
Entity fileusr1 has the following authorizations for object MQG1.IN:
```

Figure 5-35. Mixed commands in a secured queue manager (1 of 3)

Queue manager MQG1, from the previous slide, does not require connection authentication. However, when connection authentication is required, as in queue manager MQG1, an attempted control command fails with a 2035 not authorized.

As mentioned earlier, a queue manager with the **-ns** option has limited capabilities. Therefore, while you are able to use control commands by restarting the queue manager with the **-ns** option, you might want to consider the use of equivalent MQSC commands. You can provide credentials to the runmqsc utility by using the **-u** parameter with a user ID, and respond to the password prompt.

In this slide, queue manager MQG1 is restarted with the **-ns** option, and control commands can be used again. However, to resume normal operations, the queue manager must be again restarted without the **-ns** parameter.

Mixed commands in a secured queue manager (2 of 3)

```
runmqsc -u administrator MQG1
Enter password:
*****
dis authrec profile(MQG1.IN) objtype(q) principal(fileusr1)
    1 : dis authrec profile(MQG1.IN) objtype(q) principal(fileusr1)
AMQ8405: Syntax error detected at or near end of command segment
below:-
dis authrec profile(MQG1.IN) objtype(q ...)
dis authrec profile(MQG1.IN) objtype(queue) principal(fileusr1)
    2 : dis authrec profile(MQG1.IN) objtype(queue) principal(fileusr1)
AMQ8459: Not found.
dis q(MQG1.IN)
    3 : dis q(MQG1.IN)
AMQ8409: Display Queue details. .... .... .... ....
dis authrec profile(MQG1.IN) objtype(queue) principal(administrator)
    4 : dis authrec profile(MQG1.IN) objtype(queue)
principal(administrator)
AMQ8864: Display authority record details.
    PROFILE(MQG1.IN)
    ENTITY(administrator@WS2008R2X64)      ENTTYPE(PRINCIPAL)
    OBJTYPE(QUEUE)
    AUTHLIST(BROWSE,CHG,CLR,DLT,DSP,GET,INQ,PUT,PASSALL,PASSID,SET,SETALL,
SETID)
```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-36. Mixed commands in a secured queue manager (2 of 3)

In this slide, you type equivalent commands to display object authorities by using the MQSC DISPLAY AUTHREC command.

When you start working with the MQSC commands, although the results and objectives are equivalent, you discover that some subtle differences in syntax and behavior exist between the two types of commands.

The first in the slide is runmqsc, which is accessed by including the -u with the user ID. Before the runmqsc session is available, you are prompted for the password.

The equivalent MQSC command is typed in the runmqsc session. However, soon you find that in the MQSC command, the abbreviation for queue is not valid.

The command is retyped, and this time “queue” is spelled for the object type parameter. The response for ID fileusr1 is now “Not found”. However, if you display the queue name, the queue is there. In this case, what is not found is any authorization for user fileusr1 to queue MQG1.IN.

You can confirm by typing the same query, but this time for user administrator. You then see the authorizations to queue MQG1.IN that user administrator has.

Mixed commands in a secured queue manager (3 of 3)

```

set authrec profile(MQG1.IN) objtype(queue) principal(fileusr1)
authadd(put)
      5 : set authrec profile(MQG1.IN) objtype(queue)
principal(fileusr1) authadd(put)
AMQ8862: IBM MQ authority record set.

refresh security type(authserv)

      6 : refresh security type(authserv)
AMQ8560: IBM MQ security cache refreshed.

end
      7 : end
6 MQSC commands read.
One command has a syntax error.
One valid MQSC command could not be processed.

dspmqaut -m MQG1 -n MQG1.IN -t q -p fileusr1
Entity fileusr1 has the following authorizations for object MQG1.IN:

      put

```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-37. Mixed commands in a secured queue manager (3 of 3)

In this slide, you now, as the administrator user, grant user fileusr1 put authority to queue MQG1.IN. You then refresh the authorization service security in the queue manager, and exit the runmqsc session.

The queue manager is still running with the `-ns` option. You can now repeat the query into the authorizations to queue type object MQG1.IN for queue manager MQG1 that user fileusr1 has. You see the put authority that was granted in the runmqsc session.

Queue manager MQG1 eventually needs to be restarted without the ns option.

If you are not familiar with setting and displaying object authorizations, a good approach is to define some queues and channels and to experiment with removing and granting authorities. An excellent source for this work can be found in the documents that you can obtain by using the search string that is provided at the end of this unit.

Troubleshooting an object authorization problem (1 of 2)

```
set MQSAMP_USER_ID=fileusr1
```

```
C:\>amqspput MQG1.IN MQG1
Sample AMQSPUTO start
Enter password: websphere
target queue is MQG1.IN
MQOPEN ended with reason code 2035
unable to open queue for output
Sample AMQSPUTO end
```

- User passed connection authentication and channel authentication challenges
- Then, another 2035: Review queue manager MQG1 error log

```
2/7/2017 11:30:01 - Process(3936.21) User(MUSR_MQADMIN)
Program(amqzlaa0.exe)
          Host(WS2008R2X64) Installation(IBMMQV9)
          VRMF(9.0.0.0) QMgr(MQG1)
AMQ8077: Entity 'fileusr1@ws2008r2x64' has insufficient authority to
access object 'MQG1.IN'.
EXPLANATION:
The specified entity is not authorized to access the required object.
The following requested permissions are unauthorized: put
ACTION: Ensure that the correct level of authority has been set for
this entity against the required object, or ensure that the entity
is a member of a privileged group.
```

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-38. Troubleshooting an object authorization problem (1 of 2)

In this slide, a test is run to show an object authorization problem, and how to investigate it. You might find the process familiar.

The `MQSAMP_USER_ID` variable is used to specify a low privilege user, fileusr1, and satisfy the connection authentication credentials. User fileusr1 attempts to put a message to queue MQG1.IN, in the MQG1 queue manager, which results in a 2035. Each time a 2035 surfaces, you check the queue manager log, in this case the MFTS queue manager log.

The log record confirms the name of the user and the name of the object that the user attempted to use. Later in the message, it provides additional information that the authorization that is required is put authority.

Troubleshooting an object authorization problem (2 of 2)

- Solution:

- Start a `runmqsc` session for queue manager MQG1 (optionally `setmqaut`)
- After refresh, have user retest for successful access

```
set authrec profile(MQG1.IN) objtype(queue) principal(fileusr1)
authadd(put)

  1 : set authrec profile(MQG1.IN) objtype(queue)
principal(fileusr1) authadd(put)
AMQ8862: IBM MQ authority record set.

refresh security type(authserv)

  2 : refresh security type(authserv)
AMQ8560: IBM MQ security cache refreshed.

... ... ... ... ...
```

```
C:\>amqspput MQG1.IN MQG1
Sample AMQSPUT0 start
Enter password: web1sphere
target queue is MQG1.IN
abc
```

```
Sample AMQSPUT0 end
```

Figure 5-39. Troubleshooting an object authorization problem (2 of 2)

To resolve the 2035, the `MQSC SET AUTHREC` command is used to grant user `fileusr1` put authorization to queue `MQG1.IN`. Notice that a refresh of the authorization service is also issued.

The test is repeated, and this time `fileusr1` had adequate authorizations to put a message to the `MQG1.IN` queue, in queue manager `MQG1`.

In the example, a principal is granted authorizations. However, ideally you would have the user in a group and grant authorization to the group rather than to a single user.

Problems and troubleshooting recap



- What was the error entry in the queue manager log?
- Is the problem in a distributed or cluster message channel, or a client channel?
- Did you issue **DIS CLUSQMGR** or **DIS CHS** for the target queue manager?
- Is the channel in retry?
- Is the 2035 problem due to connection authentication, channel authentication, or object authorizations?
- Did the error entry include the word “blocked”?
- Do you have two queue managers by the same name in a cluster?

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-40. Problems and troubleshooting recap

You now review some of the considerations you need to revisit when you are researching a security or other problem.

- First, ensure that you are in the queue manager log, that is, the log under the `qmgrs` directory in distributed IBM MQ platforms. Then, carefully select the error entry, and accompanying entries for the date and time stamp when the error occurred. Many practitioners tend not to read the log entries carefully, and miss the problem.
- Is your problem due to connection authentication, or channel authentication? Remember, if you see a message with “blocked”, you are looking at channel authentication. A connection authentication problem is normally resolved by providing credentials. A channel authentication blocked problem is solved by creating an adequate channel authentication rule or record.
- What channel are you having a problem with? If you tried to look at the channel status, remember that a `CLUSSDR` channel output in the `DIS CHS` command is expected. However, a `CLUSSDR` displayed for the `DIS CLUSQMGR` command indicates an error, possibly in one of the cluster channel definitions. This error might also include an error when spelling the cluster name.
- Do you by any chance have two like named queue managers in the cluster? It is expected that the reason is that you moved a cluster queue manager to another server and then failed to

remove the older version of the queue manager from the cluster. To remove the older version, you use the `RESET` command with the QMID of the older queue manager.

mq t rob what happens in vegas ibm.com

All News Maps Images Shopping More Settings Tools

About 53,200 results (0.74 seconds)

Mission: Messaging: Understanding WebSphere MQ authorization - IBM
[www.ibm.com > Learn > WebSphere](#) ▾
 Mar 3, 2010 - T.Rob Wyatt ... At its simplest, IBM WebSphere MQ authorization controls grant a set of privileges to a single entity for a single object. ... setmqaut -m VENUS -t qmgr -g fruit -all +inq +connect ... When this happens, the application does not fail so there is nothing I look forward to meeting you in Las Vegas!

Comment lines: T.Rob Wyatt: WebSphere MQ security heats up - IBM
[www.ibm.com > Learn > WebSphere](#) ▾
 Nov 7, 2007 - Are your MQ channels as secure as they should be? What you need to know about recent developments in IBM WebSphere MQ security and, ...
 Missing: happens vegas

IBM MQ security and clusters © Copyright IBM Corporation 2017

Figure 5-41. Search string to additional IBM MQ security reference

The link in this slide is provided as a next step in your work with authorizations. Although these links were written some years back, they are still applicable to authorizations today.

Since a link itself might change, rather than a link you use the search string:

"mq t rob what happens in vegas ibm.com"

Unit summary (1 of 2)

- Summarize security areas and terminology
- Describe IBM MQ connection authentication
- Identify the IBM MQ connection authentication initial settings in a new queue manager
- Explain how to modify connection authentication and respond to the new settings
- Describe channel authentication and the types of channel authentication records
- Describe the channel authentication initial default settings and rules in a new queue manager
- Explain channel authentication rule precedence
- Describe the channel authentication back-stop rule
- Explain how to set, change, and remove channel authentication records
- List considerations to observe for a successful channel authentication implementation

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-42. Unit summary (1 of 2)

Unit summary (2 of 2)

- Describe IBM MQ object authorizations
- Explain how to use the dspmqaut and setmqaut control commands to display and grant object authorizations
- Explain how to use the IBM MQ script commands DISPLAY AUTHREC and SET AUTHREC to display and grant object authorizations
- Differentiate when to use the object authorization control commands, and when to use the object authorization IBM MQ script commands

IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-43. Unit summary (2 of 2)

Review questions (1 of 2)

1. Select all that apply. How do you check the connection authentication settings of a queue manager by using MQSC commands?
 - A. Use the MQSC command `DIS QMGR CONNAUTH`
 - B. Use the MQSC command `DIS CHLAUTH`
 - C. Use the MQSC command
`DIS AUTHINFO(authinfo.name) CHCKLOCL CHCKCLNT`
 - D. All answers are correct
2. True or False: To determine whether channel authentication is enabled in the queue manager, you use the MQSC command `DIS CHLAUTH`.



Review questions (2 of 2)

3. Select the best answers. You type a `dspmqaut` control command but receive a “**You are not authorized ...**” response. What are your options?
- A. Restart the queue manager with the `-ns` option and repeat the command
 - B. Use the equivalent MQSC `DISPLAY AUTHREC` command in a `runmqsc` session
 - C. Repeat the `dspmqaut` command by including the `-u` user name parameter
 - D. All answers are correct



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-45. Review questions (2 of 2)

Review answers (1 of 2)



1. Select all that apply. How do you check the connection authentication settings of a queue manager by using MQSC commands?
 - A. Use the MQSC command `DIS QMGR CONNAUTH`
 - B. Use the MQSC command `DIS CHLAUTH`
 - C. Use the MQSC command
`DIS AUTHINFO(authinfo.name) CHCKLOCL CHCKCLNT`
 - D. All answers are correct

The answer is A and C. You first check what `AUTHINFO` record is active in the queue manager `CONNAUTH` parameter, and then check the `CHCKLOCL` and `CHCKCLNT` settings of the `AUTHINFO` record set in `CONNAUTH`.

2. True or False: To determine whether channel authentication is enabled in the queue manager, you type the MQSC command `DIS CHLAUTH`.

The answer is True. Next, you need to display the `CHLAUTH` rules in the queue manager to determine how to proceed.

Review answers (2 of 2)

3. Select the best answers. You type a `dspmqaut` control command but receive a “**You are not authorized ...**” response. What are your options?
- A. Restart the queue manager with the `-ns` option and repeat the command
 - B. Use the equivalent MQSC `DISPLAY AUTHREC` command in a `runmqsc` session
 - C. Repeat the `dspmqaut` command by including the `-u` user name parameter
 - D. All answers are correct

The answer is A and B, but you might find B less disruptive to use.



Exercise: Working with IBM MQ security

© Copyright IBM Corporation 2017

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

Figure 5-48. Exercise: Working with IBM MQ security

Exercise objectives (1 of 2)

- Review the commands that are used to harden connection authentication and channel authentication
- Harden connection authentication and channel authentication in a queue manager
- Create a type ADDRESSMAP channel authentication record to allow a specific queue manager to connect to a queue manager with strict channel authentication
- Differentiate between connection authentication and channel authentication entries in the queue manager log
- Create a type QMGRMAP channel authentication record to allow the cluster member queue managers to interact with a cluster queue manager with strict channel authentication
- Invoke the runmqsc utility with credentials to access a queue manager with required connection authentication



Exercise objectives (2 of 2)

- Use the dspmqaut and setmqaut control commands to display and set object authorities
- Use the MQSC DISPLAY AUTHREC and SET AUTHREC commands to display and set object authorities
- Use control commands in a queue manager with required connection authentication



IBM MQ security and clusters

© Copyright IBM Corporation 2017

Figure 5-50. Exercise objectives (2 of 2)

Unit 6. Influencing workload balancing behavior

Estimated time

01:00

Overview

This unit describes capabilities that can change the base workload balancing in a clustered environment.

How you will check your progress

- Checkpoints
- Lab exercise

References

IBM Knowledge Center

Unit objectives

- Describe the factors to consider before altering base workload management
- Explain when cluster workload management takes place
- Summarize the cluster workload management algorithm
- Explain how to identify and remove message affinities that negatively impact load balancing
- Describe how to use various queue attributes to influence load balancing
- Describe how to use various channel attributes to influence load balancing
- Describe how to use the queue manager properties to influence load balancing
- Describe when and how to use a customized workload balancing exit

Before you begin

- Baseline the expected load distribution before implementing any of the customizations included in this unit
 - Is there a reliable source of message volumes and sizes?
- Baseline and document the expected load balancing behavior
 - Are the configuration decisions as simple as possible?
 - Is clear documentation available for the configurations and message flow behavior expectations?
 - How is the IBM MQ administration sourced?
 - Is the IBM MQ administration team aware of the customized configuration?
 - If load balancing exits are included, who supports them and who is available to resolve a production problem during a peak load period?
 - How serious would an extended load balancing problem be during a peak load period?

Workload balancing in clusters

- If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to
- Uses the cluster workload management algorithm and cluster workload-specific attributes to determine the best queue manager
- Specify the cluster workload channel attributes on the cluster-receiver channels at the target queue managers

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-3. Workload balancing in clusters

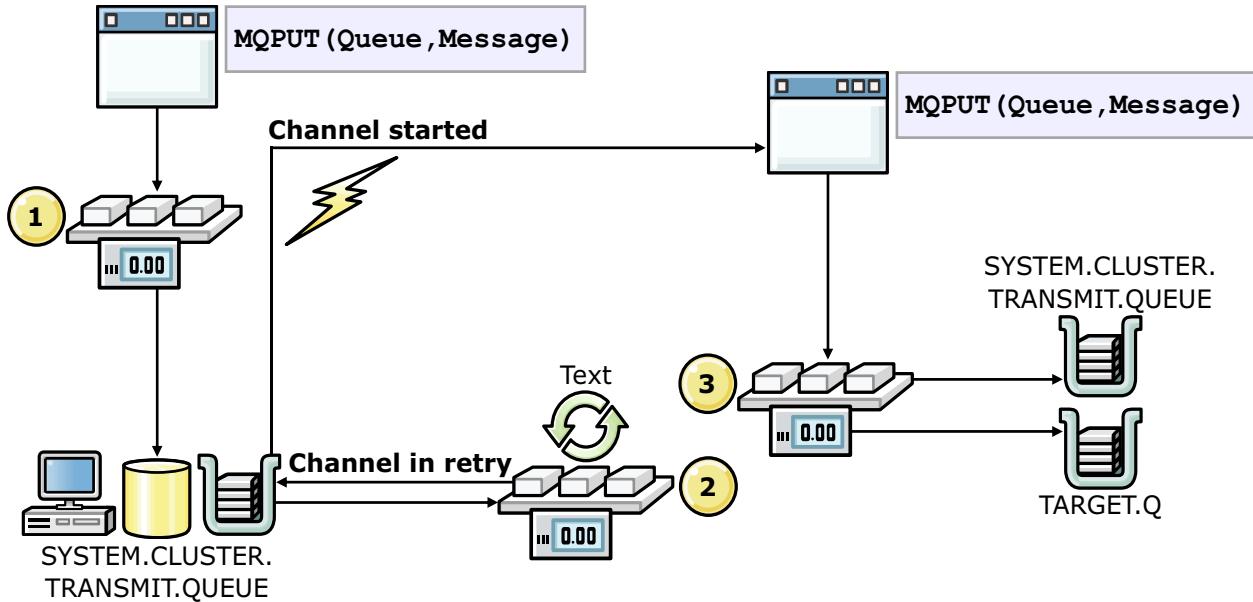
Cluster support allows more than one queue manager to host an occurrence of the same queue. Therefore, two or more queue managers can be clones of each other, capable of running the same applications and having local definitions of the same queues. This technique can be used to implement queues for two reasons:

- To increase the capacity available to process messages
- To introduce an ability to fail over work from one server to another and improve availability of the service

This architecture can 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.

When workload balancing occurs



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-4. When workload balancing occurs

In a typical cluster, a message is put to a queue manager in the cluster. In some cases, the message is destined for a remote queue manager that is also in the cluster. In this scenario, workload balancing can occur in three places.

1. Workload balancing can occur at the MQOPEN or MQPUT, depending on the bind options that are associated with the message affinity.

Regardless of the bind options, 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 when a channel from a queue manager to a remote 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.

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.

3. Workload balancing can occur at the 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.

How message affinity affects workload balancing

- If BIND_NOT_FIXED is specified, workload balancing can occur at the MQOPEN or MQPUT
- When an application opens a queue with BIND_ON_OPEN, the round-robin selection is started only in the open call, rather than for each message
- Default option of a queue definition is BIND_ON_OPEN with DEFBIND(OPEN) queue parameter

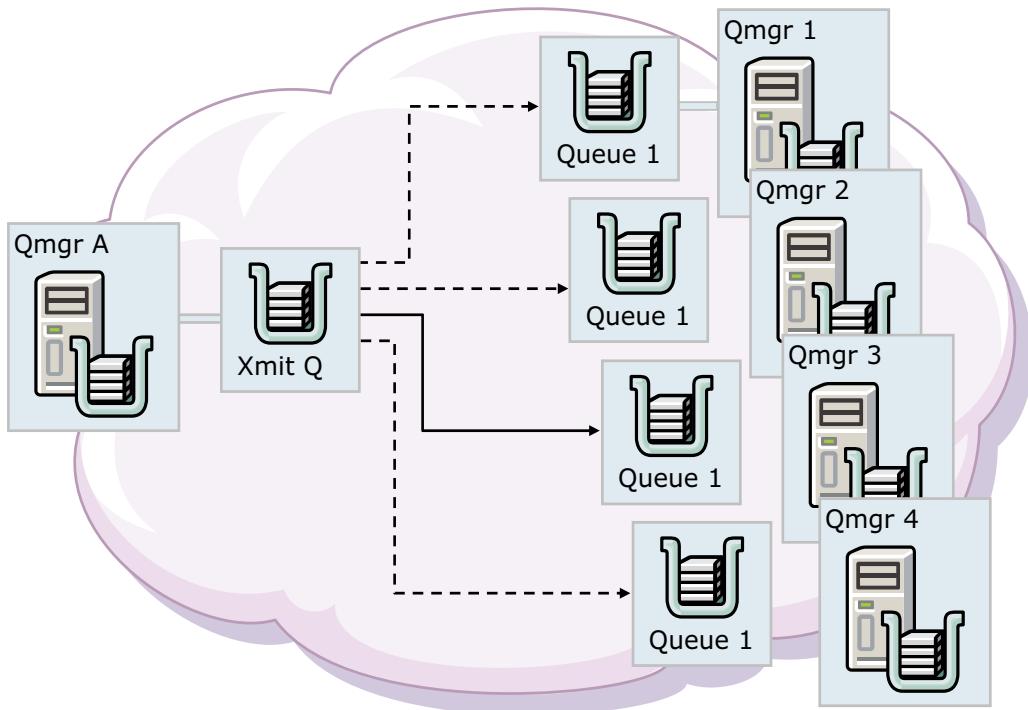
Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-5. How message affinity affects workload balancing

Message affinity and bind options are described in detail later in this unit.

Workload: Many queue instances



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

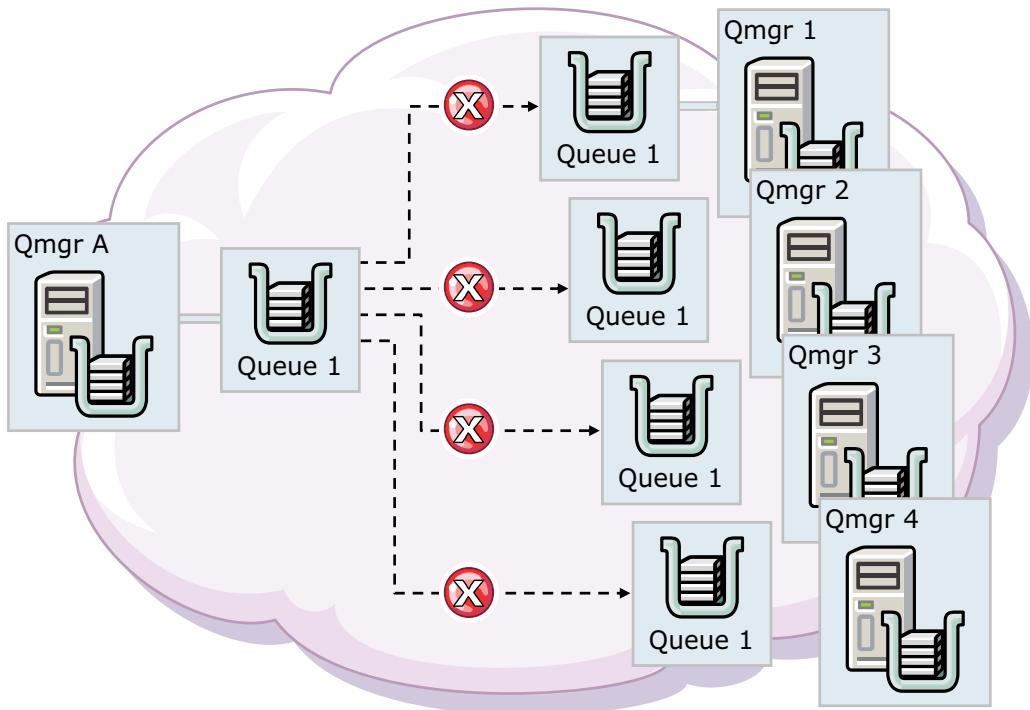
Figure 6-6. Workload: Many queue instances

Before a queue instance can be considered eligible for participation in round-robin selection, several checks are made to confirm that the queue instance is available. The full algorithm is as follows:

- First, if an instance of the queue is available in the local queue manager, then it is chosen regardless of the availability of other instances.
- If no local instance is available, the channel, queue manager, and queue instances are checked to confirm that they are available.
- If more than one channel to the same queue instance is available, the one with the higher network priority (NETPRTY) is preferred to one with lower priority.
- The least recently used queue instance and channel that remain are selected.

You can use a cluster workload exit to alter the default routing behavior.

Workload: A local queue instance



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-7. Workload: A local queue instance

If a queue manager has an available local instance of the queue, messages are put in the local queue instance and the workload management algorithm is not used.

If the local queue becomes unavailable, the workload management algorithm selects another instance of the same queue by using cluster techniques.

The workload balancing is reduced when you have:

- A local instance of the queue
- Message affinities that require the use of a single instance of the destination queue

Cluster workload (CLWL) management options

- Use-queue (CLWLUSEQ)
- Rank
 - Channel rank (CLWLRANK)
 - Queue rank (CLWLRANK)
- Priority
 - Channel net priority (NETPRTY)
 - Channel priority (CLWLPRTY)
 - Queue priority (CLWLPRTY)
- Most recently used channel (CLWLMRUC)
- Channel weighting (CLWLWGHT)
- Multiple cluster transmission queues

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-8. Cluster workload (CLWL) 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
- Allow 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 unit.

CLWL use-queue basics

- Allows remote queues to be chosen when a local queue exists
- Messages that a cluster channel receives must be put to a local queue if one exists

```
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

```
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

Figure 6-9. CLWL use-queue basics

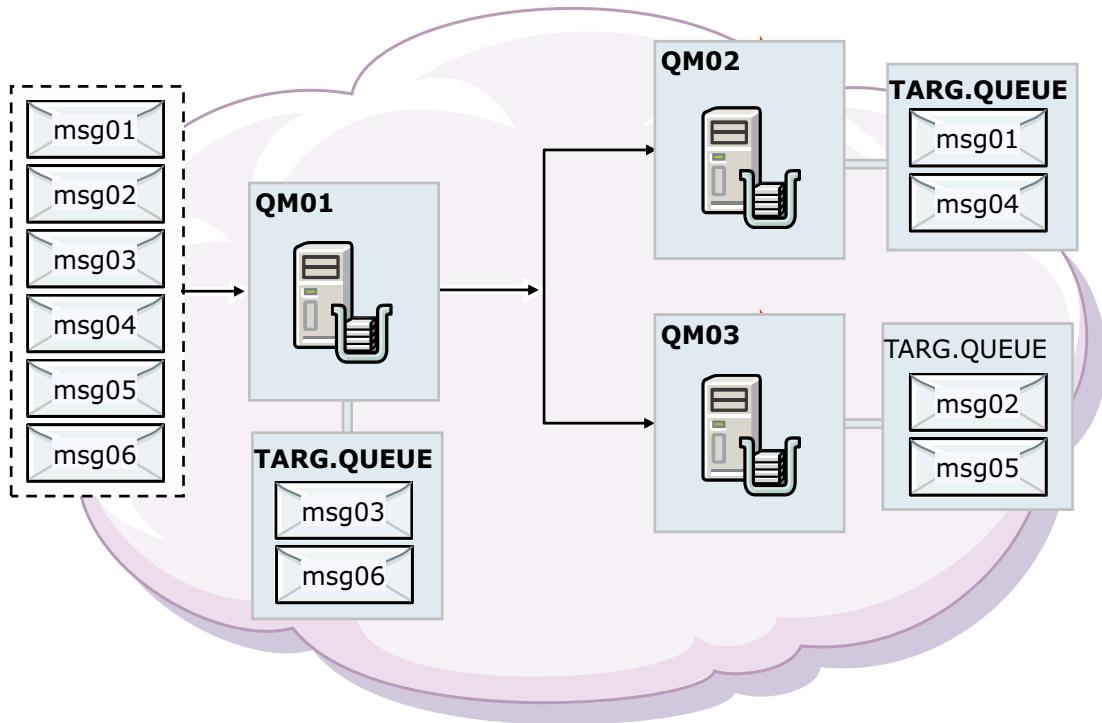
The cluster workload use-queue attribute, **CLWLUSEQ**, specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance. An exception is in cases where the MQPUT originates from a cluster channel.

The valid options for the cluster workload use-queue attribute on a queue are **LOCAL**, **ANY**, and **QMGR**.

- If you specify **LOCAL**, the local queue is the only target of the MQPUT operation.
- 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 **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.

The valid options for the use-queue attribute on a queue manager are **LOCAL** and **ANY**.

CLWL use-queue any



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

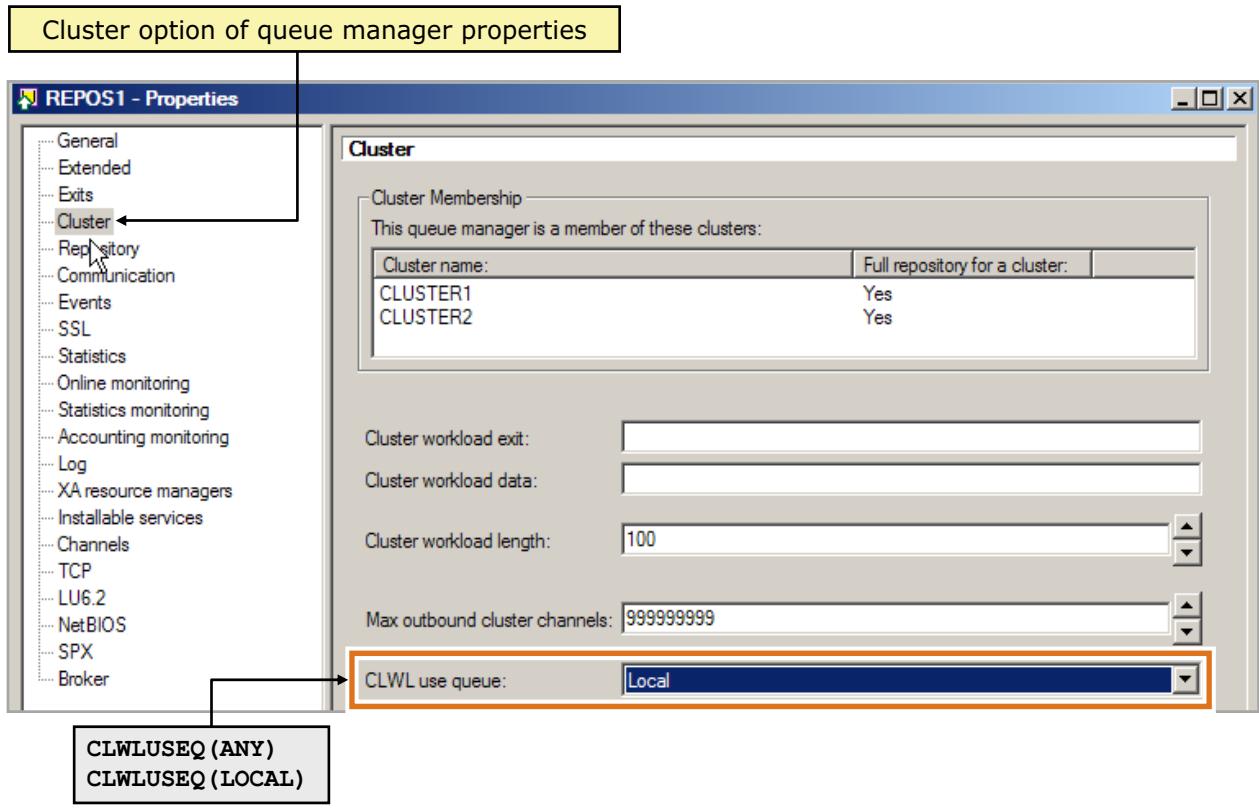
Figure 6-10. CLWL use-queue any

The figure shows a cluster with three queue managers. Each queue manager contains a local queue named TARG.QUEUE.

When the workload “use queue any” option is configured, messages are sent with a round-robin approach. The first message is sent to QM02, the second message is sent to QM03, and the third message is sent to QM01. The process continues until it has no more messages.

IBM Training IBM

Specifying CLWL use-queue on a queue manager



Influencing workload balancing behavior

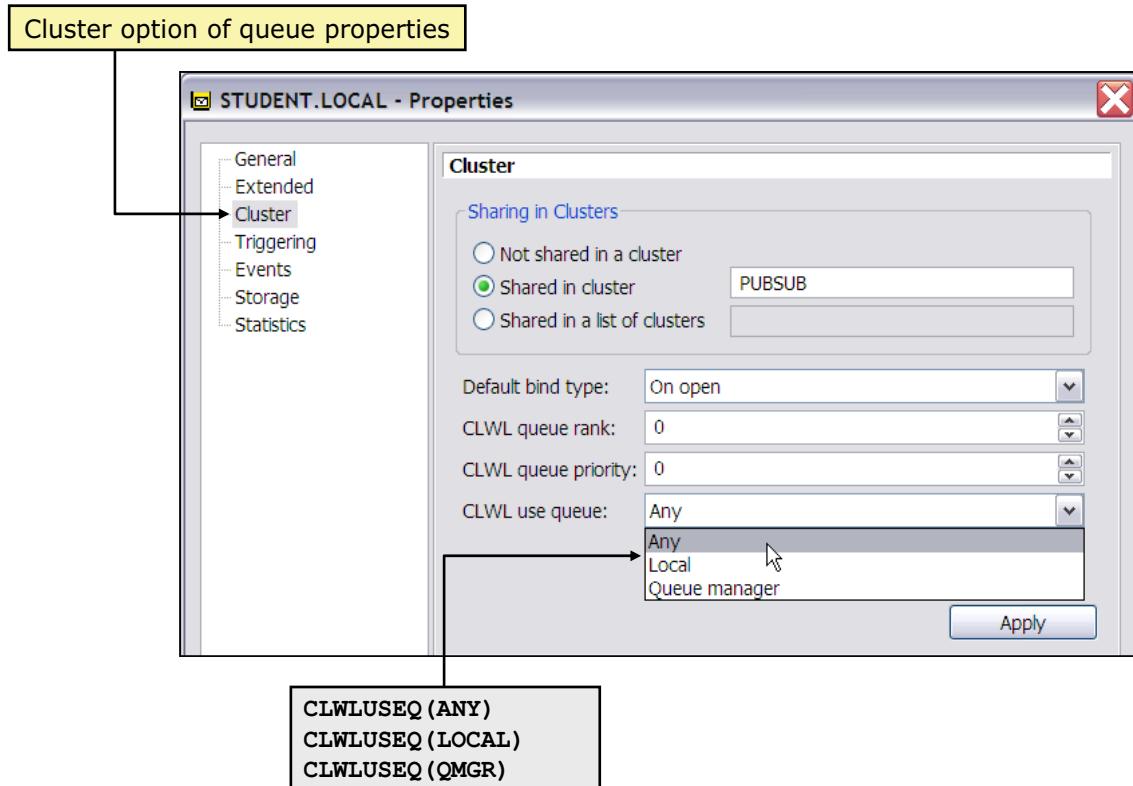
© Copyright IBM Corporation 2017

Figure 6-11. Specifying CLWL use-queue 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.

Specifying CLWL use-queue on a queue



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-12. Specifying CLWL use-queue on a queue

The figure shows how to configure the cluster workload use-queue attribute for queue in the queue **Cluster** properties.

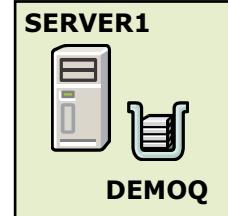
- Setting the **CLWL use queue** property to **Any** is equivalent to specifying `CLWLUSEQ(ANY)` in an MQSC command.
- Setting the **CLWL use queue** property to **Local** is equivalent to specifying `CLWLUSEQ(LOCAL)` in an MQSC command.

Setting the **CLWL use queue** property to **Queue manager** is equivalent to specifying `CLWLUSEQ(QMGR)` in an MQSC command.

CLWL use-queue example

- Start with the use-queue defaults
 - 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 are delivered to Server1

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-13. CLWL use-queue example

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.

CLWL 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( )
```

Influencing workload balancing behavior

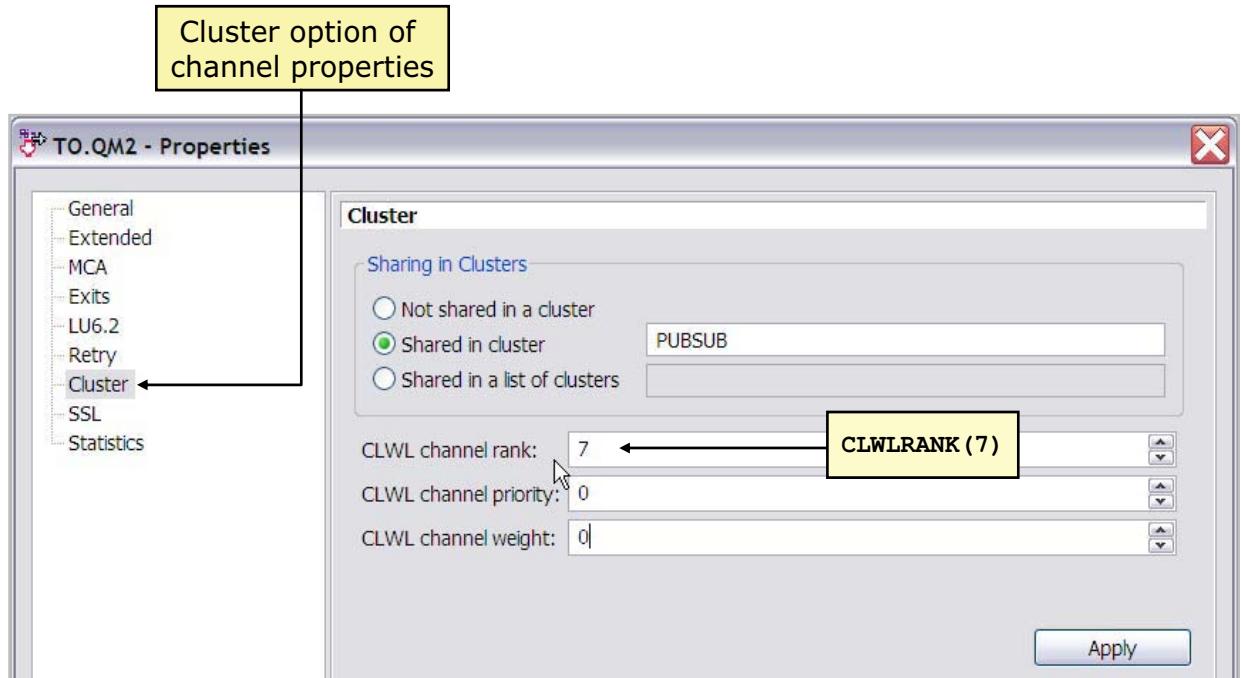
© Copyright IBM Corporation 2017

Figure 6-14. CLWL 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. This 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 `CLWLRANK` attribute on cluster-receiver channel definitions only.

Specifying CLWL channel rank



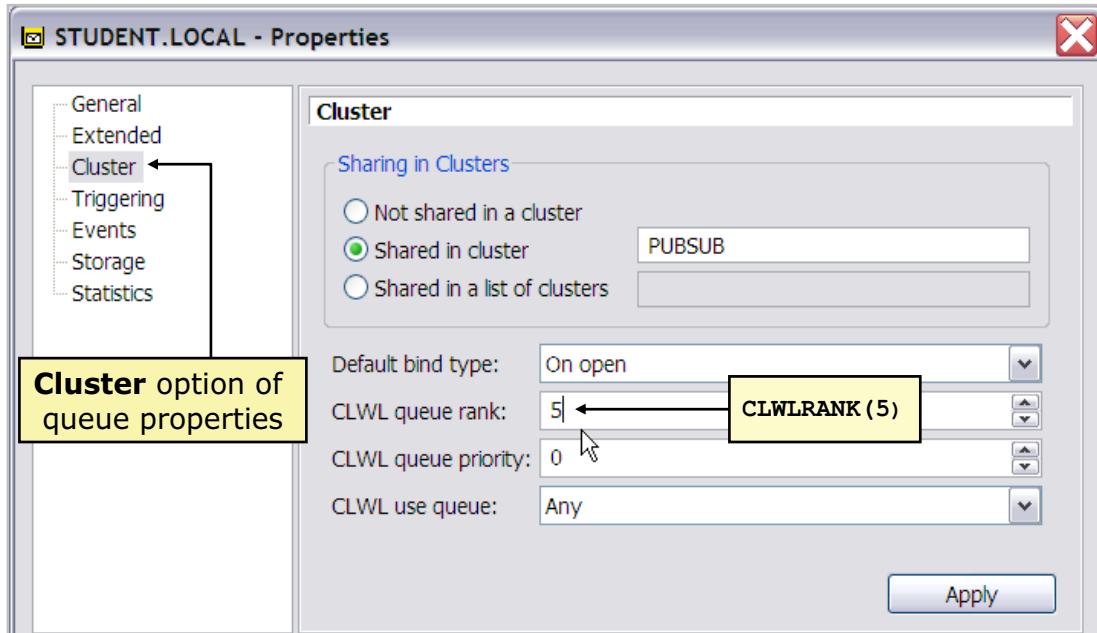
Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-15. Specifying CLWL channel rank

The figure shows how to configure the cluster workload rank attribute (`CLWL RANK`) for a channel in the IBM MQ Explorer channel **Cluster** properties.

Specifying CLWL queue rank



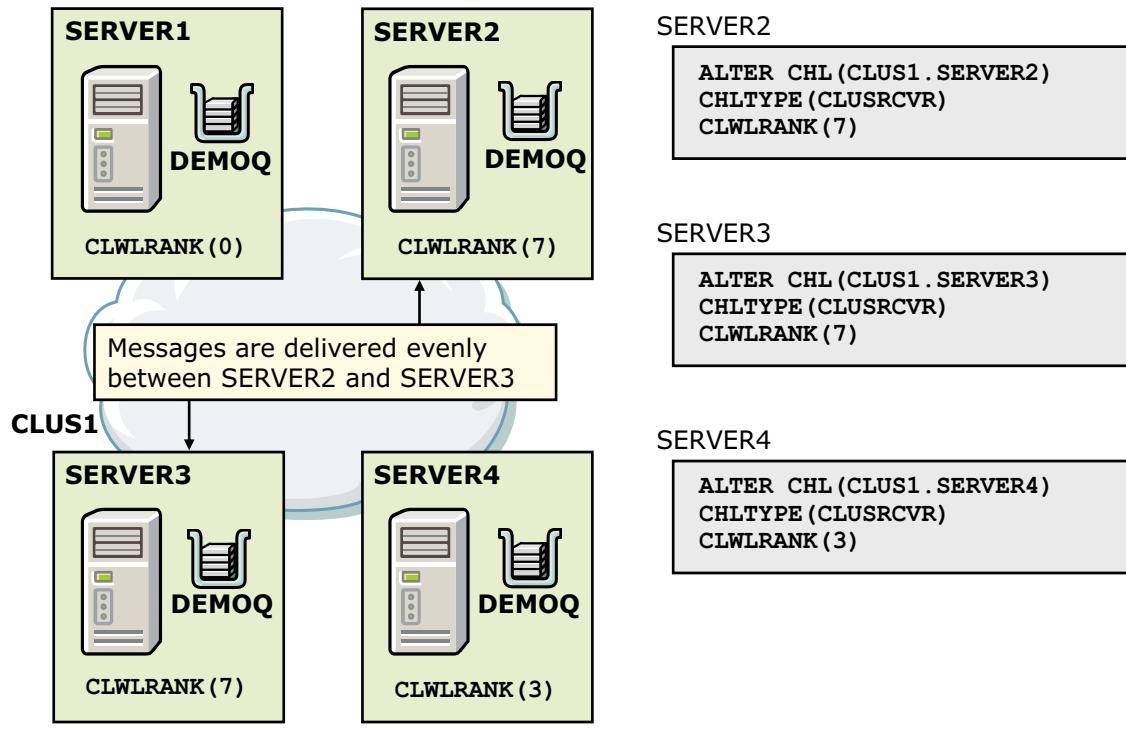
Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-16. Specifying CLWL queue rank

The figure shows how to configure the cluster workload rank attribute (`CLWL RANK`) for a channel in the IBM MQ Explorer channel **Cluster** properties.

CLWL channel rank example



Influencing workload balancing behavior

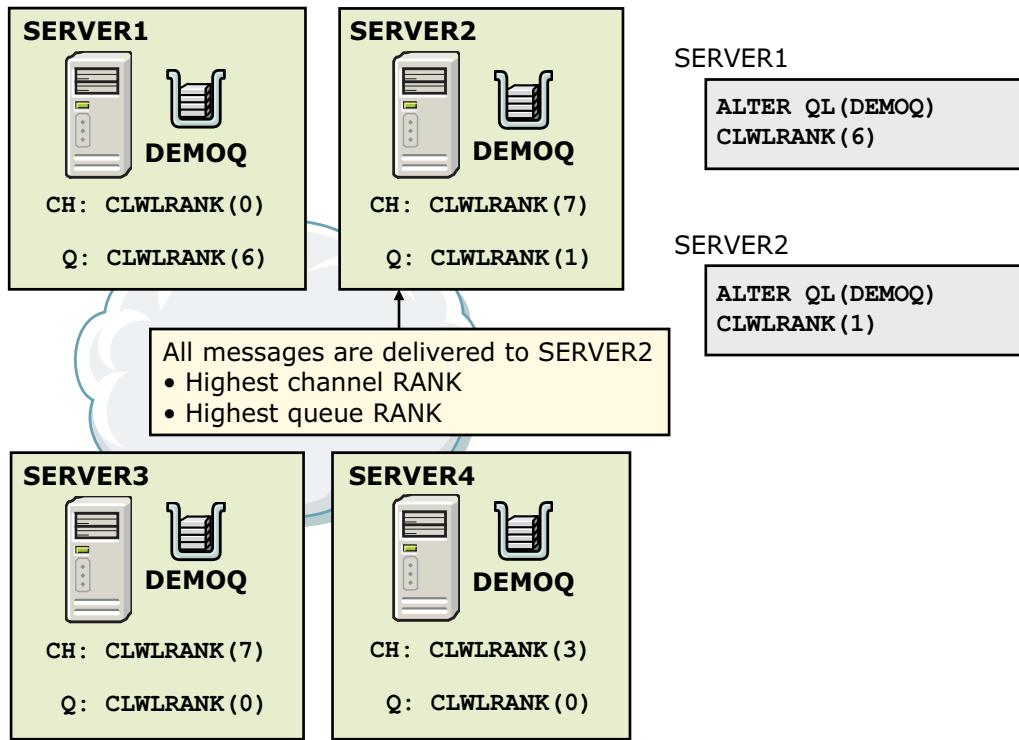
© Copyright IBM Corporation 2017

Figure 6-17. CLWL channel rank example

Initially, the CLWL channel rank is set to 0 for all queue managers. The MQSC modifies the CLWL channel ranks for SERVER2, SERVER3, and SERVER4. The channel rank for SERVER2 and SERVER3 is set to a value that is greater than the value for SERVER4.

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).

CLWL queue rank example



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-18. CLWL queue rank example

CLWL rank can be applied to channels and queues. 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. As channel rank is more powerful than queue rank, the most highly ranked queue (on SERVER1) is not chosen.



Note

The Queue Rank is shown below the Channel Rank. For example, on SERVER1 the CLWLrank for the channel is 0, and for the queue it is 6.

NETPRTY channel attribute

- Specifies the priority for a cluster-receiver channel in the range 0 – 9
- Use to make one network the primary network, and another network the backup network
- Given a set of equally ranked channels, clustering chooses the path with the highest priority when multiple paths are available



Specify this attribute on the cluster-receiver channel at the target queue manager

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-19. NETPRTY channel attribute

The `NETPRTY` channel attribute specifies the priority for a cluster-receiver channel. The value must be in the range 0 – 9, where 0 is the lowest priority and 9 is the highest.

Use the `NETPRTY` attribute to make one network the primary network, and another network the backup network. Given a set of equally ranked channels, clustering chooses the path with the highest priority when multiple paths are available.

A typical example of using the `NETPRTY` channel attribute is to differentiate between networks that have different costs or speeds and connect the same destinations.



Note

Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing that you specify on the matching cluster-sender channel is likely to be ignored.

CLWL 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( )
```

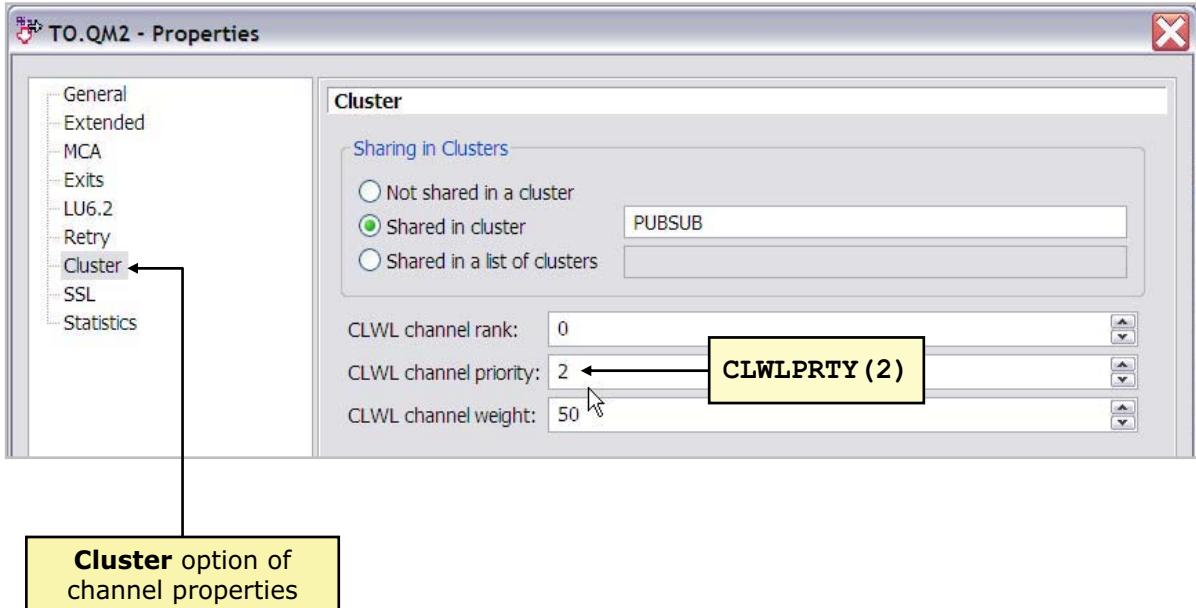
Figure 6-20. CLWL priority basics

Use the CLWLPRTY queue manager attribute to ensure that IBM MQ selects a specific destination queue manager in preference to others with a lower priority. IBM MQ selects the destinations with the highest priority.

Messages go to the highest priority queue manager until it becomes unavailable, and then they go to the next priority queue manager. IBM MQ obtains the priority of queue managers after checking channel status. Only accessible queue managers are available for selection, and it allows IBM MQ to prioritize, where multiple destinations are available.

Where two destinations are possible, you can use this attribute to allow one queue manager to act as a “failover,” when the other queue manager becomes unavailable.

Specifying CLWL channel priority



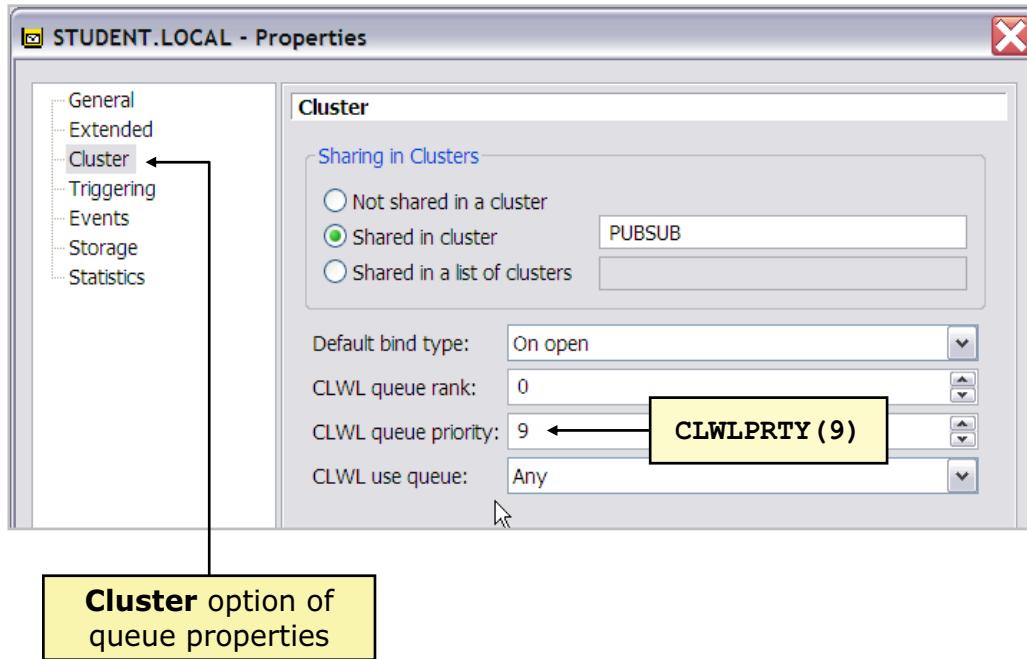
Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-21. Specifying CLWL channel priority

The figure shows how to configure the cluster workload priority attribute (`CLWLPRTY`) for a channel in the IBM MQ Explorer channel **Cluster** properties.

Specifying CLWL queue priority



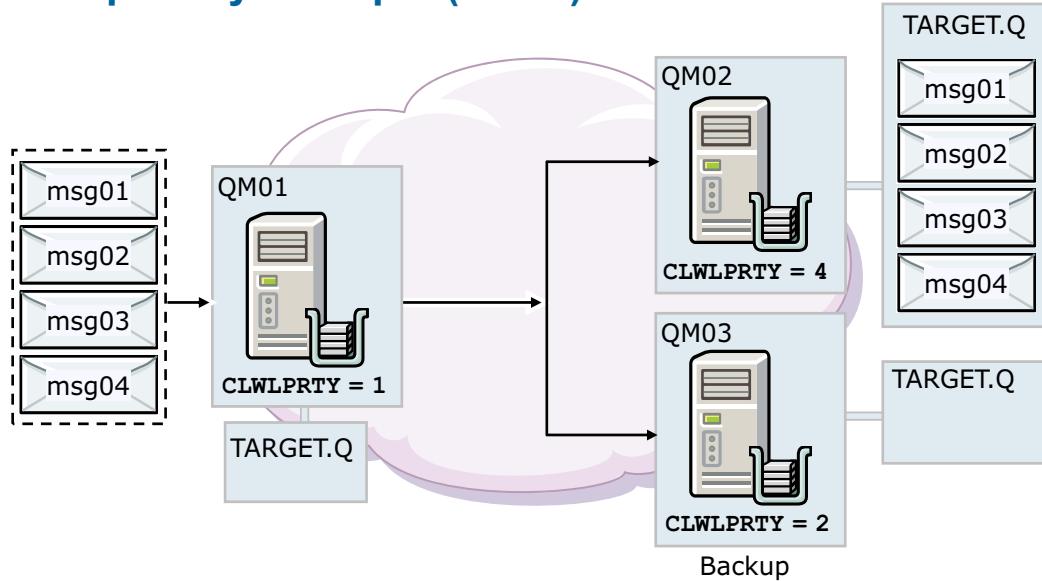
Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-22. Specifying CLWL queue priority

The figure shows how to configure the cluster workload priority attribute (CLWLPRTY) for a queue in the IBM MQ Explorer queue **Cluster** properties.

CLWL priority example (1 of 2)



- CLWLPRTY cluster channel priority selects priority destination (workload management)
 - Select queue manager with active channel status

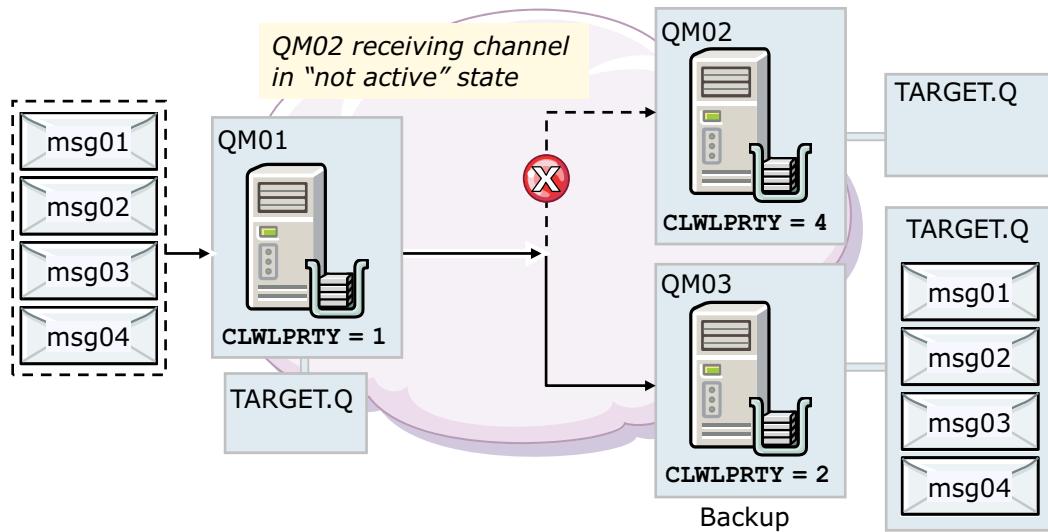
Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-23. CLWL priority example (1 of 2)

The figure shows three queue managers. QM01 has CLWPRTY set to 1, QM02 has CLWPRTY set to 4, and QM03 has CLWPRTY set to 2. Assuming that QM02 is available, all the messages are sent to QM02 because it has the highest priority.

CLWL priority example (2 of 2)



- IBM MQ determines the next highest queue manager priority available

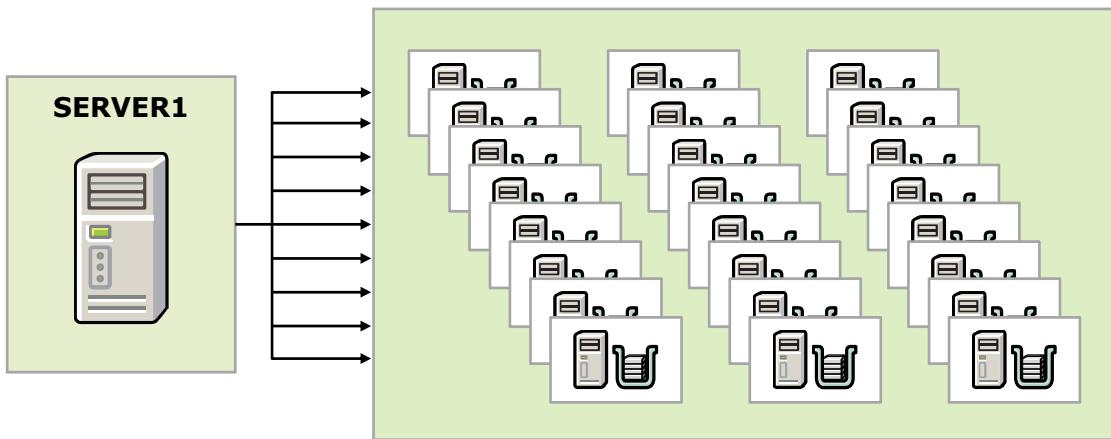
Figure 6-24. CLWL priority example (2 of 2)

The figure shows the same cluster as the previous figure, except now, the channel to QM02 is not active.

If the QM02 cluster-receiver channel is not active, IBM MQ sends the messages to another queue manager based on the CLWLPRTY parameter priority scheme. In this example, the messages are sent to QM03 because the cluster-receiver channel in QM02 is not active.

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 that it has hundreds of outbound channels
 - Active channel limits can limit cluster size



- **Solution:** Use the most recently used channels attribute (CLWLMRUC)

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-25. 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 a specified number of most recently used channel destinations
- MQSC: **ALTER QMGR CLWLMRUC()**
 - Range: 1 – 999999999
 - Default: 999999999
- Example:

From the default CLWLMRUC(999999999), reduce the MRUC value to restrict the number of destinations in the round robin

```
ALTER QMGR CLWLMRUC(2)
```

Figure 6-26. Most recently used channels basics

The `CLWLMRUC` queue manager attribute specifies the maximum number of most recently used cluster channels that are considered for use by the cluster workload choice algorithm. The `CLWLMRUC` attribute is a value of 1 – 999999999.

`CLWLMRUC` is a local queue manager attribute that is not propagated through the cluster. It is made available to cluster workload exits and the cluster workload algorithm that chooses the destination for messages.

If the `CLWLMRUC` attribute is set to 2 on a queue manager, messages are delivered to only two of the four destinations.

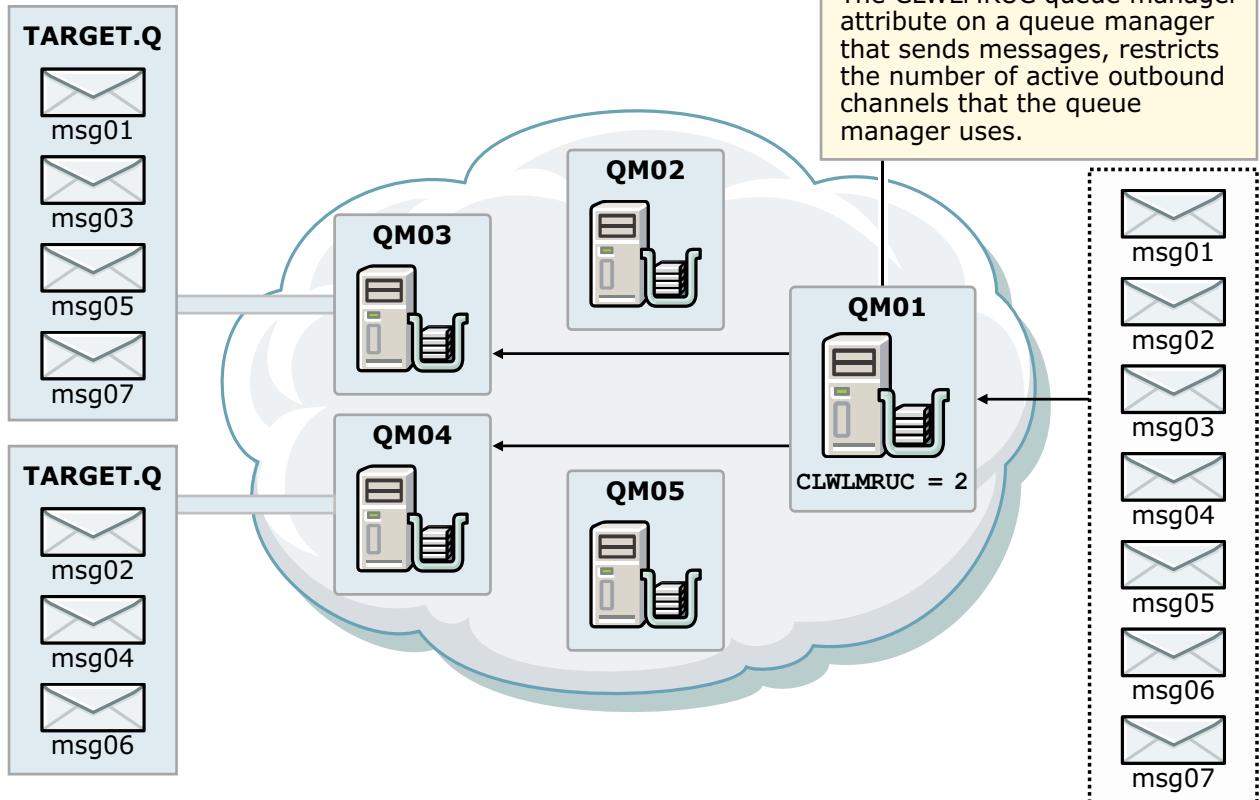


Note

The MRU channels cannot be reliably identified externally, although this information is available to cluster workload exits.

When using MRU, the number of active channels a server can run does not limit the cluster size.

CLWL most recently used channel



Influencing workload balancing behavior

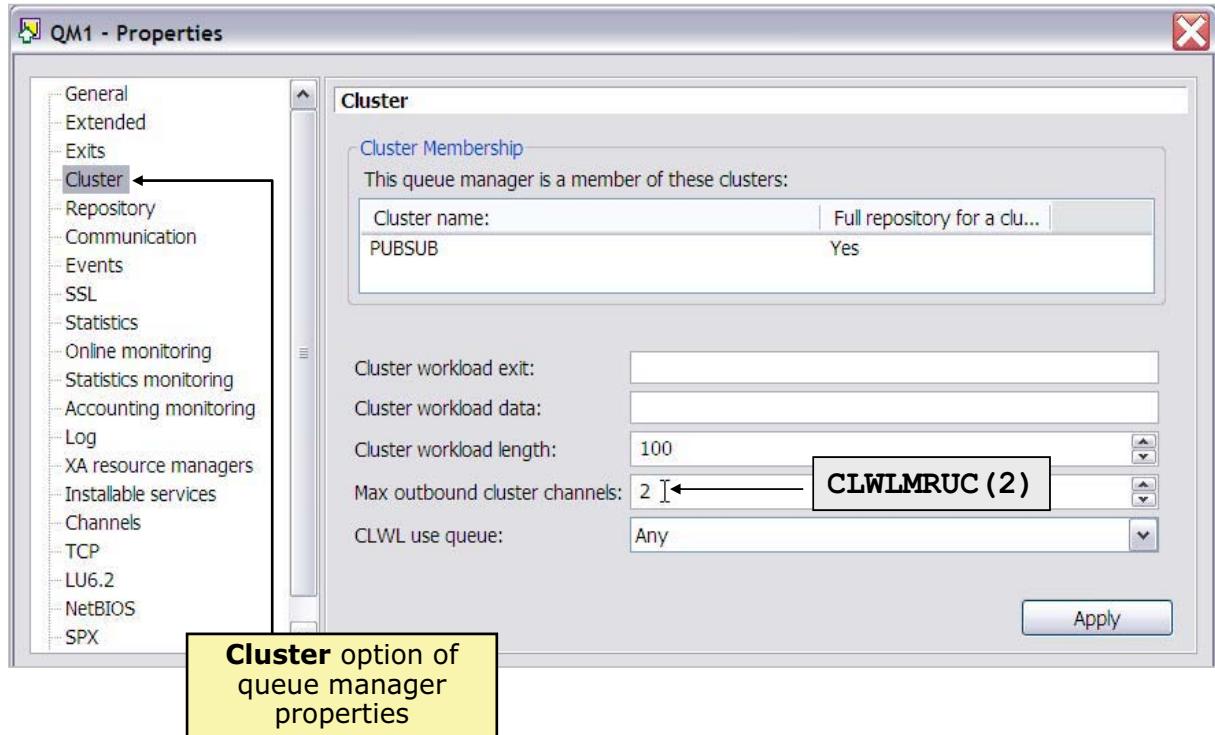
© Copyright IBM Corporation 2017

Figure 6-27. CLWL most recently used channel

In this example, `CLWLMRUC` is set to 2 on the gateway queue manager. As a result, messages are sent to QM03 and QM04.

The `CLWLMRUC` attribute is used by the local queue manager only. This attribute can also be useful to ensure that a cluster channel does not take up all the possible maximum active channels, thus leaving some channels for other cluster channels.

Specifying CLWL most recently used channel



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-28. Specifying CLWL most recently used channel

The figure shows how to configure the `CLWLMRUC` attribute for a queue manager in the IBM MQ Explorer queue manager **Cluster** properties.

CLWL 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
- Specify on cluster-receiver channel

Example:

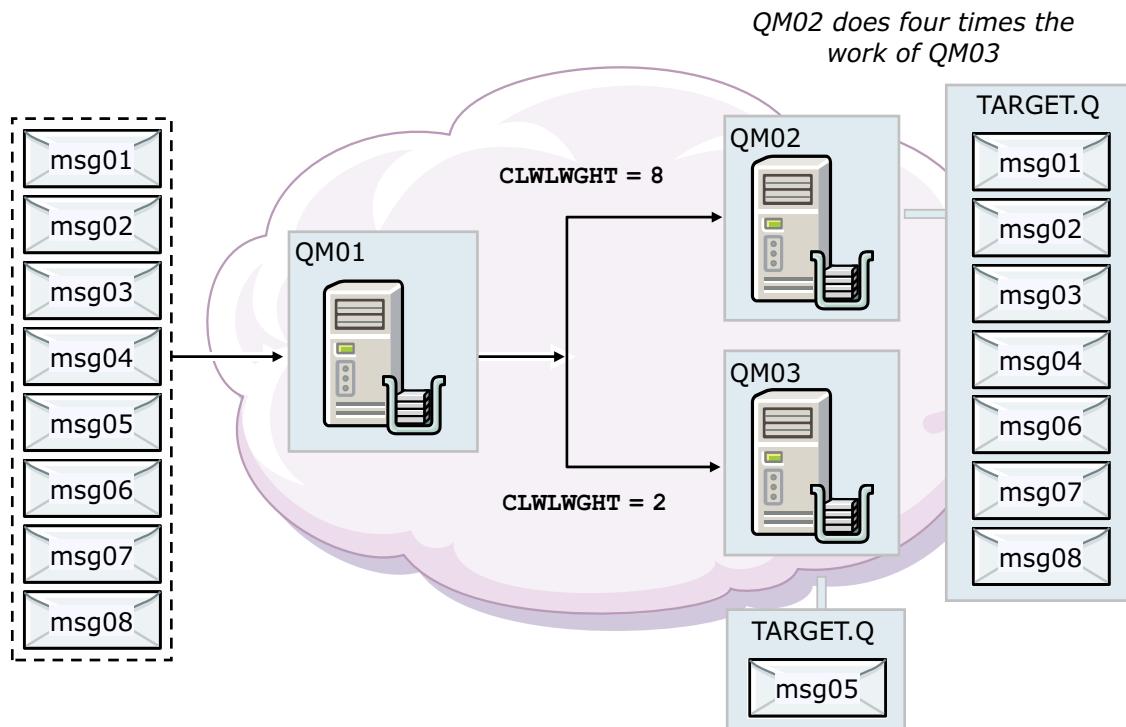
```
DEFINE CHL(CLUS1.SERVER1) CHLTYPE(CLUSRCVR) CLUSTER(CLUS1) ...
CLWLWGHT( )
```

Figure 6-29. CLWL 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.

To apply a weight to a channel for workload management purposes, use the CLWLWGHT attribute on a cluster-receiver channel so that the proportion of messages that are sent down the channel can be controlled. The value must be in the range 1 – 99 where 1 is the lowest rank and 99 is the highest.

Clustering weighted round-robin methodology



Influencing workload balancing behavior

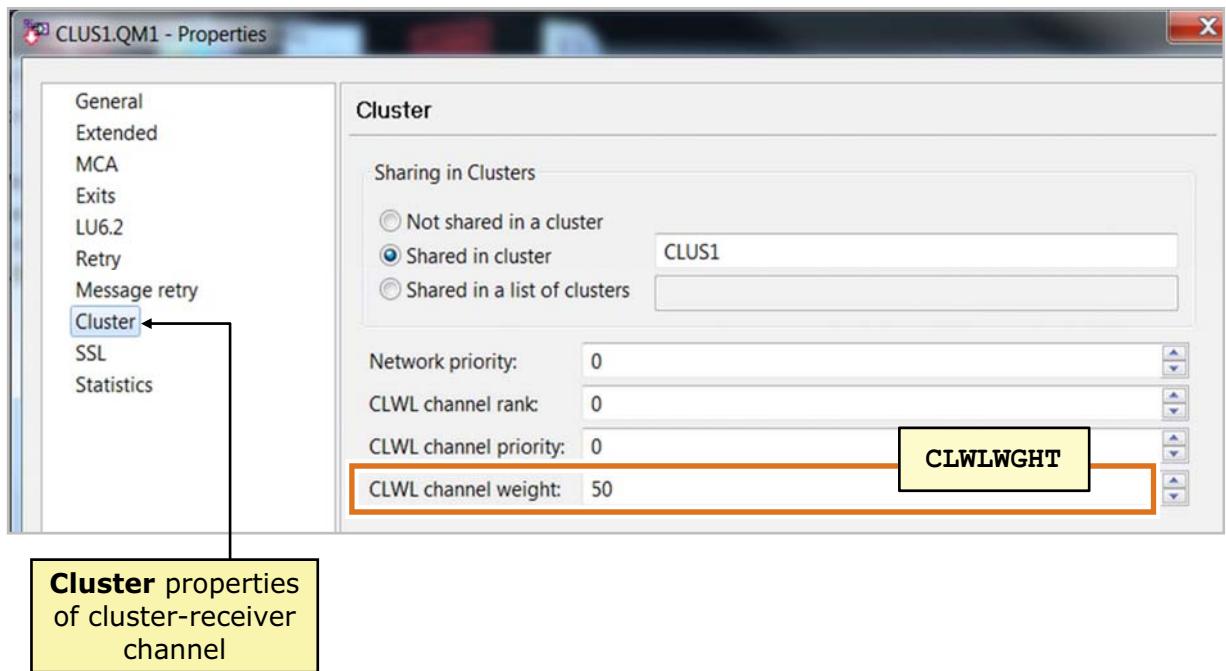
© Copyright IBM Corporation 2017

Figure 6-30. Clustering weighted round-robin methodology

Use the CLWLWGHT attribute on cluster-receiver channels to ensure that servers with more processing power are sent more messages. The higher the channel weight, the more messages are sent over that channel.

In the example, the cluster-receiver channel on QM02 has a CLWLWGHT of 8, and the cluster-receiver channel on QM03 has a CLWLWGHT of 2. Based on those settings, QM02 processes four times more messages than QM03. So for every four messages that are sent to QM02, one message is sent to QM03.

Specifying CLWL channel weight



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-31. Specifying CLWL channel weight

The figure shows how to configure the CLWLWGHT attribute for a cluster-receiver channel in the IBM MQ Explorer channel **Cluster** properties.

Cluster workload management attributes

QUEUES	QUEUE MANAGERS	CHANNELS
CLWLPRTY	CLWLUSEQ	CLWLPRTY
CLWLRANK	CLWLMRUC	CLWLRANK
CLWLUSEQ		CLWLWGHT

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-32. Cluster workload management attributes

This figure lists the attributes that are used when determining 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 of 2)

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 (CLWLUSEQ)

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-33. Cluster workload management algorithm: Summary (1 of 2)

Every time a message is put on a queue, the cluster workload management algorithm runs one time.

This figure summarizes the workload management algorithm. For a detailed description of every step in the algorithm, see “The cluster workload management algorithm” topic in the IBM Knowledge Center.

Cluster workload management algorithm: Summary (2 of 2)

5. Evaluate channel rank (CLWLRANK)
6. Evaluate channel status
7. Evaluate channel net priority (NETPRTY)
8. Evaluate channel priority (CLWLPRTY)
9. Evaluate queue priority (CLWLPRTY)
10. Evaluate most recently used (CLWLMRUC)
11. Evaluate least recently used with channel weight (CLWLWGHT)

Cluster Queue Monitoring sample program (`amqsclm`)

- Uses the IBM MQ cluster workload balancing features to direct messages to instances of queues with active applications
 - Program runs against each queue manager in the cluster that hosts queues, monitors the queues, and reroutes messages as necessary
 - Provided as source (`amqsclm.csmp`le) to allow the user to understand the mechanics of the tool and customize where needed

- Moves queued messages from instances of the queue where no consumers are attached to instances of the queue with consumers
 - Prevents the buildup of messages on an instance of a cluster queue to which no active application is attached
 - Allows all messages to be processed effectively
 - Provides more versatility in the processing of messages

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-35. Cluster Queue Monitoring sample program (`amqsclm`)

The Cluster Queue Monitoring sample program (`amqsclm`), ensures that messages are directed to the instances of clustered queues with active applications. This sample program allows all messages to be processed effectively even when a system is asymmetrical, such as when applications are not attached to all queue managers.

The Cluster Queue Monitoring sample program also moves queued messages from instances of the queue where no applications are attached to instances of the queue with active applications.

The Cluster Queue Monitoring sample program allows for more versatility in the use of clustered queue topologies where applications are not under the direct control of the queue managers. It also gives a greater degree of high availability in the processing of messages.

The Cluster Queue Monitoring sample program provides a monitoring program that can be run against each queue manager in the cluster. It monitors the queues and reroutes the messages as necessary.

The source (`amqsclm.csmp`le) for the Cluster Queue Monitoring sample program is available to understand the mechanics of the tool and customize it if necessary.

Cluster Queue Monitoring sample program logic

- Single program that is set to run against each queue manager and monitor one or more cluster queues
- Uses cluster workload priority attribute (CLWLPRTY) of queues to determine target queue
- Monitoring process polls the state of the queues on a defined interval
- If no applications are attached:
 - CLWLPRTY of the queue is set to zero (if not already set)
 - Cluster workload balancing reroutes the messages to active instances of the queue in the cluster
- If applications are attached, CLWLPRTY of the queue is set to one (if not already set)
- Define `amqsclm` as a queue manager service to ensure that it starts with the queue manager

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-36. Cluster Queue Monitoring sample program logic

The Cluster Queue Monitoring sample program uses IBM MQ cluster workload balancing mechanics. It uses the cluster workload priority of individual queues (CLWLPRTY) to send messages to instances of queues with the highest cluster priority.

The Cluster Queue Monitoring sample program polls the state of the queues on a defined interval.

Define the Cluster Queue Monitoring sample program as a queue manager service to ensure that it is started with each queue manager.

For more information about the Cluster Queue Monitoring sample program, see the IBM Knowledge Center.

Message affinities

- Message affinities (DEFBIND) inhibit scalability, workload balancing, and continuous operations

- To remove affinities, consider the following possibilities:
 - Carrying state information in the messages
 - Maintaining state information in nonvolatile storage accessible to any queue manager
 - Replicating read-only data so that it is accessible to more than one queue manager

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-37. Message affinities

Some applications are written to expect a sequence of related messages and are unable to process individual messages received out of the expected sequence. Applications that are designed by using this method have *message affinities*.

Before starting to use a cluster with multiple definitions of the same queue, examine your applications to see whether any of them have message affinities. The design goal is to have no affinities in new applications.

Message affinities inhibit scalability, workload balancing, continuous operations.

- Scalability: Only one queue instance can be used for the whole sequence of messages.
- Workload balancing: The algorithm is started only one time for the whole sequence of messages, selecting one queue instance for all messages instead of distributing messages between all queue instances.
- Continuous operations: After one message of the related messages chooses a destination, the rest of the messages cannot be rerouted.

Other reasons why messages affinities are bad (independently of clustering) are that they imply that some resource, such as storage, is held in a server for a long time. Message affinities also tend to mean that resources such as rows in a database are locked.

Handling message affinities

- If you cannot remove message affinities, you can force related messages to be delivered by using the same channel and to the same queue manager
 - Set the MQOO_BIND_ON_OPEN or MQOO_BIND_ON_GROUP option on the MQOPEN call
 - Name a specific destination on the MQOPEN call
 - Return the queue manager name in the reply-to queue manager field
 - Write a customized cluster workload exit program

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-38. Handling message affinities

Consider the changes that are listed to remove message affinities from an application:

- Carry state information in the messages
- Maintain state information in nonvolatile storage accessible to any queue manager, for example in a DB2 database
- Replicating read-only data so that it is accessible to more than one queue manager

If it is not appropriate to modify the applications to remove message affinities, the figure lists some other possible solutions to the problem. For more information about the advantages and disadvantages of each solution, see the “Handling message affinities” topic in the IBM Knowledge Center.

Workload balancing BIND 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 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 not fixed
 - Each message is bound to a destination at MQPUT
 - Workload balancing is done on every message
 - Do not create affinities
- Bind as queue definition
 - Queue specifies the binding option instead of application

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-39. Workload balancing BIND options

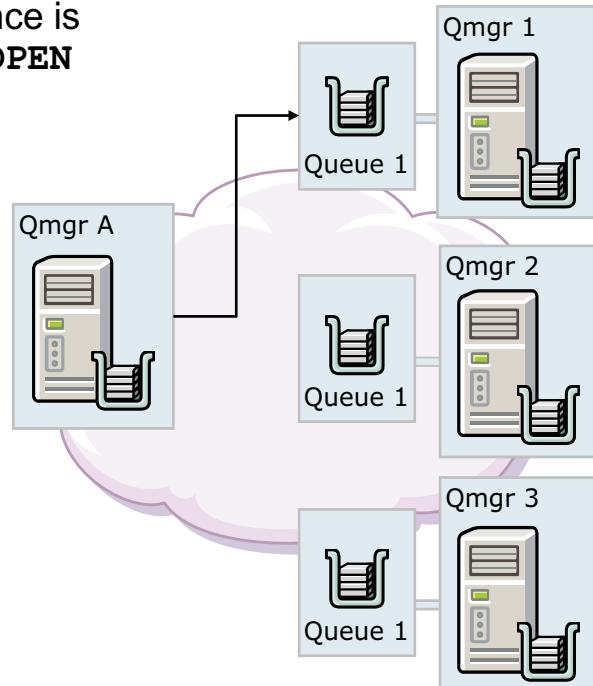
An option on the MQOPEN call allows the application to specify that the target queue manager needs to be fixed when multiple instances of the same queue are within a cluster. That is, all messages that are put to the queue that specifies the object handle that is returned from the MQOPEN call must be directed to the same queue manager, by using the same route.

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.

BIND_ON_OPEN

- Only one queue instance is selected for each `MQOPEN`



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

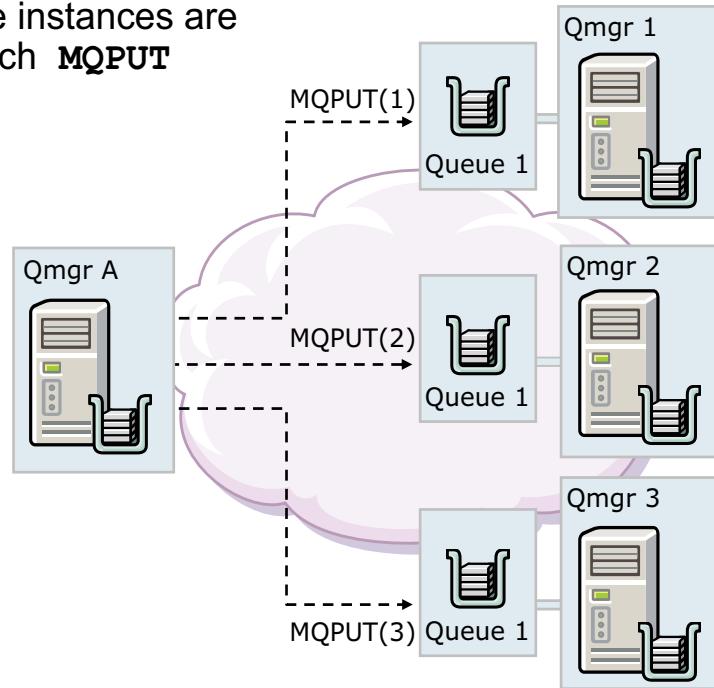
Figure 6-40. `BIND_ON_OPEN`

Applications with `BIND_ON_OPEN` message affinities must ensure that all messages are delivered in sequence to the same queue manager.

To ensure the delivery sequence, an application instance must open the queue with the `MQOO_BIND_ON_OPEN` on the `MQOPEN` call. Alternatively, an application can use the `MQOO_BIND_AS_Q_DEF` on `MQOPEN` with the `DEFBIND(OPEN)` attribute on the queue definition.

BIND_NOT_FIXED

- Different queue instances are selected for each MQPUT



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-41. `BIND_NOT_FIXED`

Using `MQOO_BIND_NOT_FIXED` increases performance, continuous operations, and scalability because the cluster can work with multiple destination queues.

`BIND_NOT_FIXED` stops the local queue manager from binding the queue handle to an instance of the destination queue. As a result, successive `MQPUT` calls that use this handle send the messages to different instances of the destination queue, or to the same instance but by different routes. It also allows the instance that is selected to be changed later by the local queue manager, by a remote queue manager, or by a message channel agent (MCA), according to network conditions.

Customizing workload balancing

- Supplied workload management algorithm
 - Round robin is always used on MQOPEN
 - Round robin is possible on each MQPUT
- Other requirements require user exit
 - Cheap network routes
 - Message content
- Exit controls where messages directed
 - If no exit is specified, the supplied workload management exit is used

Influencing workload balancing behavior

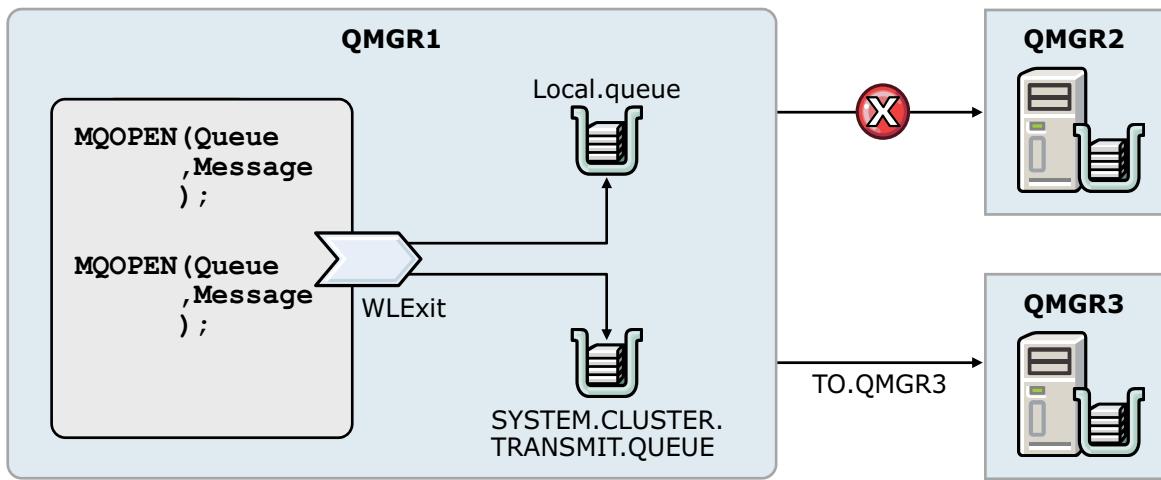
© Copyright IBM Corporation 2017

Figure 6-42. Customizing workload balancing

The default behavior of the workload balancing algorithm always uses round-robin. In cases where the destination queue is opened to allow for distribution among all available queue instances, the round-robin algorithm is started at MQOPEN, and also at every MQPUT. If a queue is opened to accommodate message affinities (only one queue instance is used for all messages), then it is used at MQOPEN only.

However, round-robin might not be required for some reason. If so, it is necessary to override that behavior with a user-written cluster workload exit.

Cluster workload exits



- Called when accessing a cluster queue on MQOPEN, MQPUT1, and MQPUT
- If the target queue manager is not fixed at MQOPEN time, the target queue manager is chosen at the time of the MQPUT call
- On operating systems other than z/OS, the queue manager loads the new cluster workload exit the next time the queue manager is started

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-43. Cluster workload exits

A user-written cluster workload exit is started in the same manner as the supplied workload management algorithm. That is, it is called when an MQOPEN (or an MQPUT1) occurs and again at each MQPUT. However, if the queue was opened to direct all messages to a single instance of the queue, then the exit is not called at MQPUT even if the destination queue manager fails.

If the messages are being distributed to multiple instances of the queue and a destination queue manager fails or becomes unavailable, the exit is called again to select a new target queue manager.

Writing cluster workload exits

- No MQI calls allowed
- Similar to channel exits
- Special considerations on z/OS
- Name cluster workload exits in the queue manager definition
 - Example: **ALTER QMGR CLWLEXIT (myexit)**
- Indicate queue manager for messages on the queue manager definition
 - Example: **ALTER QMGR CLWLADATA ('cluster-name.queue-manager')**
- Sample exits provided with IBM MQ

Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-44. Writing cluster workload exits

Programmers need to know about some basic restrictions and requirements. z/OS has many more restrictions. It is important that programmers review the restrictions and understand what they mean before writing the exit.

Restrictions and requirements are described in greater detail in the product documentation.

The exit name is specified as an attribute of the queue manager definition:

```
ALTER QMGR CLWLEXIT(exitname)
```

The sample cluster workload exit that IBM MQ supplies is called `amqswlm0.c` on distributed operating systems. On z/OS, the sample is supplied in both C (CSQ4BCF1) and in assembly language (CSQ4BAF1).

Unit summary

- Describe the factors to consider before altering base workload management
- Explain when cluster workload management takes place
- Summarize the cluster workload management algorithm
- Explain how to identify and remove message affinities that negatively impact load balancing
- Describe how to use various queue attributes to influence load balancing
- Describe how to use various channel attributes to influence load balancing
- Describe how to use the queue manager properties to influence load balancing
- Describe when and how to use a customized workload balancing exit

Review questions

1. True or False: The most recently used attribute can be specified on both a queue manager and a channel definition.
2. 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.



Influencing workload balancing behavior

© Copyright IBM Corporation 2017

Figure 6-46. Review questions

Review answers

- 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 cluster workload most recently used attribute is specified on the queue manager only.



- 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.

Exercise: Working with workload balancing options

© Copyright IBM Corporation 2017

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

Figure 6-48. Exercise: Working with workload balancing options

Exercise objectives

After completing this exercise, you should be able to:

- Send messages to cluster queues by using round-robin distribution
- Use channel priority to influence the cluster workload
- Use channel weights to influence the cluster workload
- Use the queue manager to influence the cluster workload

Unit 7. Publish/subscribe and clusters

Estimated time

01:00

Overview

This unit teaches you how to work with the publish/subscribe messaging style. After an introduction to the history of publish/subscribe, you learn key terminology that refers to the co-existing publish/subscribe capabilities. The unit continues with a description of the publish/subscribe components, and concludes with an explanation of the options that can be implemented for a clustered publish/subscribe environment.

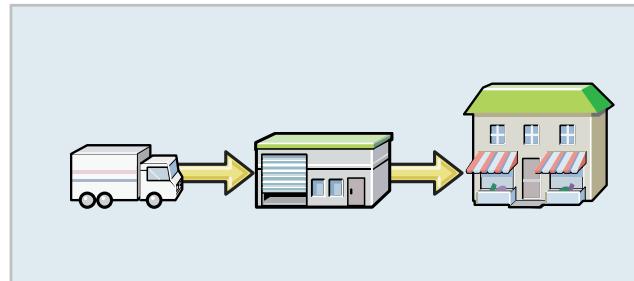
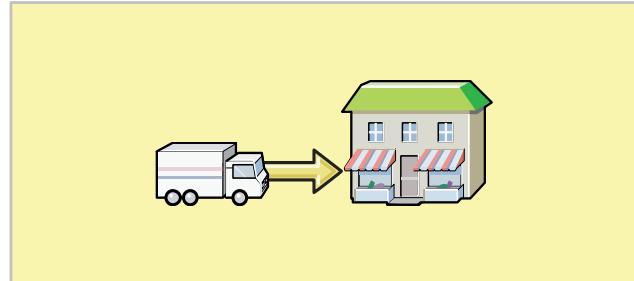
Unit objectives

- Differentiate between publish/subscribe and point-to-point messaging
- Summarize how the history of publish/subscribe influences its functions and terminology
- Identify the basic components of publish/subscribe
- Describe key properties of topics, subscriptions, and publications
- Describe the publish/subscribe cluster topologies
- Explain how to configure a direct routing publish/subscribe cluster
- Explain how to configure a topic host routing publish/subscribe cluster

Point-to-point and publish/subscribe

- Point-to-point application
 - Sends messages to a predefined destination
 - Application does not need to know where the destination is
 - WebSphere MQ locates the target through object definitions

- Publish/subscribe application
 - Publishes messages to an interim destination according to a topic
 - Interested recipients subscribe to the topic
 - No explicit connection between publishing and subscribing applications



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-2. Point-to-point and publish/subscribe

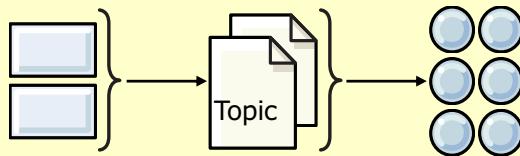
With publish/subscribe, the sending or publishing of messages and the subscription or getting of messages is done by using the topic tree as an intermediary. It is not apparent where the message is going, or how it shows up, as the publish/subscribe process handles those details.

Publish/subscribe uses distributed and cluster channels, but the intermediary publish/subscribe process determines the source and destination of the messages. Publish/subscribe adds another level of application decoupling.

Distributed publish/subscribe patterns

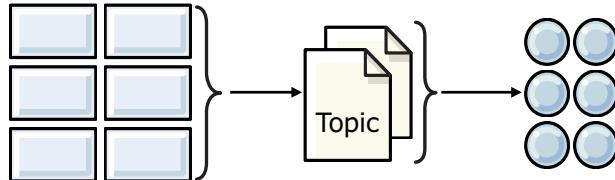
Few-to-many:

- Research
- News tickers



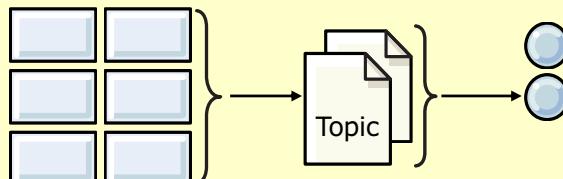
Many-to-many:

- Prices
- Quotations



Many-to-few:

- Orders
- Check inventory



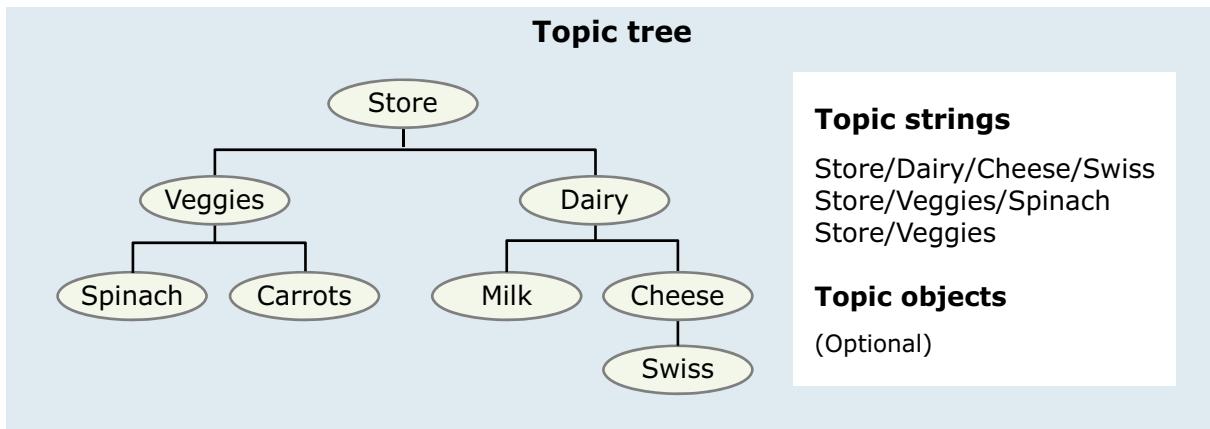
Key: Publisher Subscriber

Figure 7-3. Distributed publish/subscribe patterns

One of the reasons that publish/subscribe gained popularity was the flexibility of configurations that are possible to achieve. The publish/subscribe patterns apply to stocks, sports, sales, and other popular applications.

Forms of publish/subscribe are even embedded in software products such as IBM Integration Bus, formerly WebSphere Message Broker, IBM MQ Managed File Transfer, and in telemetry mobile applications.

Publish/subscribe basic components



Publishing application

Use:
MQOPEN
 topic Store/Veggies/Spinach
MQPUT

- Point-to-point opens a queue
- Publish/subscribe opens a topic

Subscribing application

Use:
MQSUB verb
 to Store/Veggies

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-4. Publish/subscribe basic components

At the center of publish/subscribe is the topic tree. A topic tree is a hierarchical structure that contains and organizes topics. A topic tree's organization is influenced according to how the topics are created. The topic tree might be compared to file system with directories and subdirectories.

Topics can be created by using the create topic commands or by specifying the topic for the first time in a publication or subscription. Topics are structured into topic strings. Topic strings have separate categories that are separated by a slash (/) character.

When messages are sent in a publish/subscribe application, they are published, or put to a topic instead of a queue. A topic can be an explicitly defined object, or a publish or a subscribe operation to the topic string might dynamically create a topic. Published messages are called publications.

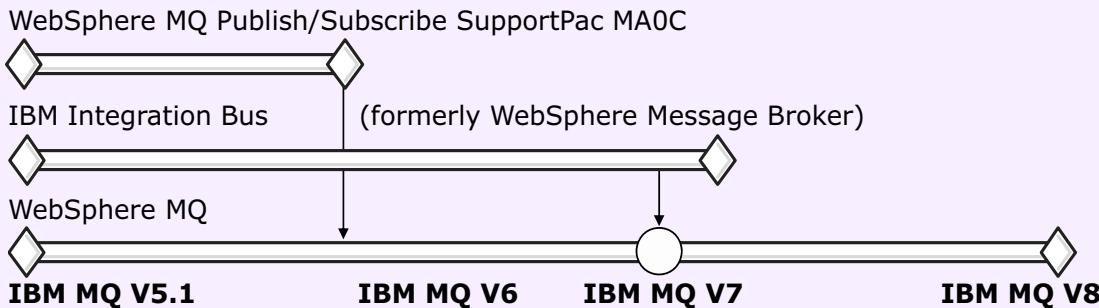
To get the publications, applications subscribe to the topics or topic strings. The subscription can be an explicitly defined object, or a dynamic subscription created by an application.

You have the topic tree, topics, publications and publishers, and subscribers and subscriptions.

To better understand today's publish/subscribe implementation, you look at publish/subscribe over the years.

Current publish/subscribe functionality: A little history

Publish/subscribe brokers



IBM MQ publish/subscribe APIs

Command message-based publish/subscribe API



JMS publish/subscribe API



XMS publish/subscribe API



Native MQI publish/subscribe API



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-5. Current publish/subscribe functionality: A little history

The implementation and terminology of publish/subscribe can be confusing. Queue managers have a “mode” or PSMODE parameter. “Managed queues” and “queued publish/subscribe” are mentioned. Why such terminology?

Publish/subscribe started out as IBM MQ Support pack MA0C. Publishing and subscribing with the original support pack involved the use of the RFH command message-based API to send subscription and publication requests. WebSphere Message Broker also had a separate publish/subscribe mechanism and used WebSphere Message Broker capabilities to do the publish/subscribe actions. Both the support pack and WebSphere Message Broker referred to the publish/subscribe engine as a “broker”.

As publish/subscribe evolved, it was included in the IBM MQ product. The publish/subscribe API was also changed and augmented. This course does not address migration of the “old” publish/subscribe applications. However, it is good to keep in mind that the need to address or handle a migration might surface if you work with an existing application that uses the “old” API and broker function.

A good example of the publish/subscribe heritage is the subscription object PSPROP attribute, which deals with the ways that related message properties are added to messages that are sent to the defined. You look at PSPROP later in this unit.

Publish/subscribe terminology baseline (1 of 2)

- Terms that are used to refer to the original publish/subscribe support pack engine or to the WebSphere Message Broker engine:
 - Broker
 - Queued publish/subscribe, command-message-based publish/subscribe
- Ways the “new” V7+ publish/subscribe is referred to
 - Integrated publish/subscribe
 - Publish/subscribe interface
 - Publish/subscribe MQI
- Managed publish/subscribe
 - Publish/subscribe is used dynamically without creating defined queues or topics
 - “Managed” can also apply to a dynamically created queue or a dynamically created topic

Figure 7-6. Publish/subscribe terminology baseline (1 of 2)

This slide covers some of the terminology that you might see in the IBM Knowledge Center and also notice when the channel initiator comes up.

Publish/subscribe terminology baseline (2 of 2)

- Administered:
 - Creating an explicitly defined object for publish/subscribe such as a topic or a subscription
 - An administered object such as a subscription can have a managed queue
- Streams:
 - Concept that is used in queued publish/subscribe to separate the flows for different topics
 - Default stream is available to V7+ queue managers if queued publish/subscribe is enabled
 - Named streams require a queue definition added to
SYSTEM.QPUBSUB.QUEUE.NAMELIST

Figure 7-7. Publish/subscribe terminology baseline (2 of 2)

This slide covers some of the terminology that you might see in the IBM Knowledge Center.

Publish/subscribe functionality

- **PSMODE** queue manager attribute controls publish/subscribe versions
 - **ENABLED:** The default, which indicates that both the **publish/subscribe engine** and **queued publish/subscribe** are running
 - **COMPAT:** Publish/subscribe engine active, and queued publish/subscribe stopped
 - **DISABLED:** Both publish/subscribe engine and queued publish/subscribe are stopped

```
.....
AMQ5052: The queue manager task 'PUBSUB-DAEMON' has started.
.....
AMQ5975: 'WebSphere MQ Distributed Pub/Sub Controller' has started.
.....
AMQ5975: 'WebSphere MQ Distributed Pub/Sub Fan Out Task' has
started.
.....
AMQ5806: Queued Publish/Subscribe Daemon started for queue manager
MQ01.
.....
```

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-8. Publish/subscribe functionality

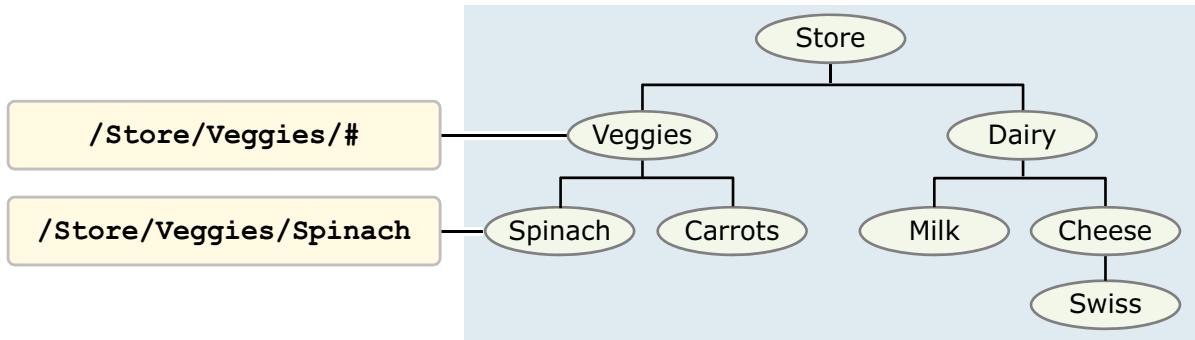
PSMODE is a queue manager attribute across the z/OS and distributed platforms that control the type of publish/subscribe functionality that is enabled in a queue manager.

In the snapshot of the channel initiator start console messages, you can see how the publish/subscribe engine starts, and then later the “old” publish/subscribe, that is the “queued” publish/subscribe process also starts.

The predominant modes are:

- **ENABLED:** If you are not sure what the mix of applications is, then use this mode. It supports the old and new publish/subscribe, within any limitations that are documented in the IBM Knowledge Center.
- **COMPAT:** The “old” publish/subscribe is not active. Only the new engine is started. If you are sure that no old publish/subscribe applications exist, then use this mode.
- **DISABLED:** However, if publish/subscribe is not being used, startup of the publish/subscribe process can be stopped other software components outside the applications, such as WebSphere Message Broker or IBM MQ Managed File Transfer might require queued publish/subscribe to be active. Always confirm what runs in the environment before setting PSMODE to DISABLED.

Topic tree, topic strings, topic nodes, and topic objects



- Topic strings are the common sets of slash structured text that applications publish and subscribe to
- Topic nodes are a segment of the topics
- Topic objects are explicitly defined topics, and are also called administered topics
- The topic tree is a hierarchical structure that holds topic strings and nodes
 - When a topic is published to, subscribed to, or defined as an administered topic object, the topic tree expands to include the new topic



It is **critical** to implement governance and exercise control over the topic tree.

Figure 7-9. Topic tree, topic strings, topic nodes, and topic objects

This slide elaborates on topic-related terminology.

Topic trees can take many “shapes”. While some topic trees are hierarchical, others can be “flat”, for instance, can have more horizontal nodes.

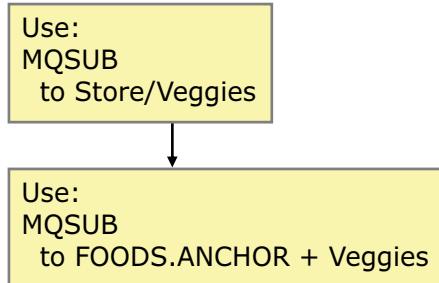
Since topics nodes can be added in so many ways, it is easy for a topic tree to grow larger than expected. When an application publishes or subscribes, it must search the topic tree. Searching a large topic tree, or a topic tree with unused nodes, eventually results in performance impacts to an application.

The term “topic node” is distinct from the term “topic object”, as the topic object is defined as an IBM MQ object with the MQSC command “`DEFINE TOPIC`”.

The next slide shows the advantages of a defined topic object.

Topic objects

- Provide an administrative control point for the topic tree
 - Configuration attributes
 - Security profiles
 - Topic tree isolation
- Provides flexibility if the topic tree changes later
- Defined topic objects for publish/subscribe are optional



```

TOPIC(FOODS.ANCHOR)
TOPICSTR('Store')
CLUSTER( )
DURSUB(ASPARENT)
DEFPSIST(ASPARENT)
...
  
```

```

TOPIC(FOODS.ANCHOR)
TOPICSTR('State/County/City/Store')
CLUSTER( )
DURSUB(ASPARENT)
DEFPSIST(ASPARENT)
...
  
```

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-10. Topic objects

The ability to dynamically add a topic provides significant flexibility to an organization, but topic objects provide extra flexibility.

Imagine that an organization started locally with one store. This store was successful and grew to a national entity. What happens to all applications that published or subscribed to the original store?

By having an explicitly defined topic object as an anchor topic, an application can use the anchor topic with other nodes in the topic string, and automatically be part of the larger scope. If the applications used the FOODS.ANCHOR topic to start, do not be concerned if extra nodes must be added ahead of the original hierarchy.

The defined topic also helps as a point to establish “parental” attributes and apply security profiles.

Topic object attributes: DISPLAY TOPIC view

- **ASPARENT** values
 - Taken from the first parent administrative node that is found in the topic tree
 - If no administrative nodes are found, values are taken from SYSTEM.BASE.TOPIC
- **DEFPSIST** is persistence that is used when the application uses the **MQPER_PERSISTENCE_AS_TOPIC_DEF** option

<pre>DIS TOPIC(FOODS*) ALL TOPIC (FOODS.ANCHOR) TYPE (LOCAL) QSGDISP (QMGR) CLUSTER () TOPICSTR (Store) DEFPRTY (ASPARENT) DEFPSIST (ASPARENT) DURSUB (ASPARENT) NPMSGDLV (ASPARENT) PMSGDLV (ASPARENT) MDURMDL ()</pre>	<pre>PUB (ASPARENT) SUB (ASPARENT) PUBSCOPE (ALL) SUBSCOPE (ALL) PROXYSUB (FIRSTUSE) WILDCARD (PASSTHRU) CLROUTE (DIRECT) DESCR () DEFPRESP (ASPARENT) USEDLQ (ASPARENT) CUSTOM () ALTDATE (2014-09-15)</pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-11. Topic object attributes: DISPLAY TOPIC view

This slide shows a topic object display and introduces the concept of inheritance.

At the base, or starting topic of the topic tree is a topic object called SYSTEM.BASE.TOPIC. If topic objects are defined to use default attributes, and no other topic objects are between the topic string and the SYSTEM.BASE.TOPIC that change the attributes, the object attributes are inherited from the SYSTEM.BASE.TOPIC object.

When a topic is displayed with the “DIS TOPIC” command, you see the value **ASPARENT** in many of the attributes. This value indicates that the topic object definition takes its attributes from the next higher topic object definition.

The default values for topic attributes are listed in the IBM Knowledge Center.

Topic object attributes: DISPLAY TPSTATUS view

- **TPSTATUS** shows some actual values in place of **ASPARENT**

```

TPSTATUS( Store/Veggies/Spinach )
TYPE(TOPIC)
DEFPRESP(SYNC)
DEFPSIST(NO)
DEFPRTY(0)
DURSUB(YES)
PUB(ENABLED)
SUB(ENABLED)
ADMIN( )
MDURMDL(SYSTEM.DURABLE.MODEL.QUEUE)
MNDURMDL(SYSTEM.NDURABLE.MODEL.QUEUE)
...

```

```

...
NPMSGDLV(ALLAVAIL)
PMSGDLV(ALLDUR)
RETAINED(YES)
PUBCOUNT(0)
SUBCOUNT(2)
PUBSCOPE(ALL)
SUBSCOPE(ALL)
CLUSTER( )
USEDLQ(YES)
CLROUTE(NONE)

```

Figure 7-12. Topic object attributes: DISPLAY TPSTATUS view

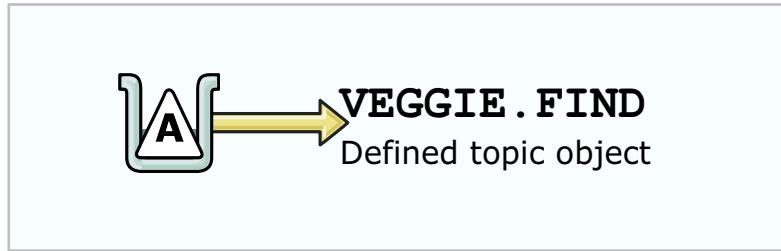
To check the status of a topic, use the **DISPLAY TPSTATUS** command. When displaying the status, the actual inherited attributes, rather than “**ASPARENT**”, are shown on the display.

If you need to match all topic status, you use the number sign ('#') instead of the asterisk, for example:

```
DIS TPSTATUS( '#' ) ALL
```

Remember that if the topic tree is large, the command might cause quite a bit of searching.

Topic alias



- Alias queue that points to an explicitly defined topic object
- Topic object must be defined in same queue manager as the alias queue
- Set TARGTYPE attribute to TOPIC in the QALIAS definition

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-13. Topic alias

If an application currently puts messages onto a queue, it can be made to publish to a topic by making the queue name an alias for the topic.

Topic administration options

- Administer topics with MQSC commands and IBM MQ Explorer
- **DEFINE, ALTER, DISPLAY, DELETE, and DISPLAY TPSTATUS**
- Wildcard character for topic status is “#”: **DIS TPSTATUS (#)**

```
MQ11 DIS PUBSUB ALL . . .
TYPE (LOCAL)
QMNAME (MQ11)
STATUS (ACTIVE)
SUBCOUNT (8)
TPCOUNT (11)
```

```
DIS TPSTATUS (#) SUBCOUNT WHERE (SUBCOUNT GT 0)
```

```
TPSTATUS (SYSTEM.BROKER.ADMIN.STREAM/MQ/MQ11 /StreamSupport )
TYPE (TOPIC) SUBCOUNT (1)

TPSTATUS (Store/Veggies)
TYPE (TOPIC) SUBCOUNT (2)

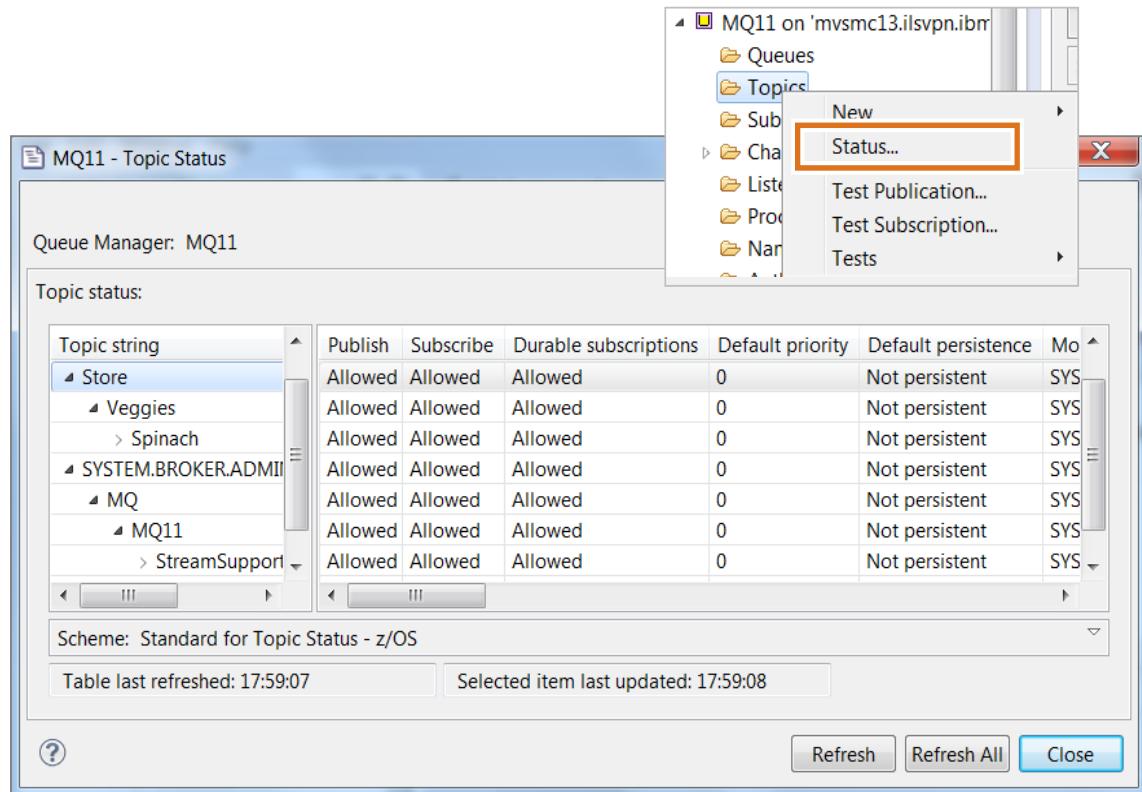
TPSTATUS (Store/Veggies/Spinach)
TYPE (TOPIC) SUBCOUNT (2)
```

Figure 7-14. Topic administration options

To administer topics as subscriptions, MQSC commands can be used with the `rwmqsc` tool. IBM MQ Explorer can also be used to administer publish/subscribe objects.

IBM Training

Topic status with IBM MQ Explorer



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-15. Topic status with IBM MQ Explorer

IBM MQ Explorer provides capabilities to create topics, display the status, and also offers the capability to test a publication and subscription.

The topic string hierarchy can be seen in the IBM MQ Explorer display.

When using the test panes, the Test Subscription should be started before the Test Publication.

Subscriptions: The three aspects (1 of 2)

- Subscriptions link topics to queues
- Three aspects to subscriptions:
 1. Subscription creation and deletion
 2. Subscription queue management
 3. Subscription lifetime
- Subscription creation and deletion
 - Application-created subscription uses **MQSUB** to create and **MQCLOSE** to delete
 - Administrative or explicitly defined subscription object
- Subscription queue management
 - Managed where the subscription creates and deletes the queue: **DESTCLASS MANAGED**
 - Unmanaged where a queue is explicitly defined or provided: **DESTCLASS PROVIDED**

SUB (VEGETARIAN.SEARCH)

DURABLE (YES)

SUBTYPE (ADMIN)

TOPICSTR (Store/Veggies)

DEST (SYSTEM.MANAGED.DURABLE.**CDC 456AF9B060E32)**

TOPICOBJ (FOODS.ANCHOR)

DESTCLAS (MANAGED)

EXPIRY (UNLIMITED)

SUB (VEGGIE.FIND)

DURABLE (YES)

SUBTYPE (ADMIN)

TOPICSTR (Store/Veggies)

DEST (VEGGIES.QLOCAL)

TOPICOBJ (FOODS.ANCHOR)

DESTCLAS (PROVIDED)

EXPIRY (UNLIMITED)

Figure 7-16. Subscriptions: The three aspects (1 of 2)

Subscriptions might be one of the most confusing components of publish/subscribe. Some of the terminology that is used for subscriptions might appear to be contradictory, but breaking down the different aspects of subscription should prove helpful. Subscriptions have three distinct aspects:

- Creation and deletion: How was this subscription created? You have two considerations: created by an application, or created administratively with an explicitly defined subscription object.
- Subscription queue management: How is the queue used by the subscription created or handled? If the subscription DESTCLASS attribute is set to MANAGED, the queue is taken care of by IBM MQ. These queues are dynamic queues with the prefix "SYSTEM.MANAGED." The next node in the queue name will be DURABLE or NONDURABLE, which brings you to the third subscription aspect.
- Subscription lifetime has two considerations: The first consideration is a durable subscription, which means that the life of the subscription does not depend on an application. The second type is non-durable, which means that the life of the subscription depends on an application.

The three subscription aspects are summarized in the next slide.

Subscriptions: The three aspects (2 of 2)

- Subscription lifetime
 - Durable subscriptions: Lifetime is independent of applications
 - Non-durable subscriptions: Lifetime is driven by applications

Valid subscription combinations

Queue management →	Managed		Unmanaged	
Lifetime →	Durable	Non-durable	Durable	Non-durable
Creation and deletion ↓				
Administrative	Yes	No	Yes	No
Application	Yes	Yes	Yes	Yes

Figure 7-17. Subscriptions: The three aspects (2 of 2)

This slide summarizes the three aspects of subscriptions: creation and deletion, queue management, and lifetime. You identify these three aspects in the DISPLAY SUB output.

Subscription commands: DIS SUB(VEGET*) ALL

- Lists subscription details, including the topic string to which it subscribes

```

DIS SUB (VEGET*) ALL
SUB(VEGETARIAN.SEARCH)
SUBID(C3E2D8D4D4D8F1F14040404040 ...
DURABLE (YES)
SUBTYPE (ADMIN)
DISTYPE(RESOLVED)
TOPICSTR(Store/Veggies)
DEST (SYSTEM.MANAGED.DURABLE.CDC4 ...
DESTQMGR(MQ11)
DESTCORL(C3E2D8D4D4D8F1F1404040 ...
TOPICOBJ(FOODS.ANCHOR)
PSPROP (MSGPROP)
PUBACCT(00000000000000000000000000 ...
PUBAPPID()
PUBPRTY(ASPB)
...
  
```

```

... continued
USERDATA()
SUBUSER(INGM000)
SUBLEVEL(1)
WSCHEMA(TOPIC)
SUBSCOPE(ALL)
DESTCLAS(MANAGED)
VARUSER(ANY)
SELECTOR()
SELTYPE(NONE)
EXPIRY (UNLIMITED)
REQONLY(NO)
CRDATE(2014-09-16)
CRTIME(16.07.06)
ALTDATE(2014-09-16)
ALTTIME(16.07.06)
END SUB DETAILS
  
```

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-18. Subscription commands: DIS SUB(VEGET*) ALL

This set of slides shows the MQSC commands available to display subscriptions. MQSC commands can be used with the `rwmqsc` tool. Equivalent capabilities are provided on IBM MQ Explorer.

The display for this slide shows the three aspects:

- Queue management: DESTCLASS = MANAGED, no explicit queue definition.
- Lifetime: DURABLE. See the DEST queue name that is prefixed with SYSTEM.MANAGED.
- Creation and deletion: SUBTYPE = ADMIN, explicitly defined subscription object. A subscription that is created by an MQSUB would display SUBTYPE = API. Review the SUBTYPE attribute in the IBM Knowledge Center for other possible MQSUB values.

An important attribute of a subscription object is PSPROP, which mitigates version-related behaviors by dealing with the way publish/subscribe message properties are added to messages sent to the subscription. You can prevent any properties from being added by setting PSPROP to NONE. Other values are applicable to PSPROP depending on the version of publish/subscribe and the applications used. Understand your publish/subscribe environment and applications, and review PSPROP attribute values in IBM Knowledge Center.

Subscription commands: DISPLAY PUBSUB ALL

- Shows how many topics subscribed to the local queue manager

```
MQ11 DIS PUBSUB ALL
...
DIS PUBSUB DETAILS
TYPE (LOCAL)
QMNAME (MQ11)
STATUS (ACTIVE)
SUBCOUNT (8)
TPCOUNT (13)
END PUBSUB DETAILS
```

Figure 7-19. Subscription commands: DISPLAY PUBSUB ALL

This set of slides shows the MQSC commands available to display subscriptions. MQSC commands can be used with the `runmqsc` tool. WebSphere MQ Explorer can also be used to administer publish/subscribe objects.

For IBM MQ Explorer, right-click the correct queue manager, and then select **Status > Publish/Subscribe**.



Subscription commands: **DISPLAY SUB (*) TOPICSTR DEST**

- Shows all subscriptions with the focus on the topic strings that they subscribe to and the associated queue

*Figure 7-20. Subscription commands: DISPLAY SUB(*) TOPICSTR DEST*

This set of slides shows the MQSC commands available to display subscriptions.

Subscription-related command: DISPLAY QLOCAL

- Displaying queues that are associated with subscriptions

```
dis q(SYSTEM.MANAGED.DURABLE.CDC456AF9B060E32) curdepth
QUEUE (SYSTEM.MANAGED.DURABLE.CDC456AF9B060E32)
TYPE (QLOCAL)
QSGDISP (QMGR)
CURDEPTH (0)
```

```
MQ11 DIS QLOCAL(VEGGIES) CURDEPTH
QUEUE (VEGGIES)
TYPE (QLOCAL)
QSGDISP (QMGR)
CURDEPTH (1)
```

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-21. Subscription related command: DISPLAY QLOCAL

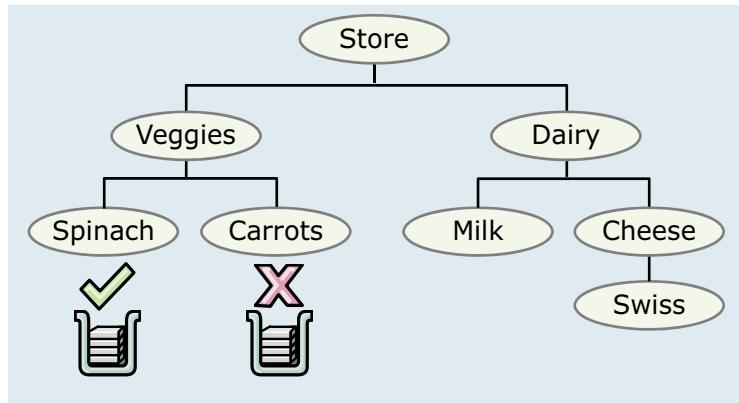
This set of slides shows the MQSC commands available to display subscriptions. MQSC commands can be used with the `runmqsc` tool. Equivalent capabilities are provided on IBM MQ Explorer.

To view the associated queues, select the Queues menu item in IBM MQ Explorer.

When looking for managed, or dynamic queues on IBM MQ Explorer, make sure to select the “Show Temporary Queues” toggle switch at the upper right of the queue display pane, and then look for the SYSTEM.MANAGED.* named queues.

Publications

- What is considered a successful publication?
- IBM MQ default behavior depends on the persistence of the message and the durability of the subscription
- Persistent messages
 - If a publication cannot be delivered to one or more durable subscriptions: Publish fails
 - If a publication cannot be delivered to a non-durable subscription: Publish still succeeds
- Publications to non-persistent messages are deemed successful regardless of the actual outcome
- Default behavior is controlled through the **PMSGDLV** and **NPMSGDLV** topic attributes
- Publication ending up in the dead-letter queue is considered successful; controlled through the **USEDLQ** topic attribute



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-22. Publications

When you start to work with publish/subscribe, it is important to determine what constitutes a successful return code on a publication, as this feedback might be misleading. Always test that a subscription was indeed received.

Topic attributes influence the handling of publications. These attributes might be inherited from the SYSTEM.BASE.TOPIC. PMSGDLV and NPMSGDLV to determine how to treat persistent, or non-persistent messages during a publication.

- ASARENT: Behavior inherited.
- ALL: Message must be delivered to all subscribers. If one subscriber fails, no other subscribers receive the message, and the call fails.
- ALLAVAIL: Message is delivered to all available subscribers regardless of whether any subscriber fails.
- ALLDUR: Delivery failures to non-durable subscribers do not send an error to the MQPUT call. Delivery failures to durable subscribers cause no subscribers to receive the message and the MQPUT call to fail.

USEDLQ can be set to YES or NO. If NO, or if the queue manager does not specify a DLQ value, the message is treated per PMSGDLV or NPMSGDLV. If USEDLQ is set to YES, then delivery to the dead-letter queue is treated as a successful publication.

Retained publications

- Messages that are published to a topic string are received by applications that are subscribed to the topic at the time the message is published
- Applications that subscribe to the topic after the publication occurs do not get the messages
- Retained publications
 - The queue manager retains the most recent publication for each topic
 - Applications that subscribe after the publication takes place receive the most recently retained message
- The use of retained options is for specific use cases, such as infrequent publications

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

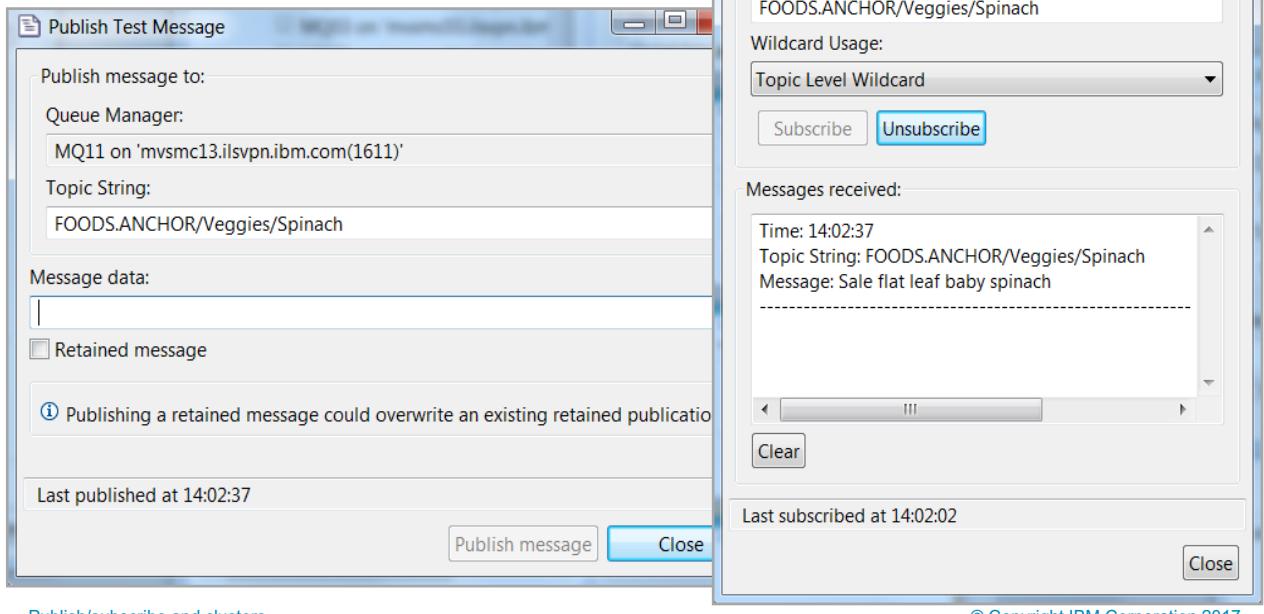
Figure 7-23. Retained publications

Retained publications are specified by using the `MQPMO_RETAIN` PMO option by the application.



Publish/subscribe testing in IBM MQ Explorer

1. Right-click **Topics** for a menu
2. Subscribe to a topic
3. Publish to the same topic



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-24. Publish/subscribe testing in IBM MQ Explorer

This slide shows another look at the IBM MQ Explorer publish/subscribe testing capabilities. When using these panes, always subscribe to the topic first.

Publish/subscribe lifecycle descriptions

- Provide a good understanding on how and why to select different attributes for the way publish/subscribe is implemented
- Aid in planning and design stages
- Three available lifecycle scenarios:
 - Managed, non-durable subscriber
 - Managed, durable subscriber
 - Unmanaged, durable subscriber

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-25. Publish/subscribe lifecycle descriptions

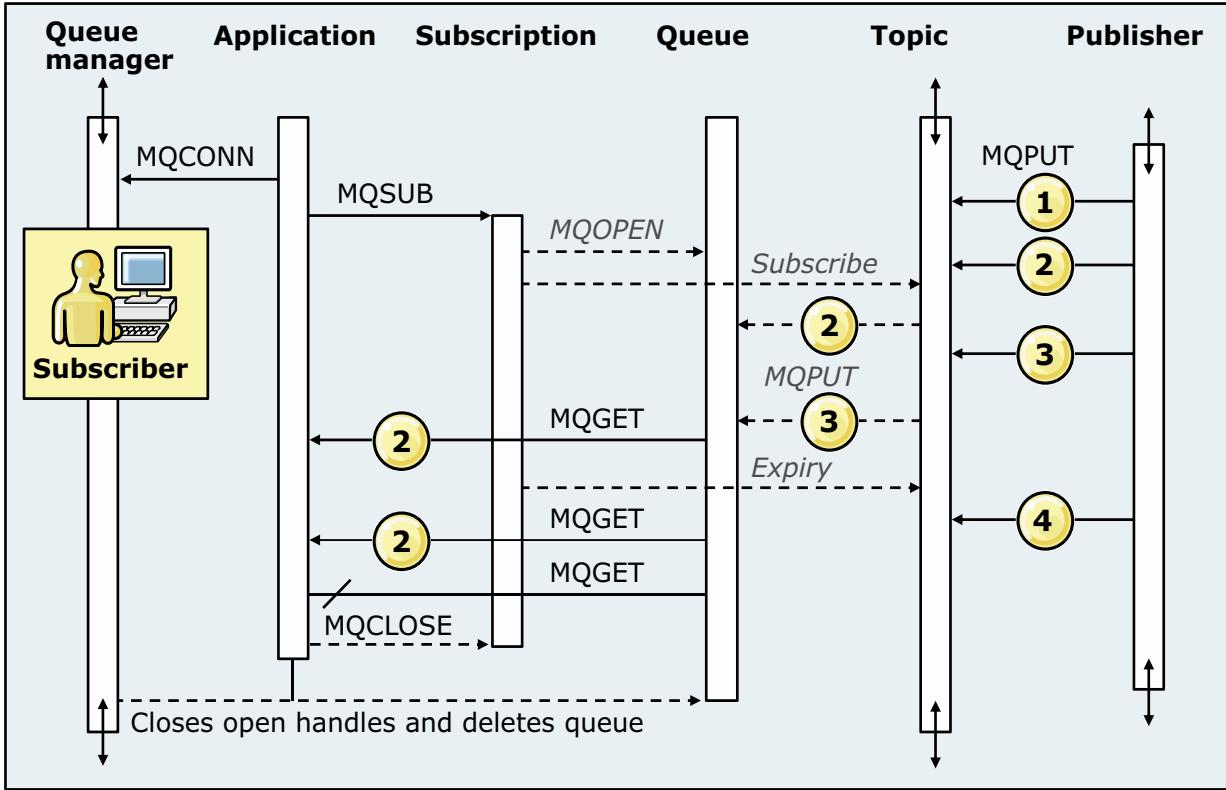
The publish/subscribe topic can fill over a week-long course. While this class must adhere to the basic scope, the publish/subscribe lifecycle descriptions are introduced so that you can use them as a follow-up if you need to develop publish/subscribe applications. The lifecycle topics are helpful when designing and troubleshooting publish/subscribe applications.

Three lifecycle descriptions are available in the IBM Knowledge Center.

IBM Training



Publish/subscribe lifecycles: Managed non-durable subscriber



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-26. Publish/subscribe lifecycles: Managed non-durable subscriber

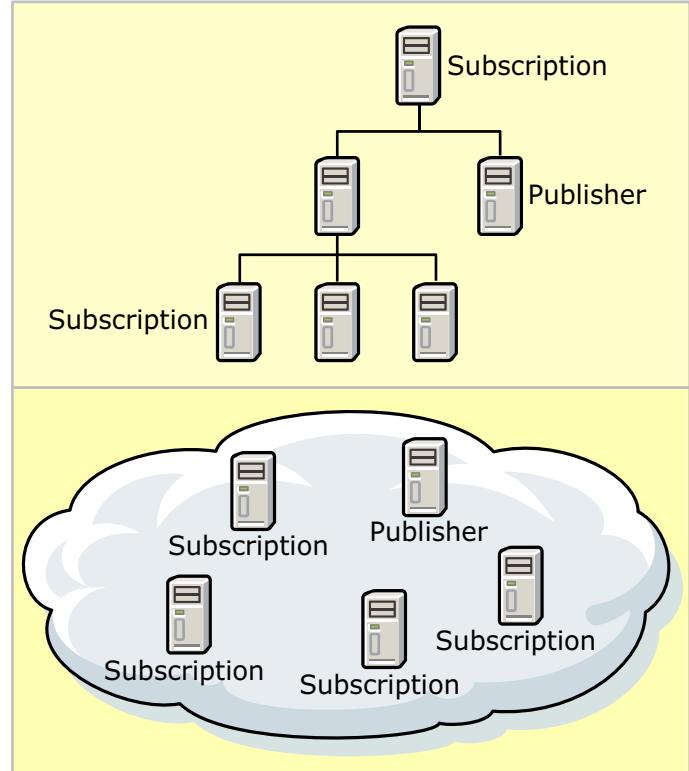
This slide is used to familiarize you with the lifecycle descriptions available in the IBM MQ Knowledge Center. **The text in that accompanies this slide is copied directly from the IBM Knowledge Center.**

1. The application creates a subscription on a topic that was already published to twice. When the subscriber receives its first publication, it receives the second publication, which is the currently retained publication.
2. The queue manager creates a temporary subscription queue and a subscription for the topic.
3. The subscription has an expiry. When the subscription expires, no more publications on the topic are sent to this subscription, but the subscriber continues to get messages that were published before the subscription expired. Subscription expiry does not affect publication expiry.
4. The fourth publication is not placed on the subscription queue. Consequently, the last MQGET does not return a publication.
5. Although the subscriber closes its subscription, it does not close its connection to the queue or the queue manager.

The queue manager cleans up shortly after the application ends. Because the subscription is managed and non-durable, the subscription queue is deleted.

Distributed publish/subscribe

- Extends subscriptions and publications from a single queue manager to other queue managers in the IBM MQ infrastructure
- Uses distributed IBM MQ infrastructure
- Hierarchies:
 - Indirect many-to-many connectivity
 - Direct one-to-many connectivity
- Publish/subscribe clusters:
 - Many-to-many connectivity
 - Direct routing or topic host routing many-to-many connectivity
 - Based on IBM MQ clustering



Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-27. Distributed publish/subscribe

You learned about the basics of publish/subscribe, but all the items so far – topics, subscriptions, and publications – might take place within the same queue manager. In this slide, you learn about extending publish/subscribe to the IBM MQ infrastructure.

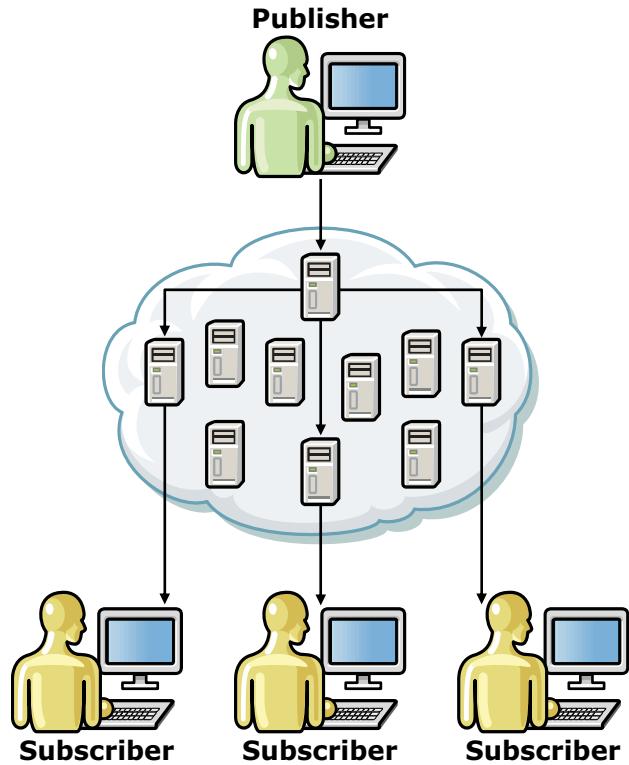
A publish/subscribe infrastructure can consist of:

- Publish/subscribe hierarchies
- Publish/subscribe clusters
- A combination of hierarchies and clusters

With IBM MQ V8 and later, publish/subscribe clusters can be direct, or topic host based.

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 a specific queue manager
- Use the **ALTER QMGR PSCLUS()** command to control whether this queue manager participates in publish/subscribe activity across any clusters in which it is a member



[Publish/subscribe and clusters](#)

© Copyright IBM Corporation 2017

Figure 7-28. Publish/subscribe clusters

Using clusters in a publish/subscribe topology provides 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.
- This topology has no single point of failure. 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 workload.

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 modifying 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
 - A 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 becomes aware of the clustered topic and shares subscription knowledge for any topic strings within that branch of the topic tree

[Publish/subscribe and clusters](#)

© Copyright IBM Corporation 2017

Figure 7-29. 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 that 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)
```

Commands for displaying cluster topic:

DISPLAY TOPIC(FOOTBALL)	
DISPLAY TOPIC(FOOTBALL) TYPE(LOCAL)	<i>Local objects only</i>
DISPLAY TOPIC(FOOTBALL) TYPE(ALL)	<i>Local and cluster objects</i>
DISPLAY TOPIC(FOOTBALL) TYPE(CLUSTER)	<i>Cluster objects only</i>
DISPLAY TOPIC(FOOTBALL) CLUSINFO	
DISPLAY TCLUSTER(FOOTBALL)	
DISPLAY TOPIC(FOOTBALL) TYPE(ALL) CLUSINFO	

Figure 7-30. 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(SPORTS)	TYPE(CLUSTER)
TOPICSTR(/global/sports/football)	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

Figure 7-31. Display cluster topic example

The figure shows an example of the **DISPLAY TOPIC TYPE(CLUSTER)** command.

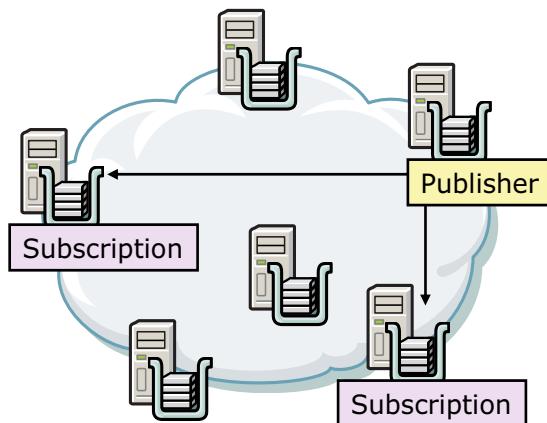
The CLROUTE attribute identifies the publish/subscribe cluster routing as direct or topic host. Cluster routing options are described in more detail next.

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 queue manager of the publisher to every queue manager in the cluster with a matching subscription
- Default mode

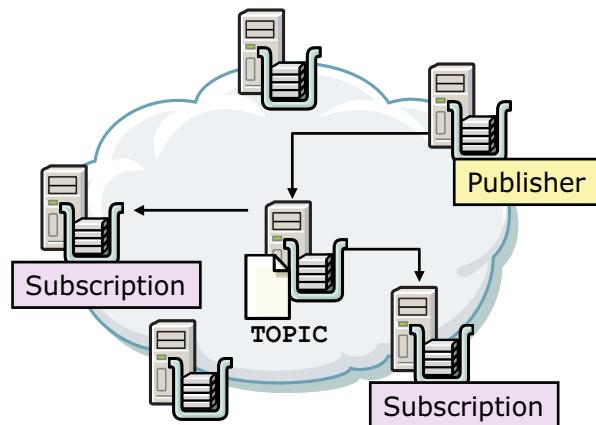


Publish/subscribe and clusters

Figure 7-32. Cluster modes for routing publications

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



© Copyright IBM Corporation 2017

IBM MQ V8 and later supports two types of routing for publish/subscribe clusters.

In a direct routing cluster, each queue manager in the cluster is 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, including full or partial repositories
- 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 the topic definitions have a mismatch, a conflict is reported
 - Typically used with topic host routing to avoid single point of failure

Example:

On QM1:

```
DEF TOPIC(SCORES) TOPICSTR('/Sport/Scores') +
CLUSTER(DEMO) CLROUTE(TOPICHOST)
```

On QM2:

```
DEF TOPIC(SCORES) TOPICSTR('/Sport/Scores') +
CLUSTER(DEMO) CLROUTE(TOPICHOST)
```

Figure 7-33. 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. They remain usable 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 to be separated. In publish/subscribe clusters that have only 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 uses 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
- Check the **CLROUTE** setting for all topics on all hosts in the cluster with the MQSC command: **DISPLAY TCLUSTER(*) CLROUTE**
- After a topic object has been clustered, you cannot change the value of the **CLROUTE** property unless you remove the topic from the cluster first (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

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-34. Routing behavior for publish/subscribe clusters

This figure lists some tips for detecting and handling publish/subscribe routing problems with when you are using clustered publish/subscribe messaging.

For more tips, see the “Routing for publish/subscribe clusters: Notes on behavior” topic in the IBM Knowledge Center.

Topic host routing requirements

- Publications are forwarded to these topic host queue managers, who then forward to queue managers with matching subscriptions
- Only the queue managers where the topic object is defined are aware of everyone else
- The following queue managers must be at IBM MQ V8 or above:
 - 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 are not aware of the topic host route and behave as if it is not defined in the cluster

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-35. Topic host routing requirements

To use topic host routing in a cluster, the following queue managers must be at IBM MQ V8 or later:

- 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 is not defined 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

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-36. Scaling

The entries in this slide compare 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

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-37. 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. A topic host routed cluster 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

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-38. Configuration

This figure compares how configuration differs for the supported publish/subscribe topologies.

Direct routed clusters require the least amount of configuration. Queue managers 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 removing queue managers from this type of topology.

Hierarchies require the most configuration and planning. It is also more difficult to add or remove queue managers from a publish/subscribe hierarchy without affecting other queue managers in the topology.

Availability

- Direct routed cluster
 - An unavailable queue manager affects the subscribers and publishers on that queue manager only
- Topic host routed cluster
 - An unavailable queue manager can affect other queue managers if the unavailable queue manager hosts the only definition of a topic
 - Hosting different topics on different queue managers can increase availability

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-39. Availability

This slide 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
 - Direct and topic host routed clusters can be configured to avoid single points of failure
- Publication performance
 - Both topologies use similar queue manager techniques for transferring messages
 - Both 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

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-40. 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.

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)

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

Figure 7-41. 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
- Check the channels
- Check the topic status
- Watch for publication movement
- Look for a buildup of messages on a queue

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-42. Problem determination (2 of 2)

The following list provides some important considerations with problem determination:

- Double-check your configuration. Remember that 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 messages have problems, they 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 or unregister 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 with cluster name

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-43. 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 is 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

- When stopping a queue manager, local subscribers are disconnected, but they might not notify other queue managers in the publish/subscribe cluster or hierarchy
 - Use **DISPLAY SUBSTATUS** to identify subscribing applications
 - Disconnect subscribing applications before stopping a queue manager or channel initiator
- If subscribers are not disconnected first
 - Durable subscribers: Transmission queue buildup is required
 - Nondurable subscribers: Transmission queue buildup might occur
- Try to avoid backlog of messages on
SYSTEM.CLUSTER.TRANSMIT.QUEUE

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-44. 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 shutting 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

- Differentiate between publish/subscribe and point-to-point messaging
- Summarize how the history of publish/subscribe influences its functions and terminology
- Identify the basic components of publish/subscribe
- Describe key properties of topics, subscriptions, and publications
- Describe the publish/subscribe cluster topologies
- Explain how to configure a direct routing publish/subscribe cluster
- Explain how to configure a topic host routing publish/subscribe cluster

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-45. Unit summary

Review questions

1. True or False: The most recent V7+ publish/subscribe implementation is referred to as *queued publish/subscribe* and *command-based publish/subscribe*.
2. How can you tell whether a subscription was explicitly defined, or created with **MQSUB** in the output of **DIS SUB** or **DIS SBSTATUS** commands?
 - A. Check the displayed subscription attribute USERDATA
 - B. Check the displayed queue manager attribute PSMODE
 - C. Check the displayed subscription attribute PSPROP
 - D. Check the displayed subscription attribute SUBTYPE
3. True or False: In public/subscribe terminology, “managed” and “administered” have opposite meanings.
4. True or False: If the **PSMODE** in the queue manager is set to **COMPAT**, the *queued publish/subscribe* interface is running.

Publish/subscribe and clusters

© Copyright IBM Corporation 2017

Figure 7-46. Review questions



Review answers (1 of 2)

- True or False: The most recent V7+ publish/subscribe implementation is referred to as *queued publish/subscribe* and *command-based publish/subscribe*.

The answer is False. The V7+ publish/subscribe implementation is referred to as *integrated publish/subscribe*. *Queued publish/subscribe* is the older version.

- How can you tell whether a subscription was explicitly defined, or created with **MQSUB** in the output of **DIS SUB** or **DIS SBSTATUS** commands?
 - Check the displayed subscription attribute USERDATA
 - Check the displayed queue manager attribute PSMODE
 - Check the displayed subscription attribute PSPROP
 - Check the displayed subscription attribute SUBTYPE

The answer is D. An application defined subscription shows SUBTYPE(API), while a subscription object shows SUBTYPE(ADMIN).



Review answers (2 of 2)

3. True or False: In public/subscribe terminology, “managed” and “administered” have opposite meanings.

The answer is True. A managed queue is dynamically defined, while an administered topic or a subscription object is explicitly defined.



4. True or False: If the `PSMODE` in the queue manager is set to `COMPAT`, the *queued publish/subscribe* interface is running.

The answer is False. `PSMODE (ENABLED)` indicates that both the *queued publish/subscribe* engine, and the publish/subscribe engine are active.

Exercise: Configuring a publish/subscribe cluster

© Copyright IBM Corporation 2017

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

Figure 7-49. Exercise: Configuring a publish/subscribe cluster

Exercise objectives

- Create a cluster topic and review its status
- Create subscriptions and publications to a topic
- Use the `dsmpqrte` tool to identify the path of a message in the publish/subscribe cluster
- Configure and verify topic host routing

Unit 8. Cluster design considerations

Estimated time

00:30

Overview

This unit covers complex cluster design and implementation options such as overlapping clusters and overlapping cluster gateways.

Unit objectives

- Summarize environmental factors that might hinder the effectiveness of a complex cluster implementation
- Describe techniques to consider when dividing a large organization of systems
- Explain how to distinguish among classes of service when designing a clustering solution
- Explain how to define and manage an overlapping cluster
- Describe how to define a cluster gateway in an overlapping cluster environment

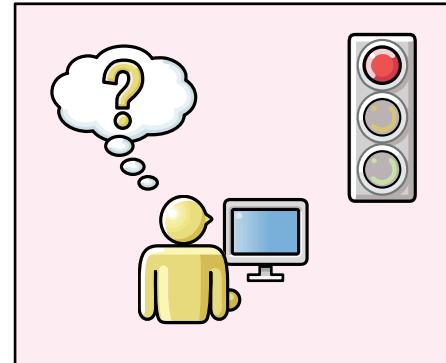
Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-1. Unit objectives

Before you start, consider your environment

- Is the cluster design as simple as possible?
- Are all overlapping cluster queue managers under the control of the same IBM MQ administrator team
- Is there concise documentation of the cluster environment available to the IBM MQ team
- Is the IBM MQ team able to manage the more complex configuration
- Is the team that sets up IBM MQ monitoring part of the IBM MQ administrators, or part of the monitoring tool team?



- Do you allow queue managers that are external to your company to join your cluster
- Is the degree of configuration complexity justified?

[Cluster design considerations](#)

© Copyright IBM Corporation 2017

Figure 8-2. Before you start, consider your environment

This unit presents complex cluster design options that can be implemented to meet different message flow objectives. As you see throughout this unit, IBM MQ cluster mechanisms provide configuration flexibility to meet organizational and message volume factors. However, it is critical to ensure that the environment can be adequately maintained and supported.

Before considering the workload balancing capabilities that are introduced in an earlier unit, and the more advanced configurations that are presented in this unit, you need to understand your organization's ability to maintain the environment. This need is especially pertinent during critical processing times.

Some examples are as follows:

- Is your support structure internal, or outsourced?
- If the support team is outsourced, is the entire support team knowledgeable enough about working with mixed clusters to resolve a mixed cluster problem during a peak period?
- Do your administrators have control of all the queue managers in an overlapping cluster? For example, if the cluster has unexpected traffic, is your team able to investigate any procedures that might need to be stopped, such as repeated `REFRESH CLUSTER` commands?
- Is the team that designs the cluster aware of all the security consequences of allowing external organizations to join the cluster, particularly when it comes to SSL/TLS certificates?

- Does the team that is monitoring the system have sufficient MQ skills to know when an issue requires the assistance of an MQ administrator and when it does not?

The monitoring issue can cause significant problems. Sometimes the monitoring team has expertise in the monitoring tool framework, but they might not have enough depth in IBM MQ and require close work with an IBM MQ administrator. ***This caveat is included due to an actual production occurrence.*** When a cluster is monitored, a tool might sense an outage in a queue manager in an external organization, and page out your administrator team to handle the problem. Getting a call at 3:00 AM for a problem that is outside your control is a waste of resources.

When a cluster infrastructure is designed, designers must always have security, maintenance, and support present in the design. On the contrary, an inadequately integrated cluster might prove to introduce more problems than benefits.

A good general rule is to return to the question, ***is the complexity justified?*** A successful cluster implementation is one that can be supported during critical peak processing periods. Keep this caveat in mind as you continue with the material in this unit.

Reasons for dividing large systems

- Combine related applications
- Isolate data and users
- Reduce network resources
- Reduce IBM MQ definitions
- Isolate environments
- Allow independent administration
- Provide different service classes and security levels

Cluster design considerations

© Copyright IBM Corporation 2017

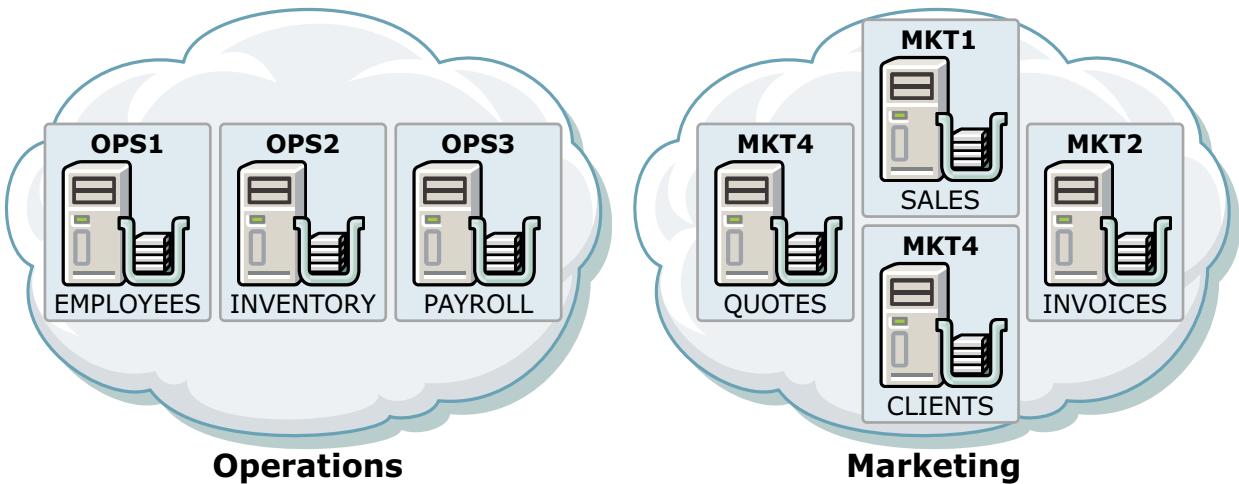
Figure 8-3. Reasons for dividing large systems

You can divide large organizations of systems into more than one cluster for different reasons. Considerations include:

- Applications: Human resources applications can be separated from applications such as customer information or inventory.
- Data: Data warehouse queue managers can be separated so that car rental information and hotel information can be controlled separately.
- Security: Highly secure information such as payroll or customer credit card information can be stored on a high security cluster.
- Administration: Clusters can be created based on local administration of resources. For example, different administrators might administer North American clusters and Asia Pacific clusters.
- Network: Many high availability environments are set up to protect against network failures. Queue managers within a cluster contain alternative queue managers that are connected to the cluster through a different network.
- Environment: Many IBM MQ users create different clusters that are based on the type of environment. Production, development, and quality assurance clustered environments allow for complete testing, which can help the integrity of the production environment.

Some considerations can become more relevant than others, depending on the classes of service that you need to create for the clusters in the organization.

Combining related applications



Cluster design considerations

© Copyright IBM Corporation 2017

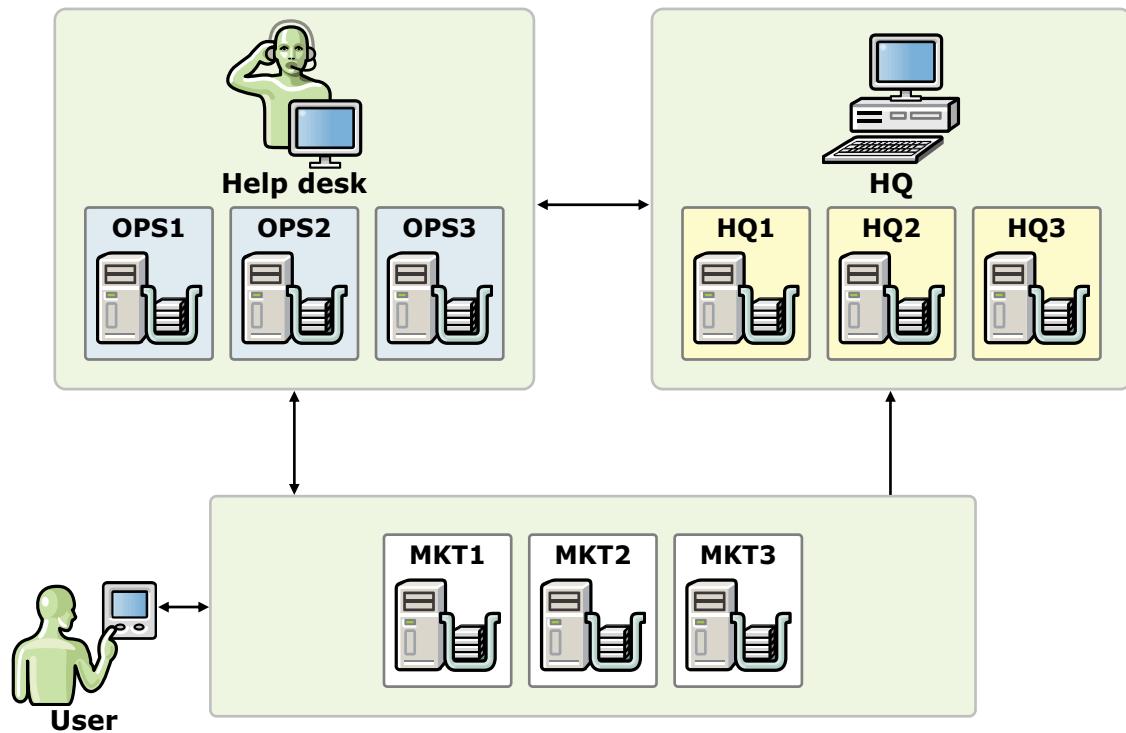
Figure 8-4. Combining related applications

Typically a cluster contains queue managers that are logically related in some way and need to share some data and applications.

A company with different departments can choose to group the queue managers in many small clusters, each one with data and applications that are related to the business of the department. This configuration supports independent administration for the applications and establishment of different classes of service.

For example, the cluster named “Operations” handles human resources applications and does not need to view or get information from the cluster that is named “Marketing” that controls the accounts of the company. Since clusters can have different configurations for supporting different services, cluster “Marketing” can restrict access for workers from the other departments.

Isolate data and users



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-5. Isolate data and users

You can choose to divide large organizations to isolate the data and users for many reasons, such as:

- Visibility
- Priority
- Speed of processing

For example, a company might create a separate cluster to isolate sensitive data, such as financial transactions. This cluster would have a strict security policy, and the access would be restricted to queue managers and applications in the same network.

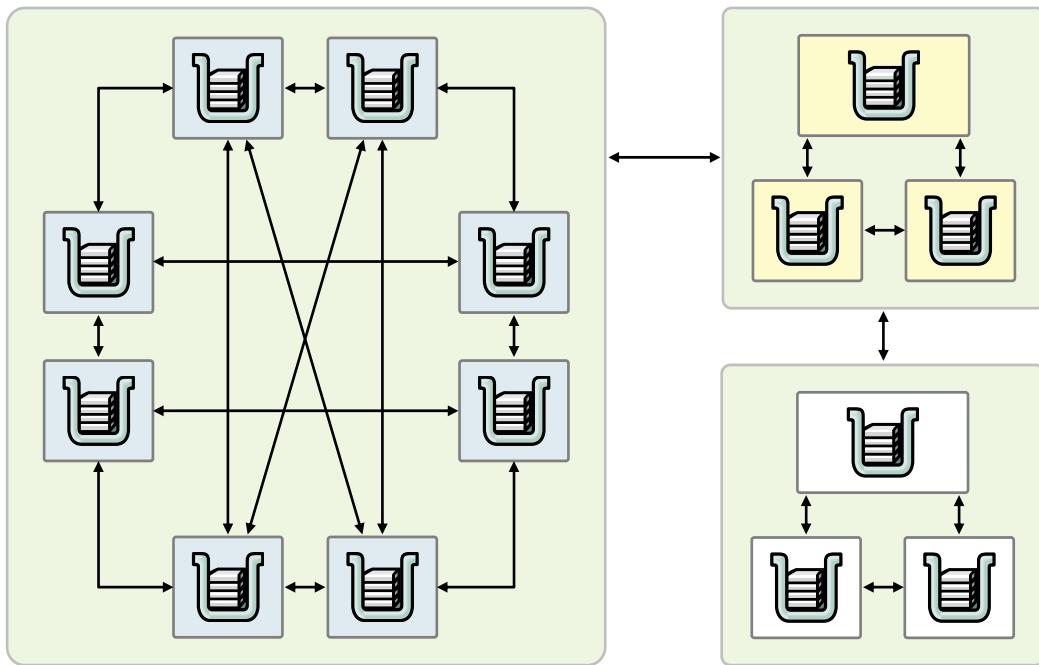
A cluster for connecting external users and handling their messages would isolate it from the other clusters, eliminating external threats.

Another option would be to group frequently used or priority applications in a cluster with faster processing.

Later, you see how clusters can be interconnected through channels with different classes of service.

Reduce resources

- Fewer IBM MQ definitions and administration
- Reduce network communication



Cluster design considerations

© Copyright IBM Corporation 2017

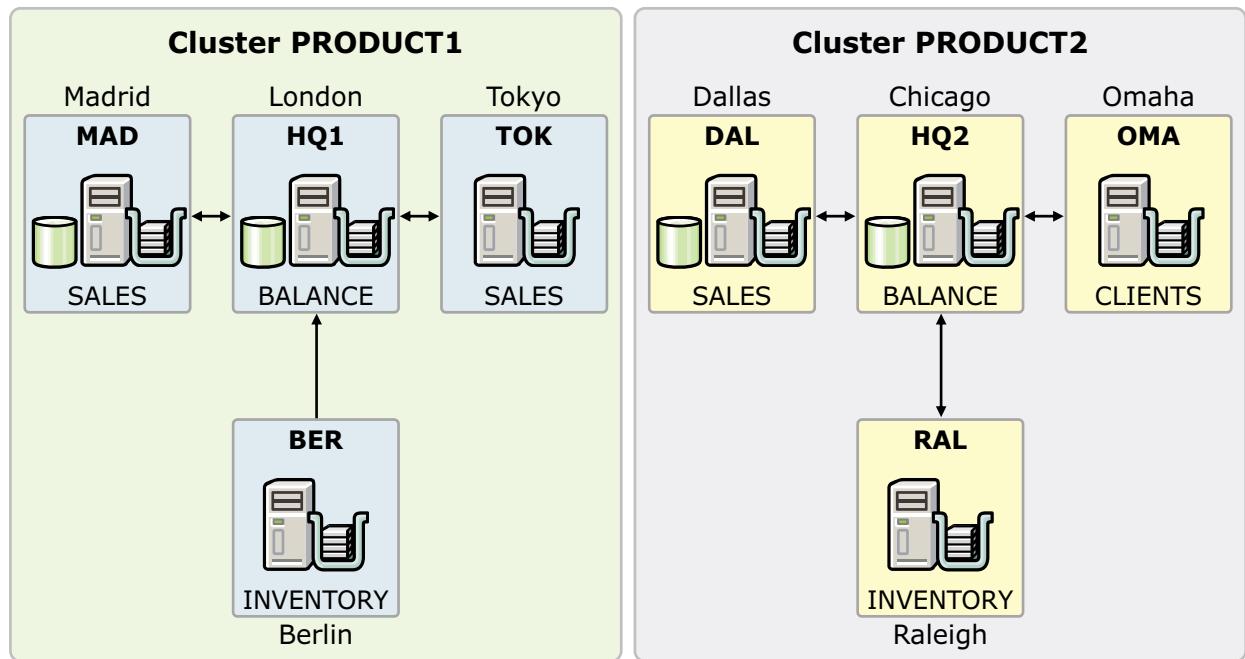
Figure 8-6. Reduce resources

In an IBM MQ cluster, each queue manager automatically creates the required channel and queue definitions to communicate with other queue managers in the same cluster. If each queue manager communicates with every other queue manager in the same cluster, many definitions are created automatically.

Many channel definitions increase the amount of system administration and the use of IBM MQ resources. Moreover, establishing communications between every queue manager in a cluster leads to a risk of using excessive network resources.

Dividing large clusters reduces the communications to smaller numbers of cluster queue managers and results in creating fewer automatically defined cluster objects. This configuration allows easier administration because fewer channels need to be administered, and the queue manager uses fewer resources. In this configuration, the communication between the clusters occurs across a queue manager that is chosen to act as a gateway.

Independent administration



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-7. Independent administration

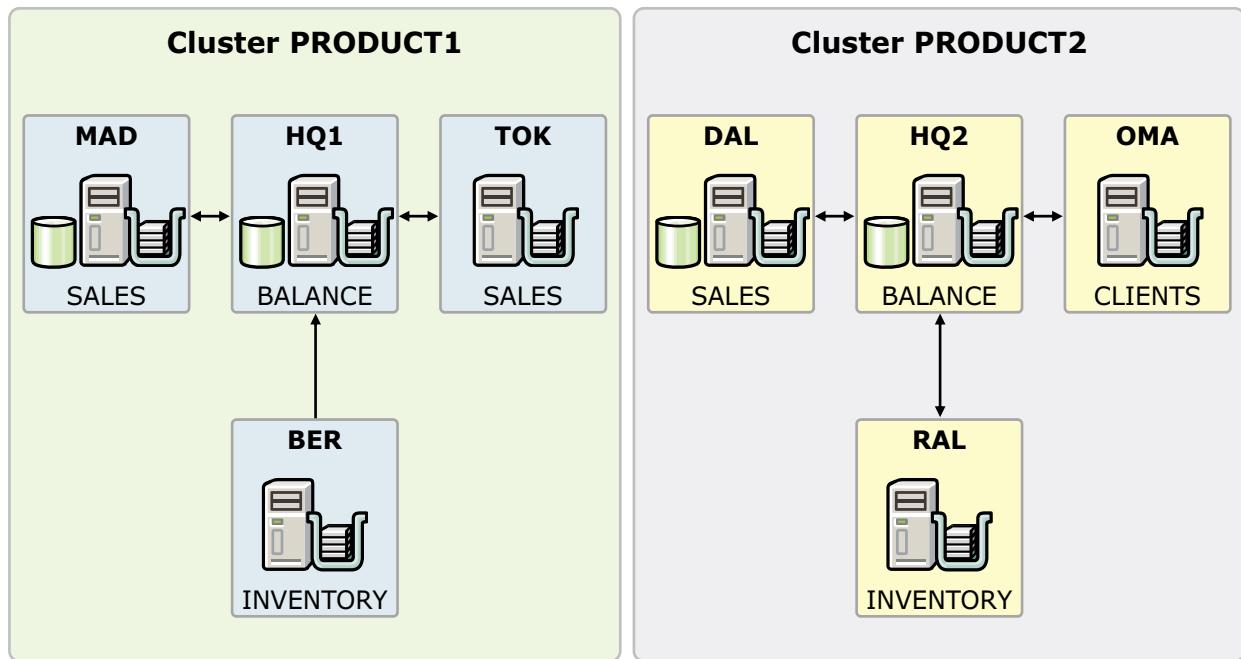
You can group queue managers in clusters to allow independent administration of different organizations. The system administrator for each cluster chooses what to do with the queue manager objects.

Consider 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 its own data and definitions and can implement a different security policy.

The figure contains two clusters with the same queue names. The clusters can have similar systems, but the messages are processed separately and they can have different security policies.

Different environments

- Production and test environments
- Different operating systems and hardware



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-8. Different environments

You might organize your clusters to create separate test and production environments. You can create clusters such that the queue managers in it are clones of each other, able to run the same applications and with local definitions of the same queues.

With the test environment, you can simulate the production environment and identify problems in applications, queue manager definitions, and any other potential areas before implementing the production environment.

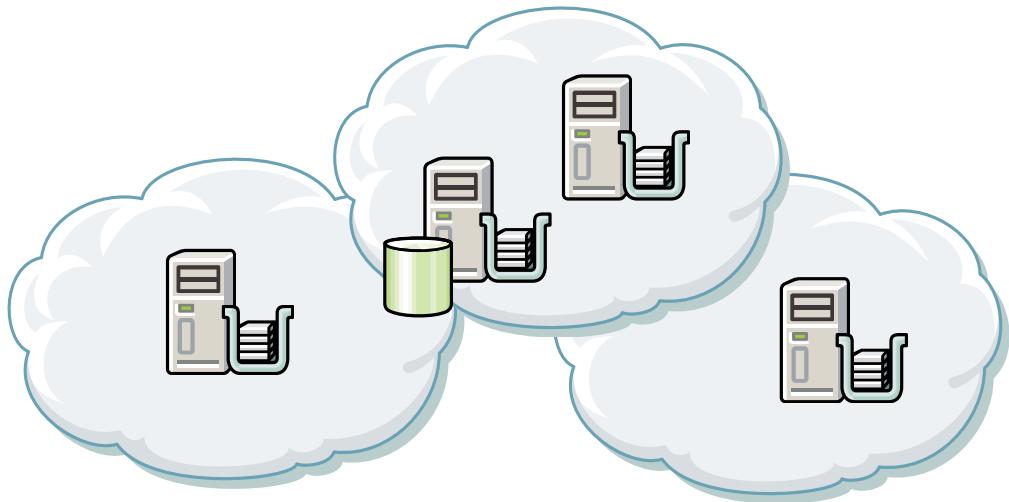
Test and production clusters should use the same:

- Queue manager names
- Applications
- Definitions and configurations

Queue managers on different operating systems or running different versions of IBM MQ can be grouped in separate clusters to allow full use of all features and resources available.

Why overlapping clusters?

- Restrict the visibility of the queues and queue manager names across the clusters
- Create classes of service
- Allow independent applications to be administered separately
- Allow different organizations to have their own administration



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-9. Why overlapping clusters?

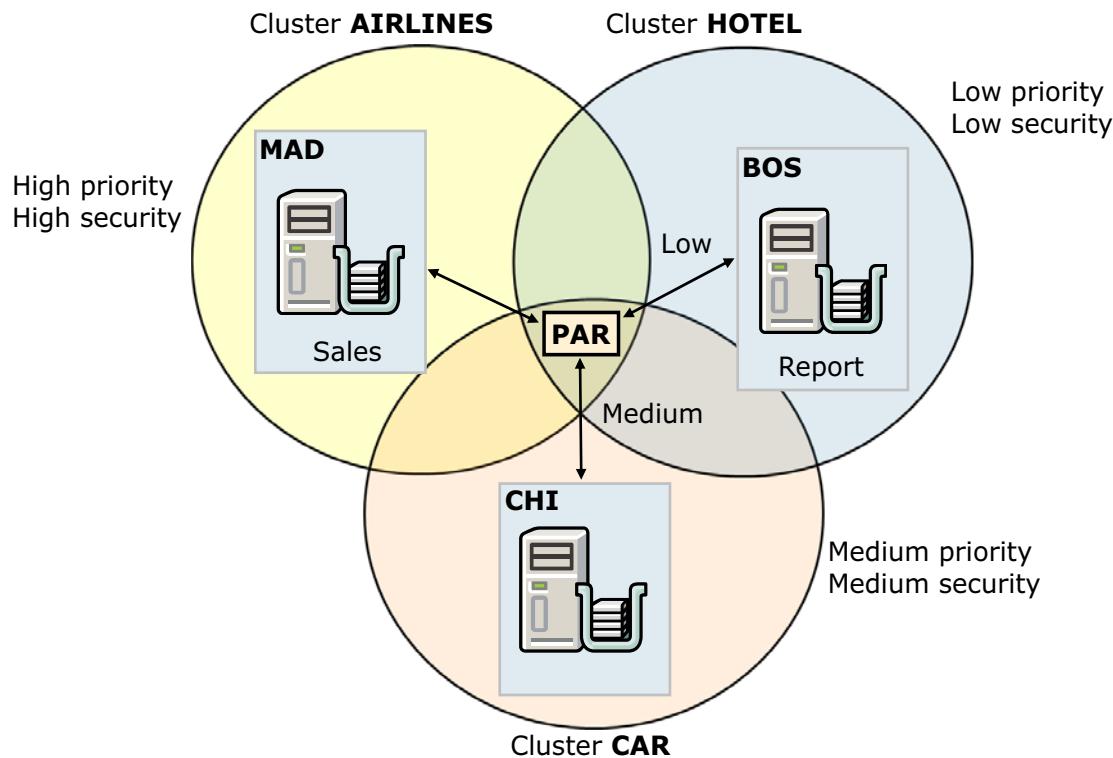
You can group all your queue managers into multiple smaller clusters with one or more queue managers in each acting as a bridge, overlapping another cluster. ***In this set of slides, the term “bridge” is used to refer to a queue manager that belongs to more than one cluster. It does not refer to a cluster queue manager that is used to connect to non-cluster queue managers as presented in an earlier unit.***

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

Classes of service and security



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-10. Classes of service and security

You can group queue managers into IBM MQ clusters with different configurations for each group to allow different classes of service between all clusters. For example, the applications that run on queue managers in the cluster AIRLINES can handle messages faster than in the other clusters to provide higher performance.

You can configure your channels in overlapping clusters to provide different classes of service, by using different:

- Security checking
- Queue manager channel priority
- Bandwidth

You can interconnect clusters by using channels with different priorities. Channels with a higher priority can move messages faster than channels with lower priority. Also, you can use a network with high bandwidth to provide faster throughput for a cluster. In the figure, the overlapping clusters exchange messages between each other by using the PAR queue manager. The AIRLINES cluster exchange messages by using channels with a high priority. The HOTEL cluster is connected through channels with low priority. The CAR cluster exchanges messages through channels with medium priority.

The clusters can exchange messages on channels with different security levels. Channels that implement SSL provide more secure communication. For example, cluster AIRLINES might have all queue managers that implement SSL for security.

Design considerations for multiple clusters

- Consider a cluster as a single unit of administration
- Where multiple clusters meet (which separate teams or individuals administer), have clear policies in place for controlling administration of the gateway queue managers
- Treat overlapping clusters as a single namespace for channel and queue manager names
- A clear understanding of who owns what, with enforceable rules and conventions, helps clustering run smoothly when overlapping clusters

Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-11. Design considerations for multiple clusters

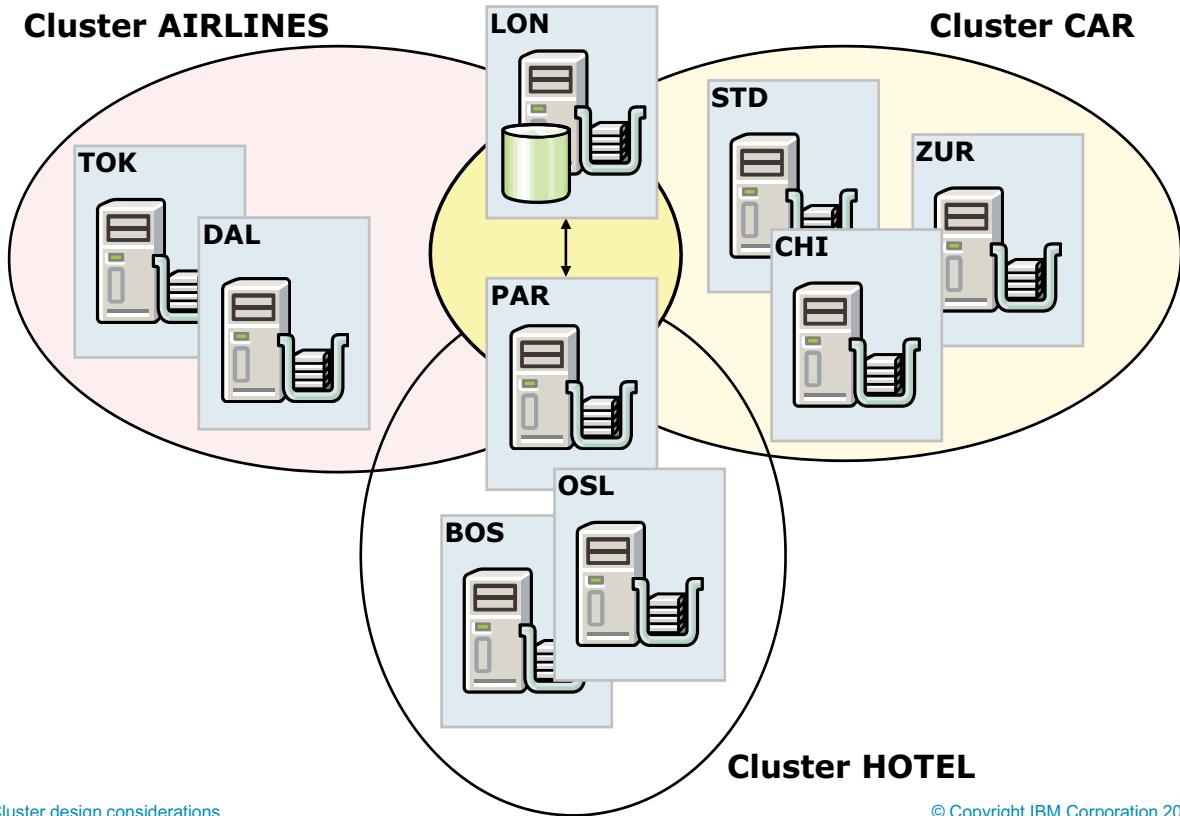
Although IBM MQ clusters are “loosely coupled,” it is useful to consider a cluster as a single unit of administration. This concept is used because the interaction between definitions on individual queue managers is critical to the smooth functioning of the cluster. For example, when using workload balanced cluster queues, it is important for a single administrator or team to understand the full set of possible destinations for messages. This set of possible destinations depends on definitions that are spread throughout the cluster.

Where multiple clusters meet, it is important to establish clear policies for the administration of the gateway queue managers.

It is useful to treat overlapping clusters as a single namespace and ensure that channel names and queue manager names are unique throughout both clusters. Administration is much easier when objects are named uniquely throughout the entire topology.

Sometimes administrative and system management cooperation is essential. For example, you must ensure cooperation between organizations that own different clusters that need to overlap. A clear understanding of who owns what, and enforceable rules and conventions, helps clustering run smoothly when overlapping clusters.

Overlapping cluster considerations



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-12. Overlapping cluster considerations

A queue manager can belong to more than one cluster by defining cluster-sender and cluster-receiver channels to each cluster that you want the queue manager to belong to.

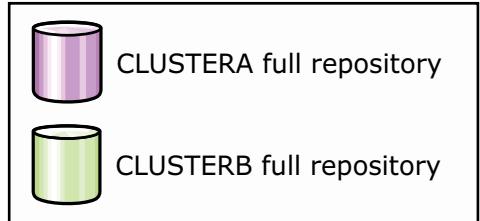
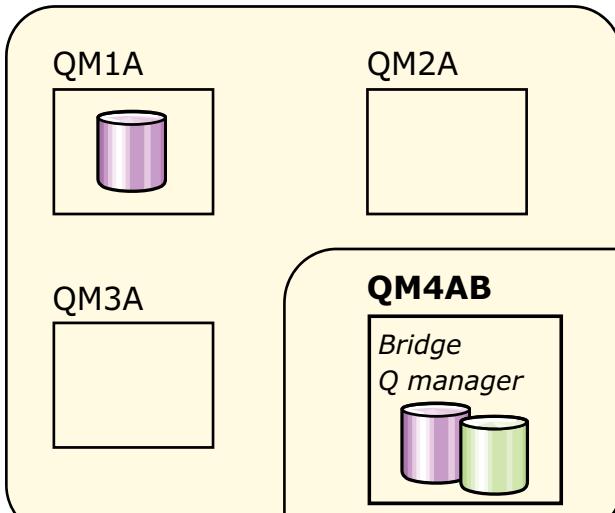
In overlapping clusters, the queues that are defined on queue managers that belong to a single cluster are not visible in other clusters. Only the bridge queue managers are visible to both clusters. One or more bridge queue managers are possible.

When a queue manager is a member of more than one cluster, you can take advantage of a *namelist* to reduce the number of definitions. A queue can be advertised in more than one cluster by adding the `CLUSNL` parameter on the `DEFINE QLOCAL` command to specify a namelist. The namelist contains the names of the clusters instead of using the `CLUSTER` attribute.

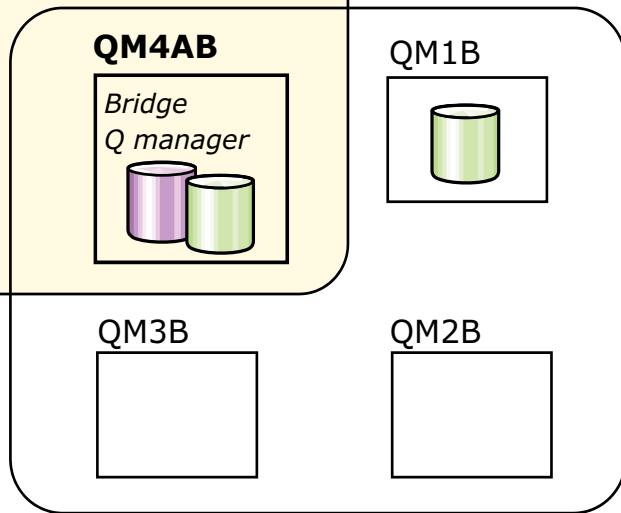
If two clusters with the same name are merged, they cannot be separated again.

Bridge queue managers

CLUSTER A



CLUSTER B



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-13. Bridge queue managers

In this use of the term “bridge queue managers”, bridge queue managers are members of more than one cluster, and are visible to more than one cluster.

Membership in more than one cluster is implemented by defining cluster-sender and cluster-receiver channels for each cluster of which the queue manager is a member. Use a namelist to simplify definitions.

Queues that are defined on queue managers that belong to a single cluster are visible only in that cluster.

A bridge queue manager can be a full repository or a partial repository. As shown in this example, the bridge queue manager can act as a full repository for more than one queue manager cluster.

Bridge queue manager configuration

- Use a namelist to identify clusters and reduce the number of definitions

Example:

```
DEFINE NAMELIST(CLUSTERS) NAMES(CLUSTERA,CLUSTERB) +
ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)
```

- Add a cluster-receiver channel for new cluster

- Option 1: Use CLUSNL in channel definition

Example:

```
DEF CHL(TO.QM4AB) CHLTYPE(CLUSRCVR) +
CLUSTER(' ') CLUSNL(CLUSTERS) +
CONNNAME(QM4AB's address)
```

- Option 2: Create a unique cluster-receiver channel for each cluster for explicit control

Example:

```
DEF CHL(CLUSTERA.QM4AB) CHLTYPE(CLUSRCVR) +
CLUSTER('CLUSTERA') CONNAME(QM4AB's address)
DEF CHL(CLUSTERB.QM4AB) CHLTYPE(CLUSRCVR) +
CLUSTER('CLUSTERB') CONNAME(QM4AB's address)
```

- Add a cluster-sender channel to new cluster repository

[Cluster design considerations](#)

© Copyright IBM Corporation 2017

Figure 8-14. Bridge queue manager configuration

This figure summarizes the steps for configuring a bridge queue manager.



Important

Using a namelist in the channel definition (as described in Option 1) reduces the number of channels but means that administrators share ownership of an overlapping cluster.

Creating a unique cluster-receiver and cluster-sender to each cluster (as described in Option 2) provides more explicit control over the channels.

Bridges: Joining more than one cluster

```
DEFINE NAMELIST(CLUSTERS)
NAMES(CLUSTERA,CLUSTERB)

ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)

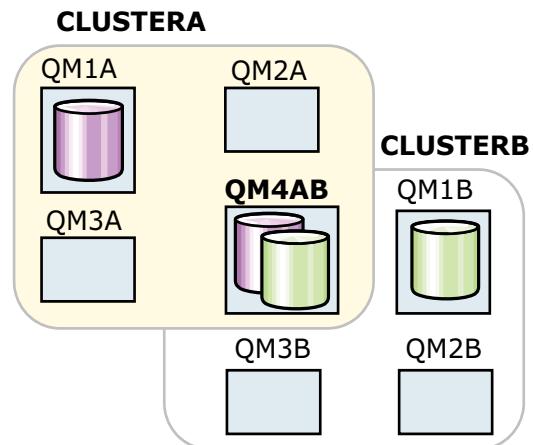
DEF CHL(CLUSTERA.QM4AB) +
CHLTYP(CLUSRCVR)CLUSTER('CLUSTERA') +
CONNNAME(QM4AB's address) +
DESCR('Cluster rcvr for CLUSTERA')

DEF CHL(CLUSTERB.QM4AB) +
CHLTYP(CLUSRCVR)CLUSTER('CLUSTERB') +
CONNNAME(QM4AB's address) +
DESCR('Cluster rcvr for CLUSTERB')

DEF CHL(CLUSTERA.QM1A) CHLTYP(CLUSSDR) +
CLUSTER(CLUSTERA) +
CONNNAME(QM1A's address) +
DESCR('Cluster sender for CLUSTERA')

DEF CHL(CLUSTERB.QM1B) CHLTYP(CLUSSDR) +
CLUSTER(CLUSTERB) +
CONNNAME(QM1B's address) +
DESCR('Cluster sender for CLUSTERB')
```

Definitions for QM4AB



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-15. Bridges: Joining more than one cluster

The figure shows the commands for defining a namelist that combines CLUSTERA and CLUSTERB.

Defining a namelist that combines CLUSTERA and CLUSTERB and attaching that namelist to the queue manager REPOSNL makes the queue manager a full repository for both clusters.

This example also shows how to define separate cluster-sender and cluster-receiver channels for the bridge queue manager.

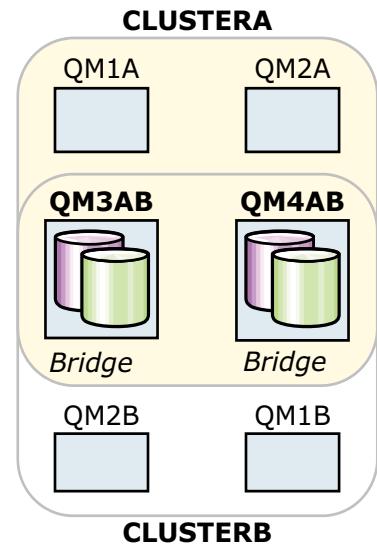
Bridges: Two bridge queue managers

```
DEFINE NAMELIST(CLUSTERS) NAMES(CLUSTERA,CLUSTERB)

ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)

DEF CHL(TO.QM4AB) CHLTYPE(CLUSRCVR) +
CLUSTER(' ') CLUSNL(CLUSTERS) +
CONNNAME(QM4AB's address) +
DESCR('Cluster receiver for CLUSTERS')

DEF CHL(TO.QM3AB) +
CHLTYPE(CLUSSDR) +
CLUSTER(' ') CLUSNL(CLUSTERS) +
CONNNAME(QM3AB's address) +
DESCR('Cluster sender for CLUSTERS')
```



- Definitions for QM4AB to join CLUSTER A and CLUSTER B that use a name list for cluster-sender and cluster-receiver channels
- Optionally, can define unique cluster-sender and cluster-receiver channels
- Similar definitions would be made for QM3AB

[Cluster design considerations](#)

© Copyright IBM Corporation 2017

Figure 8-16. Bridges: Two bridge queue managers

This example shows two bridge queue managers.

In this example, the bridge queue managers contain the full repositories for both clusters. In a real application, these queue managers might be dedicated to managing the clusters and not have any application queues defined.

You might choose to set the full repositories in separate facilities. For example, CLUSTER A might be in Dallas with QM3AB and CLUSTER B might be in Munich with QM4AB. The two sites can be duplicates of each other regarding application queues.

Put across clusters with a bridge queue manager (1 of 2)

- Use a name list in queue definition to advertise queue to both clusters

Example:

```
DEFINE NAMELIST(CLUSTERS) NAMES(CLUSTERA,CLUSTERB)
DEF QL(Q4AB) CLUSNL(CLUSTERS)
```

- Create alias queue on the bridge queue manager to allow queue in one cluster to be accessed by naming the queue on the MQOPEN call for putting of replies

Example:

```
DEFINE QALIAS(Q1A) CLUSNL(CLUSTERS) TARGET(Q1) +
TARGTYPE(QUEUE) DEFBIND(NOTFIXED)
```

- Add the cluster queue manager alias definition for all the clustered queue managers to the gateway queue

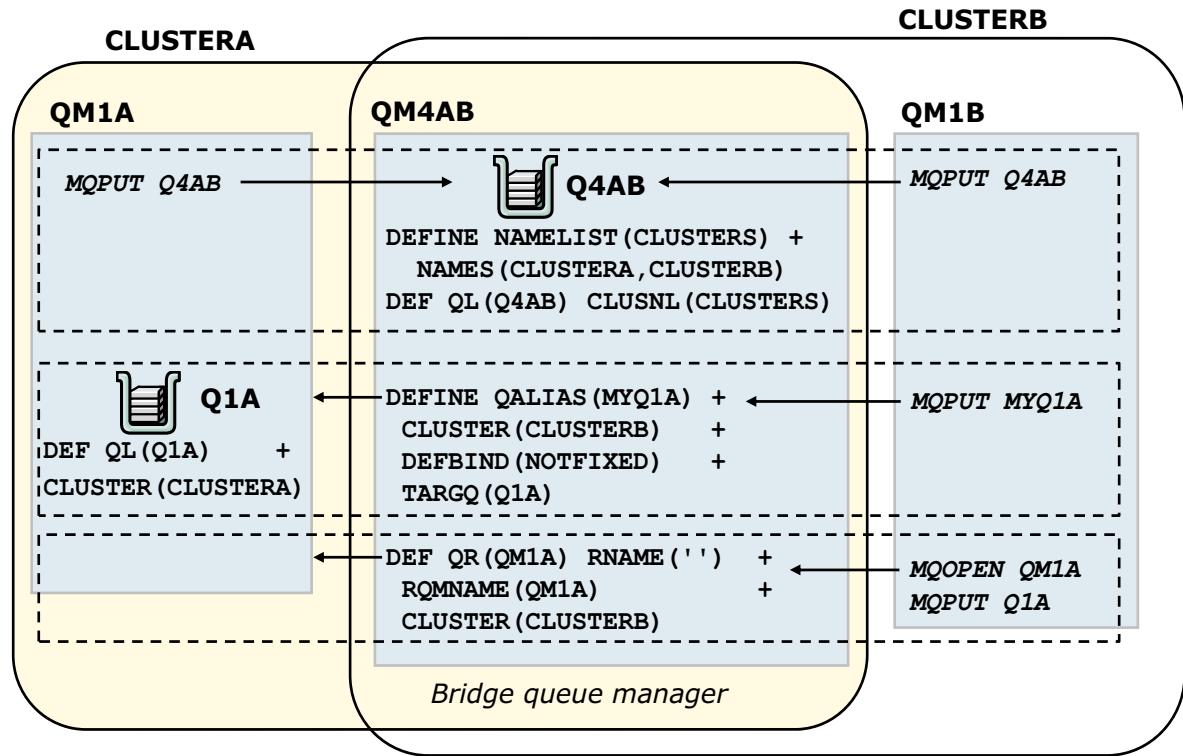
Example:

```
DEF QR(QM1A) RNAME('') RQMNAME(QM1A) +
CLUSTER(CLUSTERB)
```

Figure 8-17. Put across clusters with a bridge queue manager (1 of 2)

This figure summarizes the configuration steps for putting a message across clusters by using a bridge queue manager.

Put across clusters with a bridge queue manager (2 of 2)



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-18. Put across clusters with a bridge queue manager (2 of 2)

As shown in the top of the figure, by using a namelist in the queue definition, queue Q4AB is advertised to both CLUSTER A and CLUSTER B.

In the bottom of the figure, queue Q1A is made visible in CLUSTER B by the use of a QALIAS in the bridge queue manager.



Note

When you open a queue, you need to set DEFBIND to either NOTFIXED or QDEF. If DEFBIND is left as the default, OPEN, the queue manager resolves the alias definition to the bridge queue manager that hosts it. The bridge does not forward the message.

The queue manager alias on the bridge queue manager allows any queue on QM1A to be accessed by naming QM1A explicitly on the MQOPEN call. This technique is useful for supporting the putting of replies.

Gateway queue managers

- Avoids a difficult-to-manage network of point-to-point channels
- Provides a good place to manage such issues as security policies
- Two methods for implementing a gateway queue manager:
 - Place one or more queue managers in both clusters that use a second cluster-receiver channel and share administration of both clusters
 - Pair a queue manager in a cluster with an external queue manager by using traditional point-to-point channels

Cluster design considerations

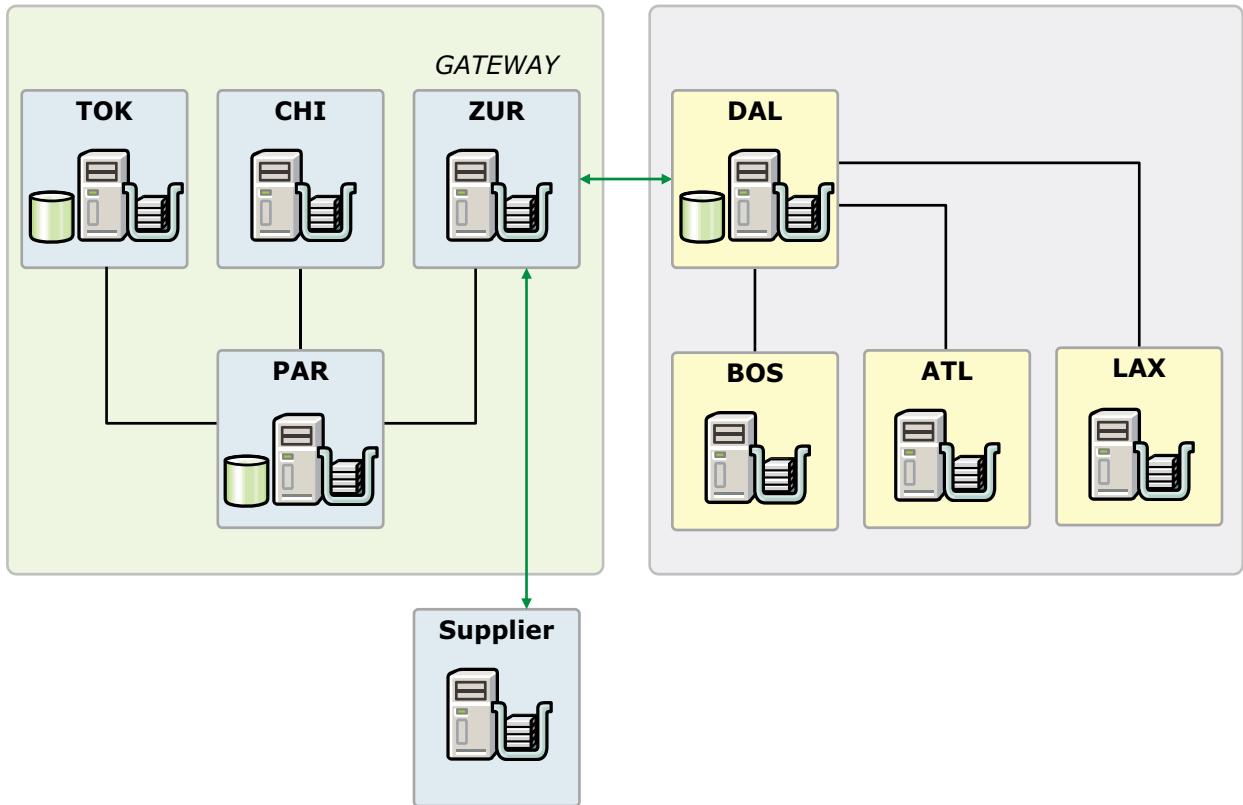
© Copyright IBM Corporation 2017

Figure 8-19. Gateway queue managers

A gateway queue manager can be used to allow queue managers that are not in the cluster to access resources in the cluster.

Placing one or more queue managers in both clusters and defining a second cluster-receiver channel requires fewer administrative definitions but means that administrators of both clusters share ownership.

Cluster gateway with paired queue managers



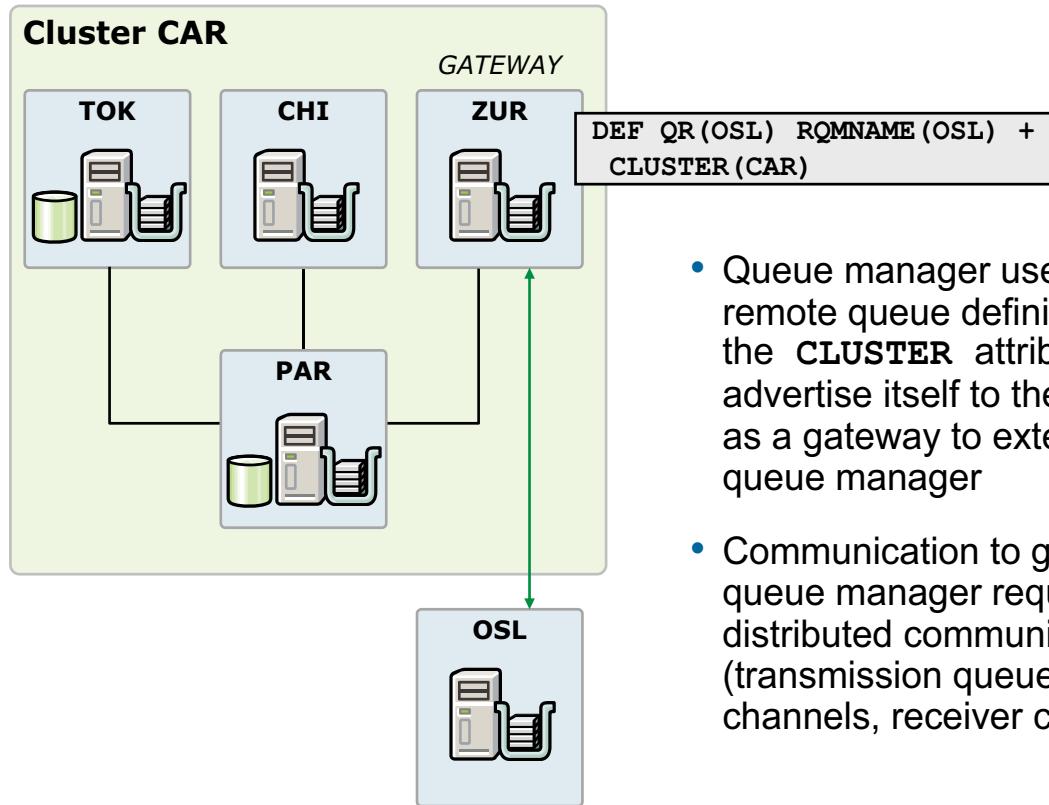
Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-20. Cluster gateway with paired queue managers

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.

Gateways to and from clusters



Cluster design considerations

© Copyright IBM Corporation 2017

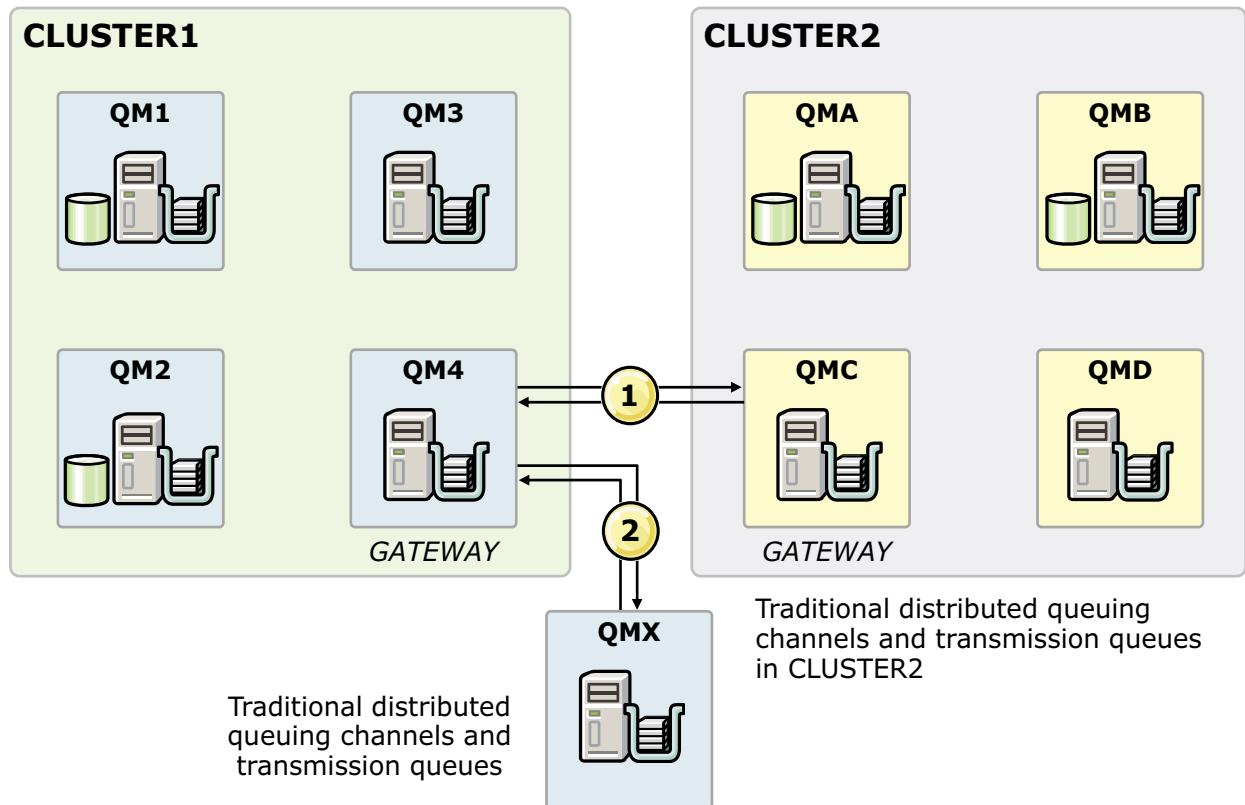
Figure 8-21. Gateways to and from clusters

It is possible for one or more of the queue managers in a cluster to advertise themselves as gateways to a remote queue manager by creating cluster remote queues.

The communication between the queue managers outside the cluster and the gateways is the same as in distributed communications. It requires transmission queues, and sender and receiver channels.

In the example, the definitions to reach the queue manager OSL are made in the queue manager ZUR. They are advertised to the whole cluster by adding the cluster name to the CLUSTER attribute of the remote queue definitions that point to the OSL queue manager.

Connected gateway queue managers



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-22. Connected gateway queue managers

In this example, queue manager QMX is connected to a gateway queue manager QM4 on cluster CLUSTER1. The gateway queue manager on CLUSTER1 is also connected to the gateway queue manager on CLUSTER2.

With this configuration, CLUSTER2 can communicate with queue manager QMX by passing messages to QM4 on CLUSTER1. QM4 can then send the message to QMX.

Put outside the cluster

- Option 1: Use a queue manager alias to remap the queue manager name when sending messages and advertise the alias to the whole cluster by adding the CLUSTER attribute to the remote-queue definition
 - When an application on the queue manager that made the definition puts a message to QREMOTE queue manager, the local queue manager resolves the name to RQNAME
 - If the local queue manager is not called RQNAME, put the message on the cluster transmission queue to be moved to RQNAME

Example: `DEF QREMOTE (QM1) RNAME (' ') RQNAME (QM1) +
XMITQ (QM1.XMITQ) CLUSTER (CLUSTER1)`

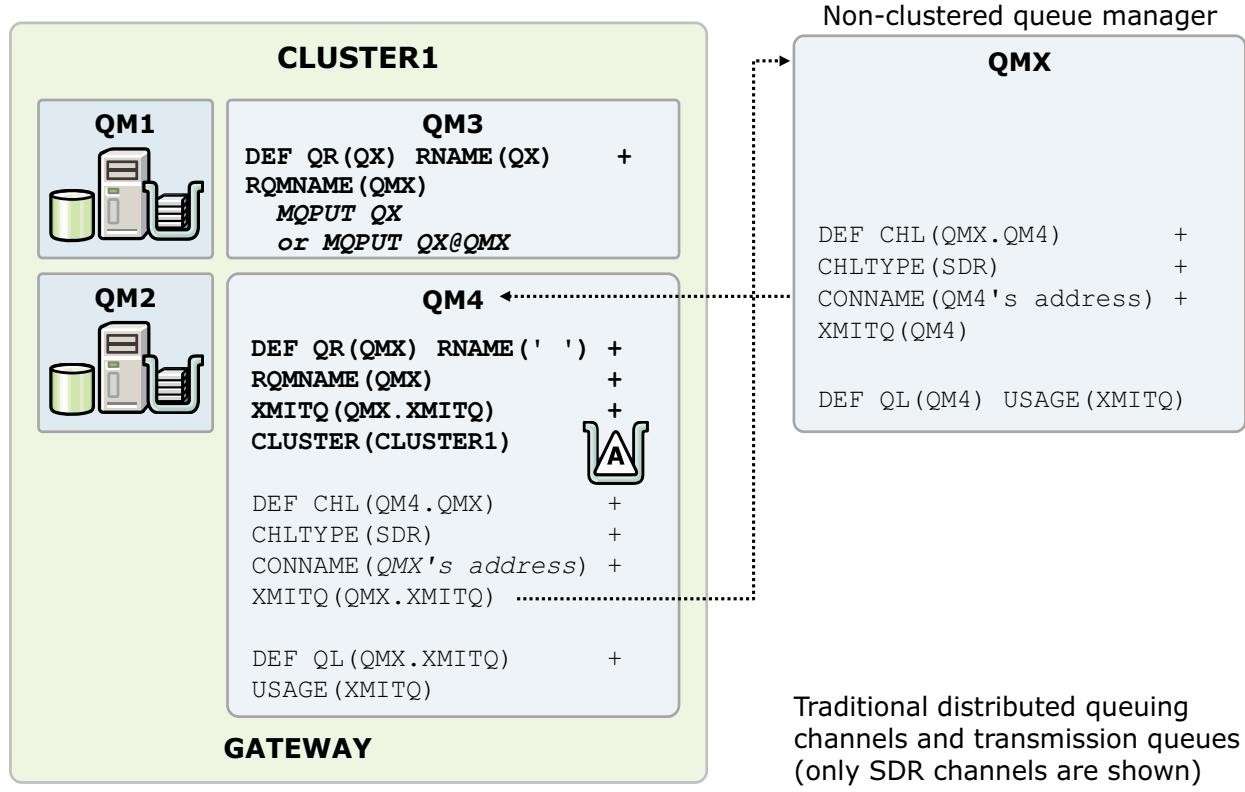
- Option 2: Create a QREMOTE definition on gateway queue manager that advertises the remote queue to the cluster

Example: `DEFINE QREMOTE (Q3) RNAME (Q3) RQNAME (QM3) +
CLUSTER (DEMO)`

This figure summarizes the configuration options for putting a message to a queue manager that is outside of the cluster.

Examples of both options are provided on the next pages. The examples assume that standard distributed queuing configuration is complete between the external queue manager and the gateway queue manager.

Put outside the cluster by using queue manager alias



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-24. Put outside the cluster by using a queue manager alias

This example shows how to allow access to all QMX queues throughout CLUSTER1. In the example, the CLUSTER parameter on the definition of QR(QMX) on QM4 allows access to the cluster.

When a message arrives at a queue manager, the transmission header is interrogated to determine where the message must go. If the queue manager in the header matches a queue manager alias definition, the queue manager substitutes the RQMNAME from its alias definition into the transmission header.

It is possible to use this method to direct messages to another queue manager or to ensure that the queue manager name points to the local queue manager.

Put outside the cluster by using a clustered QREMOTE

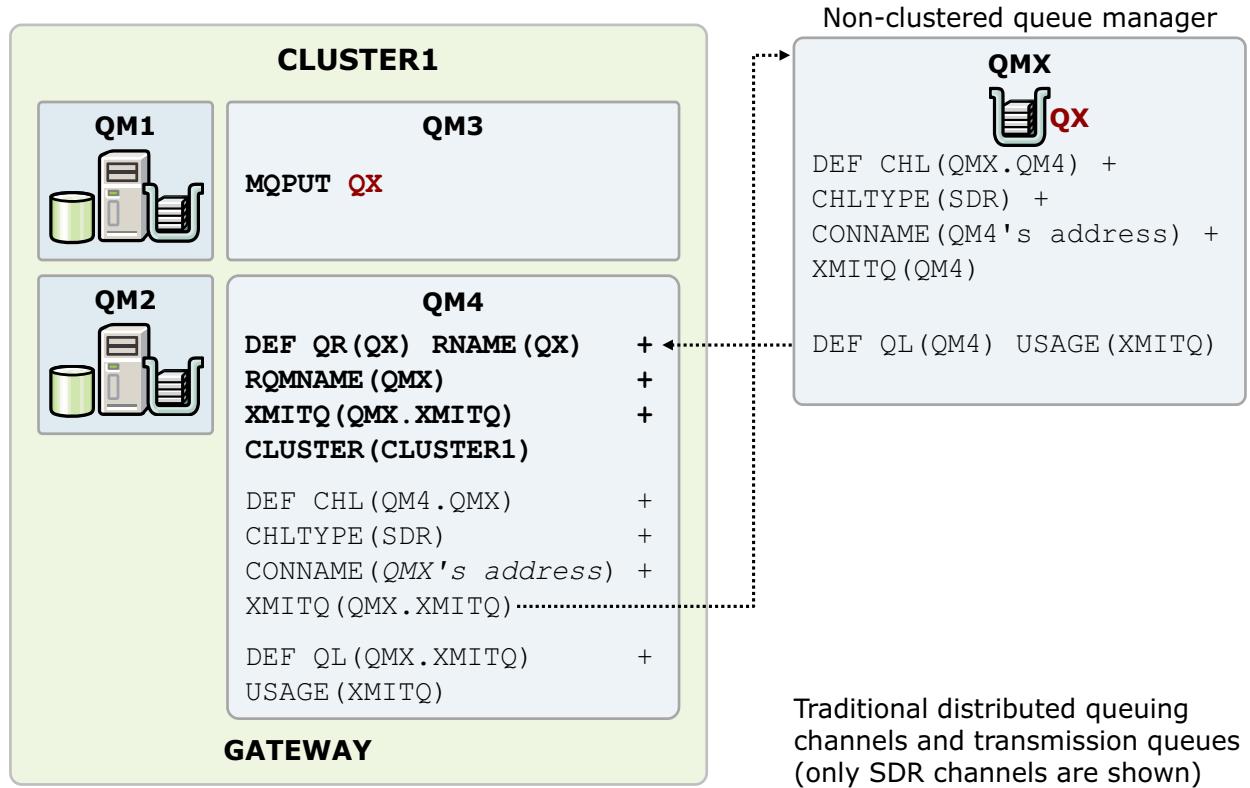


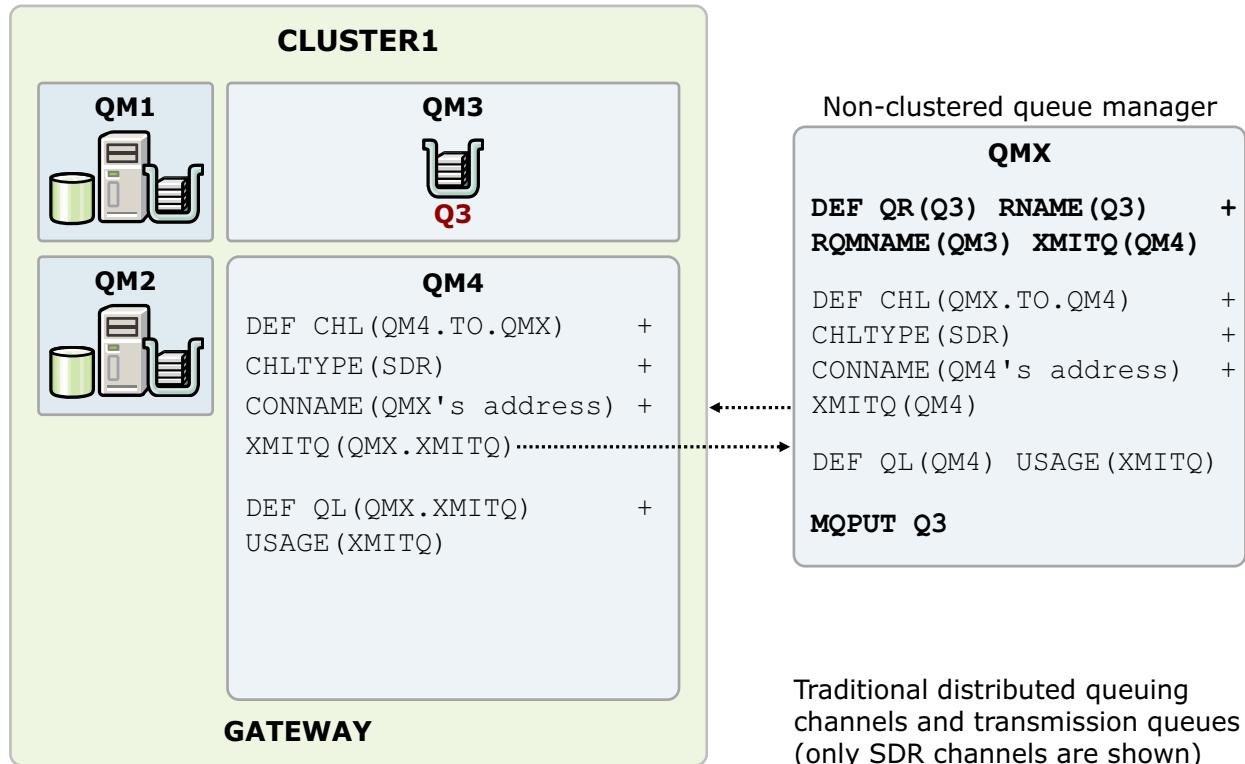
Figure 9.25 But what

© Copyright IBM Corporation 2017

Figure 8-25. Put outside the cluster by using a clustered QREMOTE

This example shows putting outside the cluster with a clustered remote queue (QREMOTE). This technique allows access to the single QMX queue that is named QX throughout CLUSTER1.

Put into the cluster by using QREMOTE



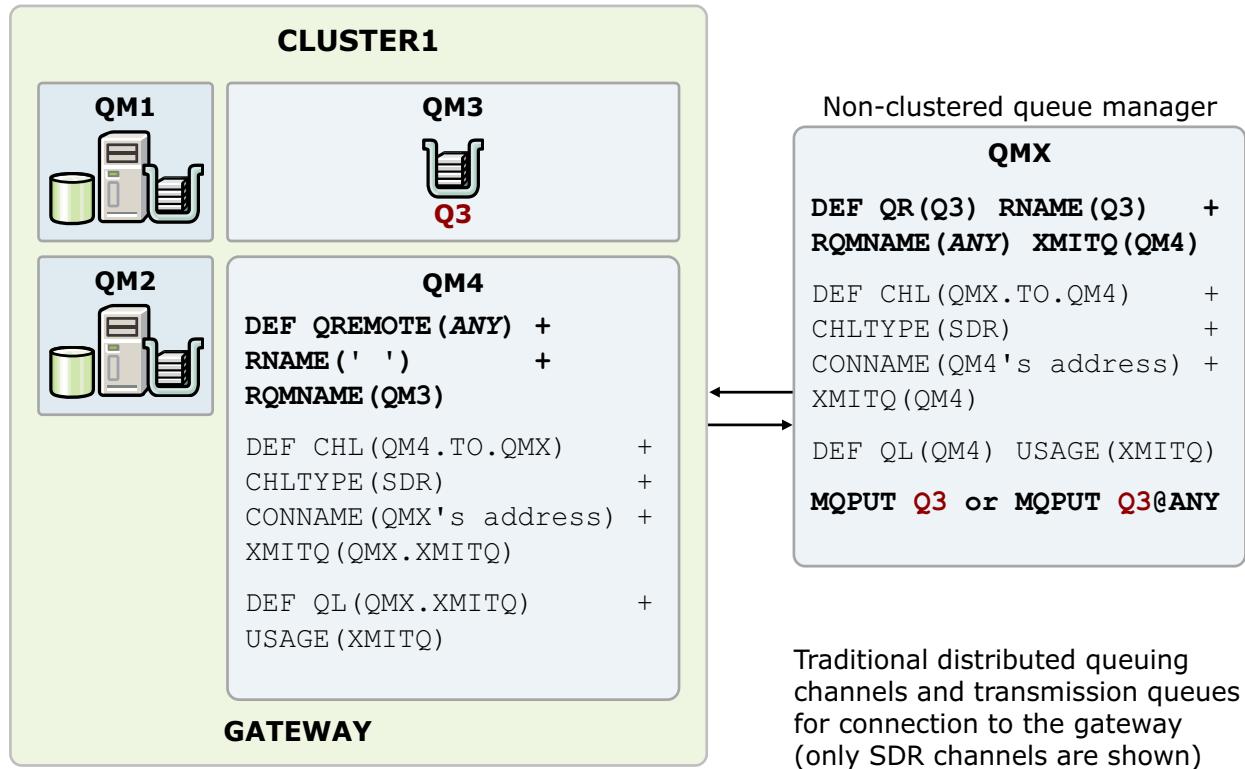
Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-26. Put into the cluster by using QREMOTE

As shown in this example, when the message arrives at QM4, QM4 inspects the transmission queue header and recognizes QM3 as a cluster member. The message is then forwarded to QM3 by putting the message to the SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Put into the cluster by using a queue manager alias



Cluster design considerations

© Copyright IBM Corporation 2017

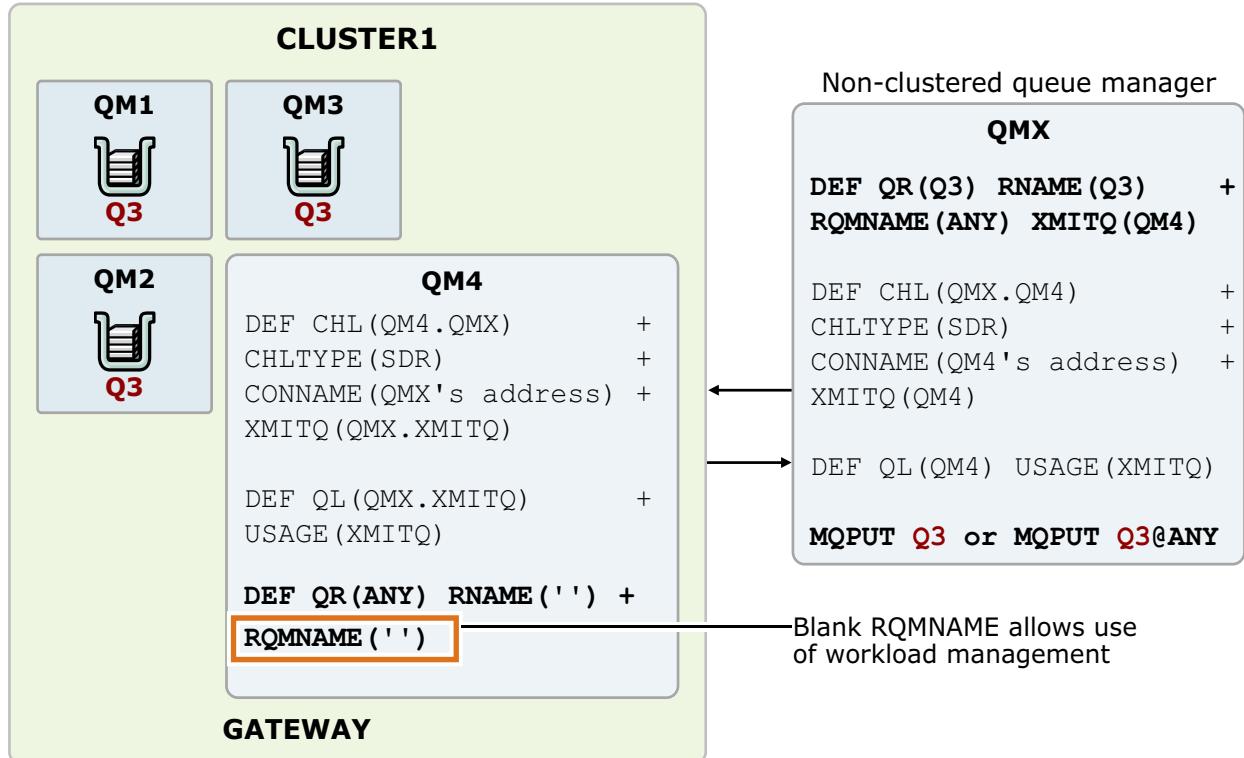
Figure 8-27. Put into the cluster by using a queue manager alias

When a queue manager receives a message, it extracts the name of the destination queue and queue manager from the transmission header. It looks for a queue manager alias definition with the same name as the queue manager in the transmission header. If it finds one, it substitutes the RQMNAME from the queue manager alias definition for the queue manager name in the transmission header.

When messages arrive at QM4, QM4 inspects the transmission queue header, recognizes ANY as a queue manager alias, and maps the queue manager name to null. The messages are then routed to any queue manager in the cluster that hosts an instance of Q3. The gateway should not have an instance of the destination queue.

For more examples, see the “Queue manager aliases and clusters” topic in the IBM Knowledge Center.

Use workload management when putting to a cluster



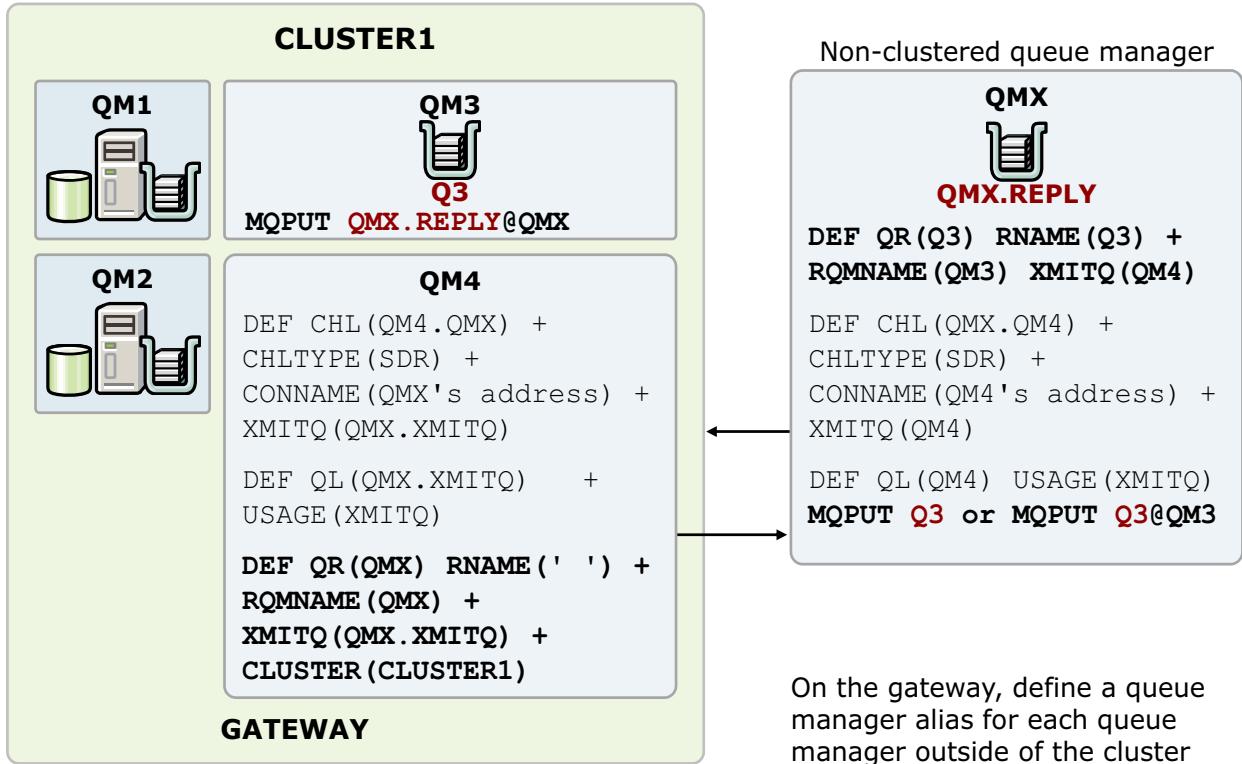
Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-28. Use workload management when putting to a cluster

When messages arrive at QM4, QM4 inspects the transmission queue header, recognizes ANY as a queue manager alias, and maps the queue manager name to null. The messages are then routed to any queue manager in the cluster that hosts an instance of Q3. The gateway should not have an instance of the destination queue.

Reply to a queue manager outside the cluster



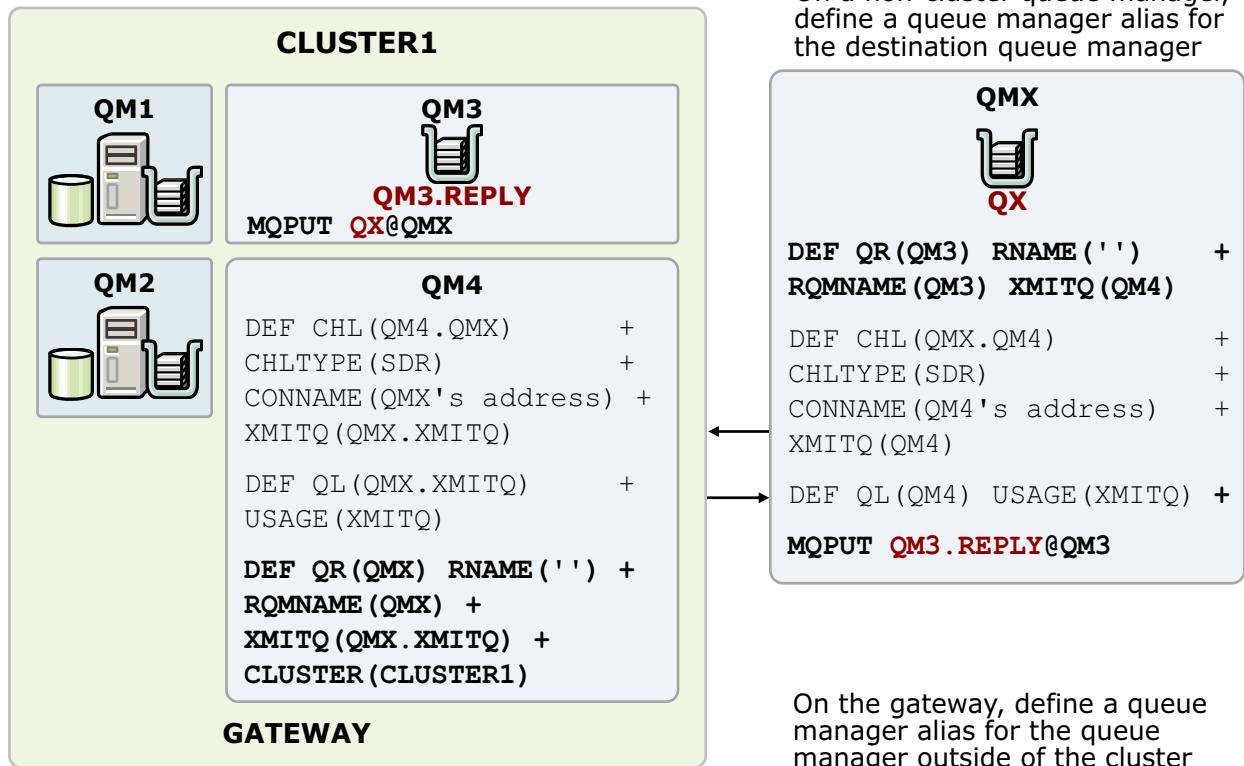
Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-29. Reply to a queue manager outside the cluster

For each queue manager outside the cluster that needs to receive replies, a queue manager alias must be defined on a gateway and advertised to the cluster.

Reply from a queue manager outside the cluster



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-30. Reply from a queue manager outside the cluster

A queue manager outside the cluster must have a queue manager alias for each queue manager in the cluster to which it sends a message. The queue manager alias must also specify the name of the transmission queue to the gateway queue manager.

In this example, QMX needs a queue manager alias definition for QM3 (the destination), and must also specify the transmission queue for the gateway queue manager (QM4):

```
DEFINE QR(QM3) RNAME('') RQMNAME(QM3) XMITQ(QM4)
```

For each queue manager inside the cluster that needs to receive replies, a queue manager alias must be defined on the queue manager outside the cluster.

Clustering channel rank between clusters

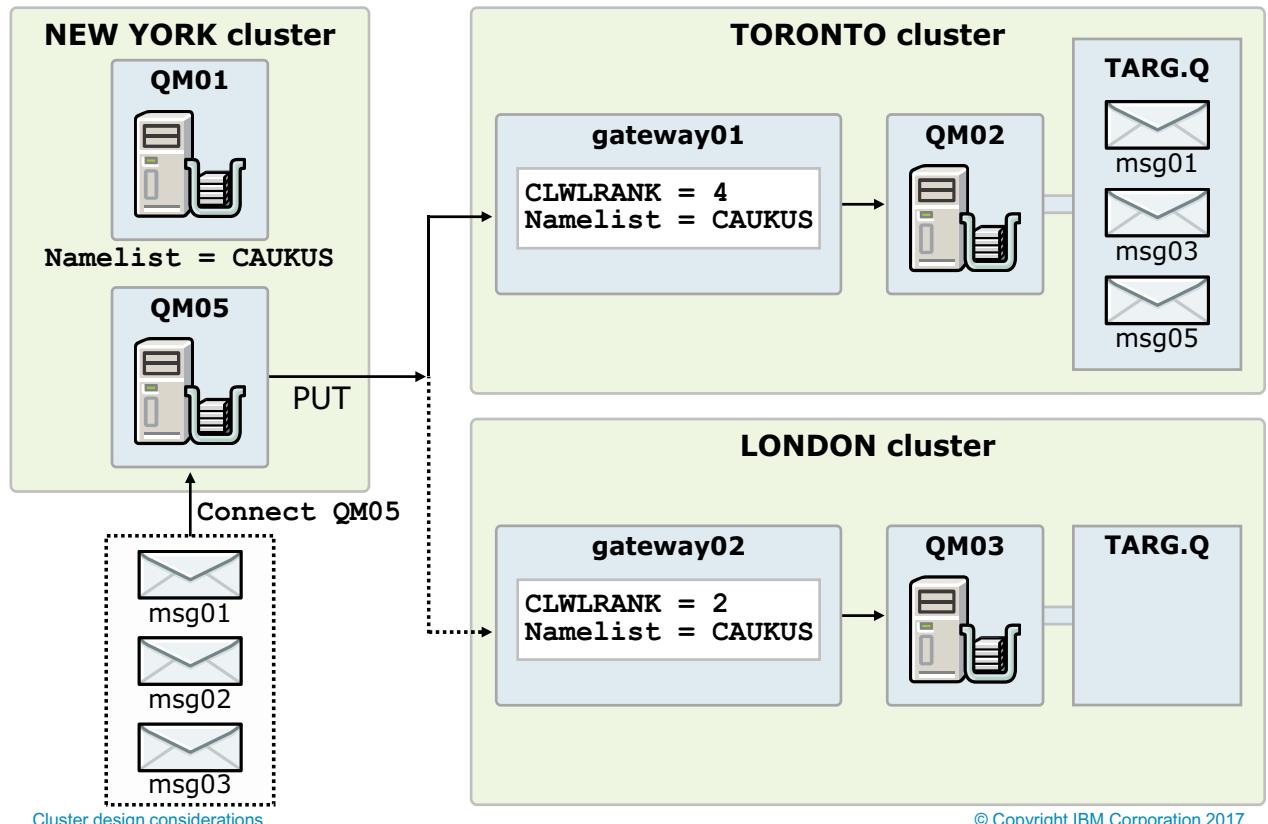


Figure 8-31. Clustering channel rank between clusters

This example shows workload management between multiple clusters by using the CLWL rank.

In this example, CLWL rank is used at the queue manager sender channel across multiple clusters. You can set a high priority cluster path from within a namelist attribute on the sending channel, which allows a queue manager to be a part of multiple clusters.

If you want more specific control over the final destination for messages that are sent to a queue manager in another cluster, use the rank attribute. Set the rank of the gateway queue managers at the intersection of the clusters so that messages take a specified route through the interconnected clusters towards a preferred destination.

In the example, messages arrive at a gateway queue manager, QM05. The gateway queue manager can send the messages to QM02 with a rank of 4, or QM03 with a rank of 2. The messages are automatically sent to the queue manager with the highest rank, in this case QM02. IBM MQ obtains the rank of queue managers before checking channel status.

Rank even allows for non-accessible queue managers for a message target, and allows messages to be routed through the network.

Unit summary

- Summarize environmental factors that might hinder the effectiveness of a complex cluster implementation
- Describe techniques to consider when dividing a large organization of systems
- Explain how to distinguish among classes of service when designing a clustering solution
- Explain how to define and manage an overlapping cluster
- Describe how to define a cluster gateway in an overlapping cluster environment

Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-32. Unit summary

Review questions

1. What can be used to make a cluster object visible to another overlapped cluster? (choose two)
 - A. Cluster channel
 - B. Local queue
 - C. Aliases
 - D. Namelist

2. For connecting an isolated queue manager to a cluster, you must define: (choose two)
 - A. Cluster-receiver channel
 - B. Transmission queues
 - C. Queue manager alias
 - D. Sender and receiver channels



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-33. Review questions

Review answers

1. What can be used to make a cluster object visible to another overlapped cluster? (choose two)

- A. Cluster channel
- B. Local queue
- C. Aliases
- D. Namelist

The answer is C and D.

2. For connecting an isolated queue manager to a cluster, you must define: (choose two)

- A. Cluster-receiver channel
- B. Transmission queues
- C. Queue manager alias
- D. Sender and receiver channels

The answer is B and D.



Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-34. Review answers

Exercise: Configuring an overlapping cluster

© Copyright IBM Corporation 2017

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

Figure 8-35. Exercise: Configuring an overlapping cluster

Exercise objectives

- Implement overlapping clusters
- Verify and test the overlapping clusters

Cluster design considerations

© Copyright IBM Corporation 2017

Figure 8-36. Exercise objectives

Unit 9. Course summary

Estimated time

00:15

Overview

This unit summarizes the course and provides information for future study.

How you will check your progress

- Checkpoint

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 9-1. Unit objectives

Course objectives (1 of 2)

- Describe the basic IBM MQ components
- Identify which IBM MQ objects are used to impact routing in a cluster environment
- Identify who in your organization can impact the health of a cluster and the need for adequate communication
- Describe the correct role of a cluster in a highly available IBM MQ infrastructure
- Describe the differences and similarities between administering clustered and non-clustered IBM MQ environments
- Describe how to configure, verify, and troubleshoot an IBM MQ cluster
- Identify the various channels that are present in a cluster environment and how each is created
- Describe how to use separate transmission queues in a clustered queue manager
- Explain how to remove a queue manager from a cluster on a permanent or temporary basis

[Course summary](#)

© Copyright IBM Corporation 2017

Figure 9-2. Course objectives (1 of 2)

Course objectives (2 of 2)

- Explain IBM MQ connection authentication
- Explain IBM MQ channel authentication
- Describe IBM MQ object authorizations
- Explain how to troubleshoot security challenges in a cluster
- List ways to influence workload balancing in a cluster
- Describe the history and basic components of IBM MQ publish/subscribe
- Explain the considerations and details of implementing publish/subscribe in an IBM MQ clustered environment
- Describe cluster design architectural considerations
- Summarize the benefits of design and configuration simplicity in a cluster implementation
- Explain how to configure overlapping clusters

[Course summary](#)

© Copyright IBM Corporation 2017

Figure 9-3. Course objectives (2 of 2)

To learn more on the subject

- IBM Training website:
www.ibm.com/training
- “Clustering presentation,” by Morag Hughson
 - Good presentation about clustering, as well as examples of object definitions for routing in and out of a cluster
 - Use your favorite browser to search for string: Websphere MQ clustering confex Morag
- “Understanding high availability with WebSphere MQ,” by Mark Hiscock
 - While this article is dated May 2005, it remains applicable to today’s IBM MQ environment and contains an excellent background for Unit 2
 - Use your favorite browser to search for string:
Mark Hiscock understanding high availability with Websphere MQ
- Experience with implementation and use of z/OS queue sharing groups is available in course WM312, *IBM MQ V8 Advanced System Administration for z/OS*
- IBM Knowledge Center

[Course summary](#)

© Copyright IBM Corporation 2017

Figure 9-4. 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

[Course summary](#)

© Copyright IBM Corporation 2017

Figure 9-5. 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 9-6. Unit summary

IBM Training 

Course completion

You have completed this course:
Designing, Implementing, and Managing IBM MQ V9 Clusters
Any questions?



[Course summary](#)

© Copyright IBM Corporation 2017

Figure 9-7. Course completion

Appendix A. List of abbreviations

ACL	Access control list
AMQP	Advanced Message Queuing Protocol
API	application programming interface
ASCII	American Standard Code for Information Interchange
CCSID	coded character set identifier
CICS	Customer Information Control System
CLWL	cluster workload
DLQ	dead-letter queue
DN	distinguished name
DPL	distributed program link
EJB	Enterprise JavaBeans
ESM	external security manager
IBM	International Business Machines Corporation
ID	identifier
IMS	Information Management System
IP	Internet Protocol
ISPF	Interactive System Productivity Facility
IT	information technology
HTTP	Hypertext Transport Protocol
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Platform, Standard Edition
JAAS	Java Authentication and Authorization Service
JAF	JavaBeans Activation Framework
JCL	job control language
JDBC	Java Database Connectivity
JMS	Java Message Service
JTA	Java Transaction API
LDAP	Lightweight Directory Access Protocol
MCA	message channel agent
MQ	Message Queue
MQI	Message Queue Interface
MQMD	Message Queue Message Descriptor
MQSC	MQ script

MQTT	MQ Telemetry Transport
OAM	object authority manager
OS	operating system
PaaS	platform as a service
PCF	Programmable Command Format
QM	queue manager
QMID	queue manager ID
RACF	Resource Access Control Facility
RFH	rules and formatting header
SCM	storage class memory
SMDS	shared message data set
SOA	service-oriented architecture
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TCP	Transmission Control Protocol
z/OS	z Series operating system



IBM Training

IBM
®

© Copyright International Business Machines Corporation 2017.