

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

Maximize your investment in
WebSphere MQ

Discover new features that bring
value to your business

Learn from scenarios with
sample configurations



Cezar Aranha
Craig Both
Barry Dearfield
Carolyn Elkins
Alexander Ross
Jamie Squibb
Mark Taylor

Redbooks



International Technical Support Organization

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

February 2013

Important: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (February 2013)

This edition applies to Version 7, Release 1, Modification 0 and Version 7, Release 5, Modification 0 of WebSphere MQ for Multiplatform (product number 5724-H72) and to Version 7, Release 1, Modification 0 of WebSphere MQ for z/OS (product number 5655-R36).

© Copyright International Business Machines Corporation 2013. All rights reserved.

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

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team who wrote this book	xiv
Now you can become a published author, too!	xvi
Comments welcome	xvi
Stay connected to IBM Redbooks	xvi
Part 1. Introduction	1
 Chapter 1. Overview	3
1.1 Executive summary	4
1.2 The scope of this book	4
1.3 Intended audience	4
1.4 What is covered in this book	5
1.5 What is not covered by this book	5
1.6 Other related publications	6
 Chapter 2. Concepts of messaging	7
2.1 The business case for message-oriented middleware	8
2.2 Application simplification	10
2.3 Example scenarios	11
2.3.1 Retail kiosks	11
2.3.2 Faster bank payments	11
2.3.3 Airport information	11
2.4 Messaging in service-oriented architecture	12
 Chapter 3. Introduction to WebSphere MQ	13
3.1 Messaging with WebSphere MQ	14
3.1.1 A history of WebSphere MQ	15
3.2 Core concepts of WebSphere MQ	16
3.2.1 Asynchronous messaging	17
3.2.2 WebSphere MQ clients	18
3.2.3 WebSphere MQ Telemetry clients	18
3.2.4 Application programming interfaces	19
3.2.5 Reliability and integrity	20
3.2.6 WebSphere MQ messaging styles	21
3.2.7 WebSphere MQ topologies	22
3.2.8 Availability	27
3.2.9 Security	27
3.2.10 Management and monitoring	28
3.3 Diverse platforms	29
3.4 Relationships with other products	30
3.4.1 WebSphere Message Broker	30
3.4.2 WebSphere Application Server	31
 Chapter 4. Getting started with WebSphere MQ	33

4.1	Messages	34
4.1.1	Message descriptor.....	34
4.1.2	Message properties	36
4.2	WebSphere MQ Objects	36
4.2.1	Queue manager	36
4.2.2	Queues	37
4.2.3	Topics	39
4.2.4	Channels.....	39
4.3	Configuring WebSphere MQ	41
4.3.1	Creating a queue manager.....	41
4.3.2	Managing WebSphere MQ objects	42
4.4	Writing applications	43
4.5	Triggering	48
4.6	Configuring a WebSphere MQ client.....	51
4.7	Security	53
4.8	Configuring communication between queue managers	54
4.8.1	How to define a connection between two systems.....	56
4.9	Summary	58
	Chapter 5. What is new in WebSphere MQ V7.1 and V7.5	59
5.1	What is new in WebSphere MQ V7.1	60
5.1.1	Multiple installation support.....	60
5.1.2	Enhanced security	60
5.1.3	Enhanced clustering	61
5.1.4	Multicast protocol for publish/subscribe	61
5.1.5	Improved availability and scalability on z/OS	61
5.1.6	Improved performance on distributed platforms	61
5.1.7	Management of distributed platforms	62
5.1.8	Support for cloud environments	62
5.1.9	Application activity reports	62
5.1.10	Clients	62
5.1.11	Channels.....	63
5.2	What is new in WebSphere MQ V7.5	63
5.2.1	Integrated installation	64
5.2.2	Enhanced clustering	64
5.2.3	Java application identification	64
5.2.4	Simplified AMS integration	64
5.2.5	RFC 5280 certificate validation policy.....	64
5.2.6	Managed File Transfer enhancements	65
	Part 2. WebSphere MQ V7.1 new features and enhancements	67
	Chapter 6. WebSphere MQ V7.1 product installation enhancements	69
6.1	WebSphere MQ Telemetry integrated feature	70
6.1.1	WebSphere MQ Telemetry functional overview	70
6.1.2	WebSphere MQ V7.1 installation with MQTT	71
6.2	Relocatable installations for UNIX and Linux	72
6.2.1	Why relocatable installations are important for WebSphere MQ V7.1	72
6.2.2	How to install into a non-default location.....	72
6.2.3	Further considerations for custom location installations	74
	Chapter 7. Multiple installation support on Windows, UNIX, and Linux	75
7.1	Managing installations	76
7.1.1	Installing multiple copies of WebSphere MQ	76

7.1.2 Applying maintenance-level upgrades	78
7.1.3 Working with installations	79
7.1.4 Primary installation	81
7.1.5 Managing the environment	82
7.2 Managing queue managers	84
7.2.1 Displaying queue managers	84
7.2.2 Administering queue managers	85
7.2.3 Migrating a queue manager to another installation	85
7.2.4 Using WebSphere MQ Explorer	86
7.3 Managing applications	88
7.3.1 Environment considerations	88
7.3.2 Connecting to queue managers	95
7.3.3 Using .NET applications	96
7.3.4 Exit and installable service considerations	96
7.3.5 Other restrictions	97
Chapter 8. Enhanced security	99
8.1 Controlling access on channels	100
8.1.1 The role of enhanced channel access control in WebSphere MQ	100
8.1.2 Blocking based on IP address (BLOCKADDR)	102
8.1.3 Message channel agent user ID mapping	102
8.1.4 Blocking based on user ID (BLOCKUSER)	105
8.1.5 Configuration of channel authentication records	105
8.1.6 IP address pattern matching	117
8.1.7 SSLPEER pattern matching	118
8.1.8 Verifying channel authentication records	121
8.2 Remote queue access control	123
8.2.1 Access permissions on remote objects before WebSphere MQ V7.1	123
8.2.2 Access permissions on remote objects in WebSphere MQ V7.1	125
8.3 Enhanced cryptographic support with IBM Global Security Kit V8	127
8.3.1 Local installation of GSKit 8	127
8.3.2 WebSphere MQ commands for certificate management	128
8.3.3 Increased CipherSpec support	128
8.3.4 Extended distinguished name attribute support	129
8.3.5 Enhanced cryptographic hardware support on Windows, UNIX, and Linux	129
8.4 Cryptographic standards compliance	130
8.4.1 FIPS 140-2 compliance on Windows, UNIX, and Linux	130
8.4.2 SUITEB compliance on Windows, UNIX, and Linux	131
8.4.3 FIPS 140-2 compliance on z/OS	131
Chapter 9. Granular control over dead-letter queue usage	133
9.1 Channel message delivery	134
9.2 Publication delivery for topics	135
9.3 The USEDLQ attribute	136
9.3.1 USEDLQ channel attribute	136
9.3.2 USEDLQ topic attribute	136
9.4 Defining USEDLQ on channels	137
9.4.1 Setting USEDLQ on channels by using WebSphere MQ Explorer	137
9.4.2 Setting USEDLQ on channels by using MQSC commands	138
9.4.3 Setting USEDLQ on channels by using ISPF panels	139
9.5 Defining USEDLQ on topics	141
9.5.1 Setting USEDLQ on topics by using WebSphere MQ Explorer	141
9.5.2 Setting USEDLQ on topics by using MQSC commands	142

Chapter 10. Dumping and restoring a queue manger configuration	143
10.1 Dumping a queue manager	144
10.1.1 Dumping WebSphere MQ objects	147
10.1.2 Dumping WebSphere MQ object access authorizations	149
10.2 Restoring a queue manager configuration	152
10.2.1 Restoring WebSphere MQ objects by using MQSC commands	152
10.2.2 Restoring WebSphere MQ object access authorities	153
Chapter 11. Shared message data set and message offloading (z/OS)	155
11.1 Shared message data sets	156
11.1.1 Support for large messages on shared queues	156
11.1.2 Queue manager behavior for large messages	159
11.1.3 Defining shared message data sets	161
11.1.4 Migration from DB2	166
11.2 Offload rules	167
11.2.1 Offload definition	168
11.3 Other SMDS commands	169
11.4 Monitoring	170
11.4.1 DISPLAY CFSTATUS command	170
11.4.2 DISPLAY USAGE command	174
11.4.3 New messages	175
11.4.4 New System Management Facility (SMF) data	180
11.5 Recovery	182
11.6 Application impact	183
Chapter 12. Coupling facility connectivity loss improvements (z/OS)	185
12.1 Resiliency	187
12.2 Coupling facility structure failures	187
12.3 What happens when a failure is reported for the administration structure	191
12.4 What happens when a failure is reported for an application structure	192
12.5 Actions to pursue when coupling facility connectivity is lost	192
12.6 Automatic application structure recovery	195
12.7 RESET CFSTRUCT ACTION(FAIL) command	195
12.8 Enabling resilience	196
12.9 Queue manager reactions to loss of connectivity for coupling facility structures	199
12.9.1 Loss of connectivity to the administration structure	199
12.9.2 Loss of connectivity to the application structure	199
12.9.3 Partial loss of connectivity to an application structure	200
12.9.4 Total loss of connectivity to an application structure	200
Chapter 13. Extended integration with IMS (z/OS)	203
13.1 Resource monitoring	204
13.1.1 New console messages	205
13.1.2 Using shared queues	206
13.2 Requesting a response with commit mode 0	206
Chapter 14. CSQINPT DD added to queue manager startup JCL (z/OS)	209
14.1 Introducing CSQINPT	210
14.2 Changes to commands	211
14.3 New sample initialization input data sets	211
Chapter 15. CICS 4.2 group attach (GROUPUR)	213
15.1 Extending the availability of WebSphere MQ to CICS	214
15.2 Enabling Group units of recovery	220

Part 3. WebSphere MQ 7.5 new features and enhancements	229	
Chapter 16. Installation enhancements in WebSphere MQ V7.5		231
16.1 WebSphere MQ Advanced Message Security installation	232	
16.1.1 WebSphere MQ Advanced Message Security overview	232	
16.1.2 Installing AMS with WebSphere MQ V7.5	232	
16.2 WebSphere MQ Managed File Transfer installation	233	
16.2.1 WebSphere MQ Managed File Transfer overview	233	
16.2.2 Installing MFT with WebSphere MQ V7.5	233	
16.3 Extended transactional client installation	236	
Chapter 17. Clustering enhancements on Windows, UNIX, and Linux		237
17.1 Clustering enhancements overview	238	
17.1.1 Overview of changes to clustering	238	
17.1.2 Benefits of multiple transmission queues	238	
17.2 Enabling multiple cluster transmission queues	239	
17.2.1 Creating cluster transmission queues dynamically	239	
17.2.2 Defining cluster transmission queues manually	241	
17.3 Using the same transmission queue for multiple channels	244	
17.4 Priority of queue selection	245	
17.5 Switching the transmission queue that is used by a channel	245	
17.6 Switching channel transmission queues with runswchl	246	
17.7 Displaying the transmission queue that is used	247	
17.8 Reverting to the default configuration	248	
Chapter 18. Certificate validation policies		249
18.1 Digital certificate validation on Windows, UNIX, and Linux	250	
18.1.1 Basic path validation policy	251	
18.1.2 Standard path validation policy	251	
18.1.3 What changed in WebSphere MQ V7.5	252	
18.2 Configuration of strict RFC 5280 compliance	252	
18.2.1 Queue manager configuration	252	
18.2.2 Client application configuration	254	
Part 4. Scenarios	257	
Chapter 19. Coexistence: A staged migration on Windows, UNIX, and Linux		259
19.1 Initial configuration	260	
19.2 Preparing the scenario	261	
19.2.1 Installing WebSphere MQ V7.0.1	261	
19.2.2 Preparing the sample applications	261	
19.2.3 Creating the queue managers	262	
19.2.4 Creating the WebSphere MQ objects	262	
19.2.5 Verifying the initial configuration	264	
19.3 Installing WebSphere MQ V7.1	266	
19.3.1 Installing WebSphere MQ V7.1 on Windows	266	
19.3.2 Installing WebSphere MQ V7.1 on UNIX and Linux	267	
19.4 Preparing the first application	268	
19.5 Migrating the first queue manager	270	
19.6 Completing the migration	271	
19.7 Uninstalling WebSphere MQ V7.0.1	272	
19.8 Applying WebSphere MQ V7.1 fix pack 1	273	
19.8.1 Installing a second copy of WebSphere MQ V7.1	273	
19.8.2 Installing WebSphere MQ V7.1 fix pack 1	275	

19.8.3 Applying fix pack 1 to the queue managers	275
19.8.4 Removing fix pack 1 from the queue managers	276
Chapter 20. Channel authentication records: Controlling remote user activity	277
20.1 Environment overview	278
20.1.1 Assumptions	279
20.1.2 Administration and message routing overview	279
20.2 Machine configuration	281
20.2.1 Windows WebSphere MQ configuration details	281
20.2.2 Linux WebSphere MQ configuration details	293
20.2.3 z/OS WebSphere MQ configuration details	301
20.3 User authorities	308
20.3.1 User authorities on Windows	308
20.3.2 User authorities on Linux	309
20.4 Starting channels and following a message	309
20.5 Testing the BLOCKUSER channel authentication records	311
Chapter 21. Clustering: Multiple cluster transmission queues	313
21.1 Preparing the scenario	314
21.2 Enabling the automatic creation of transmission queues	319
21.3 Giving AP2 its own queue	321
21.4 Creating an overlapping cluster	323
21.5 Manually defining specific transmission queues with a manual switch	327
Chapter 22. Shared queues: Using the new capabilities	331
22.1 General scenario methodology	332
22.2 No offloading shared queue capacity test	334
22.3 SMDS offload: Using the default offload rules	341
22.4 SMDS offload: Large SMDS data sets	349
22.5 SMDS offload: Effects of offload rule changes	355
22.6 SMDS offload: Effects of 100% message body offload	362
22.7 SMDS offload: Mixed message sizes	368
22.8 DB2 to SMDS offload migration	370
22.9 SMDS offload: Message body availability	376
Chapter 23. GROUPUR: Using Group units of recovery with CICS	379
23.1 Preparing the scenario	380
23.2 Enabling indoubt units of work before Group units of recovery are enabled	382
23.3 Configuring the environment to support Group units of recovery	405
23.4 Enabling indoubt units of work after Group units of recovery are enabled	409
Chapter 24. Resiliency: Improving availability	417
24.1 Preparing the scenario	418
24.2 Coupling facility application structure issues	419
24.2.1 Partial loss of connectivity for one queue manager	419
24.2.2 Coupling facility structure failure	427
24.2.3 CFSTRUCT CFCONLOS attribute set to TERMINATE	430
24.3 Coupling facility administration structure failure	431
24.4 Loosing connectivity to coupling facility administration structure	436
Appendix A. MQSC scripts for the Coexistence scenario	445
Defining the objects for queue manager QM1	446
Defining the objects for queue manager QM2	447
Appendix B. WebSphere MQ for z/OS 7.1 System Management Facility changes ..	451

SMF 115 data changes	452
New shared message data sets SMF115 information: QESD record	452
New SMF115 fields in existing structures	455
CSQDQSST: Storage manager statistics structure	455
CSQDQWS1: Mapping macro for SMF 115 subtype 2.....	455
CSQDWQ - queue statistics mapping macro	455
CSQDWTAS: Task-related statistics mapping macro.....	455
Appendix C. Additional material	457
Locating the web material	457
Using the web material.....	458
System requirements for downloading the web material	458
Downloading and extracting the web material	458
Glossary	459
Related publications	465
IBM Redbooks	465
Other publications	465
Online resources	466
Help from IBM	467

Notices

This information was developed for products and services offered in the U.S.A.

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

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

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM 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 supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX®

CICS®

DB2®

IBM®

IMS™

MQSeries®

MVS™

Parallel Sysplex®

PowerHA®

PureApplication™

RACF®

Redbooks®

Redbooks (logo) ®

Smarter Planet®

System i®

System z®

WebSphere®

z/OS®

The following terms are trademarks of other companies:

Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

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

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication is divided into four parts:

- ▶ Part 1, “Introduction” on page 1 introduces message-oriented middleware and the WebSphere® MQ product. It explains how messaging technologies are implemented in WebSphere MQ and shows how to get started with configuring a WebSphere MQ environment. This part briefly lists the new features of WebSphere MQ V7.1 and V7.5.
- ▶ Part 2, “WebSphere MQ V7.1 new features and enhancements” on page 67 introduces the enhancements to WebSphere MQ in Version 7 Release 1. It provides a description of the new features, their business value, and usage examples. It describes enhancements to WebSphere MQ for multiplatforms and z/OS®. Examples of features that are discussed in this part include multiple installation support for multiplatforms, enhanced security with channel authentication records, enhanced clustering, improved availability and scalability on z/OS, and more.
- ▶ Part 3, “WebSphere MQ 7.5 new features and enhancements” on page 229 introduces the enhancements to WebSphere MQ in Version 7 Release 5 for multiplatforms. It provides a description of the new features, their business value, and usage examples. Examples of enhancements that are discussed in this part include new installation options, such as the bundling of WebSphere MQ Advanced Message Security and WebSphere MQ Managed File Transfer.
- ▶ Part 4, “Scenarios” on page 257 contains practical scenarios that demonstrate how the new features and enhancements work and how to use them.

In summary, the introduction gives a broad understanding of messaging technologies and WebSphere MQ. It helps you understand the business value of WebSphere MQ. It provides introductory information to help you get started with WebSphere MQ. No previous knowledge of the product and messaging technologies is assumed. The remaining parts of this book discuss enhancements to previous versions of WebSphere MQ. The information helps you understand the benefits of upgrading to WebSphere MQ V7.1 and V7.5 and how to implement the new functions. Knowledge of WebSphere MQ V7.0 and earlier versions is assumed.

This book provides details about IBM WebSphere MQ product features and enhancements that are required for individuals and organizations to make informed application and design decisions prior to implementing a WebSphere MQ infrastructure or begin development of a WebSphere MQ application. This publication is intended to be of use to a wide-ranging audience.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Cezar Aranha is a consultant for messaging integration middleware in the IBM Global Accounts (IGA) organization, in IBM Brazil. Cezar's areas of expertise include WebSphere MQ, WebSphere Message Broker, and WebSphere Transformation Extender. He is part of the Web Middleware Enablement team that supports several IBM internal customers. Cezar has 23 years of experience in the IT industry; he joined IBM in 2005 as an IT specialist. Prior to his IBM career, Cezar worked for a large international bank in Brazil as project manager and technical support. He is currently working on his MBA at the Universidad de Positivo in Brazil.



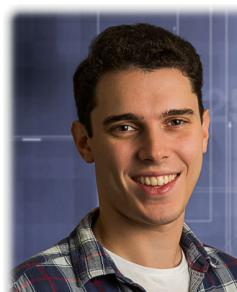
Craig Both is a Software Engineer at the IBM Hursley Laboratory in the UK. He has been with IBM for over 10 years, working with WebSphere products, specifically working in test teams for WebSphere MQ and WebSphere Enterprise Service Bus. During this time, he focused specifically on the security features of WebSphere MQ, and overall involvement in functional and system testing of the product as a whole. His current role is WebSphere MQ Level 3 Support, Team Lead.



Barry Dearfield is a consultant in the IBM STG Lab Services organization. Barry has more than 35 years of experience in IBM mainframes. His areas of expertise include CICS®, CICS tools, WebSphere MQ, WebSphere Message Broker, and programming in systems and application areas. He has updated the IBM assembler class and conducted online instructor led classes for a worldwide audience.



Carolyn (Lyn) Elkins is an IT Specialist for Advanced Technical Skills in the United States, with an emphasis on WebSphere MQ, WebSphere Message Broker, and WebSphere MQ-FTE on System z® hardware. She has over 25 years of experience in all phases of software design, development, testing, and operations. Lyn came to IBM as a professional hire, having experience at other software vendors and production environments. She has a degree in Computer Science from East Tennessee State University.



Alexander Ross is a software engineer at the IBM Hursley laboratory in England where he has worked for over 2 years in the WebSphere MQ development team. Alex has been involved in the last two releases and now works on developing WebSphere MQ for zOS. He holds a Masters of Engineering in Electronic and Electrical Engineering from the University of Portsmouth, UK.



Jamie Squibb is a Software Engineer at the IBM Hursley Laboratory in the UK. He is a team leader in the WebSphere MQ development organization, where he has worked since joining IBM as a graduate in 2001. His focus is predominantly on the System z platform, but for many years he developed and maintained the infrastructure that is used to test WebSphere MQ and WebSphere Message Broker on all platforms. He also helped to test support for multiple installations on Windows, Unix, and Linux. He has a degree in Computer Science from the University of Warwick in the UK.



Mark Taylor has worked for IBM at the Hursley laboratory in England for over 25 years, Mark has worked in various development and services roles. He wrote code for the early versions of MQSeries®, porting it to numerous UNIX operating systems. He now works in the Technical Strategy department where he is responsible for defining the functions that are included in new releases of WebSphere MQ.

Thanks to the following people for their contributions to this project:

- ▶ David McCann
WebSphere MQ development - IBM Hursley development laboratory
- ▶ Tony Sharkey
WebSphere MQ performance - IBM Hursley development laboratory
- ▶ Gwydion Tudur
WebSphere MQ development - IBM Hursley development laboratory
- ▶ Pete Siddall
WebSphere MQ development - IBM Hursley development laboratory
- ▶ Eugene Kuehlthau
IBM Advanced Technical Skills
- ▶ Mitch Johnson
IBM Advanced Technical Skills
- ▶ Angel Rivera
WebSphere MQ Distributed Platforms Level 2 Support - IBM Raleigh

The project that produced this publication was managed by Marcela Adan, IBM Redbooks Project Leader - International Technical Support Organization, Raleigh Center

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:
<http://www.ibm.com/redbooks/residencies.html>

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
<http://www.ibm.com/redbooks>
- ▶ Send your comments in an email to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



Part 1

Introduction

This part begins with an introduction to message-oriented middleware and the WebSphere MQ product. The concept of messaging is covered, and how those concepts are provided within WebSphere MQ. It then gives a short description of what is new in WebSphere MQ V7.1 and V7.5. An overview is provided on how it fits within the service-oriented architecture (SOA).

This part consists of the following chapters:

- ▶ Chapter 1, “Overview” on page 3
- ▶ Chapter 2, “Concepts of messaging” on page 7
- ▶ Chapter 3, “Introduction to WebSphere MQ” on page 13
- ▶ Chapter 4, “Getting started with WebSphere MQ” on page 33
- ▶ Chapter 5, “What is new in WebSphere MQ V7.1 and V7.5” on page 59



Overview

This chapter provides an overview of this entire IBM Redbooks publication. It covers the scope of material, the intended audience, and assumptions that are made concerning the reader.

This chapter contains the following sections:

- ▶ Executive summary
- ▶ The scope of this book
- ▶ Intended audience
- ▶ What is covered in this book
- ▶ What is not covered by this book
- ▶ Other related publications

1.1 Executive summary

The power of WebSphere MQ is its flexibility combined with reliability, scalability, and security. This flexibility provides many design and implementation choices. Making informed decisions from this range of choices can simplify the development of applications and the administration of a WebSphere MQ messaging infrastructure.

Applications that access a WebSphere MQ infrastructure can be developed by using a range of programming paradigms and languages. These applications can execute within a substantial array of software and hardware environments. Customers can use WebSphere MQ to integrate and extend the capabilities of existing and varied infrastructures in the information technology (IT) system of a business.

WebSphere MQ V7.1 and V7.5 were released in November 2011 and June 2012, respectively. There are many new functions in V7.1, which is available on all the major WebSphere MQ platforms. While there is some new function in V7.5, this release is primarily a packaging change, with several previously separate products now being part of a common installation and maintenance mechanism. This version is available only on UNIX, Linux, and Windows platforms. Because the two versions are so similar and delivered with only a few months gap between them, both versions are covered in this book.

1.2 The scope of this book

This publication covers the core enhancements made in WebSphere MQ V7.1 and V7.5 and the concepts that must be understood.

A broad understanding of the product features is key to making informed design and implementation choices for the infrastructure and the applications that access it.

Details of new areas of function for WebSphere MQ are introduced throughout this book, such as the changes to installation, shared queues on z/OS, enhanced security features, and clustering enhancements.

1.3 Intended audience

This book provides details about IBM WebSphere MQ product features and enhancements that are required for individuals and organizations to make informed application and design decisions before they implement a WebSphere MQ infrastructure or begin development of a WebSphere MQ application. This publication is intended to be of use to a wide-ranging audience.

1.4 What is covered in this book

The four parts of this book provide a sample environment in which to gain an understanding of new WebSphere MQ concepts. This book consists of the following parts:

- ▶ Part 1, “Introduction” on page 1 gives the reader a basic understanding of messaging middleware technologies and the relationship to IBM products, which requires no previous technical knowledge. It also gives a short description of all the enhancements which are part of WebSphere MQ V7.1 and V7.5.
- ▶ Part 2, “WebSphere MQ V7.1 new features and enhancements” on page 67 provides technical details about the new features and enhancements in WebSphere MQ V7.1
- ▶ Part 3, “WebSphere MQ 7.5 new features and enhancements” on page 229 provides technical details about the new features and enhancements in WebSphere MQ V7.5.
- ▶ Part 4, “Scenarios” on page 257 illustrates many of the features that are described in parts 2 and 3 in multi-faceted scenarios. This information assumes a knowledge of many of the basic features that were introduced in previous versions of WebSphere MQ.

1.5 What is not covered by this book

For more information about specific aspects of MQ that were closely integrated in V7.1 and V7.5, see the following publications:

- ▶ *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, SG24-8054, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/sg248054.html>
- ▶ *Secure Messaging Scenarios with WebSphere MQ*, SG24-8069, which is available at this website:
<http://www.redbooks.ibm.com/redpieces/abstracts/sg248069.html>
- ▶ *Getting Started with WebSphere MQ File Transfer Edition V7*, SG24-7760, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/sg247760.html>
- ▶ *High Availability in WebSphere Messaging Solutions*, SG24-7839, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/sg247839.html?Open>

Although aspects of these products and components are mentioned, this book concentrates on the core queue manager changes that were made in these releases.

1.6 Other related publications

For more information about the features of earlier versions of WebSphere MQ, see the following publications:

- ▶ *WebSphere MQ V6 Fundamentals*, SG24-7128, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/sg247128.html>
- ▶ *WebSphere MQ V7.0 Features and Enhancements*, SG24-7583, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/sg247583.html>



Concepts of messaging

This chapter discusses the reasons why a business might require a messaging solution, different requirements for messaging, and gives some examples of where messaging was used.

This chapter contains the following sections:

- ▶ The business case for message-oriented middleware
- ▶ Application simplification
- ▶ Example scenarios
- ▶ Messaging in service-oriented architecture

2.1 The business case for message-oriented middleware

Business environments are constantly changing. Applications that were written 20 years ago need to exchange data with applications written last week. Examples of this changing environment can be one company that is merging with another, a new partner to communicate with, an application that is used internally within a company is now exposed to customers, or different departments within a company need to share programs.

The growth of Internet banking requires services, such as managing payments or querying account information, to be made available through a range of channels. The core data can be held in a database on a mainframe, but a user of a browser requires a front-end web application server to interact with that database. As new delivery channels are created, such as a smartphone application, easy ways for that new mechanism to interact are needed. The smartphone application must communicate with the same applications and database without changing the database.

The evolutionary speed of IT systems makes essential the ability to integrate across many environments with multiple applications reliably and quickly.

Messaging is an effective way to connect these systems. It hides many of the details of communication from the application developer and gives a simple interface. Simplifying allows the developer to concentrate on the business problem instead of worrying about matters such as recovery, reliability, and operating system differences.

Another feature of messaging solutions is the decoupling of one application from another. Many mechanisms for communicating between applications require that both are available at the same time. Messaging uses an *asynchronous* model, meaning that an application that is generating messages does not have to execute at the same time as an application consuming those messages. Reducing the requirements for simultaneous availability reduces complexity and can improve overall availability. Messages can be sent to specific applications or distributed to many different applications at the same time.

Figure 2-1 shows basic patterns for connectivity.

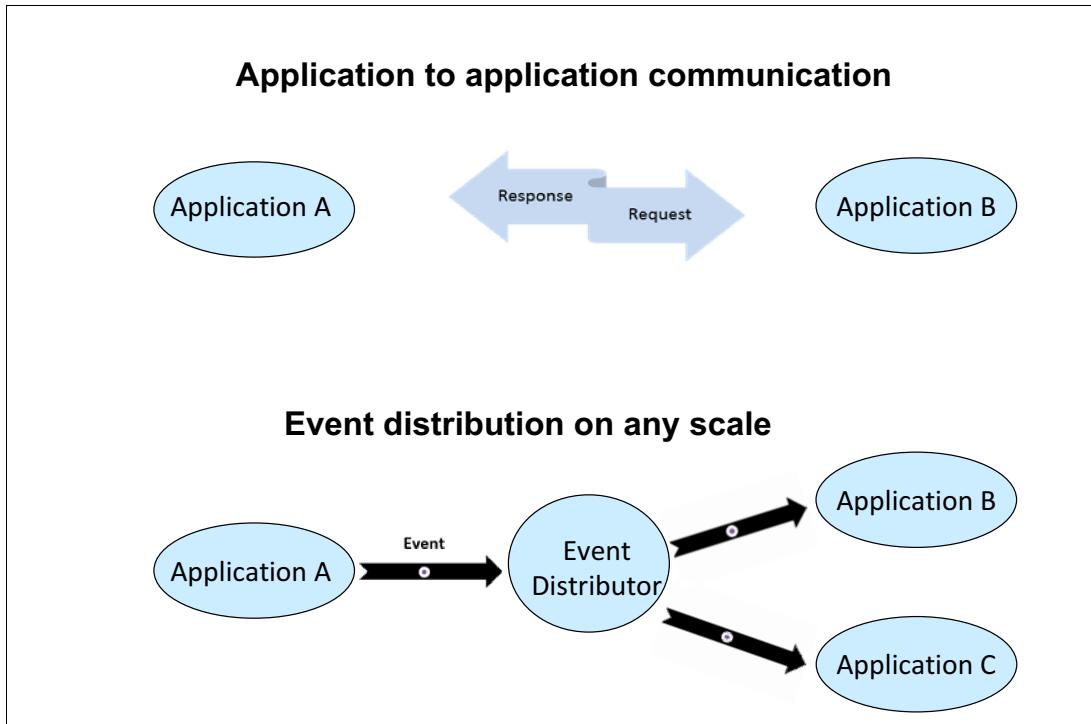


Figure 2-1 Using messaging to connect applications and distribute data

Asynchrony is used in various ways. It can queue large volumes of work that is then submitted once a day by a batch process. Work also can be submitted immediately and then processed when the receiving application finishes dealing with a previous request.

Asynchronous processing does not imply long response times. Many applications are built and successfully execute by using messaging for real-time interactive operations.

2.2 Application simplification

One of the key features of messaging is to remove complexity from application code. Worrying about reliability or recovery is the job of the middleware product or component, such as the messaging provider. Developers can focus on writing the business logic.

The following list factors should be dealt with by middleware and hidden from application developers, regardless of the business application:

- ▶ Once-only processing

Business transactions normally happen exactly once. Middleware needs to ensure all the systems that are involved in that transaction do their job, exactly once. There must be no loss or duplication. Applications must not write complex code for recovery or to reverse partially completed operations (compensation) code if individual systems fail.

- ▶ Ubiquity

Run your applications on the platform where it makes the most sense. Be able to integrate applications seamlessly on a wide range of operating systems and hardware environments.

- ▶ Easy to change

The use of industry standards can help you become more responsive because skills can be maintained and reused across projects. New applications can be rapidly written and deployed.

- ▶ Easy to extend

Scale your applications rapidly to any volume without downtime. Add more processing without application change. Roll out new applications and features quickly and safely without downtime.

- ▶ Compliance

Meet enterprise, industry, and regulatory requirements easily, especially as those requirements evolve. A middleware implementation should assist with auditing and other compliance objectives, which gives a consistent interface for all applications.

- ▶ Performance and availability

Make the best use of the systems. Provide services for high and continuous availability.

- ▶ Security

Provide a secure environment. Data is protected from loss, modification, and reading. Losing sensitive data (or failing to comply with regulations) costs time, money, and reputation.

2.3 Example scenarios

This section gives examples of where messaging solves a real-world problem.

2.3.1 Retail kiosks

A retailer wants to respond rapidly to online customer needs and have a consistent view of data that is shown at over 25,000 self-service kiosks. Data is changing rapidly, and must be distributed rapidly to these kiosks.

The network of kiosks is connected by a messaging service to the central databases and changes are pushed out rapidly. Over 14,000 transactions are processed each second.

2.3.2 Faster bank payments

Government regulations that limit payment clearing times require faster payment processes through near-instant money transfers between banks.

A high-performance payment solution is developed that uses a messaging solution to route message traffic between systems. Many existing components of the older payment system were adapted to work with the new messaging layer, which preserved the investment in these programs. The government requirement cleared payments in under two hours. However, this solution is technically capable of completing its processing in seconds. It handles millions of transactions every day.

2.3.3 Airport information

At an airport, rapid changes in government regulations are coupled with increasing passenger volumes. This airport needs a flexible IT infrastructure that is populated with seamlessly integrated systems. The airport is supporting daily business operations with an inflexible heterogeneous infrastructure.

A messaging-based solution establishes a shared integration layer, which enables the seamless flow of multiple types of data between the client's heterogeneous systems. For example, airport staff can now easily access important information, such as flight schedule changes from multiple systems. It can also flexibly integrate new systems to accommodate evolving government regulations.

2.4 Messaging in service-oriented architecture

Service-oriented architecture (SOA) is a business-centric IT architectural approach. This approach supports business integration as linked, repeatable, business tasks or services. Users build composite applications by using SOA. Composite applications draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes. SOA is an architectural style that makes this support possible. For more information about SOA, see this website:

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

The most important characteristic of SOA is the flexibility to treat elements of business processes and the underlying information technology infrastructure as secure, standardized, components (services). These components can be reused and combined to address changing business priorities. SOA requires that applications can interact with each other.

This book is not intended to discuss SOA in detail, but a critical aspect of implementing the architecture is to have connectivity. The connectivity infrastructure includes the following main attributes:

- ▶ Reliable
- ▶ Secure
- ▶ Time flexible and resilient
- ▶ Transactional
- ▶ Incremental
- ▶ Ubiquitous
- ▶ Basis for Enterprise Service Bus

Any messaging component that fits in the connectivity infrastructure must satisfy and support all of these capabilities. A flexible and robust messaging backbone is a key component of SOA and provides the transport foundation for an enterprise service bus (ESB).



Introduction to WebSphere MQ

This chapter introduces WebSphere MQ and how it implements messaging. Core concepts are described, and major features described.

This chapter contains the following sections:

- ▶ Messaging with WebSphere MQ
- ▶ Core concepts of WebSphere MQ
- ▶ Diverse platforms

3.1 Messaging with WebSphere MQ

WebSphere MQ is the market-leading messaging integration middleware product. Introduced in 1993 (under the MQSeries name), it focused on providing an available, reliable, scalable, secure, and high-performance transport mechanism to address requirements that were described in Chapter 3, “Introduction to WebSphere MQ” on page 13.

WebSphere MQ also is in the business of connecting systems and applications, regardless of platform or environment. It is essential to be able to communicate between a GUI desktop application that is running on Windows and a CICS transaction that is running on z/OS. That value of universality is core to the product, and has not changed in the time it has been available. What changed is the range of environments in which WebSphere MQ can or must live.

There are newer platforms, environments, requirements for qualities of service, and newer messaging patterns. Security is more important as systems are made accessible to more users across an enterprise and beyond it. Performance and scalability requirements increased. Regulators and auditors imposed more controls on what can or must be done. Systems, which need access to enterprise data, became both more powerful (faster, more CPUs, and so on) and much less powerful (sensors, tablets, and mobile phones). Therefore, WebSphere MQ evolved.

There are now more ways for applications to reach a WebSphere MQ queue manager and get access to any existing applications that were already WebSphere MQ-enabled. New applications can be written that use alternative interfaces and still maximize the reliability and performance of WebSphere MQ. Those new applications do not need a complete replacement of existing infrastructure. The applications work with what you already have and know how to manage.

At the same time as adding new interfaces, protocols, and environments, much WebSphere MQ workload continues to be executed in mainframe-based data centers. Efficient use of, and integration with, the capabilities of the z hardware and operating systems is critical. As this book shows, WebSphere MQ V7.1 for z/OS significantly improves the performance, capacity, and availability that can be achieved when it is running in a sysplex.

3.1.1 A history of WebSphere MQ

Figure 3-1 shows a timeline of WebSphere MQ versions and some highlights of features and products that were made available.

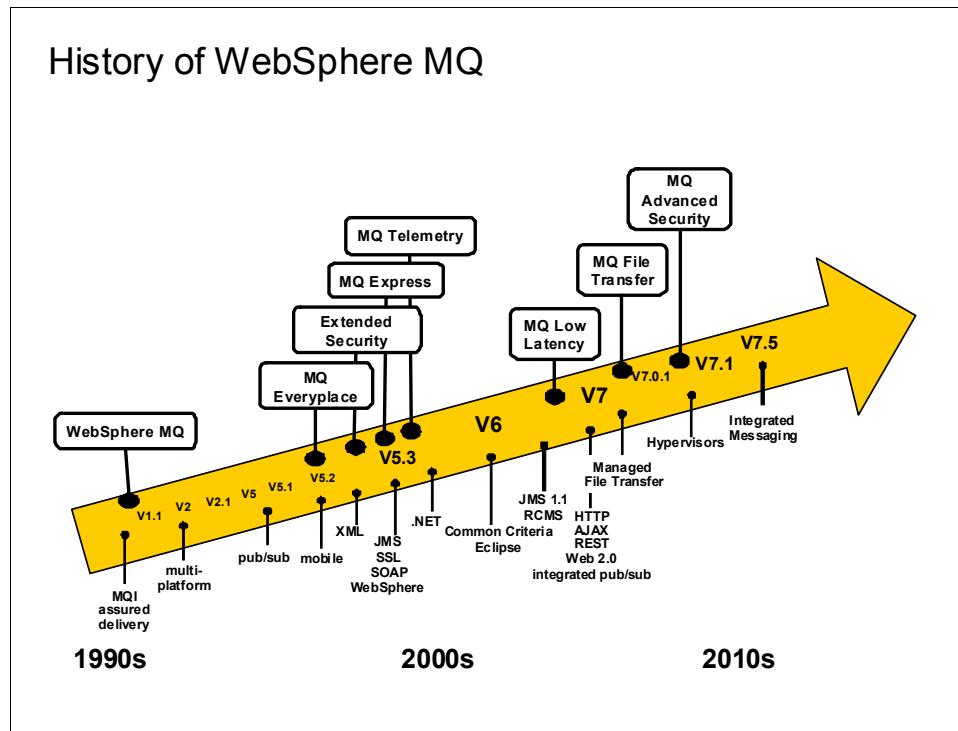


Figure 3-1 A history of WebSphere MQ

In 2008, WebSphere MQ V7.0 was a major release that featured enhanced application programming interfaces (APIs) and a fully integrated publish and subscribe component. These enhancements made the messaging pattern a first-class way of working. In 2009, WebSphere MQ V7.0.1 was primarily an update to improve availability and performance.

Although there was nearly a two and-a-half year gap before V7.1 was released in 2011, there were incremental improvements. The improvements were to the core WebSphere MQ product and new products.

WebSphere MQ File Transfer Edition (WebSphere MQ File Transfer Edition) is a solution for integration of applications that are file-based, which makes it possible to reliably transfer files between systems. It provides management, monitoring, and security of those transfers. WebSphere MQ File Transfer Edition uses existing skills and infrastructure when it uses WebSphere MQ as its backbone transfer medium.

Advanced Message Security (AMS) is a solution to the problem of securing message contents, even when those messages are at rest in queues or logs. There are a number of regulatory frameworks that require data to be protected from viewing, changing, or being stored on disks in the clear, even during processing. One example of these regulations is the Payment Card Industry (PCI) for dealing with credit card data. There are other regulatory bodies for other types of data (for example, healthcare), which have similar requirements.

WebSphere MQ File Transfer Edition and WebSphere MQ Advanced Message Security were integrated into the WebSphere MQ V7.5 package, with WebSphere MQ File Transfer Edition being renamed to Managed File Transfer. There are two variations of V7.5, which are functionally identical but include different licensing terms. The WebSphere MQ Advanced offering can be ordered under a single part number and price, which includes licenses to use the Managed File Transfer and AMS features. The original V7.5 offering has these features as separately orderable and priced components.

3.2 Core concepts of WebSphere MQ

Data is transferred between applications in messages. A message is a container that consists of the following parts:

- ▶ WebSphere MQ Message Descriptor (MQMD): Identifies the message and contains more control information. Examples of more information are the type of message and the priority that is assigned to the message by the sending application.
- ▶ Message properties: An optional set of user-definable elements that describes the message without being part of the payload. Applications that receive messages can choose whether to inspect these properties.
- ▶ Message data: Contains the application data. The structure of the data is defined by the application programs that use it, and WebSphere MQ is largely unconcerned with its format or content.

Within a WebSphere MQ message queuing infrastructure, the nodes are called *queue managers*. The queue manager is responsible for accepting and delivering messages. Multiple queue managers can run on a single physical server or on a wide network of servers across various hardware and operating system platforms.

Each queue manager provides facilities for reliable messaging.

The queue manager maintains *queues* of all messages that are waiting to be processed or routed. Queue managers are tolerant of failures and maintain the integrity of business-critical data that flows through the message queuing infrastructure. If failures occur, such as the queue manager, machines, or power loss, messages can be recovered. Recovery occurs when the queue manager is restarted and any transactions are brought to a consistent state.

Within the infrastructure, the queue managers are connected by logical *channels* over a communications network. Messages automatically flow across these channels from the initial producer of a message to the eventual consumer of that message that is based on the configuration of the queue managers in the infrastructure. Changes can be made to the configuration of queues and channels transparently for the applications. For example, a receiving application can be moved to a new machine, and a route to that machine can be defined without needing any changes to sending applications.

Figure 3-2 shows these essential elements.

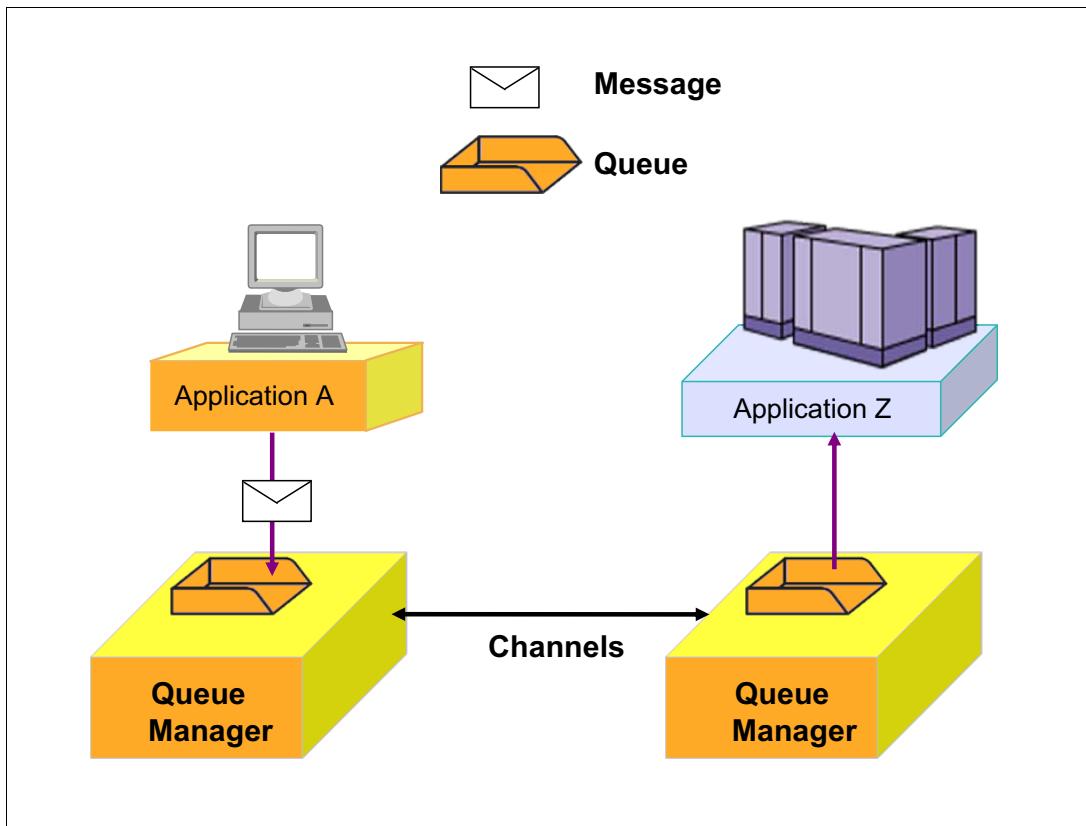


Figure 3-2 Basic WebSphere MQ components to connect applications

3.2.1 Asynchronous messaging

Two applications that must communicate, whether hosted on the same machine or separate machines, might be designed originally to communicate directly and synchronously. This configuration was a common messaging technique that was used before the introduction of WebSphere MQ and is still often used. For example, the use of HTTP as a communications protocol is a synchronous operation.

In the synchronous model, the two applications exchange information by waiting for the partner application to become available and then sending the information. If the partner application is unavailable for any reason, including if it is busy communicating with other applications, the information cannot be sent.

All intercommunication failures that can occur between the two applications must be considered individually by the applications. This consideration happens whether the applications are on the same machine or on different machines that are connected by a network. The process requires a protocol for sending the information, confirming receipt of the information, and sending any subsequent reply.

Placing a WebSphere MQ infrastructure between the two applications allows this communication to become asynchronous. One application places information for the partner in a message on a queue, and the partner application processes this information when it is available to do so. If required, it can then send a reply message back to the originator. The applications do not need to be concerned with communication failures or recovery.

3.2.2 WebSphere MQ clients

A WebSphere MQ client is a light-weight component of WebSphere MQ that does not require the queue manager runtime code to reside on the client system. It enables an application, which is running on a machine where only the client runtime code is installed, to connect to a queue manager that is running on another machine. The application also can perform messaging operations with that queue manager. Such an application is called a *client* and the queue manager is referred to as a *server*.

The use a WebSphere MQ client is an effective way of implementing WebSphere MQ messaging and queuing. The following benefits come from the use a WebSphere MQ client:

- ▶ There is no need for a licensed WebSphere MQ server installation on the client machine.
- ▶ Hardware requirements on the client system are reduced.
- ▶ System administration requirements on the client system are reduced.
- ▶ An application that uses a WebSphere MQ client can connect to multiple queue managers on different machines.

There are several implementations of the WebSphere MQ client that allow work in different language environments, including Java and .Net classes. There is no client for z/OS. On the z/OS platform, applications can connect only to queue managers within the same logical partition (LPAR).

Important: Because there must be a synchronous communication protocol between the client and the queue manager, WebSphere MQ clients require a reliable and stable network.

3.2.3 WebSphere MQ Telemetry clients

WebSphere MQ Telemetry (MQTT) protocol was designed for small footprint, low-processing, low bandwidth, unreliable networked systems in which even a WebSphere MQ client was too heavyweight. MQTT features a simpler API to minimize the processing needed. The MQTT protocol is public and a number of non-IBM implementations exist. For more information, see this website:

<http://www.mqtt.org>

The first implementations of MQTT were in places like sensors on pipelines, but it is also key to integration with newer environments, such as tablets and mobile phones. In the Smarter Planet® vision of interconnected, integrated systems, much of that integration is expected to be with MQTT. The MQTT integration can be as a raw API and protocol or working with solutions such as IBM Worklight that simplify how you build, deploy, and manage applications in the mobile world.

The MQTT feature brings together several pieces of existing technology and simplifies the experience for a seamless integration with WebSphere MQ applications.

It is critical that the queue managers to which MQTT devices connect can handle the levels of scale that those solutions demand. Thousands of devices can be generating data and require connectivity. Benchmarks show that a single queue manager can handle 100,000 MQTT connections. Messages from these devices are then passed to WebSphere MQ enabled applications for further processing. Solutions such as smart meters for power utilities, health monitors, and traffic management systems in cities were implemented by using MQTT services.

3.2.4 Application programming interfaces

Applications can use WebSphere MQ with several programming interfaces.

In many cases, applications are written (or rewritten) to use these interfaces. However, there are circumstances, particularly on z/OS, in which existing applications can be unchanged because WebSphere MQ provides bridges to and from their environments. In particular, there are bridges for CICS and IMS™ transactions that convert WebSphere MQ messages. The messages are converted into input for these transactions and the reverse is done for returned responses.

Message Queue Interface

The native programming interface is the Message Queue Interface (MQI). The MQI consists of the following elements:

- ▶ Calls or verbs through which programs can access the queue manager and its facilities.
- ▶ Structures that programs use to pass data to, and get data from, the queue manager.
- ▶ Elementary data types for passing data to, and getting data from, the queue manager.
- ▶ Classes in object-oriented languages for accessing data, the queue manager, and its facilities.

Many programming languages and styles are supported depending on the software and hardware platform, such as C, COBOL, Java, and C#.

Standardized APIs

The use of a standardized API can add flexibility when services are accessed through a message queuing infrastructure. This publication uses the term *standardized API* to represent APIs that are not proprietary to an individual product, such as WebSphere MQ.

The following examples of standardized APIs can be used to access services that are provided through a WebSphere MQ infrastructure:

- ▶ Java Message Service (JMS)
- ▶ IBM Message Service Client (XMS)

Wide adoption of these APIs can occur across multiple products. For example, the JMS API is an industry-standardized API for messaging within the Java Enterprise Edition (Java EE) specification.

These standardized APIs generally feature less function than the MQI when they use a less capable messaging service than WebSphere MQ. The standards were written based on a common subset of functions that were available at the time on all relevant messaging providers.

HTTP

There is a defined mapping between HTTP operations (**GET**, **POST**, **DELETE**) and a subset of MQI functions. A queue manager can be configured to include an HTTP server that is associated with it and then process HTTP requests directly. This configuration makes it simple for applications to interact with WebSphere MQ enabled systems without needing any WebSphere MQ code installed. For example, a smartphone features an HTTP interface that user-written applications can use, but there might be no WebSphere MQ client available for that device.

3.2.5 Reliability and integrity

WebSphere MQ provides various features to maintain reliable delivery of messages.

Persistent and non-persistent messages

Messages that contain critical business data, such as receipt of payment for an order, should be reliably maintained and must not be lost in the event of a failure.

Some messages might contain only query data, where the loss of the data is not crucial because the query can be repeated. In this case, performance is considered more important than data integrity.

To deal with these opposite requirements, WebSphere MQ uses two types of messages. The following message types (persistent and non-persistent) are used:

- ▶ Persistent messages: WebSphere MQ does not lose a persistent message through network failures, delivery failures, or restart of the queue manager.
Each queue manager keeps a failure-tolerant recovery log of all actions that are performed upon persistent messages. This log is sometimes referred to as a *journal*.
- ▶ Non-persistent messages: WebSphere MQ optimizes the actions that are performed on non-persistent messages for performance.
Non-persistent message storage is based in system memory, so it is possible they can be lost in situations such as network errors, operating system errors, hardware failure, queue manager restart, and internal software failure.

WebSphere MQ assures once-only delivery of persistent messages and it assures at-most-once delivery of non-persistent messages. These persistence options allow developers to write applications with the knowledge that there are no duplicated messages.

Units of work

Many operations that are performed by an application cannot be considered in isolation. An application might need to send and receive multiple messages as part of one overall action. Only if all of these messages are successfully sent or received should any messages be sent or received. These actions are considered to be a unit of work (UOW). All WebSphere MQ implementations support transactional operations, with sets of messages that are being committed or backed out as a single action.

An application that processes messages might also need to perform coordinated work against other resources and the WebSphere MQ infrastructure. For example, it might perform updates to information in a database that is based upon the contents of each message. The actions of retrieving the message, sending any subsequent reply, and updating the information in the database must complete only if all actions are successful.

UOWs that are performed by applications that access a WebSphere MQ infrastructure can include sending and receiving messages and updates to databases. On distributed platforms, WebSphere MQ can coordinate all resources to ensure that a UOW is completed only if all actions within that unit of work complete successfully. On all platforms, WebSphere MQ can also participate in global units of work that are coordinated by other products. For example, actions against a WebSphere MQ infrastructure can be included in global UOWs that are coordinated by WebSphere Application Server and DB2®. On z/OS, global UOWs are often coordinated by CICS, IMS, or RRS.

Interconnected queue managers

The communication that is performed across channels between queue managers is tolerant of network failures. The communication protocol that is used between queue managers maintains the once-only or at-most-once delivery that is indicated by the message's persistence level.

3.2.6 WebSphere MQ messaging styles

There are two basic styles of messaging: point-to-point and publish/subscribe.

Point-to-point

The point-to-point style is built around the concept of message queues. Messages are stored on a queue by a source application, and a destination application retrieves the messages. This configuration provides the capability of storing and forwarding messages to the destination application in an asynchronous or decoupled manner. Synchronous Request/Reply designs also can be implemented by using point-to-point asynchronous messaging.

The source application needs to know the destination of the message. The queue name is usually enough when alias queues, remote queues, or clustered queue objects are used. However, some designs might require the source application to know the name of the remote queue manager.

Point-to-point messaging is normally used when there is known to be a single producer and a single consumer of the messages.

For example, an application that is running on Linux might inquire about a piece of information in a database that is only directly readable by using a CICS transaction. The application sends a message with the inquiry details, a CICS transaction reads that message, reads the database, and sends the response back in another message. In this example, a single CICS transaction can be processing requests from many instances of the Linux application and sending unique responses to each of those instances.

Publish/Subscribe

WebSphere MQ publish/subscribe allows the information provider to be further decoupled from consumers of that information.

In point-to-point messaging, connected applications need to know the name of the queues through which they interact. The queue names might not actually be the same in the sender and receiver, as the WebSphere MQ administrator can define aliases and other mechanisms to route to other systems.

In publish/subscribe, the applications agree on the name of a *topic*. The producer of a message is known as a *publisher*, and the message is labeled with a topic. The consumers of messages (subscribers) tell WebSphere MQ that they are interested in a topic, and each is sent a copy of the relevant messages.

A topic is simply a text string, normally with a hierarchical structure, such as /Price/Fruit/Apples. A subscriber might use this topic exactly, or subscribe to topics with a wildcard so there is no need to know all of the topics that might be used by the publisher.

There can be a single subscriber, many, or none to any particular topic. The publisher does not need to know how many subscribers exist.

The classic example of publish/subscribe is to distributed stock prices: many people are interested in the price of a particular company's stock, and there is a single provider publishing that price. Traders can join and leave the feed at any time, and the provider does not need to know when this action occurs. Another example might be for a retailer to distribute catalog changes from a central site to all branches in a region.

3.2.7 WebSphere MQ topologies

A topology that consists of a single queue manager that is running on the same machine as its applications includes limitations of scale and flexibility. WebSphere MQ Clients can allow applications to run on remote machines. The connections between the applications and the queue manager require a reliable and stable network.

The limitations of this topology can be eliminated without alteration to the applications by using distributed messaging. Applications that access a service can use a queue manager that is hosted on the same machine as the application, which provides a fast connection to the infrastructure. WebSphere MQ channels extend the connectivity by providing transparent asynchronous messaging with applications that are hosted by other queue managers on remote machines by using a communications network.

The following common types of distributed messaging are available:

- ▶ Hub and Spoke
- ▶ WebSphere MQ Clustering

Hub and Spoke

Applications that access a service connect to their local queue manager. Usually this connection occurs as a direct connection to a queue manager that runs on the same machine. A client connection can also be used to a queue manager on the same machine or to a queue manager on a different machine over a fast and reliable network. For example, an application in a branch office needs to access a service at headquarters. Asynchronous communication occurs through a local queue manager that acts as a spoke to the service application that is hosted on a hub queue manager.

The machine that hosts a hub queue manager can host all of the applications that provide the central services, for example, headquarters applications and databases.

This type of architecture is developed by manually defining the routes from the spoke queue managers to the hub queue manager or many hub queue managers. Multiple services that are provided by the infrastructure can be hosted on different hub queue managers or through multiple queues on the same hub queue manager.

Figure 3-3 shows a simple hub and spoke design. Messages can be sent from Branch A to Branch D, but only by passing through the Corporate Office queue manager.

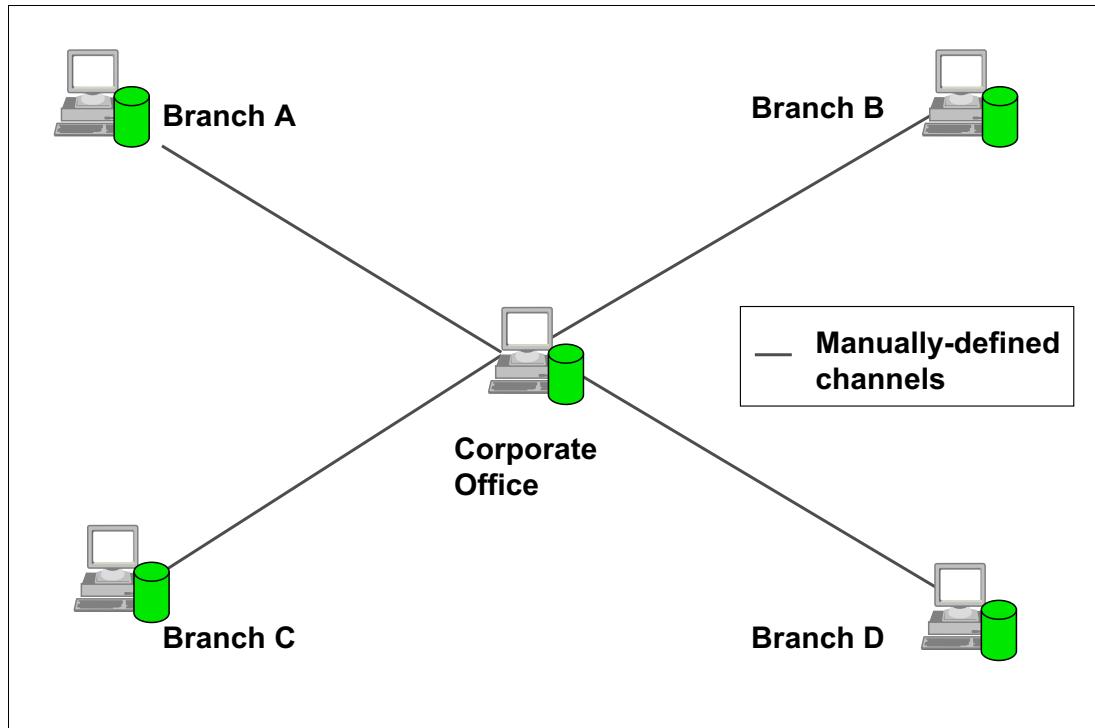


Figure 3-3 A simple hub and spoke configuration

Manually defined routes do not have to follow a hub and spoke design. It is possible to have all the queue managers in the enterprise be directly connected one to another. But when the number of queue managers grow, maintaining these routes can be a time-consuming activity.

Figure 3-4 shows what happens when direct connections are created between all queue managers and the number of queue managers grows.

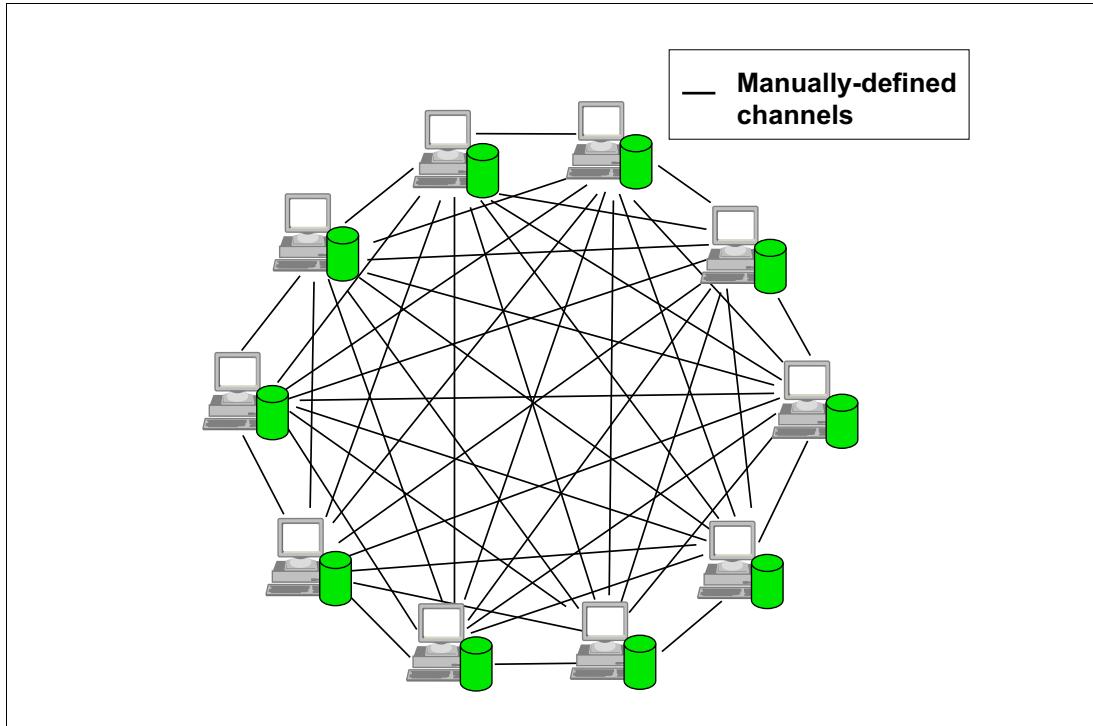


Figure 3-4 Any-to-any direct paths without clustering

WebSphere MQ cluster

A more flexible approach is to join many queue managers in a dynamic logical network that is called a *queue manager cluster*. A cluster allows multiple instances of the same service to be hosted through multiple queue managers.

Applications that request a particular service can connect to any queue manager within the queue manager cluster. When applications make requests for the service, the queue manager to which they are connected uses a workload balancing algorithm to spread these requests across all available queue managers that host an instance of that service.

Figure 3-5 shows an example of workload balancing. The message that is sent from Application A can be processed by any instance of Application Z. The cluster chooses one of the routes that are based on various criteria, including knowledge of which route it previously chose and which routes are still available. By default, each instance of Application Z should be sent the same number of messages from Application A.

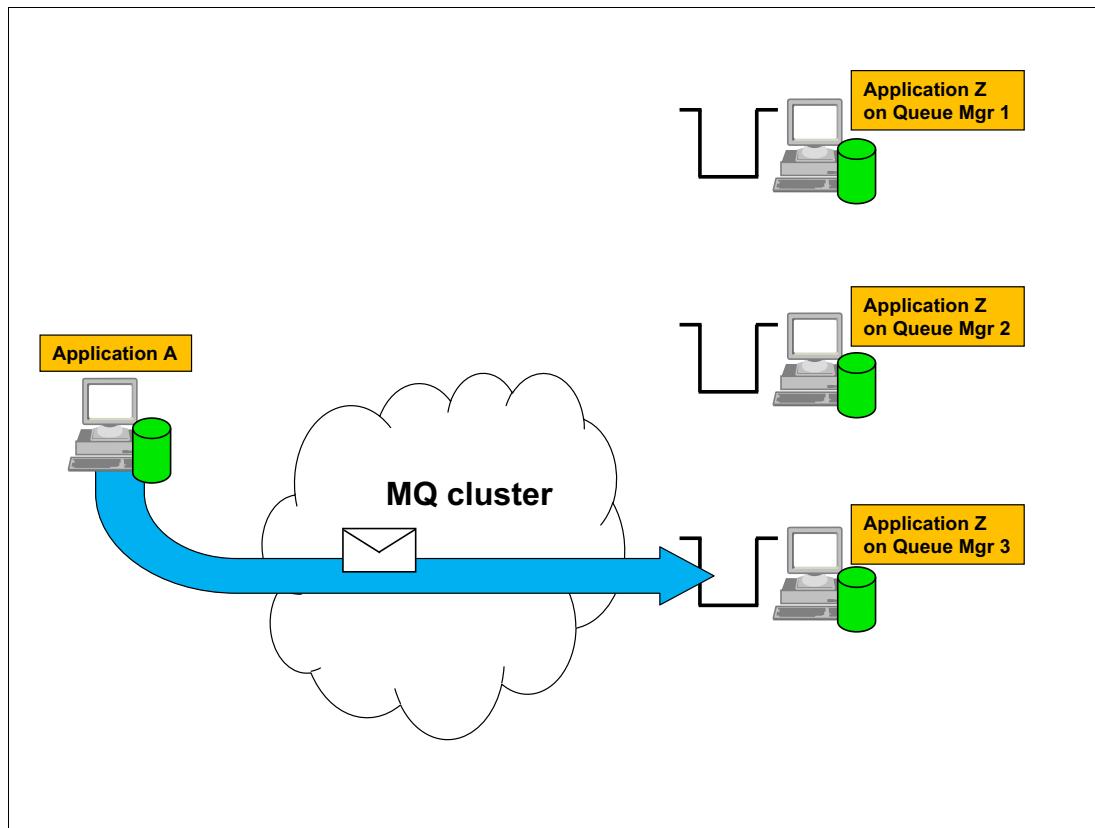


Figure 3-5 Workload balancing in a WebSphere MQ cluster

This configuration allows a pool of machines to exist within the WebSphere MQ cluster, each hosting a queue manager and the applications that are required to provide the service. This topology is especially useful in a distributed environment where capacity is scaled to accommodate the current load through multiple servers rather than one high-capacity server. A server can fail or be shut down for maintenance and service is not lost.

Important: The word *cluster* can have many meanings. In particular, it is often used to refer to high availability (HA) solutions that are based on failover technology. WebSphere MQ clusters can be used effectively with HA clusters, but they are providing different services.

The use of WebSphere MQ cluster technology can also simplify administration tasks because most of the message channels for application data transport are maintained by the cluster and do not have to be explicitly created.

Figure 3-6 shows the simplified administration that is possible with a WebSphere MQ cluster. Messages flow directly between any two queue managers, without needing to pass through a hub. Only a single initial definition is required to a designated queue manager, which is known as a *full repository*. From that point, relationships are automatically discovered when required and the associated channels are defined without intervention.

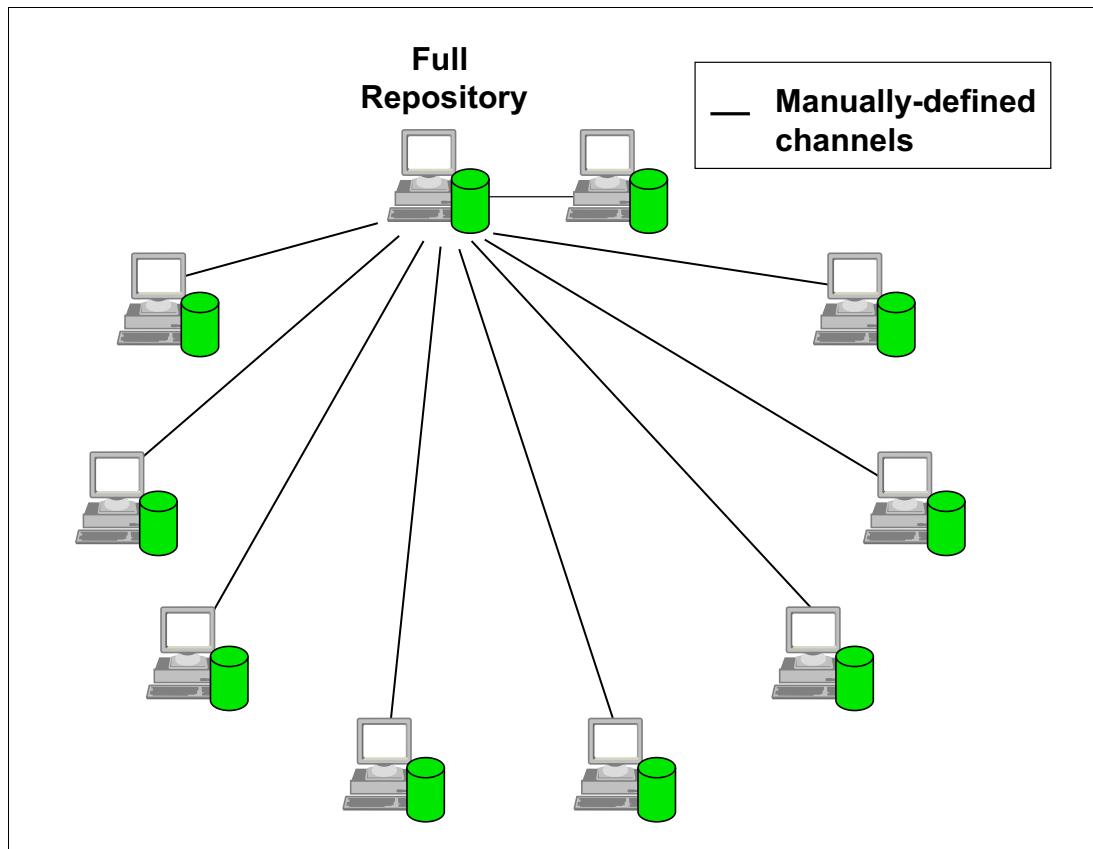


Figure 3-6 Simplified administration with a WebSphere MQ cluster and automatic definition

3.2.8 Availability

WebSphere MQ provides and integrates with the following range of technologies for availability:

- ▶ Platform-specific products, such as PowerHA® (for AIX®), Microsoft Cluster Services (for Windows), and ARM (for z/OS), can be used to control restart and recovery of queue managers in the event of failures.
- ▶ On distributed platforms, WebSphere MQ has a feature that is known as *multi-instance support*, which allows the queue manager to restart itself automatically on a backup machine.
- ▶ On z/OS, WebSphere MQ integrates with operating system and hardware-provided services for a sysplex. Shared queues can provide continuous availability, with messages that are being processed even when a whole LPAR fails and takes out a queue manager.
- ▶ WebSphere MQ clusters are not a complete answer to availability, but they can form part of a fuller HA solution.

For a more detailed description of WebSphere MQ's HA capabilities, see *High Availability in WebSphere Messaging Solutions*, SG24-7839.

3.2.9 Security

WebSphere MQ features the following range of security features to protect against unauthorized access to resources and data:

- ▶ Access Control

WebSphere MQ can be configured to permit only certain users or groups to connect or to use resources, such as queues or topics.

The permissions can be granular, for example, to say that a user is permitted to put messages to a particular queue, but not to get messages from that same queue. New CHLAUTH records are used for controlling access through channels.

- ▶ SSL/TLS

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are industry-standardized technologies that provide assurance of identity and data privacy. SSL and TLS combine with WebSphere MQ client applications that connect to queue managers by using a communications network infrastructure. These security layers also work with queue manager-to-queue manager distributed queuing and clustering by using a network.

SSL and TLS provide similar capabilities and build upon similar principles for establishing identity. TLS is often considered the successor of SSL because it provides some enhanced security features. SSL or TLS can be used for all communication that is performed over a network within a WebSphere MQ infrastructure.

- ▶ AMS

The Advanced Message Security feature or component supplies another layer of security on individual messages. It ensures that messages cannot be read or changed by unauthorized users, even when the message is not being transmitted on a network but is stored on a queue or recorded in a recovery log file.

3.2.10 Management and monitoring

WebSphere MQ provides many interfaces and services for configuring, managing, and monitoring its operation.

For example, you can define and alter queue attributes, be sent alerts when a queue gets full or a channel stops unexpectedly, and obtain statistics on how much traffic is passing through a queue manager or individual applications.

WebSphere MQ operation management is performed by using either of the following primary ways:

- ▶ You can enter text commands (known as MQSC) into a console or scripting environment. For example, the following command creates a queue with specified attributes:
DEFINE QLOCAL(MY.QUEUE) DESCRIPT('Test Queue') DEFPSIST(YES)
- ▶ Programmatic control is done with WebSphere MQ messages that are sent to, or read from, well-known queues. The messages include documented formats, called the programmable command format (PCF), and many tools and products exist to create and use them.

An Eclipse-based tool that is called the WebSphere MQ Explorer is shipped as part of the WebSphere MQ product. This GUI makes it easy to start using WebSphere MQ.

Figure 3-7 shows the WebSphere MQ Explorer interface.

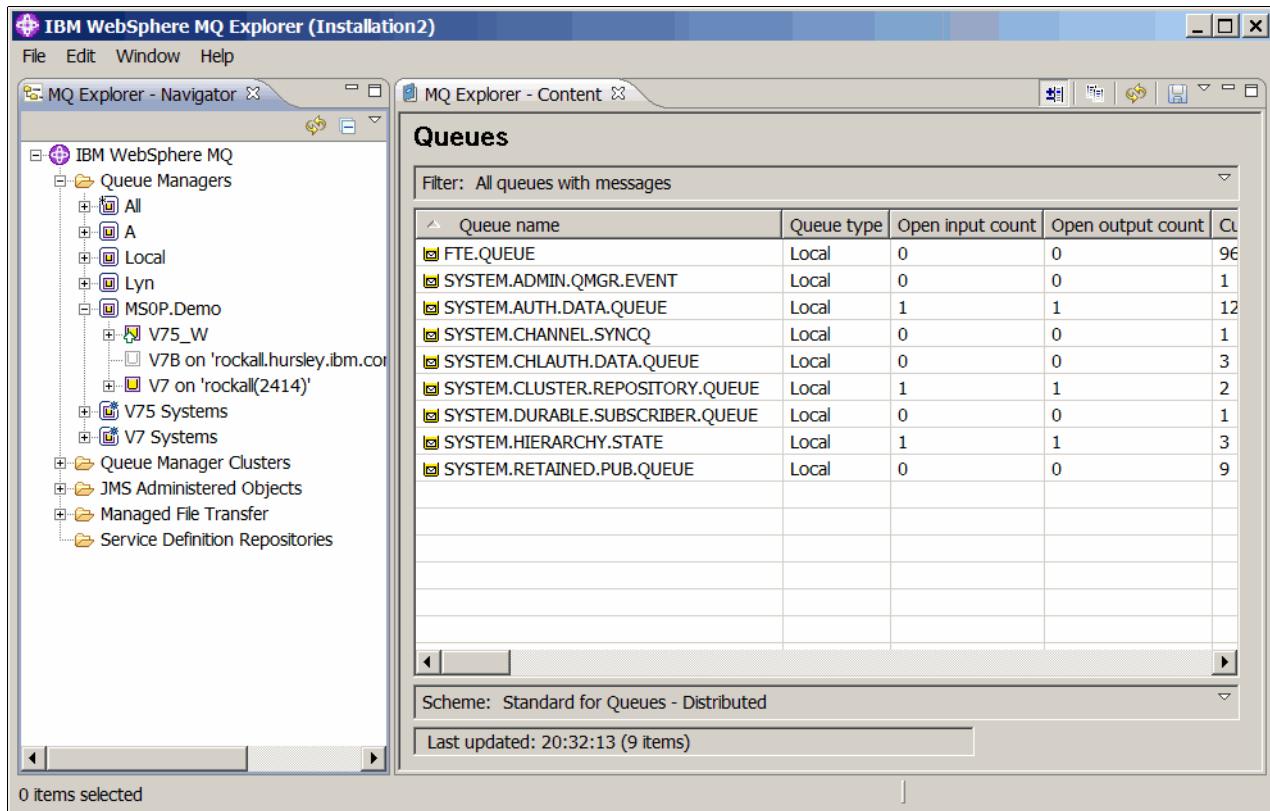


Figure 3-7 WebSphere MQ Explorer

3.3 Diverse platforms

WebSphere MQ provides simplified communication between applications that are running on many different hardware platforms and operating systems that are implemented by using different programming languages. This configuration enables a business to choose the most appropriate infrastructure components for implementing or accessing services within their system. The messaging infrastructure understands the differences between the underlying hardware and software on which individual nodes are running.

Some conversion of character data might be required for the data to be readable across different hardware and software platforms. For example, WebSphere MQ can convert between ASCII and EBCDIC code pages. It can also convert to and from Unicode characters. The messaging infrastructure can be configured to perform this conversion transparently so that each message is valid when it is retrieved at the destination.

For more information about supported platforms for WebSphere MQ, see the following websites:

- ▶ WebSphere MQ Product Information:
<http://www.ibm.com/support/docview.wss?uid=swg27007431>
- ▶ WebSphere MQ System Requirements:
<http://www.ibm.com/software/integration/wmq/requirements>

Distributed Platforms: When it is important to point out platform differences, the term *Distributed Platforms* is often used to cover all of the WebSphere MQ implementations, except for z/OS. The term refers to WebSphere MQ on Windows, UNIX, Linux, and i. Do not confuse Distributed Platforms with references to Distributed Queueing.

3.4 Relationships with other products

One of the strengths of WebSphere MQ is that it does not require the use of any other specific products. The WebSphere word at the front of the product name indicates a branding within IBM, one of many products that share the brand. It does not force the use of any other WebSphere branded products. For example, if you want to use another vendor's Java Platform, Enterprise Edition application server, WebSphere MQ implements standard interfaces to support such a product.

However, the following products are frequently used with WebSphere MQ:

- ▶ WebSphere Message Broker
- ▶ WebSphere Application Server

3.4.1 WebSphere Message Broker

Enterprise systems consist of many logical endpoints, such as off-the-shelf applications, services, packaged applications, web applications, devices, appliances, and custom built software. These endpoints expose the following set of inputs and outputs:

- ▶ Connection protocols, such as WebSphere MQ, TCP/IP, database, HTTP, files, FTP, SMTP, and POP3
- ▶ Data formats, such as C or COBOL structures, XML, industry-specific (SWIFT, EDI, HL7), and user-defined

An Enterprise Service Bus (ESB) can be imagined as something that runs at the center of this wide-range of connections, applications, and protocols. It performs tasks, such as routing and data transformation, so that any endpoint can communicate with any other endpoint. Traffic passes through an ESB without requiring a complex and unmaintainable mesh of individual point-to-point connections.

WebSphere Message Broker is an ESB product that connects these endpoints in meaningful ways to simplify application and device integration. It requires that WebSphere MQ is also installed on systems where it runs. This requirement is necessary partly because MQ-enabled applications are an important class of connectivity that it must support. It also uses queue manager-provided services, such as transaction coordination, publish/subscribe, and the reliable storage of messages. One of the WebSphere Message Broker administration tools (the WebSphere Message Broker Explorer) runs inside WebSphere MQ Explorer to give a single point of control for both products.

One example of how WebSphere Message Broker makes integration simple is to consider a typical retail enterprise. There are many different end points inside and outside the organization, with data in multiple formats and protocols, such as TLOG, files, and JSON/HTTP. When new capabilities must be blended in (such as support for mobile applications and analytics), the flexibility of WebSphere Message Broker makes it simple to add these functions without disrupting existing services.

For more information about WebSphere Message Broker, see the following resources:

- ▶ *Using WebSphere Message Broker V8 in Mid-Market Environments*, SG24-8020, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/sg248020.html>
- ▶ WebSphere Message Broker Library:
<http://www-01.ibm.com/software/integration/wbimessagebroker/library/>

3.4.2 WebSphere Application Server

WebSphere Application Server is a product that hosts and runs applications. At its core, it is a Java Platform, Enterprise Edition environment that implements and supports a range of open standards. One such standard that Java Platform, Enterprise Edition applications often use is the Java Message Service (JMS).

Java Platform, Enterprise Edition application servers make it possible to use any vendor's JMS implementation, but there is no requirement on how simple it is to use a non-native JMS provider. WebSphere Application Server includes the runtime Java client code and the administration panels that make it easy to connect to a WebSphere MQ queue manager. This configuration means that web applications can reliably send messages to, and get responses from, any other MQ-enabled application that might be running in the enterprise.

For more information about WebSphere Application Server, see the following resources:

- ▶ *WebSphere Application Server V8.5: Technical Overview Guide*, REDP-4855, which is available at this website:
<http://www.redbooks.ibm.com/abstracts/redp4855.html>
- ▶ WebSphere Application Server Library:
<http://www-01.ibm.com/software/webservers/appserv/was/library/>



Getting started with WebSphere MQ

This chapter shows how to get started with configuring a WebSphere MQ environment. Adapted and brought up-to-date from *WebSphere MQ Primer: An Introduction to Messaging and WebSphere MQ*, REDP-0021-01, which was published in 1999, this chapter shows how the elements that were described in this book's introduction are implemented and used.

This chapter contains the following sections:

- ▶ Messages
- ▶ WebSphere MQ Objects
- ▶ Configuring WebSphere MQ
- ▶ Writing applications
- ▶ Triggering
- ▶ Configuring a WebSphere MQ client
- ▶ Security
- ▶ Configuring communication between queue managers
- ▶ Summary

4.1 Messages

As previously described, a message is a container that consists of the following parts:

- ▶ WebSphere MQ Message Descriptor (MQMD): Identifies the message and contains more control information, such as the type of message and the priority that is assigned to the message by the sending application.
- ▶ Message Properties: An optional set of user-definable elements that describes the message without being part of the payload. Applications that receive messages can choose whether to inspect these properties.
- ▶ Message data: Contains the application data. The structure of the data is defined by the application programs that use it. WebSphere MQ is largely unconcerned with its format or content.

An individual message can contain up to 100 MB of data. There are no constraints on the format of this data. It can be readable text, binary, XML, or any combination of these formats.

4.1.1 Message descriptor

The MQMD is a fixed structure that describes the message. Some fields within the structure describe the message format, some are used for routing, some describe the sender of the message, and some tell the queue manager how to process the message.

The following important fields are part of the MQMD:

- ▶ **MsgType**

This field describes the high-level purpose of the message. The most frequently used values in this field are datagram, request, reply, and report. A datagram is set by the sending application when it is not expecting any response from the receiving application.

- ▶ **MsgID or CorrelID**

These fields are used to identify a specific request or reply message. The programmer can set a value in one or both fields or have WebSphere MQ automatically create a unique ID. These fields are 24 bytes long and are not treated as characters that can be converted between code pages, such as ASCII to EBCDIC.

A normal pattern for application programs is that the message ID that is generated by WebSphere MQ when a message is put to a queue is used to identify responses that are sent back to the designated reply queue. The program that receives the request message copies the incoming message ID into the correlation ID of the reply message. This configuration allows the originating program (the one that gets the reply) to instruct the queue manager to return a specific message in the reply queue instead of getting the first message from the queue. This configuration is important when multiple applications are reading messages from the same queue.

- ▶ **Persistence flag**

This flag selects whether the message should be treated as persistent (and therefore recoverable if there is a system failure), or whether the message is non-persistent and can be discarded when an error occurs. A programmer can explicitly set persistence to be on or off, or allow the value to be controlled by a system administrator.

Queues can hold a combination of persistent and non-persistent messages.

- ▶ Priority

Messages have a Priority set as a value between 1 and 10. An administrator can define that messages are retrieved in priority order, instead of true FIFO sequencing.

- ▶ PutTime and PutDate

The queue manager records the PutTime and PutDate when the message was initially put to a queue. These values are in Coordinated Universal Time (UTC), not the local time, to allow a consistent view of time when messages pass between systems in different timezones.

- ▶ Expiry

Messages might include Expiry time. After this time passes, the message cannot be returned to any application. This field is used for situations where a message is not going to be useful after a while. For example, after a request message is sent, an application often waits a finite period (for example, 10 seconds) for a reply to be returned and show an error if nothing is received in that time. Knowing that, the request and response messages can have a 10-second expiry time set. If there is a system failure, the messages are removed from the queues when they are no longer useful and therefore do not waste disk or memory space.

- ▶ ReplyToQ and ReplyToQMgr

When a request message is sent, the responding application must be told where to send the reply, which often is done by providing the *ReplyToQ* and *ReplyToQMgr* names in the initial request.

- ▶ Format

This field describes the message data. It is often set to MQSTR (to denote text data) or NONE (to denote binary data), but there are other possible values. In some cases, the body of the message is made up of other WebSphere MQ defined structures and the format field tells WebSphere MQ what to do with that data. User-defined formats can also be used. The format setting causes data conversion to take place when it is needed. For example, it causes WebSphere MQ to convert text data between ASCII and EBCDIC code pages (the CodedCharSetId field) when the sending and receiving applications are on different systems that use those character sets.

- ▶ PutApplType and PutApplName

Information about the sending application (program name and path) and the platform it is running on are set in these fields.

- ▶ Report

This field is used to request information about the message as it is processed. For example, the queue manager can send a report message to the sending application when it puts the message in the target queue or when the receiving application gets it off the queue.

- ▶ BackoutCount

▶ Each time a message is backed out, the BackoutCount is increased. An application can check this counter and act on it. For example, send the message to a different queue where the reason for the backout is analyzed by an administrator.

- ▶ GroupId and MsgSeqNumber

Messages can be sent as part of a related group. The receiving application can then wait until all messages from the group arrive before its processing is started. Message grouping also assures that the order the messages are sent in is preserved. The GroupId and MsgSeqNumber fields in the MQMD define this processing.

4.1.2 Message properties

An application can set arbitrary properties of a message that are not part of the fixed MQMD structure. For example, a message can include an assigned *color* property, and then be described as red or green. Properties can be strings, numbers, or boolean values.

These properties are not considered part of the message data, but optionally can be viewed and modified by applications. Properties also can be used to select the messages that are returned to an application. For example, an application can request that it is sent all of the green messages but none of the red messages.

Message properties are most commonly used in publish/subscribe applications to provide an additional layer of selection beyond just the topic. It is similar to a SQL query that uses the WHERE clause to return specific rows from a database table.

4.2 WebSphere MQ Objects

There are 12 types of *object* that can be configured in WebSphere MQ, and various other resources (for example, security access controls, or subscriptions) that are managed by WebSphere MQ. The most important objects from this set are queues, channels, topics, and the queue manager.

Objects feature names that are up to 48 characters long (20 characters for channels), and are case-sensitive. A queue that is named `queue.abc` is different from a queue that is named `QUEUE.ABC`. These queues can be defined on a queue manager, although it is confusing to do so. Objects include attributes or properties that define the behavior when they are used.

Some objects feature subtypes, which in turn include different sets of properties. For example, the `CONNNAME` is an attribute of SENDER channels but not of RECEIVER channels.

4.2.1 Queue manager

The queue manager is the fundamental resource in WebSphere MQ. Almost all other resources are owned by a queue manager. The only resource that is an exception is the queue-sharing group, which consists of a set of queue managers on z/OS.

It is the queue manager that manages storage of data and recovery after a failure. The queue manager coordinates all the applications that are updating messages on queues, and deals with the isolation and locking that is required to maintain consistency. The queue manager maintains statistics and status information for reporting as needed.

A queue manager features many attributes to control its processing. For example, various types of events can be enabled or disabled to report on important activity in the system. Resource constraints, such as the maximum number of queues an application can have open simultaneously are defined here. Publish/subscribe capabilities are enabled at the queue manager level.

The first task an WebSphere MQ administrator must perform is to create a queue manager. After a queue manager is created and started, the other objects can be defined on that queue manager by using WebSphere MQ's management commands and APIs.

4.2.2 Queues

Queues are used to store messages that are sent by programs or are pointers to other queues or topics. There are a number of different queue types, and some special uses of queues.

Queue types

Within a queue manager, the following types of queues are defined:

- ▶ Local queue

A local queue is the only place where messages are physically stored. All other queue types must be defined to point at a local queue.

- ▶ Alias queue

An alias queue is usually a pointer to a local queue. The use of an alias queue definition is a convenient way to allow different applications to use different names for the same real queue. An alias queue can also be used as a pointer to a topic for publish/subscribe applications.

- ▶ Remote queue

A remote queue is another type of pointer. A remote queue definition describes a queue on another queue manager. When an application opens a remote queue, the queue manager to which the application is connected ensures that messages are tagged and stored locally in preparation for sending to another queue manager. Applications cannot read messages from remote queues.

- ▶ Model queue

A model queue is a template that can be used by an application to dynamically create a real queue. These templates are often used to create a unique queue for reply messages, and then automatically deleted when the application ends.

- ▶ Dynamic queue

These queues are created by the process of opening a model queue. They are real local queues, but cannot be explicitly created through administration interfaces. There are two subtypes of dynamic queue: a temporary dynamic queue (TDQ) is automatically deleted when the application that caused it to be created ends, a permanent dynamic queue (PDQ) survives the creating application and can be reused by another application.

One common misconception is to talk about *persistent* or *non-persistent queues*. With one exception, there are no such things. Queues can hold persistent and non-persistent messages, although they do feature an attribute to define the default persistence value for messages that are put to them if the application does not explicitly set the value. The exception to this rule is a TDQ, which can be used only for non-persistent messages.

Special queues

The following queues have particular purposes:

- ▶ Transmission queue

A transmission queue is a local queue with the USAGE(XMITQ) attribute configured. It is a staging point for messages that are destined for a remote queue manager. Typically, there is one transmission queue for each remote queue manager that the local queue manager can connect to directly. If the destination is unavailable, messages build up on the transmission queue until the connection can be successfully completed. Transmission queues are transparent to the application. When an application opens a remote queue, the queue manager internally changes that remote queue to a reference to the relevant transmission queue and messages are put there.

- ▶ Initiation queue

An initiation queue is a local queue to which the queue manager writes a *trigger message* when certain conditions are met on another local queue. For example, this task is done when a message is put into an empty message queue or in a transmission queue. Such a trigger message is transparent to the programmer, but is a powerful mechanism to cause applications or channels to start executing only when there is work to be processed. For more information about this process, see 4.5, “Triggering” on page 48.

- ▶ Dead-letter queue

A dead-letter queue (DLQ) is usually defined when a queue manager is created. It is used for the following situations when a queue manager cannot deliver a message across a channel:

- The destination queue is full.
- The destination queue does not exist.
- Message puts were inhibited on the destination queue.
- The sender is not authorized to use the destination queue.
- The message is too large.

When these conditions are met, the messages are written to the DLQ. At most, a queue manager features one DLQ.

Messages are only sent to the DLQ after they are successfully put to a local transmission queue first.

If an application that is putting a message receives an error that indicates the output queue is full, the message was put to a DLQ. The message was never accepted by the queue manager for processing.

- ▶ Event queue

The queue manager generates event messages when certain events occur. For example, when a queue is nearly full or when an application is not authorized to open a queue. These event messages are written to one of the predefined event queues and can then be processed by management tools. All event queues include similar names, which indicate the type of event that is held there. SYSTEM.ADMIN.QMGR.EVENT or SYSTEM.ADMIN.CHANNEL.EVENT are examples of queue manager event queues.

- ▶ Cluster queue

A cluster queue is a local queue that is configured to advertise its existence within a MQ cluster. Applications can refer to this queue without needing any additional definitions, and all the queue managers in the cluster know how to route messages to it. There often are multiple cluster queues of the same name within the cluster, and selection of which one to be used is made by the sending queue manager by using a workload balancing algorithm.

4.2.3 Topics

WebSphere MQ V7.0 introduced a fully integrated publish/subscribe capability. With this introduction came the ability to define *topic* objects.

A topic is a character string that describes the nature of the data that is published in a publish/subscribe system. Instead of including a specific destination address in each message, a publisher assigns a topic to the message. The queue manager matches the topic with a list of subscribers who subscribed to that topic, and delivers a copy of the published message to each of those subscribers.

Although any character string can be used as the topic, topics often are chosen that fit into a hierarchical tree structure. That structure helps with establishing security policies and makes it easy for applications to subscribe to multiple topics in a single step.

Topic objects represent a subset of the topics that are used by applications.

There can be many thousands of topics that are used, but no administrator wants to create definitions of all of these topics. The topic objects are used as control points in the tree of in-use topics where different configurations are needed.

For example, subbranches of the topic tree have different access control requirements. Topic objects are used to assign authorizations to any topic lower in the tree. Whenever a decision on access control is required, the queue manager finds the topic object that is the closest parent in the tree to the application-provided topic string, and uses that object's definition.

4.2.4 Channels

A *channel* is a logical communication link. In WebSphere MQ, the following types of channels (with various subtypes) are available:

- ▶ Message channels

A message channel connects two queue managers via message channel agents (MCAs). These channels are unidirectional, in that messages are sent only in one direction along the channel.

An MCA is a program that transfers messages from a transmission queue to a communication link, and from a communication link into the target queue.

There are subtypes of message channel. The subtypes are distinguished by whether they are used for receiving or sending messages, whether they initiate the communication to the partner queue manager, and whether they are used within a cluster.

A channel definition on one queue manager must be matched with a channel definition of the same name with an appropriate subtype on the partner queue manager.

The different subtypes of message channels are known as SENDER, RECEIVER, SERVER and REQUESTER with CLUSTER-SENDER and CLUSTER-RECEIVER used for clustered configurations. While other combinations are possible, the most common configuration is to have a SENDER channel defined on one queue manager with an identically named RECEIVER channel on a second queue manager.

When the first versions were released, there was no single dominant network protocol. WebSphere MQ was designed to support several network protocols for the MCAs. These other protocols can still be used, but it is now rare to find anything other than TCP/IP.

- ▶ MQI channels

A Message Queue Interface (MQI) channel connects an MQ client to a queue manager across a network. An MQI channel is bidirectional in that one channel can be used for a client to send and receive messages. The two subtypes of MQI channel are SVRCCONN (a definition that is used by the queue manager) and CLNTCONN (a definition that is used by the client program).

For two-way messaging between queue managers, there are usually two symmetric pairs of message channels defined. One pair is for messages that are flowing in one direction; the other pair handles the reverse traffic.

Channels are usually configured so that they automatically start when messages are available on the associated transmission queue. This configuration uses *trigger* attributes that are set on the transmission queue.

When an MQ cluster is in use, it is not necessary to manually define these channels. The cluster has enough information to create channels dynamically when needed.

Figure 4-1 shows a simple configuration that connects two queue managers with these definitions.

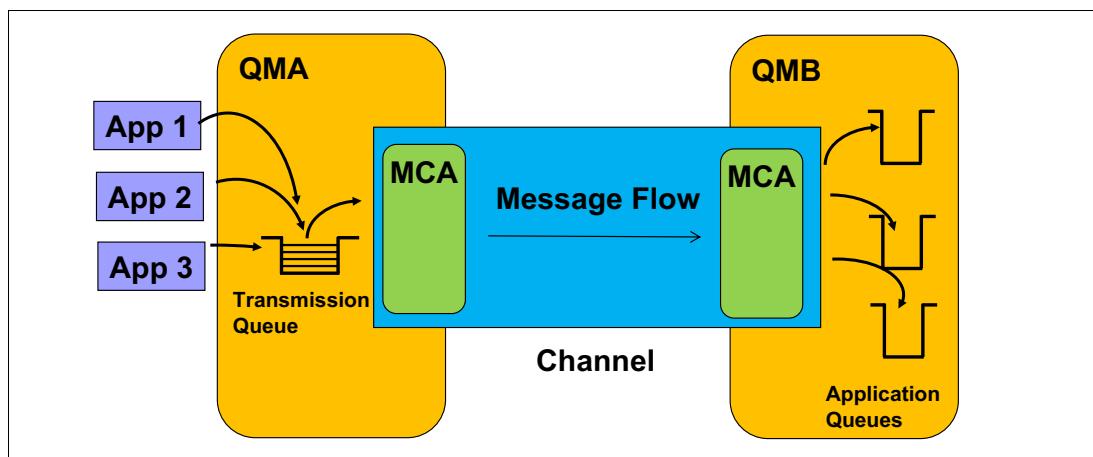


Figure 4-1 Using a channel to send messages from QMA to QMB

Listeners

Another part of the inter-communications configuration is the concept of a *listener*. Listeners are programs that wait for inbound network connection requests and then start the appropriate channel to handle the communication.

All queue managers need at least one listener to be running if they are going to receive messages or client connections. A standard configuration is to have a TCP/IP listener that waits for connections on port 1414. If multiple queue managers are running on the same system, they require individual listeners to be configured on different ports.

4.3 Configuring WebSphere MQ

This section shows some basic steps on getting started with WebSphere MQ. It uses Windows as the operating system example. Other distributed platforms are similar.

On z/OS, the initial steps are different and might involve several different administrators. These steps are fully described in the Information Center.

After the queue manager is running, managing the WebSphere MQ object definitions is the same process, regardless of platform. The graphical WebSphere MQ Explorer is an easy way to create and modify definitions, but the examples that are shown here use the command-line mechanisms.

4.3.1 Creating a queue manager

The first thing an WebSphere MQ administrator must do is create a queue manager.

By using the `crtmqm` command, a queue manager is created. Figure 4-2 shows an example of creating a queue manager called QMA.

```
C> crtqmq QMA
WebSphere MQ queue manager created.
Directory 'C:\mqm\qmgrs\QMA' created.
The queue manager is associated with installation 'Installation2'.
Creating or replacing default objects for queue manager 'QMA'.
Default objects statistics : 77 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

Figure 4-2 Creating a queue manager

Important: Queue manager names are case-sensitive.

There are no limits to the number of queue managers that can be created and run on a server. It is possible to designate one queue manager as the default for the server, and any application or administration task uses that default if no queue manager name is used. However, it is clearer to have no default queue manager and always explicitly use the name.

There are default definitions for objects every queue manager needs, such as model queues. These objects are created automatically.

To start this queue manager and make it available for applications, use the `strmqm` command, as shown in Figure 4-3.

```
C> strmqm QMA
WebSphere MQ queue manager 'QMA' starting.
The queue manager is associated with installation 'Installation2'.
5 log records accessed on queue manager 'QMA' during the log replay phase.
Log replay for queue manager 'QMA' complete.
Transaction manager state recovered for queue manager 'QMA'.
WebSphere MQ queue manager 'QMA' started using V7.5.0.0.
```

Figure 4-3 Starting the queue manager

It is now possible for applications to start putting and getting messages, and for further configuration to be done by using the WebSphere MQ administration tools.

4.3.2 Managing WebSphere MQ objects

On distributed platforms, the **runmqsc** program provides a console into which management commands can be entered. On z/OS, the same commands can be entered through panels, System Display and Search Facility (SDSF), or submitted as jobs.

The queue manager, which must be running before the **runmqsc** command is used, works in the following ways:

- ▶ By entering the commands directly.
- ▶ By using input direction to work with a text file that contains a list of commands. For example, **runmqsc QMA < commands.mqsc** causes the commands in the file to be run.

The commands that are shown in Figure 4-4 are examples that were used to create the local queue QUEUE1. They create a channel to send messages to another queue manager, and create the transmission queue that is associated with the channel. To end the session, enter **end**.

```
C> runmqsc QMA
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QMA.

define qlocal('QUEUE1')
  1 : define qlocal('QUEUE1')
AMQ8006: WebSphere MQ queue created.
define channel('T0.QMB') chltype(sdr) xmitq('QMB') connname(hostB)
  2 : define channel('T0.QMB') chltype(sdr) xmitq('QMB') connname(hostB)
AMQ8014: WebSphere MQ channel created.
define qlocal('QMB') usage(xmitq)
  3 : define qlocal('QMB') usage(xmitq)
AMQ8006: WebSphere MQ queue created.
end
  4 : end
3 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.

C>
```

Figure 4-4 Defining objects that use **runmqsc**

There are commands to define, alter, and delete all the properties of all WebSphere MQ objects, including queues, channels, and the queue manager. The **runmqsc** command can also be used to modify queue managers on remote machines, but that is beyond the scope of this introduction.

Important: Attributes in **runmqsc** commands are automatically folded to uppercase unless they are enclosed in single quotation mark characters. WebSphere MQ is case-sensitive for object names. Forgetting this fact is a common mistake that often results in an UNKNOWN_OBJECT_NAME error when you are trying to open a queue.

4.4 Writing applications

This section provides an outline of the WebSphere MQ programming interface, the MQI. It includes a brief program that is written in C to give an idea of how WebSphere MQ programs look. There are many sample programs included with WebSphere MQ to demonstrate more of the features. These samples are written in a range of languages, including C, COBOL, C#, and Java. The Java examples show the use of the full-function MQI (in its object-oriented form) and JMS.

The MQI

The MQI features 26 verbs. There were approximately 13 verbs until WebSphere MQ V7.0 was released. After that release, the number of verbs doubled to support publish/subscribe and message property operations. The most commonly used verbs are shown in Table 4-1. Each verb includes a few parameters.

Many of the parameters to these verbs are structures that contain other fields that are used by the verb. New releases of WebSphere MQ often introduce functions that are controlled by other fields in these structures. To maintain compatibility with existing applications, these structures contain a version number, which is incremented whenever the structure is extended. Applications that are compiled against the older definitions include the original structure versions. The queue manager does not reference the newer fields. To maintain compatibility, the number of parameters for each verb does not change with new releases.

Success or failure of each call is indicated by two parameters, which are included in every verb, the Reason and CompCode values.

Table 4-1 The main MQI verbs

MQI Verb	Description
MQCONN/MQCONNX	Connect to a queue manager
MQDISC	Disconnect from a queue manager
MQOPEN	Open a specific queue or topic
MQCLOSE	Close a queue or topic
MQPUT	Put a message on a queue, or publish a message
MQGET	Get a message from a queue, or receive a publication
MQPUT1	MQOPEN + MQPUT + MQCLOSE
MQSUB	Create a subscription
MQINQ	Inquire about the properties of an object
MQSET	Set properties of an object
MQBEGIN	Begin a globally coordinated (XA) unit of work
MQCMIT	Commit a unit of work
MQBACK	Back out

The main MQI verbs provide the following functions:

► **MQCONN and MQCONNX**

These verbs create a connection with a queue manager. MQCONN was the original verb, but when more parameters were required, a new verb (MQCONNX) was introduced so that established applications did not need to be changed. MQCONNX gives much more control over the connection, including the ability to specify everything that is needed to create a client connection to a queue manager on a different machine.

An application normally creates the connection when it starts to run, and does not disconnect until the application ends. Creating the connection is the most expensive (time-consuming) verb in the MQI. This fact is especially if a secure socket must be created to a remote queue manager, which means that it should be used infrequently.

► **MQDISC**

This verb disconnects the application from the queue manager. If there is an uncommitted transaction, calling MQDISC commits that transaction. If an application ends or abends without calling MQDISC, the transaction is rolled back.

► **MQOPEN**

This verb makes an object, such as a queue or topic, available to the application. A queue must be opened before an application can put or get messages. Parameters to this verb include the object name and an indication of how the application intends to use it. For example, flags say whether the application puts or gets messages (or both). This information is used to carry out security checks. Applications that are not permitted to access the object in the requested way are rejected.

It is possible to open a queue for exclusive use, or to share it with other applications. Sharing a queue can improve performance because more work can be processed in parallel.

► **MQCLOSE**

The opposite of MQOPEN, this verb says that the application is no longer interested in the use of this object. Many users believe that MQCLOSE commits any transactional work by using the queue, which is not true.

► **MQPUT**

This verb puts a message to a queue or publishes a message on a topic.

► **MQGET**

This verb retrieves a message from a queue. This verb can return immediately or wait a specified time until a message that satisfies a certain criteria arrives on the queues.

There also are options to retrieve a message non-destructively (*browse*), including the capability of browsing all messages in sequence on a queue.

► **MQPUT1**

This verb is a short-hand combination of MQOPEN, MQPUT, and MQCLOSE. It exists because a typical application pattern is to read incoming messages, perform some work, and then send a message back to a unique reply queue owned by another application. If the reply queue is only used once, this verb is more efficient than the three verbs it replaces.

There is no corresponding MQGET1 verb because the pattern of getting only a single message from a particular queue is uncommon. Even applications that use a unique, dynamically created reply queue must open the model queue before it tries to retrieve from the queue, so the short-hand could not work.

- ▶ MQSUB

The MQSUB verb creates a subscription to a topic. The subscription can include wildcards so that multiple topics are referenced by the same subscription. Published messages that match the topic are sent to a queue that is ready for MQGET to process them. The subscriber's queue can be named explicitly as a parameter to MQSUB or automatically created.

- ▶ MQINQ

The MQINQ verb requests information about the queue manager or one of its objects. Not all attributes of queues or queue managers can be inquired on by using this verb, but some important verbs can be inquired on.

- ▶ MQSET

This verb changes the attributes of an object. It has a similar subset of capabilities to the MQINQ verb.

- ▶ MQBEGIN

The MQBEGIN verb begins a unit of work that is coordinated by the queue manager and might involve external XA-compliant resource managers. It is used to coordinate transactions that use queues (MQPUT and MQGET under sync point) and database updates (SQL commands). This verb is not required if only WebSphere MQ resources are updated by the application.

This verb is available only on locally connected applications on distributed platforms. It is not available on z/OS or WebSphere MQ clients. On WebSphere MQ clients, an external coordinator such as an application server is required. On z/OS, applications use Resource Recovery Services (RRS) as the coordinator.

- ▶ MQCMIT

This verb specifies that a sync point was reached. Messages that are put as part of a unit of work are made available to other applications. Messages that are retrieved as part of a unit of work are permanently deleted from the queue. If the queue manager is acting as a global transaction coordinator, this process also invokes the XA commit process in other resource managers.

If the application is using an external transaction coordinator, such as WebSphere Application Server or CICS, it must not call MQCMIT. Transactional control is performed by using the coordinator's APIs.

- ▶ MQBACK

The MQBACK verb tells the queue manager to back out all message puts and gets that occurred since the last sync point. Messages that are put as part of a unit of work are deleted. Messages that are retrieved as part of a unit of work are reinstated on the queue. If the queue manager is acting as a global transaction coordinator, this process also invokes the XA backout process in other resource managers.

If the application is using an external transaction coordinator, such as WebSphere Application Server or CICS, it must not call MQBACK. Transactional control is performed by using the coordinator's APIs.

A code fragment

The code fragment that is shown in Figure 4-5 on page 46 is a simple example that is intended to give a flavor of how WebSphere MQ applications are written.

Figure 4-5 on page 46 shows the operations that are needed to put a message on one queue and get the reply from another queue. The sample program amqsreq0.c, which is shipped with WebSphere MQ, contains similar function. There is no error handling in this fragment.

Important: The CompCode and Reason fields contain completion codes for the verbs. CompCode is a simple indicator of success, warning, or failure. The Reason variable is a more detailed code, and gives a more precise cause of any failure.

```
MQHCONN HCon; // Connection handle
MQHOBJ HObj1; // Object handle for queue 1
MQHOBJ HObj2; // Object handle for queue 2
MQLONG CompCode, Reason; // Return codes
MQLONG options;
MQOD od1 = {MQOD_DEFAULT}; // Object descriptor for queue 1
MQOD od2 = {MQOD_DEFAULT}; // Object descriptor for queue 2
MQMD md = {MQMD_DEFAULT}; // Message descriptor
MQPMO pmo = {MQPMO_DEFAULT}; // Put message options
MQGMO gmo = {MQPMO_DEFAULT}; // Get message options

// 1 Connect application to a queue manager.
strcpy (QMName,"QMA");
MQCONN (QMName, &HCon, &CompCode, &Reason);

// 2 Open a queue for output
strcpy (od1.ObjectName,"QUEUE1");
MQOPEN (HCon,&od1, MQOO_OUTPUT, &HObj1, &CompCode, &Reason);

// 3 Open input queue
options = MQOO_INPUT_AS_Q_DEF;
strcpy (od2.ObjectName, "QUEUE2");
MQOPEN (HCon, &od2, options, &HObj2, &CompCode, &Reason);

// 4 Put a message on the queue
strcpy(mqmd.ReplyToQ,"QUEUE2");
pmo.Options |= MQPMO_NO_SYNCPOINT;
MQPUT (HCon, HObj1, &md, &pmo, 100, &buffer, &CompCode, &Reason);

// 5 Close the output queue
MQCLOSE (HCon, &HObj1, MQCO_NONE, &CompCode, &Reason);

// 6 Get message
gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT;
gmo.WaitInterval = 10 * 1000;
buflen = sizeof(buffer - 1);
memcpy (md.MsgId, MQMI_NONE, sizeof(md.MsgId));
memset (md.CorrelId, 0x00, sizeof(MQBYTE24));
MQGET (HCon, HObj2, &md, &gmo, buflen, buffer, 100, &CompCode, &Reason);

// 7 Close the input queue
options = 0;
MQCLOSE (HCon, &HObj2,options, &CompCode, &Reason);

// 8 Disconnect from queue manager
MQDISC (HCon, &CompCode, &Reason);
```

Figure 4-5 A code fragment

Comments:

1 This statement connects the application to the queue manager with the name QMA. If the parameter QMName does not contain a name, the default queue manager is used. A handle that references the queue manager is stored in the HCon variable. This handle must be used in all subsequent verbs that are used on this connection.

2 To open a queue, the queue name must be moved into the object descriptor that is used for that queue. This statement opens QUEUE1 for output only (open option MQOO_OUTPUT). The handle to the queue and values in the object descriptor are returned. The handle Hobj1 must be specified in the MQPUT.

3 This statement opens QUEUE2 for input only by using the queue-defined defaults.

For simplicity in this example, a predefined local queue was assumed. In practice, many applications use a model queue in the MQOPEN. In that case, returned values from the MQOPEN contain the real name of a dynamically created queue, and that value is put into the ReplyToQ field in the MQMD.

4 MQPUT places the message assembled in a buffer on a queue. MQPUT features the following parameters:

- ▶ The handle of the queue manager (from MQCONN).
- ▶ The handle of the queue (from MQOPEN).
- ▶ The message descriptor.
- ▶ A structure that contains options for the put. The MQPMO_NO_SYNCPOINT flag explicitly ensures that the operation does not start a transaction.
- ▶ The message length.
- ▶ The buffer that contains the data.

5 This statement closes the output queue. Because the queue is predefined, no close processing takes place (MQOC_NONE).

6 For the get, the MQGMO_WAIT option is used to wait for up to 10 seconds for the reply message. The MQGET needs the length of the buffer as an input parameter. Because there are no other criteria specified, the first message from the queue is read. If no message arrives within the timeout, the Reason code is 2033 (MQRC_NO_MSG_AVAILABLE). The application must be able to manage such failures.

7 This statement closes the input queue.

8 The application disconnects from the queue manager.

4.5 Triggering

Triggering is a mechanism that is used by WebSphere MQ to automatically start applications only when there is work available for those applications to process. It can save system resources because an application does not need to be permanently running. Instead, the application perhaps can be sitting in a long-running MQGET call, waiting for messages to appear on its input queue.

It relies on several attributes that are set on queues, and a specialized application known as a *trigger monitor*. The trigger monitor's purpose is to wait until a controlling trigger message arrives, and then start the real application. Although this configuration might sound as though one long-running program was exchanged for another, a single trigger monitor can start many different applications.

There is usually one trigger monitor that is running for each type of application that needs to be started. For example, CKTI is a trigger monitor that can start only CICS transactions on z/OS while runmqdnm is a trigger monitor that starts .Net programs on Windows. For all distributed platforms, runmqtrm is the standard trigger monitor that starts local programs or scripts.

Configuring triggering

The following requirements are needed to trigger an application:

- ▶ The target queue that is used as input by the application to be started must have triggering conditions specified, as shown in the following example:

```
DEFINE QLOCAL(A_Q) REPLACE +
    TRIGGER +
    TRIGTYPE(first) +
    INITQ(SYSTEM.DEFAULT.INITIATION.QUEUE) +
    PROCESS(proc1) +
    DESCR('This is a triggered queue')
```

- ▶ The *process* object that is associated with the target queue defines the name and type of the program that must be started. The use of WINDOWSNT is an historic artifact but now refers to any currently supported Windows platform, as shown in the following example:

```
DEFINE PROCESS(proc1) REPLACE +
    DESCR('Process to start server program') +
    APPLTYPE(WINDOWSNT) +
    APPPLICID('c:\test\myprog.exe')
```

- ▶ There must be an initiation queue defined. This queue can be any local queue and does not need any special attributes. A trigger monitor is the only application that gets messages from this queue. If multiple trigger monitors are running, each must have its own initiation queue defined.
- ▶ The triggered application must be written to expect a defined set of parameters on the command line. It should also be written to behave in a certain way. Any triggered application must use a short wait time when input messages are retrieved, and to loop around its input queue until there are no more messages and the wait time expires. The application developer must also expect occasions when there are no messages available for the program to process.

Figure 4-6 shows the logic that is needed in a triggered application.

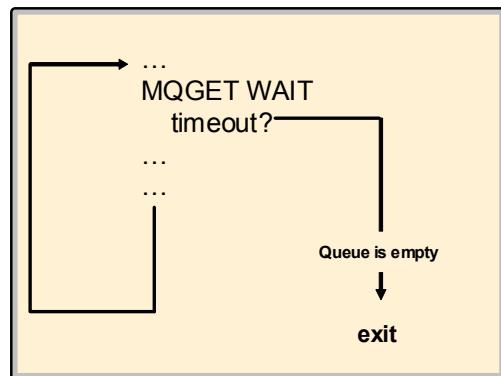


Figure 4-6 A triggered application

All triggered applications must loop until the input queue is empty and include a non-zero timeout on the MQGET.

How triggering works

Figure 4-7 shows the logic of triggering. Here, Program A sends a message to A_Q to be processed by Program B. Program B is not running when the message is sent and needs to be started.

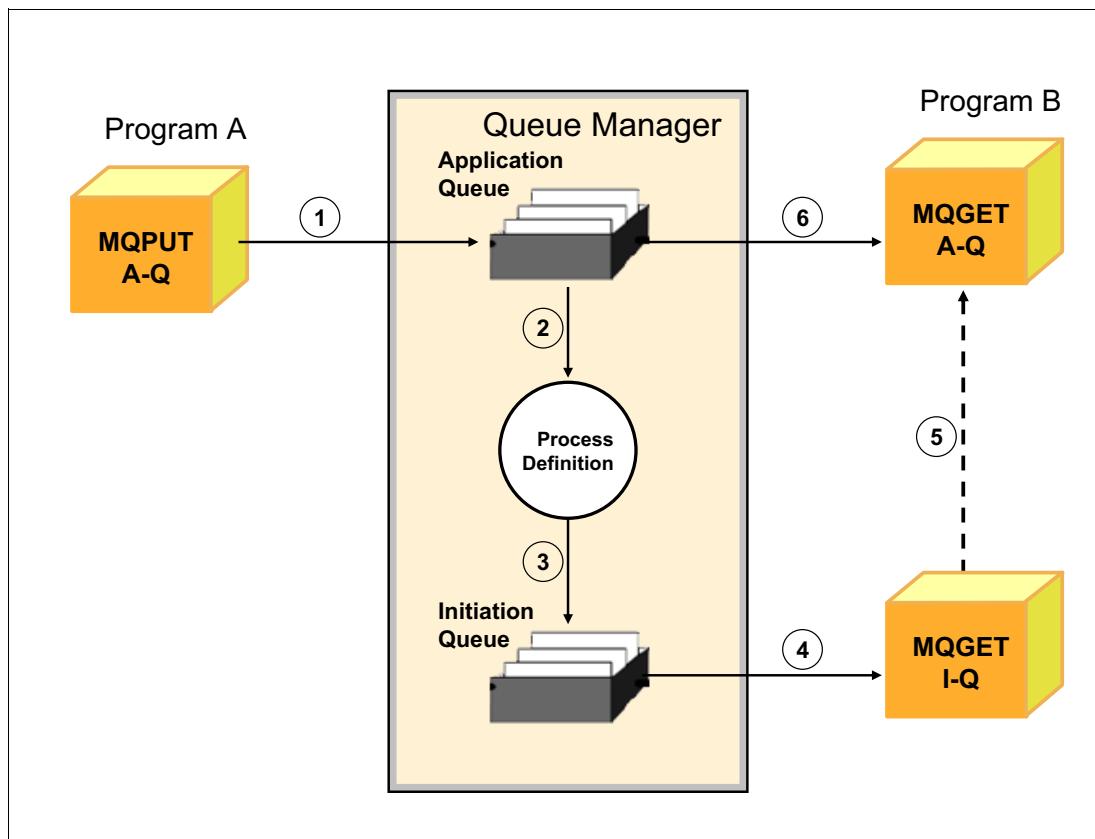


Figure 4-7 Triggering an application

The WebSphere MQ triggering mechanism uses the following process:

1. Program A issues an MQPUT verb and puts a message into A_Q for Program B.
2. The queue manager processes this verb and puts the message into the application queue.
3. The queue manager also finds out that the queue is triggered. It creates a trigger message and looks in the process definition that is named in the queue definition to find the name of the required application and then insert it into the trigger message. The trigger message is put into the initiation queue.
4. The trigger monitor gets the trigger message from the initiation queue.
5. The trigger monitor then starts the program specified.
6. The application program starts running and issues an MQGET verb to retrieve the message from the application queue.

The trigger type that is used depends on how the application is written. The following choices are available, but the recommendation is to use TRIGGER(FIRST), unless there are compelling reasons to choose one of the other values:

► FIRST

A trigger message is put in the initiation queue only when the target queue changes from empty to non-empty.

► EVERY

Every time a message is put in the target queue, a trigger message is also put in the initiation queue.

► n messages

A trigger message is put in the initiation queue when there are n messages in the target queue. For example, it might be worth starting a batch program only when the queue holds 1,000 messages.

Regardless of the chosen trigger type, any triggered application must follow the rule about looping until there is no more input. Even with TRIGGER(FIRST), no application should assume that there will always be only a single input message.

4.6 Configuring a WebSphere MQ client

Many WebSphere MQ applications are run as clients, which connect to a queue manager on a different machine. This section describes how to define and test the connection between a WebSphere MQ client and its server.

Figure 4-8 shows that the MQ client is installed in the client machine. Clients and servers are connected with MQI channels. An MQI channel consists of pair of definitions, the Client Connection (CLNTCONN) and Server Connection (SVRCONN) channels.

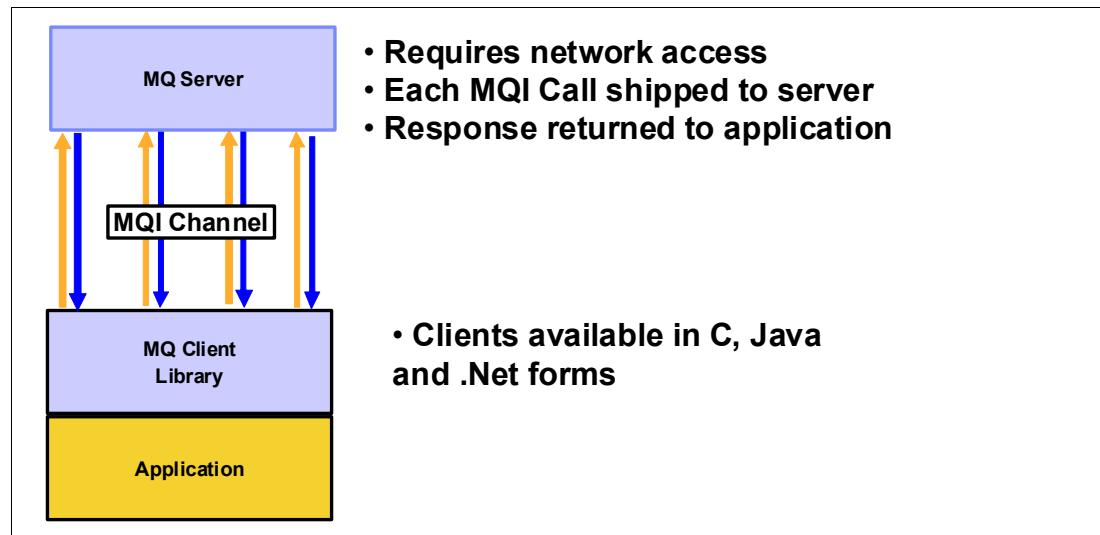


Figure 4-8 A client/server connection

How to define a client/server connection

There are three different client implementations (in C, Java, and .Net) that are suitable for different application environments. The principles are the same regardless of language. The main difference is how to define the connection to the queue manager.

C clients

The simplest way to define the client connection channel is to set the MQSERVER environment variable, as shown in the following example:

```
set MQSERVER=CHAN1/TCP/9.24.104.206(1414)
```

Where:

- ▶ MQSERVER is the name of the environment variable. This name is fixed.
- ▶ CHAN1 is the name of the channel to be used for communication between client and server. There must be a corresponding SVRCONN channel of this name defined on the server.
- ▶ TCP denotes that TCP/IP is to be used to connect to the machine with the address that follows the parameter.
- ▶ 1414 is the default port number for WebSphere MQ. It is not required if the listener on the server side is also using this default value. However, specifying the port number makes it clearer if changes are needed in future.

Java clients

For the WebSphere MQ client for Java, equivalents to the environment variables are set in the application code. The following example shows the simplest statements to include in a Java program:

```
import com.ibm.mq.*;
MQEnvironment.hostname = "9.24.104.456";
MQEnvironment.channel = "CHAN1";
MQEnvironment.port = 1414;
```

.Net clients

As with the Java clients, variables can be set in the application program to define the connection, as shown in the following example:

```
properties = new Hashtable();
properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
properties.Add(MQC.HOST_NAME_PROPERTY, hostName);
properties.Add(MQC.PORT_PROPERTY, port);
properties.Add(MQC.CHANNEL_PROPERTY, channelName);
```

More complex configurations

There are many more ways to define client configurations, and there are many more attributes possible on these channels. For example, secure configurations might require SSL/TLS definitions. Designs for high availability might use groups of definitions and multiple addresses to connect to.

These other attributes cannot be set by using the environment variables. Instead, often they are set by creating the Client Channel Definition Table (CCDT). The CCDT is a file that is created by defining CLNTCONN channels in a queue manager. The generated file is then copied to the machine where the client program is running and referenced during the connection process. An administrator that is using the CCDT enters the following command:

```
DEFINE CHANNEL('CHAN1') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('9.24.104.206(1414)')
```

JMS and XMS programs also include the option of the use of the CCDT or similar definitions that are created in a JNDI repository.

The CCDT and JNDI mechanisms remove the need for connection information from the client application programs. Therefore, often it is easier to change connection information administratively. But, all client environments also include the option of programmatically defining the full connection details.

Configuring the queue manager

On the queue manager, the following minimum required definition for the corresponding SVRCONN channel is needed:

```
DEFINE CHANNEL('CHAN1') CHLTYPE(SVRCONN)
```

4.7 Security

This publication is not the place for a discussion of security. However, the following basic points must be made, especially those points that are related to client connections:

- ▶ From WebSphere MQ V7.1 onwards, the default configuration blocks most client connections. This block can be disabled by using the **ALTER QMGR CHLAUTH(DISABLED)** command, but this use is discouraged.
- ▶ To get going quickly with a WebSphere MQ client without completely disabling the channel security rules, set an explicit user ID for the SVRCONN channel, as shown in the following example:

```
DEFINE CHANNEL('CHAN1') CHLTYPE(SVRCONN) MCAUSER('userA')
```

Where userA is a user ID that is defined on the machine where the queue manager is running. If it is also the user ID under which the client application is running, the following command permits client connections:

```
SET CHLAUTH('CHAN1') TYPE(USERMAP) CLNTUSER('userA') USERSRC(MAP)  
MCAUSER('userA') ADDRESS('*') ACTION(ADD)
```

- ▶ The userA identity needs access to the queue manager and resources, such as queues. For the program fragment that is shown in Figure 4-5 on page 46, the following commands are needed on a distributed platform if it connects as a client:

```
setmqaut -t qmgr -m QMA -p userA +connect  
setmqaut -t q -n QUEUE1 -p userA +put  
setmqaut -t q -n QUEUE2 -p userA +get
```

If you are connecting to a z/OS queue manager, the external security manager, such as RACF®, needs comparable definitions. Also, on UNIX and Linux systems, security is group-based, so applying authorizations to a user actually results in changes to group permissions.

- ▶ Client channels can be protected by using SSL/TLS certificates and algorithms. Configuring those values is also beyond the scope of this chapter.

4.8 Configuring communication between queue managers

Figure 4-9 shows the key parts and architecture of WebSphere MQ when a message is sent between two systems. The application programs use the MQI to communicate with the queue manager. The queuing system consists of the following parts:

- ▶ Queue Manager
- ▶ Listener
- ▶ Trigger Monitor (optional)
- ▶ Channel Initiator
- ▶ Message Channel Agent (MCA) or mover

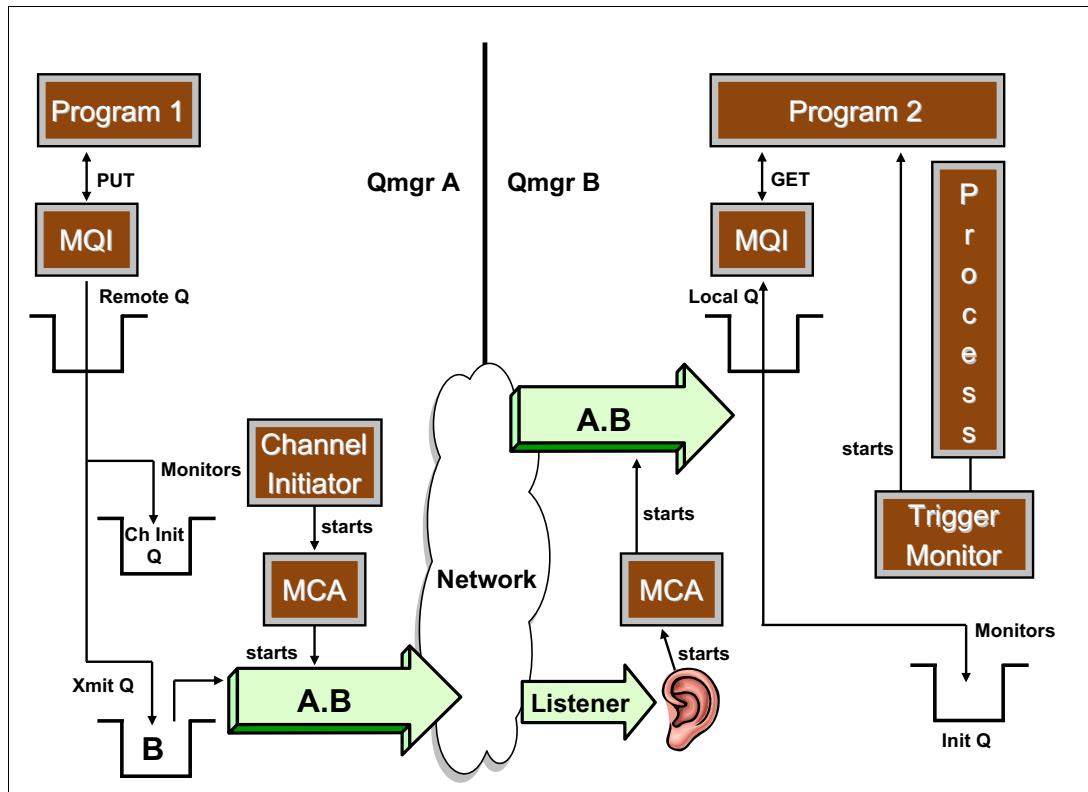


Figure 4-9 Sending messages to a remote application

When the application program wants to put a message on a queue, it issues an MQOPEN call followed by an MQPUT. The queue manager checks whether the queue referenced in the MQOPEN is local or remote.

If it is a remote queue, the subsequent MQPUT of a message is placed into the transmission (xmit) queue. The queue manager adds a header that contains information from the remote queue definition, such as the destination queue manager name and destination queue name.

There does not need to be a remote queue definition if the application explicitly completes the name of the remote queue manager. The queue manager then works out which transmission queue to use, or returns an error that there is no suitable transmission queue defined. This approach is often used by applications that are sending reply messages and removes the need for explicit definitions of all possible reply queues.

Important: Each remote queue must be associated with a transmission queue. Usually, all messages that are destined for one remote queue manager use the same transmission queue. The transmission queue is usually given the same name as the remote queue manager.

Transmission is done via channels. Channels can be started manually or automatically. To start a channel automatically, the transmission queue must be associated with a channel initiation queue. Figure 4-9 on page 54 shows that the queue manager puts a message into the transmission queue and another message into the channel initiation queue. This queue is monitored by the *channel initiator*.

The channel initiator is part of WebSphere MQ that must be running to monitor initiation queues. When the channel initiator detects a message in the initiation queue, it starts the message channel agent (MCA) for the particular channel. This program moves the message over the network to the other machine by using the sender part of the unidirectional message channel pair.

On the receiving end, a *listener* program must be started. The listener, which is also supplied with WebSphere MQ, monitors a specified port. By default, this port is the port that is assigned to WebSphere MQ, 1414. When a connection request arrives from the sending queue manager, the listener starts the receiving channel. This receiving MCA moves the message into the specified local queue from where the receiving application gets the message.

Important: Both channel definitions, sender and receiver, must have the same name. Create a symmetric set of definitions with a different channel name for the reverse path to send a reply to the original application.

The program that processes the incoming message can be started manually or automatically. Use triggering to start the program automatically.

4.8.1 How to define a connection between two systems

Figure 4-10 shows the required objects for connecting two queue managers. The definitions assume that two applications are communicating via fixed queue names.

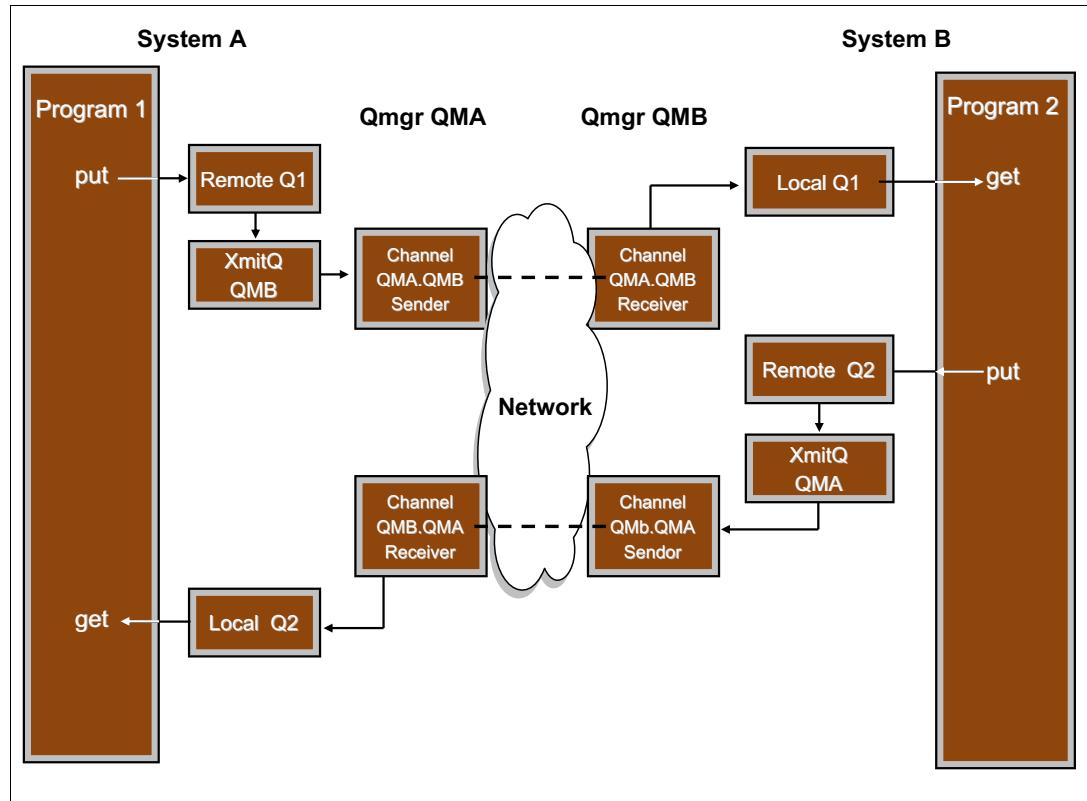


Figure 4-10 Communication between two queue managers

Each system needs the following components:

- ▶ A remote queue definition that mirrors the local queue in the receiver machine and links to a transmission queue (Q1 in system A and Q2 in system B).
- ▶ A transmission queue that holds all messages that are destined for the remote system until the channel transmits them (QMB in system A and QMA in system B).
- ▶ A sender channel that gets messages from the transmission queue and transmits them to the other system by using the existing network (QMA.QMB in system A and QMB.QMA in system B).
- ▶ A receiver channel that receives messages and puts them into a local queue (QMB.QMA in system A and QMA.QMB in system B).
- ▶ A local queue from which the program gets its messages (Q2 in system A and Q1 in system B).

In each system, you must define the appropriate queue manager objects. The objects are defined in the two columns that are shown in Table 2 on page 57. The queue managers must be running before the objects are defined.

Important: The use of WebSphere MQ clustering removes the need for some of these definitions, but requires more initial setup. Adding new systems to an existing cluster is easy, but getting started with a cluster is beyond the scope of this chapter.

The channels can then be configured to start automatically or manually.

Table 2 Objects defining connection between two queue managers

System A (QMA)	System B (QMB)
DEFINE QREMOTE(Q1) + RNAME(Q1) RQMNAME(QMB) + XMITQ(QMB)	DEFINE QLOCAL(Q1)
DEFINE QLOCAL(QMB) + USAGE(xmitq)	
DEFINE CHANNEL(QMA.QMB) + CHLTYPE(sdr) + XMITQ(QMB) + TRPTYPE(tcp) + CONNNAME(machine2)	DEFINE CHANNEL(QMA.QMB) + CHLTYPE(rcvr) + TRPTYPE(tcp)
DEFINE QLOCAL(Q2)	DEFINE QREMOTE(Q2) + RNAME(Q2) RQMNAME(QMA) + XMITQ(QMA)
	DEFINE QLOCAL(QMA) + USAGE(xmitq)
DEFINE CHANNEL(QMB.QMA) + CHLTYPE(rcvr) + TRPTYPE(tcp)	DEFINE CHANNEL(QMB.QMA) + CHLTYPE(sdr) + XMITQ(QMA) + TRPTYPE(tcp) CONNAME(machine1)

Starting communication manually

After the objects are defined on the queue managers, the sender channels in each direction must be started. Figure 4-11 shows the commands that are used to start the listener and channel for queue manager QMA.

```
C> strmqm QMA
WebSphere MQ queue manager started
...
C> start runmqlsr -t tcp -m QMA -p 1414
C> runmqsc QMA
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QMA.

start channel (QMA.QMB)
  1: start channel (QMA.QMB)
AMQ8018 start WebSphere MQ channel accepted
end
  2 : end
1 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.

C>
```

Figure 4-11 Manually starting a queue manager and channels

The first command starts queue manager QMA. The next command starts the listener. It listens on behalf of QMA on port 1414. On Windows, the **start** command runs the listener in a separate window; on UNIX platforms, use the & character to run the listener in the background. The third command starts **rwmqsc** in interactive mode and the channel QMA.QMB is started.

For the other queue manager, issue equivalent commands. Also, start the applications in both systems.

How to start communication automatically

The channel initiator is a special version of a trigger monitor that is used to start channels. By default, the channel initiator is automatically started when the queue manager starts. The only other requirement for channels to be automatically started is to set the trigger attributes on the transmission queue.

To have the transmission queue triggered, add the parameters that are shown in the following example in bold:

```
DEFINE QLOCAL(QMB) REPLACE +
    USAGE(xmitq) +
TRIGGER
TRIGTYPE(first) +
INITQ(SYSTEM.CHANNEL.INITQ)
```

Fourth parameter: On z/OS, a fourth parameter is needed that names the channel that is to be started. Add the TRIGDATA(QMA.QMB) parameter to the command that is shown in the previous example.

The channel now starts automatically when there are any messages to be transferred.

The DISCINT attribute of a channel controls whether a channel should stop if there are no more messages to be transferred. This attribute is used to release resources, such as network sockets and memory that might not be needed. Triggering then restarts the channel with no manual intervention if a message arrives on the transmission queue.

4.9 Summary

This chapter provided an outline of many of the core aspects of WebSphere MQ, including how to administer it and write application programs. There are many more capabilities in the product to explore, but understanding these basic components is a good start towards learning the next level of detail.



What is new in WebSphere MQ V7.1 and V7.5

WebSphere MQ V7.1 was made available in November 2011 and V7.5 in June 2012. This chapter briefly describes the new features of these releases. The remainder of this book describes detailed scenarios that demonstrate many of these features.

This chapter contains the following sections:

- ▶ What is new in WebSphere MQ V7.1
- ▶ What is new in WebSphere MQ V7.5

5.1 What is new in WebSphere MQ V7.1

A one-word summary of the features that were introduced or updated in WebSphere MQ V7.1 is *simplification*. With this release, it becomes much easier to install, configure, migrate, manage, and use queue managers.

This section gives a brief description of the content of the release.

5.1.1 Multiple installation support

With V7.1, you can install more than one copy of WebSphere MQ on UNIX, Linux, and Windows platforms (z/OS already features this capability). The ability to install more than one copy of WebSphere MQ results in the following advantages:

- ▶ This ability simplifies migration strategies, as you can continue to use one version of WebSphere MQ and only gradually migrate applications to a new version, without the need for parallel hardware.
- ▶ When WebSphere MQ is installed, a system administrator can choose the directory into which the code is installed. There is no longer a requirement to use /opt/mqm or /usr/mqm on the UNIX or Linux platforms.
- ▶ Third-party applications can embed WebSphere MQ under their own private directory. They also can choose which versions of WebSphere MQ they support without worrying about the visible version of WebSphere MQ that a user might be using.
- ▶ A system administrator can leave an existing copy of WebSphere MQ V7.0.1.6 (or later) on a machine, and this new feature works alongside it. There is no need to move to V7.1 before you start to use the multiple installation capabilities.

Queue managers include a new attribute, VERSION, which shows the level of function of the queue manager. It is accurate to a fix pack level (going beyond the CMDLEVEL attribute) and can be programmatically queried.

5.1.2 Enhanced security

WebSphere MQ V7.1 provides channel authentication records that define rules that are to be applied when a queue manager or client attempts to connect through a channel. A number of elements about the partner can be checked, and choices are made about whether to allow the connection to proceed. For example, you can block connections from specific IP addresses or block connections from specific user IDs. Many of these functions were previously available only by using channel exits. It is now much simpler to configure and manage the definitions.

SSL/TLS channels can use newer and stronger cryptographic algorithms.

You can grant access explicitly to remote clustered queues instead of giving users access to the SYSTEM.CLUSTER.TRANSMIT.QUEUE. This feature improves the granularity of access control. This ability already was possible on z/OS and there is now a consistent security model across all platforms.

For more information about the new security features, see *Secure Messaging Scenarios with WebSphere MQ*, SG24-8069.

5.1.3 Enhanced clustering

A new option for clustered messages, called *bind on group*, allows application programs to ensure that groups of messages are delivered to the same destination within a WebSphere MQ cluster.

A new sample program, `amqsc1m`, monitors clustered queues to ensure that messages are being processed. Source code for this sample is shipped, so it is easy to modify and extend.

5.1.4 Multicast protocol for publish/subscribe

The WebSphere MQ Clients now support a multicast protocol for higher performance publish/subscribe operations. All subscribers to a topic can see a publication at the same time. The protocol also gives greater scalability because more subscribers do not cause more traffic on a network; the same single transmission in the network is seen by all subscribers.

Most existing publish/subscribe applications that use the C client libraries (not Java or .Net) can use this feature without any code change; only administrative actions are needed to enable multicast.

5.1.5 Improved availability and scalability on z/OS

A new design for how large messages on shared queues are stored improved performance, reduced CPU costs, and increased the capacity of these queues. This design uses VSAM instead of DB2 BLOBs and is known as the shared message data set feature.

Queue managers can now reconnect and recover from coupling facility or communication failures for administration and application structures. The queue managers can be configured to tolerate an outage or behave as they did in prior releases.

CICS 4.2 now supports *Group Attach*, which allows a single CICS region to connect to any one of the queue managers in the same LPAR that are part of the same queue-sharing group. This feature improves availability because the CICS region does not need reconfiguration if one queue manager is not running.

IMS health status reports are recognized by a queue manager, so unexpected bursts of messages do not overwhelm IMS.

5.1.6 Improved performance on distributed platforms

Benchmarks show performance improvements on almost all tests run, when comparing WebSphere MQ V7.1 to V6.0 and V7.0. Performance reports for most platforms are available at this website:

<http://www-01.ibm.com/support/docview.wss?uid=swg27007197>

For example, SupportPac MP6Q for AIX shows the following results:

- ▶ When testing 2 Kb non-persistent messages in local, client, and distributed queuing environments, Version 7.1 has 28% higher throughput than V6.0.2.11, 30% higher throughput than V7.0, and 35% higher than V7.0.1.6
- ▶ When testing 2 Kb persistent messages in local, client, and distributed queuing environments, Version 7.1 has 64% higher throughput than V6.0.2.11, 36% higher throughput than V7.0, and 48% higher throughput than V7.0.1.6

Changes to the design of the WebSphere MQ Explorer reduce its footprint and improve performance. For example, connecting to 100 queue managers within the Explorer shrunk from 53 seconds to 7 seconds on a test machine.

5.1.7 Management of distributed platforms

A new command, `dmpmqcfg`, is available to extract queue manager configuration details so that definitions can be easily restored or replicated. This command replaces the MS03 `saveqmgr` SupportPac that many customers used. It is equivalent to the z/OS `MAKedef` command. This command can run as a client that is connecting to older queue manager versions, so there is no need to upgrade all systems before you start to use the command.

Multi-instance queue managers on Windows no longer require domain controller privileges.

Security commands, such as `setmqaut`, now include equivalent MQSC commands. Administrators do not need to drop out of the MQSC command environment to set access control on resources, such as created queues.

5.1.8 Support for cloud environments

The WebSphere MQ Hypervisor Editions make it simple to deploy and replicate common configurations of queue managers.

Queues and Topics that are used by web applications that use JMS messaging can be automatically created when those applications are deployed by using IBM Workload Deployer or the IBM PureApplication™ Systems.

These capabilities support patterns often called Virtual Systems (also known as Infrastructure as a Service, IaaS) and Virtual Applications (also known as Platform as a Service, PaaS). The patterns feature a wide range of uses, including setting up of development, test, and production environments.

5.1.9 Application activity reports

The queue managers on Distributed platforms generate events that trace the MQI calls that are made by an application. This feature can be used in various ways; for example, to keep an audit trail of applications, to determine which queues or topics are being used, or to carry out problem determination when applications misbehave.

5.1.10 Clients

There is no longer a requirement to compile or link applications twice when they might be used as clients and when they connect to queue managers on the same machine. The MQI libraries can dynamically switch between behaviors.

API exits are available on the C clients, which matches the interface that is available for locally attached applications.

A client is now supported on the System i® platform, which allows RPG and C programs to connect to remote queue managers.

5.1.11 Channels

The following enhanced features for channels are available in this release:

- ▶ Channel version

The channel status reports show the version of client or queue manager that is at the other end of the channel. This configuration helps system administrators determine whether other systems should be upgraded.

- ▶ Batch limit

A new mechanism for batching messages that are sent between queue managers gives more predictable latency when there is a mixture of large and small messages. In addition to counting the number of messages (*batch size*), the channels can also count the total number of bytes that are transferred (*batch limit*). Whichever is reached first causes completion of the batch.

- ▶ Per-channel dead letter queue

When there are problems with sending messages across a channel (for example, when the target queue is full) the behavior of the channel depends on whether a dead letter queue (DLQ) is defined on the queue manager. Previously, this setting was a global setting for all channels on a queue manager.

Applications that required the behavior that is associated with having no DLQ were configured to use a different queue manager than those applications that preferred to have a DLQ. With V7.1, each channel on the queue manager can include an independent setting. The USEDLQ attribute tells some channels to behave as though there is no DLQ on the queue manager, even if one is defined.

5.2 What is new in WebSphere MQ V7.5

The primary goal of the V7.5 release was to simplify the installation and maintenance experience when multiple products from the WebSphere MQ family were used. Advanced Message Security (AMS) and Managed File Transfer (formerly known as WebSphere MQ File Transfer Edition) already were using the same mechanisms as the base WebSphere MQ product on z/OS. Because of this configuration, there was no need to ship a V7.5 on that platform. Therefore, V7.5 is on Windows, UNIX, and Linux only. With this integration, the *Edition* word in WebSphere MQ File Transfer Edition no longer made sense so that component was renamed Managed File Transfer (MFT).

However, there still are some new functions in this release, which are described in the following sections:

- ▶ 5.2.1, “Integrated installation”
- ▶ 5.2.2, “Enhanced clustering”
- ▶ 5.2.3, “Java application identification” on page 64
- ▶ 5.2.4, “Simplified AMS integration” on page 64
- ▶ 5.2.5, “RFC 5280 certificate validation policy” on page 64
- ▶ 5.2.6, “Managed File Transfer enhancements” on page 65

5.2.1 Integrated installation

AMS and File Transfer components were added to the installation images for WebSphere MQ. AMS and MFT must still be licensed if you use these capabilities. This change was made to have common installation and maintenance mechanism, which makes it much easier to work with them. There is a single Information Center containing documentation for all of these features. Licensing can be further simplified by ordering the WebSphere MQ Advanced product. This product includes the same function, but gives a single part number for ordering all the licensable components at once.

5.2.2 Enhanced clustering

To improve application isolation and make it easier to monitor the traffic that is being sent to different clustered destinations, a queue manager can automatically create multiple transmission queues. Instead of all clustered traffic sharing SYSTEM.CLUSTER.TRANSMIT.QUEUE, independent queues can be used.

5.2.3 Java application identification

When WebSphere MQ Java client programs connect to a queue manager, now they can be individually identified. Instead of appearing as WebSphere MQ Client for Java, the name of the program can be set explicitly or automatically derived from the main class of the program.

Figure 5-1 shows two Java programs that are connected to a single queue manager. One of these programs uses the V7.5 Java client and features a non-default application name; the other program uses an older client.

Applications connected to "V7 on 'rockall(2414)'":		
App name	App type	App description
MQ Explorer 7.5.0	Queue manager	WebSphere MQ Channel
WebSphere MQ Client for Java	Queue manager	WebSphere MQ Channel
runmqchi	Channel initiator	WebSphere MQ Channel Initiator
amqrrmfa	Queue manager	WebSphere MQ Cluster Repository
amqpccsea	Queue manager	WebSphere MQ Command Server

Figure 5-1 Two versions of Java client that are connected to a queue manager

5.2.4 Simplified AMS integration

Previous implementations of AMS used various techniques to intercept application calls to WebSphere MQ and apply the wanted level of protection to messages. This process was simplified, with the AMS function now built into the V7.5 application libraries and only queue manager configuration is needed to enable AMS. For example, there is no longer a requirement to configure an API Exit on queue managers.

5.2.5 RFC 5280 certificate validation policy

The specification of how SSL/TLS certificates are validated changed over time. WebSphere MQ version 7.5 allows control of how strictly the certificate chain validation conforms to the latest industry security standard in this area, RFC 5280.

5.2.6 Managed File Transfer enhancements

The logger component of MTF records file transfer operations to long-term storage for permanent archiving and later audit reviews. MTF originally wrote only to relational databases, such as DB2 and Oracle. With WebSphere MQ V7.5, the logger can now write to a flat text file as an alternative storage mechanism. This configuration makes it easier to read directly instead of writing SQL queries to extract the data.

It is easier to control starting and stopping of MFT agent processes.

WebSphere MQ V7.1 new features and enhancements

WebSphere MQ V7.1 introduces some new features and enhancements to some of the existing features. This part describes the new features and enhancements including, the ability to install multiple copies of WebSphere MQ on a single machine and enhancements to channel access security that use channel authentication records.

Also described in this part are the new installation options that are available with WebSphere MQ V7.1.

This part consists of the following chapters:

- ▶ Chapter 6, “WebSphere MQ V7.1 product installation enhancements” on page 69
- ▶ Chapter 7, “Multiple installation support on Windows, UNIX, and Linux” on page 75
- ▶ Chapter 8, “Enhanced security” on page 99
- ▶ Chapter 9, “Granular control over dead-letter queue usage” on page 133
- ▶ Chapter 10, “Dumping and restoring a queue manager configuration” on page 143
- ▶ Chapter 11, “Shared message data set and message offloading (z/OS)” on page 155
- ▶ Chapter 12, “Coupling facility connectivity loss improvements (z/OS)” on page 185
- ▶ Chapter 13, “Extended integration with IMS (z/OS)” on page 203
- ▶ Chapter 14, “CSQINPT DD added to queue manager startup JCL (z/OS)” on page 209
- ▶ Chapter 15, “CICS 4.2 group attach (GROUPUR)” on page 213



WebSphere MQ V7.1 product installation enhancements

This chapter describes the installation enhancements and new options that are available during WebSphere MQ V7.1 installation on Windows, UNIX, or Linux. The two features that are described here are the inclusion of WebSphere MQ Telemetry (on limited platforms) as an integrated option with the base product and relocatable installations.

This chapter contains the following sections:

- ▶ WebSphere MQ Telemetry integrated feature
- ▶ Relocatable installations for UNIX and Linux

6.1 WebSphere MQ Telemetry integrated feature

When WebSphere MQ V7.0.1 is used with WebSphere MQ Telemetry (MQTT), the installation steps are two distinct and separate tasks. WebSphere MQ V7.1 includes the MQTT installable package with its base product installation to ease the integration of the features. This section gives an overview of the MQTT functionality and describes the features of the WebSphere MQ V7.1 installation process.

For more information about the use of MQTT and the functionality it provides, see *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, SG24-8054.

6.1.1 WebSphere MQ Telemetry functional overview

The following IBM Redbooks Solution Guide provides an excellent starting point for understanding the functionality of MQTT:

<http://www.redbooks.ibm.com/abstracts/tips0876.html>

MQTT is a lightweight protocol that is supported by small, pervasive devices that enable them to communicate with each other and with other systems or applications. Combined with WebSphere MQ, those devices can connect and communicate with enterprise applications and web services that are hosted by WebSphere MQ.

WebSphere MQ Telemetry is split into two components that comprise the following features:

- ▶ The MQTT service that runs on the WebSphere MQ server.
- ▶ The MQTT clients that are distributed to remote devices and applications.

For more information about the functionality of WebSphere MQ Telemetry when used with WebSphere MQ V7.1, see the topic *Introduction to WebSphere MQ Telemetry* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/tt00001_.htm

6.1.2 WebSphere MQ V7.1 installation with MQTT

When WebSphere MQ V7.1 is installed, there is a new option to also install MQTT at the same time. Figure 6-1 shows the installation panel from a Windows machine. The details of the packages to install on the other supported distributed platforms are described in the next sections.

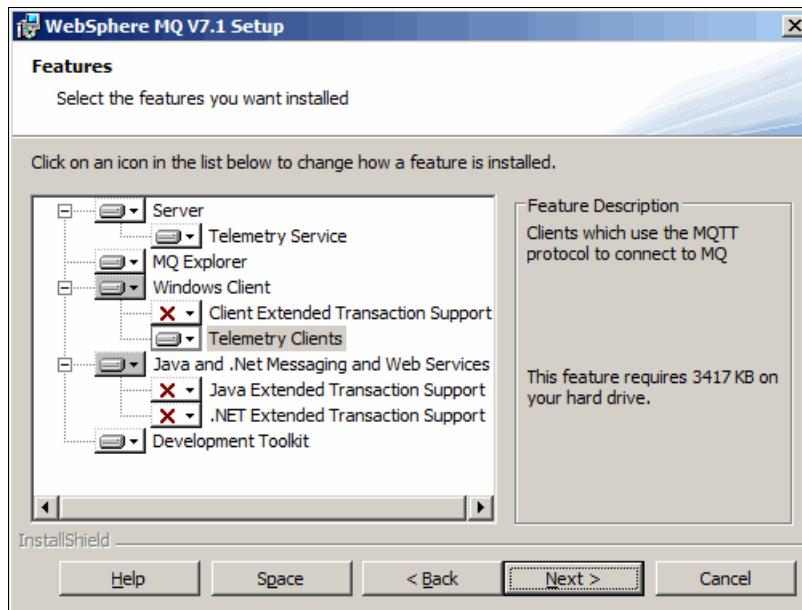


Figure 6-1 Selecting the WebSphere MQ Telemetry features for installation on Windows

After the WebSphere MQ installation program is started and accepting the license agreement, the options that relate to how to proceed with the installation are displayed. The first set of options is to install the Typical components, the Compact components (the minimum set required), or to proceed with the Custom option that allows the user to select exactly which components to install. Select **Custom** and click **Next**, and **Next** again to show the Features panel. Figure 6-1 shows the panel and the features that are available to select for installation.

The Telemetry Service and Telemetry Clients can be installed independently of one another. However, the clients cannot connect to a queue manager without the service present and running. Installing the service without the clients is a likely option if the clients are to be installed elsewhere for connection to this queue manager.

Click **Next** from the panel that is shown in Figure 6-1 to progress to the installation summary window before you proceed with the installation of the features that were selected.

The MQTT packages on AIX, Linux x86_64, and Linux s390x use the following names:

- ▶ AIX:
 - mqm.xr.clients
 - mqm.xr.service
- ▶ Linux:
 - MQSeriesXRClients
 - MQSeriesXRService

The MQTT service is not supported on any other platforms.

6.2 Relocatable installations for UNIX and Linux

WebSphere MQ V7.1 introduces the ability to install multiple copies of the product on the same Windows, UNIX, or Linux machine. Chapter 7, “Multiple installation support on Windows, UNIX, and Linux” on page 75 describes this feature in detail. This section focuses specifically on the ability to install WebSphere MQ in a non-default location on UNIX and Linux. The ability to install in a non-default location always was available when the product was installed on Windows.

6.2.1 Why relocatable installations are important for WebSphere MQ V7.1

Although it is possible to install WebSphere MQ V7.1 with WebSphere MQ V7.0.1.6 (or later), V7.0.1.6 does not support installations in a non-default location on the file system. This fact makes it necessary to be able to install the V7.1 product into a non-default location to avoid file conflicts and to allow a proper separation of the two versions on a single machine.

Similarly, installing multiple copies of WebSphere MQ V7.1 on the same machine requires a level of separation between the installations, which is facilitated by the installation location on the file system.

The benefits of the use of a relocated installation are clear when there are multiple installations to be made on a single machine. However, the relocatable installation feature can be used even if only one installation of WebSphere MQ V7.1 is required. This feature offers greater flexibility in the installation of WebSphere MQ for the user. In a single installation environment, there are no inherent benefits for installing into a non-default location.

6.2.2 How to install into a non-default location

For more information about the considerations that are necessary when you are planning the installation location of WebSphere MQ V7.1, see the topic *Choosing an installation location* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/aq10285_.htm

The installation location

When you are choosing an installation location, the platform differences must be considered.

When you are installing into a custom location, AIX appends /usr/mqm to the supplied path, such that if a custom location of /usr/my_mqm_inst is supplied, the actual installation location becomes /usr/my_mqm_inst/usr/mqm.

When you are installing into a custom location on other UNIX platforms or Linux, the installation path remains as specified, such that a custom location of /opt/my_mqm_inst means that the product is installed in /opt/my_mqm_inst.

Specifying the custom installation location

This section describes the method in which the custom installation location is specified during product installation. For more information about installation instructions for the product, see the topic *Installing a WebSphere MQ server* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/zi00100_.htm

Although this information refers to the installation of the server part of WebSphere MQ, the following details apply to installing the client feature of WebSphere MQ.

AIX

For more information about non-interactive installation of WebSphere MQ server on AIX, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/aq10345_.htm

For more information about installing WebSphere MQ server on AIX, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/aq10340_.htm

When you are running a non-interactive installation on AIX, the **installp** command is used. To specify a custom installation location, use the **-R** flag followed by the installation path that is required. Remember that **/usr/mqm** is added to the front end of the path. The following example installs the full product into **/usr/my_mqm_inst/usr/mqm**:

```
installp -R /usr/my_mqm_inst -acgXYd . all
```

When the interactive installation is used, the custom location is specified as one of the options in the smit panel.

Linux

For more information about the detailed instructions for installation on Linux, see the topic *Installing WebSphere MQ server on Linux* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/lq10440_.htm

When you are installing on Linux, the **rpm** command is used. To specify a custom installation, location use the **--prefix** flag followed by the installation path required. The following example installs the specified product components into **/opt/my_mqm_inst**:

```
rpm --prefix /opt/my_mqm_inst -ivh <PackagesToInstall>
```

HP on Itanium

For more information about the non-interactive installation of WebSphere MQ server on HP on Itanium (HP-UX), see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/hq10380_.htm

For more information about installing WebSphere MQ server on HP-UX, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/hq10370_.htm

When you are running a non-interactive installation on HP-UX, the **swinstall** command is used. To specify a custom installation location, use the **,l=** flag followed by the installation path that is required. The following example installs the full product into **/opt/my_mqm_inst/**:

```
swinstall -s /path/to/installation_file.v11 MQSERIES,l=/opt/my_mqm_inst
```

When you are using the interactive installation, the same **swinstall** command is used, but without package specification. The following command launches the interactive installation on HP-UX:

```
swinstall -s /path/to/installation_file.v11
```

To configure the custom installation location by using the interactive installation, select **Actions** → **Change Product Location** and specify the location that is required.

Solaris

For more information about non-interactive installation of WebSphere MQ server on Solaris, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sq10390_.htm

For more information about installing WebSphere MQ server on Solaris, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sq10380_.htm

When you are running a non-interactive installation on Solaris, the script `silent.sh` that is provided with the WebSphere MQ installation media is used with a response file. To install the product or product components into a custom location, the `silent.sh` script must be edited. The attribute to be edited is the `MQ_INSTALLATION_PATH`. Editing the `silent.sh` script, as shown in the following example, installs the product into `/opt/my_mqm_inst/`:

```
MQ_INSTALLATION_PATH=/opt/my_mqm_inst
```

When you are using the interactive installation, the `pkgadd` command is used and the user is prompted to enter the intended installation location.

6.2.3 Further considerations for custom location installations

There is a direct impact on applications that use WebSphere MQ when the installation location is not the default. When an application runs, it loads one or more WebSphere MQ libraries. Because those libraries can be installed into a custom location, there are considerations that must be made about how the application locates the libraries it needs. These considerations also are necessary when there is a single installation that is not primary.

For more information about these considerations on a platform-by-platform basis, see 7.3, “Managing applications” on page 88.



Multiple installation support on Windows, UNIX, and Linux

This chapter describes the capability to have more than one installation of WebSphere MQ on a single machine at the same time. Although this ability was always possible on z/OS, WebSphere MQ V7.1 enables multiple installations on Windows, UNIX, and Linux for the first time.

This chapter contains the following sections:

- ▶ Managing installations
- ▶ Managing queue managers
- ▶ Managing applications

7.1 Managing installations

This section describes how to install multiple copies of WebSphere MQ on a single machine, determine what copies are installed and how to administer those copies.

Each copy of WebSphere MQ can be a client installation or a server installation.

A machine can be a physical system or a virtual image, such as a logical partition (LPAR).

7.1.1 Installing multiple copies of WebSphere MQ

To enable multiple installations of WebSphere MQ to run concurrently on the same machine, the following requirements must be met:

- ▶ It must be possible to install the product multiple times.
- ▶ Each copy of WebSphere MQ must operate without adversely affecting other installations. These effects include inter-process communication and data that is persisted on disk.

When you install WebSphere MQ, the product files are installed in the default location or a location of your choice. Each copy of WebSphere MQ must be installed in a separate directory to avoid conflicts. The ability to customize the WebSphere MQ installation location always was available on Windows. The location can now be customized on UNIX and Linux so that multiple installations are possible. For more information about how to customize the location where WebSphere MQ is installed on UNIX and Linux, see 6.2, “Relocatable installations for UNIX and Linux” on page 72.

Before the process that is used to install WebSphere MQ more than once is described, it is worth considering the following main reasons why you might want to install multiple copies of WebSphere MQ:

1. Migration from one version to another

If multiple installations are not possible, it is necessary to uninstall the existing version of the product before the version that is required can be installed. Uninstalling the existing installation requires a considerable outage to be scheduled, which can be disruptive to service level agreements. Further still, if problems are encountered during the migration, the previously working setup was removed, so it must be restored, which introduces more risk.

The ability to install multiple versions of WebSphere MQ allows you to individually migrate your queue managers and applications, which minimizes outages.

2. Applications or products that require (or are only supported on) certain versions

It is common to have several applications that use WebSphere MQ on the same machine. These applications might be self-written, or third-party products. Applications or third-party products often certify support for specific versions of prerequisite software on which they depend. If only one WebSphere MQ installation is possible, and one such application did not declare support for a version of WebSphere MQ, features, and enhancements that were introduced in that version cannot be used by other applications on the same machine.

The ability to install multiple copies of WebSphere MQ allows you to run applications that require different versions of WebSphere MQ, side-by-side, on the same machine.

Restrictions

You can install multiple copies of WebSphere MQ V7.1, or later, on a single machine. You can also have one installation of WebSphere MQ V7.0.1 at fix pack 6, or later. If you have WebSphere MQ V7.0.1.5, or earlier, you must upgrade before a second installation is possible.

WebSphere MQ V7.0.1 includes limited coexistence support, so is subject to the following restrictions:

- ▶ It must be installed in the default location on UNIX and Linux (/opt/mqm), except on AIX, where it is: /usr/mqm.
- ▶ It cannot be installed if another copy of the product is already installed.
- ▶ It is always the *primary installation*, which is described in 7.1.4, “Primary installation” on page 81.

The WebSphere MQ data path identifies the default location where queue manager logs and queue files are stored. The data path is also used to store other files, such as trace output, exits, and configuration data. All installations share a single data path, which means that every queue manager on a machine must have a unique name, irrespective of the installation with which it is associated. On Windows, the data path can be customized when the first instance of WebSphere MQ is installed; it is then fixed for all other installations. On UNIX and Linux, the location always is: /var/mqm.

On UNIX and Linux, root authority is required to install WebSphere MQ. In WebSphere MQ V7.1, or later, there are new commands with which you can administer installations. These commands also require root authority.

Repackaging WebSphere MQ by using crtmqpkg

On Linux and Solaris, an extra step is required to install other copies of WebSphere MQ because operating system restrictions prevent packages from being installed multiple times. To install a second (or subsequent) installation, you must repackage the installation files so they feature a unique package or file set name. A new command that is called **crtmqpkg** is used to repackage the installation files.

The **crtmqpkg** command takes a single parameter with which you can specify a unique suffix to be appended to the WebSphere MQ package names. This suffix can be any alphanumeric value 1 - 16 characters in length, as shown in the following example:

```
./crtmqpkg A
```

On Linux, this command generates new packages, as shown in the following example:

```
MQSeriesServer_A-7.1.0-0.x86_64.rpm
```

Naming convention: If you plan to use multiple installations, it is helpful to use a meaningful naming convention. Although you do not have to repackage the first installation, it can be helpful to do so for consistency.

When you run the **crtmqpkg** command, the generated packages are created in a subdirectory under /var/tmp on Linux, or /var/spool/pkg on Solaris, so you must ensure that there is sufficient space available.

Customized directory: You can customize the directory where the packages are generated by using the TMPDIR environment variable on Linux, or the OUTPUTDIR environment variable on Solaris. Set the environment variable by using the **export** command, before the **crtmqpkg** command is run, as shown in the following example:

```
export TMPDIR=/tmp/mqpackages
```

Shells, such as the **tcsh** command, use the **setenv** command instead of the **export** command.

7.1.2 Applying maintenance-level upgrades

To install maintenance-level upgrades, such as fix packs, you must identify the target installation to which you want to apply the updates. Updates are applied independently to each copy of WebSphere MQ, which means that different installations do not have to be at the same level of maintenance. This capability can be used to apply updates to applications and queue managers individually, instead of updating them all at the same time.

On Linux and Solaris, an extra step is required when maintenance upgrades are applied if the installation files were repackaged by using the **crtmqpkg** command. A new command that is called **crtmqfp** provides the equivalent capability for maintenance upgrades, as shown in the following example:

```
./crtmqfp A
```

On Linux, this command generates packages, as shown in the following example:

```
MQSeriesServer_A-U850308-7.1.0-1.x86_64.rpm
```

Specify suffix: You must specify the same suffix to **crtmqfp** as **crtmqpkg**.

Restoring the previous maintenance level

The procedure to remove a maintenance level upgrade is the same as is used for previous releases. If you repackaged a maintenance level by using the **crtmqfp** command, you must remove the packages with the appropriate suffix.

7.1.3 Working with installations

WebSphere MQ provides a number of other commands to assist with the administration of multiple installations. Each installation can be uniquely identified by its location or its name when these commands are used.

Displaying installations

WebSphere MQ tracks the installations on a machine in the registry on Windows and in a file that is called `/etc/opt/mqm/mqinst.ini` on UNIX and Linux. The `dspmqinst` command can be used to report this information for one or all installations.

The following information is reported about each installation:

- ▶ The installation name
- ▶ The installation description
- ▶ The installation identifier
- ▶ The installation path
- ▶ The installation version
- ▶ Whether it is the primary installation
- ▶ The installation state
- ▶ Additional platform-specific information (if applicable), such as the MSI Instance Identifier on Windows

Sample output from this command is shown in Example 7-1.

Example 7-1 Sample output from dspmqinst on Windows

```
InstName: Installation1
InstDesc: This is installation one
Identifier: 1
InstPath: C:\Program Files (x86)\IBM\WebSphere MQ
Version: 7.1.0.0
Primary: No
State: Available
MSIProdCode: {0730749B-080D-4A2E-B63D-85CF09AE0EF0}
MSIMedia: 7.1 Server
MSIInstanceId: 1
```

Naming installations

Every installation of WebSphere MQ must include a unique name, which is used to identify the installation. By default, each installation is named sequentially, starting with `Installation1`; the second installation is `Installation2`, and so on. If WebSphere MQ V7.0.1 is installed, it is always assigned the name `Installation0`.

On Windows, you can specify the name that you want to use if you select to perform a custom installation. On UNIX and Linux, the installers do not provide this ability, so another command is provided on these platforms, called `crtmqinst`. This command is only present if you installed a copy of WebSphere MQ V7.1, or later. The command adds an entry to `mqinst.ini` that associates an installation name with an installation location. If you want to assign a custom name, it must be run before the other copy of WebSphere MQ is installed so the default name is not used.

An example command that specifies the name for the installation that is installed in /opt/mqm-A is shown in the following example:

```
crtmqinst -n InstallationA -p /opt/mqm-A
```

If you run the **dspmqinst** command after the **crtmqinst** command is run, it reports the other installation. The state of the installation indicates that it is not available to be used and its version is unknown until the installation process is completed. Sample output from running the **dspmqinst** command for an installation that is defined by using the **crtmqinst** command is shown in Example 7-2.

Example 7-2 Sample extract of the output from dspmqinst of an installation defined by using crtmqinst

InstName:	InstallationA
InstDesc:	
Identifier:	2
InstPath:	/opt/mqm-A
Version:	??.?.?.?
Primary:	No
State:	Not Available

Setting the description for installations

You can provide a description for each installation that can be reported by **dspmqinst** in the InstDesc field. This field can be used to identify the purpose of the installation. For example, if it is a production installation or an installation that is undergoing quality assurance testing.

On Windows, you can set the description during the installation process if you elect to perform a custom installation. On UNIX and Linux, if you create an installation definition by using the **crtmqinst** command, you can specify its description then. An example of defining a description for an installation by using the **crtmqinst** command is shown in the following example:

```
crtmqinst -n InstallationA -p /opt/mqm-A -d "This is used by application A"
```

On Windows, UNIX, and Linux, you can modify the description for an existing installation by using the **setmqinst** command. To identify the installation, you specify one (or both) of the installation names and its path. An example of changing the description for an installation by using the **setmqinst** command is shown in the following example:

```
setmqinst -n InstallationA -d "This is used by application A"
```

Deleting installation definitions

When WebSphere MQ on UNIX and Linux is uninstalled, the entry for that installation in mqinst.ini remains, and continues to be reported by **dspmqinst**, although it is identified as unavailable. The entry is kept so the information, such as its name and description, is preserved if you are performing a single-stage migration, as required for previous versions. If you then install WebSphere MQ in the same location, the entry is reused.

To delete an entry if it is no longer required, you can use the **dltmqinst** command. This command also can be used to remove an entry that was created by using the **crtmqinst** command if it was defined incorrectly.

To delete an entry, you can specify an installation name, an installation location, or the name and location of an installation. An example of deleting an entry for an installation by using the **dltmqinst** command is shown in the following example:

```
dltmqinst -n InstallationA
```

Displaying version information

To display version information for an installation of WebSphere MQ, you can use the **dspmqver** command. This command was extended in WebSphere MQ V7.1 to report more information, including the installation name, location, the data path, and whether the installation is the primary installation. For more information about the primary installation, see 7.1.4, “Primary installation” on page 81.

Sample output from **dspmqver** is shown in Example 7-3.

Example 7-3 Sample output from dspmqver on Windows

Name:	WebSphere MQ
Version:	7.1.0.0
Level:	p000-L111019
BuildType:	IKAP - (Production)
Platform:	WebSphere MQ for Windows
Mode:	32-bit
O/S:	Windows 7 Enterprise x64 Edition, Build 7601: SP1
InstName:	Installation1
InstDesc:	This is installation one
InstPath:	C:\Program Files (x86)\IBM\WebSphere MQ
DataPath:	C:\Program Files (x86)\IBM\WebSphere MQ
Primary:	No
MaxCmdLevel:	710

The **dspmqver** command also was extended so it can report information about the other installations on the machine. Use the following command to report information about all of the installations:

```
dspmqver -i
```

7.1.4 Primary installation

Previous versions of WebSphere MQ set global environment variables on Windows and install symbolic links in /usr/bin and /usr/lib on UNIX and Linux. When these environment variables or symbolic links are set up, administrators can run the WebSphere MQ control commands without the need to add them to their PATH. Applications can also load the WebSphere MQ libraries automatically. If you installed multiple copies of WebSphere MQ on a machine, this configuration cannot be performed for every installation because the environment variables or symbolic links can locate only one of them. To perform administrative functions or load the required libraries, the environment must be configured for the appropriate installation.

One installation can be identified as the primary installation. When an installation is set to be the primary installation, the global environment setup that was performed by previous versions of WebSphere MQ is run. This setup allows scripts and applications (including third-party products) to continue to function without change.

If WebSphere MQ V7.0.1 is installed, it is always the primary installation. If all installations are at WebSphere MQ V7.1, or later, you can choose which installation is primary, or elect not to have a primary installation. The first installation on Windows is automatically set to be the primary installation. However, this setting can be changed, if required. On UNIX and Linux, there is no primary installation by default, unless WebSphere MQ V7.0.1 is installed. If a primary installation is required, you must manually set such an installation.

Setting the primary installation

To set or unset the primary installation, you use the **setmqinst** command, which provides the installation name or its location.

The following example shows setting an installation to be primary by using its name:

```
setmqinst -n Installation2 -i
```

The following example shows unsetting an installation as the primary installation by using its location:

```
setmqinst -p /opt/mqm -x
```

Important: An existing primary installation must be unset before another installation can be configured as primary.

Restrictions

Some WebSphere MQ functions on Windows require resources to be centrally registered with the operating system. These resources can be registered only once so they are registered by using the primary installation. The following capabilities are available only if a primary installation is set:

- ▶ The WebSphere MQ .NET monitor
- ▶ COM/ActiveX interface classes

For more information about these restrictions, see *Features that can be used only with the primary installation on Windows* in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/zi00730_.htm

7.1.5 Managing the environment

If you did not configure a primary installation, or you want to use a non-primary installation, you must configure your environment accordingly. To simplify this process, WebSphere MQ provides the **setmqenv** and **crtmqenv** commands.

Using setmqenv

The **setmqenv** command can be run to configure the environment for an installation so the control commands can be run by name without the need to qualify them with their complete path. It also sets a number of other environment variables to allow WebSphere MQ resources to be located. This command can be run from any installation. You specify to the command the installation to configure the environment for (based on a queue manager), an installation name, an installation path, or the installation of the command. The command is run slightly differently depending on the platform.

The following example shows running the **setmqenv** command to configure the environment for a named installation on Windows:

```
"C:\Program Files\IBM\WebSphere MQ\bin\setmqenv.cmd" -n Installation1
```

The following example shows running the same command on UNIX and Linux:

```
. /opt/mqm/bin/setmqenv -n Installation1
```

Important: The period and subsequent space at the start of the command on UNIX and Linux are important so the environment variables for the current shell are updated. If this prefix is omitted, the command runs in a separate shell, whereby it might run successfully but it does not update the environment correctly. If you use WebSphere MQ V7.1 with fix pack 1, or later, and the period and space are omitted, the command fails with the following message:

```
AMQ8595: The setmqenv command was not preceded by the source command.
```

On UNIX and Linux, the command does not add the WebSphere MQ library locations to the library path environment variable by default. If you require the library path to be updated, another parameter can be specified to request that the required locations are added to the start or end of the library path.

The following example shows running the **setmqenv** command to configure the environment for the same installation, the **-s** parameter, and request that the WebSphere MQ libraries are added to the start of the library path, the **-k** parameter:

```
. /opt/mqm/bin/setmqenv -s -k
```

Important: On UNIX and Linux, some shells, such as the standard shell, sh, on Solaris, do not accept parameters to the **crtmqenv** and **setmqenv** commands. If you use such a shell, you must run the command from the installation you want to use. You also must manually perform any optional configuration, such as setting the library path, if such configuration is required.

You also can use the **setmqenv** command to unset the environment. This command removes references to WebSphere MQ from the environment variables the command configures, including the library path. The following example shows unsetting the environment on UNIX and Linux:

```
. setmqenv -r
```

For more information about the complete set of parameters that can be specified to this command, see the *setmqenv* topic in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/zr00610_.htm

Using **crtmqenv**

If you do not want (or are unable) to use the **setmqenv** command to configure your environment, you can run the **crtmqenv** command to generate and display the list of environment variable settings. You can use this information to determine the environment setup that is required and apply the changes yourself.

The **crtmqenv** command takes similar parameters to **setmqenv**.

The following example shows running the **crtmqenv** command to configure the environment for a named installation on Windows:

```
"C:\Program Files\IBM\WebSphere MQ\bin\crtmqenv.cmd" -n Installation1
```

The following example shows running the same command on UNIX and Linux:

```
/opt/mqm/bin/crtmqenv -n Installation1
```

Period and space: You do not specify a period and space at the start of the command, as per `setmqenv`, because the `crtmqenv` command is a binary program instead of a shell script and does not update the environment.

For more information about the complete set of parameters that can be specified to this command, see the `crtmqenv` topic in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/zr00620_.htm

7.2 Managing queue managers

This section describes how to manage queue managers on a machine that features multiple installations of WebSphere MQ.

7.2.1 Displaying queue managers

To display all the queue managers on a machine, use the `dspmq` command. This command is used to display the queue managers that were defined and their status; for example, the queue managers that are running. In WebSphere MQ V7.1, the `dspmq` command is extended so that you can view the installation with which each queue manager is associated. This information is not shown by default. To view the installations with which the queue managers are associated, use the `-o` parameter with the value `installation`, or `a11`. The following example shows running the `dspmq` command to display this information:

```
dspmq -o installation
```

Sample output from this command is shown in Example 7-4.

Example 7-4 Sample output from dspmq -o installation

```
QMNAME(QM1) INSTNAME(Installation0) INSTPATH(/opt/mqm) INSTVER(7.0.1.6)
QMNAME(QM2) INSTNAME(Installation0) INSTPATH(/opt/mqm) INSTVER(7.0.1.6)
QMNAME(QM3) INSTNAME(Installation1) INSTPATH(/opt/mqm71) INSTVER(7.1.0.0)
```

If WebSphere MQ V7.0.1 is installed and you run the `dspmq` command from that installation, it can display all the queue managers on the machine. However, it cannot display the installations with which they are associated. It is also unable to display the status of queue managers that are associated with a version 7.1 installation, or later. If it cannot display the status of a queue manager, it reports that the status is unavailable. Example 7-5 shows output from the WebSphere MQ V7.0.1 version of the `dspmq` command when a queue manager, QM3, is created by using WebSphere MQ V7.1, or later.

Example 7-5 Sample output from dspmq when the status of a queue manager cannot be reported

```
QMNAME(QM1) STATUS(Running)
QMNAME(QM2) STATUS(Ended normally)
QMNAME(QM3) STATUS(Status not available)
```

7.2.2 Administering queue managers

To use the control commands to administer a queue manager, you must configure your environment for the installation with which it is associated. If you do not want to use a primary installation, you can configure the environment manually or use the `setmqenv` command. For more information about configuring your environment, see 7.1.5, “Managing the environment” on page 82.

When you create a queue manager, it is associated with the same installation as the `crtmqm` command. You must use the control commands from the same installation to then administer it, unless you associate it with a different installation.

If you attempt to administer a queue manager by using control commands from a different installation, they fail and issue a message as shown in the following example:

AMQ5691: Queue manager 'QM1' is associated with a different installation ('Installation0').

7.2.3 Migrating a queue manager to another installation

You can change the installation that a queue manager is associated with by using the `setmqm` command. You can use this command to perform the following tasks:

- ▶ Move a queue manager to another installation at a later version of WebSphere MQ.
- ▶ Move a queue manager to another installation at the same version. The maintenance level can be the same, higher, or lower than the level of the current installation.

Important: After you start a queue manager by using one installation, you cannot use the `setmqm` command to associate it with an installation at an earlier version. For example, you cannot associate a queue manager with a WebSphere MQ V7.1 installation if you started it by using a WebSphere MQ V7.5 installation. The `setmqm` command might succeed, but the queue manager fails to start.

To change the installation that a queue manager is associated with, you must stop the queue manager and then run the `setmqm` command from the target installation. If you attempt to perform the association by using the `setmqm` command from the queue manager's current installation or another installation on the machine, the command fails.

setmqm command: The `setmqm` command does not exist in WebSphere MQ V7.0.1. If you use this command to change the association of a WebSphere MQ V7.0.1 queue manager to a WebSphere MQ V7.1, or later, installation, you cannot reverse the change. Therefore, this migration consideration is important.

The `setmqm` command requires the following parameters:

- ▶ The name of the queue manager.
- ▶ The name of the installation.

Although the name of the installation can be inferred from the `setmqm` command, you must specify it for confirmation.

The following example shows running the `setmqm` command to change queue manager QM1 to be associated with installation Installation1:

```
setmqm -m QM1 -n Installation1
```

Important: If a queue manager includes no associated installation and WebSphere MQ V7.0.1 is not installed, the `strmqm` command automatically associates the queue manager with the installation from which it was run. You do not need to run the `setmqm` command beforehand.

7.2.4 Using WebSphere MQ Explorer

WebSphere MQ Explorer is an Eclipse UI for administering queue managers that is available on Windows, Linux x86, and Linux x86-64. Each installation of WebSphere MQ includes its own copy of WebSphere MQ Explorer. When you start WebSphere MQ Explorer for an installation of WebSphere MQ V7.1, or later, the installation to which it relates is shown in the title bar, as shown in Figure 7-1.

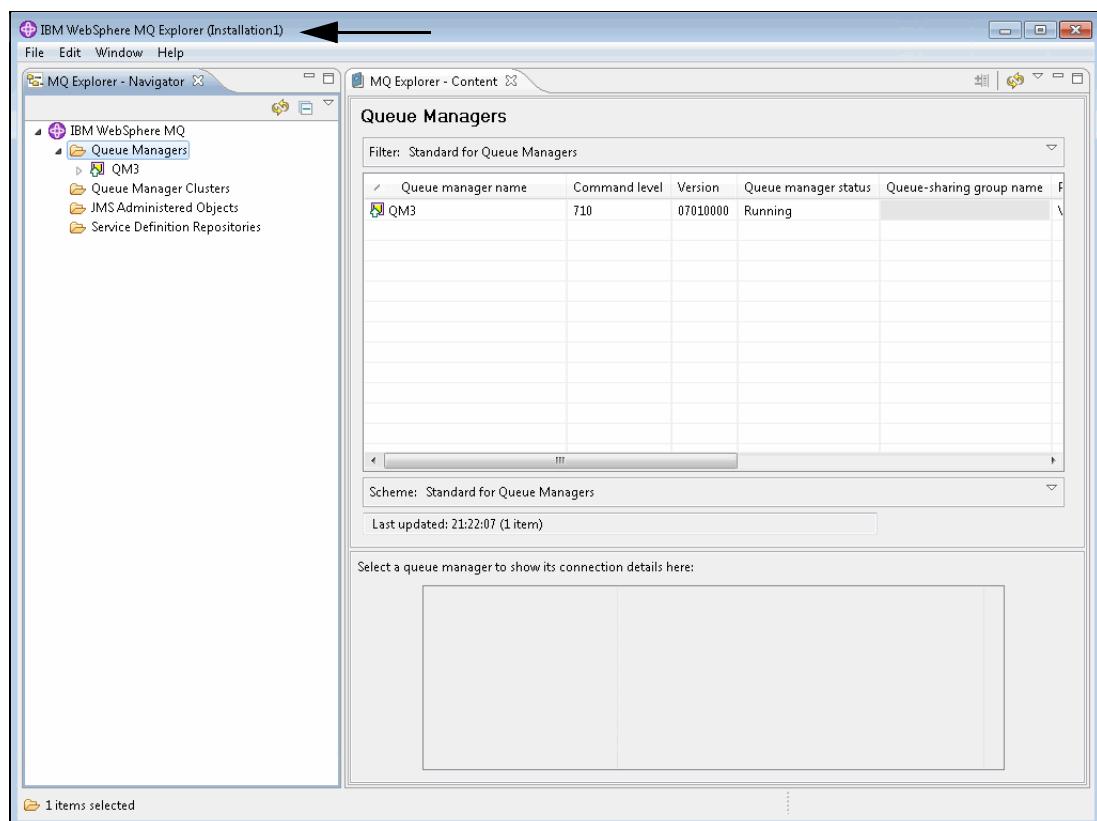


Figure 7-1 WebSphere MQ Explorer now identifies its installation in the title bar

When WebSphere MQ Explorer is used, the only local queue managers that are shown are the queue managers that are associated with the same installation. To administer other local queue managers, you must run WebSphere MQ Explorer from the installation with which they are associated, or connect to them as remote queue managers. A remote connection to queue manager QM1 that is associated with a different installation is shown in Figure 7-2.

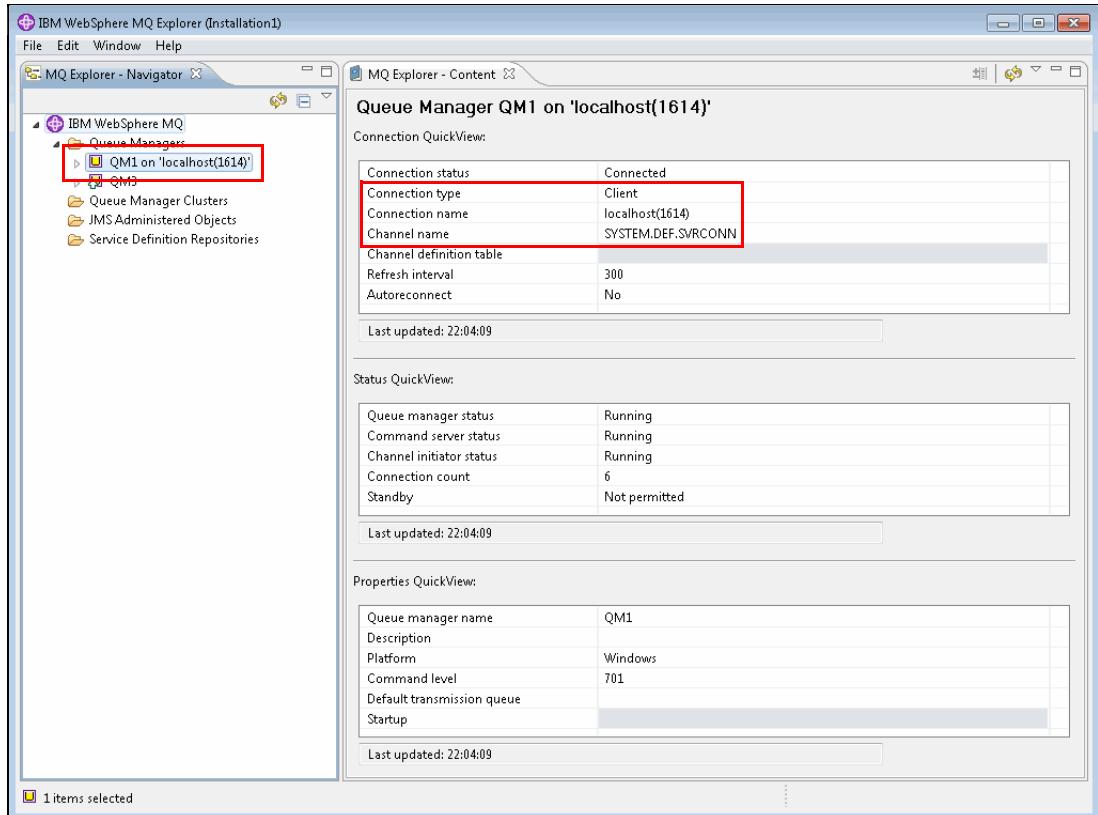


Figure 7-2 A remote connection to a queue manager associated with a different installation

7.3 Managing applications

This section describes how to manage WebSphere MQ applications when multiple installations are present, or you have a single installation in a non-default location.

7.3.1 Environment considerations

When a WebSphere MQ application is run, it must load one or more, WebSphere MQ libraries that provide the Message Queuing Interface (MQI). The libraries that it must load are dependent on how it was linked when it was built. For example, if the application connects to a queue manager by using local bindings, it must load the `mqm` library. If it connects to a queue manager by using a client connection, it must load the `mqic` library. The capability to install WebSphere MQ V7.1, or later, in any location on UNIX and Linux affects how these libraries are located at run time. How these libraries are loaded depends on the application environment.

How libraries are loaded

To understand the implications of installing WebSphere MQ multiple times on a machine, or having a single installation in a non-default location, it is necessary to explain how libraries are loaded. When an application is run, the operating system must first load any external libraries that it uses so that external function calls (such as the MQI) are available. How these libraries are loaded is dependent on the operating system.

On Windows, libraries are searched for in the following order:

1. The directory where the application resides.
2. The current working directory.
3. The global PATH environment variable.
4. The user's PATH environment variable.

On UNIX and Linux, libraries are searched for in the following order:

1. The library path environment variable.

This variable is called `LD_LIBRARY_PATH`. On AIX, the environment variable `LIBPATH` is also used. On HP-UX, the environment variable that is called `SHLIB_PATH` is also used.

2. An embedded runtime search path (`RPATH`), if present in the application.
3. The default library path.

This path can be configured by the system administrator, but it usually contains the `/lib` and `/usr/lib` directories, or their 64-bit equivalents.

Important: On UNIX and Linux, the library path environment variable is not used for setuid or setgid applications. A setuid or setgid application is one that was configured by using the `chmod` command to always run as the user that owns it, the group that owns it, or both. These applications must locate the WebSphere MQ libraries by using an embedded `RPATH`, or use the default library path to use the primary installation. This restriction, which is enforced by the operating system, is required so that libraries that are loaded by applications that run with alternative privileges cannot be modified by another user.

Each operating system provides tools to determine what an application is linked with and, on UNIX and Linux, the built-in runtime search path (`RPATH`) of the application.

Library loading considerations for AIX

To determine the libraries that an application requires on AIX, you can use the **ldd** command. This command lists the libraries and attempts to locate them by using the current environment and options, such as the runtime search path that is encoded in the application. An example invocation of the **ldd** command and its output is shown in Example 7-6.

Example 7-6 Using the ldd command to determine an application's dependencies on AIX

```
1dd amqsput

amqsput needs:
    /usr/lib/libc_r.a(shr_64.o)
    /usr/lib/libpthread.a(shr_xpg5_64.o)
    /usr/mqm/lib64/libmqm_r.a(libmqm_r.o)
    /unix
    /usr/lib/libcrypt.a(shr_64.o)
    /usr/lib/threads/libc.a(shr_64.o)
    /usr/mqm/lib64/libmqe_r.a(libmqe_r.o)
    /usr/lib/libdl.a(shr_64.o)
    /usr/lib/libiconv.a(shr4_64.o)
```

To determine the built-in RPATH on AIX, you can use the **dump** command to list the Import File Strings. An example invocation of the **dump** command that reports the header of the loader section for a 64-bit application is shown in Example 7-7.

Example 7-7 Using the dump command to determine an application's RPATH on AIX

```
dump -X 64 -H amqsput

amqsput:

***Loader Section***
      Loader Header Information
VERSION#      #SYMtableENT      #RELOCent      LENidSTR
0x00000001    0x00000013      0x00000021      0x00000194

#IMPfi1ID     OFFidSTR        LENstrTBL      OFFstrTBL
0x00000004    0x00000410      0x000000ca      0x000005a4

***Import File Strings***
INDEX PATH                      BASE          MEMBER
0/usr/mqm/lib64:/usr/vac/lib:/usr/lib/threads:/usr/vacpp/lib:/usr/lib:/lib:/
1                         libc_r.a        shr_64.o
2                         libpthread.a   shr_xpg5_64.o
3                         libmqm_r.a    libmqm_r.o
```

Library loading considerations for HP-UX

To determine the libraries that an application requires on HP-UX, you can use the **1dd** command. This command lists the libraries and attempts to locate them by using the current environment and options, such as the runtime search path that is encoded in the application. An example invocation of the **1dd** command and its output is shown in Example 7-8.

Example 7-8 Using the ldd command to determine an application's dependencies on HP-UX

```
1dd amqspput

amqspput:
  libmqm_r.so => /opt/mqm/lib64/libmqm_r.so
  libcl.so.1 => /usr/lib/hpux64/libcl.so.1
  libpthread.so.1 => /usr/lib/hpux64/libpthread.so.1
  libc.so.1 => /usr/lib/hpux64/libc.so.1
  libdl.so.1 => /usr/lib/hpux64/libdl.so.1
  libmqe_r.so => /opt/mqm/lib64/libmqe_r.so
  libI077.so.1 => /usr/lib/hpux64/libI077.so.1
  libunwind.so.1 => /usr/lib/hpux64/libunwind.so.1
  libdl.so.1 => /usr/lib/hpux64/libdl.so.1
  libm.so.1 => /usr/lib/hpux64/libm.so.1
  libdl.so.1 => /usr/lib/hpux64/libdl.so.1
  libuca.so.1 => /usr/lib/hpux64/libuca.so.1
```

To determine the built-in RPATH on HP-UX, you can use the **chatr** command to show the shared library dynamic path search. An example invocation of the **chatr** command is shown in Example 7-9 on page 91. This example shows that for this application, the following search order for libraries is used before the default library path is resorted:

1. LD_LIBRARY_PATH environment variable
2. SHLIB_PATH environment variable
3. Embedded RPATH

However, the two environment variables, although first and second in the search list, are not used to locate libraries because they were disabled. These entries might be disabled because the application was linked by using the `+noenvvar` option, which was included in the instructions on how to build applications for previous versions of WebSphere MQ.

*Example 7-9 Using the **chatr** command to determine an application's RPATH on HP-UX*

```
chatr amqsput

amqsput:
    64-bit ELF executable
    shared library dynamic path search:
        LD_LIBRARY_PATH      disabled first
        SHLIB_PATH           disabled second
        embedded path        enabled third /opt/mqm/lib64
    shared library list:
        libmqm_r.so
        libcl.so.1
        libpthread.so.1
        libc.so.1
        libdl.so.1
    shared library binding:
        immediate nonfatal
        global hash table disabled
        global hash table size 1103
    shared library mapped private disabled
    runtime checks disabled
    shared library segment merging disabled
    shared vtable support disabled
    explicit unloading disabled
    linkage table protection disabled
    segments:
        index type      address      flags size
            8 text 4000000000000000 z---c- D (default)
            9 data 6000000000000000 ---m-- D (default)
    executable from stack: D (default)
    kernel assisted branch prediction enabled
    lazy swap allocation for dynamic segments disabled
    nulptr dereferences trap enabled
    address space model: default
    caliper dynamic instrumentation disabled
```

You can also use the **chatr** command to update the shared library dynamic path search order without having to recompile or re-link an application. To change these attributes, you can add the **+b** and **+s** parameters to the **chatr** command to enable or disable the embedded RPATH and environment variables. The use of the **chatr** command to enable the use of the environment variables for an application is shown in the following example:

chatr +s enable <application name>

For more information about the **chatr** command, see its manual page, which can be accessed by using the **man** command, for example:

man chatr

Library loading considerations for Linux

To determine the libraries that an application requires on Linux, you can use the **ldd** command. This command lists the libraries and attempts to locate them by using the current environment and options, such as the runtime search path that is encoded in the application. An example invocation of the **l dd** command and its output is shown in Example 7-10.

Example 7-10 Using the ldd command to determine an application's dependencies on Linux

```
l dd amqspput
```

```
linux-vdso.so.1 => (0x00007fff1fffff000)
libmqm_r.so => /opt/mqm/lib64/libmqm_r.so (0x00007fce8d4be000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x0000003111600000)
libdl.so.2 => /lib64/libdl.so.2 (0x0000003110e00000)
libc.so.6 => /lib64/libc.so.6 (0x0000003111200000)
libmqe_r.so => /opt/mqm/lib64/libmqe_r.so (0x00007fce8cb09000)
/lib64/ld-linux-x86-64.so.2 (0x0000003110a00000)
libm.so.6 => /lib64/libm.so.6 (0x0000003111a00000)
librt.so.1 => /lib64/librt.so.1 (0x0000003112200000)
```

To determine the built-in RPATH on Linux, you can use the **readelf** command to list the Dynamic section that is encoded in the application. An example invocation of the **readelf** command is shown in Example 7-11.

Example 7-11 Using the readelf command to determine an application's RPATH on Linux

```
readelf -d amqspput
```

Dynamic section at offset 0x16d0 contains 25 entries:

Tag	Type	Name/Value
0x0000000000000001	(NEEDED)	Shared library: [libmqm_r.so]
0x0000000000000001	(NEEDED)	Shared library: [libpthread.so.0]
0x0000000000000001	(NEEDED)	Shared library: [libdl.so.2]
0x0000000000000001	(NEEDED)	Shared library: [libc.so.6]
0x000000000000000f	(RPATH)	Library rpath: [/opt/mqm/lib64]
0x0000000000000001d	(RUNPATH)	Library runpath: [/opt/mqm/lib64]
0x000000000000000c	(INIT)	0x400848
0x000000000000000d	(FINI)	0x401364
0x0000000000000004	(HASH)	0x400290
0x0000000000000005	(STRTAB)	0x4004b8
0x0000000000000006	(SYMTAB)	0x400320
0x000000000000000a	(STRSZ)	484 (bytes)
0x000000000000000b	(SYMENT)	24 (bytes)
0x0000000000000015	(DEBUG)	0x0
0x0000000000000003	(PLTGOT)	0x5018c0
0x0000000000000002	(PLTRELSZ)	312 (bytes)
0x0000000000000014	(PLTREL)	RELA
0x0000000000000017	(JMPREL)	0x400710
0x0000000000000007	(RELA)	0x4006e0
0x0000000000000008	(RELASZ)	48 (bytes)
0x0000000000000009	(RELAENT)	24 (bytes)
0x000000006ffffffe	(VERNEED)	0x4006c0
0x000000006fffffff	(VERNEEDNUM)	1
0x000000006ffffff0	(VERSYM)	0x40069c
0x0000000000000000	(NULL)	0x0

Applications that are compiled by using some versions of the `gcc` command (for example, version 3.2.x), can include an embedded runtime search path (RPATH) that cannot be overridden by using the `LD_LIBRARY_PATH` environment variable. If the RPATH is used to locate WebSphere MQ libraries, the environment cannot be used to override the search path for these applications. If WebSphere MQ is installed in a different location, these applications might need to be rebuilt to allow the libraries to be found. The runtime search path cannot be overridden if the RPATH symbol is present and the RUNPATH symbol is not present in the output of the `readelf` command.

Library loading considerations for Solaris

To determine the libraries that an application requires on Solaris, you can use the `ldd` command. This command lists the libraries and attempts to locate them by using the current environment and options, such as the runtime search path that is encoded in the application. An example invocation of the `ldd` command and its output is shown in Example 7-12.

Example 7-12 Using the ldd command to determine an application's dependencies on Solaris

```
ldd amqspput
```

```
libmqm.so =>      /opt/mqm/lib64/libmqm.so
libthread.so.1 =>      /usr/lib/64/libthread.so.1
libsocket.so.1 =>      /usr/lib/64/libsocket.so.1
libc.so.1 =>      /usr/lib/64/libc.so.1
libnsl.so.1 =>      /usr/lib/64/libnsl.so.1
libdl.so.1 =>      /usr/lib/64/libdl.so.1
libmqe.so =>      /opt/mqm/lib64/libmqe.so
libmp.so.2 =>      /lib/64/libmp.so.2
libmd.so.1 =>      /lib/64/libmd.so.1
libscf.so.1 =>      /lib/64/libscf.so.1
libm.so.2 =>      /usr/lib/64/libm.so.2
librt.so.1 =>      /usr/lib/64/librt.so.1
libdoor.so.1 =>      /lib/64/libdoor.so.1
libutil.so.1 =>      /lib/64/libutil.so.1
libgen.so.1 =>      /lib/64/libgen.so.1
libaio.so.1 =>      /lib/64/libaio.so.1
/platform/SUNW,T5240/lib/sparcv9/libc_psr.so.1
/platform/SUNW,T5240/lib/sparcv9/libmd_psr.so.1
```

To determine the built-in RPATH on Solaris, you can use the **elfdump** command to list the Dynamic section that is encoded in the application. An example invocation of the **elfdump** command is shown in Example 7-13.

Example 7-13 Using the elfdump command to determine an application's RPATH on Solaris

```
elfdump -d amqsput

Dynamic Section: .dynamic
index  tag          value
[0]   NEEDED      0x16a      libmqm.so
[1]   NEEDED      0x174      libthread.so.1
[2]   NEEDED      0x183      libsocket.so.1
[3]   NEEDED      0x147      libc.so.1
[4]   NEEDED      0x192      libnsl.so.1
[5]   NEEDED      0x19e      libdl.so.1
[6]   INIT        0x100001138
[7]   FINI        0x100001148
[8]   RUNPATH    0xa9
/opt/mqm/lib64:/usr/lib/64:/usr/ccs/lib/64:/usr/ucb/lib/64:/opt/SUNWappc:/
[9]   RPATH        0xa9
/opt/mqm/lib64:/usr/lib/64:/usr/ccs/lib/64:/usr/ucb/lib/64:/opt/SUNWappc:/
[10]  HASH         0x100000178
[11]  STRTAB       0x100000668
[12]  STRSZ         0x2f2
[13]  SYMTAB       0x1000002c0
[14]  SYMENT       0x18
[15]  CHECKSUM     0xab9
[16]  VERNEED      0x100000960
[17]  VERNEEDNUM   0x1
[18]  PLTRELSZ     0x168
[19]  PLTREL       0x7
[20]  JMPREL       0x1000009c0
[21]  RELA         0x100000990
[22]  RELASZ       0x198
[23]  RELAENT      0x18
[24]  REGISTER     0x20
[25]  REGISTER     0x7
[26]  DEBUG         0
[27]  FEATURE_1    0x1          [ PARINIT ]
[28]  FLAGS         0             0
[29]  FLAGS_1       0             0
[30]  PLTGOT       0x100101800
```

The instructions for how to build applications on Solaris for previous versions of WebSphere MQ include linking them with the mqmcs and mqmzse libraries. These libraries are now loaded automatically by WebSphere MQ and applications no longer should be linked with them. You must update your applications if you want to use them with multiple installations of WebSphere MQ, or a single installation in a non-default location. This requirement is necessary because loading an incorrect version of these libraries can cause conflicts or runtime errors. You can update applications by recompiling and linking without the `-lqmcs` or `-lqmzse` link options. To modify an application that you obtained from a third-party provider, you must contact that provider to request an update for this software.

Loading WebSphere MQ V7.1, or later, libraries

When WebSphere MQ V7.1, or later, is used, you must decide how your applications are going to load the WebSphere MQ libraries on which they depend. The following questions must be considered:

- ▶ What is the application environment?
- ▶ Is it possible to change the application?
- ▶ Are you using, or going to use, multiple installations of WebSphere MQ, or a single installation in a non-default location?
- ▶ Are you using a primary installation?

If you are going to use a single, primary installation in the default location, existing applications continue to function without change. Otherwise, it might be necessary to make changes to ensure that the correct libraries are loaded.

The following options are available for loading the correct libraries:

- ▶ Changing the embedded runtime search path (RPATH) on UNIX and Linux.

This option explicitly defines the location where the WebSphere MQ dependencies reside. However, you must recompile and re-link the applications. You also must change them whenever you install WebSphere MQ in a different location.
- ▶ Set the library path environment variable on UNIX and Linux, or the PATH environment variable on Windows.

This task can be done manually, or by using the **setmqenv** command. Although you do not have to recompile your applications, changing the environment might have other implications. The library path environment variable is not honored for setuid or setgid applications. This method might not work for all applications. For more information, see “How libraries are loaded” on page 88.
- ▶ Set a primary installation.

You cannot have a WebSphere MQ V7.1, or later, primary installation while WebSphere MQ V7.0.1 is installed.

7.3.2 Connecting to queue managers

When an application connects to a queue manager on the same machine, it must load the WebSphere MQ libraries from the same installation the queue manager is associated with to ensure compatibility. If there are multiple installations of WebSphere MQ, the libraries an application needs to load can change as queue managers are switched between them. If an application connects to multiple queue managers, it also can mean that libraries from more than one installation need to be loaded. To simplify this process, the WebSphere MQ libraries, as of version 7.1, are able to automatically load the required libraries when they are connecting to a queue manager that is associated with a different installation.

The ability to load the required libraries automatically means that if an application is configured to load its dependencies from a WebSphere MQ V7.1, or later, installation, it can connect to any local queue manager. If an application is configured to load its dependencies from a WebSphere MQ V7.0.1 installation, it cannot connect to queue managers at version 7.1, or later, on the same machine unless it uses a client connection. This restriction exists because the version 7.0.1 libraries are incapable of loading their version 7.1, or later, counterparts.

Client versus local bindings connections

Before WebSphere MQ V7.1, you must link an application with the `mqm` library to connect to a queue manager on the local machine by using inter-process communication, or the `mqic` library to connect over the network. In WebSphere MQ V7.1, the `mqm` library is extended so that if a local connection cannot be established because the required installation is not present, it automatically attempts a client connection instead. In addition, a new `MQCNO_CLIENT_BINDING` option was added to `MQCONN`. This option allows an application that is linked with the `mqm` library to programmatically request a client connection instead. This option is not supported on z/OS or some types of applications, such as Java or .NET.

7.3.3 Using .NET applications

To use a .NET application when there is no primary installation, or to use the assemblies that are provided with a different installation, you need to update the application configuration file or the `DEVPATH` environment variable. Applications can use the primary installation without change because the .NET assemblies and policy files for this installation are registered to the global assembly cache (GAC).

For more information about how to update the application configuration file or use the `DEVPATH` environment variable, see *Connecting .NET applications in a multiple installation environment* in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/za00120_.htm

7.3.4 Exit and installable service considerations

WebSphere MQ supports user-written exits and installable services. These exits and services can make MQI calls to enrich their function. For example, an API exit, when called, might put a message to an audit queue. To work with WebSphere MQ V7.0.1, or earlier, an exit or installable service must be linked with a WebSphere MQ library if it makes one or more MQI calls. If multiple copies of WebSphere MQ are installed on a machine, or WebSphere MQ is installed in a non-default location, exits, and installable services must not be explicitly linked with any WebSphere MQ libraries. This configuration is necessary to avoid conflicts that are caused by loading multiple versions of the same library.

Instead of requiring that an exit or installable service is linked with one or more WebSphere MQ libraries, a new MQI structure that is called MQIEP was introduced in WebSphere MQ V7.1. This structure provides entry points for the MQI functions that should be called instead.

Important: If you have an existing exit or installable service that you want to use with WebSphere MQ V7.1, or later, you must update it to use the MQIEP if you have multiple installations of WebSphere MQ on the same machine, or you have installed WebSphere MQ in a non-default location. The MQIEP is not supported in WebSphere MQ V7.0.1, so the original exit must be used for that installation.

The MQIEP structure is defined in a new header file called `cmqec.h`.

To locate the MQIEP structure at run time, use the following interfaces:

- ▶ API exits: The Hconfig parameter in the MQAXP structure.
- ▶ Channel exits: The pEntryPoints parameter in the MQCXP structure.
- ▶ Cluster workload exits: The pEntryPoints parameter in the MQWXP structure.
- ▶ Data conversion exits: The pEntryPoints parameter in the MQDXP structure.
- ▶ Installable service: The Hconfig parameter.
- ▶ Pre-connect exits: The pEntryPoint parameter in the MQNXP structure.
- ▶ Publish exits: The pEntryPoint parameter in the MQPSXP structure.

For compatibility, before the MQIEP is accessed, exits should verify its availability by using the criteria that is documented in *Writing and compiling exits and installable services* in the WebSphere MQ Information Center, which is available at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/zd00140_.htm

The following example shows the use the MQIEP to call MQPUT in a channel exit, which taken from the Information Center:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,  
                                              Hobj,  
                                              &md,  
                                              &pmo,  
                                              messlen,  
                                              buffer,  
                                              &CompCode,  
                                              &Reason);
```

To allow different exits for queue managers or applications by using different versions of WebSphere MQ, exits can be loaded from an installation-specific directory under the WebSphere MQ data path. For example, if Installation1 is used, an exit is looked for under MQ_DATA_PATH/exits/Installation1 before MQ_DATA_PATH/exits. This capability allows queue managers that are migrated from one installation to another to use different versions.

Important: The MQIEP is not supported on z/OS or IBM i.

7.3.5 Other restrictions

The following restrictions apply to applications that use multiple installations of WebSphere MQ:

- ▶ Message handles that use the special value of MQHC_UNASSOCIATED_HCONN are restricted to the use the first installation for which libraries are loaded in a process.
- ▶ Data conversion exits that are generated by using the `crtmqcvx` command must be regenerated by using WebSphere MQ V7.1, or later.
- ▶ There are a number of restrictions for fast path applications.

For more information about these restrictions, see *Restrictions for applications using multiple installations* in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/z100320_.htm



Enhanced security

This chapter describes the features that were introduced in WebSphere MQ V7.1 that can be used to enhance the security configuration of a complex WebSphere MQ environment. The topics that are covered include enhancements to cryptographic functions that are used for Secure Socket Layer (SSL) and Transport Layer Security (TLS) communication, and simplified queue manager configuration options to control access to WebSphere MQ resources.

This chapter contains the following sections:

- ▶ Controlling access on channels
- ▶ Remote queue access control
- ▶ Enhanced cryptographic support with IBM Global Security Kit V8
- ▶ Cryptographic standards compliance

8.1 Controlling access on channels

The facilities that are described here provide granular control over remote access to the queue manager. This control takes the form of allowing or blocking the connection. Where a connection is successful, these facilities can directly influence the subsequent access the user on the end of the connection has to the queue manager resources.

The features that are described in this section are applicable to all platforms that are supported by WebSphere MQ V7.1.

8.1.1 The role of enhanced channel access control in WebSphere MQ

Controlling access to a queue manager can be split into two parts. Initially, the actual connection to the queue manager and then the access to the resources that are on that queue manager (its queues, topics, and so on).

Within WebSphere MQ, there are a number of measures that can be taken to apply security to the environment. These measures range from identity verification through to auditing, to confidentiality and integrity of data. The implications of the security configuration must be considered carefully to ensure that the correct users can gain access. Of those users, each one can perform only their intended functions. Before a user can perform any functions at all on the queue manager, the connection request must be allowed, and it is this task that channel access control deals with. Channel access control is only one part of a secure environment and it has implications on subsequent security configuration options that are available within WebSphere MQ.

Channel access control performs two vital roles: outright blocking of access and explicit mapping of credentials to a specific user identifier. These functions always were possible by using a security exit. However, channel access control facilitates the same functionality without the need for any user written code. The configuration is simplified by the definition of rules that are applied to incoming connection requests. These rules are channel authentication records and are referred to as CHLAUTH records when WebSphere MQ script commands (MQSC) are used.

Channel authentication record behavior is defined by its type. The ultimate decision to block or allow a connection can be the result of applying one or more rules in sequence.

Figure 8-1 shows the points at which an incoming connection can be affected by channel authentication records that are defined on the queue manager, if the firewall rules allow the connection through. If the incoming connection is allowed through the WebSphere MQ listener blocking and the WebSphere MQ channel blocking or mapping, the resulting connection credentials are subject to checking against access control lists (ACLs) for the WebSphere MQ resources that are being used.

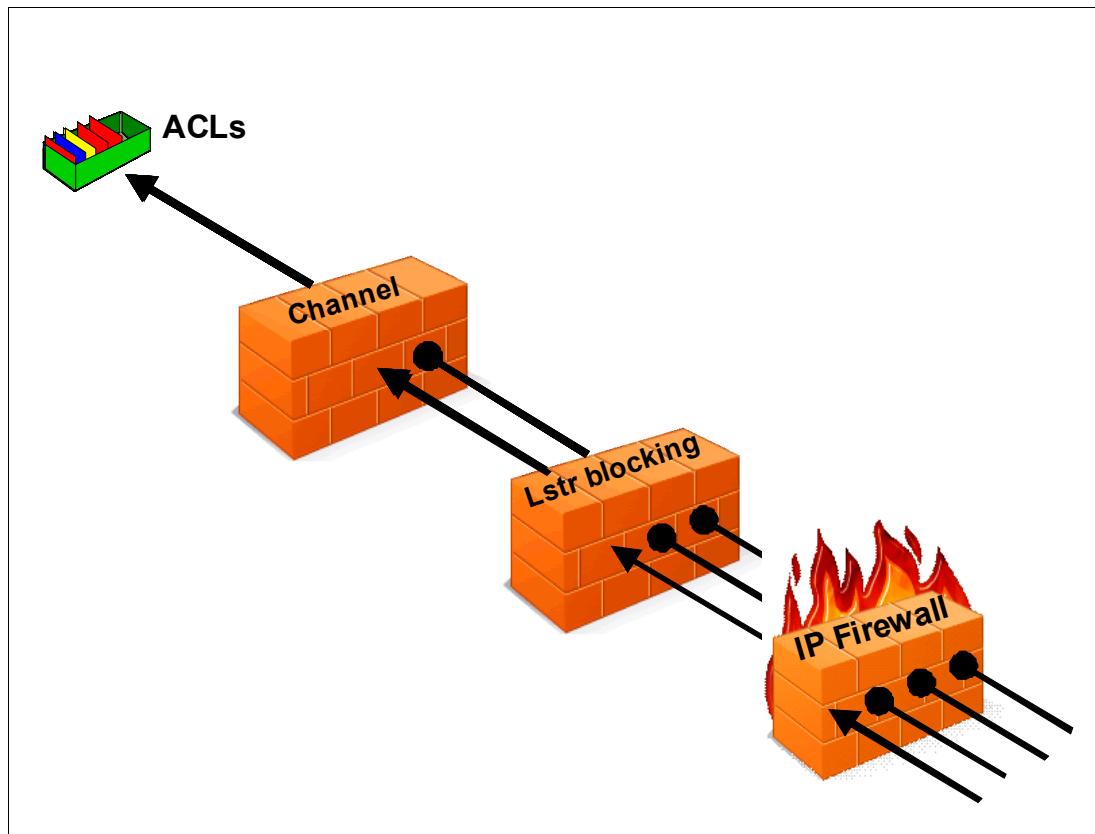


Figure 8-1 Channel authentication records applied on channels and the listener

For more information about the configuration options regarding channel authentication records and the various types that are available, see 8.1.5, “Configuration of channel authentication records” on page 105.

Channel authentication records are defined by the channel name to which they apply. The first point of matching against a rule is a check whether the rule applies to the channel name. The channel name for a rule can be explicit, matching a single channel, partially wildcarded, matching a number of channels, or fully wildcarded, matching all channels.

Channel authentication records are applied to the receiving end of a communication request only.

8.1.2 Blocking based on IP address (BLOCKADDR)

A firewall takes the place of blocking incoming and outgoing connections to a system based on predefined rules. The remote IP address of an incoming connection can be used as the basis for these rules, allowing known addresses through the firewall and blocking connections from any unknown. However, a firewall is a piece of hardware or software in its own right, and often the process of updating and implementing new rules in a firewall needs scheduling and planning.

WebSphere MQ provides a channel authentication record type that can be used as a temporary measure to implement the firewall rules until the firewall can be updated. Blocking based on IP address The MQSC keyword BLOCKADDR is used to define a rule to block that based on IP addresses. The use of one or more BLOCKADDR rules alleviates the risk of potential attacks while proper firewall rules are planned and implemented. The rule can be defined quickly and is active immediately. It is also easily removed from the WebSphere MQ configuration after the rule is implemented in the firewall.

On Windows, UNIX, and Linux, one of the key features of the use of BLOCKADDR is that the queue manager does not necessarily need to be running for the rules to take effect. After the WebSphere MQ listener is running, the queue manager can be brought up or down without the need to restart the listener. The independence of the listener process from the queue manager requires that the listener is not automatically started and stopped in line with the queue manager. The incoming connection is blocked by the listener, and so if the queue manager is down and the listener is still running, rogue connections are still blocked. This feature has particular use a potential denial of service attack occurs.

A repeated attempt to connect might result in high central processing unit (CPU) usage, which results in a failure of the WebSphere MQ listener to process valid connection attempts. Dealing with this attack is usually the role of the firewall. However, until the firewall rules are updated, WebSphere MQ minimizes the effects of a denial of service attack. When a blocked IP address is detected, the socket that the incoming connection comes in on is held open for a short time, which stalls the repetitiveness of the connection attempt. Therefore, the listener can continue to process valid connection requests.

Each BLOCKADDR rule can block one or more IP addresses by stating a list of addresses or a range of addresses, or by using wildcards to match all of the addresses on a subnet.

For more information about how to configure a rule of this type, see “Using MQSC” on page 107.

8.1.3 Message channel agent user ID mapping

When a WebSphere MQ channel starts, the effective user ID that is used to perform tasks is known as the message channel agent user (MCAUSER). This section describes the impact of channel authentication records on the MCAUSER.

Why user ID mapping is useful

Access to WebSphere MQ resources and the ability to perform actions against those resources is closely tied to the user ID that is being used. Channel authentication records relate specifically to users that are coming in on a remote connection.

Without channel authentication records, the option of using a security exit remains to manage user IDs that are sent from a remote system. The use of a security exit requires user-written code. In terms of time, a security exit is potentially an expensive approach. Without careful consideration, configuration remote connections, after they are connected to WebSphere MQ, are likely to encounter MQRC_NOTAUTHORIZED (return code 2035) errors. This error often is because of the remote user ID either not being defined on the local system, or its access to resources is incorrectly configured. If there is a single remote user, this configuration error is a relatively small administration task to correct. However, when the number of remote users (each with unique user IDs) is scaled up to hundreds or thousands, this administrative task quickly becomes unmanageable.

Channel authentication records allow the task of mapping user IDs to be simplified without the need for user-written code. The credentials of an incoming connection can be examined and mapped to a local user ID without the need to create and administer a user ID of the same name as the remote user ID. Minimizing the user ID administration has potential savings across an enterprise and between businesses that communicate with one another over WebSphere MQ.

For more information about the use of mapping credentials from remote entities to local user IDs without the need to define incoming remote user IDs on the local system, see Chapter 20, “Channel authentication records: Controlling remote user activity” on page 277.

There are four types of channel authentication records that can be used, depending on the credentials to be mapped. For more information about and examples of how to configure rules of these types, see “Using MQSC” on page 107.

Mapping that is based on IP address

One of the primary identifiers of the origin of an incoming connection is its IP address, although an IP address alone does not necessarily identify a unique user ID. The mapping that is based on IP address (ADDRESSMAP) channel authentication record type allows mapping of an IP address to a specific user ID. The parameter is not limited to a single IP address and it can range from a single wildcard that represents all addresses to a specific range of addresses to one specific single address.

Mapping that is based on SSL/TLS distinguished name

The use of SSL/TLS with WebSphere MQ covers a number of security issues; for example, digital certificates that are used for authentication at channel start and encryption of data that is exchanged between two trusted parties. Where security is a priority, the use of SSL/TLS is common with internal business communications and inter-company communications.

For more information about the concepts of the SSL/TLS protocols, see the topic *Cryptographic security protocols: SSL and TLS*, and its subtopics in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy10630_.htm

One of the identifying aspects of a digital certificate is its distinguished name (DN). A DN is made up of several attributes and the combination of those attributes identifies the certificate's owner uniquely.

For more information about the DN attributes that are supported by WebSphere MQ V7.1 and above, see the topic *Distinguished Names* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy10570_.htm

The mapping that is based on SSL/TLS distinguished name (SSLPEERMAP) channel authentication record allows mapping of a DN that is received on the incoming connection request to a specific user ID. The string that is used to match against when you are deciding whether to map need not to be represented as a single, specific DN. It can be a subset of DN attributes such that the rule matches against anything that contains those attributes (even if the DN that is received contains other attributes). Equally, it can be a string that contains wildcarded DN attributes, such that the rule matches against anything that contains those attributes and includes any value for those attributes.

For more information about and some examples of how the pattern matching on DN attributes works, see 8.1.7, “SSLPEER pattern matching” on page 118.

This feature extends the ability that is already available on channel definitions to accept only SSL/TLS connections from certificates that match a particular DN string by mapping the DN string to a user ID. For more information about the SSLPEER channel attribute, see the topic *DEFINE CHANNEL* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sc10950_.htm

SSLPEERMAP extends the filtering on the channel attribute by not only blocking certain DNs from connecting, but specifically mapping DNs to a known local user ID. It also forces the message channel agent (MCA) to run as that user ID.

In addition to mapping that is based on the DN, it is possible to specify that the map occurs only if the DN originated from a particular IP address. As in the case of ADDRESSMAP, the address to match against in combination with the DN can be a specific address or a range of addresses.

Mapping that is based on asserted client user ID

User IDs are a familiar concept in all aspects of information technology, and their use is no less prevalent in WebSphere MQ. User IDs are of particular importance when security between a client application and an WebSphere MQ queue manager is considered. A client application runs anywhere and connects to a queue manager over a network.

Upon requesting a connection to a queue manager, a client application sends over its credentials, in particular the user ID that it is running as. The mapping that is based on asserted client user (USERMAP) channel authentication record can be used to map the user ID that the client asserts as its own to a known local user ID. In this case, rather than the client's asserted user ID being used to check against ACLs, it is the mapped user ID that is verified against the ACLs for the resources to which it is requesting access.

The user ID must match exactly the ID that is specified in the channel authentication record for the rule to apply.

In addition to mapping that is based on the client asserted user ID, it is possible to specify that the map occurs only if the user ID originated from a particular IP address. As in the case of ADDRESSMAP, the address to match against in combination with the client-asserted user ID can be a specific address or a range of addresses.

This rule type is applicable to server connection (SVRCONN) channels only.

Mapping that is based on remote queue manager name (QMGRMAP)

The USERMAP channel authentication record type applies to a client's connection to queue managers. When a queue manager connects to another queue manager, for example by using a sender-receiver channel pair, no user ID is passed to the receiving channel.

Therefore, the mechanism that is employed with a USERMAP channel authentication record cannot function. Instead, the queue manager name is passed over, which can be used to map. The mapping that is based on the remote queue manager name (QMGRMAP) channel authentication record type maps from the remote queue manager name to the local known user ID.

The queue manager name that is used to match against does not need to be a single explicit name. The name can be specified with wildcards to match multiple queue manager names.

In addition to mapping that is based on the remote queue manager name, it is possible to specify that the map occurs only if the remote queue manager name originated from a particular IP address. As in the case of ADDRESSMAP, the address to match against in combination with the queue manager name can be a specific address or a range or addresses.

This rule type is applicable to message channels that are running between two queue managers.

8.1.4 Blocking based on user ID (BLOCKUSER)

Except for the BLOCKADDR type rule, all other rules revolve around the mapping to or blocking of a user ID. The BLOCKADDR type rules result in an immediate decision about whether to allow the connection to progress. The mapping rules decide to map to a user ID, or potentially leave the incoming user ID unchanged. The blocking based on user ID (BLOCKUSER) type rule can be used to outright block a user from connecting to the queue manager.

BLOCKUSER rules are the last rules to be checked. If these rules are defined, they run after all other rules are checked and after any security exit is run. Even if a mapping rule was matched and an MCAUSER was mapped, the mapped user ID is checked against any BLOCKUSER rules, which also match.

The BLOCKUSER rule can be used to specify one or more explicit user IDs to be blocked in a comma-separated list.

For more information about and some examples of how to configure a rule of this type, see “Using MQSC” on page 107.

8.1.5 Configuration of channel authentication records

Channel authentication records can be configured by using the following methods:

- ▶ MQSC
- ▶ Programmable command format (PCF) messages
- ▶ WebSphere MQ Explorer

This section describes these methods and provides some examples of the options that are available when an environment is configured to use channel authentication records.

Configuring channel authentication records, irrespective of the method that is employed to do so, can be broken down into pieces. This structured approach can help turn the written or verbal requirement for the rules into the correct commands or sets of options in WebSphere MQ Explorer.

The following questions are useful to ask when you are considering which rule to use and how it should be configured:

- ▶ Which channels should this rule apply to?
- ▶ What are the credentials that this rule should validate against?
- ▶ Is this rule aiming to allow the connection or prevent the connection?
- ▶ Is this rule sufficient on its own?

The question of whether the rule is sufficient on its own is a key question to consider because often time a rule is created for a specific purpose but it alone does not cater for other situations.

For example, consider the following requirement. Access to the queue manager must be allowed only from addresses on the same subnet as the queue manager.

The following MQSC rule definition might be used to implement this requirement:

```
set ch1auth(*) type(ADDRESSMAP) ADDRESS('9.20.1-58.*') USERSRC(CHANNEL)
```

This rule alone, however, does not implement the full requirement. For example, an IP address that does not match the ADDRESS parameter still proceeds with the connection because there is no rule that is configured to block connections from other IP addresses. The following rule also is required to implement this requirement fully:

```
set ch1auth(*) type(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
```

This rule is more generic than the first rule and is therefore only matched if the more specific rule is not matched. The combination of these two rules is necessary to fully implement the stated requirement. Other examples of the options are shown in the following sections.

Using MQSC

Configuring channel authentication records by using MQSC is done by using the **set** command. Configuring a channel authentication record in MQSC resembles the following example:

```
set ch1auth(<generic-channel-name>) type(<rule-type>) <type-specific-options>
warn(<yes|no>) action(<add|replace|remove|removeall>)
```

Where:

- ▶ generic-channel-name

The name of the channel or channels when wildcards are used. When BLOCKADDR is used, this parameter must be '*'.

- ▶ rule-type

One of the following types: BLOCKADDR, ADDRESSMAP, SSLPEERMAP, USERMAP, QMGRMAP, or BLOCKUSER.

- ▶ type-specific-options

The credentials the rule should match against and is dependent on the type parameter.

- ▶ warn

Whether to have this rule generate a warning event message when it might match, or to be active and perform the block. To generate warnings channel events, it must be enabled on the queue manager (default to no).

- ▶ action

Whether to add a rule, replace an existing rule, remove a rule, or remove all rules of the specified type (default to add).

The type-specific-options provide the details that, when checked with the generic channel name parameter, make up the rule definition. For each rule type, the following mandatory options are available:

- ▶ BLOCKADDR

The list of addresses to be blocked is specified with the ADDRLIST keyword and is a comma-separated list of generic IP addresses.

- ▶ ADDRESSMAP

The address to match against is specified with the ADDRESS keyword. This parameter must be a single entry, but can be a generic specification.

- ▶ SSLPEERMAP

The DN to match against is specified with the SSLPEER keyword. This parameter must be a single entry, but can be a generic specification. This type can be optionally paired with the ADDRESS keyword.

- ▶ USERMAP

The client-asserted user name to match against is specified with the CLNTUSER keyword. This parameter must be a single entry and an exact specification. This type can be optionally paired with the ADDRESS keyword.

- ▶ QMGRMAP

The remote queue manager name to match against is specified with the QMNAME keyword. This parameter must be a single entry, but can be a generic specification. This type can be optionally paired with the ADDRESS keyword.

- ▶ **BLOCKUSER**

The list of user IDs to be blocked is specified with the USERLIST keyword and is a comma-separated list of exact user IDs. The exception to this restriction is the *MQADMIN keyword, which can be used in the USERLIST parameter. *MQADMIN represents any member of the WebSphere MQ administrators group. USERLIST does not allow group-based blocking, other than with *MQADMIN.

The mapping rules also allow a specification of to which user ID should be mapped if the credentials of the connection match the rule. How the user ID is mapped depends on the USERSRC keyword with the following options:

- ▶ **MAP**

This value tells WebSphere MQ to map to the user ID that is specified in the MCAUSER parameter of the rule that was matched. This value is the default option that makes the MCAUSER parameter mandatory in the default mode.

- ▶ **CHANNEL**

This value tells WebSphere MQ to allow the user ID that was received on the channel to act as the MCAUSER.

- ▶ **NOACCESS**

This value is an equivalent way to block access by using a mapping rule. The use of NOACCESS is useful as a catch all to supplement the specific mapping rules that are configured.

MQSC examples

The following list shows some examples of MQSC commands with a description of the requirement the rules implement:

- ▶ Block all privileged WebSphere MQ users from connecting to the queue manager:

```
set chlauth('*) type(BLOCKUSER) USERLIST('*MQADMIN')
```

- ▶ Do not allow connections to the queue manager over any channel that starts with SYSTEM, except for the SYSTEM.ADMIN.SVRCONN channel:

```
set chlauth('SYSTEM.*') type(ADDRESSMAP) ADDRESS('*) USERSRC(NOACCESS)  
set chlauth('SYSTEM.ADMIN.SVRCONN') type(ADDRESSMAP) ADDRESS('*)  
USERSRC(CHANNEL)
```

Important: The three rules that configured here exist by default when a V7.1 or above, queue manager is created. During initial testing of a configuration, applications, and queue managers might receive MQRC_NOT_AUTHORIZED errors because of these rules. Channel authentication records can be disabled in MQSC by using the command **alter qmgr chlauth(DISABLED)**.

If a queue manager is migrated to WebSphere MQ V7.1 or above from a previous version, the default rules are created but the queue manager attribute CHLAUTH is automatically set to DISABLED by default.

- ▶ Allow a known user ID (user1) to connect to channel MY.SVRCONN.USER1. Upon connection, the MCAUSER should run as a user ID that is defined locally to the queue manager (localuser1). The remote user ID is trusted only when it comes from a specific IP address (9.44.168.251), and all other access should be blocked from accessing this channel:

```
set chlauth('MY.SVRCONN.USER1') type(USERMAP) CLNTUSER('user1') +
MCAUSER('localuser1') ADDRESS('9.44.168.251')
set chlauth('MY.SVRCONN.USER1') type(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
```

- ▶ Block all SSL connections to the queue manager, where the peer DN is a known rogue attacker (the DN contains an organization name of FakeCo, and always specifies two organizational unit attributes, the first starting with letters Dev and the second containing any value):

```
set chlauth('*') type(SSLPEERMAP) SSLPEER('0=FakeCo,OU=Dev*,OU=*') +
USERSRC(NOACCESS)
```

- ▶ Prevent the warehouse queue managers (the names of which all end with WRHOUSE) from accidentally joining the back-office WebSphere MQ cluster because of administrative errors. Cluster channels are all defined as with '.CLUS.' in their name:

```
set chlauth('.CLUS.*') type(QMGRMAP) QMNAME('*WRHOUSE') USERSRC(NOACCESS)
```

Using programmable command format messages

Programmable command format (PCF) messages can be used to perform automated administration of WebSphere MQ resources, particularly as networks grow in size and complexity. For more information about PCF messages, see the topic *Programmable command formats reference* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/zr00120_.htm

This section describes the parameters that can be used in a program that uses PCF messages to administer channel authentication records. The options that are available are the same as the options that are described in “Using MQSC” on page 107 and they are functionally equivalent.

For more information about the structure of a PCF message to configure a channel authentication record, see the topic *Set Channel Authentication Record* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/pc20640_.htm

The key parameter, and the parameter that determines which other parameters must be supplied, is the *Type* parameter. This parameter can be one of the following values:

- ▶ MQCAUT_BLOCKADDR: Functionally equivalent to the MQSC BLOCKADDR keyword.
- ▶ MQCAUT_ADDRESSMAP: Functionally equivalent to the MQSC ADDRESSMAP keyword.
- ▶ MQCAUT_SSLPEERMAP: Functionally equivalent to the MQSC SSLPEERMAP keyword.
- ▶ MQCAUT_USERMAP: Functionally equivalent to the MQSC USERMAP keyword.
- ▶ MQCAUT_QMGRMAP: Functionally equivalent to the MQSC QMGRMAP keyword.
- ▶ MQCAUT_BLOCKUSER: Functionally equivalent to the MQSC BLOCKUSER keyword.

For more information about the mandatory parameters for each of these parameters, see the topic *Set Channel Authentication Record* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/pc20640_.htm

Because PCF messages are issued as part of a program, there are error codes that a program receives as a result of issuing the PCF command MQCMD_SET_CHLAUTH_REC, which fails. For more information about these error codes, see the topic *Set Channel Authentication Record* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/pc20640_.htm

Using WebSphere MQ Explorer

Channel authentication records can be administered by using WebSphere MQ Explorer. The use of WebSphere MQ Explorer is a convenient way to manage and administer remote queue manager resources from a central location. WebSphere MQ Explorer provides a wizard for creating new channel authentication records. This section shows an example of the wizard windows.

The following example shows the use of the channel authentication wizard to define a rule that blocks access based on SSL/TLS DN matching, by using WebSphere MQ Explorer. The example assumes that a queue manager already exists and matches against the following attributes:

- ▶ A channel name that begins with SYSTEM.
- ▶ A DN name that matches on the following attributes:
 - CN=*
 - L=Raleigh
 - OU=ITSO
- ▶ The IP address from which the connection originates starts with 9.44.168, followed by any number between 100 and 200 in the fourth place of the address.

The following steps show how to configure a channel authentication record to block access that is based on SSL/TLS DN matching:

1. Expand the queue manager in the navigator pane, and expand the Channels section of the queue manager.
2. Right-click **Channel Authentication Records** and select **New → Channel Authentication Record...** as shown in Figure 8-2.

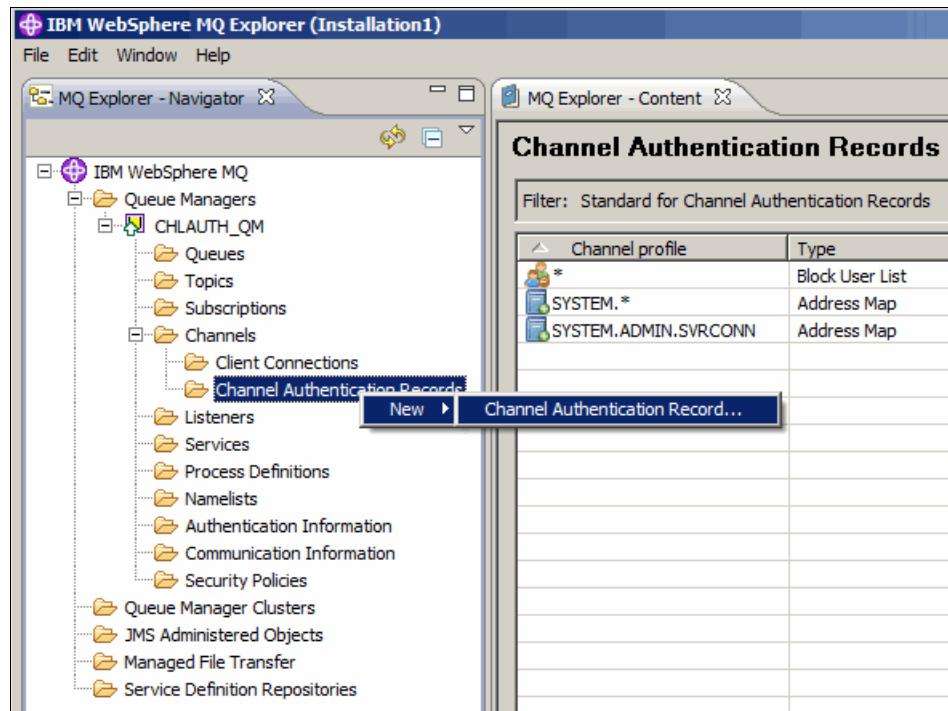


Figure 8-2 Starting the channel authentication record wizard

3. Figure 8-3 shows the first panel of the channel authentication record wizard. Click **Next**.

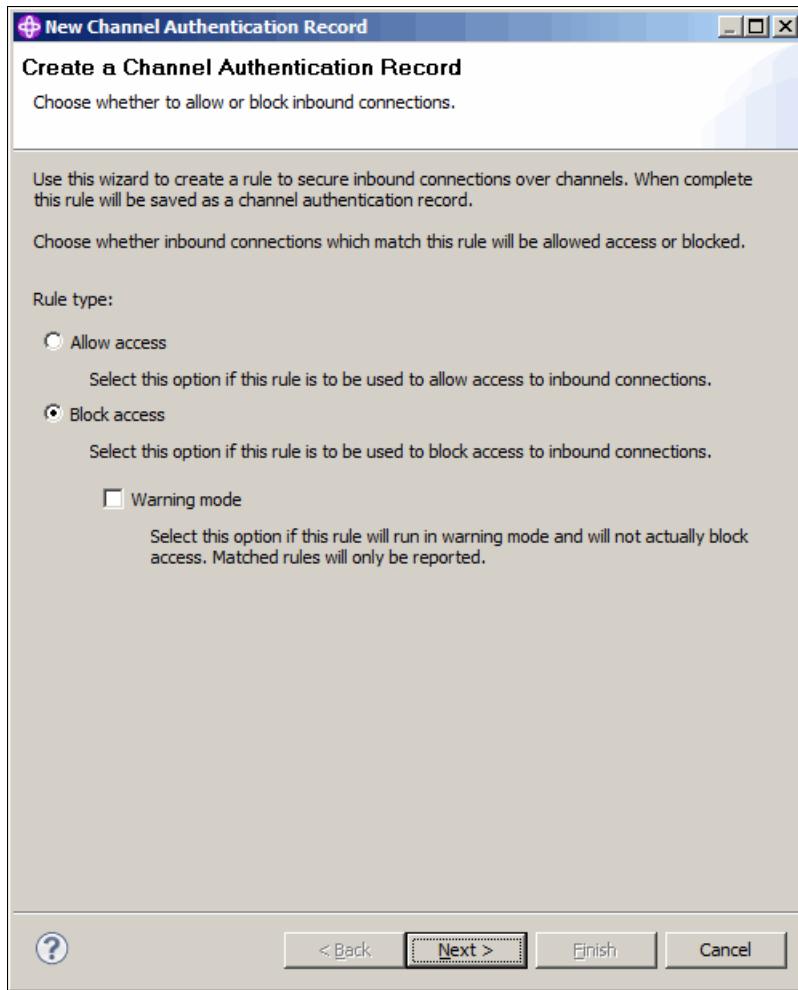


Figure 8-3 The first panel of the channel authentication record wizard

4. Figure 8-4 shows the panel with the options that are available to select the type of the channel authentication record. This panel is where the decision about the credentials to match against is made. In this example, select **SSL/TLS subject's Distinguished Name**. Click **Next**.

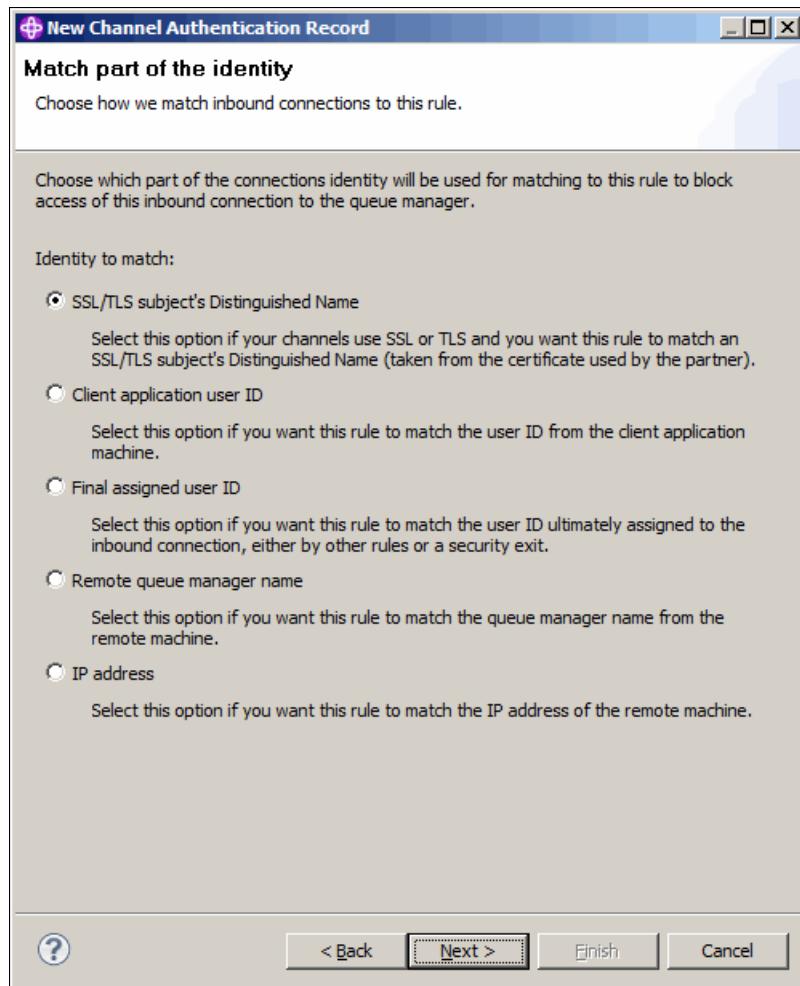


Figure 8-4 Selecting the type of channel authentication record

5. Figure 8-5 shows the panel that is used to configure to which channels this rule should apply. This rule applies to all channels that are named SYSTEM.*. Enter SYSTEM.* into the Channel profile box. Click **Show matching channels** to populate the table with the channels that are defined on the queue manager that match the specified channel profile. Click **Next**.

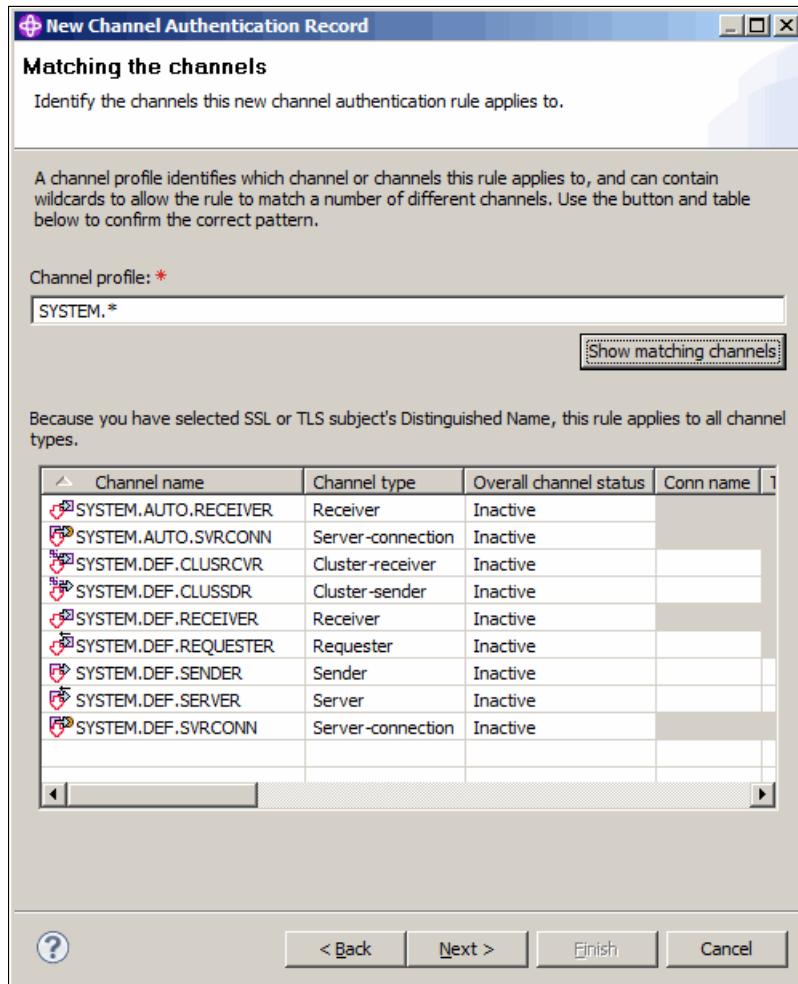


Figure 8-5 Defining which channels are affected by this rule

6. Figure 8-6 shows the panel where the precise credentials that are used to match against this rule are defined. Related examples are provided on the panel that are based on the type of channel authentication record that is being created. Enter CN=*,L=Raleigh,O=ITSO into the **SSL/TLS subject's Distinguished Name pattern** field and 9.44.168.100-200 into the **IP address pattern** field. Click **Next**.

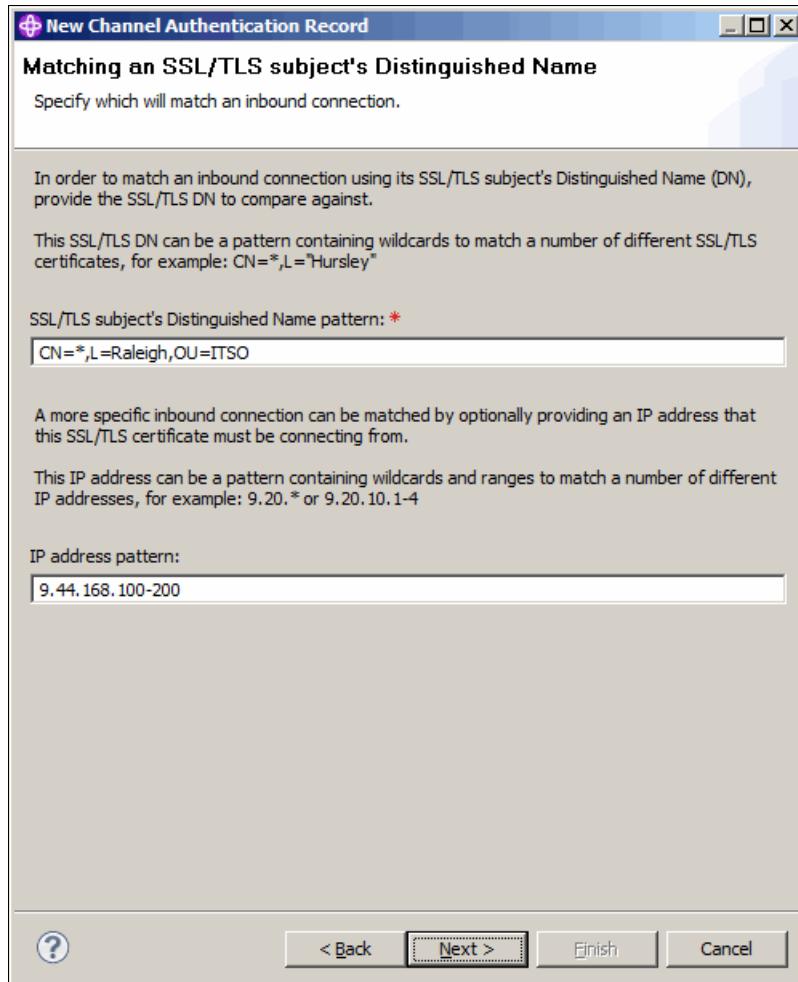


Figure 8-6 Defining the credentials which should be matched against

7. Enter a textual description of the channel authentication record to identify the function of the rule in the **Description of rule** field on the panel that is shown in Figure 8-7. Click **Next**.

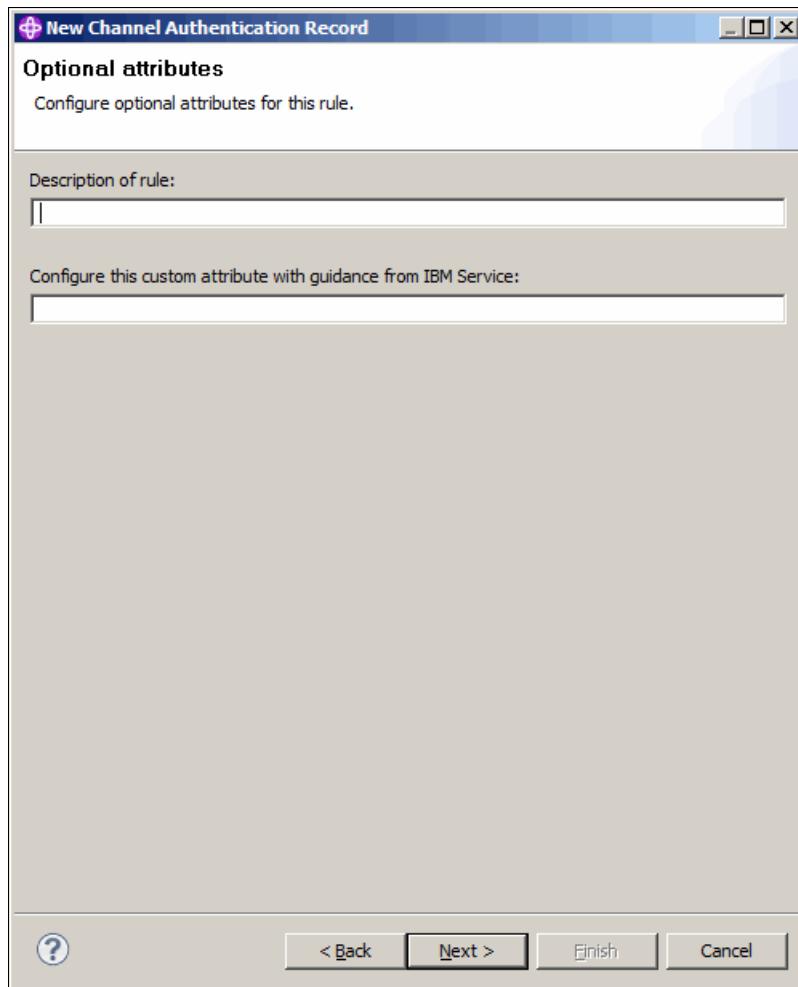


Figure 8-7 Optionally provide a description of the channel authentication record

8. Figure 8-8 shows the summary panel for the channel authentication record wizard. To create the record, click **Finish**. This final panel is useful in verifying that the configuration that is done reflects the requirement. It provides a worded explanation of the rule that is being created and the MQSC command that is used to create the record. The MQSC command can be pasted into a script for reuse or modification when scripting changes to channel authentication records on other queue manager definitions.

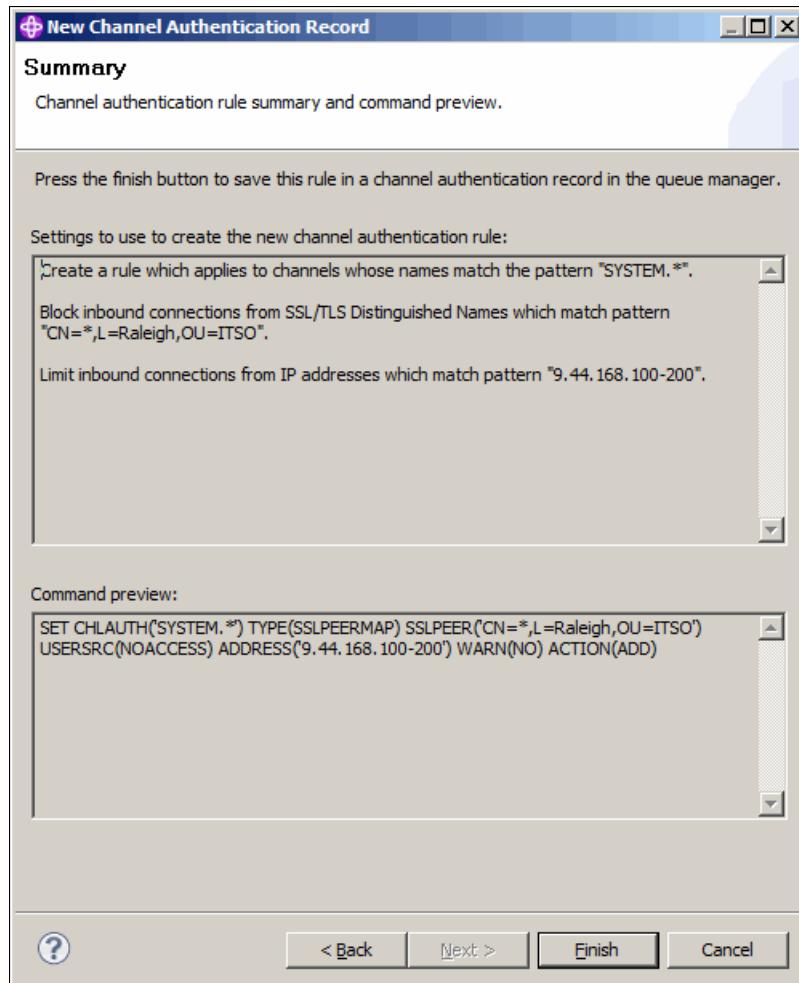


Figure 8-8 Summary panel that shows the channel authentication record definition to be created

8.1.6 IP address pattern matching

The specification of IP addresses in channel authentication records can range from a simple, single address to a specification of an address that contains ranges and wildcards. The following example shows a valid generic IP address that channel authentication records can match against:

9-14.*.250.25-35

This generic IP address represents any IP address that meets the following criteria:

- ▶ In the first position of the IP address, any number between 9 and 14, inclusive.
- ▶ In the second position of the IP address, any number.
- ▶ In the third position of the IP address, 250 only.
- ▶ In the fourth position of the IP address, any number between 25 and 35, inclusive.

The same rules for IP address matching apply to IPv6 addresses as they do for IPv4 addresses.

It is possible to create multiple channel authentication records of the same type, but covering different IP address ranges. However, no two records of the same type can contain IP address entries that contain ranges that overlap. Example 8-1 shows the result of trying to create two channel authentication records that include overlapping IP address ranges.

Example 8-1 Overlapping IP address ranges in MQSC

```
set chlauth(*) type(ADDRESSMAP) ADDRESS('9-14.*.250.25-35') MCAUSER('user1')
  1 : set chlauth(*) type(ADDRESSMAP) ADDRESS('9-14.*.250.25-35')
MCAUSER('user1')
AMQ8877: WebSphere MQ channel authentication record set.

set chlauth(*) type(ADDRESSMAP) ADDRESS('9-14.*.250.30-40') MCAUSER('user1')
  2 : set chlauth(*) type(ADDRESSMAP) ADDRESS('9-14.*.250.30-40')
MCAUSER('user1')
AMQ9781: IP address overlaps with previous definition.
```

In the fourth position of the IP addresses, the range 25 - 35 overlaps with the range 30 - 40.

Warning: Configuring multiple channel authentication records that contain IP addresses that do not overlap but do contain mutually exclusive ranges in the same position results in an AMQ9781 error if WebSphere MQ 7.1.0.0 is used. This issue is corrected in PTF UK84179, in Fix Pack V7.1.0.1 and affects only V7.1

8.1.7 SSLPEER pattern matching

When an SSLPEERMAP channel authentication record is used, the DN value to match against can be an exact reference to a DN or it can be a generic reference that potentially matches multiple incoming DNs. If there is a single channel authentication record, specifying the DN to match can be a mixture of exact and generic attributes, including partial generic attributes. The following example shows possible combinations:

```
MAIL=*>us.ibm.com,CN=*,O=IBM,OU=ITSO,OU=Dev*,L=Raleigh,S=NC
```

If an attribute is specified in the channel authentication record, it must be present in the incoming DN to be considered a match. However, if the incoming DN contains attributes that are not specified in the channel authentication record, for matching purposes they are ignored. As an example, consider the following SSLPEERMAP channel authentication record:

```
set chlauth('MY.SSL.RCVR') type(SSLPEERMAP) SSLPEER('CN=*,OU=IBM,L=Ral*')
MCAUSER('my_ssl_user')
```

The previous rule maps any incoming DNs matching the SSLPEER value to the user ID my_ssl_user. Now consider the following incoming DNs:

```
CN=ITSO_Cert,OU=IBM,L=Raleigh,S=NC
```

Although the previous DN contains an attribute (S=NC) that is not specified in the channel authentication record, the rule is still matched because all of the attributes in the SSLPEER value appear in the incoming DN, as shown in the following example:

```
OU=IBM,L=Raleigh
```

If the previous DN is received on an incoming channel, it is not matched because it does not contain a CN attribute. The channel authentication record specifies that although the CN attribute can be anything, it must be present in the DN to be a match, as shown in the following example:

CN=ITSO_Cert_Hursley,OU=IBM,L=Hursley,C=GB

In the previous example, the CN attribute matches the rule, the OU attribute matches the rule, the C attribute is not present in the rule so matching is ignored. However, the L attribute does not match against Ra1*. Therefore, in this case, the rule as a whole does not match.

When two or more channel authentication records that use SSLPEERMAP are configured, the possibilities for incoming DNs to match multiple rules becomes evident. For more information about the logic that is applied when WebSphere MQ selects the channel authentication record to use in the case of more than one possibility, see the topic *Channel authentication records* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/zs14190_.htm

Channel authentication records are first matched based on the most specific channel profile. Consider whether one rule states the channel profile as '*' and another rule of the same type states the channel profile as 'SYSTEM.*'. When a connection request comes in on, for example, SYSTEM.DEF.SVRCONN, it always matches the 'SYSTEM.*' rule because the channel profile name is a more specific match than '*'.

The following examples follow the logic through to help explain which channel authentication record is chosen when the channel profile on the rules is the same.

When there are multiple channel authentication records defined that this DN could match, the matching rule is the rule that is defined as most specific. The dimensions to the specificity of a DN are considered in the following order:

1. The specificity precedence order of the DN attributes.
2. The generic value of each attribute.

For more information about the precedence order, in terms of specificity, of the DN attributes, see the topic *Channel authentication records* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/zs14190_.htm

SERIALNUMBER is the most specific value and DNQ is the least specific value. How specific the attribute values are is where the complexity increases. The following examples show two channel authentication record SSLPEER values that match the incoming DN and explain which one matches and why they match.

The incoming DN uses the following attributes in the examples:

MAIL=user@us.ibm.com,CN=Test Team,OU=CA,OU=WebSphere
MQ,OU=Test,OU=Security,DC=user,DC=us,DC=ibm,O=IBM,C=US,L=Raleigh,ST=NC

- ▶ Example 1: Two SSLPEERMAP channel authentication records that specify the same channel profile name, with SSLPEER values set as shown in the following rules:

Record 1: CN=Test Team,OU=CA,OU=*,OU=*,OU=*,L=Raleigh

Record 2: CN=Test Team,OU=*,OU=*,OU=*,OU=Security,L=Raleigh

In this example, the second rule is chosen. The CN attributes are equivalent to each other. Each SSLPEER expects four OU attributes to be present in the incoming DN, which is a match. Because it is possible to specify multiple OU attributes, there also is an implied order of specificity. For OU attributes, this order reads left to right, from least specific to most specific. Therefore, in this case, the most specific OU value is the one that is rightmost, and the values are not equivalent so this criteria is used to distinguish the two. Because the OU in the fourth position in the Record 1 is generic, and the same attribute in Record 2 is specified exactly, the Record 2 is considered more specific overall.

- ▶ Example 2: Two SSLPEERMAP channel authentication records that specify the same channel profile name, with SSLPEER values set as shown in the following rules:

Record 1: CN=Test Team,OU=*,OU=WebSphere*,OU=*,OU=*rity,L=Raleigh

Record 2: CN=Test Team,OU=*,OU=WebSphere*,OU=*,OU=Secu*,L=Raleigh

In this example, the differing DN attributes only are the OU values in the fourth position. Here, the values are partial matches for the value Security. The first consideration is for the number of characters in the matching value that are specified explicitly. In this case, there are four characters in both values, so this criteria cannot be used to distinguish between them. The final deciding factor is not a case-sensitive, character-by-character alphabetic selection of the explicit part of the value. In this case, r comes before s in the English alphabet, so Record 1 is matched and is considered more specific overall.

Important: OU and DC attributes are the only attributes that are permitted to appear more than once in a DN. OU attributes feature an implied order of specificity, least specific to most specific, or left to right. DC attributes feature an implied order of specificity, which is the opposite of OU attributes. Therefore, the left-most DC attribute is most specific and the rightmost DC attribute is least specific.

- ▶ Example 3: Two SSLPEERMAP channel authentication records that specify the same channel profile name, with SSLPEER values set as shown in the following rules:

Record 1: CN=Test Team,OU=CA,OU=WebSphere MQ,OU=Test,OU=Security

Record 2: MAIL=*&us.ibm.com

In this example, Record 1 is specified by using exact values in each of its attributes, and Record 2 contains a partially generic value for MAIL. Although Record 1 appears more specific, in this case, Record 2 is matched. MAIL is the second most specific DN attribute and because it is specified in Record 2 but not Record 1, Record 2 is considered more specific overall.

- ▶ Example 4: This example shows one further dimension to multiple attribute checking. Two SSLPEERMAP channel authentication records that specify the same channel profile name, with SSLPEER values set as shown in the following rules:

Record 1: OU=WebSphere MQ,O=IBM
 Record 2: OU=*,OU=*,OU=*,OU=Secur*

Record 1 specifies attributes explicitly, but is matching only one OU attribute and the O attribute. Record 2 matches four OU attributes, which all contain a generic or partially generic value. Although the attribute values in Record 1 are less generic, Record 2 is matched. The match occurs because it is considered more specific to match against four OU attributes than just one, even if the values of those attributes are generic.

8.1.8 Verifying channel authentication records

After channel authentication records are created, they are active immediately and checked on all new incoming connection requests since their creation. In the same way that channel authentication records are set, they also can be displayed. By using the **display chauth** command in MQSC, it is possible to visually verify that what was created is what was intended.

Example 8-2 shows the command and output from a **display chauth** command in MQSC, which shows the default rules for a new V7.1 queue manager.

Example 8-2 dis chauth MQSC command

```
display chauth(*) all
  1 : display chauth(*) all
AMQ8878: Display channel authentication record details.
  CHLAUTH(SYSTEM.ADMIN.SVRCONN)          TYPE(ADDRESSMAP)
  DESCRIPTOR(Default rule to allow MQ Explorer access)
  CUSTOM( )                            ADDRESS(*)
  USERSRC(CHANNEL)                   ALTDATE(2012-10-04)
  ALTTIME(13.56.03)

AMQ8878: Display channel authentication record details.
  CHLAUTH(SYSTEM.*)          TYPE(ADDRESSMAP)
  DESCRIPTOR(Default rule to disable all SYSTEM channels)
  CUSTOM( )                  ADDRESS(*)
  USERSRC(NOACCESS)           WARN(NO)
  ALTDATE(2012-10-04)         ALTTIME(13.56.03)

AMQ8878: Display channel authentication record details.
  CHLAUTH(*)          TYPE(BLOCKUSER)
  DESCRIPTOR(Default rule to disallow privileged users)
  CUSTOM( )          USERLIST(*MQADMIN)
  WARN(NO)           ALTDATE(2012-10-04)
  ALTTIME(13.56.03)
```

Similarly, this information is present in WebSphere MQ Explorer. Figure 8-9 shows the same channel authentication records as the MQSC in Example 8-2, but viewed with WebSphere MQ Explorer.

Channel profile	Type	F	C	F	Address	User source	User list	Warning	Description
MQADMIN*	Block User List						*MQADMIN	No	Default rule to disallow privileged users
SYSTEM.*	Address Map	*				No Access		No	Default rule to disable all SYSTEM channels
SYSTEM.ADMIN.SVRCONN	Address Map	*				Channel			Default rule to allow MQ Explorer access

Figure 8-9 The default channel authentication records, which are viewed with WebSphere MQ Explorer

In addition to displaying channel authentication records, you can test the rules by using WebSphere MQ. This test is of the form of a what-if test. The MQSC interface provides an extra parameter on the **display chlauth** command called MATCH. For more information about the possible values for MATCH and their function, see the topic *DISPLAY CHLAUTH* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sc14450_.htm

In particular, the RUNCHECK option can be used to verify what happens at run time when the channel authentication records are active. Example 8-3 shows the type of output the RUNCHECK option provides. The **display** command is checking what happens if a client application that is running as user ID MUSR_MQADMIN were to connect to the SYSTEM.DEF.SVRCONN channel from address 9.44.168.251. The **rwmqsc** command outputs the channel authentication record that matches. In this case, the channel is blocked as the rule states NOACCESS.

Example 8-3 MATCH(RUNCHECK) example output

```
display chlauth('SYSTEM.DEF.SVRCONN') MATCH(RUNCHECK) ADDRESS(9.44.168.251)
CLNTUSER('MUSR_MQADMIN')
  1: display chlauth('SYSTEM.DEF.SVRCONN') MATCH(RUNCHECK) ADDRESS(9.44.168.251)
CLNTUSER('MUSR_MQADMIN')
AMQ8878: Display channel authentication record details.
  CHLAUTH(SYSTEM.*)
  ADDRESS(*)
  TYPE(ADDRESSMAP)
  USERSRC(NOACCESS)
```

The RUNCHECK command option can be used as a mechanism for testing that the behavior that was implemented is the behavior that was intended.

One other way to test channel authentication records after they are defined is to use the WARN mode. Where channel authentication records are blocking access by using BLOCKADDR, BLOCKUSER, or mapping by using NOACCESS, it is possible to have the rule in place but not active. On a blocking rule, setting WARN to YES results in WebSphere MQ event messages that are generated when the rule should have been invoked to block the access. It is important to note that blocking does not occur and the incoming connection might still go ahead, the system issues a warning that the connection would have been blocked by one of the channel authentication records. The use of WARN set to YES enables the verification that the rules that are defined do not block expected users without the need for those users to suffer downtime by putting erroneous rules immediately into a live production system. Configuration errors that are found as a result of the warning messages can be rectified and retested without continued user outage. The events that are generated as a result of channel authentication records are on a queue called SYSTEM.ADMIN.CHANNEL.EVENT. The queue manager needs to have channel events enabled. The following MQSC command enables channel events:

```
alter qmgr chlev(ENABLED)
```

8.2 Remote queue access control

This section describes the new facilities that are in WebSphere MQ V7.1 for access control of remote queues and remote clustered queues. The facilities that are described are available on releases of WebSphere MQ before V7.1 for z/OS. Therefore, this section is specifically about the new features on Windows, UNIX, Linux, and IBM i.

8.2.1 Access permissions on remote objects before WebSphere MQ V7.1

Access control for WebSphere MQ on distributed platforms can be configured with the object authority manager (OAM). For more information about the full functionality of the OAM, see the topic *Object authority manager (OAM)* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/fa16420_.htm

Access can be granted (or revoked) for a user ID or group ID to perform (or disallow) certain tasks against a named WebSphere MQ resource. For example, the ability to put and get messages to and from a named queue, or the ability to subscribe to a named topic object.

A remote queue definition is a locally defined named object. A remote queue definition can be put to, and that message is then routed across a channel and to a queue that is physically on a remote queue manager. When a named object that is a remote queue definition is defined, the OAM can be used to grant access on that named object.

The following instances show when the remote queue definition is not defined locally but putting to a queue on remote queue manager is still wanted:

- ▶ When an application addresses the destination of the message explicitly by using the remote queue and queue manager name.
- ▶ When the queue that is put to is a cluster queue that is hosted on a remote queue manager.

In WebSphere MQ V7.0.1 and earlier versions, the authority check is performed against the transmission queue. The transmission queue is an object that must be in place for channels to accept messages for transfer to remote queue managers. To reduce object administrator costs, transmission queues can be used to hold messages for more than one channel, and hence potentially for multiple different queue managers. There is less access control granularity when access is checked against the transmission queue (instead of remote queue). Granting permission for one user ID to put messages to one remote queue manager via the transmission queue automatically grants that user ID permission to put to all remote queue managers accessible via that queue.

Figure 8-10 shows that user ID user1 was granted permissions to put messages to the remote queue definition TARGET.RQ.1 and to the two transmission queues that are defined on QM1. However, the intention is that user1 should not be allowed to put messages to cluster queues on QM3. Because user1 is allowed to put messages to the SYSTEM.CLUSTER.TRANSMIT.QUEUE (to reach TARGET.CCLUS.Q), user1 is implicitly granted permissions to put messages to QM3 via the SYSTEM.CLUSTER.TRANSMIT.QUEUE.

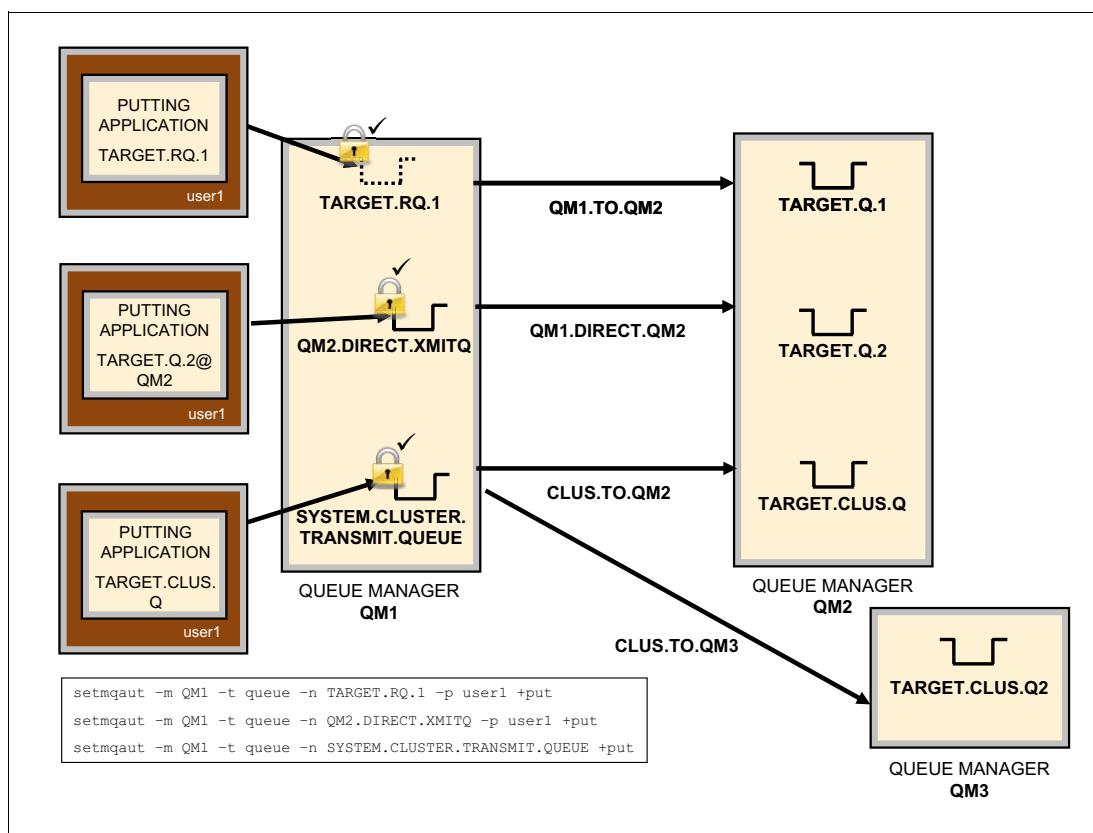


Figure 8-10 Access control on remote objects before WebSphere MQ V7.1 on distributed platforms

8.2.2 Access permissions on remote objects in WebSphere MQ V7.1

WebSphere MQ V7.1 on distributed platforms introduces the ability to grant authority to put messages to a specific remote queue manager, rather than have the access check performed against the transmission queue. The final authority to grant permission to put messages to the physical remote queue is with the administrator of the permissions of the queue's host queue manager.

To configure the permissions by using the OAM, the **setmqaut** command is provided with the WebSphere MQ installation. The **setmqaut** command features the following syntax:

```
setmqaut -m <QMgrName> -n <Profile> -t <ObjectType> -p <Principal>|-g <Group>  
Permissions
```

The following options are available for the **setmqaut** command:

- ▶ **QMgrName**

If the command is not referencing the system's default queue manager, this option represents the queue manager on which the resources are granted or revoked. This option is not required when the default queue manager is used.

- ▶ **Profile**

This option represents the named object or objects to which the authorities apply. The profile name can contain wildcard characters, and therefore match against multiple named objects of the specified type.

- ▶ **ObjectType**

This option represents the type of object to which the authorities are applied; for example, a queue, topic, channel, remote queue manager, or any other WebSphere MQ resource.

- ▶ **Principal**

This option represents the user ID to which to grant or revoke the authority. The option can be specified multiple times on the same command, each instance is preceded with the **-p** flag.

- ▶ **Group**

In addition to the user ID, it is possible to specify that authorities should be granted based on group ID. This option can be specified multiple times on the same command, each instance is preceded with the **-g** flag.

- ▶ **Permissions**

This set of keywords represents the specific authorities to be granted or revoked. Each keyword is preceded by a + or a - to represent an authority that is granted or revoked. The full list of authorities that can be used is available in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sc14450_.htm

Figure 8-11 shows that with WebSphere MQ V7.1 the user ID, user1 was granted the correct granular access (allowed only to put to cluster queues on QM2) by using fewer configuration commands. The permissions to put messages are granted based on the queue manager name, rather than the local transmission queue name. In this example, the transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE, provides access to QM2 and QM3. However, the configured permissions allow only user1 to put messages to queues on QM2.

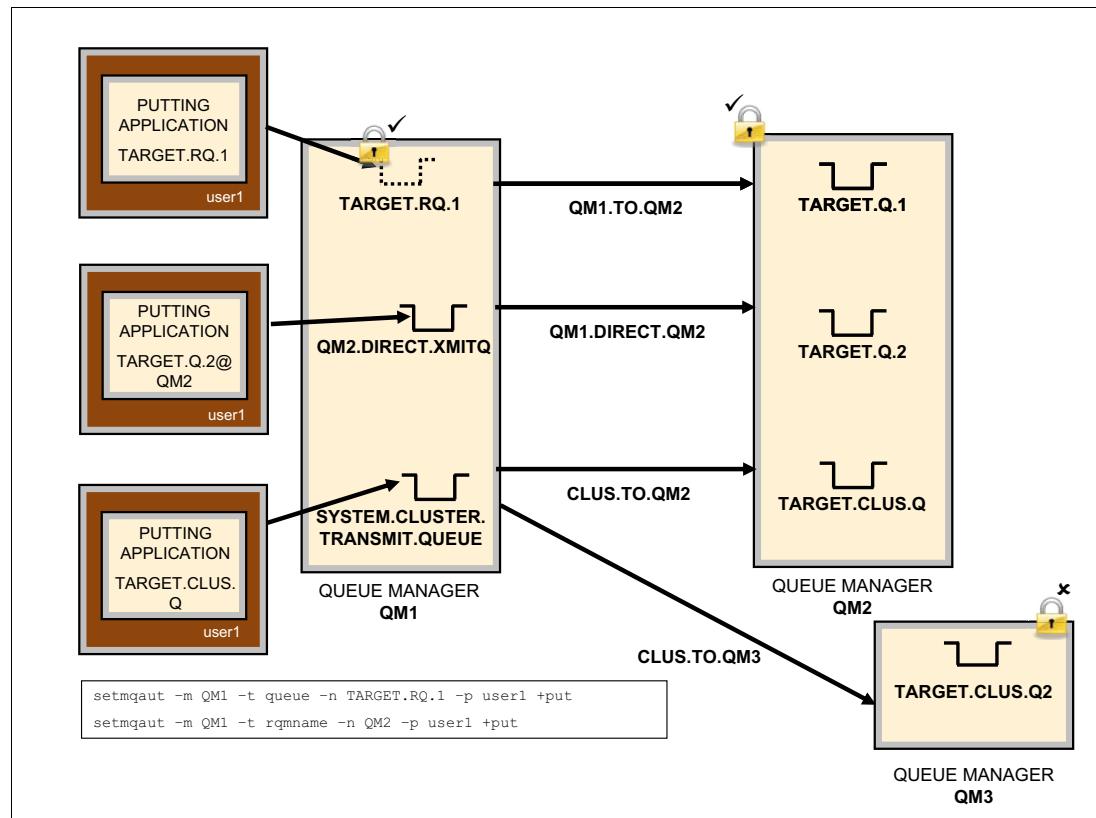


Figure 8-11 Simplified and more granular control over access to remote objects

The following command example shows the parameters and options that are used to configure the authority such that user1 can put messages to cluster queues on QM2, but not QM3:

```
setmqaut -m QM1 -t RQMNAME -n QM2 -p user1 +put
```

Combined with the following command, user1 has access to all the resources it requires and explicitly has no access to the resources that it does not require:

```
setmqaut -m QM1 -t queue -n TARGET.RQ.1 -p user1 +put
```

8.3 Enhanced cryptographic support with IBM Global Security Kit V8

WebSphere MQ uses IBM Global Security Kit (GSKit) to provide cryptographic function on Windows, UNIX, and Linux. This section describes the enhancements in cryptographic support that uses IBM GSKit V8.

Releases before WebSphere MQ V7.1 use IBM Global Security Kit (GSKit) V7 to provide cryptographic function for SSL/TLS and certificate management tools. In WebSphere MQ fix pack 7.0.1.4, the AltGSKit parameter was implemented to be able to use later versions of GSKit. Also, GSKit V8 began shipping as an optional installation package.

Many other IBM products also ship with GSKit, and WebSphere MQ V7.0 is implemented to use a system-wide installation of GSKit. The system-wide installation makes WebSphere MQ susceptible to alterations of GSKit versions by other products that results in untested combinations of WebSphere MQ and GSKit levels.

WebSphere MQ V7.1 uses GSKit V8 by default and is no longer susceptible to changes as a result of other product upgrades that also install a GSKit version. This section describes the installation configuration of GSKit V8 with WebSphere MQ V7.1 and some of the extra features now available.

Important: WebSphere MQ V7.1 requires GSKit V8 level 8.0.14.12, which is the version that is shipped with V7.1.0.0 WebSphere MQ. This prerequisite is enforced by a code check at run time.

8.3.1 Local installation of GSKit 8

Chapter 7, “Multiple installation support on Windows, UNIX, and Linux” on page 75 introduces the feature of having multiple WebSphere MQ installations on a single system. The ability to install multiple WebSphere MQ versions, which are combined with the differing versions of GSKit that are shipped with each level of WebSphere MQ, means that each installation now uses its own copy of GSKit. WebSphere MQ V7.0.1.6 uses the global installation of GSKit V7 on the machine while each installation of WebSphere MQ V7.1 and above uses its own copy of GSKit V8.

The installation location of GSKit V8 is dependent on the installation of WebSphere MQ V7.1. The changes to installation locations are described in 6.2, “Relocatable installations for UNIX and Linux” on page 72. Where WebSphere MQ V7.1 is installed in <MQ_INST_DIR>, the location of GSKit V8 is in <MQ_INST_DIR>/gskit8.

8.3.2 WebSphere MQ commands for certificate management

In versions of WebSphere MQ before V7.1, the command-line certificate management tools depend on the platform. The following commands on Windows are available:

- ▶ **gsk7cmd or runmqckm**
The Java based command-line tool that is used for certificate management
- ▶ **gsk7ikm or strmqikm**
The graphical user interface (GUI) tool that is used for certificate management
- ▶ **gsk7capicmd**
The C-based, command-line tool that is used for certificate management

On UNIX and Linux, the WebSphere MQ wrapped commands **runmqckm** and **strmqikm** are not available.

When WebSphere MQ V7.1 is used, the certificate management commands are wrapped in WebSphere MQ commands to ensure that the GSKit installation that is used is from the corresponding WebSphere MQ installation. The following commands are available:

- ▶ **runmqckm**
The Java based command-line tool that is used for certificate management
- ▶ **strmqikm**
The graphical user interface tool that is used for certificate management
- ▶ **runmqakm**
The C-based command-line tool that is used for certificate management. This command supports stronger encryption algorithms and must be used if standards compliance is required. For more information, see 8.4.1, “FIPS 140-2 compliance on Windows, UNIX, and Linux”.

For more information about the certificate management commands and their options, see the topic *Using iKeyman, iKeycmd, runmqakm, and runmqckm* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12140_.htm

8.3.3 Increased CipherSpec support

WebSphere MQ supports various CipherSpecs when SSL/TLS is used. The current list of supported CipherSpecs is available in the topic *Specifying CipherSpecs* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12870_1.htm

WebSphere MQ V7.1 supports the following SSL/TLS protocols. The protocol that is used is determined by the CipherSpec that is defined on the channel definition:

- ▶ SSL 3.0
- ▶ TLS 1.0 (also sometimes referred to as SSL 3.1)
- ▶ TLS 1.2

The progression from SSL 3.0 to TLS 1.2 does imply an order of improvement of the cryptographic functions that are covered by each protocol. Typically, TLS 1.2 CipherSpecs are stronger than SSL 3.0 CipherSpecs.

A move forward in WebSphere MQ V7.1 is the inclusion of Elliptic Curve cryptography. Elliptic Curve cryptography uses the mathematics of elliptic curves to form the basis of encryption algorithms. The CipherSpecs that support elliptic curve cryptography are those CipherSpecs that contain either or both of the following key words in their names:

- ▶ ECDHE_Elliptic Curve Diffie-Hellman Ephemeral key exchange
- ▶ ECDSA_Elliptic Curve digital signature algorithm

In addition to Elliptic Curve cryptography, WebSphere MQ V7.1 introduces the use of Secure Hash Algorithm-2 (SHA-2) for an increased level of robust data integrity.

8.3.4 Extended distinguished name attribute support

WebSphere MQ V7.1 introduces support for a range of DN attributes to identify an entity with a digital certificate. The list of supported attributes can be found in the topic *Distinguished Names* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy10570_.htm

A DN can contain only one entry of each attribute, except the following attributes:

- ▶ OU: Organizational Unit
- ▶ DC: Domain Component

The OU and DC attributes are optional but can be specified multiple times within a single DN. Because multiple entries are permitted, there is an implied order of specificity. For more information about the effects of this order, see 8.1.7, “SSLPEER pattern matching” on page 118.

8.3.5 Enhanced cryptographic hardware support on Windows, UNIX, and Linux

Cryptographic functions and certificate storage can be managed and performed by hardware modules, rather than software implementations of the same functionality. GSKit V8 introduces a set of cryptographic hardware that it supports to perform these functions. By extension, WebSphere MQ V7.1 also supports the use of the same hardware modules. For more information and for a list of supported pieces of cryptographic hardware, see the technote *Cryptography Card List for WebSphere MQ v7.1 and v7.5* at this website:

<http://www-01.ibm.com/support/docview.wss?uid=swg21516803>

Configuring WebSphere MQ to use cryptographic hardware is described in the topic *Configuring for cryptographic hardware on UNIX, Linux, or Windows systems* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12380_.htm

8.4 Cryptographic standards compliance

WebSphere MQ V7.1 includes features to enable and enforce compliance with the following strict cryptographic standards:

- ▶ Federal Information Processing Standards - 140-2 (FIPS 140-2)

The National Institute for Standards and Technology (NIST) produces recommendations and standards. FIPS is one of those standards.

- ▶ Suite B Cryptography (SUITEB)

The National Security Agency (NSA) recommends a set of interoperable cryptographic algorithms in its Suite B standard.

Standards are enforced by using configuration parameters on the queue manager or client environment. The compliance is facilitated by an appropriate choice of CipherSpec that is based on the standard. The following sections describe the options that are available for cryptographic standards compliance.

The topic *Specifying CipherSpecs* in the Information Center provides a full list of CipherSpecs and is referred to in the next sections concerning the standards compliance columns in the table at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12870_1.htm

8.4.1 FIPS 140-2 compliance on Windows, UNIX, and Linux

Compliance of FIPS at level 140-2 is available in WebSphere MQ since V6. The definition of the standard changes regularly as encryption algorithms are broken over time. Therefore, the list of CipherSpecs that meet the requirements of FIPS compliance also evolves over time. The *Specifying CipherSpecs* topic in the Information Center shows the full list of CipherSpecs that are supported by WebSphere MQ. Only CipherSpecs that are FIPS-compliant are the CipherSpecs with a yes in the FIPS column at this website;

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12870_1.htm

The queue manager includes a configuration option SSLFIPS that can be set to YES or NO by using MQSC to enforce or not use FIPS-compliant CipherSpecs. For client applications, there is a programmable option in the MQSCO structure, FipsRequired; an environment variable MQSSLFIPS, and a parameter that can be used in the client initialization file, SSLFipsRequired, which also can be set to YES or NO.

Setting to NO: Setting SSLFIPS to NO does not mean that FIPS-compliant CipherSpecs cannot be used on channel definitions. The No setting means that compliance to the standard is not enforced.

8.4.2 SUITEB compliance on Windows, UNIX, and Linux

SUITEB compliance is a new feature in WebSphere MQ V7.1. Similar to FIPS compliance, there is a queue manager configuration option to enable enforcement of SUITEB-compliant CipherSpecs.

For more information about SUITEB and the CipherSpecs that are compliant when they are used with WebSphere MQ, see the topic *NSA Suite B Cryptography in WebSphere MQ* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy11035_.htm

Important: In the table on the NSA Suite B Cryptography in WebSphere MQ page, take note of the CipherSpecs in the Suite B 128-bit or Suite B 192-bit column that are labeled as allowed.

SUITEB compliance has four levels of operation. The following options and their implications of the level of compliance are available:

- ▶ None
This option indicates no enforcement of SUITEB compliance.
- ▶ 128-bit
This option enforces the use of CipherSpecs that meets at least the minimum specification for 128-bit SUITEB compliance.
- ▶ 192-bit
This option enforces the use of CipherSpecs that meets at least the minimum specification for 192-bit SUITEB compliance.
- ▶ 128-bit, 192-bit
This option can be set explicitly and is conceptually the same as 128-bit compliance.

For more information about the methods by which SUITEB can be enforced for the queue manager and a client application, see the topic *Configuring WebSphere MQ for Suite B* in the Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy11055_.htm

8.4.3 FIPS 140-2 compliance on z/OS

WebSphere MQ V7.1 introduces FIPS enforcement on z/OS and Windows, UNIX, and Linux. Previous versions of WebSphere MQ on z/OS do not support the use of the SSLFIPS queue manager parameter and therefore have no method to enforce the FIPS 140-2 standard on its SSL/TLS connections.

The following FIPS-compliant CipherSpecs are supported on z/OS:

- ▶ TLS_RSA_WITH_AES_128_CBC_SHA
- ▶ TLS_RSA_WITH_AES_256_CBC_SHA
- ▶ TLS_RSA_WITH_3DES_EDE_CBC_SHA

Enabling FIPS compliance on the z/OS queue manager is done with the SSLFIPS attribute set to YES or NO by using MQSC.



Granular control over dead-letter queue usage

Two primary qualities of WebSphere MQ are the ability to decouple applications from the target destinations for messages and publications, and provide asynchronous communication between applications. These features allow a great deal of flexibility for workload distribution and processing.

When message delivery is local and the communicating applications are available, knowing when queues are not available or reached their capacity is straightforward. An application can put a message to a queue; if the queue is not usable, the queue manager sends an immediate response to tell the requestor there is a problem. Applications putting message to remote or clustered queues are notified only if there is a problem putting to the local representations of those queues (the transmission queues that provide the local storage). After a message is sent across a channel, no direct communication is available to the original application.

This chapter contains the following sections:

- ▶ Channel message delivery
- ▶ Publication delivery for topics
- ▶ The USEDLQ attribute
- ▶ Defining USEDLQ on channels
- ▶ Defining USEDLQ on topics

9.1 Channel message delivery

For channels, the receiving queue manager has the following decisions to make when a message arrives that cannot be put to the target queue:

1. If the queue manager includes a defined dead-letter queue (DLQ), undeliverable messages and publications are traditionally put to that queue.
2. If the put to the DLQ fails and receiver channel is configured to allow retries (MRRTY attribute on the receiver channel), the queue manager attempts the put to the target queue until the put is successful or the retry attempts are exhausted.
3. If no DLQ is available and all the retries were unsuccessful, the following steps are taken:
 - a. If the messages are persistent, the message is rejected and the channel closed.
 - b. If the messages are nonpersistent and the channel is defined to support fast delivery of nonpersistent message, undeliverable nonpersistent messages are discarded.

For more information about the DLQ, see the following topics in the Information Center:

- ▶ http://www.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/ic10420_.htm
- ▶ http://www.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/ic19430_.htm

As noted in the Information Center, putting message to the DLQ can introduce out of sequence message delivery. Out of sequence delivery is costly for some applications, even when there is good resynchronization code. Dedicating a queue manager for a single application can be as costly.

9.2 Publication delivery for topics

When an application performs an MQPUT to a topic, the queue manager attempts to deliver the publication to all of the subscriber-defined destinations. The delivery attributes of the topic define what happens if destinations are unusable.

The message delivery attributes are based on whether the individual publication is defined as persistent or non-persistent. In releases before WebSphere MQ v7.1, if a publication was delivered to a queue manager with a defined DLQ, undeliverable publications were put there. The following delivery attributes and settings are used:

1. Persistent message delivery (PMSGDLV), which includes the following options:
 - a. ASARENT: The persistent publication delivery is based on the settings that are defined for the closest parent in the topic tree.
 - b. ALL: The publication must be delivered to all subscription queues or not at all. If a subscription queue is not usable, the following steps are taken:
 - i. If a DLQ is available, the publication is delivered there.
 - ii. If no DLQ is available, the publication fails.
 - c. ALLAVAIL: The publication is delivered to subscribers that are available. This setting is used on the root topic, "SYSTEM.BASE.TOPIC". If the defined subscription queue is not usable, the following steps are taken:
 - i. If a DLQ is available, the publication is delivered there.
 - ii. If no DLQ is available, the publication fails.
 - d. ALLDUR: The publication must be delivered to all durable subscribers. If the defined subscription queue is not usable, the following steps are taken:
 - i. If a DLQ is available, the publication is delivered to that queue.
 - ii. If no DLQ is available, the publication fails.
2. NPMSGDLV: Non-persistent publication delivery. The following values and actions are available:
 - a. ASARENT: The non-persistent publication delivery is based on the settings that are defined for the closest parent in the topic tree.
 - b. ALL: The publication must be delivered to all subscribers or not at all. If a subscription queue is not usable, the following steps are taken:
 - i. If a DLQ is available, the publication is delivered to that queue.
 - ii. If no DLQ is available, the publication fails.
 - c. ALLAVAIL: The publication is delivered to subscribers that feature a usable subscription queue. This setting is used on the root topic, "SYSTEM.BASE.TOPIC". If a subscription queue is not usable, the following steps are taken:
 - i. If a DLQ is available, the publication is delivered there.
 - ii. If no DLQ is available, the publication is not delivered.
 - d. ALLDUR: The publication must be delivered to all durable subscribers. If the defined subscription queue is not usable, the following steps are taken:
 - i. If a DLQ is available, the publication is delivered to that queue.
 - ii. If no DLQ is available, the publication fails.

WebSphere MQ V7.1 implemented more granular control of DLQ, via a new attribute, USEDLQ.

9.3 The USEDLO attribute

The new attribute, USEDLO, was added to WebSphere MQ V7.1 channel and topic definitions. This attribute provides more granular control over the queue manager and channel actions when there are undeliverable messages, whether they are point-to-point or publications.

It is important to remember that the decision process and behavior that described here remains unchanged; the use of the USEDLO attribute changes the decision point.

9.3.1 USEDLO channel attribute

The USEDLO attribute is supported on the following message delivery channel types:

- ▶ Sender
- ▶ Server
- ▶ Receiver
- ▶ Requestor
- ▶ Cluster-sender
- ▶ Cluster-receiver

The attribute includes the following values:

- ▶ YES: The default value. Undeliverable messages that are coming across this channel are treated as they are in releases older than V7.1.
- ▶ NO: Undeliverable messages that are coming across are treated as though there is no DLQ, as described in “Channel message delivery” on page 134.

Important The USEDLO attribute is not negotiated between channel pairs. The value from the closest definition is used. If there is a requirement to not use a DLQ for a particular channel, set the USEDLO attribute to NO on the sender and receiver channels.

9.3.2 USEDLO topic attribute

The USEDLO attribute is supported on topic definitions and is used to define the behavior for undeliverable publications. For topics, the attribute includes the following valid values:

- ▶ ASPARENT: The attribute inherits the value from its closest administered parent in the topic tree, which is the default value for new topics. The root topic, “SYSTEM.BASE.TOPIC”, is defined by using YES.
- ▶ YES: Undeliverable publications are delivered to the DLQ if one is defined on the queue manager. If there is no DLQ, the behavior is directed by the message delivery attributes PMSGDLV and NPMSGDLV, as described in “Publication delivery for topics” on page 135.
- ▶ NO: Undeliverable publications are treated as though there is no DLQ. The actions are the same as described in “Publication delivery for topics” on page 135.

9.4 Defining USEDLQ on channels

The USEDLQ attribute for channels can be set by using the following techniques:

- ▶ WebSphere WebSphere MQ Explorer
- ▶ MQSC commands
- ▶ ISPF panels
- ▶ PCF commands

This section describes the process that is used to set the attribute for channels by using the first three techniques.

9.4.1 Setting USEDLQ on channels by using WebSphere MQ Explorer

The following procedure uses WebSphere MQ Explorer on Windows to configure the attribute USEDLQ on channels on local or remote queue manager:

1. At WebSphere MQ Explorer- Navigator, click **Queue Managers** → <your queue manager> → **Channels**, as shown in Figure 9-1.

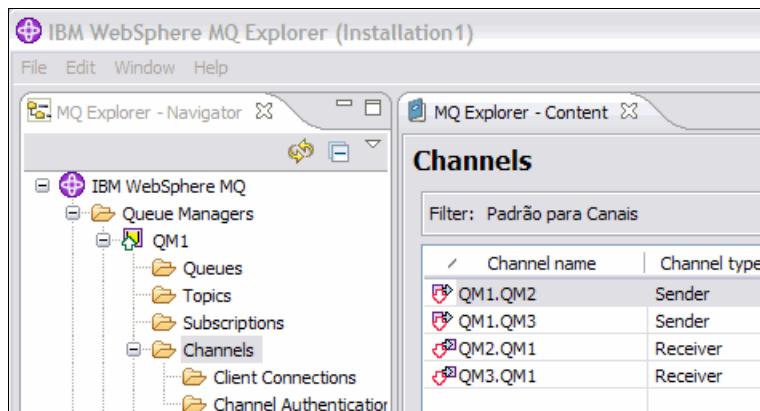


Figure 9-1 Channel view

2. Right-click <your channel> and select **Properties**, as show in Figure 9-2.

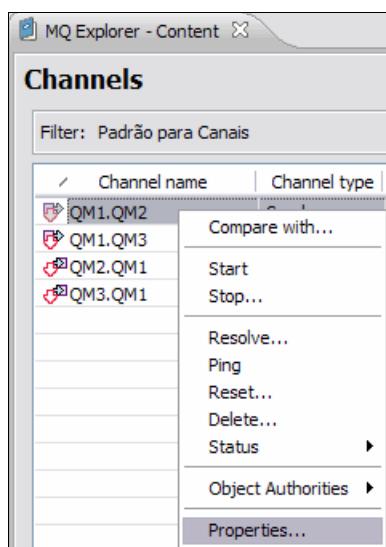


Figure 9-2 Channels Properties

3. Click **Extended**, as show in Figure 9-3.

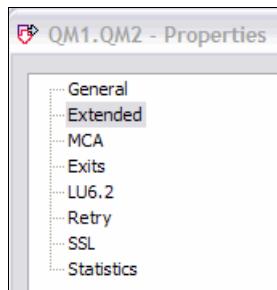


Figure 9-3 Channel properties menu

4. At the Use dead-letter queue attribute, select the value, as show in Figure 9-4.

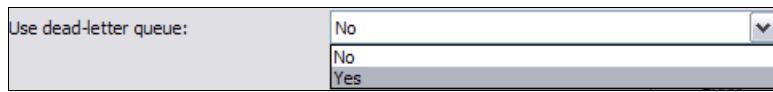


Figure 9-4 Channel use dead-letter queue attribute

5. Click **Apply**.

9.4.2 Setting USEDLQ on channels by using MQSC commands

The following commands provide simple examples of the use of the MQSC administration interface to configure the attribute on topic objects:

1. Start the MQSC interface by entering the following command:

```
runmqsc queue_manager_name
```

2. Enter the following command:

```
ALTER CHANNEL(QM1.QM2) CHLTYP(SDR) USEDLQ(YES)
```

9.4.3 Setting USEDLQ on channels by using ISPF panels

The following procedure uses ISPF panels to configure the attribute USEDLQ on receiver channels:

- As shown in Figure 9-5, at the WebSphere MQ for z/OS - Main Menu panel, complete the fields with the following information:
 - Option: 3 (Alter)
 - Object Type: CHANNEL
 - Name: <your channel>

```
IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action . . . . . . . . . . 3      0. List with filter   4. Ma
                                1. List or Display  5. Pe
                                2. Define like     6. St
                                3. Alter          7. St
                                8. Command
Object type . . . . . . . . CHANNEL +
Name . . . . . . . . . . CSQ5.TO.ITSO WIN
Disposition . . . . . . . . Q Q=Qmgr, C=Copy, P=Private, G=G
                           S=Shared, A>All

Connect name . . . . . . CSQ5 - local queue manager or gr
Target queue manager . . . CSQ5
                           - connected or remote queue manager for command in
Action queue manager . . . CSQ5 - command scope in group
Response wait time . . . . 30 5 - 999 seconds

(C) Copyright IBM Corporation 1993,2011. All rights reserved.
Command ==> F1=Help F2=Split F3=Exit F4=Prompt F9=SwapN
F12=Cancel
```

Figure 9-5 TSO ISPF Panel

2. Press Enter.

3. Browse to the ISPF panel by using PF8 and locate the Use dead-letter queue parameter, as shown in Figure 9-6. Set the parameter to Y to use the queue manager dead letter queue for undeliverable messages; or to N to stop the channel.

```

          Alter a Sender Channel - 3

Press F7 or F8 to see other fields, or Enter to alter channel

Channel name . . . . . : CSQ5.T0.ITSO_WIN
Disposition . . . . . : QMGR      CSQ5

MCA user ID . . . . . .
Nonpersistent messages . . . . E F=Fast, N=Normal
Maximum message length . . . . 4194304          0 - 104857600
Batch size . . . . . : 50           1 - 9999
Sequence number wrap . . . . 9999999999        100 - 9999999999
Heartbeat interval . . . . 300          0 - 99999 second
Keep alive interval . . . . AUTO         0 - 99999 seconds
Monitoring . . . . . : Q Q=Qmgr, L=Low, M=Medium, H=H
Security exit name . . . .
User data . . . . .
Use dead-letter queue . . . . Y Y=Yes, N=No
Last alteration time . . . : 2012-10-17 13.45.52

Command ==> _____
F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd
F10=Messages F11=Status   F12=Cancel

```

Figure 9-6 ISPF Panel with USEDLQ parameter for channel

9.5 Defining USEDLQ on topics

The USEDLQ attribute for topics can be set by using the following techniques:

- ▶ WebSphere WebSphere MQ Explorer
- ▶ MQSC commands
- ▶ PCF commands

This section describes the process that is used to set the attribute for channels by using the first two techniques.

9.5.1 Setting USEDLQ on topics by using WebSphere MQ Explorer

The following procedure uses WebSphere MQ Explorer on the Windows platform to configure the attribute USEDLQ on topics on local or remote queue manager:

1. At WebSphere MQ Explorer - Navigator, click **Queue Managers** → <your queue manager> → **Topics**, as show in Figure 9-7.

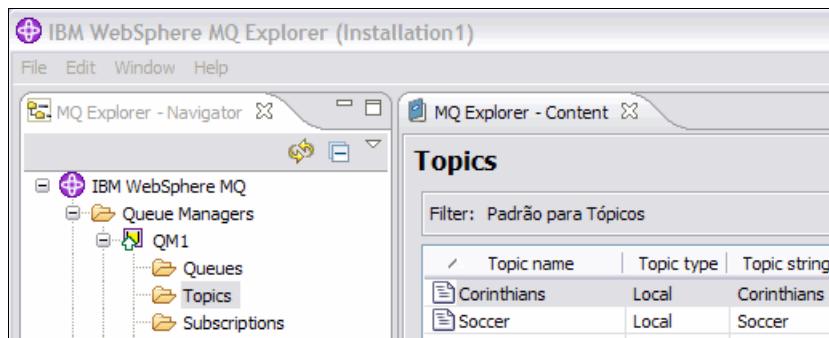


Figure 9-7 Topics view

2. Right-click <your topic> and select **Properties**, as show in Figure 9-8.

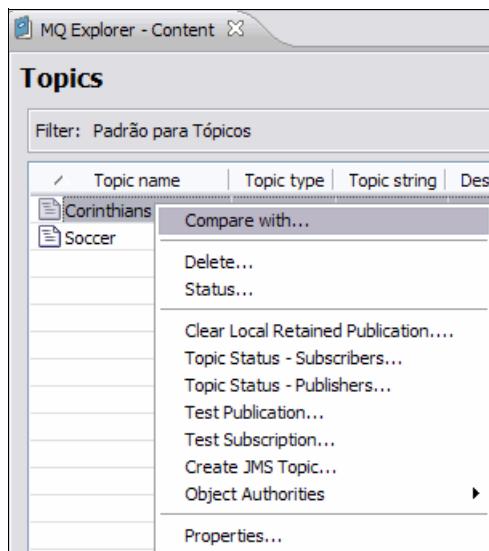


Figure 9-8 Topics properties

3. Click **General**, as show in Figure 9-9

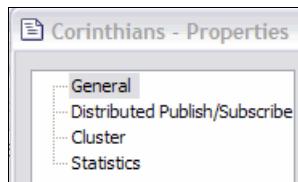


Figure 9-9 Topics properties menu

4. At the Use dead-letter queue attribute, select the value, as show in Figure 9-10.

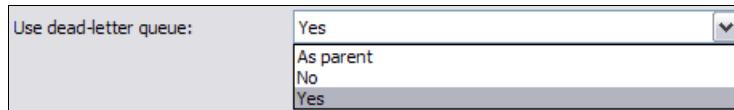


Figure 9-10 Topic use dead-letter queue attribute

5. Click **Apply**.

9.5.2 Setting USEDLQ on topics by using MQSC commands

The following commands provide simple examples of the use of the MQSC administration interface to configure the attribute on topic objects:

1. Start the MQSC interface by entering the following command:

```
runmqsc queue_manager_name
```

2. Enter the following command:

```
ALTER TOPIC(SOCCER) USEDLQ(YES)
```



Dumping and restoring a queue manager configuration

The WebSphere MQ queue manager configuration data is important for restoring a system after hardware failures that result in an unrecoverable WebSphere MQ environment. This chapter describes how the ability to save the queue manager configuration is integrated into WebSphere MQ V7.1 on Windows, UNIX, Linux, and IBM i.

This functionality replaces the content that is available in SupportPac MS03, which is also known as the **saveqmgr** command.

This chapter contains the following sections:

- ▶ Dumping a queue manager
- ▶ Restoring a queue manager configuration

10.1 Dumping a queue manager

The queue manager configuration data includes the information that is required to rebuild the queue manager resources and the access to those resources. WebSphere MQ V7.1 introduces the commands **dmpmqcfg** on Windows, UNIX, and Linux, and **DMPMQMCFG** on IBM i to dump the queue manager configuration data to different formats that can be reused to rebuild a duplicate environment.

In addition to dumping the queue manager configuration for subsequent rebuilding, the information that is dumped can be used to track changes to a queue manager's definitions and create templates to replica queue managers.

The introduction of this functionality on distributed platforms is aligned with the existing **MAKEDEF** command that already is available on z/OS.

The command **dmpmqcfg** takes the following options:

- ▶ **-m <QMgrName>**

The name of the queue manager to which **dmpmqcfg** connects. If the **-r** flag is not specified, this option also represents the queue manager to have its configuration dumped.

- ▶ **-n <ObjName>**

The name of the WebSphere MQ object to be dumped. The name can be specified exactly, or generically by using a single wildcard

- ▶ **-t <ObjType>**

The type of object to be dumped

- ▶ **-x <ExportType>**

This option can be used to filter the output to specific subtype objects

- ▶ **-o <Format>**

The type of output to be generated. This output is MQSC commands or **setmqaut** commands

- ▶ **-a**

This option is used to ensure that the output contains all attributes for all objects that are dumped rather than just the attributes that differ from the default values

- ▶ **-z**

The command can generate warnings that do not cause failure to dump. This option, if specified suppresses those warnings.

- ▶ **-c <MQSC DEF CHL CMD>**

The command can be used to connect to a remote queue manager over a client connection. This option is specified either as the keyword **default** or by an MQSC command of the form, **DEF CHL(<CHLNAME>) CHLTYPE(CLNTCONN) CONNNAME(<IPAddress(PORT)>)**

Using **dmpmqcfg** with the **-c** flag connects to the target queue manager as a client application. In this mode, **dmpmqcfg** can connect to a queue manager at any version of WebSphere MQ, and additionally can also connect to, and dump the configuration of a z/OS queue manager.

Example 10-1 shows a sample command and the subsequent output from connecting to a z/OS queue manager and dumping its topic information.

Example 10-1 Dumping a z/OS queue manager configuration by using a client connection

```
dmpmqcfg -m myQMgr -c "DEF CHL(myChannel) CHLTYPE(CLNTCONN)
CONNAME('<IPAddr>(Port)')" -t topic

*****
* Script generated on 2012-10-26 at 10.28.02
* Script generated by user 'myUser' on host 'myMachine'
* Queue manager name: myQMgr
* Queue manager platform: z/OS
* Queue manager command level: (710/750)
* Command issued: dmpmqcfg -m myQMgr -c DEF CHL(mtChannel) CHLTYPE(CLNTCONN)
CONNAME('<IPAddr>(Port)') -t topic
*****

DEFINE TOPIC('SYSTEM.BASE.TOPIC') +
    TOPICSTR('') +
    DEFPRTY(0) +
    DEFPSIST(NO) +
    DURSUB(YES) +
    NPMSGDLV(ALLAVAIL) +
    PMSGDLV(ALLDUR) +
    MDURMDL('SYSTEM.DURABLE.MODEL.QUEUE') +
    MNNDURMDL('SYSTEM.NDURABLE.MODEL.QUEUE') +
    PUB(ENABLED) +
    SUB(ENABLED) +
    PUBSCOPE(ALL) +
    SUBSCOPE(ALL) +
    DESCR('Base topic for resolving attributes') +
    DEFPRESP(SYNC) +
    USEDLQ(YES) +
* ALTDATE(2012-09-17) +
* ALTTIME(10.56.36) +
    REPLACE
DEFINE TOPIC('SYSTEM.BROKER.ADMIN.STREAM') +
    TOPICSTR('SYSTEM.BROKER.ADMIN.STREAM') +
    WILDCARD(BLOCK) +
    DESCR('Admin stream for queued Pub/Sub interface') +
* ALTDATE(2012-09-17) +
* ALTTIME(10.56.36) +
    REPLACE
DEFINE TOPIC('SYSTEM.BROKER.DEFAULT.STREAM') +
    TOPICSTR('') +
    DESCR('Default stream for queued Pub/Sub interface') +
* ALTDATE(2012-09-17) +
* ALTTIME(10.56.36) +
    REPLACE
DEFINE TOPIC('SYSTEM.BROKER.DEFAULT.SUBPOINT') +
    TOPICSTR('') +
    DESCR('Default RFH2 subscription point for queued Pub/Sub interface') +
* ALTDATE(2012-09-17) +
* ALTTIME(10.56.36) +
    REPLACE
DEFINE TOPIC('SYSTEM.DEFAULT.TOPIC') +
```

```
TOPICSTR('') +
* ALTDATE(2012-09-17) +
* ALTTIME(10.56.36) +
REPLACE
```

In Example 10-1, the queue manager command level is displayed as (710/750). The first number (in this case, 710) represents the WebSphere MQ version of the queue manager (V7.1). The second number (750) represents the version number of the **dmpmqcfg** command (V7.5). This example demonstrates the ability to connect to a queue manager by using a client connection. It also demonstrates that the target queue manager can be at a version earlier than the version of the command.

The **dmpmqcfg** command also can be used in queued mode. This mode is such that a remote queue manager's configuration can be dumped locally without needing to run the **dmpmqcfg** command on the remote machine. To operate in this mode, the local and remote queue managers must be connected by a message channel pair such that data can flow in both directions. The following options are used in queued mode:

► **-q <Rp1QName>**

The name of the queue on the remote queue manager onto which the reply to the **dmpmqcfg** command is put. This option must resolve to a remote queue definition that results in the reply message returning to the queue manager that initiated the dump request.

► **-r <RmtQMgrName>**

The name of the remote queue manager that is having its configuration dumped. This name often is the name of the transmission queue that is defined locally and used to send the request over to the remote queue manager. The name of the queue should match the name of the remote queue manager.

► **-s <MsgSeqNo>**

If the message channels that are used for the **dmpmqcfg** command also are used for normal message traffic, this option can be used to ensure that the sequence numbers on the channels remain synchronized for normal operation.

For more information about the syntax and keywords that can be used with the command options, see the topic *dmpmqcfg* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/ia_DM_PMQMCFG.htm

The IBM i command DMPMQMCFG accepts the same set of options as the **dmpmqcfg** command. For more information about the spellings of the options for IBM i, see the topic *Dump MQ Configuration (DMPMQMCFG)* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/ia_DM_PMQMCFG.htm

This section describes the usage of the commands from the point of view of WebSphere MQ objects and the authorizations that are configured to access those objects.

10.1.1 Dumping WebSphere MQ objects

For this section, WebSphere MQ objects are defined as the resources that are configured on a queue manager, such as queues, topics, channel authentication records, and listeners. The combination of the **-t** and **-x** command options that are described in the Information Center topic **dmpmqcfg** detail the full set of objects that can be dumped.

The most typical usage of the **dmpmqcfg** command is to dump all objects. If the command is run without redirecting the output to a file, the output is written to the console. Running the following command dumps all objects in MQSC command format to a file called **myQM_DUMP.out**, for a queue manager called myQM:

```
dmpmqcfg -m myQM > myQM_DUMP.out
```

Important: The queue manager must be running to run **dmpmqcfg**. Attempts to run the **dmpmqcfg** command or to dump the configuration for a queue manager that is not running fail with the following error message:

```
AMQ8146: WebSphere MQ queue manager not available.
```

The following command can be used to dump only the channel authentication records that are defined on the queue manager to the console:

```
dmpmqcfg -m myQM -x chlauth
```

Example 10-2 shows the output from the **dmpmqcfg** command, when channel authentication records only are dumped.

Example 10-2 Dumping the channel authentication records defined on a queue manager

```
*****
* Script generated on 2012-10-19 at 12.03.15
* Script generated by user 'myUser' on host 'myMachine'
* Queue manager name: myQM
* Queue manager platform: Windows
* Queue manager command level: (710/710)
* Command issued: dmpmqcfg -m myQM -x chlauth
*****
SET CHLAUTH('SYSTEM.ADMIN.SVRCONN') +
    TYPE(ADDRESSMAP) +
    DESCR('Default rule to allow MQ Explorer access') +
    ADDRESS('*') +
    USERSRC(CHANNEL) +
* ALTDATE(2012-10-19) +
* ALTTIME(12.02.34) +
    ACTION(REPLACE)
SET CHLAUTH('SYSTEM.*') +
    TYPE(ADDRESSMAP) +
    DESCR('Default rule to disable all SYSTEM channels') +
    ADDRESS('*') +
    USERSRC(NOACCESS) +
* ALTDATE(2012-10-19) +
* ALTTIME(12.02.34) +
    ACTION(REPLACE)
SET CHLAUTH('*') +
    TYPE(BLOCKUSER) +
    DESCR('Default rule to disallow privileged users') +
    USERLIST('*MQADMIN') +
* ALTDATE(2012-10-19) +
* ALTTIME(12.02.34) +
    ACTION(REPLACE)
```

The output in Example 10-2 on page 148 shows several lines that are prefixed with an asterisk (*). When this output is used to restore a queue manager configuration, these lines are ignored. They are dumped as informational fields only, and cannot be set when the objects are re-created.

An alternative to the output in Example 10-2 on page 148 is to have each MQSC command printed on a single line. The following command dumps the durable subscriptions that are defined on the queue manager myQM by using the single-line MQSC format:

```
dmpmqcfg -m myQM -x sub -o 1line
```

Example 10-3 shows the output from this command.

Example 10-3 Dumping the durable subscriptions in single-line MQSC format

```
*****
* Script generated on 2012-10-19    at 12.13.32
* Script generated by user 'myUser' on host 'myMachine'
* Queue manager name: myQM
* Queue manager platform: Windows
* Queue manager command level: (710/710)
* Command issued: dmpmqcfg -m myQM -x sub -o 1line
*****
DEFINE SUB('SYSTEM.DEFAULT.SUB') TOPICSTR('') *SUBTYPE(ADMIN) *CRDATE(2012-10-19)
*CRTIME(12:02:33) *ALTDAT(2012-10-19) *ALTTIME(12:02:33) REPLACE
```

10.1.2 Dumping WebSphere MQ object access authorizations

In addition to the objects, the access authorities to those objects also must be configured on the rebuilt queue manager. The **dmpmqcfg** command provides the ability to dump the object authority manager (OAM) configuration. For more information about the OAM, see the topic *Object authority manager (OAM)* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/fa16420_.htm

When a queue manager configuration is restored and the authorities to use the resources are restored completely, the full queue manager configuration is required. The following command dumps the entire access authority configuration for a queue manager called myQM into a file called myQM_AUTH.out in the **setmqaut** OAM command format:

dmpmqcfg -m myQM -x authrec -o setmqaut > myQM_AUTH.out

The **-o setmqaut** option can be used only with the **-x authrec** option because all other options specify object types rather than authority definitions. The **-x authrec** option can be left out because it is implied with the **-o setmqaut** option. Therefore, the following command is equivalent to the previous command:

dmpmqcfg -m myQM -o setmqaut > myQM_AUTH.out

The authorities can be filtered based on the object names that they apply to by using the **-n** option. The following command dumps the access authority definitions for any object that is named SYSTEM.DEF.SEND* to the console:

dmpmqcfg -m myQM -x authrec -o setmqaut -n SYSTEM.DEF.SEND*

Example 10-4 shows the sample output of this command.

*Example 10-4 Dumping the access authorities for objects named SYSTEM.DEF.SEND**

```
#####
# Script generated on 2012-10-19      at 13.30.05
# Script generated by user 'myUser' on host 'myMachine'
# Queue manager name: myQM
# Queue manager platform: Windows
# Queue manager command level: (710/710)
# Command issued: dmpmqcfg -m myQM -x authrec -o setmqaut -n SYSTEM.DEF.SEND*
#####
setmqaut -m myQM -t qmgr -p myUser@myMachine +altusr +chg +connect +dlt +dsp +inq
+set +setall +setid +ctrl +system
setmqaut -m myQM -t qmgr -g mqm@myMachine +altusr +chg +connect +dlt +dsp +inq
+set +setall +setid +ctrl +system
setmqaut -m myQM -t qmgr -p myUser@myMachine +crt
setmqaut -m myQM -t qmgr -g mqm@myMachine +crt
# No matching queue objects
# No matching namelist objects
# No matching process objects
setmqaut -m myQM -n SYSTEM.DEF.SENDER -t channel -p myUser@myMachine +chg +dlt
+dsp +ctrl +ctrlx
setmqaut -m myQM -n SYSTEM.DEF.SENDER -t channel -g mqm@myMachine +chg +dlt +dsp
+ctrl +ctrlx
# No matching authinfo objects
# No matching listener objects
# No matching service objects
# No matching comminfo objects
# No matching topic objects
```

The **dmpmqcfg** command identifies in its output where no objects match the specified object name (in this case, a generic name). In addition to the authorities that are required to access the specific objects that match the object name provided, **dmpmqcfg** adds a prefix to the authorities that are required to access the queue manager. Restoring access to queue manager objects is insufficient if access to connect to a queue manager was not restored.

Example 10-5 shows how the output of the following **dmpmqcfg** command was further filtered to specific objects of specific types. In this case, the command dumps the authority definitions for any channel that matches the name SYSTEM.DEF.RE* on queue manager myQM:

```
dmpmqcfg -m myQM -x authrec -o setmqaut -t channel -n SYSTEM.DEF.RE*
```

Example 10-5 Filtering the dmpmqcfg output for specific object types

```
#####
# Script generated on 2012-10-19 at 13.35.18
# Script generated by user 'myUser' on host 'myMachine'
# Queue manager name: myQM
# Queue manager platform: Windows
# Queue manager command level: (710/710)
# Command issued: dmpmqcfg -m myQM -x authrec -o setmqaut -t channel -n
SYSTEM.DEF.RE*
#####
setmqaut -m myQM -n SYSTEM.DEF.RECEIVER -t channel -p myUser@myMachine +chg +dlt
+dsp +ctrl +ctrlx
setmqaut -m myQM -n SYSTEM.DEF.RECEIVER -t channel -g mqm@myMachine +chg +dlt +dsp
+ctrl +ctrlx
setmqaut -m myQM -n SYSTEM.DEF.REQUESTER -t channel -p myUser@myMachine +chg +dlt
+dsp +ctrl +ctrlx
setmqaut -m myQM -n SYSTEM.DEF.REQUESTER -t channel -g mqm@myMachine +chg +dlt
+dsp +ctrl +ctrlx
```

Important: **setmqaut** commands can be duplicated by SET AUTHREC MQSC commands. Specifying **-x authrec** without the **-o setmqaut** option results in the same authority definitions that are generated in MQSC format. SET AUTHREC MQSC commands are applicable only to WebSphere MQ V7.1, or later.

The **dmpmqcfg** command also allows the **grtmqaut** keyword to be specified instead of **setmqaut** with the **-o** option. If the **grtmqaut** keyword is used, the authority definitions are generated so that they can be run against a queue manager on IBM i.

10.2 Restoring a queue manager configuration

Typically, the output from the **dmpmqcfg** command is stored in a file that can be used to rebuild the queue manager; possibly on another machine. This file contains the MQSC or **setmqaut** (or equivalent) commands that need be run to configure the new environment.

10.2.1 Restoring WebSphere MQ objects by using MQSC commands

The output file that is generated from the **dmpmqcfg** command can be used as an input file to the **runmqsc** tool. The syntax for the **runmqsc** tool to process commands within a file is shown in the following example:

```
runmqsc QMgrName < dmpmqcfg.output
```

An example of the use of the **dmpmqcfg** command to dump the channel authentication records for a queue manager called myQM is shown in 10.1.1, “Dumping WebSphere MQ objects” on page 147. The object definitions are output to a file called **myQM_CHLAUTH.out**. The following example uses the commands in the file **myQM_CHLAUTH.out** to re-create the same channel authentication records on a new instance of myQM:

```
runmqsc myQM < myQM_CHLAUTH.out
```

Example 10-6 shows the restoration of the channel authentication records that were created on the original queue manager.

Example 10-6 Restoring the channel authentication records previously dumped by dmpmqcfg

```
Starting MQSC for queue manager myQM.
```

```
:  
*****:  
: * Script generated on 2012-10-19 at 14.24.03  
: * Script generated by user 'myUser' on host 'myMachine'  
: * Queue manager name: myQM  
: * Queue manager platform: Windows  
: * Queue manager command level: (710/710)  
: * Command issued: dmpmqcfg -m myQM -x chlauth  
:  
*****:  
1 : SET CHLAUTH('MY_CHL') +  
:     TYPE(QMGRMAP) +  
:     QMNAME('REMOTE_QM') +  
:     MCAUSER('myUser') +  
:     USERSRC(MAP) +  
:     * ALTDATE(2012-10-19) +  
:     * ALTTIME(14.16.57) +  
:     ACTION(REPLACE)  
AMQ8877: WebSphere MQ channel authentication record set.  
2 : SET CHLAUTH('SYSTEM.ADMIN.SVRCONN') +  
:     TYPE(ADDRESSMAP) +  
:     DESCRIPTOR('Default rule to allow MQ Explorer access') +  
:     ADDRESS('*') +  
:     USERSRC(CHANNEL) +  
:     * ALTDATE(2012-10-19) +  
:     * ALTTIME(14.16.57) +  
:     ACTION(REPLACE)  
AMQ8877: WebSphere MQ channel authentication record set.
```

```

3 : SET CHLAUTH('SYSTEM.*') +
:   TYPE(ADDRESSMAP) +
:   DESC('Default rule to disable all SYSTEM channels') +
:   ADDRESS('*') +
:   USERSRC(NOACCESS) +
: * ALTDATE(2012-10-19) +
: * ALTTIME(14.16.57) +
:   ACTION(REPLACE)
AMQ8877: WebSphere MQ channel authentication record set.
4 : SET CHLAUTH('*') +
:   TYPE(BLOCKUSER) +
:   DESC('Default rule to disallow privileged users') +
:   USERLIST('*MQADMIN') +
: * ALTDATE(2012-10-19) +
: * ALTTIME(14.16.57) +
:   ACTION(REPLACE)
AMQ8877: WebSphere MQ channel authentication record set.
4 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.

```

10.2.2 Restoring WebSphere MQ object access authorities

Using the **-o setmqaut** option on the **dmpmqcfg** command results in object authority definitions are written out as OAM commands that can be run from the operating system command prompt.

The **setmqaut** commands (or **grtmqaut** commands for IBM i) can be placed into a script for the target platform, or run manually one by one from an interactive command prompt session.



Shared message data set and message offloading (z/OS)

WebSphere MQ V6 introduced support for messages on shared queues that were longer than the 63 K maximum that can be stored in a coupling facility. Before version 7.1, messages larger than 63 K were stored in a shared DB2 table and the message body size was fixed at 63 K.

WebSphere MQ version 7.1 introduces two new concepts that are covered in this chapter: shared message data sets (SMDS) and message body offloading rules.

The chapter contains the following sections:

- ▶ Shared message data sets
- ▶ Offload rules
- ▶ Other SMDS commands
- ▶ Monitoring
- ▶ Recovery
- ▶ Application impact

Important: DB2 data sharing is still required for WebSphere MQ queue-sharing groups. Shared message data sets are for message body storage only. DB2 is still used for the configuration and management of a queue-sharing group.

11.1 Shared message data sets

WebSphere MQ supports queue-sharing groups and allows WebSphere MQ to keep messages available to all queue managers in the group within a parallel sysplex. WebSphere MQ evolved from the original limited support of only non-persistent messages that were smaller than 63 K, to full support for all message sizes and types.

Messages that are larger than 63 K require special handling by the queue managers. They were not initially supported on shared queues because of the architecture of the coupling facility (CF) list structures. In the CF, each message is stored in two components: the list structure element and the associated entries. Each element is the anchor point for a message and supports as many 256-byte entries as are needed to hold the message up to the maximum size of 63 K.

Figure 11-1 shows how messages are stored within a CF structure.



Figure 11-1 WebSphere MQ Message structure in a CF

11.1.1 Support for large messages on shared queues

Support for messages that are larger than 63 K was introduced with WebSphere MQ V6. Messages that were over the limit were shared by breaking the message into two components. The message header and control information is kept in the CF and the message body is stored in a special DB2 table called CSQ.ADMIN_B_MESSAGES.

Figure 11-2 shows the relationship between the coupling facility list structures and the message bodies that are stored in DB2.

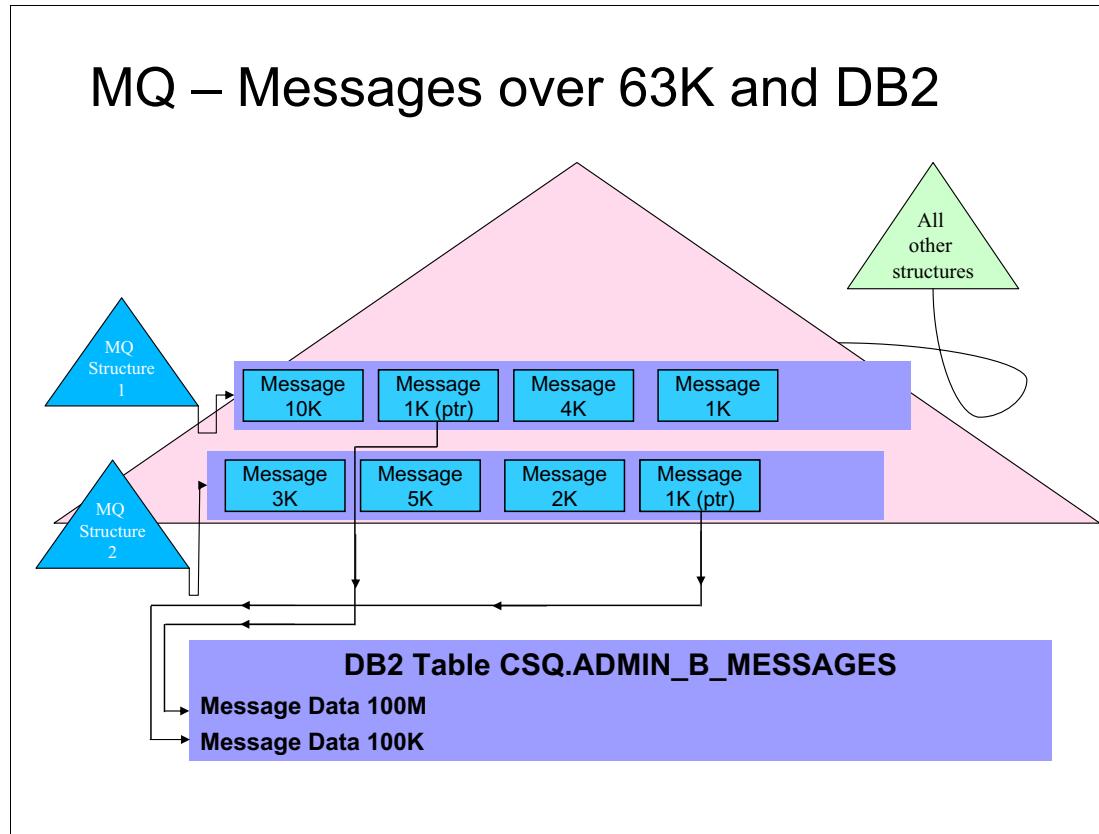


Figure 11-2 Large shared message storage that uses DB2

This technique works well when the number of messages that is over the 63-K limit is small. The typical recommendation is to keep the number of large messages to less than 10% of the message traffic. The effect of storing the message bodies in DB2 is relatively expensive in terms of CPU usage and reduced throughput rates. These costs are documented in SupportPac MP16.

In addition, as messages grow, more internal segmentation is done, which affects the cost per message and the throughput rates. As noted in the IBM abstract *Capacity Planning and Tuning for WebSphere MQ for z/OS*:

The queue manager issues all DB2 calls on behalf of any application using >63 KB shared queue messages. For virtual storage reasons the messages are segmented, if necessary, into multiple 512 KB LOB table entries containing 511 KB of message data. This has the following effects:

- ▶ *CPU costs for shared queue messages >63 KB are mostly incurred by the queue manager and DB2 address spaces rather than the application.*
- ▶ *Maximum throughput and response time can be impacted across 511 KB boundaries.*
- ▶ *CPU costs are increased across 511 KB boundaries.*

For more information, see this website:

<http://www-01.ibm.com/support/docview.wss?uid=swg24007421>

With the growth in XML use that accompanied service-oriented architecture adoption, many MQplexes found message sizes exponentially growing. The overall percentage of messages that exceeds the 63-K limit often grew faster than the message volume.

WebSphere MQ V7.1 introduces an alternative message body storage location, shared message data sets (SMDSs). SMDS are Virtual Storage Access Method (VSAM) linear data sets that are used to hold the message body instead of DB2. The location and control information still is in the coupling facility. The body (and at times the message header) is on the data set.

Figure 11-3 shows how messages are stored by using SMDS.

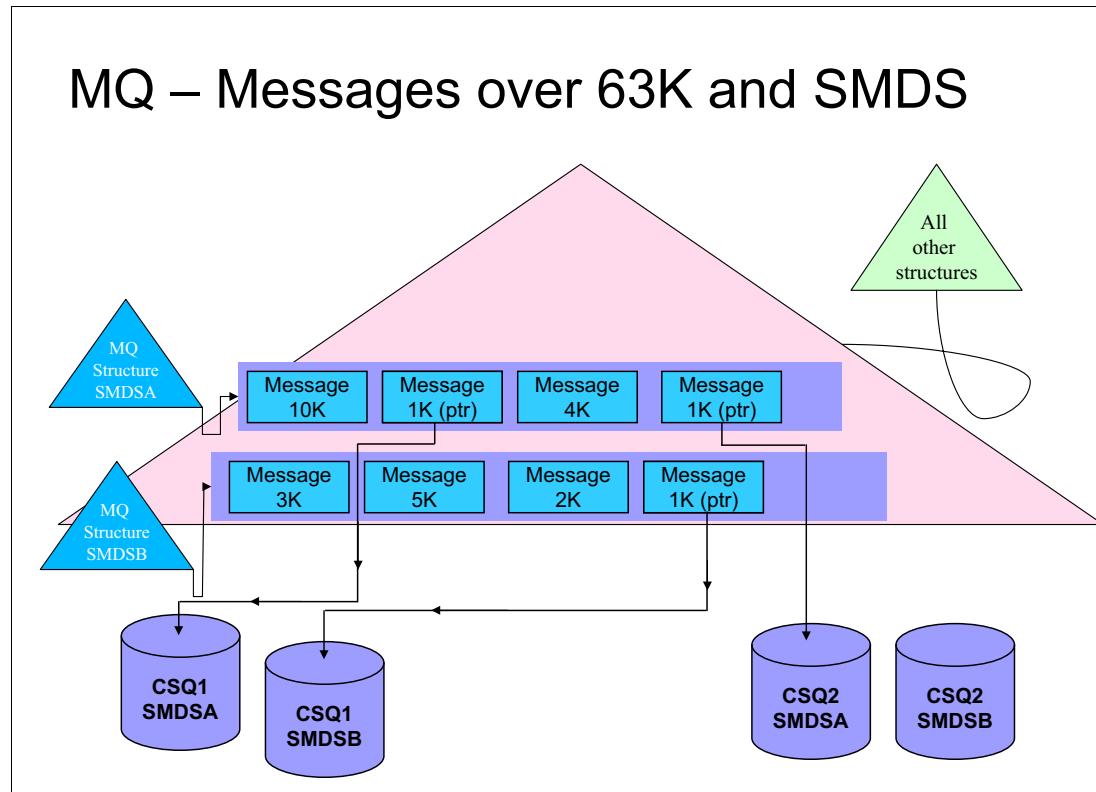


Figure 11-3 Messages over 63K and SMDS

In addition, SMDSs provide a substantial increase in the potential size of a queue. Private queues are limited to 64 Gb and SMDS-hosted queues can get as large as 16 terabytes per queue manager. The message depth that any queue can reach is typically limited by the size of the coupling facility structure because there is at least 1 K per message that is kept there.

11.1.2 Queue manager behavior for large messages

DB2 behavior

The DB2 message body table is a common resource for all queue managers in a queue-sharing group. WebSphere MQ uses the same table, CSQ.ADMIN_B_MESSAGES, for all large message storage.

Figure 11-4 shows message puts and gets with DB2 as the storage mechanism.

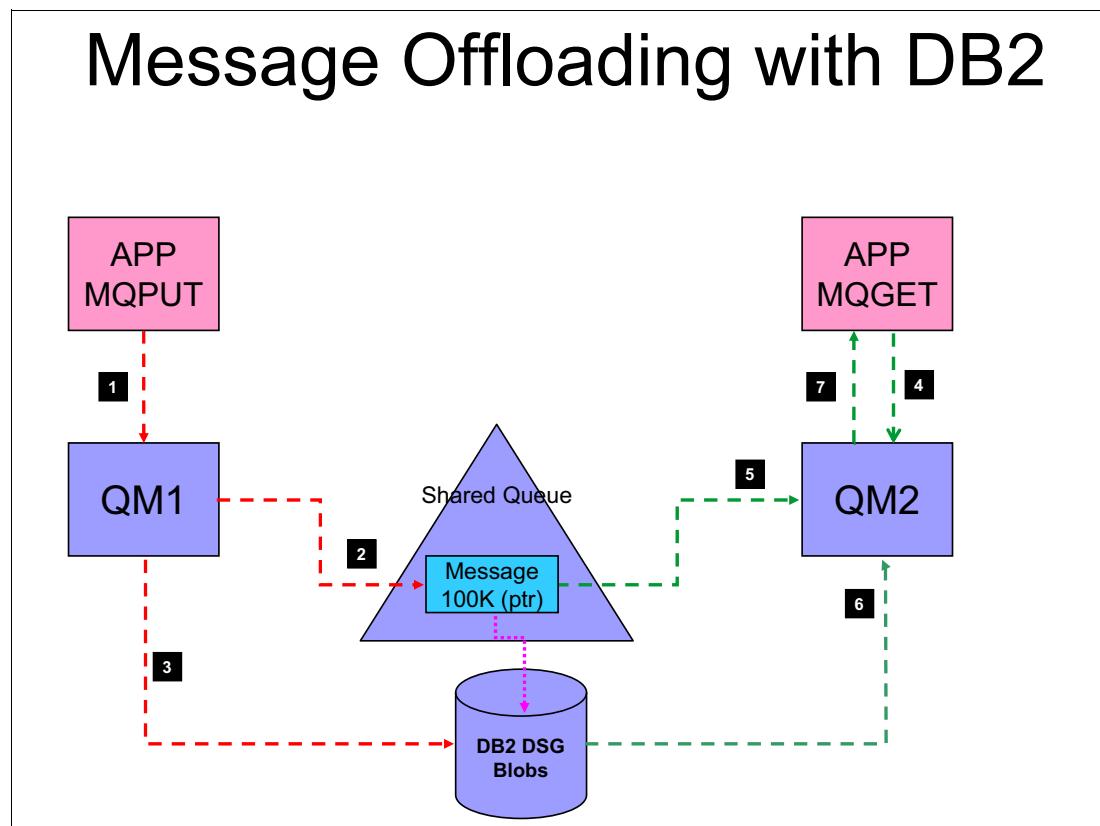


Figure 11-4 Message body offloading with DB2

In Figure 11-4, the following interaction process is used between the application, the queue manager, the coupling facility, and DB2:

1. The sending application performs an MQPUT for a 100K message on a shared queue.
2. The queue manager (QM1) puts the message header and control information in the CF structure.
3. The queue manager puts the entire message body into the messages table. When the message body exceeds 511 K, it is broken up into multiple large objects (LOBs).
4. The retrieving application issues the MQGET for the message.
5. The queue manager (QM2) retrieves the message header and control information from the CF structure.
6. The queue manager retrieves the message body from the messages table.
7. The complete message is passed to the MQGET application.
8. When the MQGET is committed, the entry in the CF and the message body in the table are deleted. (This step is not shown in Figure 11-4 on page 159.)

Shared message data set behavior

The queue manager behavior is different when shared message data sets is used for message body storage. Each queue manager has its own SMDS for each structure that is defined to use this offloading method. The individual queue managers include read/write access to their data set, and all other queue managers in the QSG include read access.

Figure 11-5 shows message puts and gets with SMDS as the storage mechanism for message bodies.

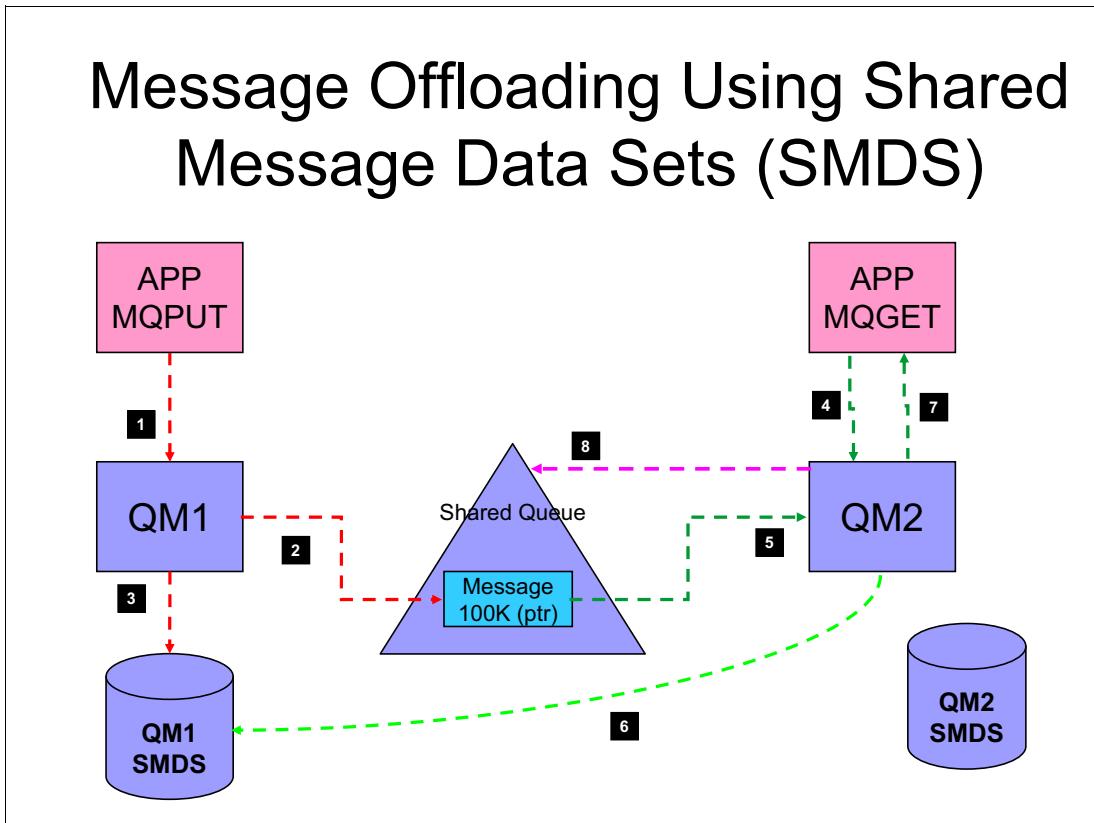


Figure 11-5 Message Body offloading with SMDS

As shown in Figure 11-5 on page 160, the interaction process between the application, the queue manager, the coupling facility, and the shared message data sets occurs as follows:

1. The sending application uses MQPUT for a 100K message on a shared queue.
2. The queue manager (QM1) puts the message header and control information in the CF structure.
3. The queue manager puts the entire message body into the SMDS for the targeted structure that is associated with this queue manager (QM1).
4. The retrieving application issues the MQGET for the message.
5. The queue manager (QM2) retrieves the message header and control information from the CF structure.
6. The queue manager retrieves the message body from the QM1 SMDS for the targeted structure.
7. The complete message is passed to the MQGET application.
8. When the MQGET is committed, the CF entry for the message is put on the deferred delete list for the owning queue manager within the CF structure. When the queue manager list monitoring task is initiated on the owning queue manager (QM1), the CF entry is freed and the shared message data set storage reclaimed.

The use of shared message data sets for message body offloading is much less expensive in terms of CPU usage. It also often provides much higher levels of throughput than DB2. For more information, see WebSphere MQ SupportPac document *MP1H Performance report – WebSphere MQ for z/OS version 7.1.0*, which is available at this website:

<http://www.ibm.com/support/docview.wss?uid=swg24031663>

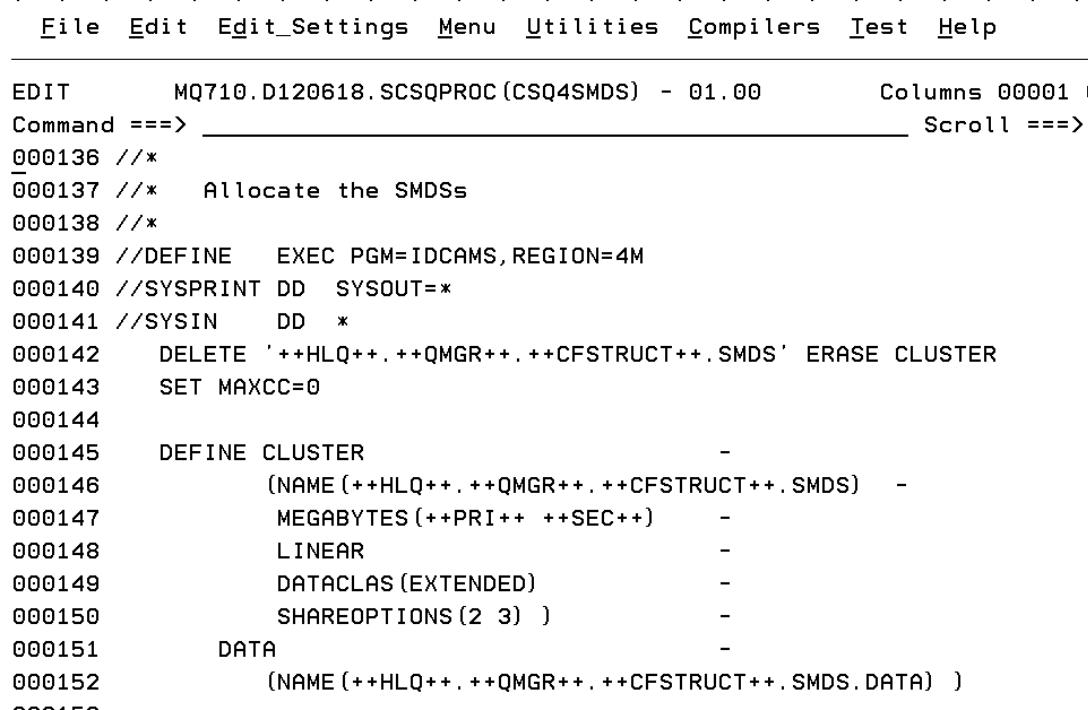
11.1.3 Defining shared message data sets

Shared message data sets must be defined as VSAM files and added to the WebSphere MQ CFSTRUCT object. The steps can be done independently and are described next.

SMDS definition and formatting

The first step in the process to use shared message data sets is defining and formatting the data sets. WebSphere MQ V7.1 provides a new sample job, CSQ4SMDS. The sample must be modified to create and format an SMDS data set for each queue manager in the queue-sharing group.

The sample job includes two steps, the first step is to delete and define the VSAM data set, as shown in Figure 11-6.



The screenshot shows an IBM editor window with the following menu bar: File, Edit, Edit_Settings, Menu, Utilities, Compilers, Test, Help. The main area displays JCL code:

```
EDIT      MQ710.D120618.SCSQPROC(CSQ4SMDS) - 01.00      Columns 00001 0
Command ==> _____ Scroll ==>
000136 /**
000137 /**   Allocate the SMDSs
000138 /*
000139 //DEFINE    EXEC PGM=IDCAMS,REGION=4M
000140 //SYSPRINT DD  SYSOUT=*
000141 //SYSIN    DD *
000142     DELETE '++HLQ++,++QMGR++,++CFSTRUCT++,SMDS' ERASE CLUSTER
000143     SET MAXCC=0
000144
000145     DEFINE CLUSTER
000146         (NAME (++HLQ++,++QMGR++,++CFSTRUCT++,SMDS) -
000147             MEGABYTES (++PRI++ ++SEC++) -
000148             LINEAR -
000149             DATACLAS(EXTENDED) -
000150             SHAREOPTIONS(2 3) )
000151     DATA
000152         (NAME (++HLQ++,++QMGR++,++CFSTRUCT++,SMDS.DATA) )
000153
```

Figure 11-6 Shared message data set delete and define

In the sample, the storage allocation is in megabytes, which is not the typical allocation unit. An important note is that if the data sets are to get larger than 4 G, they must be defined in a storage management system (SMS) data class that specifies extended format. In the sample, the data class is set to DATACLAS(EXTENDED). EXTENDED is an example of a named data class that is set up by a storage management administrator with extended addressability. This name can vary, and there can be multiple classes that support extended format in any environment. The DATACLAS value is not defined as a ‘++’ variable in the sample JCL. The value EXTENDED should be verified with a storage administrator before the attempt is made to define the VSAM data set.

If an invalid data class is specified, the define fails with the message that is shown in Example 11-1.

Example 11-1 Shared message data set definition with undefined dataclas

```
DEFINE CLUSTER(NAME(TEST.SMS.ACDS) -
               LINEAR           -
               DATACLAS(EXTENDED) -
               TRK(15 1)        -
               SHAREOPTIONS(3,3))
IGD01014I data set ALLOCATION REQUEST FAILED -
SPECIFIED DATACLAS EXTENDED DOES NOT EXIST
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12
```

Important: The SHAREOPTIONS attribute must be (2 3). This attribute allows the owning queue manager to have read/write access, and all other queue managers to have read access.

The second step formats the data sets. Formatting the shared message data set uses the same utility that formats the WebSphere MQ logs, as shown in Figure 11-7 on page 163.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      MQ710.D120618.SCSPPROC (CSQ4SMDS) - 01.00      Columns 00001 00072
Command ==> _____ Scroll ==> CSR
000154 /*
000155 //***** Bottom of Data *****
000156 /**
000157 /** Format the SMDS
000158 /**
000159 //FORM      EXEC PGM=CSQJUFMT,COND=(0,NE),REGION=0M
000160 //STEPLIB  DD   DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
000161 //          DD   DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
000162 //SYSUT1   DD   DISP=OLD,DSN=++HLQ++.++QMGR++.++CFSTRUCT++.SMDS
000163 //SYSPRINT DD   SYSOUT=*
000164 /**
***** Bottom of Data *****
```

Figure 11-7 SMDS Format step

It is not a requirement that the SMDS is formatted before first use, but it is highly recommended especially in the case of large data sets. If not formatted initially, then at first use the queue manager that performs that service. This configuration affects the response time of the first request to use the data set, possibly significantly.

Defining the SMDS to the queue-sharing group

The CFSTRUCT definition for V7.1 includes several new attributes. The following attributes are key to defining the use of SMDS:

- ▶ OFFLOAD, which defines whether offloading is allowed for this structure.
- ▶ DSGROUP, which is used to associate the data set with the WebSphere MQ structure definition.

The OFFLOAD attribute includes the following valid values:

- ▶ DB2: The default when the CFSTRUCT is migrated from a previous version, or is set up with a CFLEVEL of 4.
- ▶ SMDS: The default for new CFLEVEL 5 structures. A CFSTRUCT that was migrated to CFLEVEL 5 can be altered to use SMDS.
- ▶ None: As of this writing, this undocumented value is used for a CFSTRUCT at CFLEVEL 3 or 4. It can also be specified on a CFLEVEL 5 structure if no offloading is done.

If a message that is over 63 K is put to a queue on a CFLEVEL 3 structure, the application receives a 2366 reason code (MQRC_WRONG_CF_LEVEL). If a large message is put to queue on a CFLEVEL 4 structure, DB2 is used for the message body offload. The OFFLOAD attribute remains None.

The DSGROUP attribute, called the Generic data set name on the WebSphere MQ Explorer panels, requires a specific name format for the data sets. An asterisk (*) is used as a node (or level) within the DSGROUP to indicate the position of the queue manager name within the data set name. When the data sets are defined and formatted, there is one data set for each queue manager in the queue-sharing group. The queue manager name must be in the same place in each data set.

Figure 11-8 shows the OFFLOAD and DSGROUP attribute for a sample CFSTRUCT definition, called SMDOFF, which is used in this book. For more information about offload rules, see 11.2, “Offload rules” on page 167.

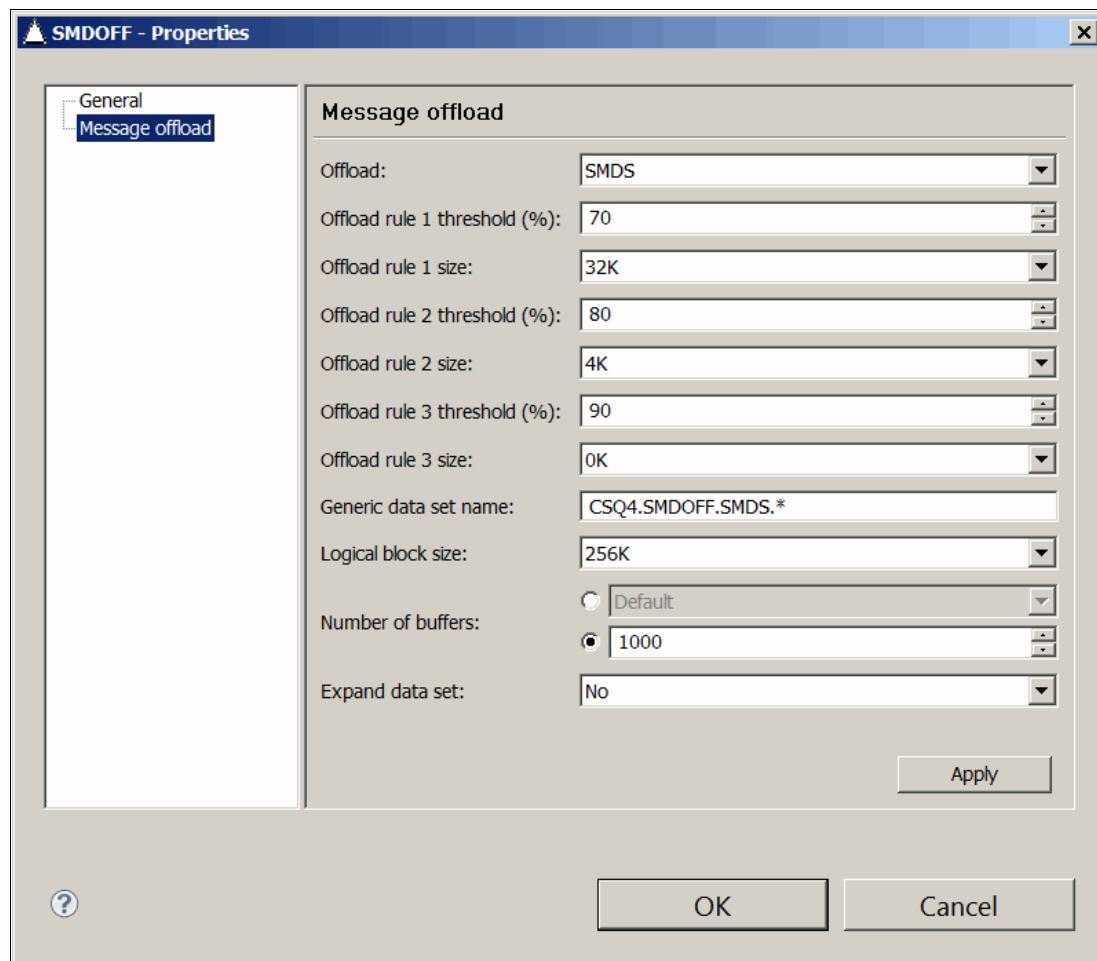


Figure 11-8 The DSGROUP attribute, called ‘Generic data set name’ in the WebSphere MQ Explorer

Important: As of this writing, the WebSphere MQ Information Center indicates that the queue manager name node also might contain other characters. However, this fact is incorrect. In setting up the environment, an attempt was made to combine other characters with the asterisk in the name. Generic names, including ‘IBM1.*.A.SMDOFF’ and ‘IBM1.A*.SMDOFF’, were attempted. Each try attempt failed with an ‘invalid use of an asterisk’ error.

Figure 11-9 shows the data set listing for the SMDOFF CFSTRUCT shared message data set for all the queue managers in the IBM1 queue-sharing group.

DSLIST - Data Sets Matching CSQ4.SMDOFF		Row 1 of 6
Command ==> =		Scroll ==> PAGE
Command - Enter "/" to select action	Message	Volume
CSQ4.SMDOFF.SMDS.CSQ1		*VSAM*
CSQ4.SMDOFF.SMDS.CSQ1.DATA		TOTMQA
CSQ4.SMDOFF.SMDS.CSQ2		*VSAM*
CSQ4.SMDOFF.SMDS.CSQ2.DATA		TOTMQB
CSQ4.SMDOFF.SMDS.CSQ3		*VSAM*
CSQ4.SMDOFF.SMDS.CSQ3.DATA		TOTMQ9

Figure 11-9 SMDS list for the SMDOFF structure

In addition to the OFFLOAD and DSGROUP attributes, the following fields manage the use of the SMDS data. The attribute name that is used in the DEFINE or ALTER CFSTRUCT commands are listed with the corresponding WebSphere MQ Explorer label. The WebSphere MQ Explorer panel where these fields are displayed is shown Figure 11-8 on page 164:

- ▶ **DSBLOCK:** Referenced on the WebSphere MQ Explorer panel as Logical block size, it is space that is allocated in blocks as needed when SMDS offloaded messages are processed.
A larger DSBLOCK size can reduce I/O for large messages, but such sizes increase the basic requirement for space.
- ▶ **DSBUFS:** Called Number of buffers on the WebSphere MQ Explorer panel, it defines the number of buffers that are used to hold messages in above the bar storage.
All buffers are allocated in above the bar (64-bit addressing) storage. This storage, like the private queue buffer pool, improves the response time when the MQPUT and MQGET are local (from the same queue manager).
- ▶ **DSEXPAND:** On the WebSphere MQ Explorer panel, this field is called Expand data set. If this field is set to yes, then the SMDS is allowed to expand when the 90% full threshold is reached.
If no secondary value was set when the SMDS was created, 10% of the existing data set size is used. During the expansion and formatting process, message throughput drops noticeably.

For more information about tuning the buffers, blocks, and the impact of data set expansion, see *SupportPac MP1H, Performance report, WebSphere MQ for z/OS version 7.1.0*, which is available at this website:

<http://www.ibm.com/support/docview.wss?uid=swg24031663>

11.1.4 Migration from DB2

Complete the following steps to migrate a whole structure from DB2 to SMDS:

1. Define and format the shared messages data sets for each queue manager.
2. Migrate the existing CFSTRUCT definition to CFLEVEL(5).
3. Alter the CFSTRUCT definition complete the following steps:
 - a. Set the OFFLOAD parameter to SMDS.
 - b. Set the DSGROUP parameter to the generic data set name.

This migration process is demonstrated in 22.8, “DB2 to SMDS offload migration” on page 370.

Queues within the structure do not have to be empty to change offload targets. The queue manager tracks where the message bodies are offloaded. This process is demonstrated in scenario “Message body storage during migration” on page 374.

Important: Like other WebSphere MQ resources, there are attributes that are associated with the object definition and attributes that are associated with the status. On the CFSTRUCT display, the Offload attribute is what is defined for the structure. The Offload use that is reported on the CFSTATUS is how message bodies are stored. In the Information Center (and other places) the two different terms are often called Offload.

When there are messages on a queue that are defined to a CFSTRUCT that migrated between DB2 and SMDS for message body offloads, the Offload use attribute that is reported by the DISPLAY CFSTATUS command is Both. The WebSphere MQ Explorer pane that is shown in Figure 11-10 shows the Offload use.

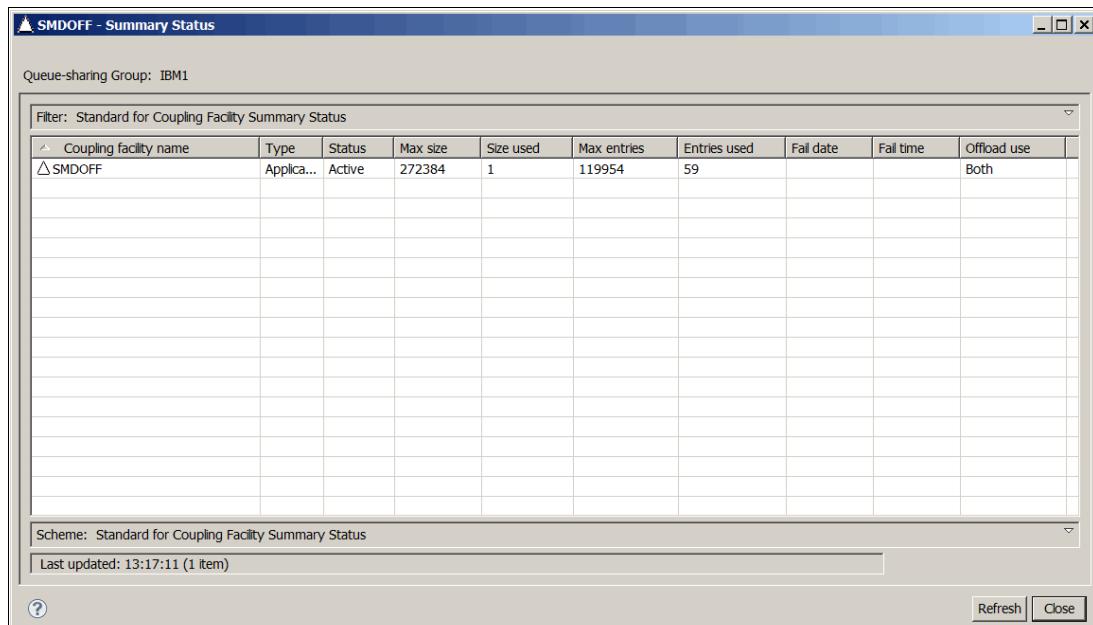


Figure 11-10 CFSTRUCT Summary Status WebSphere MQ Explorer panel

The results of the DISPLAY CFSTATUS command for the same CFSTRUCT are shown in Figure 11-11. The attribute OFFLDUSE is set to BOTH, which indicates that some message bodies are on SMDS and some bodies on DB2.

```
SDSF OUTPUT DISPLAY CSQ1MSTR STC06813 DSID      2 LINE 464      COLUMNS 02- 81
COMMAND INPUT ==>                                         SCROLL ==> CSR
18.34.41 STC06813  CSQM201I -CSQ1 CSQMDRTC  DISPLAY CFSTATUS DETAILS  413
 413          CFSTATUS (DB2OFF)
 413          TYPE (SUMMARY)
 413          CFTYPE (APPL)
 413          STATUS (ACTIVE)
 413          OFFLDUSE (BOTH)
 413          SIZEMAX (272384)
 413          SIZEUSED (1)
 413          ENTSMAX (119954)
 413          ENTSUSED (43)
 413          FAILTIME( )
 413          FAILDATE( )
 413          END CFSTATUS DETAILS
18.34.41 STC06813  CSQ9022I -CSQ1 CSQMDRTC ' DISPLAY CFSTATUS' NORMAL COMPLETION
***** BOTTOM OF DATA *****
```

Figure 11-11 DISPLAY CFSTATUS with mixed message body storage

11.2 Offload rules

Before V7.1, the message body offload size was fixed at 63 K. There were many requests to provide more flexibility. The requests included allowing WebSphere MQ administrators to set the upper limit on the message size to be stored in the CF. The requests also included a sliding scale so that when a coupling facility structure began to fill, the remaining CF could be conserved by putting more of the message body into the offload storage.

These requests were addressed with V7.1 by adding a CFLEVEL value and offload rules to the CFSTRUCT definition. Three rules can be defined for each CFSTRUCT that is at the new CFLEVEL(5). Each rule includes two attributes: the coupling facility structure percentage full and the associated maximum message size.

The rules are simple; when the CF structure reaches the full percentage in the rule, the message body for all messages that are larger than the maximum message size is offloaded to the selected external storage. The rules should be in sequence that is based on the CF full percentage. If the CF full percentage is set to 0, the rule applies to all messages. If the message size is set to 0, then all messages with a length of greater than 122 bytes are offloaded. The 122-byte value could change; it is the number of bytes left in a 1 K (1024 byte) CF page after the message header and control information.

User properties are added to the length of the overall message when the queue manager makes the offload decision. For example, if the offload rule specifies that all message bodies over 2 K are to be offloaded, a message that is 1680 bytes with no user properties is stored in the CF structure. The same size message with the shortest possible user properties('x=y') is offloaded.

11.2.1 Offload definition

The offload definition or changes to the offload type and rules can be made by using the following options:

- ▶ The WebSphere MQ Explorer
- ▶ MQSC commands
- ▶ The command entry panel from the WebSphere MQ ISPF screens
- ▶ PCF commands

Important: The CFSTRUCT ISPF panel does not contain offload options or rules.

If an attempt is made to define an offload type and rule for a CFSTRUCT that is not at CFLEVEL 5, errors occur and they are not always clear. For example, if an attempt is made to add an offload specification to a CFSTRUCT that is at CFLEVEL 3 via the WebSphere MQ Explorer, the error that is shown in Figure 11-12 can be displayed. Other cryptic messages also were reported.

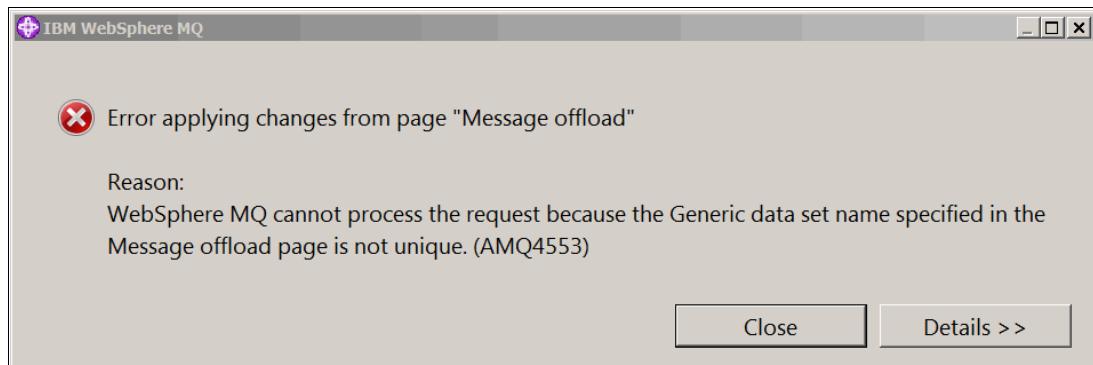


Figure 11-12 Error message when an offload rule is added to a lower-level structure

If the ALTER CFSTRUCT command is entered on z/OS, the message is much clearer, as is shown on Figure 11-13.

```
Display Filter View Print Options Search Help
-----
SDSF OUTPUT DISPLAY CSQ1MSTR STC28105 DSID      2 LINE 1,055  COLUMNS 21- 100
COMMAND INPUT ==> ■
CSQM174E -CSQ1 CSQMACFS 'OFFLOAD' KEYWORD IS NOT ALLOWED WITH 888
CFLEVEL(3) - THIS KEYWORD REQUIRES CFLEVEL(5)
CSQM150I -CSQ1 CSQMACFS 'DSGROUP' AND 'OFFLOAD' VALUES ARE INCOMPATIBLE
CSQM090E -CSQ1 CSQMACFS FAILURE REASON CODE X'00D44004'
CSQ9023E -CSQ1 CSQMACFS 'ALTER CFSTRUCT' ABNORMAL COMPLETION
***** BOTTOM OF DATA *****
```

Figure 11-13 Error message as displayed is z/OS

Offload rules: Offload rules apply to DB2 and SMDS message body offloading.

11.3 Other SMDS commands

The following SMDS-focused commands were added in WebSphere MQ V7.1 to provide more management and inquiry capability:

- ▶ **DISPLAY SMDSCONN:** Displays the availability of the shared message data set for one or more queue managers.
- ▶ **START SMDSCONN:** Starts a previously stopped connection to a shared message data set (or sets).
- ▶ **STOP SMDSCONN:** Stops the connection to a shared message data set (or sets), often to perform maintenance.
- ▶ **RESET SMDS:** Changes the status of the connection to a shared message data set (or sets).

DISPLAY SMDSCONN

The **DISPLAY SMDSCONN** command gives information about the connection status to the SMDS for one or all of the queue managers in the queue-sharing group. In Example 11-2, the request is for the connection status for all of the queue managers. The response, which is shown in Example 11-3, is from the perspective of the queue manager where the command is issued. As of this writing, the shared message data set is not given in the response. This command also is available from the WebSphere MQ Explorer.

Example 11-2 DISPLAY SMDSCONN command

```
-CSQ1 display SMDSCONN(*) CFSTRUCT(DB20FF)
```

Example 11-3 DISPLAY SMDSCONN response

14.14.37 STC06867	CSQM201I -CSQ1 CSQMDRTC	DISPLAY SMDSCONN DETAILS	138
138	SMDSCONN(CSQ1)		
138	CFSTRUCT(DB20FF)		
138	OPENMODE(UPDATE)		
138	STATUS(OPEN)		
138	AVAIL(NORMAL)		
138	EXPANDST(NORMAL)		
138	END SMDSCONN DETAILS		
14.14.37 STC06867	CSQM201I -CSQ1 CSQMDRTC	DISPLAY SMDSCONN DETAILS	139
139	SMDSCONN(CSQ2)		
139	CFSTRUCT(DB20FF)		
139	OPENMODE(READONLY)		
139	STATUS(OPEN)		
139	AVAIL(NORMAL)		
139	EXPANDST(NORMAL)		
139	END SMDSCONN DETAILS		

START SMDSCONN

This command starts, or restarts, the connection to a shared message data set for a specified CFSTRUCT. The **START** command can be for an individual queue manager or for all queue managers in the queue-sharing group and often is used after a **STOP SMDSCONN** to reconnect after maintenance or an error. This command does not include a WebSphere MQ Explorer interface.

For more information, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/sc12345_.htm

STOP SMDSCONN

This command stops the connection to a shared message data set and deallocates it for a specified CFSTRUCT. The **STOP** command can be for an individual queue manager or for all queue managers in the queue-sharing group. This command does not have a WebSphere MQ Explorer interface.

For more information, see this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/sc13445_.htm

RESET SMDS

The **RESET** command gives administrators a way to alter the status or current access of a shared message data set. Often, this command is used during a maintenance cycle of a shared message data set or after an error.

For more information, see:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/sc12845_.htm

11.4 Monitoring

Shared message data sets are similar to any of the resources that are used by WebSphere MQ. They require monitoring and occasional tuning to provide optimal performance. In addition to the normal monitoring that is performed for all queues and resources (such as queue depths and whether there are any processes that are accessing a queue), there are commands that are specifically tailored for shared message data set monitoring.

There also is new System Management Facility (SMF) data that is captured for statistical tracking (SMF 115 data) and task accounting reporting (SMF 116 class 3 data).

11.4.1 DISPLAY CFSTATUS command

The **DISPLAY CFSTATUS** command was updated for WebSphere MQ V7.1 to add the SMDS type. The command is used to monitor the overall status of the coupling facility structure and shows the status of the shared message data sets for one or more queue managers in the queue-sharing group. An example command to display the status of the SMDS defined to every queue manager in the queue-sharing group is shown in Example 11-4

Example 11-4

-CSQ1 DISPLAY CFSTATUS(SMDOFF) TYPE(SMDS)

The response to this command is shown in Figure 11-14. In the response, the SMDS attribute features the name of the queue manager. In this example, there are three queue managers in the queue-sharing group, so an entry is reported for each, though in the example only two are shown. The status can be different for each queue manager.

```
SDSF OUTPUT DISPLAY CSQ1MSTR STC28105 DSID      2 LINE 1,717  COLUMNS 21- 100
COMMAND INPUT ===> ■
SCROLL ===> CSR
CSQ9022I -CSQ1 CSQMDRTC ' DISPLAY CFSTATUS' NORMAL COMPLETION
CSQM293I -CSQ1 CSQMDRTC 3 CFSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I -CSQ1 CSQMDRTC DISPLAY CFSTATUS DETAILS  388
CFSTATUS (SMDOFF)
TYPE (SMDS)
SMDS (CSQ1)
STATUS (ACTIVE)
ACCESS (ENABLED)
RCVTIME (12.38.25)
RCVDATE (2012-10-06)
FAILTIME ( )
FAILDATE ( )
END CFSTATUS DETAILS
CSQM201I -CSQ1 CSQMDRTC DISPLAY CFSTATUS DETAILS  389
CFSTATUS (SMDOFF)
TYPE (SMDS)
SMDS (CSQ2)
STATUS (ACTIVE)
ACCESS (ENABLED)
```

Figure 11-14 *DISPLAY CFSTATUS where the type is SMDS output*

The **DISPLAY CFSTATUS** command for the SMDS type also is supported by the WebSphere MQ Explorer. Complete the following steps to access the information:

1. Right-click **Coupling Facility Structures** in the WebSphere MQ Explorer navigator, as shown in Figure 11-15.

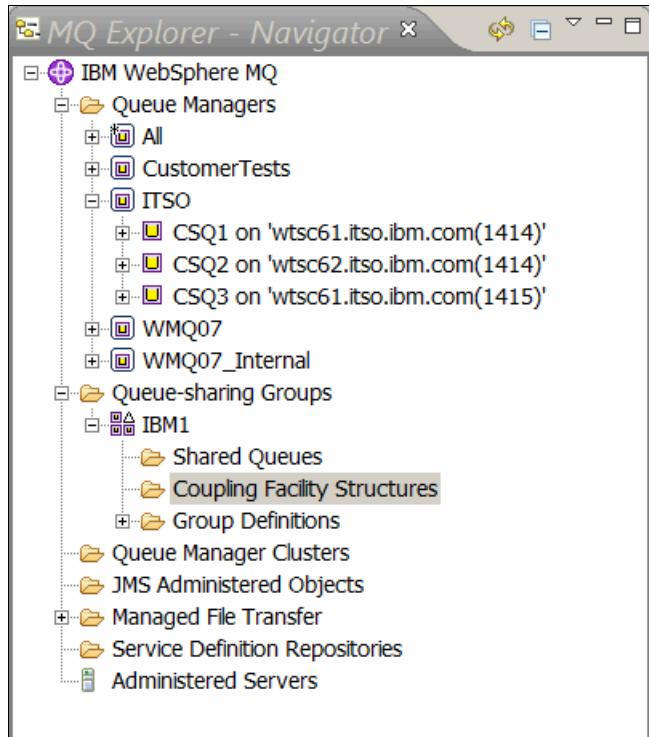


Figure 11-15 Start the display SMDS status from the WebSphere MQ Explorer

2. In the WebSphere MQ Explorer Content window, right-click **CFSTRUCT**. In the example, the structure that is selected is SMDOFF.

3. Select **Status** → **SMDS Status**, as shown in Figure 11-16.

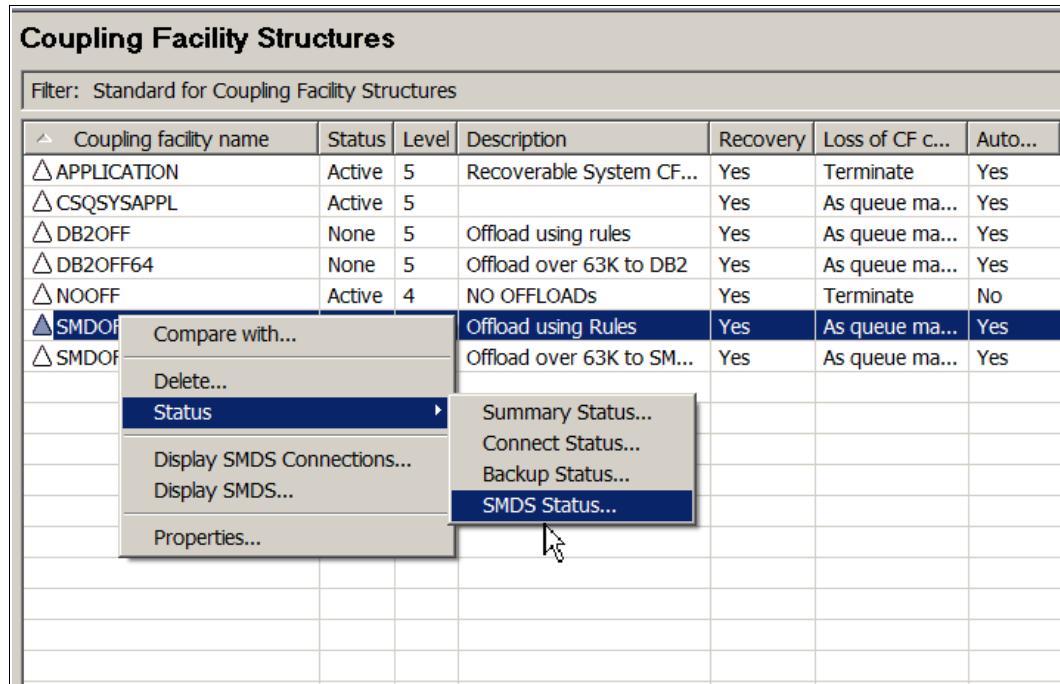


Figure 11-16 WebSphere MQ Explorer SMDS Status selection

4. The SMDS Status panel is displayed, as shown in Figure 11-17.

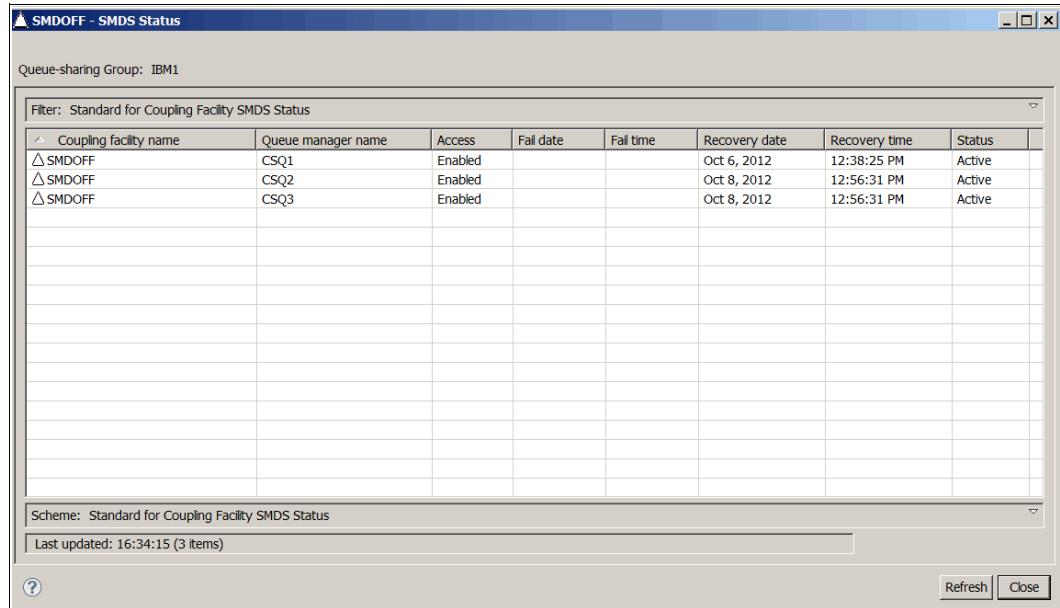


Figure 11-17 WebSphere MQ Explorer SMDS status display

11.4.2 DISPLAY USAGE command

The SMDS type was added to the **DISPLAY USAGE** command in WebSphere MQ V7.1. The response to the command provides information to administrators about the current use of the SMDS data set. Each queue manager reports the use for its SMDS data sets only. Therefore, to get total usage, the command should be issued across all of the queue managers. The command to display the SMDS usage for queue manager CSQ1 is shown in Example 11-5.

Example 11-5 The DISPLAY USAGE command

```
-CSQ1 DISPLAY USAGE TYPE(SMDS)
```

The output of the command looks as shown in Figure 11-18. The queue depth is actually 20, but only 10 of the messages were put from queue manager CSQ1.

```
SDSF OUTPUT DISPLAY CSQ1MSTR STC28105 DSID      2 LINE  COMMAND ISSUED
COMMAND INPUT ===>                                         SCROLL ===> CSR
RESPONSE=SC61
CSQE280I -CSQ1 SMDS usage ...
  Application   Offloaded      Total   Total data   Used data   Used
  structure     messages       blocks    blocks      blocks      part
  _SMDOFF          10           281        280          4         1%
End of SMDS report
```

Figure 11-18 DISPLAY USAGE output

For monitoring, issue the display usage command for SMDS at regular intervals on each queue manager. This command does not have an equivalent command from the WebSphere MQ Explorer.

11.4.3 New messages

Messages to support the use of SMDS were added with WebSphere MQ V7.1. If monitoring software is used to track messages that were issued from queue managers and alert administrators about those messages that are critical, the messages that are listed include a preliminary recommendation for action. In Table 11-1 the Alert, Monitor, or No Action column represents the recommendation at the time of this writing. The standards for alerting and monitoring vary by environment.

Check the system logs: While there might not be a required immediate action, it is always recommended to check the system logs for unexpected events. Monitoring tools are only as good as the alerts that are built into them.

Table 11-1 WebSphere MQ SMDS messages

Message Number	Information	Alert, Monitor, or No action?	Recommended Activity or Response
CSQE280I	SMDS current usage	N	Track the SMDS usage over time.
CSQE285I	SMDS buffer usage	N	Track the SMDS usage over time.
CSQE111I	Structure struct-name are set to failed state to allow recovery of failed SMDS data sets.	M	While informational, if this message is seen frequently, it might indicate a problem in the environment.
CSQM176E	SMDS cannot currently be reset to keyword(value).	N	Investigate whether the request to reset the SMDS status was made correctly.
CSQE244I	SMDS(qmgr-name) CFSTRUCT(struc-name) now has DSEXPAND(value).	M	No response is necessary unless the value is unexpected. The DSEXPAND attribute of the CFSTRUCT definition was altered to the value reported in this message.
CSQE243I	SMDS(qmgr-name) CFSTRUCT(struc-name) now has DSBUFS(value).	N	The DSBUFS attribute of the CFSTRUCT definition was altered to the value that is reported in this message. No action is necessary, unless the value is unexpected.
CSQE276I	The SMDS buffer pool for CFSTRUCT(struc-name) was increased to buffer-count buffers.	N	The buffer count was successfully increased. No action is necessary, unless the value is unexpected.
CSQE274E	The SMDS buffer pool for CFSTRUCT(struc-name) could not be created because sufficient storage was not available.	A	As of this writing, the documented action is to alter the region size to resolve the issue. The region size does not address the problem because the SMDS buffer pools are allocated in above-the-bar storage. Increasing the MEMLIMIT parameter addresses the problem.
CSQE278I	The SMDS buffer pool for CFSTRUCT(struc-name) was decreased to buffer-count buffers.	N	The buffer count was successfully decreased. No response is necessary, unless the value is unexpected.

Message Number	Information	Alert, Monitor, or No action?	Recommended Activity or Response
CSQE241I	SMDS(qmgr-name) CFSTRUCT(struc-name) now includes STATUS(status).	N	The status of the CFSTRUCT was altered to the value that is shown in the message. This value might be a result of a RESET SMDS command.
CSQE242I	SMDS(qmgr-name) CFSTRUCT(struc-name) now includes ACCESS(access).	N	The access of the CFSTRUCT was altered to the value that is shown in the message. This value might be as a result of a RESET SMDS command.
CSQE212I	Formatting is complete for SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname.	N	The formatting of an SMDS data set was completed successfully.
CSQE215I	Further expansion of SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname is not possible because the maximum number of extents were allocated.	M	Applications can receive storage medium full reason codes (2561). There are multiple responses to this situation that depend on the circumstances. If this issue is a problem with the initial size of the SMDS, allocation and formatting of a larger SMDS might be necessary. The code also can indicate a problem with the programs or processes that should be draining the queues. Adding server program instances also might be required.
CSQE234I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname was empty, so it requires formatting.	N	This message can explain response time problems.
CSQE252I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname space map is rebuilt by scanning the structure.	N	This message indicates that normal recovery is taking place.
CSQE235I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname was not fully formatted, so it requires more formatting.	N	There is a delay before the SMDS is available for use while the additional formatting is taking place.
CSQE230E	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname saved space map cannot be used because the time stamp time1 does not match the last CLOSE time stamp time2 in the SMDS object.	M	If it occurs repeatedly, this message indicates a problem with the allocation of the shared message data set.
CSQE213I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname is now percentage% full.	M	This message is issued starting at 90% full and for every 2% change. When the full percentage drops to 88%, the message is no longer issued. If the usage reaches 100%, action should be taken.

Message Number	Information	Alert, Monitor, or No action?	Recommended Activity or Response
CSQE218E	Expansion of SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname was unsuccessful.	A	<p>Immediate action should be taken to resolve the issue to avoid application problems.</p> <p>Usually this indicates that there was no space available for expansion. There also might be messages from SMS that should be investigated.</p>
CSQE211I	Formatting is in progress for count pages in SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname.	N	No alert or monitoring is recommended.
CSQE277I	The SMDS buffer pool for CFSTRUCT(struc-name) was increased to actual-buffers buffers rather than the requested buffer-count because sufficient storage was unavailable.	M	If the number of buffers must be expanded to the value that is requested, investigate why storage might not be available.
CSQE279I	The SMDS buffer pool for CFSTRUCT(struc-name) was decreased to actual-buffers buffers rather than the requested buffer-count because the rest of the buffers are in use.	M	If the number of buffers must be reduced to the value that is requested, repeat the request after the queues are drained.
CSQE255I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname space map was rebuilt, message count msg-count.	N	No alert or action is recommended. Normal processing is resumed.
CSQU558E	QMGR qmgr-name entry cannot be removed from QSG qsg-name, SMDS for structure struc-name is not empty.	A	Before a queue manager can be removed from a queue-sharing group, the shared message data set must be empty. Remove all messages from the SMDS, and the queue manager can be removed.
CSQE222E	Dynamic allocation of SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname failed with return code ret-code, reason code eeeeeeee.	A	<p>Immediate action should be taken to resolve the issue.</p> <p>Usually, this issue is caused by the DSGROUP specification having the asterisk (*) in the incorrect node for the data set.</p>
CSQE223E	Dynamic deallocation of SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname failed with return code ret-code, reason code eeeeeeee.	A	<p>Immediate action should be taken to resolve the issue.</p> <p>The data set probably was flagged as unusable. Investigate the reason code.</p>

Message Number	Information	Alert, Monitor, or No action?	Recommended Activity or Response
CSQE236I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname cannot be used because there is not enough main storage available to build the space map.	A	Immediate action should be taken to resolve the issue. As of this writing, the suggested response in the Information Center is incorrect. Increasing the region size should not have an effect because the storage that is required is above the bar. Increase the MEMLIMIT to add 64-bit storage that can be used.
CSQE237	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname cannot be extended because there is not enough main storage available to build the space map.	A	Immediate action should be taken to resolve the issue. As of this writing, the suggested response in the Information Center is incorrect. Increasing the region size should not have an effect because the storage required is above the bar. Increase the MEMLIMIT to add 64-bit storage that can be used.
CSQE231E	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname cannot be used because it is not a VSAM linear data set with control interval size 4096 and SHAREOPTION (2 3).	A	Immediate action should be taken to resolve the issue. The shared message data set status is set to failed and is unusable. The data set should be reallocated by using the correct attributes.
CSQE219I	Extents refreshed for SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname, count pages added, total pages total.	M	The shared message data set might must be reallocated to a larger size.
CSQE257E	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname is smaller than the size that is recorded in the space map. The saved space map cannot be used.	A	Immediate action should be taken to resolve the issue. There is a mismatch between the saved space map of the shared message data set and what is found.
CSQE233E	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname cannot be used because the header record indicates a newly formatted data set, but it was being used already.	A	Immediate action should be taken to resolve the issue. The data set header indicates that the data set is empty, but the data set was used before. This issue might occur because the data set was reformatted when there was data present. Persistent messages can be recovered by using the RECOVER CFSTRUCT command.

Message Number	Information	Alert, Monitor, or No action?	Recommended Activity or Response
CSQE232E	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname cannot be used because the identification information (field-name) in the header record is incorrect.	A	An alert is recommended on this message. Immediate action should be taken to resolve the issue. The data set header contains unexpected information, which might mean that the data set was corrupted. Persistent messages can be recovered by using the RECOVER CFSTRUCT command.
CSQI034E	Block block-number of the message data for entry id entry-id in CFSTRUCT(struc-name) refers to SMDS(qmgr-id) control interval rci but the stored data does not match the entry ID.	A	Immediate action should be taken to resolve the issue. The identification information about message data that was found in the shared message data set does not match what was found in the coupling facility control information. If the message was persistent, the data set can be recovered by using the RECOVER CFSTRUCT command. If nonpersistent, the informational message CSQI037I is written and the message is discarded.
CSQE238I	SMDS(qmgr-name) CFSTRUCT(struc-name) data set dsname is too small to use because the initial space allocation is less than two logical blocks.	M	This issue should happen only when the shared message data set is accessed for the first time. To repair the problem, reallocate and format the data set with more space.
CSQE245I	CFSTRUCT(struc-name) now includes OFFLDUSE(offload-usage).	M	It notifies the WebSphere MQ administrators that all messages that were offloaded by using a different storage type were removed and committed.
CSQI037I	The nonpersistent message with entry IDentry-id was deleted from CFSTRUCT(struc-name) because the data could not be retrieved.	M	It is a companion of message CSQI034E. It identifies a non-persistent message that was deleted because the message information that was contained in the CF did not match what was in the shared message data set.
CSQM175E	The keyword cannot be altered because a data set is active for this structure.	M	An attempt was made to alter the DSGROUP or DSBLOCK attributes of the CFSTRUCT while the data set was active. Try the ALTER CFSTRUCT again after activity is stopped.
CSQM174E	The keyword is not allowed with CFLEVEL(clevel); this keyword requires CFLEVEL(5).	N	An attempt was made to add OFFLOAD or SMDS attributes to a CFSTRUCT definition that is not at CFLEVEL 5. Migrate the structure to level 5, or correct the structure name before the request is submitted.

11.4.4 New System Management Facility (SMF) data

Important: As of this writing, the Information Center did not contain a description of the shared message data set statistics. Also, the MP1B SupportPac that formats and prints the WebSphere MQ SMF statistics and accounting data does not include the SMDS fields. More information about the WebSphere MQ SMF data and monitoring will be available in future SupportPacs.

New self-defining SMF 115 information was added to report on the SMDS activity. The data layout for Assembler can be found in h1q.SCSQMACS (CSQDQESD). For more information and a complete list of fields and all the changes that were made to the WebSphere MQ SMF data, see Appendix B, “WebSphere MQ for z/OS 7.1 System Management Facility changes” on page 451.

From the QESD information, the following fields were identified for initial monitoring and tracking:

- ▶ **QESDSMMC**

The count of the messages that are currently in the SMDS for this queue manager. This count is not the total count of messages on the queue, only what is stored locally.

- ▶ **QESDSMFL**

The number of times the shared message data set is full and cannot be expanded further. This value must be tracked. If it is ever non-zero, an alert should be sent to the WebSphere MQ administrators. A non-zero value might be due to not allowing data set expansion, the data class not providing for extended addressing, or an application not picking up messages in a timely fashion.

- ▶ **QESDSMMM**

The maximum number of messages that are stored in the SMDS during the SMF interval that is reported. This value should be tracked. If there is an upward trend over time, it can mean that the data set should be expanded or more pulling applications should be added to reduce the likelihood of the data set becoming full.

- ▶ **QESDSMMU**

The maximum number of DSBLOCKs that are used. This value should be tracked. If there is an upward trend, the sizing of the DSBUFS and DSBLOCK attributes might need tuning. For more information, see SupportPac MP1H.

- ▶ **QESDSMMF**

The minimum number of free blocks that are available during this SMF interval. This value should be tracked. If there is a downward trend, the sizing of the DSBUFS and DSBLOCK attributes might need tuning. For more information, see SupportPac MP1H.

- ▶ **QESDBFTO**

The total number of buffers in the pool. This number is from the DSBUFS attribute of the CFSTRUCT object.

- ▶ **QESDBFUS**

The number of shared buffers that are in use during the SMF interval that is reported. This value should be tracked. If there is an upward trend, the sizing of the DSBUFS and DSBLOCK attributes might need tuning. For more information, see SupportPac MP1H.

- ▶ **QESDBFPW**

The number of waits for the pool of buffers. This number should be tracked. If it rises above zero, the WebSphere MQ administrators should investigate the possibility of adding buffers on the CFSTRUCT definition.

- ▶ **QESDBFBW**

The number of times there was a wait for the buffers . This number should be tracked. If it rises above zero, the WebSphere MQ administrators should investigate the possibility of adding buffers on the CFSTRUCT definition.

The following values all pertain to formatting extensions to the shared message data set. The values should be tracked. Although non-zero counts might not be a problem, if the application includes stringent service time requirements, dynamically extending the shared message data set affects that target. If SMDS expansion occurs frequently, the original allocation is too small or the volume of messages that are put on this queue manager is higher than expected. Reallocating the data set might be necessary to improve performance:

- ▶ **QESDIOFR**: The number of format SMDS extensions that are performed during this interval.
- ▶ **QESDIOFP**: The total number of formatted pages.
- ▶ **QESDIOFT**: The total I/O time for the format.
- ▶ **QESDIOFW**: The wait time for the formats that are issued during this SMF interval.

The following values all pertain to the write requests to the shared message data set. They should be tracked over time. Increases in the writes show more use of the SMDS. Increases in values for I/O and wait times can indicate constraints on the data set or physical storage device:

- ▶ **QESDIOWR**: The count of writes to this SMDS during this SMF interval.
- ▶ **QESDIOWP**: The total number of pages written to the SMDS during this interval.
- ▶ **QESDIOWT**: The total I/O time that is spent on the writes for this interval.
- ▶ **QESDIOWW**: The total wait time for the writes performed during this interval.

The following values all pertain to the local read requests to the shared message data set. They should be tracked over time. Increases in the local read counts show more use of the SMDS. Increases in values for I/O and wait times can indicate constraints on the data set or physical storage device:

- ▶ **QESDIORR**: The count of local reads from this SMDS during this SMF interval.
- ▶ **QESDIORP**: The total number of local pages that are read from the SMDS during this interval.
- ▶ **QESDIORT**: The total I/O time spent on the local reads for this interval.
- ▶ **QESDIORW**: The total wait time for the local reads performed during this interval.

The following values all pertain to the remote (other) read requests to shared message data sets. They should be tracked over time. Increases in the read counts show more use of the SMDS. Increases in values for I/O and wait times can indicate constraints on the data set or physical storage device:

- ▶ QESDIOOR: The count of local reads from this SMDS during this SMF interval.
- ▶ QESDIOOP: The total number of local pages that are read from the SMDS during this interval.
- ▶ QESDIOOT: The total I/O time spent on the local reads for this interval.
- ▶ QESDIOOW: The total wait time for the local reads performed during this interval.

11.5 Recovery

As with all WebSphere MQ resources, shared message data sets can be recovered if there is a problem.

During normal queue manager shutdown, information about the state of the SMDS is stored in DB2. When the queue manager is restarted, the information is read from DB2 and the time stamps are verified. If the time stamps are the same, no recovery is needed and work continues as normal.

If the time stamps do not match, then the shared message data set information must be rebuilt by using the following procedure:

1. All message entries are read for the coupling facility structure.
2. If the entry was created by this queue manager and the message body was offloaded to an SMDS, the following steps are performed:
 - a. The control information is read from the entry.
 - b. The SMDS map of message locations is rebuilt based on the control information.
 - c. If the map of messages is rebuilt and no problems are detected, work continues as normal.
3. If there is a problem with the data set or if step 2 fails, persistent messages can be recovered from the WebSphere MQ logs by using the **RECOVER CFSTRUCT** command. The following steps are performed:
 - a. The **RECOVER** command was altered to include rebuilding shared message data sets that have a status of failed. The following steps are performed:
 - i. Shared message data sets can be manually set to failed status by using the **RESET SMDS** command, if necessary.
 - ii. Data sets that are manually set to empty or were replaced with new data sets are recovered.
 - b. The rebuild is from the last **BACKUP CFSTRUCT** command.

For more information about the **BACKUP** and **RECOVER CFSTRUCT** commands, see the following websites:

- ▶ <http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/sc10810.htm>
- ▶ <http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/sc12630.htm>

11.6 Application impact

The biggest potential application impact is the new reason code, 2561 (MQRC_DATA_SET_NOT_AVAILABLE). This code indicates that the shared message data set that is associated with the queue the application is using is damaged or unavailable. This code often is seen when the SMDS is undefined and unformatted for this queue manager in the queue-sharing group. The codes also are seen if the file is unavailable or was damaged in some other way.

If an application does not handle new reason codes well, there might be unexpected abends and failures.

The second application impact is the potential for much deeper queues than was previously available.



Coupling facility connectivity loss improvements (z/OS)

In this chapter, WebSphere MQ for z/OS resiliency is described. Two new attributes and one new command were added and are related to improving WebSphere MQ resilience to coupling facility connectivity failures.

The use of the following new attributes and the command allows more control over the actions that are taken when the WebSphere MQ queue manager loses connectivity to a coupling facility:

- ▶ **CFCONLOS**

An attribute that allows control over whether a queue manager terminates or tolerates when a connection to a coupling facility or coupling facility structure is lost.

Important: The actions that are taken are explained in more detail later in this chapter.

- ▶ **RECAUTO**

An attribute that relates to if a **RECOVER CFSTRUCT** command is automatically generated if it is needed.

- ▶ **RESET CFSTRUCT ACTION(FAIL)**

A command that can be used to set a coupling facility WebSphere MQ structure to failed.

It can be used to force a rebuild of a structure into a coupling facility with better connectivity.

This chapter contains the following sections:

- ▶ Resiliency
- ▶ Coupling facility structure failures
- ▶ What happens when a failure is reported for the administration structure
- ▶ What happens when a failure is reported for an application structure
- ▶ Actions to pursue when coupling facility connectivity is lost
- ▶ Automatic application structure recovery
- ▶ RESET CFSTRUCT ACTION(FAIL) command
- ▶ Enabling resilience
- ▶ Queue manager reactions to loss of connectivity for coupling facility structures

12.1 Resiliency

Resiliency refers to the ability of a WebSphere MQ queue manager in a queue-sharing group to tolerate the loss of a coupling facility structure without terminating.

12.2 Coupling facility structure failures

The following types of failures that Sysplex services for cross system extended services (XES) data sharing report to the queue manager for a coupling facility structure:

- ▶ A failure of an entire coupling facility, or a failure of an individual structure that is used by WebSphere MQ, as shown in Figure 12-1.

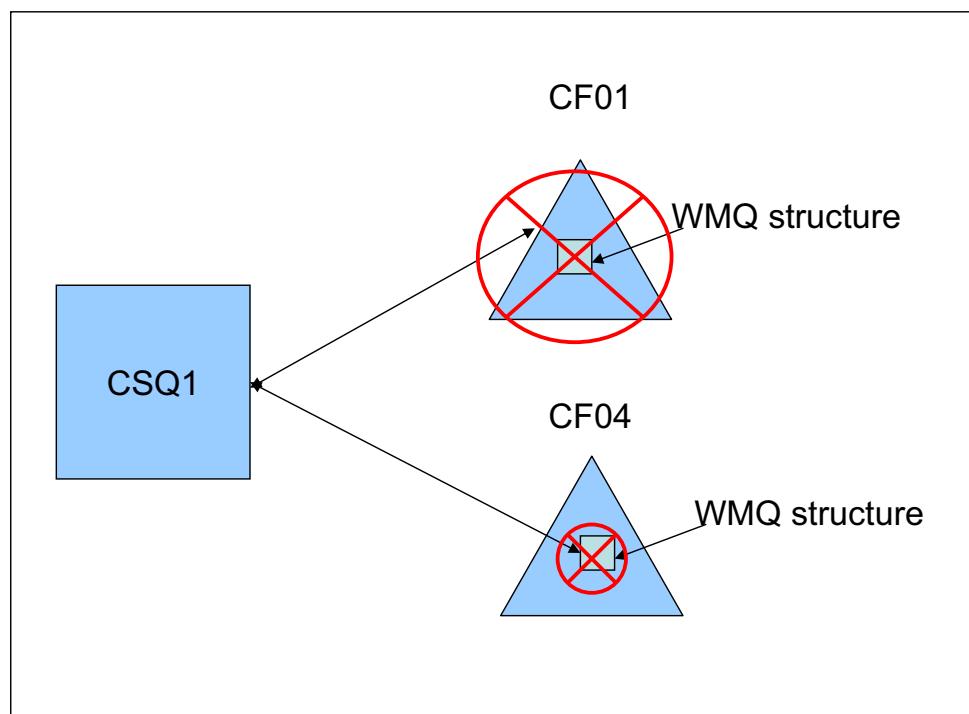


Figure 12-1 Coupling facility failure

- ▶ A connectivity failure for a coupling facility containing a structure.

A loss of connectivity does not necessarily mean a coupling facility failed. Depending on the configuration of the coupling facility paths, other queue managers might still include a path to connect to the coupling facility.

Figure 12-2 shows one member that loses connectivity to a coupling facility.

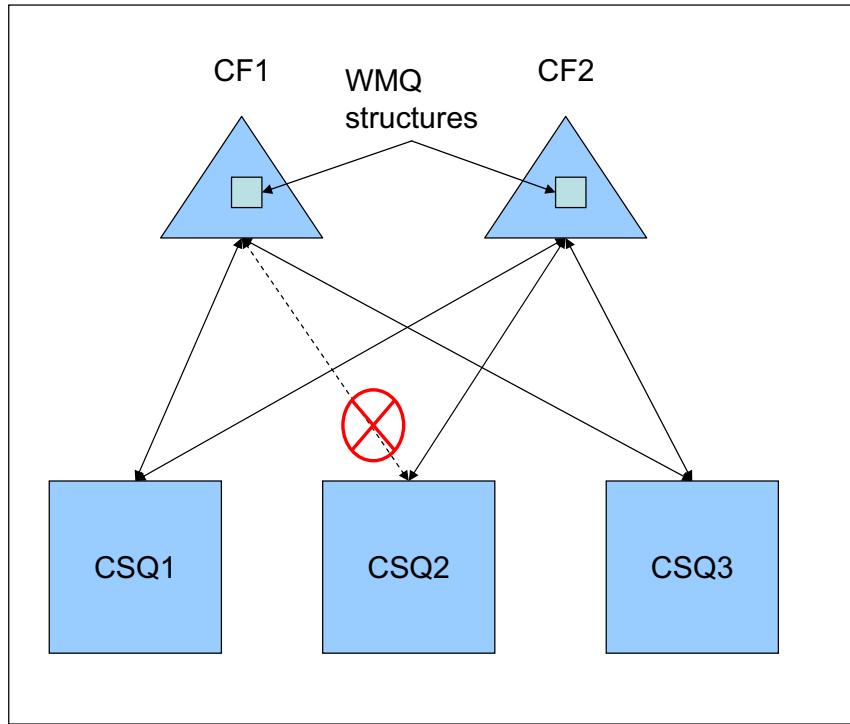


Figure 12-2 Connectivity failure: Not all members of the queue-sharing group lose connectivity

Figure 12-3 shows all members losing connectivity.

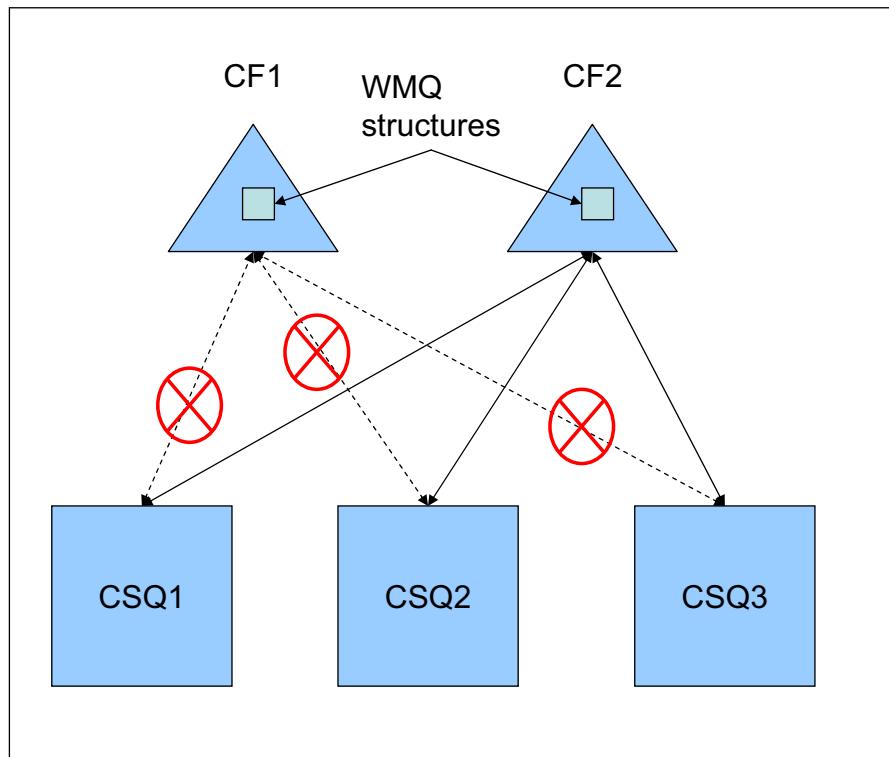


Figure 12-3 A connectivity failure: All members of a queue-sharing group lose connectivity

The coupling facility structures that are used by WebSphere MQ can be defined to use system-managed duplexing. This configuration means that if the coupling facility structure fails in a single coupling facility, the connection switches to the duplicated version and WebSphere MQ is unaware of the change in location. If there is a failure of both instances of the coupling facility structure, the queue manager takes actions that are based on the failure that is reported.

Important: Duplexing of coupling facility structures does come at a cost. z/OS and coupling facility CPU usage increases for those operations that update the structure. Coupling facility link usage also increases. These costs vary structure by structure and depend on the amount of duplexing that is done for each structure. For more information, see the IBM Redbooks publication *High Availability in WebSphere Messaging Solution*, SG24-7839-00, which is available at this website:

<http://w3.itso.ibm.com/abstracts/sg247839.html?Open>

Frequent backups: It is important to perform frequent backups of the coupling facility structures to perform a recovery in a reasonable amount of time. This backup rate reduces the time that is needed to replay the logs from the last backup. It also reduces the need to use archived logs.

If a coupling facility or a coupling facility structure fails, only persistent messages are recovered because non-persistent messages are not written to the log files, which are used to recover the messages.

A queue manager takes different actions when it detects a failure that is based on the following factors:

- ▶ The type of error that is reported by the XES component.
- ▶ The structure type (application or administration).
- ▶ The queue manager version (V6, V7, and so on).
- ▶ The CFLEVEL of the WebSphere MQ CFSTRUCT object (2, 3, 4 or 5); not the CFLEVEL of the coupling facility microcode.

Partial loss of connectivity is defined as the condition in which one or more (but not all) logical partitions (LPARs) in the sysplex lose connectivity to a coupling facility where the structure is defined.

Total loss of connectivity is defined as the condition in which all LPARs in the sysplex lose connectivity to a coupling facility.

When an individual WebSphere MQ queue manager loses connectivity to one or more coupling facility structures, the availability was increased by its ability to tolerate the loss and continue to service the following queues:

- ▶ Non-shared queues
- ▶ Shared queues in other coupling facilities to which it has connectivity

The CFCONLOS attribute setting is used to determine whether the queue manager terminates or tolerates a connectivity loss when it receives a loss of connectivity event.

When a queue manager detects a connectivity failure, it requests a z/OS System Managed Rebuild of the failed structure.

The structure is built in the coupling facility with the best system-wide connectivity if another coupling facility is available.

If the connectivity loss was a total loss of connectivity, a WebSphere MQ **RECOVER CFSTRUCT** command can be used to recover persistent messages into the structure.

Figure 12-4 shows the automatic rebuild of a structure.

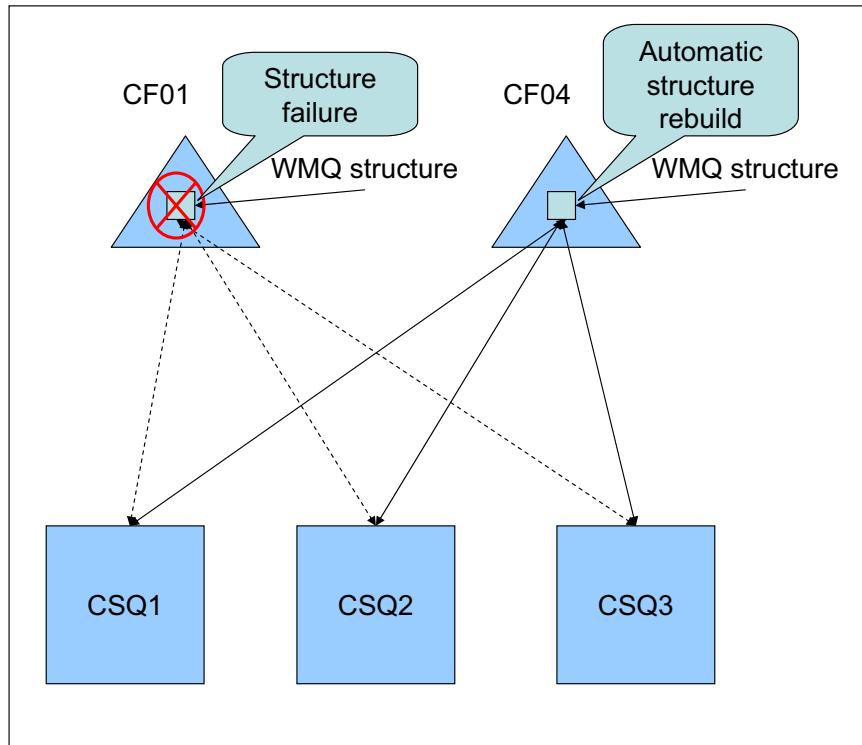


Figure 12-4 Structure rebuild

Figure 12-4 represents a partial loss of connectivity where one or more of the queue managers lose connectivity to the CF01 coupling facility structure.

A z/OS System Managed Rebuild of the failed structure is requested by a queue manager.

z/OS rebuilds the structure in CF04 and notifies the queue managers that the structure is now available.

If the original failure was a connectivity loss to the structure and the structure remained undamaged, the messages within the structure are copied as part of the structure rebuild.

If the original failure was the result of a damaged structure, the contents of the structure require a **RECOVER CFSTRUCT** command. In this case, only persistent messages are recovered because nonpersistent messages are not written to the logs for recovery.

Figure 12-4 shows the structure is rebuilt in CF04.

The following information should be considered before resiliency is enabled.

If a queue manager loses coupling facility connectivity, tolerates the loss, and no other coupling facility is available to rebuild the coupling facility structure, the following actions are taken:

- ▶ The queue manager continues to service non-shared queues.
- ▶ Shared queues cannot be serviced that are in the failed structure.
- ▶ Shared queues in other structures continue to be serviced.
- ▶ Applications can continue to connect to the queue manager even though the shared queues they need in the failed structure are unavailable.

API calls: API calls, such as MQGET or MQPUT to the queues within the failed structure, receive a reason code that indicates the queues are unavailable until connectivity is reestablished.

12.3 What happens when a failure is reported for the administration structure

WebSphere MQ includes two different types of structures. Each queue-sharing group includes an administrative structure that is used to contain state information. The other structures, which are known as application structures, are used to contain shared queue messages.

If a structure is an administration structure, the queue manager is version 7.1 or above, and the queue manager attribute CFCONLOS is set to tolerate, the structure is reallocated and rebuilt without the queue manager terminating.

Important: If the structure is an administration structure and the queue manager is V7.0.1, the structure terminates. At start, it can rebuild the data in the administration structure for all queue managers so you do not have to start every queue manager in the queue sharing group (QSG). This ability is important because you cannot recover application structures until the administration structure is rebuilt.

The following actions on shared queues are suspended until the queue manager reconnects to the administration structure and finishes rebuilding the entries in the structure:

- ▶ Opening of shared queues.
- ▶ Committing or backing out units of recovery.
- ▶ Serialized applications that are connecting to or disconnecting from the queue manager.

Backup or recover of coupling facility application structures cannot be performed until all queue managers in the queue-sharing group rebuild their portion of the coupling facility administration structure.

If a queue manager is not running at the time of the failure, or if it terminates before its portion of the rebuild is complete, another queue manager within the queue-sharing group can rebuild its coupling facility entries if it is version 7.0.1 or above.

If there are queue managers on earlier releases in the queue-sharing group, each queue manager must be restarted so that it rebuilds its portion of the coupling facility administration structure.

12.4 What happens when a failure is reported for an application structure

You can configure a queue manager at version 7.1 to tolerate losing connectivity to application structures at CFLEVEL(5) or higher without terminating.

If the queue manager is configured to tolerate losing connectivity, the following results are observed:

- ▶ The queue manager continues to run and applications that do not use the structure continue to run.
- ▶ Applications that use queues in the failed structure receive a MQRC_CF_STRUC_FAILED error. This error also might be returned when there is a loss of connectivity.
- ▶ After the rebuild is complete, the application can open and perform shared queue processing.

Queue managers that are connected to application structures abend if the following conditions exist:

- ▶ The application structure is at CFLEVEL(5) and loss of connectivity toleration is not enabled.
- ▶ The application structure is at CFLEVEL(4) or lower.

12.5 Actions to pursue when coupling facility connectivity is lost

The WebSphere MQ parameter CFCONLOS determines the action a queue manager takes when connectivity is lost to the administration or an application coupling facility structure is defined at CFLEVEL(5).

The value of the CFCONLOS queue manager attribute determines what action is taken when connectivity is lost to the administration coupling facility structure.

The value of the CFCONLOS application coupling facility structure attribute determines what action is taken when connectivity is lost to the corresponding application coupling facility structure.

In WebSphere MQ Explorer, this field corresponds to the field titled Loss of coupling facility connectivity.

Figure 12-5 shows the WebSphere MQ Explorer dialog that you use to configure a CFSTRUCT object. It shows that the CFCONLOS attribute can be configured on the General panel.

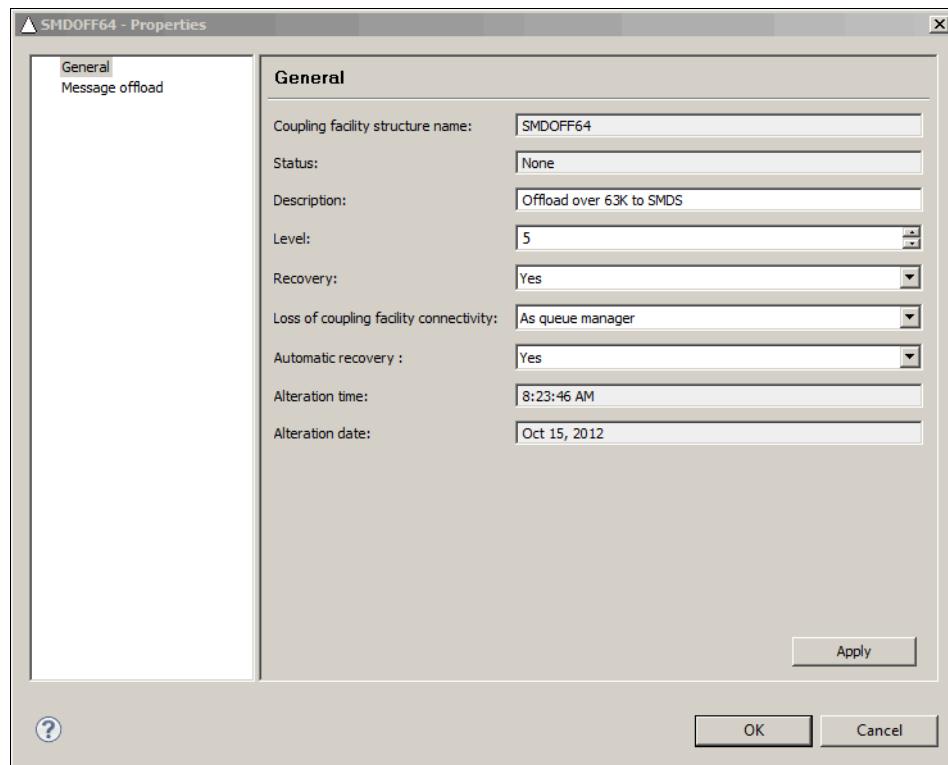


Figure 12-5 Configuring the CFCONLOS attribute for a CFSTRUCT object

Figure 12-6 on page 194 shows the WebSphere MQ Explorer dialog that you use to configure the queue manager object. It shows that the CFCONLOS attribute can be configured on the Extended panel.

Figure 12-5 on page 193 shows the setting of CFSTRUCT CFCONLOS attribute.

Figure 12-6 on page 194 shows the setting of queue manager CFCONLOS attribute.

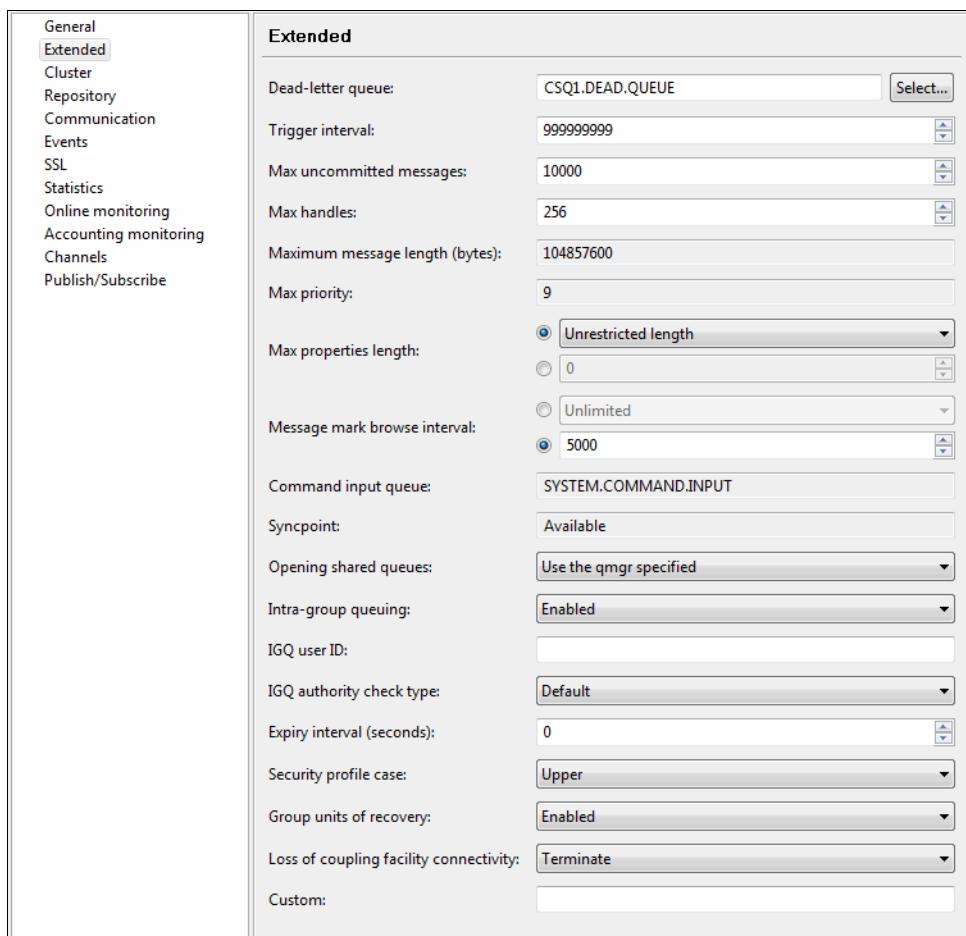


Figure 12-6 Configuring the CFCONLOS queue manager attribute

CFCONLOS is specified in both the queue manager and coupling facility structure object definitions. When set in the coupling facility structure it can be set to default to the queue manager value.

The following values for CFCONLOS define the queue manager actions that are to be taken. The values are valid for the queue manager and coupling facility attributes:

► TERMINATE

Specifies the queue manager terminates when it loses connectivity to the coupling facility structure.

This value is the only permissible value for coupling facility structures that are not at CFLEVEL(5) and the default value for existing coupling facility structures that are altered to CFLEVEL(5).

► TOLERATE

Specifies the queue manager does not terminate if loss of connectivity to a coupling facility structure occurs.

The ASQMGR value is valid for application structures only. It specifies the CFONLOS setting in the queue manager object definition is not used. This value is the default value for new structures that are defined as CFLEVEL(5).

12.6 Automatic application structure recovery

The RECAUTO attribute determines whether the RECOVER CFSTRUCT command should be run automatically for an application structure if it is necessary to restore access.

RECOVER CFSTRUCT must be run for an application structure either automatically or manually if the structure fails or a total loss of connectivity occurred for the coupling facility where the structure is located.

A queue manager initiates RECOVER CFSTRUCT for a structure automatically if RECAUTO is set to YES and one of the following events occur:

- ▶ The structure failed or is empty when the queue manager connects to it when the manager is starting.
- ▶ The structure fails while it is connected to a queue manager
- ▶ A total loss of connectivity to the structure occurs when it is connected to a queue manager

In WebSphere MQ Explorer this field corresponds to the field titled Automatic recovery.

RECAUTO is specified only in the coupling facility structure object definitions.

The following values for RECAUTO define the recovery actions that are to be taken.

- ▶ YES
 - Automatically recover the structure and its associated shared message data sets which also need recovery. It can be set only if the CFSTRUCT is at CFLEVEL(5)
- ▶ NO
 - Does not automatically recover the structure or its associated shared message data set.

12.7 RESET CFSTRUCT ACTION(FAIL) command

By using this command, you can set the status of a coupling facility structure to FAILED if the coupling facility structure is CFLEVEL(5) or above.

This command can be used to manually generate coupling facility failures on test systems, or force a structure to be rebuilt by using RECOVER CFSTRUCT.

It must be issued only on one queue manager in the QSG.

If connectivity to a coupling facility structure is lost by one or more queue managers, a system-managed rebuild of the coupling facility structure is attempted.

Another reason to run this command is that after a rebuild, another coupling facility with better connectivity in which all of the queue managers can connect becomes available.

Failing the structure causes it to be deleted in the coupling facility (CF). When it is re-created by RECOVER CFSTRUCT, it is created in the CF that features the highest availability, which is subject to any coupling facility resource management (CFRM) policy constraints.

CRFM policy: The CRFM policy contains the attributes for a coupling facility structure. When a rebuild of the coupling facility structure is requested, it is rebuilt by using the attributes in the CRFM policy.

12.8 Enabling resilience

Complete the following steps to ensure the ability of the queue manager to initiate a system-managed rebuild to re-create a structure into an available coupling facility:

1. Use the **DISPLAY XCF,COUPLE,TYPE=CFRM** command to determine the format of the CFRM couple data set.
2. Verify that the CFRM couple data set is formatted by specifying ITEM NAME(SMREBLD) NUMBER(1).

ITEM NAME(SMREBLD) NUMBER(1) is shown as SMREBLD(1) on the CFRM display, as shown in Figure 12-7.

```
D XCF,COUPLE,TYPE=CFRM
IXC358I 15.18.53 DISPLAY XCF 532
CFRM COUPLE DATA SETS
PRIMARY   DSN: SYS1.XCF.CFRM00
           VOLSER: TOTCD3    DEVN: 98F0
           FORMAT TOD        MAXSYSTEM
           02/13/2011 16:48:47    20
           ADDITIONAL INFORMATION:
           FORMAT DATA
           POLICY(10) CF(8) STR(512) CONNECT(32)
           SMREBLD(1) SMDUPLEX(1) MSGBASED(1)
```

Figure 12-7 CFRM policy display

3. Verify that an alternative coupling facility is available for use.
4. Verify that the alternative coupling facility in the CFRM preference list for all defined WebSphere MQ coupling facility structures by using the **DISPLAY XCF** command.
5. Use the **DISPLAY XCF,STRUCTURE,STRNM=strnx*** command to display the coupling facility structures.

Replace strnx with the queue-sharing group name to display all of the related coupling facility structures.

Figure 12-8 shows that more than one alternate coupling facility is defined for the structure on the PREFERENCE LIST: line.

```
STRNAME: IBM1SMDOFF64
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE      : 512 M
  POLICY INITSIZE: 272144 K
  POLICY MINSIZE : 204108 K
  FULLTHRESHOLD  : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX          : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF03      CF04      CF02
  ENFORCEORDER    : NO
  EXCLUSION LIST IS EMPTY
```

Figure 12-8 Display of the XCF Structure command

6. Migrate all of the queue managers in the queue-sharing group to WebSphere MQ V71 or above with the OPMODE in the ZPRM member that is set to enable WebSphere MQ v71 functions.

The OPMODE can be set in the CSQ6SYSP portion of the ZPRM member, as shown in the following example:

```
OPMODE=(NEWFUNC,710),
```

7. Change all application structures to CFLEVEL(5) for all structures that must tolerate loss of connectivity.
8. Set the values for CFCONLOS in the queue manager and CFSTRUCTS objects to tolerate the loss of connectivity. In WebSphere MQ Explorer, this field corresponds to the field titled Loss of coupling facility connectivity.
9. Set the value for CFSTRUCT RECAUTO in the CFSTRUCTS to YES to allow automatic recovery. In WebSphere MQ Explorer, this field corresponds to the field titled Automatic recovery.

Figure 12-9 shows the values for CFLEVEL, CFCONLOS, and RECAUTO for a CFSTRUCT object on the WebSphere MQ Explorer coupling facility structure panel.

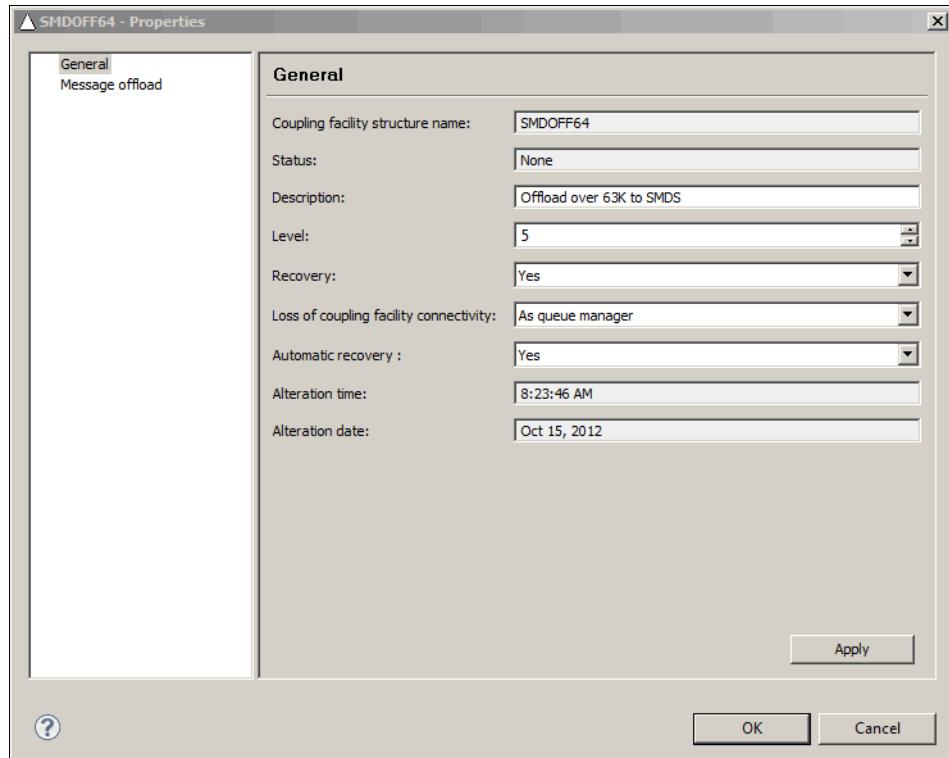


Figure 12-9 WebSphere MQ Explorer CFSTRUCT display panel

12.9 Queue manager reactions to loss of connectivity for coupling facility structures

The behavior for the loss of connectivity to a coupling facility structure depends on whether the structure is an administration or application structure.

12.9.1 Loss of connectivity to the administration structure

If any queue manager in the queue-sharing group is lower than WebSphere MQ V7.1, all queue managers abend with reason code 00C510AB.

Any queue managers in the queue-sharing group that were not configured to tolerate the loss of connectivity still abend.

If any queue manager at WebSphere MQ V71, or higher, loses connectivity to the administration structure and it is configured to tolerate the loss of connection, the following actions:

- ▶ All queue managers in the queue-sharing group disconnect.
- ▶ All queue managers in the queue-sharing group that tolerate the loss of connectivity attempt to connect to the administration structure, which causes it to be reallocated and rebuilt in a different coupling facility.
- ▶ The coupling facility structure is reallocated in the coupling facility with the best connectivity to all LPARs in the sysplex.

Important: The coupling facility where the structure is reallocated might not be the coupling facility with the best connectivity to all LPARs that have active queue managers.

If a queue manager tolerates the loss of the coupling facility administration structure and cannot reconnect to the administration structure, the following actions occur:

- ▶ Some shared queue operations are not available until the queue manager reconnects to the administration structure.
- ▶ The queue manager automatically reconnects when a suitable coupling facility becomes available.

The failure of a queue manager to connect to the administration structure at start is not tolerated and the queue manager abends.

12.9.2 Loss of connectivity to the application structure

If the structure is at CFLEVEL(4), or lower, or the structure is at CFLEVEL(5) and is not configured to tolerate loss of connectivity, the queue manager abends with reason code 00C510AB.

When the application structure is configured to tolerate loss of connectivity, all queue managers that lost connectivity disconnect from the application structure.

The next actions of the queue manager depend on whether the connectivity was a partial or total loss.

12.9.3 Partial loss of connectivity to an application structure

If a queue manager at version 7.1 is active on an LPAR that did not lose connectivity to the structure, it initiates a system-managed rebuild. This rebuild requests that z/OS moves the structure to a coupling facility with better connectivity.

If the application structure rebuild is successful, the following actions occur:

- ▶ Queue managers that retained connectivity still include access to queues on the affected application structure.

Important: Operations that are performed to some queues on the affected application structure might experience a delay until the rebuild process is complete.

- ▶ Persistent and non-persistent messages are copied to the new application structure.
- ▶ Access to the queues is restored for queue managers that are on LPARs that include connectivity to the alternative coupling facility.

If the application structure cannot be rebuilt or some queue managers do not include connectivity to the newly built application structure, the following actions occur:

- ▶ Connected queue managers include access to shared queues in the application structure.
- ▶ Non-connected queue managers do not include access to the shared queues until the connectivity is restored to the application structure.
- ▶ Queue managers automatically connect to the application structure when it becomes available and access to the shared queues is restored.

12.9.4 Total loss of connectivity to an application structure

z/OS deallocates the application structure. All systems in the sysplex lose connectivity and an attempt is made to connect to the application structure.

Important: It is likely that all non-persistent messages in the application structure are lost after a total loss of connectivity to an application structure.

If the application structure is defined with RECAUTO parameter set to YES, it is recovered automatically after a total loss of connectivity. The recovery starts when an alternative coupling facility is available or a coupling facility becomes available.

If the application structure is defined with the RECAUTO parameter set to NO, the following actions occur:

- ▶ Recovery can be started by issuing the RECOVER CFSTRUCT command.
- ▶ Persistent messages are recovered.

Because this process involves reading the queue manager logs, it takes time to complete. The length of time varies by the number of logs and the amount of data that it must read on each log.

Important: It is recommended that structure backups be taken regularly to reduce the amount of time that is needed to restore access to the shared queues in the application structure.

- ▶ Non-persistent messages are lost.

An attempt to reconnect to the unrecoverable application structure is made when the following conditions exist:

- ▶ An application attempts to open a shared queue in the application structure.
- ▶ A notification is received from the system that new coupling facility resources are available.

If a suitable coupling facility is available to allocate the unrecoverable application structure, a new structure is allocated and access to shared queues in the structure is restored.

Important: Because persistent messages cannot be written to an application structure that is defined as not recoverable and non-persistent messages are not recoverable, all prior messages in the structure are lost.



Extended integration with IMS (z/OS)

This chapter describes the features that were introduced in WebSphere MQ V7.1 to enhance the integration with IMS when the WebSphere MQ-IMS bridge is used. This component of WebSphere MQ for z/OS allows WebSphere MQ messages to drive transactions in IMS by using the IMS Open Transaction Manager Access interface.

This chapter contains the following sections:

- ▶ Resource monitoring
- ▶ Requesting a response with commit mode 0

13.1 Resource monitoring

In IMS version 10, the Open Transaction Manager Access (OTMA) component was enhanced with a resource monitoring capability. OTMA monitors resources in IMS and notifies OTMA clients, such as the WebSphere MQ-IMS bridge, if these resources are operating normally, suffering from a degradation, or unavailable.

The following notifications are sent by OTMA:

- ▶ Whenever WebSphere MQ connects to IMS
- ▶ Whenever a change of status occurs
- ▶ As a heartbeat notification, every 60 seconds

When messages are put to an IMS Bridge enabled queue, the transaction requests that they represent are sent to IMS. These transaction requests are queued in IMS until an IMS region is available to process them. IMS obtains storage for each transaction that is queued, which is known as the *Transaction Instance Block* (TIB). IMS requires this storage to be below the line, which means that it must be addressable by using a 24-bit address. The available storage within this addressable range is limited. If transactions are received by IMS faster than it can process them, the available storage is exhausted and IMS can fail.

To help mitigate this issue, IMS provides the capability to restrict the number of pending transactions each OTMA client can send. After the backlog in IMS reaches this limit, IMS rejects further transaction requests from the OTMA client until the backlog is reduced to a manageable size. This limit can be set for each OTMA client to a number 200 - 9999 by using the **/START TMEMBER** command. The use of this command to set the limit for a queue manager that is called MQ1E is shown in the following example:

```
/START TMEMBER MQ1E INPUT 200
```

OTMA reports that IMS is suffering a degradation in performance if the transaction backlog for an OTMA client reaches 80% of this limit. A threshold also can be set for the total backlog of all TMEMBERs. A degradation in performance is reported to all OTMA clients if this threshold is reached. If the backlog reaches 100% of the configured limit for a TMEMBER, OTMA reports that IMS is flooded and is unable to accept further transaction requests from that client.

You can view the configured limits for each TMEMBER by using the **/DISPLAY OTMA** command, an example of which is shown in Example 13-1. The TIB column reports the current backlog for each TMEMBER and the INPT column reports the configured limit. The configured limit for the server is the threshold that is used to report a warning for the total backlog of all TMEMBERs.

Example 13-1 Output from an IMS /DISPLAY OTMA command

DFS000I	GROUP/MEMBER	XCF-STATUS	USER-STATUS	SECURITY	TIB	INPT	SMEM
DFS000I		DRUEXIT	T/O	TPCNT			
DFS000I	IM10AGRP						
DFS000I	-IYEBZIOA	ACTIVE	SERVER	NONE	0	8000	
DFS000I	-IYEBZIOA	N/A	0	0			
DFS000I	-MQ1E	ACTIVE	FLOOD	NONE	200	200	
DFS000I	-MQ1E		120	0			
DFS000I	-MQ1F	ACTIVE	ACCEPT	TRAFFIC	NONE	0	5000
DFS000I	-MQ1F		120	0			

In Example 13-1 on page 204, the status for queue manager MQ1E indicates that it is flooded. When a TMEMBER's status becomes flooded, IMS suspends all of its TPIPEs to prevent further transaction requests from being sent. IMS resumes the TPIPEs when the backlog reduces to 50% of the configured limit.

WebSphere MQ for z/OS V7.1 is enhanced to understand these notifications. When a queue manager receives a warning notification that IMS is struggling to process the volume of transaction requests WebSphere MQ is sending, it reduces the rate at which subsequent requests are sent. This function provides IMS with an opportunity to reduce the existing backlog in an attempt to avert a full flood condition. If a WebSphere MQ for z/OS queue manager is connected to multiple IMS partners, the throttling of the rate that requests are sent is performed independently for each IMS. If only one IMS is degraded, the flow of transactions to the other IMS partners is unaffected.

Upon receipt of a notification from OTMA that the flood or flood warning condition is relieved, WebSphere MQ for z/OS gradually increases the rate that transaction requests are sent to IMS until the maximum rate is reached. Gradually increasing the rate reduces the likelihood that a flood or flood warning condition is immediately reintroduced.

This change in behavior becomes effective when new functions for version 7.1 are enabled by using the OPMODE system parameter in the queue manager's ZPARM. It is not expected that applications should observe a change in response time because there is no change unless a flood or flood warning condition results. The change means that messages remain queued on a WebSphere MQ queue instead of being queued in IMS. As IMS notifies WebSphere MQ when the backlog is still 50%, this notification provides time for WebSphere MQ to resume, or increase its rate, before the remaining requests are processed.

PTF UK45082: If IMS version 10 is used, you should ensure that PTF UK45082 is applied.

13.1.1 New console messages

To allow a WebSphere MQ administrator to monitor the status of each IMS partner, the queue manager issues a console message whenever IMS reports a change in status. These messages can be used to understand any increase in depth for one or more IMS Bridge enabled queues.

When a flood warning condition arises because the backlog for a TMEMBER reaches 80%, IMS issues console message DFS1988W, as shown in the following example:

```
DFS1988W OTMA INPUT MESSAGES FROM MQ1E HAVE REACHED 80% OF THE MAXIMUM MESSAGE  
LIMIT 200
```

IMS repeats this message whenever the backlog increases by another 5%.

WebSphere MQ issues console message CSQ2040I, which identifies the IMS partner that is degraded. This message indicates that a reduction in the rate of transactions is sent to that IMS became effective, as shown in the following example:

```
CSQ2040I MQ1E CSQ2CTL0 OTMA MESSAGE FLOOD STATUS=WARNING FOR PARTNER,  
XCFGNAME=IM10AGRP XCFMNAME=IYEBZIOA
```

Despite the reduction in the rate that transaction requests are sent, if a full flood condition results, IMS issues console message DFS1989E, as shown in the following example:

```
DFS1989E OTMA INPUT MESSAGES FROM MQ1E HAVE REACHED THE MAXIMUM MESSAGE LIMIT  
200
```

WebSphere MQ issues console message CSQ2041I to indicate that all TPIPEs are suspended by IMS until its existing backlog is reduced, as shown in the following example:

```
CSQ2041I MQ1E CSQ2CTL0 OTMA MESSAGE FLOOD STATUS=FLOODED FOR PARTNER,  
XCFGNAME=IM10AGRP XCFMNAME=IYEBZIOA
```

When a flood condition is relieved, IMS issues console message DFS0767I, as shown in the following example:

```
DFS0767I OTMA MESSAGE FLOOD CONDITION HAS BEEN RELIEVED FOR MEMBER MQ1E
```

When a flood or flood warning condition is relieved, WebSphere MQ issues console message CSQ2042I to negate the previously issued CSQ2040I or CSQ2041I message. The CSQ2042I message indicates that the rate that transactions are sent to IMS is increased until the maximum rate is achieved, as shown in the following example:

```
CSQ2042I MQ1E CSQ2CTL0 OTMA MESSAGE FLOOD RELIEVED FOR PARTNER,  
XCFGNAME=IM10AGRP XCFMNAME=IYEBZIOA
```

Important: WebSphere MQ for z/OS issues these console messages only if the version 7.1 new functions are enabled.

13.1.2 Using shared queues

It is common to use a queue-sharing group and include shared queues that are configured to be used by the WebSphere MQ-IMS bridge. In this setup, each queue manager often is paired with an IMS, and the IMS partners form an IMSplex.

The performance of each IMS can vary independently, which means that a degradation can be encountered by one or more IMS partners, while the others remain unaffected. The queue managers that are connected to the degraded IMS partners throttle the rate they send transaction requests to them. If each queue manager in the queue-sharing group is paired with a different IMS, the other queue managers that are connected to the healthy IMS partners automatically process a greater proportion of the transaction requests on the shared queues. This configuration provides for natural workload balancing to compensate for the degradation.

13.2 Requesting a response with commit mode 0

WebSphere MQ V7.0.1 introduced a new flag in the MQIIH, called MQIIH_IGNORE_PURG, to request that the TMAMIPRG indicator should be set in the OTMA prefix when the transaction request is sent to IMS. This indicator requests that OTMA ignores PURG calls on the TP PCB when a transaction is run by using commit mode 0. This function was added in IMS to help customers convert transactions between commit mode 0 and commit mode 1. Commit mode 0 is also called commit-then-send. Commit mode 1 is also called send-then-commit.

WebSphere MQ V7.1 introduces another flag in the MQIIH to support a related capability in IMS. This flag, called MQIIH_CM0_REQUEST_RESPONSE, requests that the TMAMHRSP indicator should be set in the OTMA prefix when a transaction request is sent to IMS by using commit mode 0. This flag requests that IMS generates a response if the IMS application ends without replying to the IOPCB, or message switching to another application. The purpose of this feature is to ensure that a response is always returned, which for commit mode 0 is not otherwise assured.

If IMS generates a response instead of the application program, the response that is returned to WebSphere MQ is shown in the following example:

DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY

To use the MQIIH_IGNORE_PURG and MQIIH_CM0_REQUEST_RESPONSE flags, an application is required to specify an MQIIH header in messages that it sends.



CSQINPT DD added to queue manager startup JCL (z/OS)

In this chapter, we introduce the new data definition statement that is called CSQINPT that was added to the WebSphere MQ queue manager JCL procedure. This data definition can be used to specify WebSphere MQ script commands (MQSC) to be processed when publish/subscribe initialization completes.

This chapter contains the following sections:

- ▶ Introducing CSQINPT
- ▶ Changes to commands
- ▶ New sample initialization input data sets

14.1 Introducing CSQINPT

WebSphere MQ for z/OS V7.0 introduced support for the publish/subscribe messaging model. In publish/subscribe, applications request that a queue manager delivers messages to them on one or more subjects, which are known as topics. An application registers an interest in a topic by using a subscription that also identifies to the queue manager on what queue matching messages should be placed. In this messaging model, messages are put to a topic instead of a queue.

Putting a message to a topic is known as a *publication*. When a publication is received, a queue manager determines which applications, if any, subscribed to the topic. The queue manager delivers a copy of the publication to all subscribers that are registered to the topic.

Publish/subscribe initialization occurs when support for publish/subscribe is enabled by using the PSMODE queue manager attribute. This initialization is also performed during queue manager startup if publish/subscribe support is enabled. Attempts to administer a subscription fail if they are attempted before publish/subscribe initialization completes or publish/subscribe support is disabled.

The queue manager JCL procedure supports data definition statements that allow MQSC commands to be specified. These commands are processed when a queue manager starts before applications can connect to the queue manager. The following data definition statements are used:

- ▶ CSQINP1: This statement is used to define buffer pools and pagesets.
- ▶ CSQINP2: This statement is used to define other objects.

Another data definition, called CSQINPX, can be used for MQSC commands that require the channel initiator to be running. The commands that are specified in this data set concatenation are processed when the channel initiator is started.

A new data definition, called CSQINPT, was introduced in WebSphere MQ V7.1. The MQSC commands that are specified in this data set concatenation are processed when publish/subscribe initialization completes. This definition was added because commands to administer subscriptions in CSQINP2 fail if publish/subscribe is not enabled when a queue manager starts. CSQINPT provides a means to automatically run commands to administer subscriptions if publish/subscribe support is then enabled. The CSQINPT data set concatenation can be used to ensure that important subscriptions always are defined.

The output from commands that are run via CSQINPT is written to the data set that is specified by the CSQOUTT data definition statement.

14.2 Changes to commands

In WebSphere MQ for z/OS V7.1, the MQSC commands to administer subscriptions are no longer allowed in CSQINP2, although commands to administer topics are still permitted. The following commands are no longer allowed in CSQINP2:

- ▶ **ALTER SUB**
- ▶ **DEFINE SUB**
- ▶ **DELETE SUB**
- ▶ **DISPLAY SUB**

If you have queue managers with publish/subscribe enabled, you should review the CSQINP2 data set concatenation. If any commands to administer subscriptions are present, they must be moved to the concatenation for CSQINPT.

Important: If you have commands in the CSQINP2 data set concatenation that depend on subscriptions, you might need to move them to the CSQINPT concatenation as well.

14.3 New sample initialization input data sets

WebSphere MQ for z/OS provides sample initialization input data sets for each supported data definition statement. Some of these samples provide definitions for system objects, others are templates for user definitions. The following samples were added for CSQINPT:

- ▶ CSQ4INST: The definition for the system default subscription, which was in CSQ4INSG.
- ▶ CSQ4INYT: The template for user definitions.

Important: Sample initialization input data sets, whose names include the letter S as the seventh character, contain system definitions. Samples, whose names include the letter Y in this position, are templates for you to customize with your own definitions.



CICS 4.2 group attach (GROUPUR)

In WebSphere MQ for z/OS V7.0.1, a new capability that is known as Group units of recovery was added for transactional clients and is supported for CICS.

In this chapter, we describe how Group units of recovery provides support for CICS to connect to a queue-sharing group. This functionality extends the availability of WebSphere MQ to CICS regions because they can connect to any available queue manager on the same logical partition (LPAR), if one is available and is a member of the queue-sharing group.

For more information about Group units of recovery disposition, see this website:

<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/zc14000.htm>

This chapter contains the following sections:

- ▶ Extending the availability of WebSphere MQ to CICS
- ▶ Enabling Group units of recovery

15.1 Extending the availability of WebSphere MQ to CICS

CICS TS version 4.1 introduced group attachment, which allowed a CICS region to connect to a queue-sharing group. Previously, its connection allowed a connection only to an individual queue manager. Group attachment enabled a region to automatically switch to another member of the queue-sharing group on the same LPAR (if one was available) when the current connection was broken.

If WebSphere MQ is not at version 7.1, or CICS Transaction Server is not at version 4.2, a CICS region has an affinity to the queue manager to resolve indoubt units of work. Figure Figure 15-1 shows that CICS1, when the qmgr unit of recovery is used, that another queue manager cannot resolve the unit of recovery.

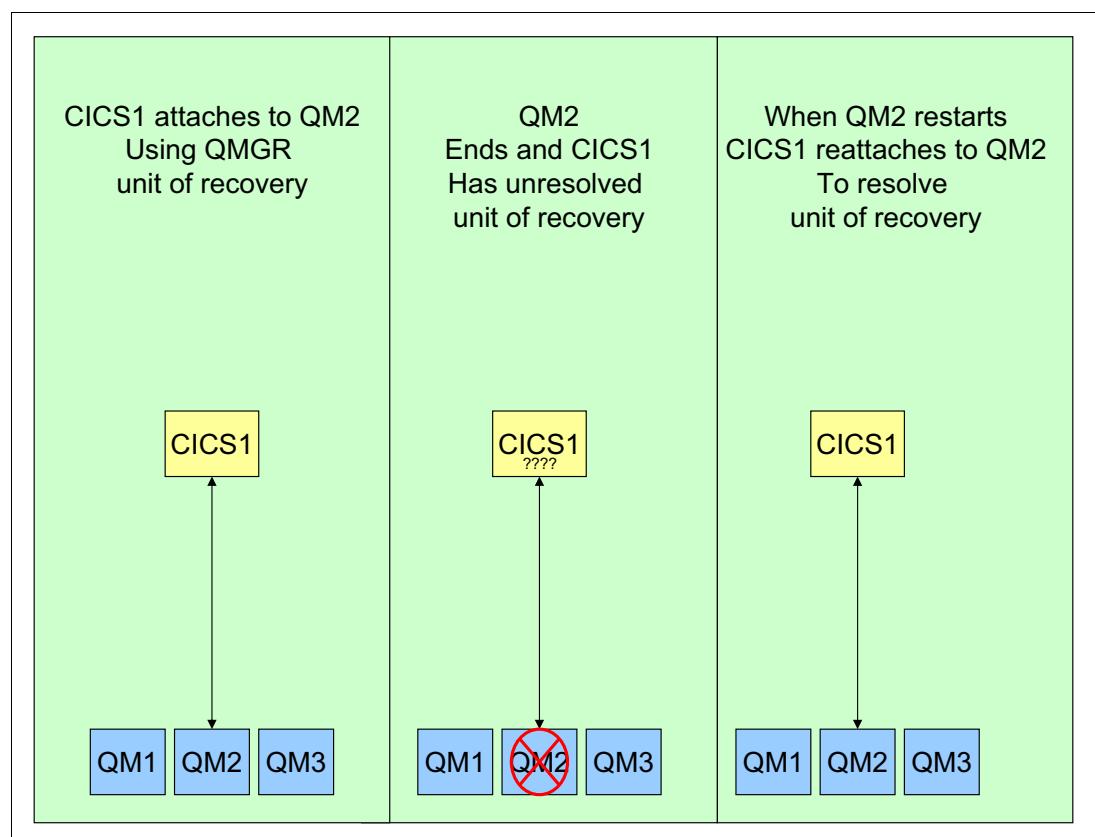


Figure 15-1 The qmgr unit of recovery

If CICS disconnects from the queue manager and creates indoubt units of work, the CICS region waits until the same queue manager becomes available to recover the indoubt units of work. The delay that is created impacts the availability of the CICS region to process WebSphere MQ messages.

The indoubt units of work might involve more resources than WebSphere MQ messages, such as DB2 or file I/O. If so, other CICS transactions that need those resources cannot obtain them until the indoubt units of work are resolved.

WebSphere MQ for z/OS version 7.1 and CICS TS version 4.2 extended the group attachment by introducing Group units of recovery. The use of these units of recovery means that if the CICS region is using group attachment and the connection to the queue-sharing group member is broken and results in indoubt units of work, the queue-sharing group member to which it reconnects resolves them.

If a CICS region connects to a named queue manager, the transactions it starts includes a *queue manager* unit of recovery disposition. Transactions with this disposition are private to the queue manager, so CICS must reconnect to the same queue manager to inquire and resolve them if they are indoubt.

If a CICS region connects to a queue-sharing group, and both support Group units of recovery, the transactions it starts has a *group* unit of recovery disposition. Transactions with this disposition are visible to all queue managers in the queue-sharing group. Therefore, CICS can reconnect to any queue manager in the queue sharing group on the same LPAR to inquire and resolve them if they are indoubt, irrespective of the queue manager on which they were started. Indoubt transactions can be inquired and resolved even if the original queue manager is unavailable.

If CICS connects to a queue-sharing group, it is logically connected to the group but is attached to a specific queue manager.

Figure 15-2 shows the relationship between the CICS region, queue-sharing group, and the member of the queue-sharing group.

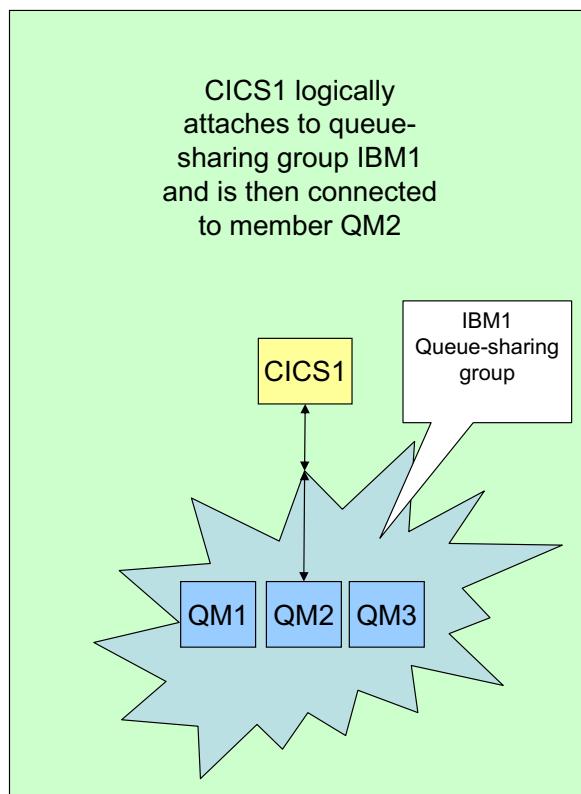


Figure 15-2 CICS attachment to a queue-sharing group

The CICS transaction CKQC can be used to show the status of the connection between CICS and WebSphere MQ, as shown in Figure 15-3. It shows the connected queue manager name in the field to the right of QMgr name =. The queue-sharing group name shows in the field to the right of Mqname =.

Figure 15-2 on page 215 shows the connection between the CICS region, queue-sharing group, and the member of the queue-sharing group.

```
CKQCM2          Display Connection panel

Read connection information. Then press F12 to cancel.

CICS Applid = CICS1 Connection Status = Connected      QMgr name= QM1
Mqname = IBM1      Tracing      = On                  API Exit = Off
Initiation Queue Name = IBM1.TOR.INITQ

----- STATISTICS -----
Number of in-flight tasks = 1      Total API calls      =
Number of running CKTI = 1
      APIs and flows analysis      Syncpoint      Recovery
----- -----
Run OK      1 MQINQ      0 Tasks      0 Indoubt      0
Futile      0 MQSET      0 Backout     0 UnResol     0
MQOPEN      1 ----- Flows ----- Commit     0 Commit      0
MQCLOSE     0 Calls       6 S-Phase     0 Backout     0
MQGET       1 SyncComp    5 2-Phase     0
GETWAIT     1 SuspReqd   0
MQPUT       0 Msg Wait    1
MQPUT1     0 Switched    0

F1=Help F12=Cancel Enter=Refresh
```

Figure 15-3 CICS CKQC WebSphere MQ Display connection window

If the CICS region attaches to the queue-sharing group and the queue manager to which it attaches ends, the CICS region automatically attaches to another member of the queue-sharing group on the same LPAR, if one is available.

Figure 15-4 shows the sequence of events to resolve a transaction with a group unit of recovery disposition.

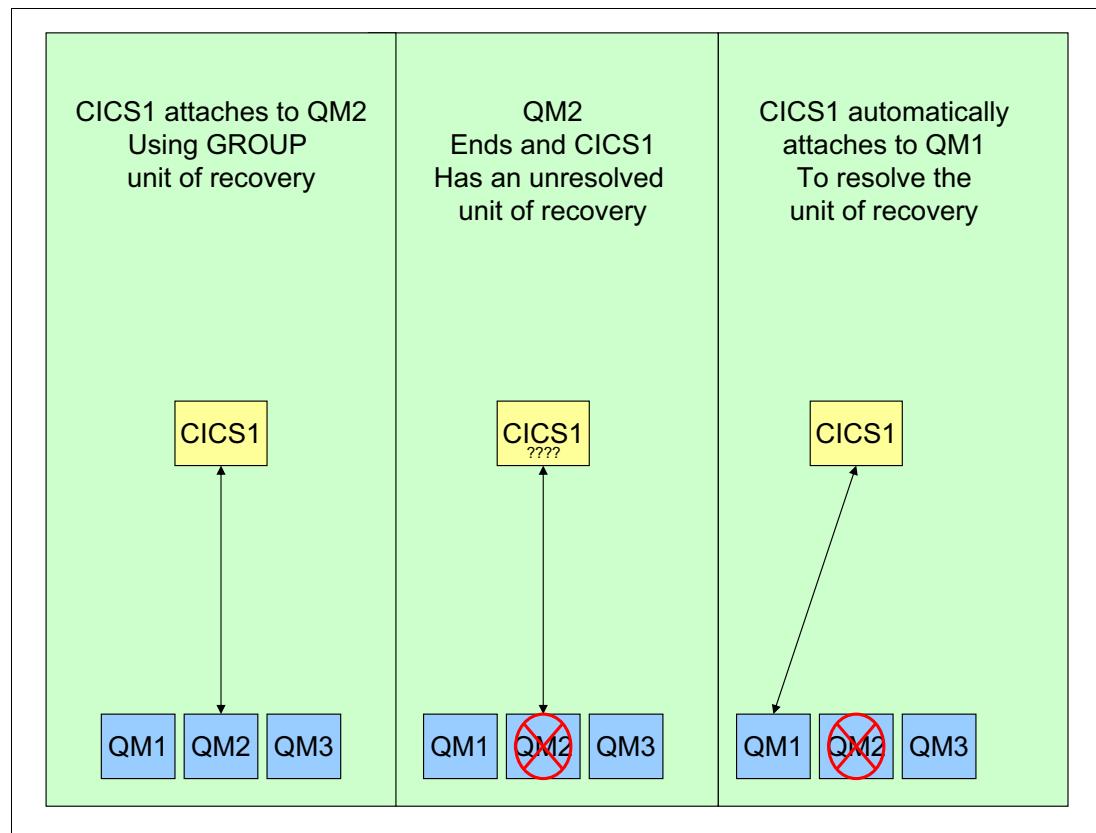


Figure 15-4 Group unit of recovery

CICS now can reduce the number of indoubt units of recovery that are related to WebSphere MQ. This ability increases availability by allowing other transactions to run and use the WebSphere MQ resources that were previous unavailable because of a failed unit of recovery.

For group unit of recovery to be used, the following conditions must exist:

- The WebSphere MQ version must be 7.1 or later.
- The queue manager GROUPPUT attribute must be enabled.

Important: To enable Group units of recovery, a number of requirements must be satisfied. For more information, see “Enabling Group units of recovery” on page 220.

- The attaching CICS region must be at version 4.2 or higher.
- The MQCONN CICS resource definition for the WebSphere MQ queue manager must specify RESYNCMEMBER(GROUPRESYNC).

Figure 15-5 shows the sample output of the **CICS CEMT Inquire MQCONN** command, which shows that Resyncmember is set to Resync. This setting means that CICS connects to the same queue manager and waits, if necessary, until the queue manager becomes active to resolve the indoubt units of work.

```
I MQCONN
STATUS: RESULTS - OVERTYPE TO MODIFY
Mqname( IBM1 )
Mqmgr(CSQ3)
Mqrelease(0710)
Connectst( Connected )
Resyncmember( Resync )
Tasks(0001)
Trigmontasks(0001)
Installtime(09/25/12 13:56:21)
Installusrid(CICSUSER)
Installagent(Grplist)
Definesource(WMQ)
Definetime(09/18/12 15:58:30)
Changetime(09/18/12 15:58:30)
Changeusrid(CICSUSER)
Changeagent(Csdapi)
Changeagrel(0670)

SYSID=WRKS APPLID=CICSPAZ
TIME: 12.31.29 DATE: 10/09/12
RESPONSE: NORMAL
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 15-5 CICS CEMT display of MQCONN which shows group units of recover are not supported

Figure 15-6 shows the sample output of the **CICS CEMT Inquire MQCONN** command, which shows that Resyncmember is set to Groupresync.

```
INQUIRE MQCONN
STATUS: RESULTS - OVERTYPE TO MODIFY
Mqname( IBM1 )
Mqmgr(CSQ1)
Mqrelease(0710)
Connectst( Connected )
Resyncmember( Groupresync )
Tasks(0001)
Trigmontasks(0001)
Installtime(10/18/12 09:04:50)
Installusrid(CICSUSER)
Installagent(Grplist)
Definesource(WMQ)
Definetime(09/18/12 16:04:35)
Changetime(10/17/12 13:27:49)
Changeusrid(CICSUSER)
Changeagent(Csdapi)
Changeagrel(0670)

SYSID=WRKS APPLID=CICSPA25
RESPONSE: NORMAL TIME: 12.34.28 DATE: 10/20/12
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 15-6 CICS CEMT display of MQCONN that shows group units of recover are supported

Important: CICS message DFHMQ0309 is issued if RESYNCMEMBER(GROUPRESYNC) is specified and Group units of recovery are not supported for CICS, or not enabled on the WebSphere MQ queue manager to which it attaches. This message is a CICS message and does not appear in the queue manager JES log.

15.2 Enabling Group units of recovery

Group units of recovery are enabled by using the queue manager attribute called GROUPUR.

When the queue manager GROUPUR parameter is enabled, the queue manager performs the following checks to verify it can participate in Group units of recovery. If any of the following checks fail, the CSQM507E or CSQM506I message is generated and the attempt to enable GROUPUR fails:

1. The queue manager is a member of a queue-sharing group.
2. The SYSTEM.QSG.UR.RESOLUTION.QUEUE exists.
3. The SYSTEM.QSG.UR.RESOLUTION.QUEUE supports persistent messages.

Important: This check requires that the CFSTRUCT CSQSYSAPPL, which is used for the queue, has the recovery attribute set to yes.

4. The SYSTEM.QSG.UR.RESOLUTION.QUEUE is indexed by correlation ID.
5. The SYSTEM.QSG.UR.RESOLUTION.QUEUE is in the system application coupling facility structure CSQSYSAPPL.
6. The queue manager name is not the same name as the queue-sharing group.
7. Group units of recovery are not restricted by the queue manager's mode of operation (OPMODE).

A numerical code that is identified in the message can be used to determine a check that failed, as shown in the following example:

```
CSQM507E -CSQ1 CSQMAMMS GROUPUR qmgr attribute was not enabled CODE=2
```

The same checks are performed during queue manager start if Group units of recovery are enabled. If any of the fail Group units of recovery are disabled, the following message is issued:

```
CSQM506I -CSQ1 CSQMQMUP GROUPUR qmgr attribute has been disabled CODE=2
```

Complete the following steps to ensure that the Group units of recovery can be enabled on a queue manager:

1. Verify that the queue manager is a member of a queue-sharing group.

Figure 15-7 shows queue manager CSQ1 is a member of the IBM1 queue-sharing group.

The screenshot shows the WebSphere MQ Explorer interface. On the left, a tree view lists 'Queue Managers' (CSQ1, CSQ2, CSQ3, CSQ4, CSQ5) and 'Queue-sharing Groups' (IBM1). Under IBM1, there are sub-items: Shared Queues, Coupling Facility Structures, and Group Definitions. On the right, the main window is titled 'Queue-sharing Group IBM1'. It contains two sections: 'Queue-sharing group information source:' and 'Members of queue-sharing group:'. The 'Information source' section shows the following details for CSQ1:

Queue manager name	CSQ1
Description	IBM WebSphere MQ for z/OS - V7.1.0
Command level	710
QMID	CSQ1.C9DB8780CBFD7CA7
Coded character set ID	500

The 'Last updated: 11:28:22' message is displayed below this section. The 'Members of queue-sharing group:' section shows the following table:

Queue manager name	Queue manager number	DB2 name	Queue manager status	DB2 connection status
CSQ1	1	D911	Active	Active
CSQ2	2	D912	Active	Active
CSQ3	3	D911	Active	Active

Figure 15-7 WebSphere MQ Explorer Queue Sharing Group display

2. Verify that the shared queue called SYSTEM.QSG.UR.RESOLUTION.QUEUE exists and supports persistent messages.
 - a. Manually perform the following steps to ensure that the Group units of recovery can be enabled on a queue manager.

Figure 15-8 shows that the queue exists.

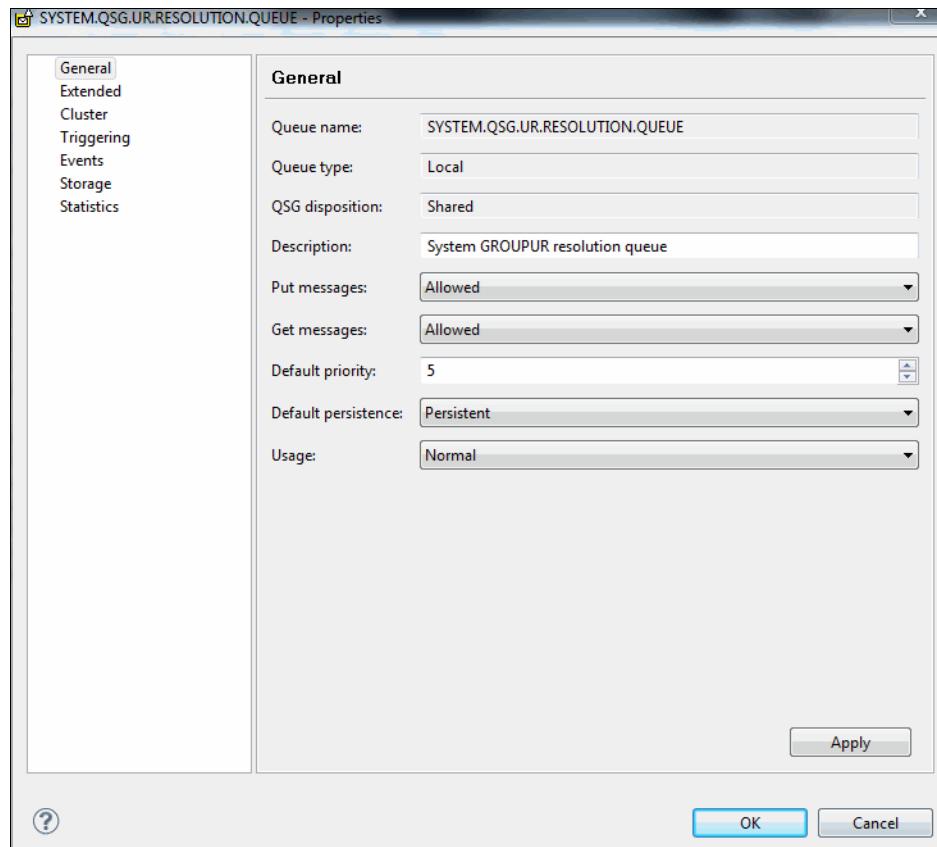


Figure 15-8 WebSphere MQ Explorer Queue general panel shows that the queue exists

Figure 15-9 shows the recovery attribute that supports persistent messages.

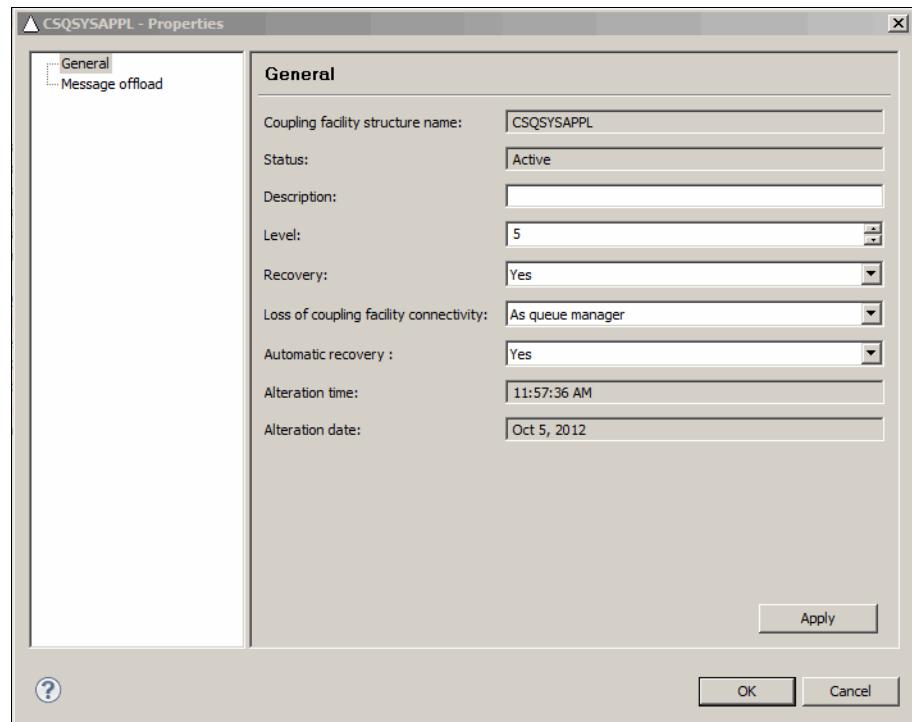


Figure 15-9 CFLEVEL 5 and Recovery Yes which supports persistent messages

- b. Verify that SYSTEM.QSG.UR.RESOLUTION.QUEUE is indexed by correlation identifier, as shown in Figure 15-10.

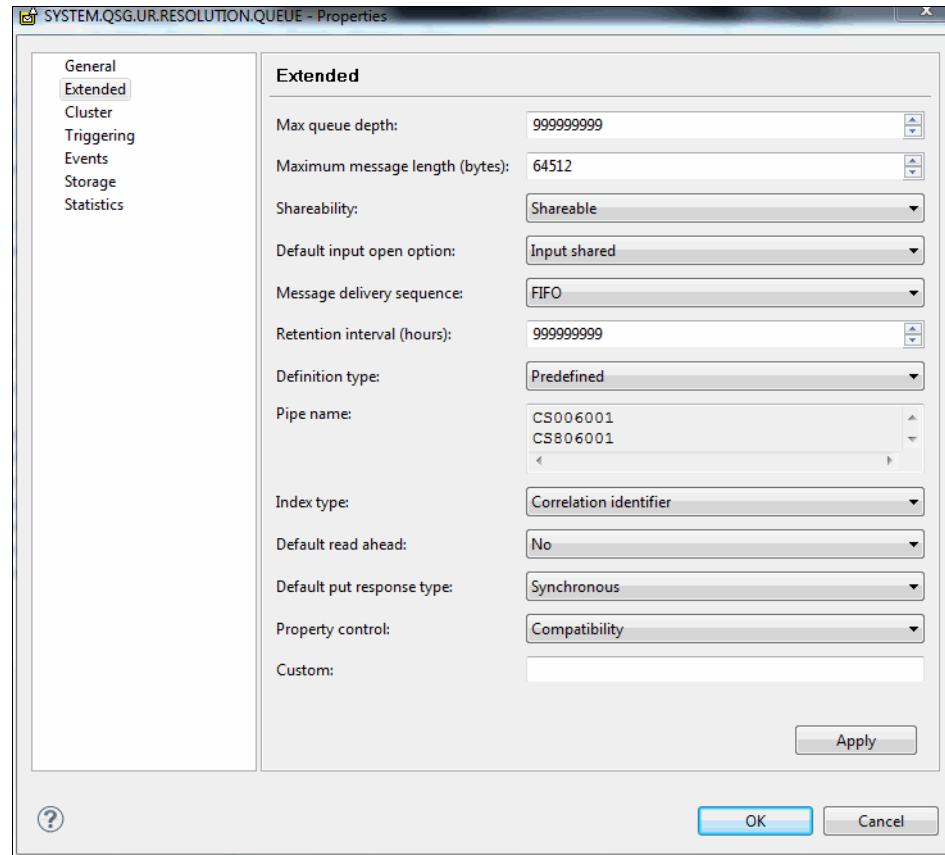


Figure 15-10 WebSphere MQ Explorer Queue extended panel

- c. Verify that SYSTEM.QSG.UR.RESOLUTION.QUEUE is in the system application coupling facility structure called CSQSYSAPPL, as shown in Figure 15-11.

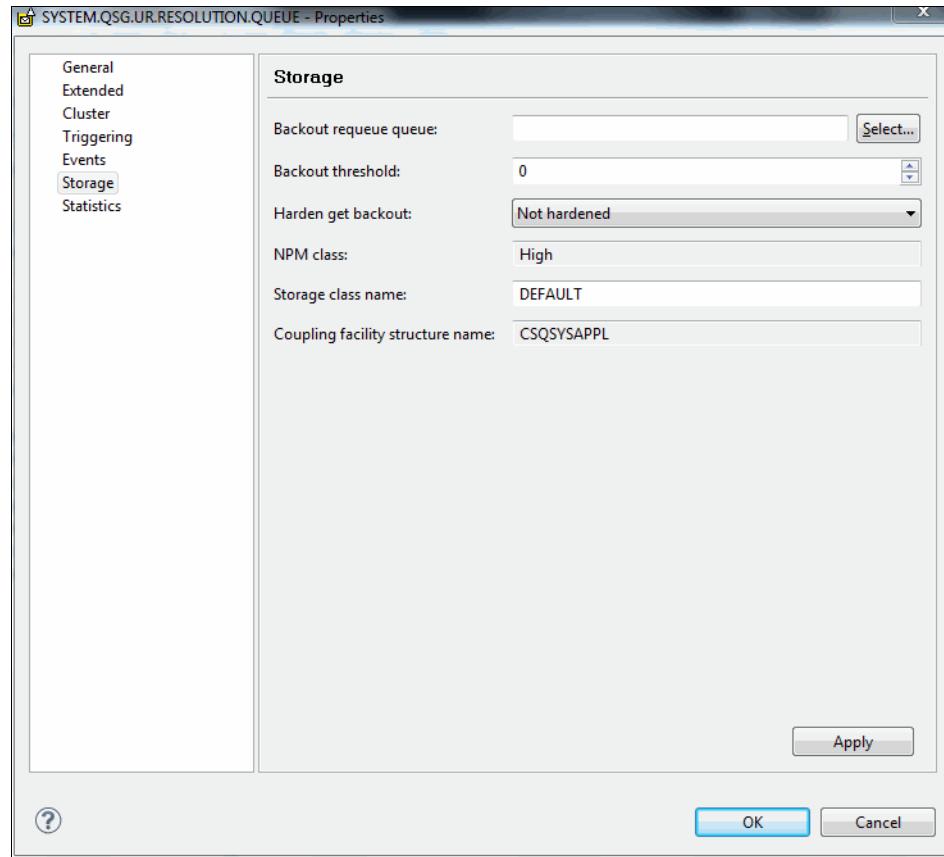


Figure 15-11 WebSphere MQ Explorer queue storage panel

3. Verify that the queue manager name is not the same as the queue-sharing group, as shown in Figure 15-12.

The screenshot shows the WebSphere MQ Explorer interface. The left sidebar navigation tree includes:

- Queue Managers: CSQ1, CSQ2, CSQ3, CSQ4, CSQ5
- Queue-sharing Groups: IBM1 (selected)
- Shared Queues
- Coupling Facility Structures
- Group Definitions
- Queue Manager Clusters
- MS Administered Objects
- Managed File Transfer
- Service Definition Repositories

The main right pane is titled 'Queue-sharing Group IBM1' and contains two sections:

- Queue-sharing group information source:**

Queue manager name	CSQ1
Description	IBM WebSphere MQ for z/OS - V7.1.0
Command level	710
QMID	CSQ1.C9DB8780CBFD7CA7
Coded character set ID	500
- Members of queue-sharing group:**

Queue manager name	Queue manager number	DB2 name	Queue manager status	DB2 c
CSQ1	1	D911	Active	Active
CSQ2	2	D912	Active	Active
CSQ3	3	D911	Active	Active

A status message at the bottom says 'Last updated: 11:28:22'.

Figure 15-12 Queue manager and queue-sharing group names

- Verify that the operational mode of the queue manager supports Group units of recovery.

The operational mode is configured in the system parameter section of the queue manager's ZPARM, as shown in the following example:

```
OPMODE=(NEWFUNC,710),
```

Important: The OPMODE attribute includes two parts. The first part sets the functionality, such as NEWFUNC. NEWFUNC enables all the new functions in v710. The second part refers to the queue manager version operational level. In this case, the level is 710 which is the current level.

Figure 15-13 shows the OPMODE attribute setting in CSQ6SYSP.

//SYSIN DD *		
CSQ6SYSP		X
CLCACHE=STATIC,	CLUSTER CACHE TYPE	X
CMDUSER=CSQOPR,	DEFAULT USERID FOR COMMANDS	X
EXITLIM=30,	EXIT TIMEOUT (SEC)	X
EXITTCB=8,	NUMBER OF EXIT SERVER TCBS	X
LOGLOAD=500000,	LOG RECORD CHECKPOINT NUMBER	X
OPMODE=(NEWFUNC,710), new function mode		X
OTMACON=(,,DFSYDRUO,2147483647,CSQ), OTMA PARAMETERS		X
QINDEXBLD=WAIT,	QUEUE INDEX BUILDING	X
QMCCSID=0,	QMGR CCSID	X
QSGDATA=(IBM1,DB9IU,D9IG,4), DATASHARING GROUP DATA		X
RESAUDIT=YES,	RESLEVEL AUDITING	X
ROUTCDE=1,	DEFAULT WTO ROUTE CODE	X
SMFACCT=NO,	GATHER SMF ACCOUNTING	X
SMFSTAT=NO,	GATHER SMF STATS	X
STATIME=30,	STATISTICS RECORD INTERVAL (MIN)	X
TRACSTR=YES,	TRACING AUTO START	X

Figure 15-13 TSO display of CSQ6SYSP of the ZPRM member

- By using WebSphere MQ Explorer, set GROUPUR to enabled.
- Verify that Group units of recovery were enabled. The following message is displayed in the queue manager log:

```
CSQM504I -CSQ1 CSQMCAMM GROUPUR support enabled
```

Figure 15-14 shows that the GROUPUR is enabled. In WebSphere MQ Explorer, the GROUPUR attribute is identified by Group units of recovery.

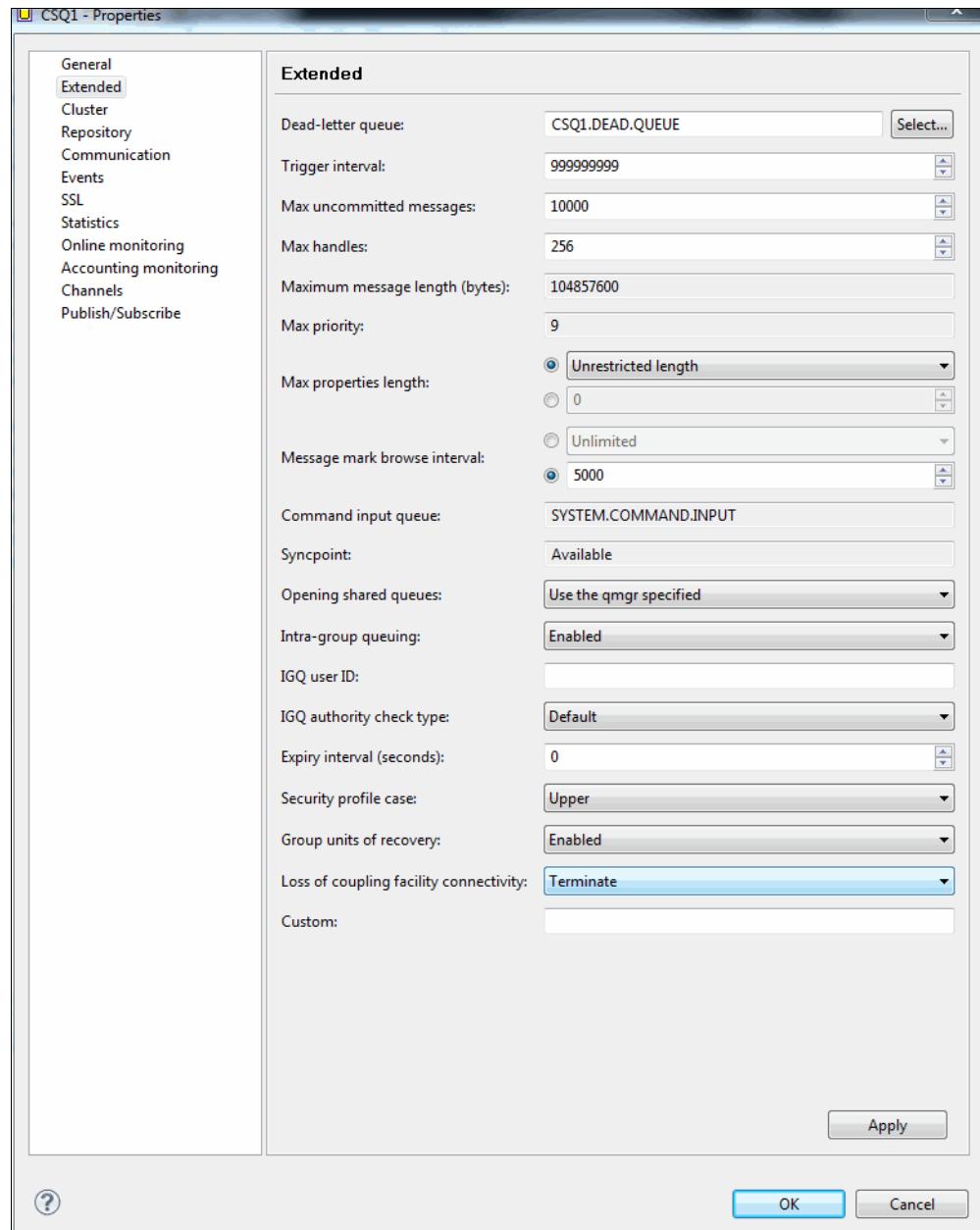


Figure 15-14 WebSphere MQ Explorer queue manager Extended panel

If Group units of recovery cannot be enabled by using WebSphere MQ Explorer, you receive an error message.

Figure 15-15 shows the error message that is generated.

Important: The following message was generated for a different queue manager than the previous displays to generate failed messages.

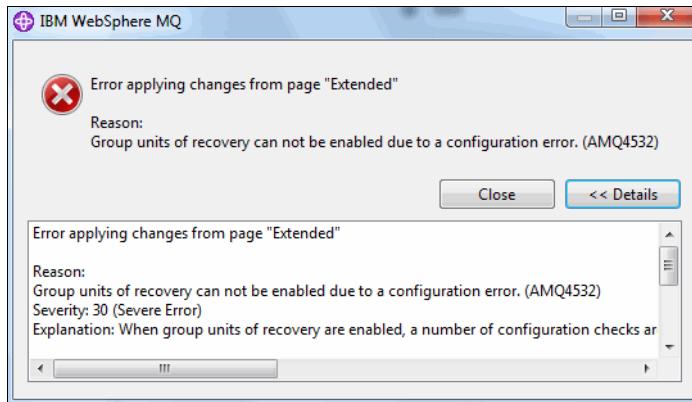


Figure 15-15 WebSphere MQ Explorer error display

WebSphere MQ 7.5 new features and enhancements

WebSphere MQ V7.5 introduces some new installation options, including the bundling of WebSphere MQ Advanced Message Security and WebSphere MQ Managed File Transfer. WebSphere MQ V7.5 also introduces an enhancement to the configuration and control of message traffic around a WebSphere MQ clustered environment.

This part describes these features and enhancements and how to use them.

This part consists of the following chapters:

- ▶ Chapter 16, “Installation enhancements in WebSphere MQ V7.5” on page 231
- ▶ Chapter 17, “Clustering enhancements on Windows, UNIX, and Linux” on page 237
- ▶ Chapter 18, “Certificate validation policies” on page 249



Installation enhancements in WebSphere MQ V7.5

This chapter describes the new installation options that are available when WebSphere MQ V7.5 is installed on Windows, UNIX, or Linux. The features that are described here are the inclusion of WebSphere MQ Advanced Message Security and WebSphere MQ Managed File Transfer as installation options with the base product, and the inclusion of the extended transactional client with the main client installation package.

This chapter contains the following sections:

- ▶ WebSphere MQ Advanced Message Security installation
- ▶ WebSphere MQ Managed File Transfer installation
- ▶ Extended transactional client installation

16.1 WebSphere MQ Advanced Message Security installation

WebSphere MQ Advanced Message Security (AMS) is available as a stand-alone extension to WebSphere MQ V7.0.1 and V7.1. With the introduction of WebSphere MQ V7.5, AMS is now available as an integrated, installable (chargeable) feature. This section provides a brief overview of AMS and describes the process of how to install it. For more information about the use of AMS in secure environments, see *Secure Messaging Scenarios with WebSphere MQ*, SG24-8069.

16.1.1 WebSphere MQ Advanced Message Security overview

WebSphere MQ provides transport-level security with features such as SSL/TLS over channels. However, by default, WebSphere MQ does not provide a method to encrypt and secure access to messages while they are at rest on queues.

If AMS is used in a WebSphere MQ environment, it is now possible to implement full end-to-end security from a sending application to a receiving application with a single installation.

16.1.2 Installing AMS with WebSphere MQ V7.5

When WebSphere MQ V7.5 is used, it is possible to install AMS at the same time as the base product in one integrated process. Figure 16-1 shows the Windows installation program. The sections that follow describe the package names that are used to install the AMS component on UNIX and Linux.

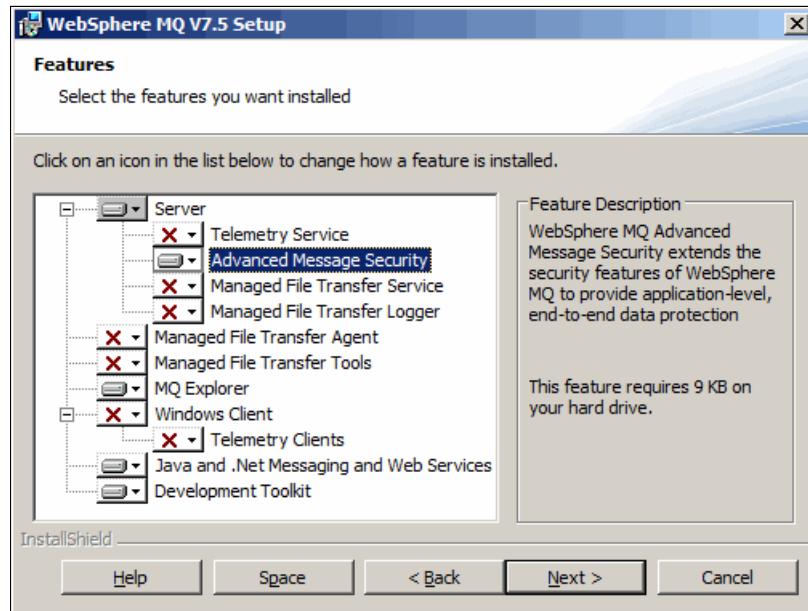


Figure 16-1 Selecting AMS from the WebSphere MQ V7.5 installation program

After the WebSphere MQ V7.5 installation program is started and the license agreement is accepted, the options to determine how to proceed with the installation are displayed. The first set of options is to install the Typical components, the Compact components (the minimum set required), or to proceed with the Custom option that allows the user to select exactly which components to install. Select **Custom** and click **Next**, which accepts the default settings until the Features window is shown. Figure 16-1 on page 232 shows the selection of AMS from the WebSphere MQ V7.5 installation options.

The following names of the AMS packages on UNIX and Linux are used:

- ▶ AIX: mqm.ams.rte
- ▶ HP-UX: MQSERIES.MQM-AMS
- ▶ Linux: MQSeriesAMS
- ▶ Solaris: mqams

16.2 WebSphere MQ Managed File Transfer installation

Managed File Transfer (MFT) is available as an additional product that states a prerequisite on a WebSphere MQ V7 queue manager. WebSphere MQ V7.5 includes MFT as an integrated (chargeable) installable option. This section provides an overview of the MFT feature and the details of how to install it with WebSphere MQ V7.5. For more information about the use of MFT, see *Getting Started with WebSphere MQ File Transfer Edition V7*, SG24-7760.

16.2.1 WebSphere MQ Managed File Transfer overview

MFT provides a mechanism to build and automate a secure environment within which files can be transferred between systems, regardless of operating system or file size. For more information about the features of the product and links to detailed information about how MFT works, see the topic *WebSphere MQ Managed File Transfer introduction* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.wmqfte.doc/wmqfte_intro.htm

In addition to the interoperation between MFT and WebSphere MQ, MFT interoperates with other IBM products, such as WebSphere Message Broker.

16.2.2 Installing MFT with WebSphere MQ V7.5

When WebSphere MQ V7.5 is used, it is now possible to install MFT at the same time as the base product in one integrated process. Figure 16-2 on page 234 and Figure 16-3 on page 234 show the Windows installation program. The sections that follow describe the package names that are used to install the MFT components on UNIX and Linux.

After the WebSphere MQ V7.5 installation program is started and the license agreement is accepted, the options that relate to how to proceed with the installation are displayed. The first set of options is to install the Typical components, the Compact components (the minimum set required), or to proceed with the Custom option that allows the user to select exactly which components to install. Select **Custom** and click **Next**, which accepts the defaults until the Features panel is shown. The Windows MFT features are listed in two sections, but each can be installed independently of one another.

For more information about the components, including where each can be installed, see the *WebSphere MQ Managed File Transfer product options* topic in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.wmqfte.doc/product_options.htm

Figure 16-2 shows that the two features (the MFT service and the MFT logger) that are listed under the Server section are selected for installation.

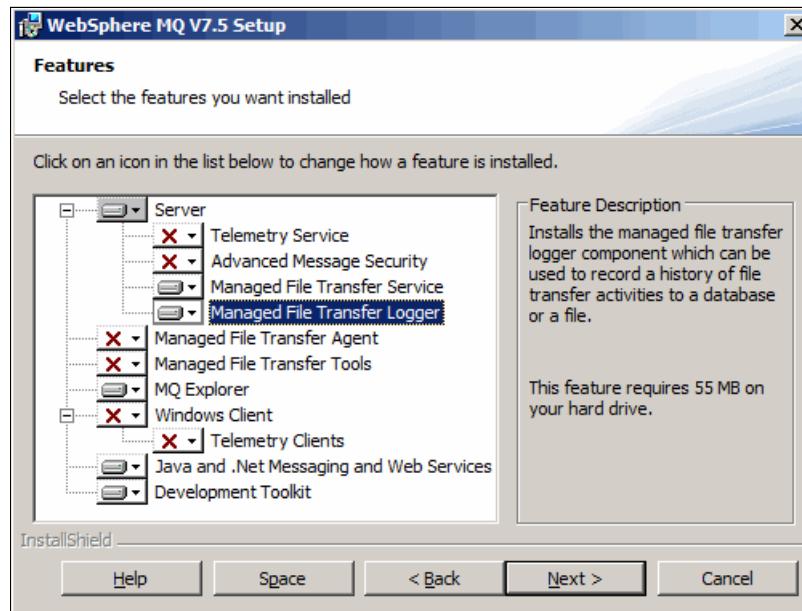


Figure 16-2 The service and logger components that are selected under Server

Figure 16-3 shows the two components that are listed as stand-alone components of MFT, the MFT agent and the MFT tools.

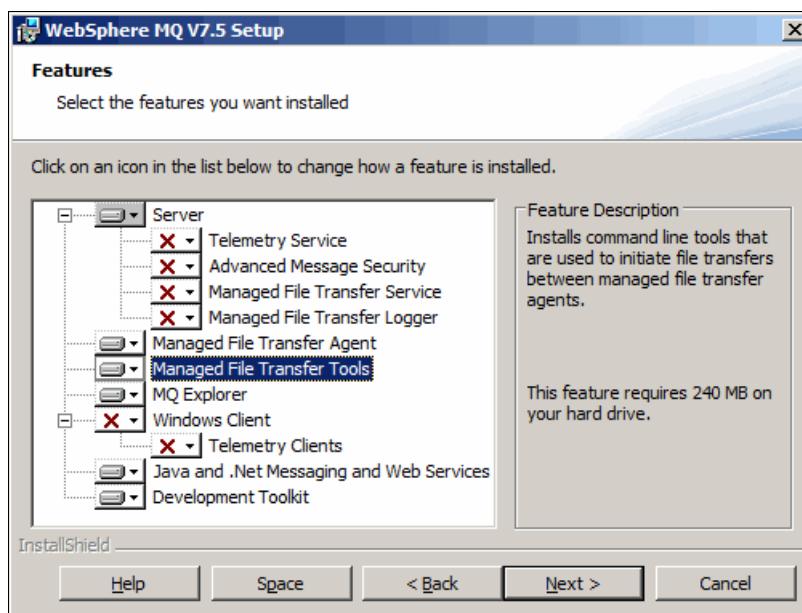


Figure 16-3 The agent and tools components that are selected as stand-alone components

The following features of MFT can be installed separately on UNIX and Linux:

- ▶ MFT base
- ▶ MFT agent
- ▶ MFT logger
- ▶ MFT service
- ▶ MFT tools

Base component: The base component contains files common to all other components so must be installed before any of the other features of MFT on UNIX and Linux systems. The base component is installed automatically on Windows and is not listed as an option.

The following package names for the installable components on each platform are used:

- ▶ AIX:
 - mqm.ft.agent
 - mqm.ft.base
 - mqm.ft.logger
 - mqm.ft.service
 - mqm.ft.tools
- ▶ HP-UX:
 - MQSERIES.MQM-FTAGENT
 - MQSERIES.MQM-FTBASE
 - MQSERIES.MQM-FTLOGGER
 - MQSERIES.MQM-FTSERVICE
 - MQSERIES.MQM-FTTOOLS
- ▶ Linux:
 - MQSeriesFTAgent
 - MQSeriesFTBase
 - MQSeriesFTLogger
 - MQSeriesFTService
 - MQSeriesFTTools
- ▶ Solaris:
 - ftagent
 - ftbase
 - ftlogger
 - ftservice
 - fttools

16.3 Extended transactional client installation

In versions of WebSphere MQ before V7.5, the option to install the WebSphere MQ Client package is split into the base client package and a chargeable subcomponent that represents the extended transactional functionality addition to the base features. Figure 16-4 shows the Features panel from the WebSphere MQ V7.1 installation program, which specifically lists the Client Extended Transaction Support.

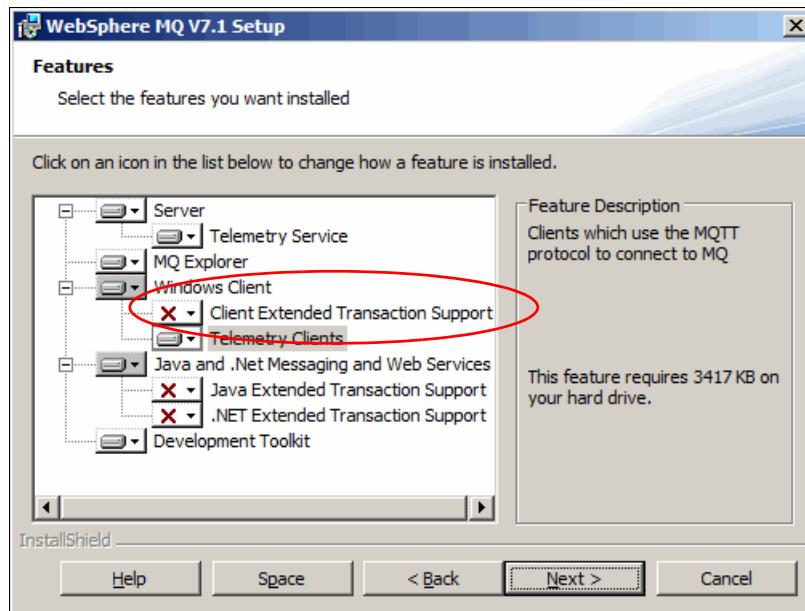


Figure 16-4 WebSphere MQ V7.1 installation that lists the Client Extended Transaction Support option

When WebSphere MQ V7.5 is used, the installation features do not include the extended transaction support as an explicit option because this function is included in the base client package. Figure 16-2 on page 234 and Figure 16-3 on page 234 show the V7.5 installation program no longer lists the extended transaction support option. This feature is installed automatically when the base client component is installed.

For more information about the function that is provided by the Extended Transactional Client, see the topic *What is an extended transactional client?* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/cs10270_.htm



Clustering enhancements on Windows, UNIX, and Linux

This chapter introduces and describes the enhancements that were made to message processing in a clustered environment when WebSphere MQ V7.5 is used on Windows, UNIX, and Linux. The enhancements include configuration options that link cluster channels with specific transmission queues rather than using a single queue for all traffic.

This chapter contains the following sections:

- ▶ Clustering enhancements overview
- ▶ Enabling multiple cluster transmission queues
- ▶ Using the same transmission queue for multiple channels
- ▶ Priority of queue selection
- ▶ Switching the transmission queue that is used by a channel
- ▶ Switching channel transmission queues with runswchl
- ▶ Displaying the transmission queue that is used
- ▶ Reverting to the default configuration

17.1 Clustering enhancements overview

This section describes the clustering enhancements in WebSphere MQ V7.5. These enhancements apply to Linux, UNIX, and Windows.

It is important to understand the basic concepts of clustering and their implementation in WebSphere MQ to fully appreciate the enhancements to cluster support in WebSphere MQ V7.5. For more information about what a cluster is and why they are used, see 3.2.7, “WebSphere MQ topologies”.

17.1.1 Overview of changes to clustering

WebSphere MQ sends messages between queue managers by using communication links called *channels*. To insulate applications from network outages or degradations, messages that are to be sent over the network are temporarily held on a special type of queue, called a *transmission queue*. WebSphere MQ confirms receipt of a message after it is stored on a transmission queue. The sending application can then continue with its own processing, leaving responsibility for delivering the message to the queue manager. The transmission queue, from which messages should be retrieved to be sent over the network, is specified in the definition for each channel. The only exception is for cluster sender channels, which in WebSphere MQ V7.1, or earlier, all use the default transmission queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE. This transmission queue is used to hold all messages that are to be sent within the cluster, irrespective of their destination.

WebSphere MQ V7.5 includes the capability to create and use multiple cluster transmission queues. Each cluster sender channel can link to its own transmission queue or share a transmission queue with a group of cluster sender channels.

17.1.2 Benefits of multiple transmission queues

The use of multiple transmission queues includes the following benefits:

- ▶ Administrators can manage network traffic more effectively by separating it in a way that suits the system. For example, network traffic for a specific application can now have its own transmission queue, which minimizes the impact on other applications.
- ▶ Monitoring is simplified because the network traffic for each channel or application can be isolated to an exclusive transmission queue.
- ▶ The likelihood of a backlog of messages on the transmission queue is reduced. The lack of a backlog prevents hitting the MAXDEPTH on the transmission queue if one or more queue managers became unavailable as message traffic can be distributed over many transmission queues.

17.2 Enabling multiple cluster transmission queues

The following methods were introduced in WebSphere MQ V7.5 to enable multiple cluster transmission queues:

- ▶ Dynamic creation of a transmission queue for each channel that is enabled by a new queue manager attribute.
- ▶ Manual creation of a transmission queue to be used by one or more channels that is configured by using a new queue attribute.

Both methods are described in the following sections.

17.2.1 Creating cluster transmission queues dynamically

A new queue manager attribute was implemented in WebSphere MQ V7.5 to enable multiple cluster transmission queues. The new attribute is referred to as DEFCLXQ in MQSC and MQIA_DEF_CLUSTER_XMIT_Q_TYPE in Programmable Command Formats (PCFs). This attribute includes the values that are shown in Table 17-1.

Table 17-1 DEFCLXQ options

	MQSC	PCF
Default (Off)	SCTQ	MQCLXQ_SCTQ
Dynamic queues enabled (On)	CHANNEL	MQCLXQ_CHANNEL

All commands that are used in this chapter are MQSC commands. For more information about PCF, see the WebSphere MQ Information Center topic *DefClusterXmitQueueType (MQLONG)* at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/fr19475_.htm

If the DEFCLXQ attribute is set to SCTQ, all cluster sender channels use SYSTEM.CLUSTER.TRANSMIT.QUEUE, unless they are configured to use a manually defined transmission queue. For more information, see 17.2.2, “Defining cluster transmission queues manually” on page 241.

The use of the default transmission queue SYSTEM.CLUSTER.TRANSMIT.QUEUE has the same behavior as versions of WebSphere MQ before V7.5. Setting DEFCLXQ to CHANNEL on a queue manager means that when any cluster sender channel starts for the first time after the change, it uses its own dynamically created transmission queue.

Important: Transmission queues are selected based on priority rules. For more information about the selection process, see 17.4, “Priority of queue selection” on page 245.

To enable dynamic multiple cluster transmission queues by using the `runmqsc` command with a queue manager, run the following command:

`ALTER QMGR DEFCLXQ(CHANNEL)`

To revert to single transmission queue usage, enter the following command:

`ALTER QMGR DEFCLXQ(SCTQ)`

Complete the following steps to make the same configuration change by using WebSphere MQ Explorer:

1. Start WebSphere MQ Explorer, which is in the <mqinstallation_directory>\bin directory called MQExplorer; for example, C:\Program Files (x86)\IBM\WebSphere MQ\bin. Use the systems GUI to browse for the application or start by using the console.
2. When WebSphere MQ Explorer is started for the first time, the Welcome window is displayed. To close it, click X. The standard WebSphere MQ Explorer window is displayed. The next time WebSphere MQ Explorer is started, it opens on the WebSphere MQ perspective, as shown in Figure 17-1.

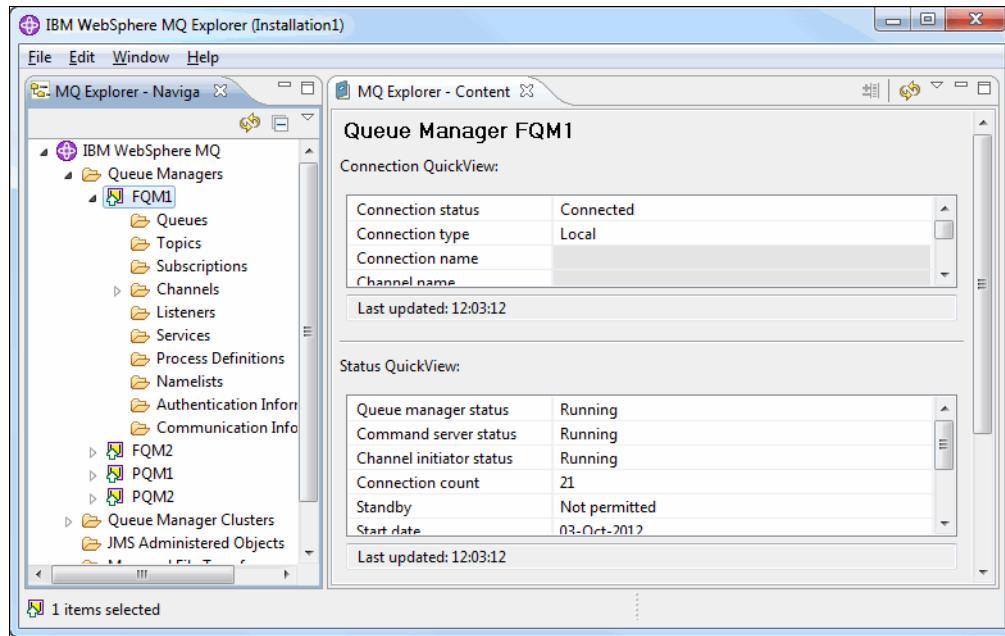


Figure 17-1 WebSphere MQ Explorer home window

3. Right-click the queue manager and select **Properties** → **Cluster**. The clustering properties dialog for the queue manager is displayed.

4. In WebSphere MQ V7.5, the default cluster transmission queue drop-down includes two options: SYSTEM.CLUSTER.TRANSMIT.QUEUE (SCTQ) and the new option Queue for each channel (CHANNEL), as shown in Figure 17-2.

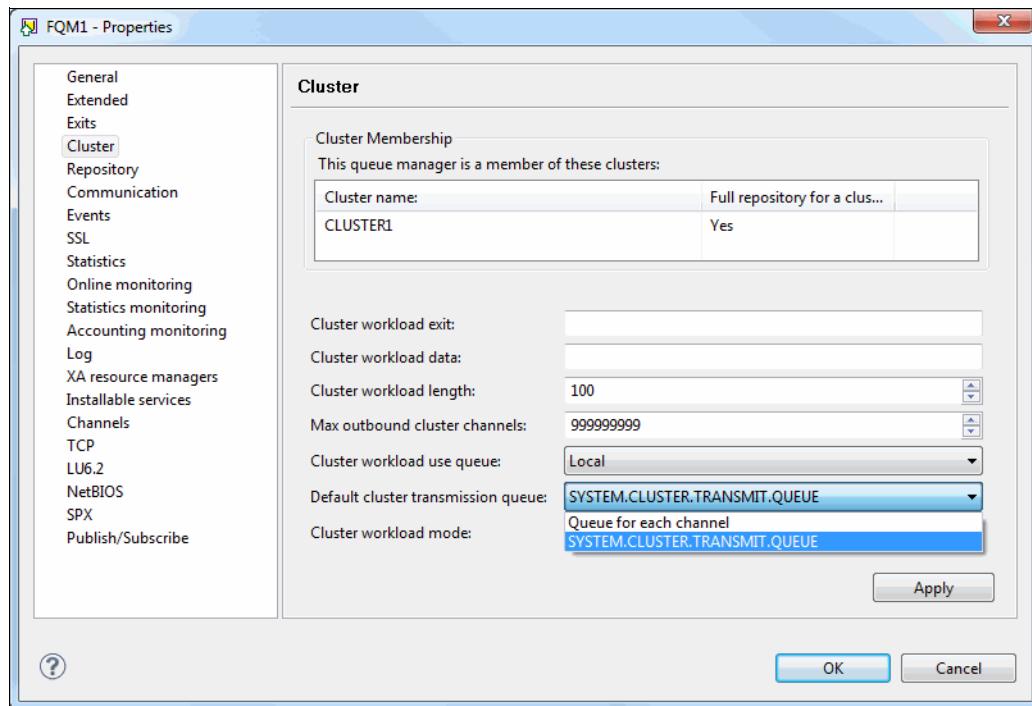


Figure 17-2 WebSphere MQ Explorer default cluster transmission queue options

Select **Queue for each channel** and click **OK**.

5. Activate the changes. The process of activating these changes in the cluster is the same as when the command line is used: restart the channel (or channels) or use the runswch1 application, as described in 17.6, “Switching channel transmission queues with runswch1” on page 246.

When a channel starts for the first time after DEFCLXQ is set to CHANNEL, the channel creates its own transmission queue. It follows the naming format of SYSTEM.CLUSTER.TRANSMIT.<cluster-sender-channel-name>. The transmission queue is defined by using the model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.

17.2.2 Defining cluster transmission queues manually

The second method to create a transmission queue to be used by one or more channels is to manually configure a transmission queue that is to be linked to one or more cluster sender channels. To allow transmission queues for cluster sender channels to be manually defined, there is a new attribute on local queues called CLCHNAME. Any queue that is to be used for this purpose needs its usage to be set to XMITQ so that it works as a transmission queue; otherwise, error AMQ8872 is returned. The transmission queue cannot be used for a cluster sender channel and a non-cluster channel at the same time.

To manually define a queue in runmqsc for a particular queue manager, run the following command:

```
DEFINE QLOCAL(<queuename>) USAGE(XMITQ) CLCHNAME(<cluster-sender-channel-name>)
```

The QLOCAL(<queueusername>) is the name of the manually defined queue that you are creating. USAGE(XMITQ) sets the queue to be a transmission queue and CLCHNAME(<cluster-sender-channel-name>) identifies the channel or channels to which the queue is linked. The next time that any applicable channels start, the new queue is linked as a transmission queue to it. You can use the runswch1 application that is described in 17.6, “Switching channel transmission queues with runswch1” on page 246 to perform the switch while the channel is stopped.

When SYSTEM.CLUSTER.TRANSMIT.QUEUE is used, the attributes for MAXDEPTH and MAXMSGL are set high. This setting is worth mirroring when you are manually defining a transmission queue. MAXDEPTH is 999999999 and MAXMSGL is 4194304 on the SYSTEM.CLUSTER.TRANSMIT.QUEUE. Changing these values ensures any manually defined queue features the same limits as the transmission queue it is replacing.

Complete the following steps to manually define a transmission queue for a cluster sender channel by using WebSphere MQ Explorer:

1. Start WebSphere MQ Explorer and expand your queue manager to display the list of objects, as shown in Figure 17-3.

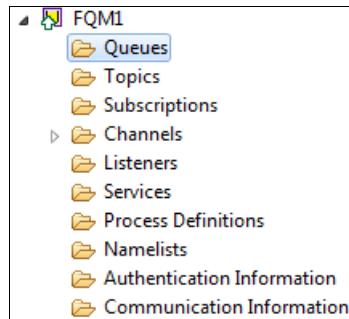


Figure 17-3 WebSphere MQ Explorer queue manager objects

2. Right-click **Queues** and select **New → Local Queue...**

3. Enter the name for the new transmission queue. Click **Next** to continue. The Create a Local Queue window is shown in Figure 17-4.

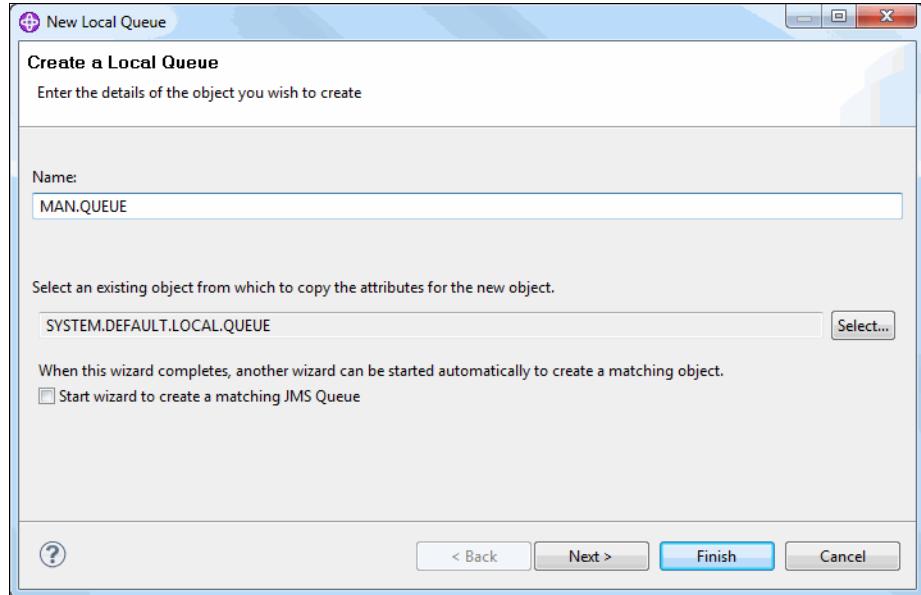


Figure 17-4 WebSphere MQ Explorer new queue dialog

4. The Change properties dialog is displayed. Click **Extended** and scroll to the bottom to enter a Cluster channel name. The Change properties window is shown in Figure 17-5.

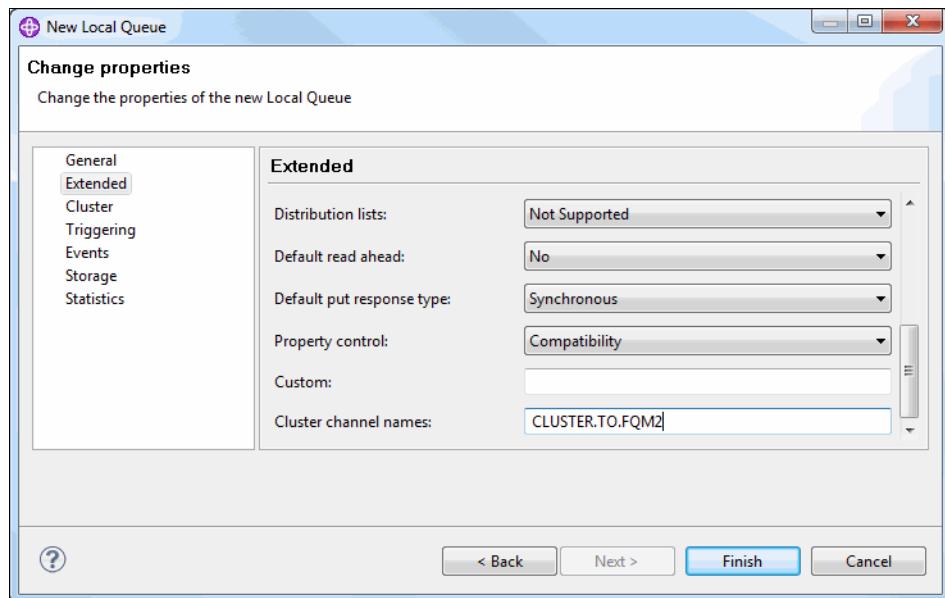


Figure 17-5 WebSphere MQ Explorer queue properties dialog

5. Click **Finish**.

The next time the chosen channel (or channels) are started, they use the manually defined queue.

17.3 Using the same transmission queue for multiple channels

A manually defined transmission queue can be used for more than one channel by using pattern matching. For example, if one queue manager is a member of two clusters, there can be a separate transmission queue for each cluster's group of cluster sender channels that the queue manager is using to communicate with the other queue managers. With two clusters (called CLUSTER1 and CLUSTER2), and channels called CLUSTER1.<queuemanger> and CLUSTER2.<queuemanager>, it is possible to have one transmission queue for each cluster by changing the CLCHNAME of two local transmission queues to CLUSTER1.* and CLUSTER2.*.

Another example is a cluster that consists of four queue managers. The first queue manager sends messages to the other three members of the cluster. Two of those queue managers are for one application (PQM1 and PQM2), and the other queue manager (FQM2) is for a second application. Traffic is separated by creating one local transmission queue that is linked to the cluster sender channels that send to PQM1 and PQM2, as shown in the following example:

```
DEFINE QLOCAL(APPLICATION1) USAGE(XMITQ) CLCHNAME(CLUSTER.PQM*)
```

FQM2 can include its own transmission queue, as shown in the following example:

```
DEFINE QLOCAL(APPLICATION2) USAGE(XMITQ) CLCHNAME(CLUSTER.FQM2)
```

The cluster configuration is shown in Figure 17-6 (the channel names are shown on the cluster sender channels).

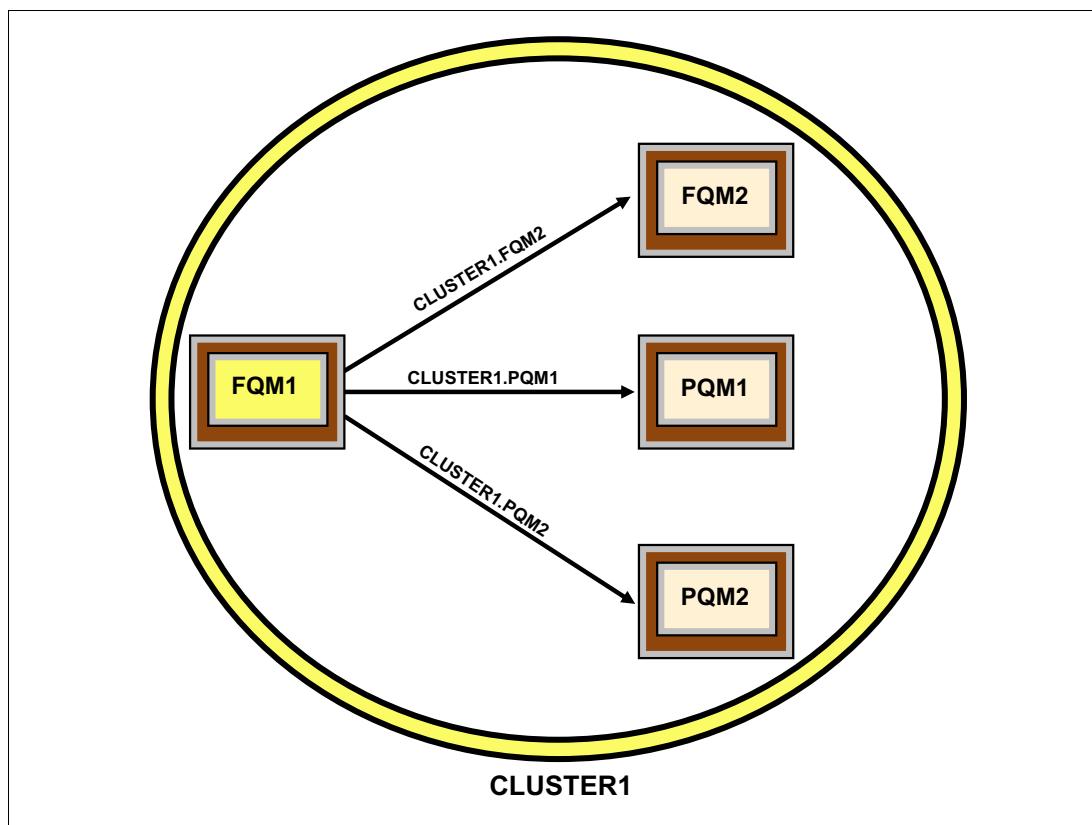


Figure 17-6 Simple cluster with multiple transmission queue setup

17.4 Priority of queue selection

Multiple transmission queues can be defined that all apply to a particular channel, but each channel can use only one transmission queue. This configuration means that when a channel starts, it must determine which transmission queue to use. The following order of priority to select the transmission queue, starting with the highest priority, is used:

1. Any transmission queue that explicitly specifies the exact channel name in the CHCLNAME attribute.
2. A transmission queue with a pattern that matched CLCHNAME. If there is more than one queue with a matching pattern, the most specific is chosen. For example, if a channel is called CLUSTER.QMA, and there are two queues that use different CLCHNAME values, such as CLUSTER* and CLUSTER.QM*. The second queue is selected.
3. The queue is selected based on the DEFCLXQ setting, which can be SYSTEM.CLUSTER.TRANSMIT.QUEUE or a dynamically created transmission queue.

Important: Remember that when you are manually creating queues, if two queues explicitly point to the same channel, you cannot be 100% sure which queue is chosen. It is best to avoid this situation. The same advice applies for the same level of specificity on pattern matched CLCHNAME attributes. While the messages still arrive on the correct queue manager, applications might be monitoring the incorrect queue.

17.5 Switching the transmission queue that is used by a channel

When a cluster sender channel starts, it queries the queue manager to determine whether it must change the transmission queue. If no change is required, it starts as normal. If there was a change in the configuration and the channel is configured to use an alternative transmission queue, the following switching process begins:

1. The channel starts by using the new transmission queue immediately, but new messages continue to be put to the original queue. A background process is started to move messages, in order, from the original queue to the new queue.

Important: If many messages are put to the old transmission queue while the switch is taking place, this process might continue for a long time because the old transmission queue cannot have any messages waiting on it for the final stage of the switch over to occur. Messages are moved in a first in, first out order.

2. When no messages remain on the original transmission queue, the new transmission queue is used for new messages.

When transmission queues are switched, a message is written to the queue manager error log so there is a record of the switch happening. When messages are moved from one transmission queue to another, this event also is logged to the queue manager.

17.6 Switching channel transmission queues with runswchl

There is a way to move messages to a new transmission queue without the need to start the channel if it is stopped. This ability is useful for administrators who might want to make this configuration change for their cluster during downtime. In WebSphere MQ V7.5, a new command that is called **runswchl** is provided to switch or query the cluster transmission queues that are associated with cluster-sender channels. For a working example of how to perform manual changes, see 21.5, “Manually defining specific transmission queues with a manual switch” on page 327.

Example 17-1 shows the parameters that are available for the **runswchl** command to manually make the changes.

Example 17-1 Display of the runswchl parameters

```
C:\>runswchl  
Usage: runswchl -m QMgrName -c ChannelName [ -n | -q ]  
-m Queue manager name.  
-c Generic channel name.  
-q Query the current configuration.  
-n Do not move messages to the new transmission queue.
```

Example 17-2 shows the output that is returned by using the **-q** (query) parameter. Notice that there is a switch that is waiting to happen. The query also shows that there are two messages that are waiting to be moved over to the new transmission queue.

Example 17-2 How to query a change on a cluster sender channel

```
C:\>runswchl -m <qmgrname> -c <cluster-sender-channel-name> -q  
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.  
Switch to transmission queue '<new_transmission_queue>' is pending  
for channel '<cluster-sender-channel-name>'. The current transmission queue  
'<old_transmission_queue>' has 2 messages queued for the channel.
```

Running the **runswchl** command without the use of the **-q** parameter causes the messages to move over and the channel switch, as shown in Example 17-3.

Example 17-3 Making the change manually on a cluster sender channel

```
C:\>runswchl -m <qmgrname> -c <cluster-sender-channel-name>  
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.  
Message move complete - 2 messages moved.  
Cluster sender channel '<cluster-sender-channel-name>' successfully switched to  
use XMITQ  
'<new_transmission_queue>'.
```

Example 17-3 shows which transmission queue the cluster sender channel is now using and how many messages were moved over to it. When the channel starts, it automatically uses the new transmission queue and processes the messages that were transferred over to it.

When the **runswchl** command is used to perform a switch, the queue manager error log records that a switch was started with the moving of messages. The log also shows when the switch and message moving process was completed.

There is also a **-n** option that can be used to make a switch without moving any messages.

Warning: The use of the **-n** parameter might maroon messages on the old transmission queue. Therefore, use this option with care. The messages are not deleted, so a switch back enables them to be processed.

17.7 Displaying the transmission queue that is used

Use the **DISPLAY CHSTATUS** MQSC command to determine which transmission queue is used for a channel. The use of this command in runmqsc against your queue manager displays information about the channel, including the transmission queue that is used. Example 17-4 shows the XMITQ attribute with the transmission queue that is used.

Example 17-4 Displaying the channel status to see the XMITQ

```
DISPLAY CHSTATUS(<cluster-sender-channel-name>)
  1 : DISPLAY CHSTATUS(<cluster-sender-channel-name>)
AMQ8417: Display Channel Status details.
  CHANNEL(<cluster-sender-channel-name>)          CHLTYPE(CLUSSDR)
  CONNNAME(127.0.0.1(1431))           CURRENT
  RQMNAME(<qmgrname>)                STATUS(RUNNING)
  SUBSTATE(MQGET)                   XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

By using WebSphere MQ Explorer, you can view which transmission queue the cluster sender channels are using. Complete the following steps to view the transmission queue:

1. Start WebSphere MQ Explorer and expand the queue manager to show the list of objects.
2. Right-click **Channels** and select **Status → Channel Status...**

Figure 17-7 shows the transmission queue that the cluster sender channels are using.

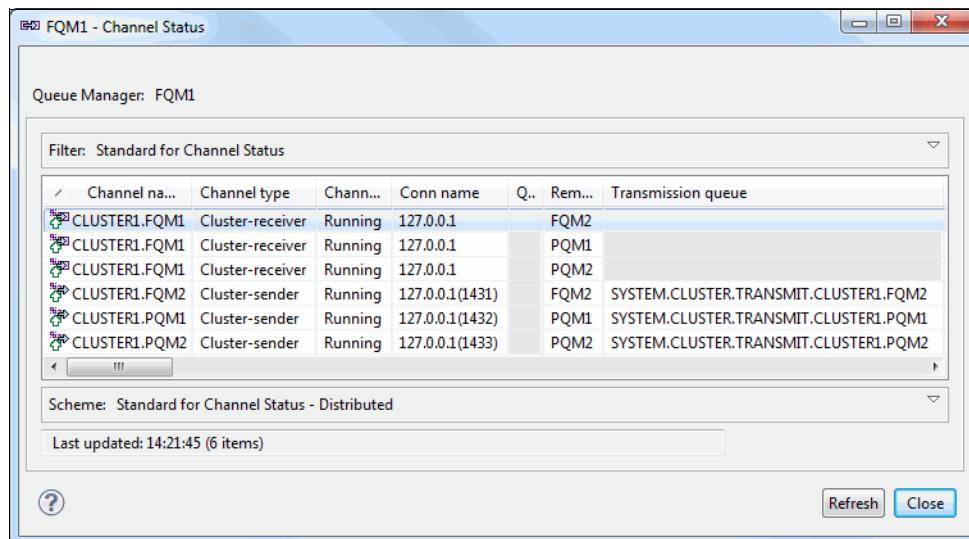


Figure 17-7 Channel Status display

17.8 Reverting to the default configuration

Complete the following steps to reset back to the default configuration:

1. Remove the configuration of any manually defined transmission queues by changing the value of their CLCHNAME attribute to blank. Alter the queue in runmqsc, as shown in the following example:

```
ALTER QLOCAL(<queueusername>) CLCHNAME('')
```

2. Change the value of the DEFCLXQ queue manager attribute to SCTQ, as shown in the following example:

```
ALTER QMGR DEFCLXQ(SCTQ)
```

Upon restarting, each channel switches back to using the SYSTEM.CLUSTER.TRANSMIT.QUEUE transmission queue as in WebSphere MQ V7.1 and earlier.



Certificate validation policies

Certificate validation is performed against the details of policies, which are described as the basic policy and the standard policy. The standard policy for certificate validation is described by RFC 5280. This chapter describes the basic and standard policies in terms of WebSphere MQ, specifically the enforcement of strict RFC 5280 compliance. Certificate validation policies are applicable to Windows, UNIX, and Linux only.

For more information about RFC 5280, see this website:

<http://datatracker.ietf.org/doc/rfc5280/>

This chapter contains the following sections:

- ▶ Digital certificate validation on Windows, UNIX, and Linux
- ▶ Configuration of strict RFC 5280 compliance

18.1 Digital certificate validation on Windows, UNIX, and Linux

Digital certificates features many uses regarding a secure environment. WebSphere MQ uses digital certificates on SSL/TLS-enabled channels. For more information about what digital certificates are and what they are used for, see the Information Center topic *Digital certificates*, which is available at this website:

<http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy1030.htm>

System security and digital certificates are susceptible to attack. During the establishment of an SSL/TLS-secured channel, WebSphere MQ validates the certificates that are in use, including the full certificate authority chain that is used to sign the certificate. The validation of the certificates is performed according to policies.

For more information about certificate chains, see the topic *How certificate chains work* in the Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy1000.htm>

The following types of policies are used to determine the validity of a digital certificate chain:

- ▶ Certificate policy: Determines which fields in a certificate are understood and processed.
- ▶ OCSP policy: Determines which fields in an online certificate status protocol (OCSP) request or response are understood and processed.
- ▶ CRL policy: Determines which fields in a certificate revocation list (CRL) are understood and processed.
- ▶ Path validation policy: Determines how the certificate, OCSP, and CRL policy types interact to determine whether the certificate chain is valid.

For more information about each of the policy types, see the topic *Certificate validation and trust policy design on UNIX, Linux, and Windows systems* in the Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12790.htm>

This section describes the policies that are used by WebSphere MQ V7.5 to perform the certificate validation, and their relationship to the RFC 5280 industry standard. The following sections describe the policy types in terms of basic policy and standard policy.

The following list of Information Center resources describe the basic and standard policies for the certificate, OCSP, and CRL type policies. In the case of the certificate, OCSP, and CRL policy types, the basic and standard policies are described together because they are the same as each other or the standard policy is an extension of the basic policy. The Information Center describes the few areas of difference between basic and standard policy for these types:

- ▶ *Basic and standard certificate policies*

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12800_.htm

- ▶ *Basic and standard OCSP policies*

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy13020_.htm

- ▶ *Basic and standard CRL policies*

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12810_.htm

Path validation policies differ in their implementation within WebSphere MQ V7.5 and are described separately in the following sections.

18.1.1 Basic path validation policy

For more information about the validation steps for a certificate chain by using only the basic path validation policy, see the topic *Basic path validation policy* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12820_.htm

The decision on the validity of the certificate chain is the result of a complex set of steps that examine the fields of the certificates that are used in a certificate chain. The basic policy does not provide the guarantee of a decision on the validity of the certificate chain. If a determination is not possible, the standard path validation policy is used.

18.1.2 Standard path validation policy

For more information about the validation steps for a certificate chain by using the standard path validation policy, see the topic *Standard path validation policy* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy12850_.htm

By default, the standard path validation policy is applied only if the basic policy fails to determine the validity of the certificate chain. Standard policy is a set of other stringent checks on the certificate chain. The performance implications of doing this check are such that the fewer checks that are required to validate a certificate chain, the faster the channel can be up and running. However, certificate validation is an important part of ensuring a secure environment and as such, performance in this area is often traded for diligent verification of entities.

The use of standard policy in WebSphere MQ V7.5 for certificate validation implies conformance to RFC 5280.

18.1.3 What changed in WebSphere MQ V7.5

Certificate validation is a part of SSL/TLS channel establishment since WebSphere MQ V6.0.1. The application of standard policy for certificate validation previously was in compliance with RFC 3280, and the ability to enforce compliance with the standard by bypassing the basic validation was not an option.

Why bypass the basic policy validation?

For examples of certificates that pass the validation check of the basic policy but fail the standard policy validation, see the topic *Certificate validation and trust policy design on UNIX, Linux, and Windows systems* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy1290_.htm

Because the basic policy is applied first and, if the certificate chain is deemed valid, the standard policy is not applied, there is the potential for deeming a certificate chain valid purely because it was not put through the necessary checks to deem it invalid. Bypassing the basic validation check allows the full set of validation checks that are supported by WebSphere MQ to be performed every time.

Strict RFC 5280 compliance in WebSphere MQ 7.5

WebSphere MQ V7.5 introduces compliance with RFC 5280, which supersedes RFC 3280 and enables a configuration option to enforce compliance always.

For more information about the options that are available for configuring RFC 5280 compliance (depending on the environment), see topic *Configuring certificate validation policies in WebSphere MQ* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/sy1026_.htm

18.2 Configuration of strict RFC 5280 compliance

Because the standard policy is applied by default only if the basic policy fails to determine compliance without bypassing the basic policy, it is not possible to enforce strict compliance with RFC 5280. This section describes the configuration options that are available for queue managers and client applications such that they are able to enforce strict compliance with RFC 5280.

18.2.1 Queue manager configuration

A new queue manager attribute controls whether strict enforcement of RFC 5280 compliance is applied to certificate chains on SSL/TLS channels. This attribute can be configured by using any of the following methods:

- ▶ MQSC
- ▶ PCF
- ▶ WebSphere MQ Explorer

MQSC

When the **runmqsc** command is used to configure certificate validation policies, the MQSC keyword CERTVPOL is used to define the compliance and includes the following possible values:

- ▶ ANY

This value is the default value and represents applying the basic policy first and applying the standard policy only if the basic policy fails to determine validity.

- ▶ RFC5280

This value bypasses the basic policy and performs only the standard policy validation. Because the only way this process can succeed is by passing the standard path validation policy, this value enforces strict compliance with RFC 5280.

The queue manager is configured to enable strict compliance with RFC 5280 by using the following command:

```
alter qmgr CERTVPOL(RFC5280)
```

PCF

The use of PCF messages to configure the queue manager requires the use of the MQIA_CERT_VAL_POLICY parameter identifier, which includes the following possible values when PCF messages are used:

- ▶ MQ_CERT_VAL_POLICY_ANY

This value is the default value and represents applying the basic policy first and applying only the standard policy if the basic policy fails to determine validity.

- ▶ MQ_CERT_VAL_POLICY_RFC5280

This value bypasses the basic policy and performs only the standard policy validation. Because the only way this process can succeed is by passing the standard path validation policy, this value enforces strict compliance with RFC 5280.

WebSphere MQ Explorer

Configuring a queue manager attribute by using the WebSphere MQ Explorer is performed by opening the queue manager properties panel. Right-click the queue manager name and select **Properties...**. In the SSL section of the properties, the field Certificate validation policy can be set to ANY or RFC5280. Setting this field to RFC5280 enforces strict compliance with the standard. Figure 18-1 shows the queue manager panel from the WebSphere MQ Explorer.

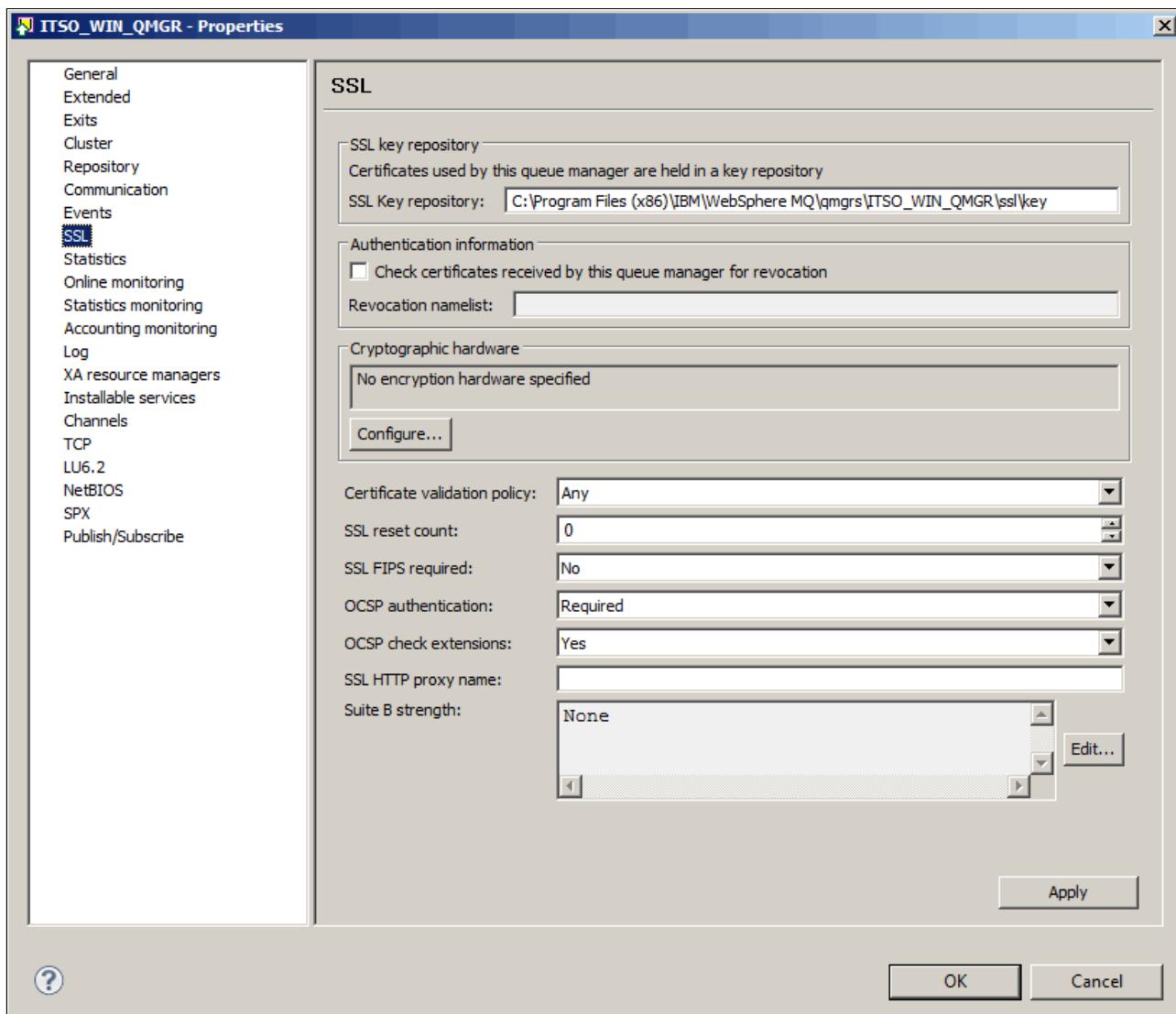


Figure 18-1 Configuring queue manager parameters by using the WebSphere MQ Explorer

18.2.2 Client application configuration

In a client application environment, the following options are available for specifying whether the RFC 5280 standard should be complied with strictly:

- ▶ Using the MQSC structure programmatically
- ▶ Using environment variables
- ▶ Using the client configuration file

This feature is only applicable to C client applications.

Using the MQSCO structure programmatically

For more information about the fields that are available to configure SSL/TLS on a client connection to a queue manager, see the topic *MQSCO - SSL configuration options* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/fr15130_.htm

When the MQSCO_VERSION_4 structure is used, the client application can optionally supply a value for CertificateValPolicy. The field can be one of the following MQLONG values:

- ▶ MQ_CERT_VAL_POLICY_ANY: This value is used for the default behavior.
- ▶ MQ_CERT_VAL_POLICY_RFC5280: This value is used for strict compliance enforcement.

Using environment variables

The environment variable MQCERTVPOL can be set to either of the following values:

- ▶ ANY: This value is used for the default behavior.
- ▶ RFC5280: This value is used for strict compliance enforcement.

For more information about and examples of how to set this environment variable on the supported platforms, see the topic *MQCERTVPOL* in the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/cs12295_.htm

Using the client configuration file

The client configuration file mqclient.ini can be used to configure many connection options for the client application to use, including SSL connections options. Example 18-1 shows a sample mqclient.ini file, which contains the SSL stanza.

Example 18-1 A sample mqclient.ini file, including the SSL stanza

```
/* Module Name: mqclient.ini                                     */
/* Type      : WebSphere MQ MQI client configuration file */#
/* Function   : Define the configuration of a client          */
/* Notes      :                                                 */
/* 1) This file defines the configuration of a client        */
/* */#
/******#                                                       */
ClientExitPath:
    ExitsDefaultPath=/var/mqm/exits
    ExitsDefaultPath64=/var/mqm/exits64

TCP:
    KeepAlive = Yes
    ClnSndBuffSize=32768
    ClnRcvBuffSize=32768
    Connect_Timeout=0

MessageBuffer:
    MaximumSize=-1
```

```
Updatepercentage=-1  
PurgeTime=0  
  
SSL:  
CertificateValPol=RFC5280  
OCSPAuthentication=REQUIRED
```

The CertificateValPol attribute, under the SSL stanza of the `mqclient.ini` file, can be set to ANY or RFC5280. Setting this attribute to RFC5280 results in strict enforcement of compliance with the standard policy.

The location of the `mqclient.ini` file is defined in the topic *Location of the client configuration file* in the Information Center, which is available at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=/com.ibm.mq.doc/cs13360_.htm

Client configuration options precedence order for certificate validation

Each of the options for configuring strict compliance with RFC5280 that are available to the client can be set independently. Therefore, the information might be duplicated in multiple places over time.

When a client application encounters multiple specifications of the certificate validation policy parameter, the option that is adhered to is determined by the following order of precedence:

1. The CertificateValPol value that is defined in the MQSCO structure.
2. The value that is defined in the MQCERTVPOL environment variable.
3. The value that is defined in the SSL stanza of the client configuration file.



Part 4

Scenarios

This part describes a number of complete scenarios that demonstrate how to use some of the new features and enhancements to WebSphere MQ that were described in parts two and three of this book.

This part consists of the following chapters:

- ▶ Chapter 19, “Coexistence: A staged migration on Windows, UNIX, and Linux” on page 259
- ▶ Chapter 20, “Channel authentication records: Controlling remote user activity” on page 277
- ▶ Chapter 21, “Clustering: Multiple cluster transmission queues” on page 313
- ▶ Chapter 22, “Shared queues: Using the new capabilities” on page 331
- ▶ Chapter 23, “GROUPUR: Using Group units of recovery with CICS” on page 379
- ▶ Chapter 24, “Resiliency: Improving availability” on page 417



Coexistence: A staged migration on Windows, UNIX, and Linux

This chapter contains a scenario that demonstrates how multiple installation support can be used to complete a staged migration of an existing setup from one release to another. The scenario migrates a single-machine configuration that consists of two queue managers and two applications from WebSphere MQ V7.0.1.6 to WebSphere MQ V7.1. It then describes how the same technique can be used to apply maintenance by migrating to WebSphere MQ V7.1.0.1.

This chapter contains the following sections:

- ▶ Initial configuration
- ▶ Preparing the scenario
- ▶ Installing WebSphere MQ V7.1
- ▶ Preparing the first application
- ▶ Migrating the first queue manager
- ▶ Completing the migration
- ▶ Uninstalling WebSphere MQ V7.0.1
- ▶ Applying WebSphere MQ V7.1 fix pack 1

19.1 Initial configuration

This section describes the WebSphere MQ V7.0.1.6 configuration that is migrated to WebSphere MQ V7.1. It can be implemented on Windows, UNIX, or Linux.

The WebSphere MQ configuration and application flow is shown in Figure 19-1.

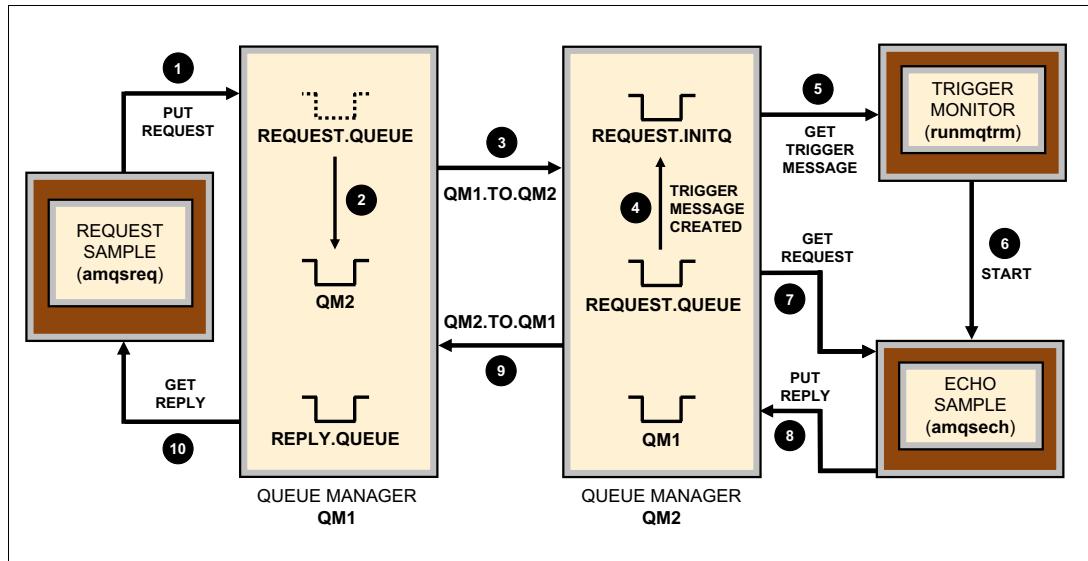


Figure 19-1 The WebSphere MQ configuration and application flow

The configuration consists of the following components:

- ▶ The WebSphere MQ Request sample application, amqsreq, which sends requests and waits for their replies.
- ▶ The WebSphere MQ Echo sample application, amqsech, which processes requests and replies by sending back the same messages.
- ▶ The WebSphere MQ supplied trigger monitor, runmqtrm, which starts the Echo sample to process the requests.
- ▶ Two queue managers that are connected by sender-receiver channels over which the request and reply messages flow.

The numbered arrows in Figure 19-1 show the following sequence of steps that occur for a single request to be processed:

1. The Request sample, amqsreq, connects to queue manager QM1 and puts a request message to the remote queue that is called REQUEST.QUEUE.
2. The request message is temporarily stored on the local transmission queue that is called QM2.
3. The request message is sent to queue manager QM2 via the QM1.TO.QM2 channel and is placed on the local queue that is called REQUEST.QUEUE.
4. WebSphere MQ generates a trigger event message and puts it on the local queue that is called REQUEST.INITQ.
5. The WebSphere MQ trigger monitor, runmqtrm, that is connected to queue manager QM2 gets the trigger event message.

6. The WebSphere MQ trigger monitor processes the trigger event and starts the Echo sample application, amqsech.
7. The Echo sample application connects to queue manager QM2 and gets the request message from the REQUEST.QUEUE queue.
8. The Echo sample application puts the same message back on the remote reply queue that is indicated in the request message; it is temporarily stored on the local transmission queue that is called QM1.
9. The reply message is sent to queue manager QM1 via the QM2.TO.QM1 channel and placed on the reply queue that is called REPLY.QUEUE.
10. The Request sample application gets the reply from the REPLY.QUEUE queue to complete the application flow.

Important: This application flow also can be implemented by using a single queue manager to which both applications can connect. Two queue managers are used in this scenario to show how queue managers can be migrated individually in a multiple installation environment.

19.2 Preparing the scenario

This section describes the process that is used to set up the initial WebSphere MQ V7.0.1.6 configuration before you migrate to WebSphere MQ V7.1.

19.2.1 Installing WebSphere MQ V7.0.1

To set up the initial configuration for this scenario, you must install WebSphere MQ V7.0.1 at fix pack 6, or later, on a Windows, Linux, or UNIX machine. You must install the optional sample applications that are provided because these applications are used instead of writing new applications.

For more information about how to install WebSphere MQ V7.0.1, see the *Quick Beginnings* topic for your operating system in the WebSphere MQ V7.0.1 Information Center at this website:

<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp>

19.2.2 Preparing the sample applications

The following sample applications are used during this scenario:

- ▶ The Echo sample, amqsech
- ▶ The Request sample, amqsreq

On UNIX and Linux, the applications are in the `MQ_INSTALLATION_ROOT/samp/bin` directory. On Windows, the applications are in the `MQ_INSTALLATION_ROOT\tools\c\Samples\bin` directory.

WebSphere MQ V7.0.1 is uninstalled later in this scenario. Copy the two applications to an alternative location outside the WebSphere MQ installation directory so that they are still available after the installation is removed.

The following instructions assume that you copied the applications to a directory that is called /samples on UNIX and Linux, or C:\samples on Windows. If you choose to use an alternative directory, you must adjust references to this location accordingly. When instructed to run the Request sample application throughout this scenario, you should do so from this location.

For more information about the Request sample application, see *The Request sample programs* in the WebSphere MQ V7.0.1 Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqz1.doc/fg17460_.htm

For more information about the Echo sample application, see *The Echo sample programs* in the WebSphere MQ V7.0.1 Information Center at this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqz1.doc/fg17590_.htm

19.2.3 Creating the queue managers

Two queue managers that are called QM1 and QM2 must be created and then started by using the **crtmqm** and **strmqm** commands. For more information about these commands, see *The control commands* in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/fa15550_.htm

19.2.4 Creating the WebSphere MQ objects

Some WebSphere MQ objects must be defined on the queue managers. You can define the objects by using the information in Table 19-1, Table 19-2, Table 19-3 on page 263, and Table 19-4 on page 263. You also can use the commands that are provided in Appendix A, “MQSC scripts for the Coexistence scenario” on page 445.

Listeners

Listeners that are required for this scenario are listed in Table 19-1.

Table 19-1 List of listeners

QM name	Listener name	Port number	Controlled by
QM1	QM1.LISTENER	1614	Queue manager
QM2	QM2.LISTENER	1615	Queue manager

The listeners are required to enable the channel connections between the two queue managers.

Processes

A process definition is required to trigger the Echo sample application, as listed in Table 19-2.

Table 19-2 List of processes

QM name	Process name	Application ID
QM2	ECHO	<path to amqsech>

The application identifier should specify the fully qualified path to the Echo sample application; for example, on UNIX and Linux, as shown in the following example:

/samples/amqsech

Or, on Windows, as shown in the following example:

C:\Samples\amqsech.exe

If you copied the sample applications to a different location to than what was suggested in 19.2.2, “Preparing the sample applications” on page 261, you must adjust this path accordingly.

Queues

Queues that are required for this scenario are listed in Table 19-3.

Table 19-3 List of queues

QM name	Queue name	Queue type	Parameters
QM1	QM2 REPLY.QUEUE REQUEST.QUEUE	Local Local Remote	USAGE: Transmission USAGE: Normal RNAME: REQUEST.QUEUE RQMNAME: QM2 XMITQ: QM2
QM2	QM1 REQUEST.INITQ REQUEST.QUEUE	Local Local Local	USAGE: Transmission USAGE: Normal INITQ: REQUEST.INITQ PROCESS: ECHO TRIGGER: Yes TRIGTYPE: First USAGE: Normal

Channels

Channels that are required for this scenario are listed in Table 19-4.

Table 19-4 List of channels

QM name	Channel name	Channel type	Connection name	Transmission queue
QM1	QM1.TO.QM2 QM2.TO.QM1	Sender Receiver	<hostname>(1615)	QM2
QM2	QM2.TO.QM1 QM1.TO.QM2	Sender Receiver	<hostname>(1614)	QM1

The channels are required to enable the messages to be sent between the two queue managers.

19.2.5 Verifying the initial configuration

This section describes how to verify that the configuration is working correctly before the migration is started. This verification process ensures that a request can be sent and processed and that its reply successfully received.

Complete the following steps to verify the setup:

1. Ensure that both queue managers are running by using the **dspmq** command to display their status. The command requires no parameters.
2. Ensure that the WebSphere MQ objects were configured correctly by using the information that is in 19.2.4, “Creating the WebSphere MQ objects” on page 262, or the sample MQSC commands that are in Appendix A, “MQSC scripts for the Coexistence scenario” on page 445.
3. Ensure that the listener on each queue manager is running.

If you restart the queue managers after the listener objects are defined, the listeners should start automatically.

4. Start the QM1.T0.QM2 sender channel on queue manager QM1 by using the following MQSC command:

START CHANNEL(QM1.T0.QM2)

5. Start the QM2.T0.QM1 sender channel on queue manager QM2 by using the following MQSC command:

START CHANNEL(QM2.T0.QM1)

6. Start the trigger monitor on queue manager QM2 by using the following command:

runmqtrm -m QM2 -q REQUEST.INITQ

You should see output similar to the output that is shown in Example 19-1.

Example 19-1 Initial output when the trigger monitor is started

```
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.  
10/10/2012 10:40:05 AM : WebSphere MQ trigger monitor started.
```

```
10/10/2012 10:40:05 AM : Waiting for a trigger message
```

7. In a second command prompt, start the Request sample application so it connects to queue manager QM1 by using the following command:

amqsreq REQUEST.QUEUE QM1 REPLY.QUEUE

You should see output similar to the output that is shown in Example 19-2.

Example 19-2 Initial output when amqsreq is started

```
Sample AMQSREQ0 start  
server queue is REQUEST.QUEUE  
replies to REPLY.QUEUE
```

- Send a request message by using the Request sample application and verify that a reply message is received.

To send a message, enter some text, such as Hello World, and press **Enter**. The trigger monitor starts the Echo sample application after the request is transmitted to queue manager QM2. The trigger monitor then waits for the next trigger event to occur. An example of the output from the trigger monitor is shown in Example 19-3.

Example 19-3 Output from the trigger monitor when it starts amqsech to process a request

```
/samples/amqsech 'TMC 2REQUEST.QUEUE          ECHO
/samples/amqsech
QM2
Sample AMQSECHA start
Hello world
MQGET ended with reason code 2033
Sample AMQSECHA end
10/10/2012 10:45:14 AM : End of application trigger.
```

The MQGET fails with reason code 2033 after the request is processed because no other messages are available on the queue. This reason code is not an error.

To send another request message, enter some more text and press **Enter**. When you finish sending requests, press **Enter** without entering any text. The Request sample application retrieves the replies, outputs them to the terminal, and ends. An example of the output from the Request sample application is shown in Example 19-4.

Example 19-4 Output from amqsreq when it retrieves the replies to its requests

```
response <Hello world>
no more replies
Sample AMQSREQ0 end
```

Request sample application: The Request sample application waits for up to a minute to receive each reply message. Therefore, there is a short delay after the last reply is received before the application ends.

Important: If you encounter an error when the scenario is run, you should check that the WebSphere MQ objects are configured correctly and the listeners and channels are running. The queue manager error logs and application output might also provide more information to identify the problem.

19.3 Installing WebSphere MQ V7.1

The next step is to install WebSphere MQ V7.1 alongside WebSphere MQ V7.0.1. You do not have to stop your existing queue managers or uninstall WebSphere MQ V7.0.1 to complete this installation.

19.3.1 Installing WebSphere MQ V7.1 on Windows

If you install WebSphere MQ V7.1, or later, on Windows, and there already is another installation of WebSphere MQ on the machine. You see the window that is shown in Figure 19-2. By using this window, you specify whether you want to upgrade an existing installation or install a new copy of WebSphere MQ. For this scenario, select **Install leaving the existing installation(s) untouched** and click **Next**.

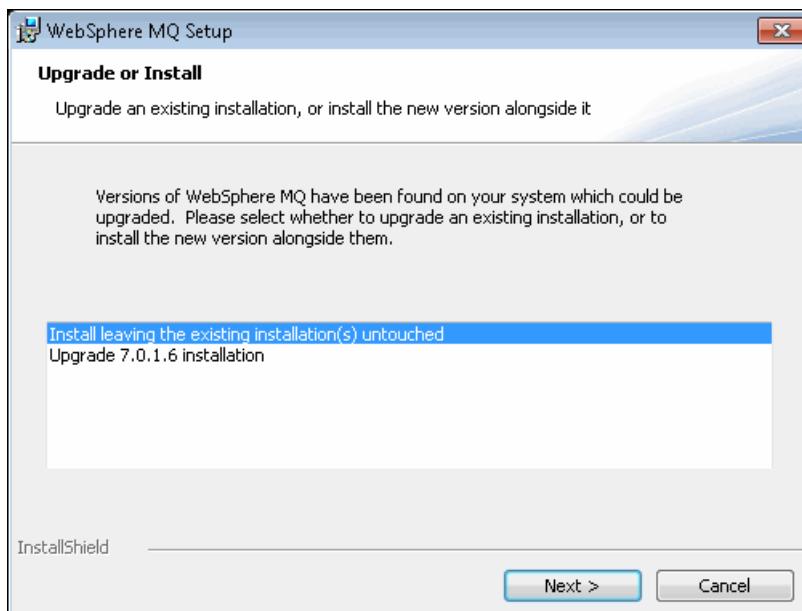


Figure 19-2 Select to upgrade or install a new copy of WebSphere MQ on Windows

To install WebSphere MQ V7.1 alongside WebSphere MQ V7.0.1, a non-default location must be used so as not to conflict with the existing installation. After the license agreement is accepted, you see the window that is shown in Figure 19-3 on page 267. By using this window, you install by using default options or choose to customize them. If you select **Typical**, the installation wizard generates a unique installation location. This option uses the default installation path and a unique suffix, such as C:\Program Files\IBM\WebSphere MQ_1. If you do not want to install WebSphere MQ V7.1 in this location, select **Custom** before you click **Next**. You can customize the location before you continue.

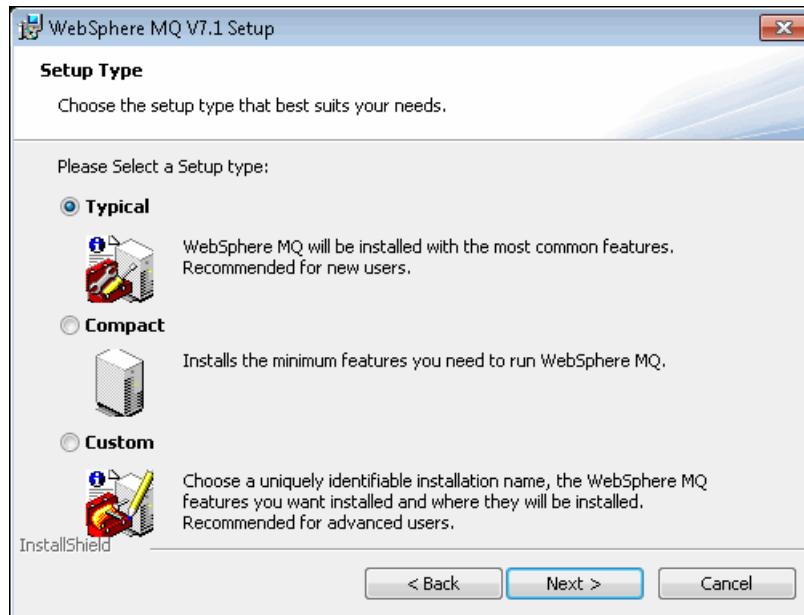


Figure 19-3 The window with which you install with default options or choose to customize them

Follow the instructions in remaining windows to complete the installation. After the Prepare WebSphere MQ wizard runs, you have two WebSphere MQ services, one for each installation, as shown in Figure 19-4. If you hover the mouse pointer over the service for a WebSphere MQ V7.1, or later, installation, its name is identified in parentheses.



Figure 19-4 A WebSphere MQ service for each installation on Windows

19.3.2 Installing WebSphere MQ V7.1 on UNIX and Linux

To install WebSphere MQ V7.1, you must specify a non-default location for this installation so as not to conflict with WebSphere MQ V7.0.1. Before you start, you might want to review 6.2, “Relocatable installations for UNIX and Linux” on page 72 on how to install WebSphere MQ in a non-default location.

On Linux and Solaris, you also must repackage the installation files by using the **crtmqpkg** command before you attempt the installation. For information about the **crtmqpkg** command, see “Repackaging WebSphere MQ by using crtmqpkg” on page 77.

19.4 Preparing the first application

This step updates the environment for the Request sample application so it uses the WebSphere MQ V7.1 libraries instead of the libraries that are supplied with WebSphere MQ V7.0.1. This step must be performed before queue manager QM1 is migrated; otherwise, the application cannot connect after the migration is complete.

Some applications must be updated before they can be used with relocated installations, or on systems with multiple installations present. Some of these considerations are described in 7.3, “Managing applications” on page 88 and in the topic *Loading WebSphere MQ version 7.1 libraries* in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/za00130_.htm

One of these considerations is with which WebSphere MQ libraries an application is linked. The documentation on how to build applications for previous versions of WebSphere MQ include instructions to link with one or more of the following libraries, depending on the operating system that is used:

- ▶ mqm or mqic
- ▶ mqmcs
- ▶ mqmzse

For an application to work correctly with multiple or non-default installations, it must not be linked with mqmcs or mqmzse. This restriction is necessary because the mqm library automatically loads the correct version of the other two libraries, depending on the installation with which a queue manager is associated. If an application is linked with them directly, conflicts can arise.

The pre-built sample applications that are shipped with WebSphere MQ V7.0.1 are linked with these two libraries on UNIX and Linux. If you are using one of these platforms, use the **ldd** command to list the dependencies for the amqsreq and amqsech applications. An invocation of this command and its output on Linux are shown in Example 19-5.

Example 19-5 Output from the ldd command for the WebSphere MQ V7.0.1 amqsreq sample on Linux

```
ldd amqsreq

linux-vdso.so.1 => (0x00007ffffaafff000)
libmqm_r.so => /opt/mqm/lib64/libmqm_r.so (0x00007f2071b7c000)
libmqmcs_r.so => /opt/mqm/lib64/libmqmcs_r.so (0x00007f20717a2000)
libmqmzse.so => /opt/mqm/lib64/libmqmzse.so (0x00007f20716a1000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x0000003111600000)
libdl.so.2 => /lib64/libdl.so.2 (0x0000003110e00000)
libc.so.6 => /lib64/libc.so.6 (0x0000003111200000)
libmqz_r.so => /opt/mqm/lib64/libmqz_r.so (0x00007f20713b9000)
/lib64/ld-linux-x86-64.so.2 (0x0000003110a00000)
```

To remove these dependencies, you must recompile or re-link each application so that they are not linked with these two libraries. Instead of rebuilding the sample applications in this scenario, the versions that are shipped with WebSphere MQ V7.1 are used.

If you are using UNIX or Linux, replace the amqsreq application in the directory that you used in 19.2.2, “Preparing the sample applications” on page 261 with the pre-built version that is shipped with WebSphere MQ V7.1. You do not have to update the amqsech application now. If you are using Windows, you do not have to perform this step.

The application is now updated, if required, so it is compatible with the use of WebSphere MQ V7.1 on the same machine as WebSphere MQ V7.0.1. The next step is to update the environment so that the WebSphere MQ V7.1 libraries are used.

To update the environment for the Request sample application, run the **setmqenv** command from the command prompt that is used to run the sample. This command changes the location from which the WebSphere MQ libraries are loaded. For information about the **setmqenv** command, see “Using setmqenv” on page 82.

If a Windows machine is used, run the following command:

```
MQ_INSTALLATION_ROOT\bin\setmqenv -s
```

If a UNIX or Linux machine is used, run the following command:

```
. MQ_INSTALLATION_ROOT/bin/setmqenv -s -k
```

You should substitute MQ_INSTALLATION_ROOT for the location where you installed WebSphere MQ V7.1; for example, /opt/mqm71.

On all platforms, confirm you set up your environment correctly by running the **dspmqver** command. Check that the version of WebSphere MQ that is reported is 7.1.

On UNIX and Linux, you also should use the **ldd** command to verify that the WebSphere MQ libraries that are required by the Request sample application are loaded from the WebSphere MQ V7.1 installation. An invocation of this command and its output on Linux are shown in Example 19-6. It shows the WebSphere MQ libraries are now loaded from /opt/mqm71 (where WebSphere MQ V7.1 is installed) instead of /opt/mqm.

Example 19-6 Output from ldd on Linux showing the libraries are loaded from the V7.1 installation

```
ldd amqsreq
```

```
linux-vdso.so.1 => (0x00007fff4c9ef000)
libmqm_r.so => /opt/mqm71/lib64/compat/libmqm_r.so (0x00007f6ed290f000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x0000003111600000)
libdl.so.2 => /lib64/libdl.so.2 (0x0000003110e00000)
libc.so.6 => /lib64/libc.so.6 (0x0000003111200000)
libmqe_r.so => /opt/mqm71/lib64/libmqe_r.so (0x00007f6ed1f5c000)
/lib64/ld-linux-x86-64.so.2 (0x0000003110a00000)
libm.so.6 => /lib64/libm.so.6 (0x0000003111a00000)
librt.so.1 => /lib64/librt.so.1 (0x0000003112200000)
```

Updating the environment by using the **setmqenv** command should not prevent the Request sample application from connecting to queue manager QM1 before it is migrated to WebSphere MQ V7.1. This capability is important because it allows all applications to be updated beforehand.

Important: On UNIX and Linux, the WebSphere MQ V7.1 pre-built sample applications include an embedded runtime search path (RPATH). An embedded RPATH means that running the **setmqenv** command is not necessary to load the WebSphere MQ libraries in the same installation. The use of the **setmqenv** command is included in the instructions for this scenario because it is required on Windows. Also, not all applications include an embedded RPATH, or if they do, it might specify an incorrect location.

Rerun the Request sample to verify it still runs successfully and replies are received from the Echo sample application. You might need to restart the two sender channels if they stopped because of inactivity. If you must restart the trigger monitor, it should be run from a console that is not configured for WebSphere MQ V7.1 by using the `setmqenv` command.

19.5 Migrating the first queue manager

The next step is to migrate queue manager QM1 from WebSphere MQ V7.0.1 to WebSphere MQ V7.1. When you are migrating to a version of WebSphere MQ before V7.1, this operation requires a considerable outage because WebSphere MQ must be removed before the new version can be installed in its place. Migrating a queue manager to WebSphere MQ V7.1, or later, by using the `setmqm` command requires a much shorter outage.

Complete the following steps to migrate queue manager QM1:

1. Ensure that the environment is configured for the WebSphere MQ V7.1 installation by using the `setmqenv` command, as described in 19.4, “Preparing the first application” on page 268.
2. Run the `dspmqinst` command to confirm the name of the WebSphere MQ V7.1 installation.
3. End the Request sample application if it is still running. You do not have to stop the trigger monitor that is connected to queue manager QM2 or the Echo sample application if it is running.
4. Stop the two sender channels.
5. Stop queue manager QM1 by using the `endmqm` command in the WebSphere MQ V7.0.1 installation. You can run this version of the command by fully qualifying its location; for example, on Linux:
`/opt/mqm/bin/endmqm -w QM1`
6. Use the `setmqm` command to change queue manager QM1 so that it is associated with the WebSphere MQ V7.1 installation. Use the command, as shown in the following example:
`setmqm -m QM1 -n Installation1`
7. To complete the migration process, start queue manager QM1 by using the `strmqm` command in the WebSphere MQ V7.1 installation. You should see output similar to the output that is shown in Example 19-7.

Example 19-7 Output from strmqm when a queue manager at version 7.1 is started for the first time

```
WebSphere MQ queue manager 'QM1' starting.  
The queue manager is associated with installation 'Installation1'.  
5 log records accessed on queue manager 'QM1' during the log replay phase.  
Log replay for queue manager 'QM1' complete.  
Transaction manager state recovered for queue manager 'QM1'.  
Migrating objects for queue manager 'QM1'.  
Default objects statistics : 6 created. 0 replaced. 0 failed.  
WebSphere MQ queue manager 'QM1' started using V7.1.0.0.
```

Important: After you start the queue manager by using WebSphere MQ V7.1, you cannot start it by using WebSphere MQ V7.0.1 because migrating a queue manager to an earlier version of WebSphere MQ is not supported, except on z/OS.

8. Use the **dspmq** command in the WebSphere MQ V7.1 installation to display the installation with which each queue manager is associated, as shown in the following example:

```
dspmq -o installation
```

You should see that QM1 is associated with the WebSphere MQ V7.1 installation, and QM2 is associated with the WebSphere MQ V7.0.1 installation. This configuration shows that different queue managers can use different installations on the same machine at the same time. An example of the output you should see from this command is shown in Example 19-8.

Example 19-8 Output from dspmq -o installation

QMNAME(QM1)	INSTNAME(Installation1)	INSTPATH(/opt/mqm71)	INSTVER(7.1.0.0)
QMNAME(QM2)	INSTNAME(Installation0)	INSTPATH(/opt/mqm)	INSTVER(7.0.1.6)

9. Restart the two sender channels and rerun the Request sample application to verify that the configuration still works after queue manager QM1 is migrated.

19.6 Completing the migration

To complete the migration to WebSphere MQ V7.1, queue manager QM2 and the applications that connect to it must be updated in the same manner.

Complete the following steps to complete the migration process:

1. On UNIX and Linux, replace the amqsech application with the pre-built version that is supplied with WebSphere MQ V7.1, as per for amqsreq in 19.4, “Preparing the first application” on page 268. You do not have to perform this step if a Windows machine is used.
2. Stop the two sender channels by using the following MQSC commands:
STOP CHANNEL(QM1.T0.QM2)
STOP CHANNEL(QM2.T0.QM1)
3. End queue manager QM2 by using the **endmqm** command in the WebSphere MQ V7.0.1 installation. The trigger monitor should end automatically.
4. Use the **setmqm** command to associate queue manager QM2 with the WebSphere MQ V7.1 installation by using the following command:
setmqm -m QM2 -n Installation1
5. Restart queue manager QM2 by using the **strmqm** command in the WebSphere MQ V7.1 installation.
6. In the command window that is used to start the trigger monitor, run the **setmqenv** command to configure the environment to locate the WebSphere MQ V7.1 libraries. You might want to see the example invocation in 19.4, “Preparing the first application” on page 268.
7. Restart the trigger monitor by using the **runmqtrm** command in the WebSphere MQ V7.1 installation.
8. Restart the two sender channels. Rerun the Request sample application to verify that the configuration still works after queue manager QM2 is migrated.

19.7 Uninstalling WebSphere MQ V7.0.1

Both queue managers and the applications now are migrated to use WebSphere MQ V7.1 instead of WebSphere MQ V7.0.1. Now it is possible to safely remove WebSphere MQ V7.0.1 because it is no longer required.

Important: To uninstall WebSphere MQ V7.0.1 you must stop all queue managers and applications that loaded a WebSphere MQ library. Therefore, you might want to defer this operation until a suitable maintenance window is available. To uninstall or apply maintenance to later versions of WebSphere MQ, you must stop only queue managers and applications that use that particular installation.

After uninstalling WebSphere MQ V7.0.1, you cannot reinstall it while another copy of WebSphere MQ is installed on the machine.

Complete the following steps to uninstall WebSphere MQ V7.0.1:

1. End the Request sample application if it is running.
2. Stop both sender channels.
3. End both queue managers. The trigger monitor should end automatically when queue manager QM2 is stopped.
4. Uninstall WebSphere MQ V7.0.1.
5. Restart the queue managers.
6. Restart the sender channels.
7. Run the `setmqenv` command to configure the environment for the Echo sample application, then restart the trigger monitor in the same command window.
8. Run the `setmqenv` command, then rerun the Request sample application to verify that the configuration still functions correctly.

19.8 Applying WebSphere MQ V7.1 fix pack 1

To apply maintenance to WebSphere MQ V7.0.1, or earlier, you must stop all queue managers and applications that use WebSphere MQ libraries before the updates to the existing installation are applied. After this step is performed, the queue managers and applications can be restarted.

The following options are available to apply maintenance to WebSphere MQ V7.1, or later:

- ▶ You can choose to apply updates to an existing installation by using the same method as was used for WebSphere MQ V7.0.1, or earlier.
- ▶ You can install another copy of WebSphere MQ, apply the maintenance to that installation, and then change your existing queue managers and applications to use the new installation instead. You can then remove the original installation if it is no longer required.

This section describes how you can apply maintenance by using the second method.

19.8.1 Installing a second copy of WebSphere MQ V7.1

The first step is to install a second copy of WebSphere MQ V7.1 on the same machine. A custom name is specified for this installation to show how it can be achieved. On Windows, you can provide the name during the installation process. On UNIX and Linux, you must use the `crtmqinst` command to specify the name. Before proceeding, you might want to review “Naming installations” on page 79, which describes the process that is used to name installations.

Instead of the use of the default name, such as `Installation2`, the following instructions show how the installation can be named `MQ71FixPack1`.

On Windows, at the start of the installation process, you see the window that is shown in Figure 19-5. This window is different from the window that is shown in Figure 19-2 on page 266 because there is no longer an installation of WebSphere MQ V7.0.1 that can be upgraded. Select **Install a new instance** and click **Next**.

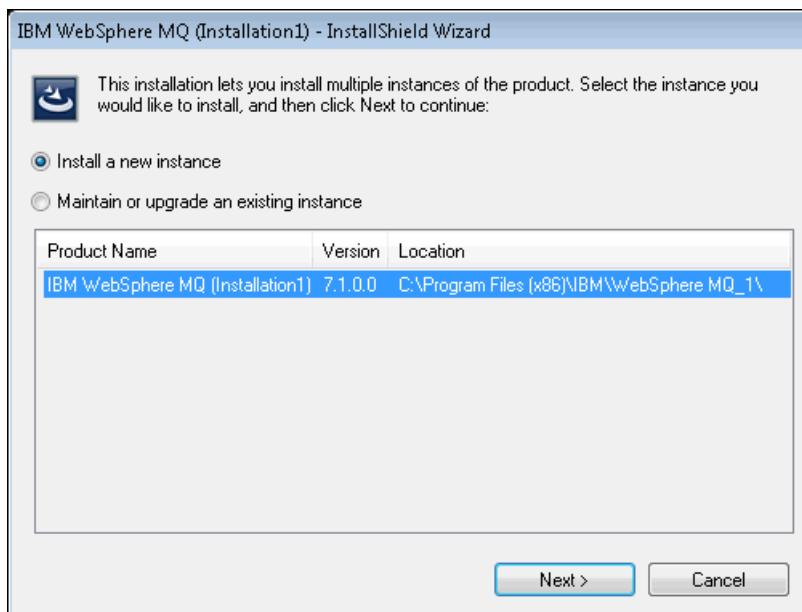


Figure 19-5 The dialog to install a new instance or maintain an existing copy of WebSphere MQ

On Windows, after the license agreement is accepted, the window that is shown in Figure 19-3 on page 267 is displayed. Select **Custom**, then click **Next** to allow the installation name to be customized. You see the window that is shown in Figure 19-6.

Change the installation name to MQ71FixPack1.

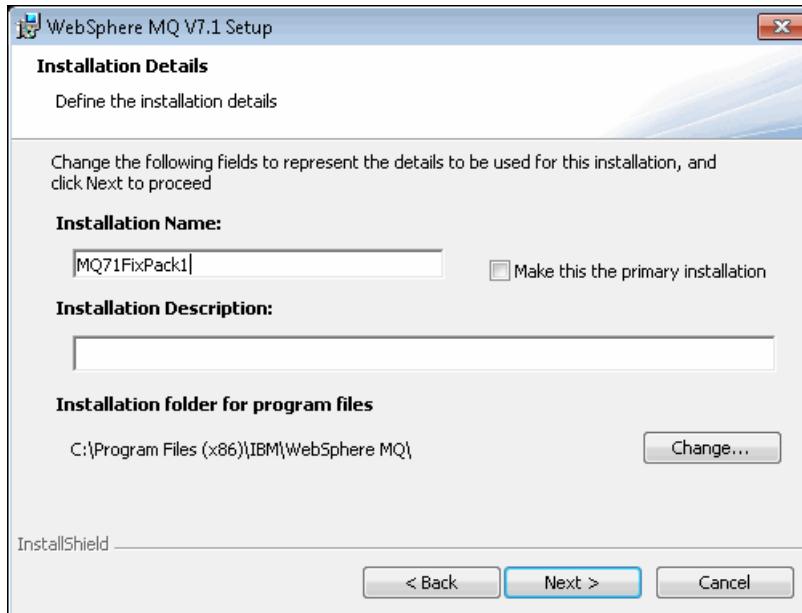


Figure 19-6 Specifying the installation name on Windows

The installation folder is the default location for WebSphere MQ because WebSphere MQ V7.0.1 was uninstalled from that path. You can update this location if you want or leave it unchanged.

On Windows, you also can automatically make the new copy of WebSphere MQ the primary installation. This option runs the **setmqinst** command at the end of the installation process. It is not available to select if WebSphere MQ V7.0.1 is still installed because WebSphere MQ V7.0.1 must always be the primary installation. You do not need a primary installation for this scenario, so you can leave this option cleared.

On Windows, click **Next** to continue, then complete the installation process.

On UNIX and Linux, run the **crtmqinst** command before the second copy of WebSphere MQ is installed. By using this command, which must be run as root, you can specify the name of the installation, as shown in the following example:

```
crtmqinst -p /opt/mqm7101 -n MQ71FixPack1
```

On UNIX and Linux, install WebSphere MQ in the same location as specified to **crtmqinst**.

19.8.2 Installing WebSphere MQ V7.1 fix pack 1

The next step is to apply maintenance to the second copy of WebSphere MQ V7.1 by installing fix pack 1.

On Windows, you are asked during the installation process which copy of WebSphere MQ to which you want to apply the fix pack. An example of the window with which you select the installation is shown in Figure 19-7. When you are installing the fix pack, you can choose to load the installation files on to the server, but not apply the fix pack to an installation. If you chose to load the installation files to the server, you can apply the fix pack to a copy of WebSphere MQ by selecting **Apply fix pack 7.1.0.1** from the IBM WebSphere MQ program group in the Start menu.

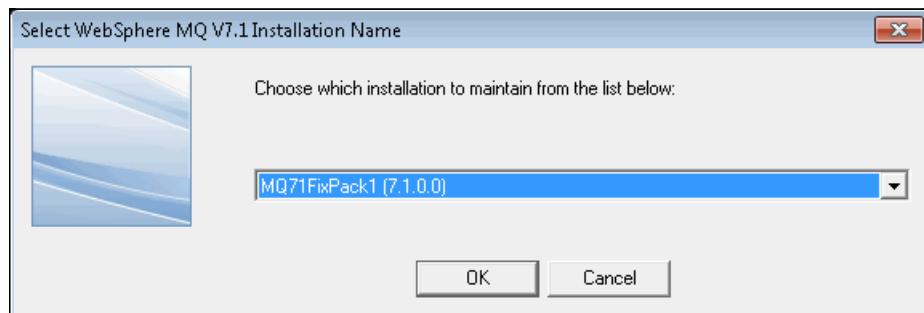


Figure 19-7 Selecting the installation to apply the fix pack to on Windows

On AIX and HP-UX, you must specify the location of the installation that you want to update. On Linux and Solaris, if you used the **crtmqpk** command when the copy of WebSphere MQ you want to update was installed, you must also use the **crtmqfp** command to repackage the installation files to use the same suffix. For more information about the use of the **crtmqfp** command, see 7.1.2, “Applying maintenance-level upgrades” on page 78.

For more information about applying maintenance to WebSphere MQ, see the topic *WebSphere MQ maintenance tasks* in the WebSphere MQ Information Center at this website:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/z100090_.htm

19.8.3 Applying fix pack 1 to the queue managers

The machine now features two installations of WebSphere MQ V7.1, one at version 7.1.0.0, and the other at version 7.1.0.1. To apply fix pack 1 to the queue managers, you use the **setmqm** command to associate them with the second installation. This process is the same process that was used to migrate the queue managers from WebSphere MQ V7.0.1 in 19.5, “Migrating the first queue manager” on page 270 and 19.6, “Completing the migration” on page 271.

However, this time you can update the queue managers without the need to update the application environments beforehand because the applications use WebSphere MQ libraries that are supplied with WebSphere MQ V7.1, or later. These libraries automatically load their counterparts when they are connecting to queue managers that are associated with another installation. You must update the application environments only if you want to remove the installation from which they are loading their initial copies of the WebSphere MQ libraries.

Migrate queue manager QM1 to use fix pack 1 by using the instructions in 19.5, “Migrating the first queue manager” on page 270. Remember to use or specify the WebSphere MQ V7.1.0.1 installation when the **setmqm** and **setmqenv** commands are run. You can confirm the queue manager is running with the maintenance that is applied by displaying the VERSION queue manager attribute, which was added in WebSphere MQ V7.1. The following MQSC command can be used to display this information:

DISPLAY QMGR VERSION

The response from a version 7.1.0.1 queue manager is shown in Example 19-9.

Example 19-9 Output of DISPLAY QMGR VERSION for a queue manager at version 7.1.0.1

AMQ8408: Display Queue Manager details.	
QMNAME(QM1)	VERSION(07010001)

Queue manager QM2 can be updated to fix pack 1 by using the same procedure.

19.8.4 Removing fix pack 1 from the queue managers

You also can use the **setmqm** command to associate a queue manager with an installation at the same version of WebSphere MQ, but which is at a lower level of maintenance. This capability means that you can follow the same sequence of steps to switch queue manager QM1 back to the original WebSphere MQ V7.1 installation if you want to roll back the application of fix pack 1.

To confirm that fix pack 1 was removed, you can use the same MQSC command to display the version of the queue manager. The response from a version 7.1.0.0 queue manager is shown in Example 19-10.

Example 19-10 Output of DISPLAY QMGR VERSION for a queue manager at version 7.1.0.0

AMQ8408: Display Queue Manager details.	
QMNAME(QM1)	VERSION(07010000)



Channel authentication records: Controlling remote user activity

This chapter describes a scenario that uses some of the features and motivations from Chapter 8, “Enhanced security” on page 99. The scenario gives specific focus to the use of channel authentication records to map one user ID to another and to block a user ID’s access to a channel. The configuration makes deliberate use of distributed platforms and z/OS to demonstrate the applicability of channel authentication records on all platforms that are supported by WebSphere MQ V7.1.

This chapter contains the following sections:

- ▶ Environment overview
- ▶ Machine configuration
- ▶ User authorities
- ▶ Starting channels and following a message

20.1 Environment overview

The scenario that is described in this chapter is deliberately simple in its architecture and is designed to further explain some of the details that are described in Chapter 8, “Enhanced security” on page 99. In particular, the scenario uses mapping channel authentication records to map and block user IDs.

This section describes the assumptions that were made about the environments that is used. This section also includes an overview of the full, round-trip route that messages take through the systems.

Figure 20-1 is an overview of the scenario, which shows the platforms that are used and a high-level view of the WebSphere MQ resources that are configured on each machine. For more information about the WebSphere MQ resource configuration, see Section 20.2, “Machine configuration” on page 281.

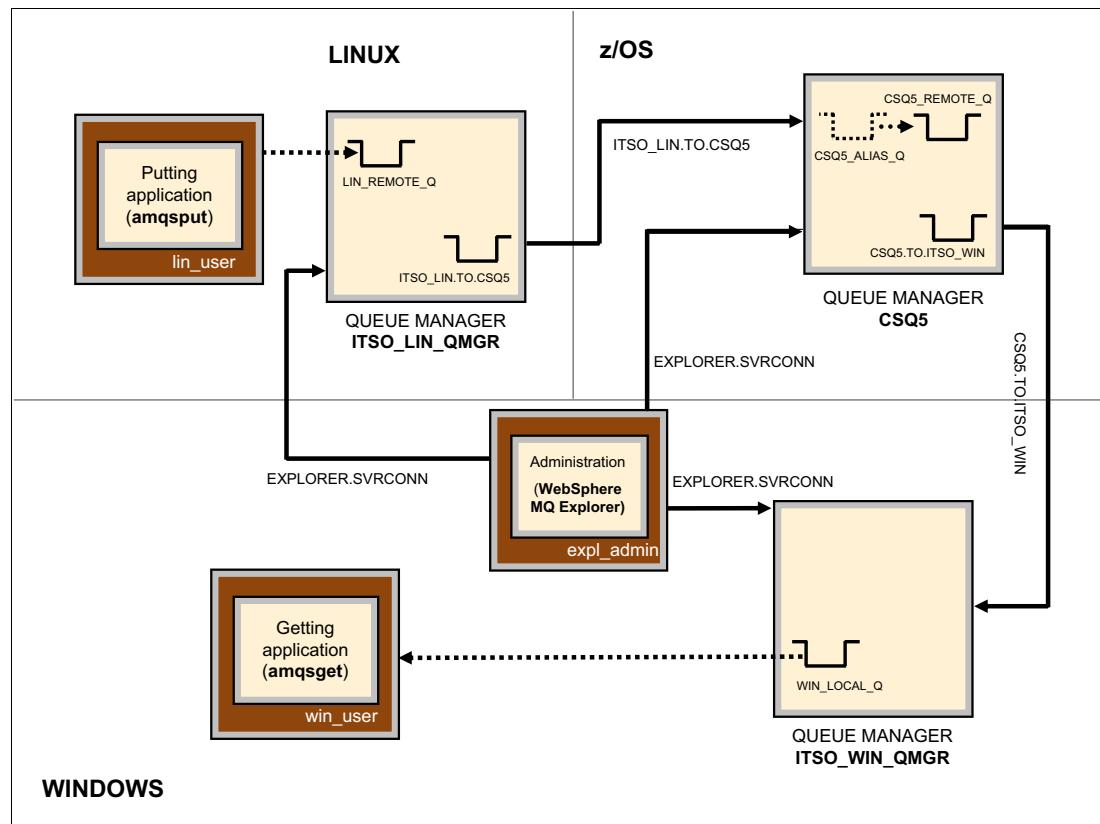


Figure 20-1 Scenario overview

20.1.1 Assumptions

This section describes the following starting assumptions about the environments that are used in this scenario and acts as the starting point from which the scenario builds:

- ▶ The scenario uses three machines that include different operating systems. The following operating systems are used in this scenario:
 - 64-bit Windows 7 Enterprise Edition
 - 64-bit RedHat Enterprise Linux Server 6.2
 - z/OS V1R13
- ▶ Each machine includes the following defined user IDs. The authorities of the Windows and Linux user IDs are defined during the scenario configuration. The z/OS user ID has the authority to administer queue manager objects and start channels. Each user ID features a different spelling:
 - Windows user IDs: win_user, win_admin, expl_admin
 - Linux user IDs: lin_user, lin_admin
 - z/OS user ID: CBOTH
- ▶ None of the user IDs that are used in the scenario are part of that operating system's WebSphere MQ administrator group, unless otherwise stated.
- ▶ The content of the messages that are sent is not important.
- ▶ WebSphere MQ V7.1 is installed on each machine and use the following configurations:
 - On Windows, the default installation options are used. The WebSphere MQ client and WebSphere MQ Explorer (and prerequisite Eclipse packages) are installed.
 - On Linux, the Server, Runtime, JRE, GSKit, SDK, and Client packages are installed.
 - On z/OS, the queue manager (CSQ5) and listener (on port 1416) exist.
- ▶ The WebSphere MQ samples are installed on the Windows and Linux systems.
- ▶ Unless otherwise stated, all of the configuration of the WebSphere MQ resources is platform-independent and can be done by using the same tools on other supported platforms.
- ▶ The specific names of resources and user IDs that are used in this scenario are indicative only, and can be replaced with any other valid values wherever they are used.
- ▶ Unless otherwise specified, each configuration command should be run as a privileged user from the terminal window of the operating system that is in use.
- ▶ Wherever WebSphere MQ Explorer is used, it is run by using the user ID expl_admin.
- ▶ The IP addresses that are used in this scenario must be changed to match your own network.

20.1.2 Administration and message routing overview

This scenario uses two separate channel authentication record types to demonstrate the mapping function, the blocking function, and the control that it is possible to have over access to WebSphere MQ resources when a connection is made over a channel.

The scenario features a route that is taken by messages put onto one queue manager and then through one other queue manager and ends up on one final queue manager. The three queue managers are all administered by using WebSphere MQ Explorer from a central location and over a server connection channel, which is subject to channel authentication record checking.

Queue manager administration

The administration of the queue managers requires a two stage configuration. Initially, the resources and authorities to enable WebSphere MQ Explorer to connect must be defined on the same machine where the queue manager is found. After these resources are set up, WebSphere MQ Explorer can be connected to perform all subsequent queue manager administration tasks from a central location. The second stage of queue manager administration is to configure the resources to implement the message routing.

Message routing

Message routing through the scenario is driven by sample applications that are supplied with WebSphere MQ V7.1. The sample `amqspput` is used to place messages into the system and `amqsget` is used to retrieve the messages from the system after they are routed around the queue managers.

The message route starts from the Linux queue manager and finishes on the Windows queue manager via the z/OS queue manager. For more information about the exact configuration of each machine and the WebSphere MQ resources on those machines, see section 20.2, “Machine configuration” on page 281.

The key point of the message routing is the mapping of the user ID that is used to put the message on each queue manager, as a result of the QMGRMAP channel authentication records. Section 20.4, “Starting channels and following a message” on page 309 reviews the message at each stage of the route and inspects its properties to show the effect of the channel authentication records on the message headers. The content of the message, or message data, is not important in this scenario.

20.2 Machine configuration

This section describes the specific details of the machines that are used for this scenario, including the steps that are required to implement the resources on each machine.

20.2.1 Windows WebSphere MQ configuration details

Figure 20-2 shows the WebSphere MQ configuration for the Windows machine that is used in this scenario.

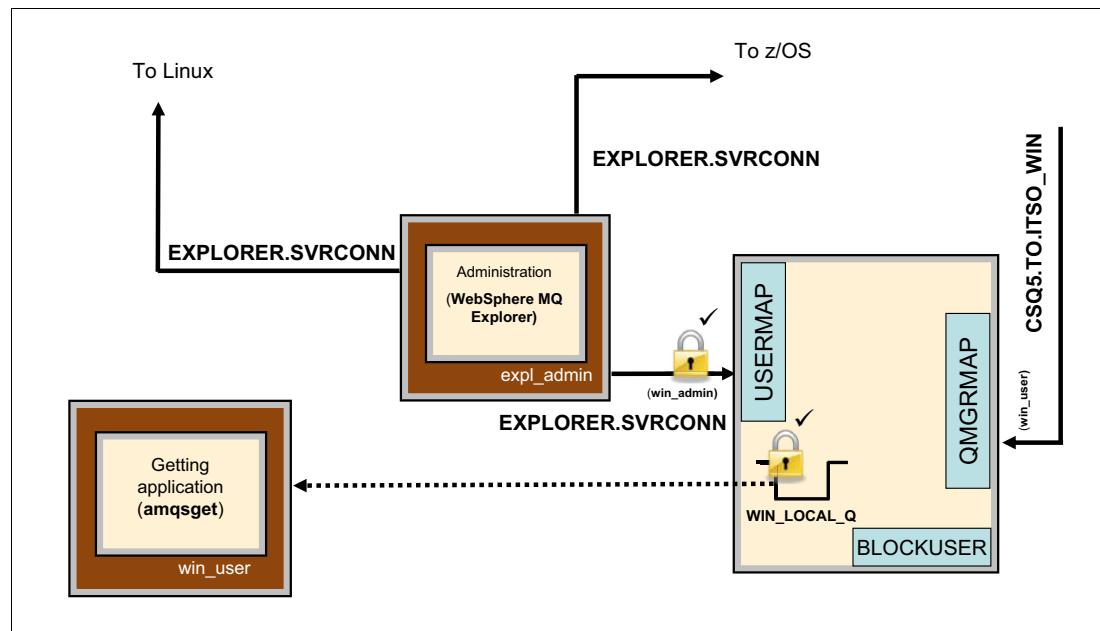


Figure 20-2 The WebSphere MQ configuration on the Windows machine

The Windows machine is used as the central point of administration for the local queue manager and the other queue managers that are used in this scenario. The administration of all of the queue managers is done over SVRCONN channels (including the Windows queue manager) to demonstrate the use of the USERMAP channel authentication record.

The following WebSphere MQ resources are defined on the Windows machine. This section describes the administration of those resources in detail:

- ▶ A single queue manager is configured with the following resources:
 - A single listener.
 - A local queue, which is used as the endpoint to receive messages.
 - A receiver (RCVR) channel, which is connected to by the z/OS queue manager.
 - A QMGRMAP channel authentication record to map the z/OS queue manager name (CSQ5) to a local Windows user ID (win_user).
 - A server connection (SVRCONN) channel, which is connected to by WebSphere MQ Explorer.
 - A USERMAP channel authentication record to map WebSphere MQ Explorer user ID (expl_admin) to an authorized local Windows user ID (win_admin).
 - A BLOCKUSER channel authentication record to block the user IDs from accessing the Windows queue manager directly.

- ▶ WebSphere MQ Explorer is used to administer the queue managers, which run as the user ID expl_admin.
- ▶ The WebSphere MQ sample application amqsgt, which runs as user win_user.

Queue manager creation and initial configuration

Complete the following steps to create the queue manager and configure it to the point at which the remaining configuration can be done by using WebSphere MQ Explorer.

1. From a Windows command prompt, create the queue manager, ITSO_WIN_QMGR, by using the **crtmqm** command.
2. Start the queue manager, ITSO_WIN_QMGR, by using the **strmqm** command.

Example 20-1 shows the output from the creation and starting of ITSO_WIN_QMGR.

Example 20-1 Creating and starting the queue manager ITSO_WIN_QMGR

```
C:\>crtmqm ITSO_WIN_QMGR
WebSphere MQ queue manager created.
Directory 'C:\Program Files (x86)\IBM\WebSphere MQ\qmgrs\ITSO_WIN_QMGR' created.
The queue manager is associated with installation 'Installation1'.
Creating or replacing default objects for queue manager 'ITSO_WIN_QMGR'.
Default objects statistics : 77 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.

C:\>strmqm ITSO_WIN_QMGR
WebSphere MQ queue manager 'ITSO_WIN_QMGR' starting.
The queue manager is associated with installation 'Installation1'.
5 log records accessed on queue manager 'ITSO_WIN_QMGR' during the log replay
phase.
Log replay for queue manager 'ITSO_WIN_QMGR' complete.
Transaction manager state recovered for queue manager 'ITSO_WIN_QMGR'.
WebSphere MQ queue manager 'ITSO_WIN_QMGR' started using V7.1.0.0.
```

3. Start runmqsc against the queue manager, ITSO_WIN_QMGR, by using the command **runmqsc ITSO_WIN_QMGR**.
4. Within runmqsc, define and verify the SVRCONN definition that is used by WebSphere MQ Explorer to access ITSO_WIN_QMGR by using the following commands:

```
def channel(EXPLORER.SVRCONN) ch1type(SVRCONN) TRPTYPE(TCP)
dis ch1(EXPLORER.SVRCONN)
```

Example 20-2 shows the commands and output of defining the SVRCONN channel that is used by WebSphere MQ Explorer.

Example 20-2 Defining and verifying the WebSphere MQ Explorer SVRCONN channel in runmqsc

```
C:\>runmqsc ITSO_WIN_QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.
Starting MQSC for queue manager ITSO_WIN_QMGR.
```

```
def channel(EXPLORER.SVRCONN) chltype(SVRCONN) TRPTYPE(TCP)
  1 : def channel(EXPLORER.SVRCONN) chltype(SVRCONN) TRPTYPE(TCP)
AMQ8014: WebSphere MQ channel created.
dis ch1(EXPLORER.SVRCONN)
  2 : dis ch1(EXPLORER.SVRCONN)
AMQ8414: Display Channel details.
  CHANNEL(EXPLORER.SVRCONN)           CHLTYPE(SVRCONN)
  ALTDATE(2012-10-10)                 ALTTIME(10.02.49)
  COMPHDR(NONE)                      COMPMMSG(NONE)
  DESCR( )                           DISCINT(0)
  HBINT(300)                          KAINT(AUTO)
  MAXINST(999999999)                 MAXINSTC(999999999)
  MAXMSGL(4194304)                   MCAUSER( )
  MONCHL(QMGR)                       RCVDATA( )
  RCVEXIT( )                         SCYDATA( )
  SCYEXIT( )                         SENDDATA( )
  SENDEXIT( )                        SHARECNV(10)
  SSLCAUTH(REQUIRED)                 SSLCIPH( )
  SSLPEER( )                          TRPTYPE(TCP)
```

-
5. Define the USERMAP channel authentication record that maps WebSphere MQ Explorer's user ID (expl_admin) to the local Windows administration user ID (win_admin) and verify the creation of the record by using the **dis chauth** command, as shown in the following example:

```
set chauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER('win_admin')
dis chauth(EXPLORER.SVRCONN)
```

Example 20-3 shows the creation and visual verification of the USERMAP channel authentication record.

Example 20-3 Creating and verifying the channel authentication record in runmqsc

```
set chauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER('win_admin')
3 : set chauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER('win_admin')
AMQ8877: WebSphere MQ channel authentication record set.
dis chauth(EXPLORER.SVRCONN)
7 : dis chauth(EXPLORER.SVRCONN)
AMQ8878: Display channel authentication record details.
    CHLAUTH(EXPLORER.SVRCONN)           TYPE(USERMAP)
    DESCRL( )                          CUSTOM( )
    ADDRESS( )                        CLNTUSER(expl_admin)
    MCAUSER(win_admin)                 USERSRC(MAP)
    ALTDATE(2012-10-10)                ALTTIME(10.05.25)
```

6. Create and start the listener on port 10000, as shown in the following example:

```
def listener(ITSO_WIN_LIST) TRPTYPE(TCP) PORT(10000)
start listener(ITSO_WIN_LIST)
```

Example 20-4 shows the creation and starting of the listener for the queue manager, ITSO_WIN_QMGR.

Example 20-4 Creating and starting the listener in runmqsc

```
def listener(ITSO_WIN_LIST) TRPTYPE(TCP) port(10000)
5 : def listener(ITSO_WIN_LIST) TRPTYPE(TCP) port(10000)
AMQ8626: WebSphere MQ listener created.
start listener(ITSO_WIN_LIST)
6 : start listener(ITSO_WIN_LIST)
AMQ8021: Request to start WebSphere MQ listener accepted.
```

7. End the runmqsc session by using the **end** command.

8. Grant the authority for the user ID win_admin to have access to the queue manager, the queues, and the channels on the queue manager by using the **setmqaut** command.

Example 20-5 shows the two commands that are necessary to grant this authority. These commands are run as a machine-wide administrator. The second and third commands specify a wildcard value for the profile name, **, which represents all of objects of the specified type (in this case, queue and channel).

Example 20-5 Granting win_admin access to the queue manager and its resources

```
C:\>setmqaut -m ITSO_WIN_QMGR -t qmgr -p win_admin +all
The setmqaut command completed successfully.
```

```
C:\>setmqaut -m ITSO_WIN_QMGR -n ** -t queue -p win_admin +all
The setmqaut command completed successfully.
```

```
C:\>setmqaut -m ITSO_WIN_QMGR -n ** -t channel -p win_admin +all
The setmqaut command completed successfully.
```

9. As user ID expl_admin, run WebSphere MQ Explorer in the following path:
 <MQ_INST_PATH>\bin\MQExplorer.exe.
10. In the Navigator pane, right-click **Queue Managers** and select **Add Remote Queue Manager...**
11. Enter the name of the queue manager (**ITSO_WIN_QMGR**) in the Queue manager name field, as shown in Figure 20-3. Leave the Connect directly option selected and click **Next**.

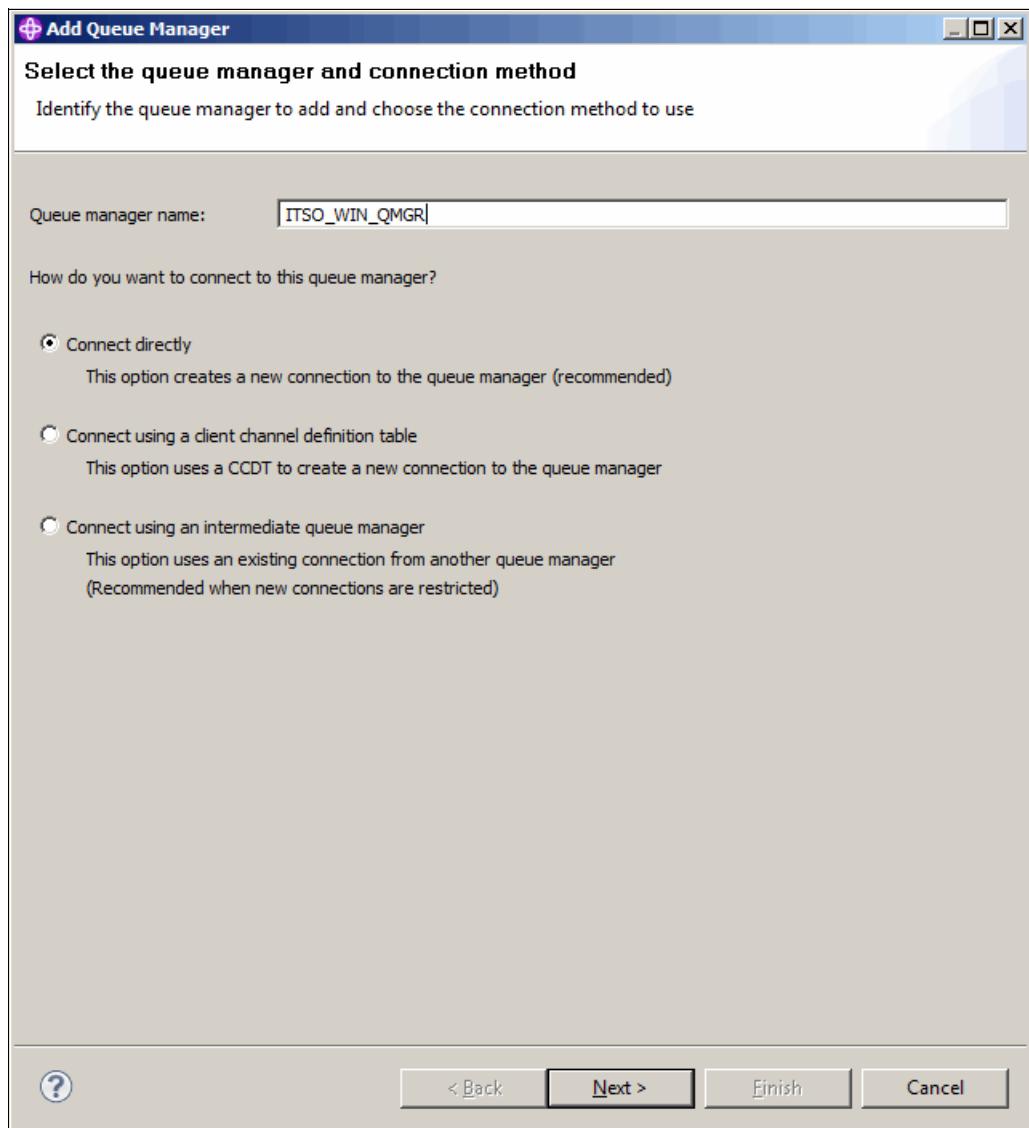


Figure 20-3 Adding the Windows queue manager to WebSphere MQ Explorer

12. Complete the connection details with the following parameters for WebSphere MQ Explorer to locate the queue manager to which to connect:
 - Host name or IP address: 9.44.168.135
 - Port number: 10000
 - Server-connection channel: EXPLORER.SVRCONN

Figure 20-4 shows the Specify connections details window.

Leave the remaining fields with the default values and click **Finish**.

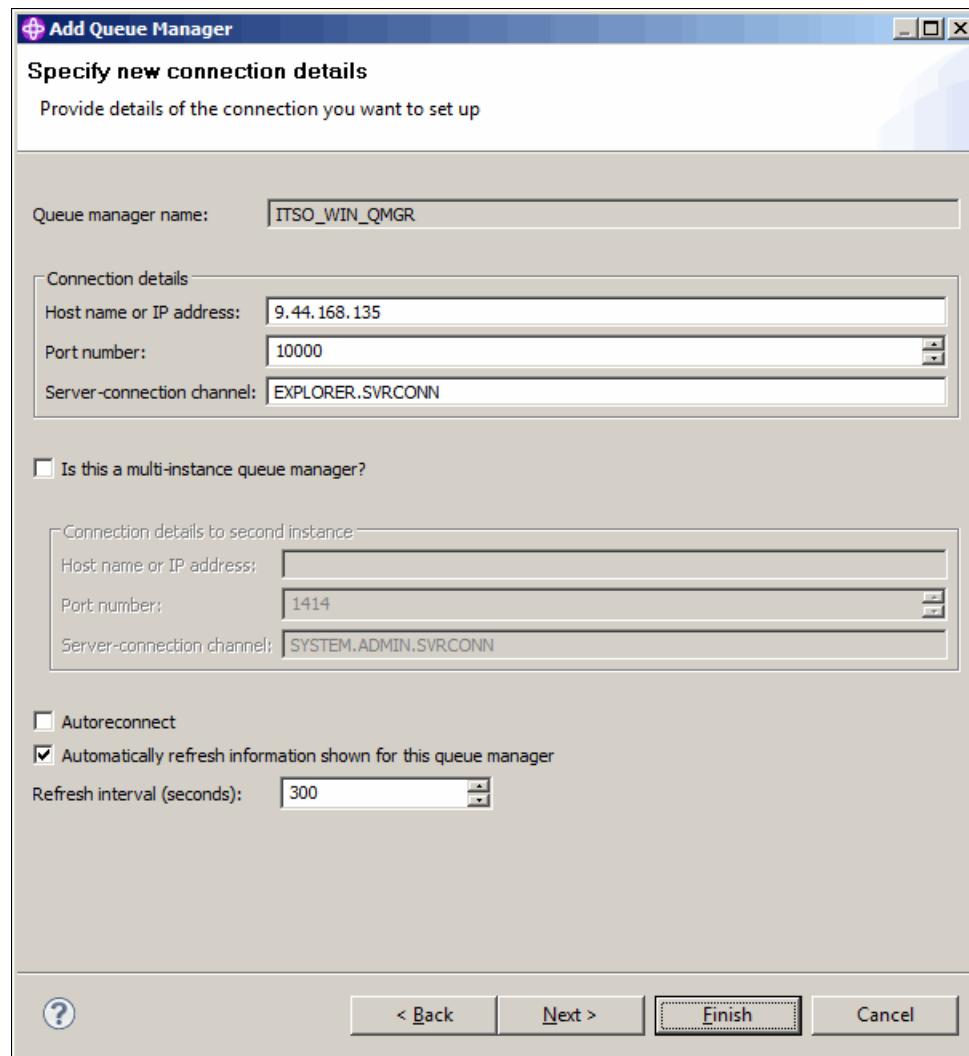


Figure 20-4 Specifying the connection details for the Windows queue manager ITSO_WIN_QMGR

13. After WebSphere MQ Explorer connects to ITSO_WIN_QMGR, the Navigator panel shows that the queue manager is available. Expand **ITSO_WIN_QMGR** → **Channels**. Figure 20-5 shows that the queue manager is connected to WebSphere MQ Explorer (the IP address and port is next to the queue manager name).

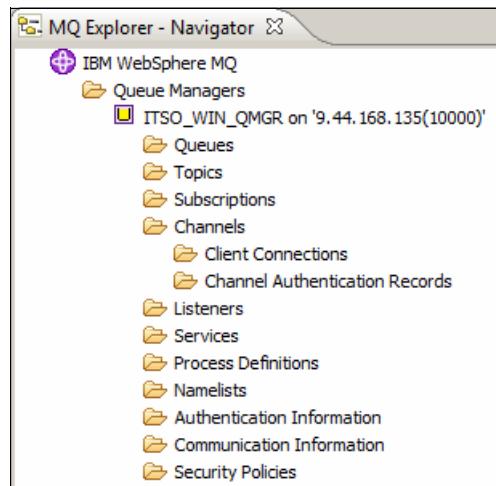


Figure 20-5 The ITSO_WIN_QMGR connected to WebSphere MQ Explorer

14. Click **Channels**. The Content window refreshes to show the non-default channel objects that are defined on the queue manager. Figure 20-6 shows the EXPLORER.SVRCNN channel definition that was previously defined (the MCA user ID field on the channel definition is blank).

Channel name	Channel type	Overall channel status	Conn name	Transmission queue	MCA user ID	Batch
EXPLORER.SVRCONN	Server-connection	Running				

Figure 20-6 The WebSphere MQ Explorer channel definition defined earlier

15. Right-click anywhere on the channel definition and select **Status → Channel Status...**

Figure 20-7 shows the channel status for EXPLORER.SVRCCONN. There is a difference between the channel definition that is shown in Figure 20-6 on page 287 and the channel status that is shown in Figure 20-7. The channel status shows information about the channel's operation. In this case, WebSphere MQ Explorer is running as user expl_admin and the channel authentication record mapped that user ID to win_admin, as shown in the MCA user ID field of the channel status.

Channel name	Channel type	Channel status	Conn name	MCA user ID
EXPLORER.SVRCCONN	Server-connection	Running	9.44.168.135	win_admin

Figure 20-7 The channel status that shows the mapped user ID in the MCA user ID field

Scenario resource configuration

Use WebSphere MQ Explorer to define the remaining resources for the Windows queue manager. The win_admin user ID that was mapped to needs authority to create queues and channels. This authority is granted by running the following commands from a command prompt:

```
setmqaut -m ITSO_WIN_QMGR -n ** -t queue -p win_admin +crt  
setmqaut -m ITSO_WIN_QMGR -n ** -t channel -p win_admin +crt
```

Complete the following steps to create the resources for this scenario:

1. Create the local queue definition, which is used as the endpoint of the message route. The queue is created with the name WIN_LOCAL_Q and all of the default values for the queue can be applied. With ITSO_WIN_QMGR expanded in WebSphere MQ Explorer, right-click **Queues** (see Figure 20-5 on page 287) and select **New → Local Queue...**. Enter the queue name and click **Finish**.
2. Create the RCVR channel. This channel is connected to by the SDR channel on the z/OS queue manager. The channel is named CSQ5.TO.ITSO_WIN. Right-click **Channels** (see Figure 20-5 on page 287) and select **New → Receiver Channel...**. Enter the channel name and click **Finish**.

This channel cannot be started until the z/OS SDR channel is created and started.

3. Create a QMGRMAP channel authentication record to map the z/OS queue manager name (CSQ5) to the local windows user ID (win_user). Right-click **Channel Authentication Records** under Channels (see Figure 20-5 on page 287) in the navigator panel and select **New → Channel Authentication Record...** to open the channel authentication record wizard.

This channel authentication rule allows access that is based on the mapping.

4. On the panel Create a Channel Authentication Record (see Figure 8-3 on page 112), select **Allow access** as the rule type. Click **Next**.
5. Select **Remote queue manager name** as the identity to match. Click **Next**.

6. In the Channel profile field, enter the RCVR channel name that was created in step 2 on page 289 and click **Show matching channels**. Figure 20-8 shows that this rule matches against only one channel (the channel that was previously created). Click **Next**.

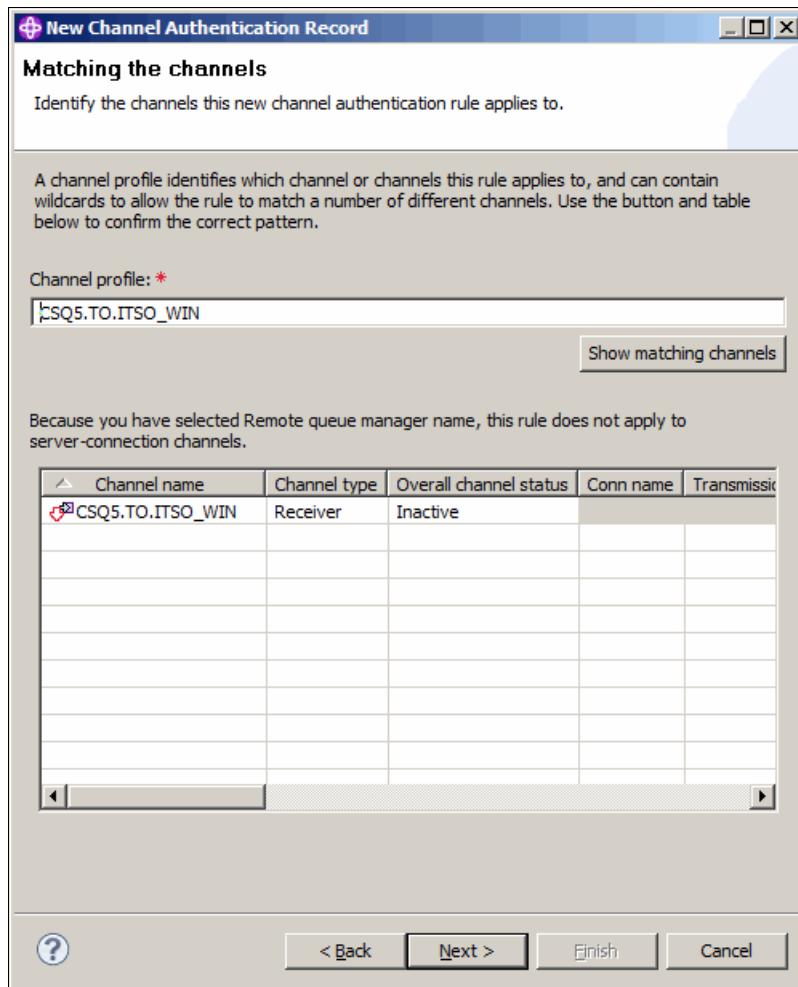


Figure 20-8 Verifying that the rule applies to the correct channel

7. In the Remote queue manager name pattern, enter the z/OS queue manager name, CSQ5. Optionally, specify an IP address on which this rule should match incoming connections. For this scenario, the IP address is set to 9.12.4.34. Click **Next**.
8. Select **Fixed user ID** and enter the user ID of the user to which should be mapped. In this case, the user ID is win_user. Click **Next** to supply a description. Click **Next** again for the rule summary.

9. Review the rule that the wizard configured and click **Finish**. Figure 20-9 shows the summary of the rule that was configured by the wizard and the MQSC command that is available for reuse.

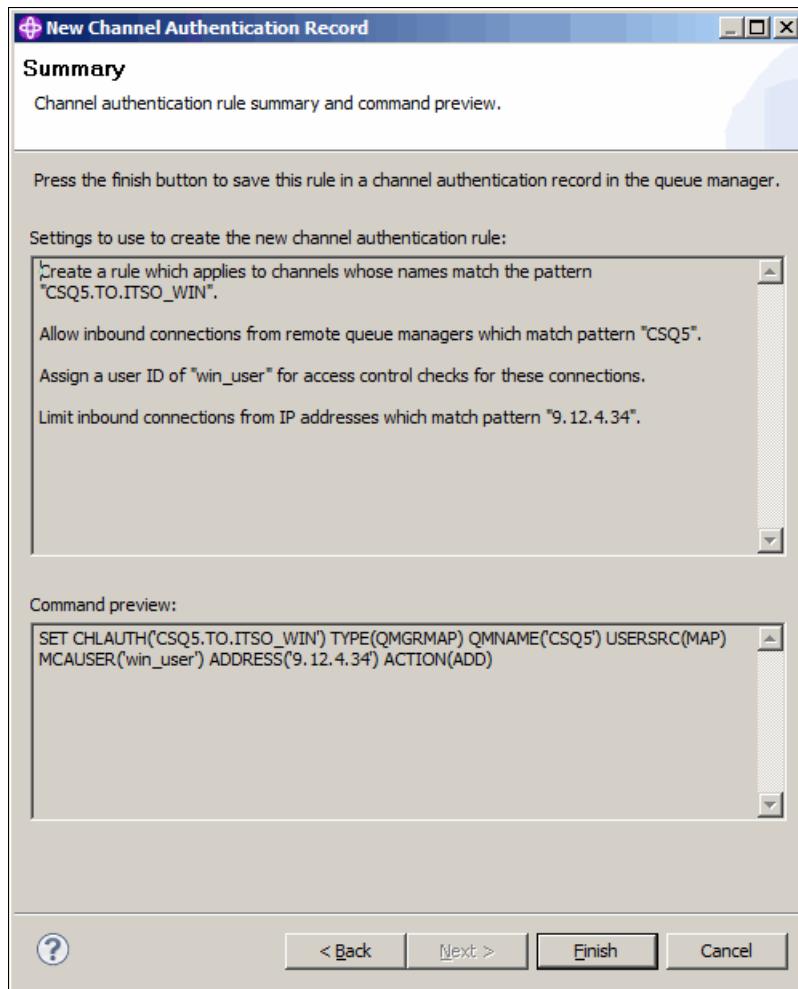


Figure 20-9 A summary of the QMGRMAP channel authentication record

The final resource that must be created on ITSO_WIN_QMGR is the BLOCKUSER channel authentication record, which ensures that the users on Linux and z/OS cannot directly access the WebSphere MQ resources that are configured on Windows.

10. As described in step 3 on page 289, open the channel authentication record wizard by right-clicking Channel Authentication Records and selecting **New → Channel Authentication Record...**
11. This record is a blocking record, so select **Block access** from the Create a Channel Authentication Record panel and click **Next**.
12. The user ID is used as the entity to match against. Because this ID is a BLOCKUSER channel authentication record, it must be run when all other mapping occurs. Therefore, select **Final assigned user ID** and click **Next**.
13. The rule is intended to block the specified users from connecting directly to the queue manager, so the channel that this rule should apply to effectively is all channels. In the Channel profile field, enter * and click **Next**.

14. The next panel is used to enter the details of the user IDs that are to be blocked. For the Windows queue manager, ITSO_WIN_QMGR, the user IDs to be blocked are the IDs that are in use on the other machines, lin_user, lin_admin from the Linux machine and CBOTH from z/OS. Figure 20-10 shows the completed panel. Click **Next** to provide a description of the rule (optional). Click **Next** to review the summary.

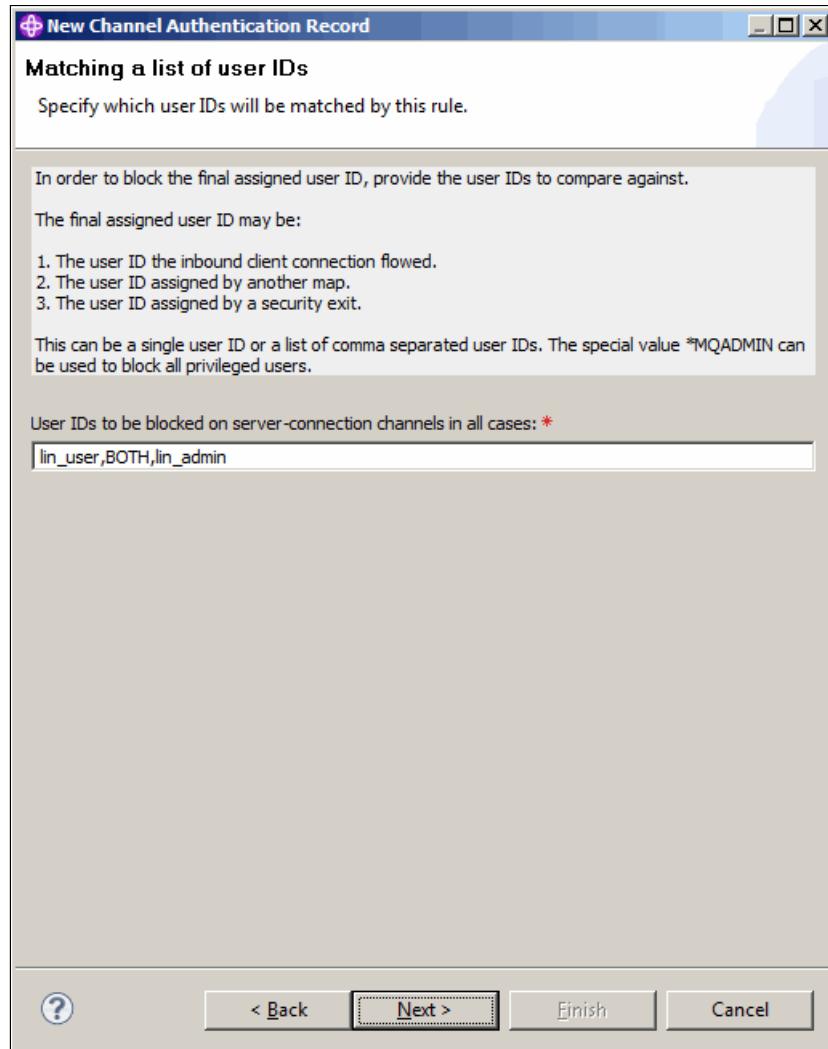


Figure 20-10 The user IDs to be blocked with this channel authentication record

15. Click **Finish** to create the rule. The existing default BLOCKUSER rule is updated with the user IDs that are specified instead of creating a rule.

20.2.2 Linux WebSphere MQ configuration details

Figure 20-11 shows the WebSphere MQ configuration for the Linux machine that is used in this scenario.

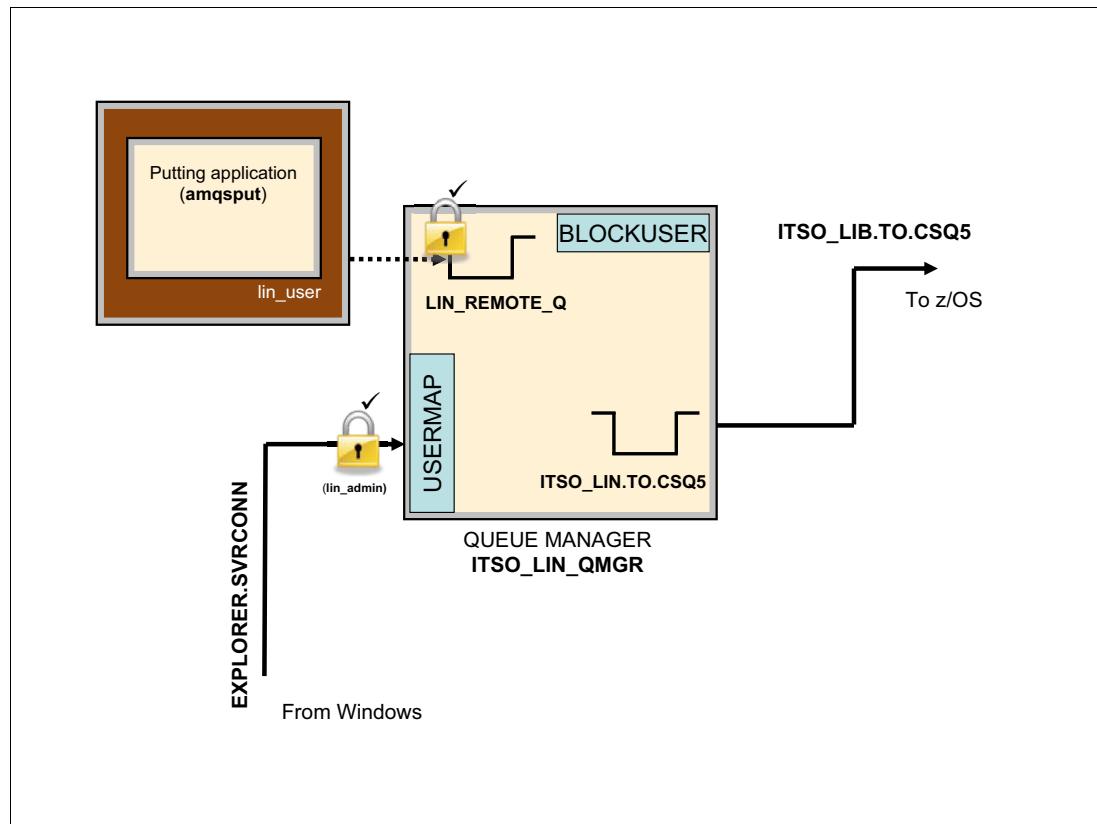


Figure 20-11 The WebSphere MQ configuration on the Linux machine

The following WebSphere MQ resources are defined on the Linux machine. This section describes the administration of those resources in detail:

- ▶ A single queue manager, which is configured with the following resources:
 - A single listener.
 - A remote queue definition, which represents the destination queue on the z/OS queue manager and is used as the start point to send messages.
 - A local queue, which is used as the transmission queue to place messages on the channel.
 - A sender (SDR) channel, which connects to the z/OS queue manager.
 - A server connection (SVRCONN) channel, to which WebSphere MQ Explorer connects.
 - A USERMAP channel authentication record to map WebSphere MQ Explorer user ID (expl_admin) to an authorized local Linux user ID (lin_admin).
 - A BLOCKUSER channel authentication record to block the user IDs on the other machines from accessing the Linux queue manager directly.
- ▶ The WebSphere MQ sample application amqspput, which runs as user lin_user.

Queue manager creation and initial configuration

Complete the following steps to create the queue manager and configure it to the point at which the remaining configuration can be done by using WebSphere MQ Explorer:

1. From a Linux terminal window, create the queue manager, ITSO_LIN_QMGR, by using the **crtmqm** command.
2. Start the queue manager, ITSO_LIN_QMGR, by using the **strmqm** command.

Example 20-6 shows the output from the creation and starting of ITSO_LIN_QMGR.

Example 20-6 Creating and starting the queue manager ITSO_LIN_QMGR

```
[saw224 ~]$ crtqm ITSO_LIN_QMGR
WebSphere MQ queue manager created.
Directory '/var/mqm/qmgrs/ITSO_LIN_QMGR' created.
The queue manager is associated with installation 'Installation2'.
Creating or replacing default objects for queue manager 'ITSO_LIN_QMGR'.
Default objects statistics : 71 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
[saw224 ~]$ strmqm ITSO_LIN_QMGR
WebSphere MQ queue manager 'ITSO_LIN_QMGR' starting.
The queue manager is associated with installation 'Installation2'.
5 log records accessed on queue manager 'ITSO_LIN_QMGR' during the log replay
phase.
Log replay for queue manager 'ITSO_LIN_QMGR' complete.
Transaction manager state recovered for queue manager 'ITSO_LIN_QMGR'.
WebSphere MQ queue manager 'ITSO_LIN_QMGR' started using V7.1.0.0.
```

3. Start runmqsc against the queue manager ITSO_LIN_QMGR, by using the command **runmqsc ITSO_LIN_QMGR**.
4. Within runmqsc, define and verify the channel definition that is used by WebSphere MQ Explorer to access ITSO_LIN_QMGR by using the following commands:
def channel(EXPLORER.SVRCONN) chltype(SVRCONN) TRPTYPE(TCP)
dis chl(EXPLORER.SVRCONN)

Example 20-7 shows the commands and output of defining the SVRCONN channel that is used by WebSphere MQ Explorer.

Example 20-7 Creating and verifying the SVRCONN channel on ITSO_LIN_QMGR

```
[saw224 ~]$ runmqsc ITSO_LIN_QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.
Starting MQSC for queue manager ITSO_LIN_QMGR.

def channel(EXPLORER.SVRCONN) chltype(SVRCONN) TRPTYPE(TCP)
  1 : def channel(EXPLORER.SVRCONN) chltype(SVRCONN) TRPTYPE(TCP)
AMQ8014: WebSphere MQ channel created.
dis ch1(EXPLORER.SVRCONN)
  2 : dis ch1(EXPLORER.SVRCONN)
AMQ8414: Display Channel details.
      CHANNEL(EXPLORER.SVRCONN)          CHLTYPE(SVRCONN)
      ALTDATE(2012-10-10)                ALTTIME(13.20.39)
      COMPHDR(NONE)                    COMPMSG(NONE)
      DESCRL( )                        DISCINT(0)
      HBINT(300)                      KAINT(AUTO)
      MAXINST(999999999)              MAXINSTC(999999999)
      MAXMSGL(4194304)                MCAUSER( )
      MONCHL(QMGR)                   RCVDATA( )
      RCVEXIT( )                     SCYDATA( )
      SCYEXIT( )                     SENDDATA( )
      SENDEXIT( )                    SHARECNV(10)
      SSLCAUTH(REQUIRED)            SSLCIPH( )
      SSLPEER( )                     TRPTYPE(TCP)
```

-
5. Define the USERMAP channel authentication record that maps WebSphere MQ Explorer's user ID (expl_admin) to the local Linux administration user ID (lin_admin) and verify the record, as shown in the following example:

```
set chlauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER('lin_admin')
dis chlauth(EXPLORER.SVRCONN)
```

Example 20-8 shows the creation and visual verification of the USERMAP channel authentication record by using the previous MQSC commands.

Example 20-8 Creating and verifying the channel authentication record in runmqsc

```
set chlauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER('lin_admin')
3 : set chlauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER('lin_admin')
AMQ8877: WebSphere MQ channel authentication record set.
dis chlauth(EXPLORER.SVRCONN)
4 : dis chlauth(EXPLORER.SVRCONN)
AMQ8878: Display channel authentication record details.
    CHLAUTH(EXPLORER.SVRCONN)           TYPE(USERMAP)
    DESCRL( )                          CUSTOM( )
    ADDRESS( )                        CLNTUSER(expl_admin)
    MCAUSER(lin_admin)                USERSRC(MAP)
    WARN(NO)                           ALTDATE(2012-10-10)
    ALTTIME(13.25.04)
```

6. Create and start the listener on port 10005, as shown in the following example:

```
def listener(ITS0_LIN_LIST) TRPTYPE(TCP) PORT(10005)
start listener(ITS0_LIN_LIST)
```

Example 20-9 shows the creation and starting of the listener for the queue manager, ITSO_LIN_QMGR.

Example 20-9 Creating and starting the listener in runmqsc

```
def listener(ITS0_LIN_LIST) TRPTYPE(TCP) PORT(10005)
5 : def listener(ITS0_LIN_LIST) TRPTYPE(TCP) PORT(10005)
AMQ8626: WebSphere MQ listener created.
start listener(ITS0_LIN_LIST)
6 : start listener(ITS0_LIN_LIST)
AMQ8021: Request to start WebSphere MQ listener accepted.
```

7. End the **runmqsc** session with the command **end**.
8. Grant authority for the user ID lin_admin to access the queue manager, the queues, and the channels on the queue manager by using the **setmqaut** command.

Example 20-10 shows the commands that are necessary to grant this authority. These commands are run as a machine-wide administrator. The second and third commands specify a wildcard value for the profile name, ‘**’, which represents all of the objects of the specified type (in this case, queue and channel).

Example 20-10 Granting the user ID lin_admin authority to administer queues and channels

```
[saw224 ~]$ setmqaut -m ITSO_LIN_QMGR -t qmgr -p lin_admin +all
The setmqaut command completed successfully.
```

```
[saw224 ~]$ setmqaut -m ITSO_LIN_QMGR -n '**' -t queue -p lin_admin +all
The setmqaut command completed successfully.
```

```
[saw224~]$ setmqaut -m ITSO_LIN_QMGR -n '**' -t channel -p lin_admin +all
The setmqaut command completed successfully.
```

9. From the Windows machine that is used as the central administration point, connect WebSphere MQ Explorer (which is running as user ID expl_admin) to ITSO_LIN_QMGR. Right-click **Queue Managers** and select **Add Remote Queue Manager...**
10. Enter the queue manager name in the field that is provided and click **Next**.
11. Complete the connection details for ITSO_LIN_QMGR with the following parameters:
 - Host name or IP address: 9.42.171.245
 - Port number: 10005
 - Server-connection channel: EXPLORER.SVRCONN
 Leave the remaining fields with default values and click **Finish**.
12. After WebSphere MQ Explorer connects to ITSO_LIN_QMGR, the navigator window shows both of the queue managers that were created. Figure 20-12 shows that ITSO_WIN_QMGR and ITSO_LIN_QMGR are connected. Expand ITSO_LIN_QMGR, click **Channels**, then right-click **EXPLORER.SVRCONN** and select **Status → Channel Status...**

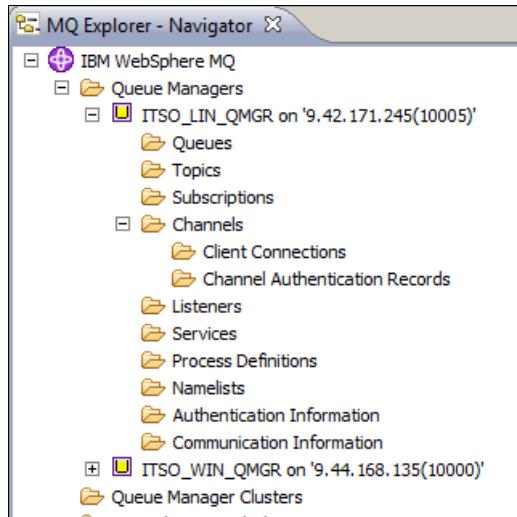


Figure 20-12 The WebSphere MQ Explorer Navigator pane with two remote queue managers

Figure 20-13 shows the channel status for EXPLORER.SVRCNN on ITSO_LIN_QMGR. The MCA user ID is mapped from expl_admin to lin_admin for this queue manager.

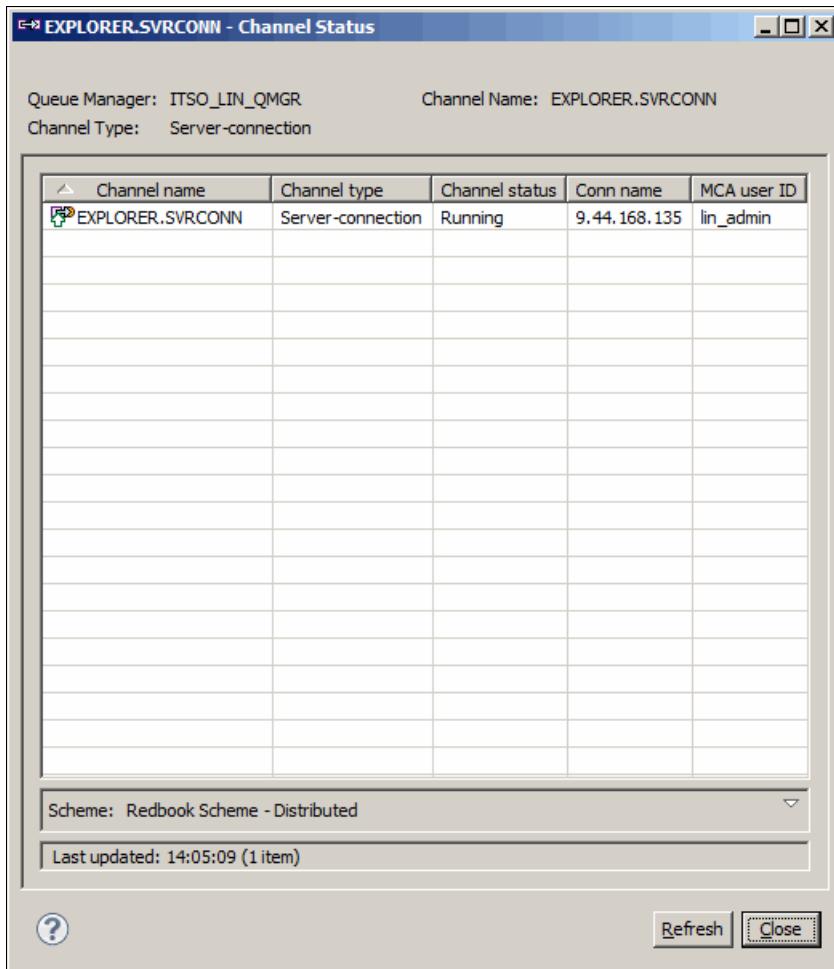


Figure 20-13 Channel status that shows the mapped user ID lin admin on ITSO LIN QMGR

Scenario resource configuration

Use WebSphere MQ Explorer to define the remaining resources for the Linux queue manager. The lin_admin user ID that was mapped to needs authority to create queues and channels. This authority is granted by running the following commands from a Linux terminal window:

```
setmqaut -m ITSO_LIN_QMGR -n '*' -t queue -p lin_admin +crt  
setmqaut -m ITSO_LIN_QMGR -n '*' -t channel -p lin_admin +crt
```

Complete the following steps to define the resources that are required for this scenario:

1. Create the transmission queue that is used to hold messages that are waiting to be sent to CSQ5 on z/OS. This queue is created with the name ITSO_LIN.TO.CSQ5. Expand the queue manager, ITSO_LIN_QMGR in WebSphere MQ Explorer, right-click **Queues** (see Figure 20-12 on page 297) and select **New → Local Queue...**
2. Enter the queue name, ITSO_LIN.TO.CSQ5, and click **Next**.
3. Change the Usage field to **Transmission**, as shown in Figure 20-14. Click **Finish**.

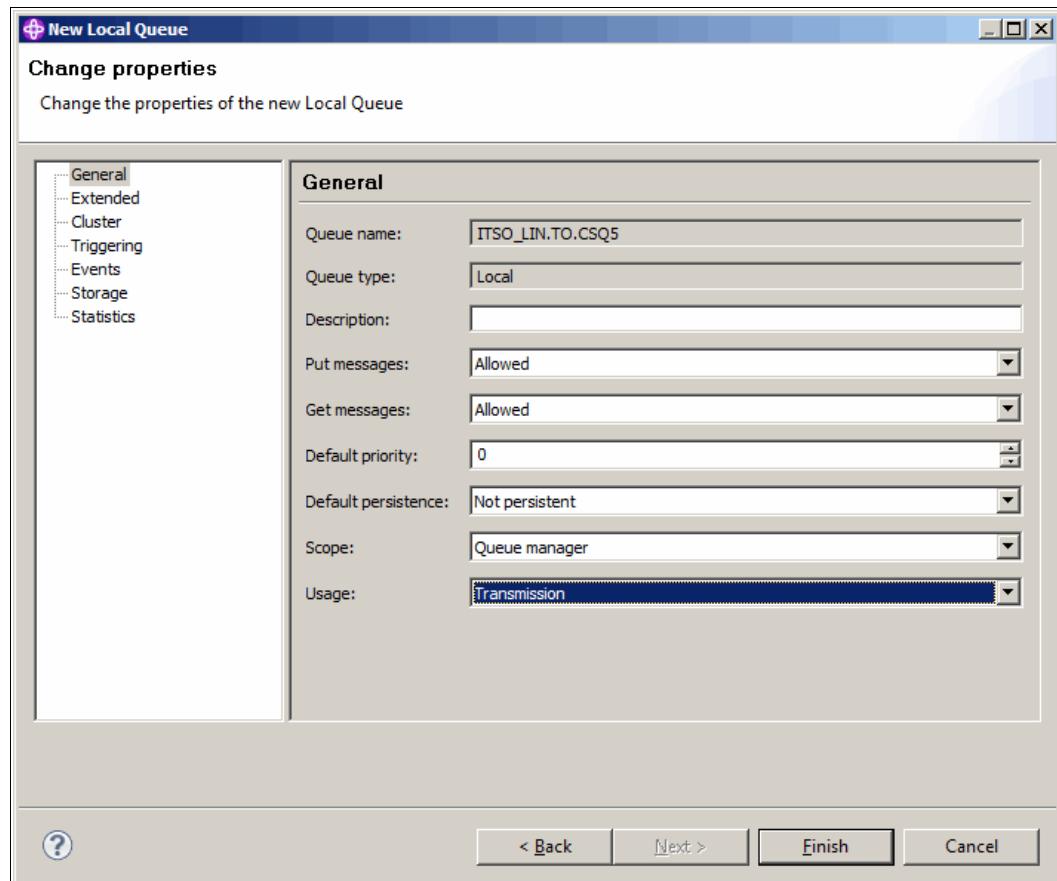


Figure 20-14 Changing the Usage of the queue from Normal to Transmission

4. Create the remote queue definition to which the user lin_user puts messages to start the messages flowing around the scenario. Right-click **Queues** and select **New → Remote Queue Definition...**
5. The remote queue is named LIN_REMOTE_Q. Enter the queue name and click **Next**.

6. Use the following values for the remote queue details:
 - Remote queue: CSQ5_ALIAS_Q
 - Remote queue manager: CSQ5
 - Transmission queue: Click **Select...** to select the transmission queue that was created, as shown in Figure 20-15. Select ITSO_LIN.TO.CSQ5. Click **OK**.

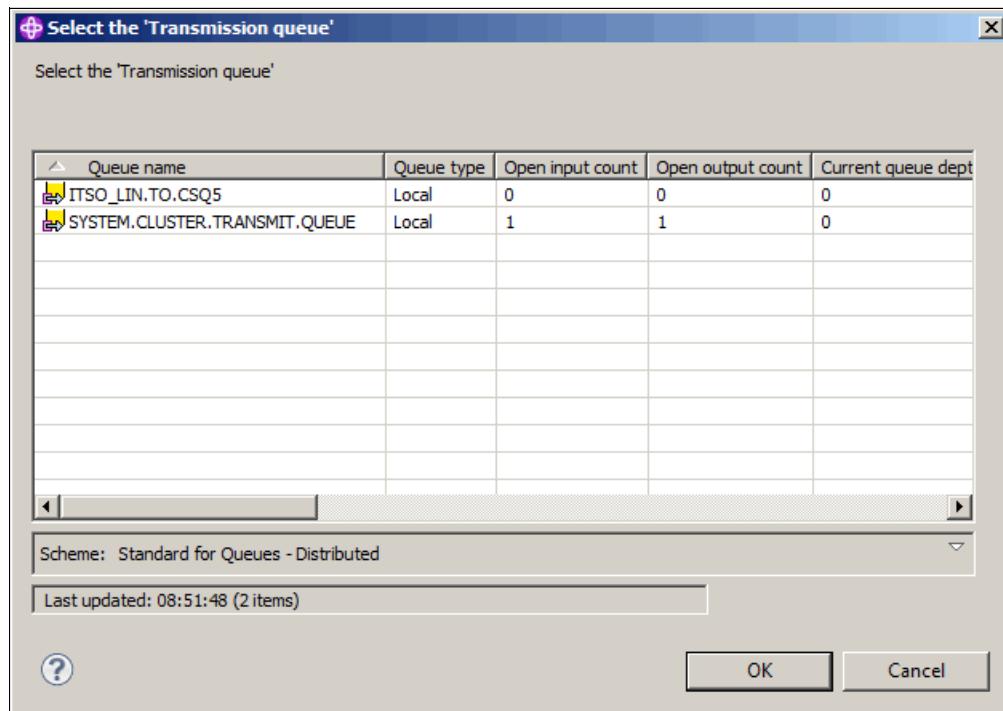


Figure 20-15 Selecting the previously defined transmission queue for the remote queue

7. Click **Finish** to create the remote queue definition.
8. Create the SDR channel from ITSO_LIN_QMGR to CSQ5. Under ITSO_LIN_QMGR, right-click **Channels** and select **New → Sender channel...**
9. The channel is named ITSO_LIN.TO.CSQ5. Enter the channel name and click **Next**.
10. There are two mandatory parameters to complete for the SDR channel: the Connection name and the Transmission queue name. Specify the following Connection name as the IP address and port of CSQ5 and click **Select...** (as shown in step 6 on page 300) to select the transmission queue for this channel:
 - Connection name: 9.12.4.34(1416)
 - Transmission queue name: ITSO_LIN.TO.CSQ5
11. Click **Finish** to create the SDR channel definition.
This SDR channel cannot be started because the corresponding RCVR channel has yet to be created on queue manager CSQ5.
12. Create the BLOCKUSER channel authentication record for ITSO_LIN_QMGR to block the users from the other systems from accessing the queue manager. The user IDs that are to be blocked are win_admin, win_user from the Windows machine, and the user ID 'CBOTH' from z/OS. Under ITSO_LIN_QMGR, expand **Channels**, right-click **Channel Authentication Records** and select **New → Channel Authentication Record...**
13. Select **Block access** and click **Next**.

14. Select **Final assigned user ID** and click **Next**.
15. This rule applies to all channels, so specify the Channel profile as * and click **Next**.
16. Enter the user IDs to block, in this case, win_user, win_admin, and CBOTH. Click **Next**.
17. Optionally, add a description and click **Next** to see the channel authentication record summary. Click **Finish** to update the rule with the specified user IDs.

20.2.3 z/OS WebSphere MQ configuration details

Figure 20-16 shows the WebSphere MQ configuration for the z/OS machine that is used in this scenario.

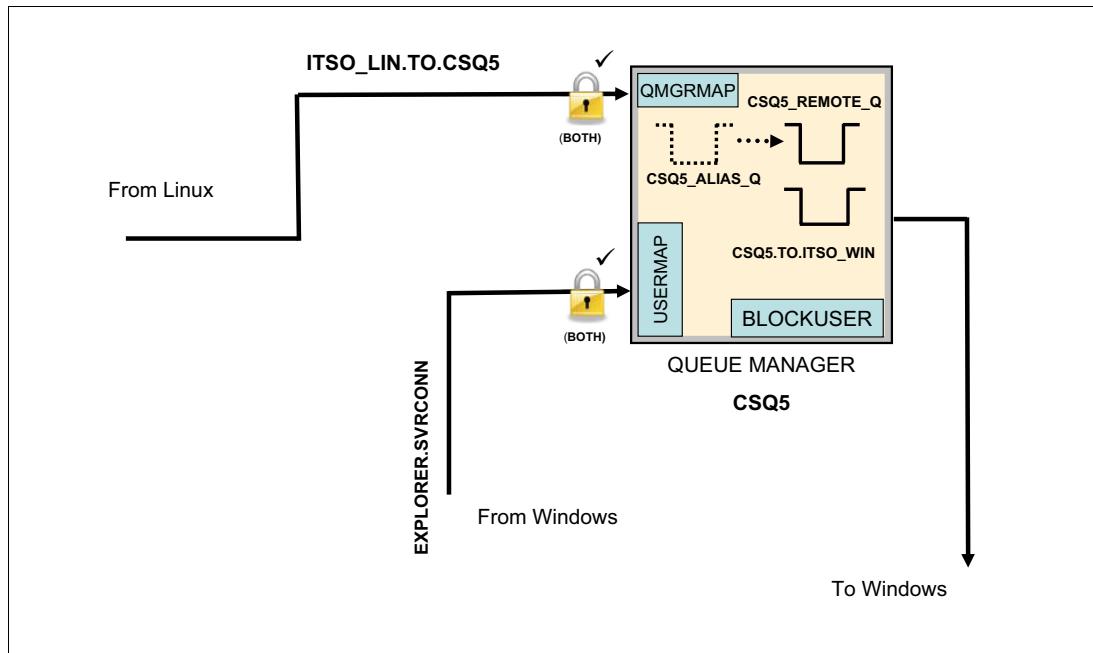


Figure 20-16 The WebSphere MQ configuration on the z/OS machine

The following WebSphere MQ resources are defined on the z/OS machine. This section describes the administration of those resources in detail:

- ▶ A single queue manager, which is configured with the following resources:
 - A single listener.
 - A remote queue definition, which represents the receiving queue on the Windows machine.
 - An alias queue definition, which represents the remote queue also defined on the z/OS queue manager.
 - A local queue, which is used as the transmission queue to place messages on the channel.
 - A receiver (RCVR) channel, which is connected to by the Linux queue manager.
 - A sender (SDR) channel, which is connecting to the Windows queue manager.
 - A QMGRMAP channel authentication record to map the Linux queue manager to a local z/OS user ID (CBOTH).
 - A server connection (SVRCONN) channel, which is connected to by WebSphere MQ Explorer.
 - A USERMAP channel authentication record to map WebSphere MQ Explorer user ID (expl_admin) to an authorized local z/OS user ID (CBOTH).
 - A BLOCKUSER channel authentication record to block the user IDs on the other machines from accessing the z/OS queue manager directly.

The starting point for the z/OS queue manager is that the queue manager exists and the listener is already running. The first step in enabling the remote configuration of the z/OS queue manager is to create the SVRCONN channel to which WebSphere MQ Explorer connects. The resource definition commands that are used in this section are MQSC commands that are run from the ISPF panels after **Action 8** is selected for Command.

Complete the following steps to configure the z/OS queue manager for this scenario:

1. Define EXPLORER.SVRCONN on queue manager CSQ5.

Example 20-11 shows the output from the following command:

```
def channel(EXPLORER.SVRCONN) chltype(SVRCONN) TRPTYPE(TCP)
dis ch1(EXPLORER.SVRCONN)
```

Example 20-11 Defining and verifying the WebSphere MQ Explorer SVRCONN channel in MQSC

```
CSQU000I CSQUTIL IBM WebSphere MQ for z/OS V7.1.0
CSQU001I CSQUTIL Queue Manager Utility - 2012-10-10 14:45:34
COMMAND TGTQMGR(CSQ5) RESPTIME(30)
CSQU127I Executing COMMAND using input from CSQUCMD data set
CSQU120I Connecting to CSQ5
CSQU121I Connected to queue manager CSQ5
CSQU055I Target queue manager is CSQ5
def channel(EXPLORER.SVRCONN) chltype(SVRCONN) trptype(TCP)
CSQN205I COUNT=      2, RETURN=00000000, REASON=00000000
CSQ9022I -CSQ5 CSQMACHL ' DEF CHANNEL' NORMAL COMPLETION
dis ch1(EXPLORER.SVRCONN)
CSQN205I COUNT=      3, RETURN=00000000, REASON=00000000
CSQM415I -CSQ5
  CHANNEL(EXPLORER.SVRCONN)
  CHLTYPE(SVRCONN)
  QSGDISP(QMGR)
  DEFCDISP(PRIVATE)
  TRPTYPE(TCP)
  DESCR( )
  DISCINT(0)
  SCYEXIT( )
  SCYDATA( )
  SENDEXIT( )
  SENDDATA( )
  RCVEXIT( )
  RCVDATA( )
  MAXINST(999999999)
  MAXINSTC(999999999)
  SHARECNV(10)
  PUTAUT(DEF)
  KAINT(AUTO)
  MONCHL(QMGR)
  ALTDATE(2012-10-10)
  ALTTIME(14.45.34)
  SSLCAUTH(REQUIRED)
  SSLCIPH( )
  SSLPEER( )
  MCAUSER( )
  MAXMSGL(4194304)
  COMPHDR(NONE)
  COMPMSG(NONE)
  HBINT(300)
CSQ9022I -CSQ5 CSQMDRTS ' DIS CHANNEL' NORMAL COMPLETION
```

- Define the USERMAP channel authentication record that maps WebSphere MQ Explorer's user ID (expl_admin) to the local authorized z/OS user ID ('CBOTH') and verify the mapping.

```
set chlauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin')
MCAUSER(CBOTH)
dis chlauth(EXPLORER.SVRCONN)
```

Example 20-12 shows the creation and visual verification of the USERMAP channel authentication record by using the **set** and **dis chlauth** MQSC commands.

Example 20-12 Creating and verifying the channel authentication record in MQSC on z/OS

```
CSQU000I CSQUTIL IBM WebSphere MQ for z/OS V7.1.0
CSQU001I CSQUTIL Queue Manager Utility - 2012-10-10 15:07:32
COMMAND TGTQMGR(CSQ5) RESPTIME(30)
CSQU127I Executing COMMAND using input from CSQUCMD data set
CSQU120I Connecting to CSQ5
CSQU121I Connected to queue manager CSQ5
CSQU055I Target queue manager is CSQ5
set chlauth(EXPLORER.SVRCONN) type(USERMAP) CLNTUSER('expl_admin') +
MCAUSER(CBOTH)
CSQN205I COUNT=      2, RETURN=00000000, REASON=00000000
CSQ9022I -CSQ5 CSQMSCA ' SET CHLAUTH' NORMAL COMPLETION
dis chlauth(EXPLORER.SVRCONN)
CSQN205I COUNT=      3, RETURN=00000000, REASON=00000000
CSQM454I -CSQ5
  CHLAUTH(EXPLORER.SVRCONN)
  TYPE(USERMAP)
  CLNTUSER(expl_admin)
  MCAUSER(CBOTH)
  USERSRC(MAP)
CSQ9022I -CSQ5 CSQMDRTS ' DIS CHLAUTH' NORMAL COMPLETION
```

Because the user ID that is mapped to (in this case, the user ID is 'CBOTH') already includes the appropriate authorities to access the queue manager resources, no other authorities are required for this scenario.

- From the Windows machine that is used as the central administration point, connect WebSphere MQ Explorer (which is running as user ID expl_admin) to CSQ5. Right-click **Queue Managers** and select **Add Remote Queue Manager...**
- Enter the queue manager name in the field that is provided and click **Next**.
- Complete the connection details for queue manager CSQ5 by using the following parameters:
 - Host name or IP address: 9.12.4.34
 - Port number: 1416
 - Server-connection channel: EXPLORER.SVRCONNLeave the remaining fields with default values and click **Finish**.

- After WebSphere MQ Explorer connects to CSQ5, the navigator pane shows all three of the queue managers that were created. Figure 20-17 shows that ITSO_WIN_QMGR, ITSO_LIN_QMGR, and CSQ5 are connected to WebSphere MQ Explorer. Expand **CSQ5**, click **Channels**, then right-click **EXPLORER.SVRCONN** and select **Status → Channel Status...**

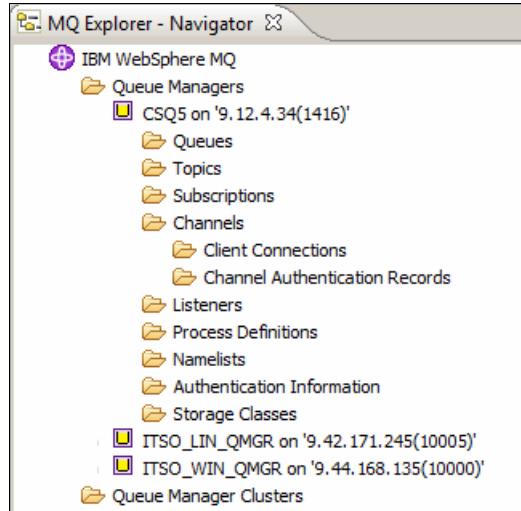


Figure 20-17 WebSphere MQ Explorer Navigator panel featuring three remote queue managers

The channel status for EXPLORER.SVRCONN on queue manager CSQ5 shows that the MCA user ID is mapped from expl_admin to CBOTH for this queue manager.

Scenario resource configuration

Use WebSphere MQ Explorer to define the remaining resources for the z/OS queue manager. After following through the steps on the Windows and Linux queue manager, complete the following steps to configure and link all of the resources:

- Create the transmission queue that is used to hold messages that are waiting to be moved to ITSO_WIN_QMGR. Expand queue manager CSQ5 in WebSphere MQ Explorer, right-click **Queues** (see Figure 20-17 on page 305) and select **New → Local Queue...**
- Specify the queue name as CSQ5.TO.ITSO_WIN and click **Next**.
- Change the queue's Usage parameter to **Transmission** and click **Finish** to create the queue.
- This queue manager includes three defined queues: the transmission queue that was previously created, an alias queue that routes any messages put to it to a remote queue definition which, in turn, routes the messages to the transmission queue to be sent over the channel. Create the remote queue definition. Right-click **Queues** and select **New → Remote Queue Definition...**
- Enter the queue name CSQ5_REMOTE_Q and click **Next**.
- The following configuration parameters for this queue should be used:
 - Remote queue: WIN_LOCAL_Q
 - Remote queue manager: ITSO_WIN_QMGR
 - Transmission queue: CSQ5.TO.ITSO_WIN
 Click **Finish** to create the queue.
- Create the alias queue definition. Right-click **Queues** and select **New → Alias Queue...**
- Enter the name of the queue as CSQ5_ALIAS_Q and click **Next**.

- The Base object parameter represents the actual queue definition to which this alias queue points. In this case, the queue that is to be used as the Base object is CSQ5_REMOTE_Q. Figure 20-18 shows the configurable options on the alias queue definition. Click **Finish** to create the alias queue.

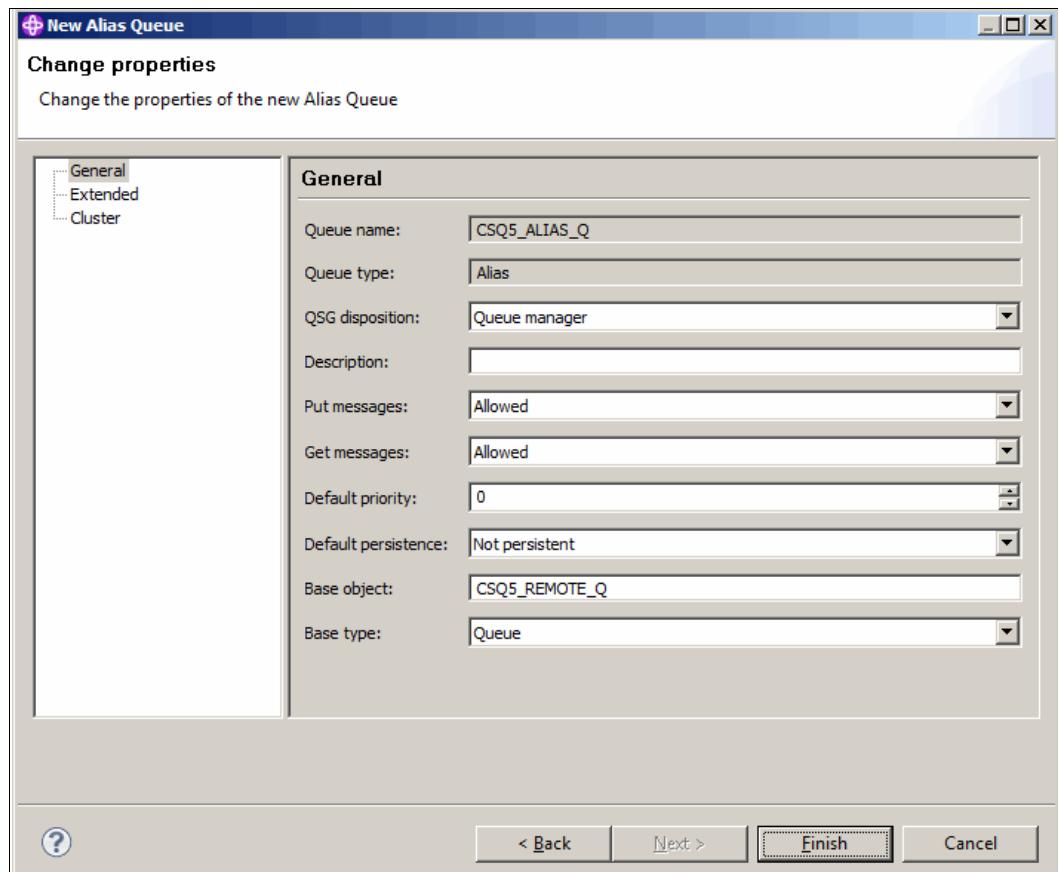


Figure 20-18 Configuring the alias queue definition on z/OS using WebSphere MQ Explorer

Complete the following steps to finalize the connections (channel definitions) between the queue managers:

- Under queue manager CSQ5 in WebSphere MQ Explorer, right-click **Channels** and select **New → Receiver Channel...** This channel is connected to by ITSO_LIN_QMGR. The channel name is ITSO_LIN.TO.CSQ5.
- Enter the channel name and click **Next**.
- Change the Transmission protocol on the RCVR channel from LU62 to TCP. Click **Finish** to create the channel.
- Create the sender (SDR) channel, which connects to ITSO_WIN_QMGR. Under CSQ5, right-click **Channels** and select **New → Sender Channel...** Enter the channel name CSQ5.TO.ITSO_WIN and click **Next**.
- The following Connection name and Transmission queue should be completed:
 - Connection name: 9.44.168.135(10000)
 - Transmission queue: CSQ5.TO.ITSO_WIN

Important: Channels that are created on z/OS include a default TRPTYPE of LU62. When the channel is created, in addition to the Connection name and Transmission queue, the Transmission protocol must be changed from LU62 to TCP.

The final steps of the configuration are to define the channel authentication records that are used to map and block functions when the channels start.

15. Define the QMGRMAP rule. Under CSQ5, expand **Channels** and right-click **Channel Authentication Records** and select **New → Channel Authentication Record...**
16. Select **Allow access** and click **Next**.
17. Select **Remote queue manager name** and click **Next**.
18. This rule applies only to the SDR/RCVR channel between CSQ5 and ITSO_LIN_QMGR. Enter the Channel profile as ITSO_LIN.TO.CSQ5 and click **Next**.
19. In the Remote queue manager name pattern field, enter ITSO_LIN_QMGR, and make the rule specific to the IP address of the machine on which the Linux queue manager is running. Enter 9.42.171.245 in the IP address pattern field, as shown in Figure 20-19 and click **Next**.

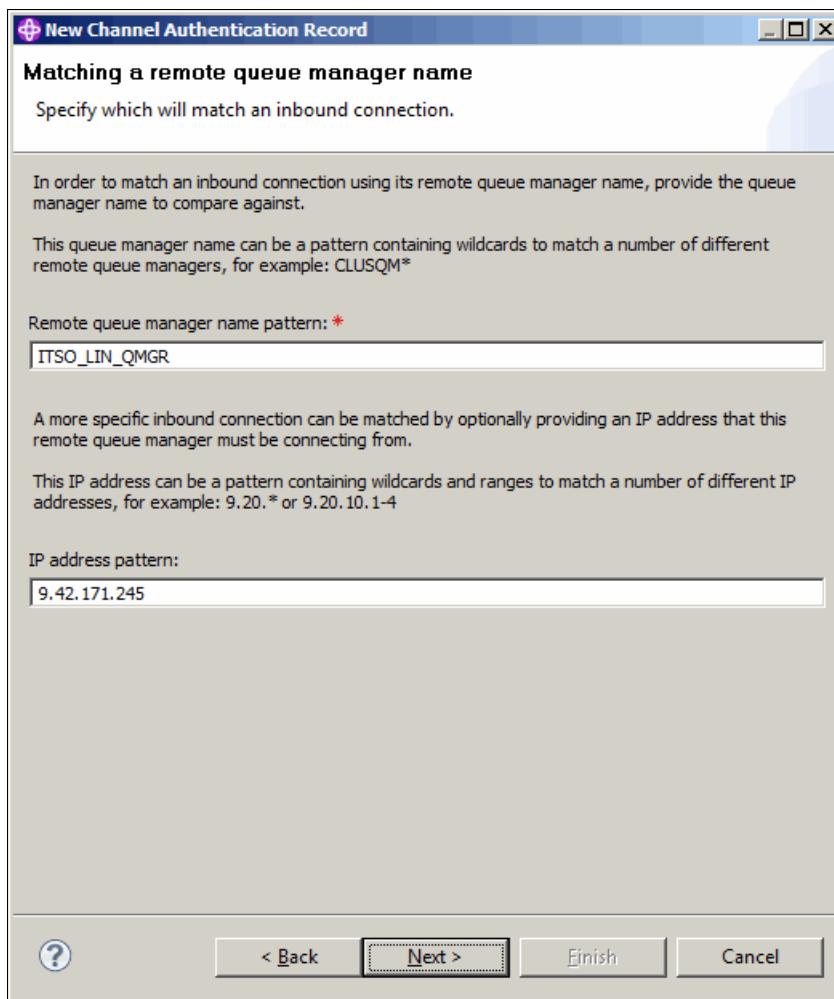


Figure 20-19 Configuring the details of the QMGRMAP rule on z/OS

20. Select **Fixed user ID** and specify the user ID **CBOTH** in the User ID field.
21. Click **Next** to optionally specify a description for this rule. Click **Next** to see the rule summary.
22. Click **Finish** to create the channel authentication record.

23. Create the BLOCKUSER channel authentication record that was used to block all other users from directly accessing the queue manager on z/OS. Under queue manager CSQ5 in WebSphere MQ Explorer, expand **Channels**, right-click **Channel Authentication Records** and select **New → Channel Authentication Record...**
24. Select **Block access** and click **Next**.
25. Select **Final assigned user ID** and click **Next**.
26. Specify the Channel profile as * because this rule applies to all channels. Click **Next**.
27. Specify the user IDs to be blocked. In this case, these IDs are win_admin, win_user, lin_admin, and lin_user. Click **Next**.
28. Enter a description for the rule (optional) and click **Next** to see a summary of the rule.
29. Click **Finish** to update the default rule with the specified user IDs.

20.3 User authorities

This section describes the authority that the entities win_user and lin_user need now that the WebSphere MQ environment is configured.

20.3.1 User authorities on Windows

The user ID win_user is used in this scenario for the following purposes:

- ▶ The mapped user ID on the RCVR channel enforces the messages that arrive from the z/OS queue manager CSQ5 to be put onto the destination queues with the authority that win_user has.
- ▶ The user ID under which the getting application amqsget is being run.

To allow the messages that are arriving from queue manager CSQ5 to be put on the intended destination queue, WIN_LOCAL_Q, the following authorities are required:

- ▶ On the queue manager, win_user is required to be able to connect, inquire, and set all of the context.
- ▶ On the destination queue, win_user must have the authority to put and set all of the context.

These authorities can be granted by using the following commands:

```
setmqaut -m ITSO_WIN_QMGR -t qmgr -p win_user +setall +connect +inq
setmqaut -m ITSO_WIN_QMGR -n WIN_LOCAL_Q -t queue -p win_user +setall +put
```

To allow win_user to get messages from the queue by using the **amqsget** command, the following authorities are required:

- ▶ The authorities that are granted with the previous commands are sufficient for connection to the queue manager.
- ▶ On the destination queue, win_user requires authority to get messages.

The get message authority is granted by using the following command:

```
setmqaut -m ITSO_WIN_QMGR -n WIN_LOCAL_Q -t queue -p win_user +get
```

20.3.2 User authorities on Linux

The user ID lin_user is used in this scenario to insert the messages into the system on ITSO_LIN_QMGR. To be allowed to put messages into the system, the user ID lin_user must be granted permission to connect to the queue manager, set all context on the queue manager, and put to the named remote queue definition. This authority can be granted by using the following commands:

```
setmqaut -m ITSO_LIN_QMGR -t qmgr -p lin_user +connect +setall  
setmqaut -m ITSO_LIN_QMGR -n LIN_REMOTE_Q -t queue -p lin_user +put
```

20.4 Starting channels and following a message

The steps in the previous sections completed the setup of the scenario. Complete the following steps to start the connections between the queue managers and put messages into the system:

1. In WebSphere MQ Explorer, under ITSO_LIN_QMGR, click **Channels** to display the channels that are defined on ITSO_LIN_QMGR.
2. Right-click the channel **ITSO_LIN.TO.CSQ5** and select **Start...** The channel list is refreshed and shows the channel in the Running state.
3. Select **Channels** under CSQ5 in WebSphere MQ Explorer to display the receiving side of the channel that was started in the previous step.
4. Right-click **ITSO_LIN.TO.CSQ5** and select **Status → Channel Status...** The channel is running with permissions that are effective for the user ID CBOTH. This user ID that is shown is the result of the QMGRMAP channel authentication record, which maps ITSO_LIN_QMGR to the user ID CBOTH.
5. Click **Close** to return to the channel definition panel for CSQ5. Right-click **CSQ5.TO.ITSO_WIN** and select **Start...** The connection between the z/OS queue manager and the windows queue manager starts.

Repeat steps 3 and 4 against the receiving side of the channel CSQ5.TO.ITSO_WIN on ITSO_WIN_QMGR to show that the channel is running with effective user ID permissions of win_user.

The full route of a message put to the queue LIN_REMOTE_Q on ITSO_LIN_QMGR to WIN_LOCAL_Q on ITSO_WIN_QMGR is completed by using the following process:

1. Message is put to LIN_REMOTE_Q with the sample amqsput running as user lin_user.
Example 20-13 shows the command to put a message (the terminal from which this command is run must be logged in as lin_user and the samples are in <MQ_INST_DIR>/samp/bin on the Linux machine).

Example 20-13 Command to put a message by using the sample application amqsput

```
[lin_user ~]$ /opt/mqm/samp/bin/amqsput LIN_REMOTE_Q ITSO_LIN_QMGR
Sample AMQSPUTO start
target queue is LIN_REMOTE_Q
This is a test message

Sample AMQSPUTO end
```

2. The command that is shown in Example 20-13 on page 310 puts the message to LIN_REMOTE_Q, which is configured to forward messages on to the z/OS queue CSQ5_ALIAS_Q via the transmission queue ITSO_LIN_TO_CSQ5 over the SDR channel of the same name.
3. Because the channel is mapped from the Linux queue manager name ITSO_LIN_QMGR to the z/OS user ID CBOTH, the message is placed on the target queue with the same permissions as the user ID CBOTH.
4. The queue CSQ5_ALIAS_Q refers to the remote queue definition CSQ5_REMOTE_Q. This remote queue definition is configured to forward messages on to the Windows queue manager ITSO_WIN_QMGR via the transmission queue CSQ5_TO_ITSO_WIN over the SDR channel of the same name.
5. This channel is configured to map the name of the z/OS queue manager to the Windows user ID win_user, and so the message is effectively put to WIN_LOCAL_Q with the permissions of the user ID win_user.
6. The user ID win_user then gets the message from the local queue, as shown in Example 20-14. (The samples are in <MQ_INST_DIR>\tools\c\Samples\Bin on Windows.)

Example 20-14 Receiving the message from the Windows queue manager

```
amqsget WIN_LOCAL_Q ITSO_WIN_QMGR
Sample AMQSGETO start
message <This is a test message>
no more messages
Sample AMQSGETO end
```

This scenario was implemented to allow all the channels to successfully connect and for the message to make a full round trip through the scenario. It is designed to show the points at which authorities can be applied, and how they can be mapped and blocked based on various credentials that are passed between the parties that are involved in the connections.

20.5 Testing the BLOCKUSER channel authentication records

The previous sections showed the QMGRMAP channel authentication records in use. By using a simple example, this section shows the effect of the BLOCKUSER rules that are defined in this scenario.

If the user `win_user` was created on the Linux machine in this scenario and it was granted group membership to the `mqm` group, by default a client that connects from a remote machine with the user ID `win_admin` is granted access to the queue manager `ITSO_LIN_QMGR`.

The BLOCKUSER rule is defined to block the user ID `win_admin` and any user in the `mqm` group from connecting. Therefore, with the simple definition of this rule, the queue manager remains protected from connections from unwanted user IDs.

This configuration is shown by simulating the following test scenario:

1. On the Windows machine, start WebSphere MQ Explorer as the user ID `win_user`.
2. Configure a connection to the remote queue manager `ITSO_LIN_QMGR`. Figure 20-20 shows a connection attempt from WebSphere MQ Explorer that is running as `win_user`.

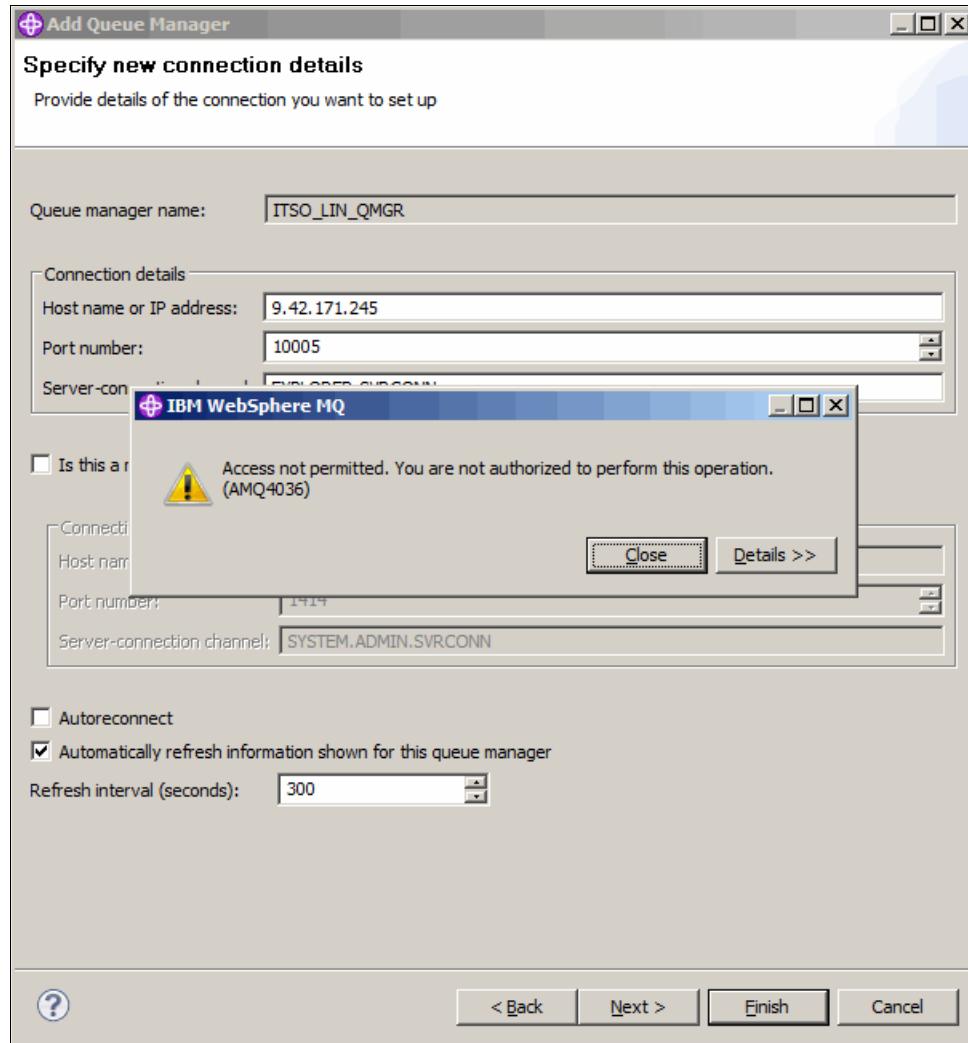


Figure 20-20 A connection attempt blocked by the BLOCKUSER rule

The client application (in this case, WebSphere MQ Explorer) receives an error message that states that authorization to perform the requested task was not permitted. The client side of the communication is not informed of the reason access was denied.

Example 20-15 shows the message that is output to the queue manager error logs, which shows that an attempt to connect was made, but the connection was blocked because of the user ID that was used.

Example 20-15 The log message that states a connection was blocked

```
Process(19831.10) User(saw224) Program(amqrmpa)
Host(sa-w224rhel-2.itso.ral.ibm.com) Installation(Installation2)
VRMF(7.1.0.0) QMgr(ITSO_LIN_QMGR)
```

```
AMQ9776: Channel was blocked by userid
```

EXPLANATION:

The inbound channel 'EXPLORER.SVRCCONN' was blocked from address '9.42.171.120' because the active values of the channel were mapped to a userid which should be blocked. The active values of the channel were 'MCAUSER(win_user) CLNTUSER(win_user)'.

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.

```
----- cmqxrmsa.c : 987 -----
```

Similar messages are generated for every platform, which describes where the connection came from, the effective user ID that the channel tried to run as, and the fact that it was blocked. The BLOCKUSER rule is run after any mapping rules take effect, so the error message might not contain the user ID that actually made the initial connection request.



Clustering: Multiple cluster transmission queues

This chapter contains an example that demonstrates how to configure a WebSphere MQ cluster to use multiple transmission queues. The scenario shows how a different transmission queue can be used for each cluster of which a queue manager is a member. The scenario also shows how a separate transmission queue can be used for each application.

These capabilities can help an administrator identify the cause of problems and reduce the effect of message backlogs on other applications, which can occur if a queue manager in a cluster is unavailable.

This chapter contains the following sections:

- ▶ Preparing the scenario
- ▶ Enabling the automatic creation of transmission queues
- ▶ Giving AP2 its own queue
- ▶ Creating an overlapping cluster
- ▶ Manually defining specific transmission queues with a manual switch

21.1 Preparing the scenario

In this scenario, you are shown how to configure an existing cluster to use multiple transmission queues. An overlapping cluster is then constructed for another application.

To prepare this scenario, an initial cluster configuration must be created, as shown in Figure 21-1. The cluster contains five queue managers and queues for two applications. The WebSphere MQ Put sample, amqsput, is used to represent the two applications, which are identified in this tutorial as AP1 and AP2. A different clustered queue is used for each application. The application AP1 puts to a queue called Q1. The application AP2 puts to a queue called Q2.

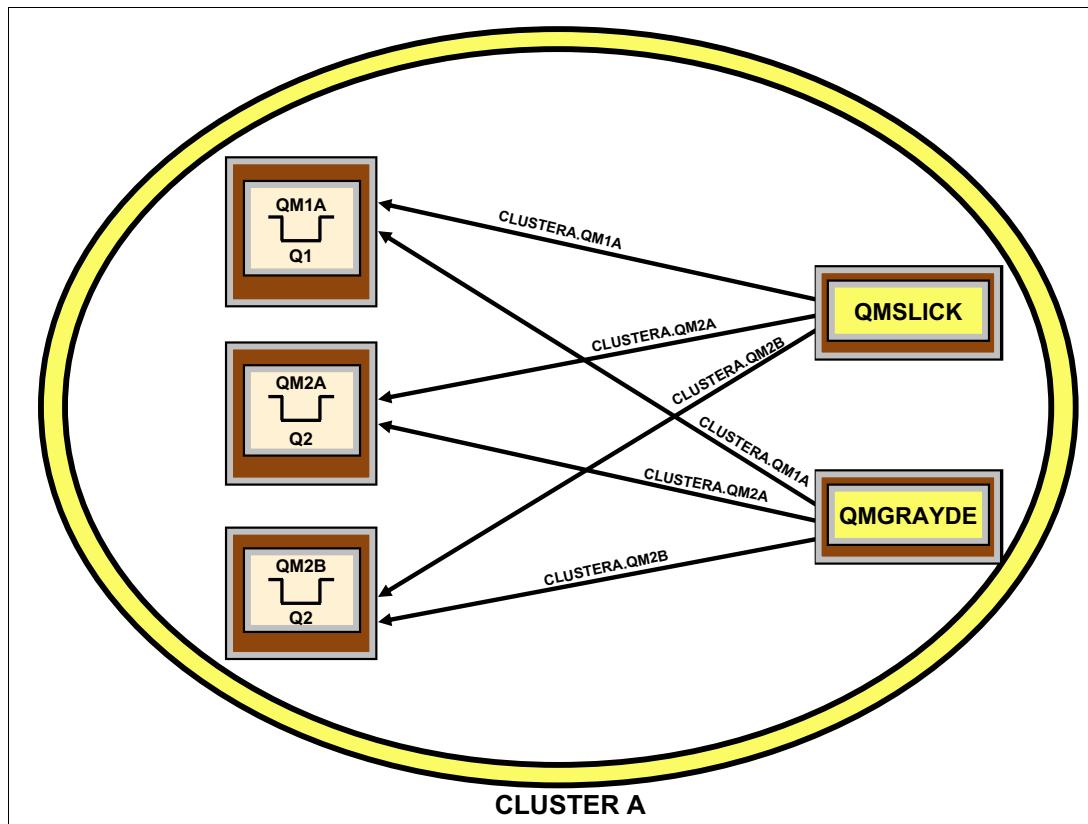


Figure 21-1 Initial cluster configuration: Channel names are shown on the cluster sender channels

In this scenario two, machines are used: one is Windows, the other Linux. Four of the queue managers are created on the Windows machine and the fifth queue manager is created on the Linux machine. To reproduce the scenario, the same machine can be used for all queue managers, or each queue manager can be created on a separate system. In the instructions that follow, the two machines are referred to as Windows (or Linux) to identify on which a step should be completed. WebSphere MQ V7.5, or later, must be used.

The following configuration for the four queue managers on the Windows machine is used:

- ▶ QMSLICK
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMGRAYDE for CLUSTER A
 - Is a full repository for CLUSTER A
- ▶ QM1A
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - A local queue that is called Q1 for CLUSTER A
 - Is a partial repository for CLUSTER A
- ▶ QM2A
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - A local queue that is called Q2 for CLUSTER A
 - Is a partial repository for CLUSTER A
- ▶ QM2B
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - A local queue that is called Q2 for CLUSTER A
 - Is a partial repository for CLUSTER A

The following configuration for the queue manager on the Linux machine is used:

- ▶ QMGRAYDE
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - Is a full repository for CLUSTER A

Complete the following steps to create the initial cluster environment:

1. On the Windows machine, create and start the four queue managers by using the following commands:

```
crtmqm QMSLICK  
crtmqm QM1A  
crtmqm QM2A  
crtmqm QM2B  
strmqm QMSLICK  
strmqm QM1A  
strmqm QM2A  
strmqm QM2B
```

2. On the Linux machine, create and start the other queue manager by using the following commands:

```
crtmqm QMGRAYDE  
strmqm QMGRAYDE
```

3. Define a listener on each queue manager to enable network communication. You do not have to use the port numbers that are indicated. If you want to use alternatives, you must adjust subsequent commands that refer to these values.
 - a. On the Windows machine, complete the following tasks:
 - i. On queue manager QMSLICK, run the following commands:


```
DEFINE LISTENER(L) TRPTYPE(TCP) port(1421) control(QMGR)
START LISTENER(L)
```
 - ii. On queue manager QM1A, run the following commands:


```
DEFINE LISTENER(L) TRPTYPE(TCP) port(1422) control(QMGR)
START LISTENER(L)
```
 - iii. On queue manager QM2A, run the following commands:


```
DEFINE LISTENER(L) TRPTYPE(TCP) port(1423) control(QMGR)
START LISTENER(L)
```
 - iv. On Queue manager QM2B, run the following commands:


```
DEFINE LISTENER(L) TRPTYPE(TCP) port(1424) control(QMGR)
START LISTENER(L)
```
 - b. On the Linux machine, run the following commands on queue manager QMGRAYDE:


```
DEFINE LISTENER(L) TRPTYPE(TCP) port(1431) control(QMGR)
START LISTENER(L)
```
4. Configure the queue managers QMSLICK and QMGRAYDE to be full repositories for the cluster. Later in this scenario, an overlapping cluster is created. A name list is used when the full repositories are configured to simplify the setup at that stage.
 - a. On the Windows machine, run the following commands on queue manager QMSLICK:


```
DEFINE NAMELIST(CLUSTER) NAMES(CLUSTERA)
ALTER QMGR REPOSNL(CLUSTER)
```
 - b. On the Linux machine, run the following commands on queue manager QMGRAYDE:


```
DEFINE NAMELIST(CLUSTER) NAMES(CLUSTERA)
ALTER QMGR REPOSNL(CLUSTER)
```
5. Configure the channels between the queue managers in the cluster so that each queue manager is connected to a queue manager where a full repository is maintained. The IP addresses that are shown in the following commands must be substituted for the machine (or machines) that you are using. The IP address 9.42.171.120 identifies the Windows machine, and the IP address 9.42.171.245 identifies the Linux machine.
 - a. On the Windows machine, perform the following tasks:
 - i. On queue manager QMSLICK, run the commands:


```
DEFINE CHANNEL(CLUSTERA.QMGRAYDE) CHLTYPE(CLUSSDR)
CONNAME('9.42.171.245(1431)') CLUSTER(CLUSTERA) BATCHHB(1)

DEFINE CHANNEL(CLUSTERA.QMSLICK) CHLTYPE(CLUSSRCVR)
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERA)
```
 - ii. On queue manager QM1A, run the commands:


```
DEFINE CHANNEL(CLUSTERA.QMSLICK) CHLTYPE(CLUSSDR)
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERA) BATCHHB(1)

DEFINE CHANNEL(CLUSTERA.QM1A) CHLTYPE(CLUSSRCVR)
CONNAME('9.42.171.120(1422)') CLUSTER(CLUSTERA)
```

iii. On queue manager QM2A, run the commands:

```
DEFINE CHANNEL(CLUSTERA.QMSLICK) CHLTYPE(CLUSSDR)
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERA) BATCHHB(1)
```

```
DEFINE CHANNEL(CLUSTERA.QM2A) CHLTYPE(CLUUSRCVR)
CONNAME('9.42.171.120(1423)') CLUSTER(CLUSTERA)
```

iv. On queue manager QM2B, run the commands:

```
DEFINE CHANNEL(CLUSTERA.QMSLICK) CHLTYPE(CLUSSDR)
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERA) BATCHHB(1)
```

```
DEFINE CHANNEL(CLUSTERA.QM2B) CHLTYPE(CLUUSRCVR)
CONNAME('9.42.171.120(1424)') CLUSTER(CLUSTERA)
```

b. On the Linux machine, run the following commands on queue manager QMGRAYDE:

```
DEFINE CHANNEL(CLUSTERA.QMSLICK) CHLTYPE(CLUSSDR)
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERA) BATCHHB(1)
```

```
DEFINE CHANNEL(CLUSTERA.QMGRAYDE) CHLTYPE(CLUUSRCVR)
CONNAME('9.42.171.245(1431)') CLUSTER(CLUSTERA)
```

6. Define the queues that are required for the two applications. A queue is required on each queue manager in the cluster that receives messages. Application AP1 sends messages to queue manager QM1A. Application AP2 sends messages to the queue managers QM2A and QM2B.

On the Windows machine, perform the following tasks:

a. On queue manager QM1A, run the following command:

```
DEFINE QL(Q1) CLUSTER(CLUSTERA) DEFBIND(NOTFIXED)
```

b. On queue manager QM2A, run the following command:

```
DEFINE QL(Q2) CLUSTER(CLUSTERA) DEFBIND(NOTFIXED)
```

c. On queue manager QM2B, run the following command:

```
DEFINE QL(Q2) CLUSTER(CLUSTERA) DEFBIND(NOTFIXED)
```

Important: Ensure that the listeners and channels are running before you continue. If a queue manager cannot communicate with other queue managers in the cluster, you might encounter errors in subsequent steps, such as the reason code 2085: MQRC_UNKNOWN_OBJECT_NAME.

Use the Put sample **amqsp** to send messages as application AP1, or application AP2, from the queue managers that are called QMSLICK and QMGRAYDE. The messages are delivered to the queues Q1 and Q2 on the other queue managers.

To send messages from the Windows machine as application AP1, run the following command:

```
amqsp Q1 QMSLICK
```

To send messages from the Windows machine as application AP2, run the following command:

```
amqsp Q2 QMSLICK
```

To send messages from the Linux machine, run the same **amqsp** commands, but replace QMSLICK with QMGRAYDE.

To send a message by using the **amqspput** command, enter some text, such as Hello World, and press Enter. To end the application when you finished sending requests, press Enter without entering any text beforehand.

To confirm that messages were delivered correctly, you can examine the number of messages on each application queue. To examine the number of messages on the queue Q1 on queue manager QM1A, use the following MQSC command:

DISPLAY QL(Q1) CURDEPTH

Sample output from **runmqsc** when this command is run is shown in Example 21-1.

Example 21-1 Checking the CURDEPTH on Q1

AMQ8409: Display Queue details.
QUEUE(Q1) TYPE(QLLOCAL)
CURDEPTH(6)

Examine the number of messages on queues Q1 and Q2 on the other queue managers.

After the number of messages are examined, use the **CLEAR QL** command to delete the messages on the three queues. Clearing the queues makes it easier to determine the outcome of subsequent steps that rerun **amqspput** to send further messages.

To delete the messages on the queue Q1 on queue manager QM1A, use **runmqsc** to connect to QM1A, then run the following command:

CLEAR QL(Q1)

Important: You might encounter an error when a queue is cleared that indicates the queue is in use. This error occurs because WebSphere MQ channels do not close queues immediately to avoid repeatedly opening and closing the same queue. You should be able to rerun the command successfully after you wait a few seconds. Alternatively, use the **amqsget** command to remove the messages.

21.2 Enabling the automatic creation of transmission queues

The initial cluster configuration is complete, so the use of multiple transmission queues can be enabled. Messages are sent only from the queue managers QMSLICK and QMGRAYDE, so these managers are the only queue managers that must be updated.

On the Windows machine, run the following command on queue manager QMSLICK:

```
ALTER QMGR DEFCLXQ(CHANNEL)
```

On the Linux machine, run the following command on queue manager QMGRAYDE:

```
ALTER QMGR DEFCLXQ(CHANNEL)
```

Each cluster-sender channel on QMSLICK and QMGRAYDE must be restarted for the configuration change to become effective.

On the Windows machine, run the following commands on queue manager QMSLICK:

```
STOP CHANNEL(CLUSTERA.QM1A)
STOP CHANNEL(CLUSTERA.QM2A)
STOP CHANNEL(CLUSTERA.QM2B)
START CHANNEL(CLUSTERA.QM1A)
START CHANNEL(CLUSTERA.QM2A)
START CHANNEL(CLUSTERA.QM2B)
```

On the Linux machine, run the following commands on queue manager QMGRAYDE:

```
STOP CHANNEL(CLUSTERA.QM1A)
STOP CHANNEL(CLUSTERA.QM2A)
STOP CHANNEL(CLUSTERA.QM2B)
START CHANNEL(CLUSTERA.QM1A)
START CHANNEL(CLUSTERA.QM2A)
START CHANNEL(CLUSTERA.QM2B)
```

Important: In 21.1, “Preparing the scenario” on page 314, the cluster configuration was verified by sending messages. If you did not put several messages (10 is more than enough), a channel might not be created between every pair of queue managers. This issue is not a problem. It means that when you try to stop the channels, one or more of the commands might fail because the channel does not exist. If a channel is created, it uses the value of the DEFCLXQ attribute.

The two queue managers are configured to use multiple cluster transmission queues. The use of multiple transmission queues can be confirmed by displaying the status of the cluster sender channels. Example 21-2 shows sample output from using the **runmqsc** command to display this status information about queue manager QMSLICK.

Example 21-2 Displaying the channel status of cluster sender channels

```
DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
  1 : DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTER.A.QMGRAYDE)          CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.245(1431))          CURRENT
  RQMNAME(QMGRAYDE)                   STATUS(RUNNING)
  SUBSTATE(MQGET)                     XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTER.A.QM1A)              CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1422))          CURRENT
  RQMNAME(QM1A)                      STATUS(RUNNING)
  SUBSTATE(MQGET)                     XMITQ(SYSTEM.CLUSTER.TRANSMIT.CLUSTER.A.QM1A)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTER.A.QM2A)              CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1423))          CURRENT
  RQMNAME(QM2A)                      STATUS(RUNNING)
  SUBSTATE(MQGET)                     XMITQ(SYSTEM.CLUSTER.TRANSMIT.CLUSTER.A.QM2A)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTER.A.QM2B)              CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1424))          CURRENT
  RQMNAME(QM2B)                      STATUS(RUNNING)
  SUBSTATE(MQGET)                     XMITQ(SYSTEM.CLUSTER.TRANSMIT.CLUSTER.A.QM2B)
```

When the status information for active channels is displayed, the transmission queue that is used by each channel is identified via the XMITQ attribute. Example 21-2 shows that the channels CLUSTER.A.QM1A, CLUSTER.A.QM2A, and CLUSTER.A.QM2B are using a different transmission queue. These queues were created dynamically by the queue manager. The channel CLUSTER.A.QMGRAYDE continues to use the SYSTEM.CLUSTER.TRANSMIT.QUEUE transmission queue because it was not restarted.

To ensure that messages are still delivered correctly within the cluster, put several messages to the queues Q1 and Q2 by using the **amqsput** command. To confirm that the messages were delivered, check the depth of each destination queue again. Clear the queues afterward to make it easier to confirm the outcome of subsequent steps.

21.3 Giving AP2 its own queue

Application AP2 sends messages to the queue Q2. This queue was defined only on the queue managers QM2A and QM2B. These queue manager names begin with a common prefix, QM2, which is unique within the cluster. It is possible to configure a manually defined transmission queue to be used exclusively for channels to these queue managers because the channel names start with a unique prefix.

Define a local transmission queue that is called QUEUE.AP2 on the queue managers QMSLICK and QMGRAYDE by using the following command:

```
DEFINE QLOCAL(QUEUE.AP2) USAGE(XMITQ) CLCHNAME(CLUSTER.A.QM2*)
```

For this change to become effective, only the cluster sender channels to QM2A and QM2B must be restarted.

On the Windows machine, run the following commands on queue manager QMSLICK:

```
STOP CHANNEL(CLUSTER.A.QM2A)  
STOP CHANNEL(CLUSTER.A.QM2B)  
START CHANNEL(CLUSTER.A.QM2A)  
START CHANNEL(CLUSTER.A.QM2B)
```

On the Linux machine, run the following commands on queue manager QMGRAYDE:

```
STOP CHANNEL(CLUSTER.A.QM2A)  
STOP CHANNEL(CLUSTER.A.QM2B)  
START CHANNEL(CLUSTER.A.QM2A)  
START CHANNEL(CLUSTER.A.QM2B)
```

After the channels are restarted, both channels on each queue manager should use the same transmission queue. Display the channel status again to confirm that the channels are using the manually defined transmission queue called QUEUE.AP2. The manually defined transmission queue is used because it takes priority over the value of the DEFCLXQ attribute.

Example 21-3 shows sample output from the use of the `runmqsc` command to display this status information about queue manager QMSLICK. The other cluster sender channels are unaffected by this change.

Example 21-3 Displaying the channel status of cluster sender channels

```
DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
 1 : DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QMGRAYDE)          CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.245(1431))        CURRENT
  RQMNAME(QMGRAYDE)                  STATUS(RUNNING)
  SUBSTATE(MQGET)                   XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QM1A)            CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1422))      CURRENT
  RQMNAME(QM1A)                    STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(SYSTEM.CLUSTER.TRANSMIT.CLUSTERA.QM1A)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QM2A)            CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1423))      CURRENT
  RQMNAME(QM2A)                    STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(QUEUE.AP2)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QM2B)            CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1424))      CURRENT
  RQMNAME(QM2B)                    STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(QUEUE.AP2)
```

21.4 Creating an overlapping cluster

To introduce more complexity, an overlapping cluster that is called CLUSTERB is now configured. An overlapping cluster occurs when a queue manager is a member of two or more clusters. Two other queue managers are required, called QM3A and CSQ4. In this scenario, the queue manager QM3A is created on the Linux machine and the queue manager CSQ4 is created on z/OS by using WebSphere MQ for z/OS V7.1. As when the original cluster configuration was set up, the machine or platform on which the queue managers are created is not important. A z/OS queue manager is used in this scenario to show that if the sending queue manager was configured to use multiple transmission queues, this configuration does not require the receiving queue manager to support this feature.

The application amqspput is used to send messages to simulate another application called AP3. This application sends messages to a queue called Q3, which is defined on the other queue managers.

Figure 21-2 shows the overlapping cluster configuration; the channel names are shown on the cluster sender channels. The first cluster, CLUSTERA, contains the queue managers that were previously used in this scenario. The second cluster, CLUSTERB, contains the two other queue managers. The queue managers QMSLICK and QMGRAYDE are members of both clusters.

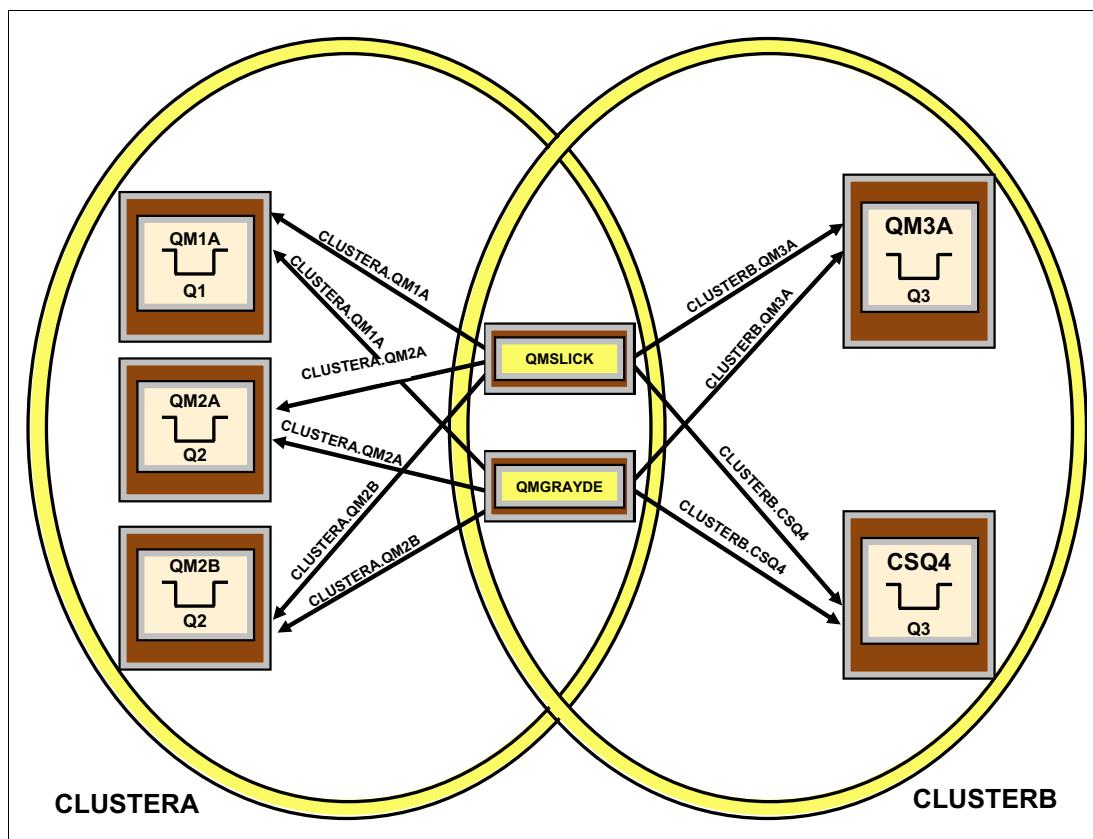


Figure 21-2 Overlapping cluster configuration

The following revised configuration on the Windows machine is used:

- ▶ QMSLICK:
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMGRAYDE for CLUSTER A
 - Is a full repository for CLUSTER A
 - A cluster receiver channel for CLUSTER B
 - A cluster sender channel to QMGRAYDE for CLUSTER B
 - Is a full repository for CLUSTER B
- ▶ QM1A:
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - A local queue that is called Q1 for CLUSTER A
 - Is a partial repository for CLUSTER A
- ▶ QM2A:
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - A local queue that is called Q2 for CLUSTER A
 - Is a partial repository for CLUSTER A
- ▶ QM2B:
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - A local queue that is called Q2 for CLUSTER A
 - Is a partial repository for CLUSTER A

The following revised configuration on the Linux machine is used:

- ▶ QMGRAYDE:
 - A single listener
 - A cluster receiver channel for CLUSTER A
 - A cluster sender channel to QMSLICK for CLUSTER A
 - Is a full repository for CLUSTER A
 - A cluster receiver channel for CLUSTER B
 - A cluster sender channel to QMSLICK for CLUSTER B
 - Is a full repository for CLUSTER B
- ▶ QM3A:
 - A single listener
 - A cluster receiver channel for CLUSTER B
 - A cluster sender channel to QMSLICK for CLUSTER B
 - A local queue that is called Q3 for CLUSTER B
 - Is a partial repository for CLUSTER B

The following configuration on the z/OS machine is used:

- ▶ CSQ4
 - A single listener
 - A cluster receiver channel for CLUSTERB
 - A cluster sender channel to QMSLICK for CLUSTERB
 - A local queue that is called Q3 for CLUSTERB
 - Is a partial repository for CLUSTER

Complete the following steps to implement the overlapping cluster environment:

1. On the Linux machine, create and start the QM3A queue manager by using the following commands:

```
crtmqm QM3A  
strmqm QM3A
```

2. Each queue manager needs a listener to enable network communication. If you want to use alternative port numbers, you must adjust subsequent commands that refer to these values.

On the Linux machine, run the following commands on queue manager QM3A:

```
DEFINE LISTENER(L) TRPTYPE(TCP) PORT(1432) CONTROL(QMGR)  
START LISTENER(L)
```

3. Configure the queue managers QMSLICK and QMGRAYDE to be full repositories for CLUSTERB.

On the Windows machine, run the following command on queue manager QMSLICK:

```
ALTER NAMELIST(CLUSTER) NAMES(CLUSTERA,CLUSTERB)
```

On the Linux machine, run the following command on queue manager QMGRAYDE:

```
ALTER NAMELIST(CLUSTER) NAMES(CLUSTERA,CLUSTERB)
```

4. Configure the channels between the queue managers in cluster CLUSTERB. The IP addresses that are shown in the following commands must be substituted for the machine (or machines) that you are using. The IP address 9.42.171.120 identifies the Windows machine. The IP address 9.42.171.245 identifies the Linux machine. The IP address 9.12.4.34 identifies z/OS.

5. On the Windows machine, run the following commands on queue manager QMSLICK:

```
DEFINE CHANNEL(CLUSTERB.QMGRAYDE) CHLTYPE(CLUSSDR)  
CONNAME('9.42.171.245(1431)') CLUSTER(CLUSTERB) BATCHHB(1)  
DEFINE CHANNEL(CLUSTERB.QMSLICK) CHLTYPE(CLUSSDR)  
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERB)
```

6. On the Linux machine, complete the following tasks:

- a. On queue manager QMGRAYDE, run the following commands:

```
DEFINE CHANNEL(CLUSTERB.QMSLICK) CHLTYPE(CLUSSDR)  
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERB) BATCHHB(1)  
DEFINE CHANNEL(CLUSTERB.QMGRAYDE) CHLTYPE(CLUSSDR)  
CONNAME('9.42.171.245(1431)') CLUSTER(CLUSTERB)
```

- b. On queue manager QM3A, run the following commands:

```
DEFINE CHANNEL(CLUSTERB.QMSLICK) CHLTYPE(CLUSSDR)  
CONNAME('9.42.171.120(1421)') CLUSTER(CLUSTERB) BATCHHB(1)  
DEFINE CHANNEL(CLUSTERB.QM3A) CHLTYPE(CLUSSDR)  
CONNAME('9.42.171.245(1432)') CLUSTER(CLUSTERB)
```

7. On z/OS, run the following commands on queue manager CSQ4:

```
DEFINE CHANNEL(CLUSTERB.QMSLICK) CHLTYPE(CLUSSDR) CONNAME('9.42.171.120(1421)')
  CLUSTER(CLUSTERB) BATCHHB(1)
DEFINE CHANNEL(CLUSTERB.CSQ4) CHLTYPE(CLUSRCVR) CONNAME('9.12.4.34(1415)')
  CLUSTER(CLUSTERB)
```

To complete the configuration of CLUSTERB, the queue Q3 is required on the two other queue managers.

On the Linux machine, run the following command on queue manager QM3A:

```
DEFINE QL(Q3) CLUSTER(CLUSTERB) DEFBIND(NOTFIXED)
```

On z/OS, run the following command on queue manager CSQ4:

```
DEFINE QL(Q3) CLUSTER(CLUSTERB) DEFBIND(NOTFIXED)
```

Use **amqspput** (acting as AP3) on QMSLICK and QMGRAYDE to put messages to Q3. You can then check to see that they arrived on both of the queue managers, which confirms that a working cluster exists. To perform this check from a command window on either machine, run the following command (for Windows):

```
amqspput Q3 QMSLICK
```

To send messages from the Linux machine, run the same **amqspput** command, but replace QMSLICK with QMGRAYDE.

To confirm that the messages were delivered, check that the depth of the queue Q3 on the two other queue managers increased.

On the Linux machine, run the following command on queue manager QM3A:

```
DISPLAY QL(Q3) CURDEPTH
```

This command returns an output that is similar to the output that is shown in Example 21-4.

Example 21-4 Checking the CURDEPTH on Q3

```
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM3A.
```

```
1 : DISPLAY QL(Q3) CURDEPTH
AMQ8409: Display Queue details.
          QUEUE(Q3)           TYPE(QLOCAL)
          CURDEPTH(7)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

The same MQSC command can be run on z/OS.

To clear these queues, run the following command on queue managers QM3A and CSQ4:

```
CLEAR QL(Q3)
```

21.5 Manually defining specific transmission queues with a manual switch

For each channel that is using AP3 to use its own transmission queue that is named personally by the administrator, the CLCHNAME attribute on a local queue can explicitly name the channel that is to link it.

On queue managers QMSLICK and QMGRAYDE, run the following commands:

```
DEFINE QLOCAL(QUEUE.AP3.QM3A) USAGE(XMITQ) CLCHNAME(CLUSTERB.QM3A)  
DEFINE QLOCAL(QUEUE.AP3.CSQ4) USAGE(XMITQ) CLCHNAME(CLUSTERB.CSQ4)
```

In this instance, the command **runswch1** is used to update the transmission queue for each channel. To allow the **runswch1** command to move any messages for the channel (from the old transmission queue to the new transmission queue) the channels must be stopped.

On the Windows machine, run the following commands on queue manager QMSLICK:

```
STOP CHANNEL(CLUSTERB.QM3A)  
STOP CHANNEL(CLUSTERB.CSQ4)
```

On the Linux machine, run the following commands on queue manager QMGRAYDE:

```
STOP CHANNEL(CLUSTERB.QM3A)  
STOP CHANNEL(CLUSTERB.CSQ4)
```

Use the **amqspput** command as before to put several messages to the queue Q3 from either queue manager. The messages continue to be stored on the old transmission queue because there are no running channels to put the messages across.

You can use the **runswch1** command to query if there is a pending change of transmission queue for a channel and if any messages must be moved. To perform this query, the queue manager QMSLICK is used and the command uses the **-q** parameter.

In a command prompt on the Windows machine, run the following command:

```
runswch1 -m QMSLICK -c CLUSTERB.QM3A -q
```

Sample output from this command is shown in Example 21-5.

Example 21-5 The use of RUNSWCHL to query a transmission queue change

```
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.  
Switch to transmission queue 'QUEUE.AP3.QM3A' is pending  
for channel 'CLUSTERB.QM3A'. The current transmission queue  
'SYSTEM.CLUSTER.TRANSMIT.CLUSTERB.QM3A' has 5 messages queued for the channel.
```

The existing transmission queue is not the default queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE, because previously in this scenario, the queue manager was changed to use dynamically created transmission queues for new and restarted channels.

The output from the **runswch1** command also shows that there are five messages on the old transmission queue that can be moved to the new transmission queue. To complete this change, run the same command again without the **-q** parameter, as shown in the following example:

```
runswch1 -m QMSLICK -c CLUSTERB.QM3A
```

In this instance, the output reports the switch occurred, as shown in Example 21-6.

Example 21-6 The use of RUNSWCHL to manually move messages

```
5724-H72 (C) Copyright IBM Corp. 1994, 2011. ALL RIGHTS RESERVED.  
Message move complete - 5 messages moved.  
Cluster sender channel 'CLUSTERB.QM3A' successfully switched to use XMITQ  
'QUEUE.AP3.QM3A'.
```

Now you can continue to manually move the messages for each channel or start all the channels. In this scenario, the channels are started.

On the Windows machine, run the following commands on queue manager QMSLICK:

```
START CHANNEL(CLUSTERB.QM3A)  
START CHANNEL(CLUSTERB.CSQ4)
```

On the Linux machine, run the following commands on queue manager QMGRAYDE:

```
START CHANNEL(CLUSTERB.QM3A)  
START CHANNEL(CLUSTERB.CSQ4)
```

Displaying the channel status again, the new transmission queue that is used by the two channels can be viewed. Enter the command that is shown in Example 21-7 into a **runmqsc** window for either queue manager to get the output shown.

Example 21-7 Displaying the channel status of cluster sender channels

```
DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
 1 : DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QMGRAYDE)          CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.245(1431))        CURRENT
  RQMNAME(QMGRAYDE)                  STATUS(RUNNING)
  SUBSTATE(MQGET)                   XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QM1A)            CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1422))      CURRENT
  RQMNAME(QM1A)                    STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(SYSTEM.CLUSTER.TRANSMIT.CLUSTERA.QM1A)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QM2A)            CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1423))      CURRENT
  RQMNAME(QM2A)                    STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(QUEUE.AP2)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERA.QM2B)            CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.120(1424))      CURRENT
  RQMNAME(QM2B)                    STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(QUEUE.AP2)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERB.CSQ4)           CHLTYPE(CLUSSDR)
  CONNAME(9.12.4.34(1415))         CURRENT
  RQMNAME(CSQ4)                   STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(QUEUE.AP3.CSQ4)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERB.QMGRAYDE)       CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.245(1431))     CURRENT
  RQMNAME(QMGRAYDE)               STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(SYSTEM.CLUSTER.TRANSMIT.CLUSTERB.QMGRAYDE)

AMQ8417: Display Channel Status details.
  CHANNEL(CLUSTERB.QM3A)           CHLTYPE(CLUSSDR)
  CONNAME(9.42.171.245(1432))     CURRENT
  RQMNAME(QM3A)                   STATUS(RUNNING)
  SUBSTATE(MQGET)                 XMITQ(QUEUE.AP3.QM3A)
```

The XMITQ attribute for the channels that are called CLUSTERB.QM3A and CLUSTERB.CSQ4 shows the new transmission queue that is used.

The scenario is complete. It would be worth trying the different options of multiple cluster transmission queue configuration with the knowledge you learned.



Shared queues: Using the new capabilities

This chapter describes the results of a number of test scenarios, where the new shared message data set and offload rules are used and compared. There is no direct comparison of costs between DB2 and SMDS message body offload (that topic is covered in SupportPac MP1H).

The information in this book should not be used as performance data. This testing was not done in a benchmark environment. The testing methods and information are intended to be patterns. These patterns can be used in any environment to determine how shared message data sets behave in that environment. For example, if most messages that are destined for a shared queue are 3.2 K, test series that compare different offload rules that use the 3.2-K message size should be performed.

This chapter contains the following sections:

- ▶ General scenario methodology
- ▶ No offloading shared queue capacity test
- ▶ SMDS offload: Using the default offload rules
- ▶ SMDS offload: Large SMDS data sets
- ▶ SMDS offload: Effects of offload rule changes
- ▶ SMDS offload: Effects of 100% message body offload
- ▶ SMDS offload: Mixed message sizes
- ▶ DB2 to SMDS offload migration
- ▶ SMDS offload: Message body availability

22.1 General scenario methodology

These scenarios focus on the use and impact of the shared message data sets. In creating these tests, other factors that contribute to maximizing the storage usage of the coupling facility structure were included. For example, the first scenario does not include offloading the message bodies. This test is to document the effect of letting the system dynamically adjust the structure size and the entry to element ration. This ability is sometimes referred to as ALLOWAUTOALT, the name of the coupling facility attribute.

Many of the scenarios are unnatural. In nearly all of the tests, there are no applications that are getting messages. In some tests, limited resource pools are used to show the effects.

The following scenarios are included:

- ▶ No offload of message bodies is possible. Messages are put from one queue manager only.
- ▶ Switched Multi-megabit Data Service (SMDS) offload by using the default offload rules to a small data set. Messages are put from one queue manager only.
- ▶ SMDS offload by using the default offload rules to a much larger data set. Messages are put from two queue managers.
- ▶ Offload rules are set to offload all of the message body when the coupling facility structure reaches 50% full. Messages are put from two queue managers.
- ▶ Offload rules are set to offload all of the message body 100% of the time. Messages are put from two queue managers.
- ▶ Offloading rules are left at 100% of the message bodies and the messages that are put are of mixed message sizes. Messages are put from two queue managers.
- ▶ DB2 to SMDS offload migration, in which there are message bodies on DB2 at the beginning of the test.
- ▶ SMDS offload of messages, test message availability when the owning queue manager becomes unavailable. Messages are put from one queue manager and retrieved from two others.

For many of the scenarios, multiple tests are run for each value in the range of the following message sizes:

- ▶ 1 K
- ▶ 4 K
- ▶ 8 K
- ▶ 16 K
- ▶ 32 K
- ▶ 64 K

The sample programs from SupportPac IP13, OEMPUTX and MGET, were used. SupportPac is available at this website:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24006892&loc=en_US&cs=utf-8&ang=en

The tests were run by using the following process:

1. The coupling facility structure or structures were rebuilt at the start of every series. This rebuild was done to show the benefits that can be achieved by using ALLOWAUTOALT (YES) on the coupling facility (CF) policy definition, and to make sure all of the test series were started from a level playing field. Example 22-1 shows the command that was used.

Example 22-1 Structure rebuild command example

```
setxcf start,rebuild,strname=ibm1nooff
```

2. Each test was run and the results were captured. The pertinent information about the test results were captured and reported in the individual scenario. In Appendix B, “WebSphere MQ for z/OS 7.1 System Management Facility changes” on page 451, the sample outputs from each series are included.
3. All tests were run with the expectation of filling the available storage, whether the shared message data set or the coupling facility. Both conditions return the same reason code, 2192, to indicate that more storage is unavailable.
4. All queues that were used were cleared following each test. The sample CSQUTIL job is shown in Example 22-2.

Example 22-2 CSQUTIL job to clear the shared queues

```
//CSQ1NSCQ JOB (999,POK),'MQ 7.1.0',NOTIFY=&SYSUID,CLASS=A,  
// MSGCLASS=T,MSGLEVEL=(1,1),REGION=OM,TIME=NOLIMIT  
/*JOBPARM L=9999,SYSAFF=SC61  
/*  
/*  
//DEFQS EXEC PGM=CSQUTIL,PARM='CSQ1'  
//STEPLIB DD DISP=SHR,DSN=MQ710.SCSQANLE  
//          DD DISP=SHR,DSN=MQ710.SCSQAUTH  
//          DD DISP=SHR,DSN=MQ710.SCSQLOAD  
//SYSPRINT DD SYSOUT=*  
//SYSIN   DD *  
        COMMAND DDNAME(CMDINP)  
//CMDINP  DD *  
CLEAR QLOCAL(N00FF.01K) QSGDISP(SHARED)  
/*  
//
```

5. Coupling facility information was captured before and after each test. In some circumstances, the information was captured after each test in the series.

22.2 No offloading shared queue capacity test

In this series of tests, the message capacity of a coupling facility structure was tested when there was no message body offload. For this scenario, rebuilding the CF structure was done between the message size changes to demonstrate the capacity changes that can be achieved by allowing the system to tune the coupling facility structure, ALLOWAUTOALT(YES). This information can be useful in planning the deployment of queues onto CF structures, when they are used with the information from SupportPacs MP1H and MP16.

The coupling facility structure was defined to WebSphere MQ with a CFLEVEL of 3, so no message body offloading is possible.

No offloading scenario: 1-K message tests

Seven tests were run in this series; tests one through six showed a continued increase in the number of messages that could be put on the queue before the OEMPUTX program terminated because of the 2192 storage medium full reason code. The final test, test seven, did not show any improvement.

Before the test, the coupling facility structure included the characteristics that are shown in Example 22-3.

Example 22-3 Coupling facility NOOFF before tests

```
IXC360I 08.39.41 DISPLAY XCF 659
STRNAME: IBM1NOOFF
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE    : 512 M
  POLICY INITSIZE: 272144 K
  POLICY MINSIZE : 204108 K
  FULLTHRESHOLD : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX         : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF02      CF03      CF04
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY
ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/11/2012 08:39:08
CFNAME        : CF02
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
                  PARTITION: 0F CPCID: 00
ACTUAL SIZE    : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL     CHANGED %
ENTRIES:       152836      375      0
ELEMENTS:      689615      802      0
EMCS:          94584       36       0
LOCKS:          1024
ACTUAL SUBNOTIFYDELAY: 5000
```

```

PHYSICAL VERSION: CA4D825C 13534231
LOGICAL VERSION: CA4230ED 020FC020
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME : IXCL0058
DISPOSITION : KEEP
ACCESS TIME : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS : 3

CONNECTION NAME ID VERSION SYSNAME JOBNAME ASID STATE
-----
CSQEIBM1CSQ101 01 00010006 SC61 CSQ1MSTR 008F ACTIVE
CSQEIBM1CSQ202 03 00030002 SC62 CSQ2MSTR 0064 ACTIVE
CSQEIBM1CSQ303 02 00020002 SC61 CSQ3MSTR 0090 ACTIVE

```

OEMPUTX from SupportPac IP13 was used to put the messages. The job stream looked as shown in Example 22-4

Example 22-4 OEMPUTX sample for the 1K message tests

```

//*****
//*
//** TEST 1 K MESSAGES WITH NO OFFLOADS *
//*
//*****
// SET APPLOAD='CSQ1V71.IP13.LOAD'
// SET MQLOAD1='MQ710.SCSQLOAD'
// SET MQLOAD2='MQ710.SCSQAUTH'
// SET MQLOAD3='MQ710.SCSQLOAD'
// SET THISPDS='CSQ1V71.IP13.CNTL'
// SET QM='CSQ1'
// SET SUMMARY='TEAMXX.DOCUMENT.SUMMARY'
// SET DB2='DSNA*'      DB2 NAME USED IN REPORTING CPU
// SET TEMPFILE='TEAMXX.ZZ2'
// SET P=' ' Nonpersistent messages
// SET MSGS=500000
// EXEC PGM=OEMPUTX,REGION=0M,
// PARM=( '-m&QM. -n&MSGS. -x &P. -l1 -c1 -s1024' )
//SYSIN DD *
-qNOOFF.01K      * request queue
-fileDD:MSGIN
-dJTEST4
//STEPLIB DD DISP=SHR,DSN=&APPLOAD.
//          DD DISP=SHR,DSN=&MQLOAD1.
//          DD DISP=SHR,DSN=&MQLOAD2.
//          DD DISP=SHR,DSN=&MQLOAD3.
//MSGIN DD DISP=SHR,DSN=&THISPDS.(SMALMSG)
//SYSPRINT DD DISP=SHR,DSN=CSQ1V71.SMDS.TESTS(NOF01KR7)
//
```

As each test is run, the output is captured in a PDS. The output from the OEMPUTX is shown in Example 22-5.

Example 22-5 OEMPUTX output

```

CSQ1MSTR.008F      10   100      0      0      0
-----
Total Transactions : 114914
Elapsed Time       : 9.444 seconds
Application CPU Time: 6.655 seconds (70.5%)
Transaction Rate   : 12167.329 trans/sec
-----
Round trip per msg :     82 microseconds
Avg App CPU per msg: 57 microseconds
-----
Jobname.ASID    TCB(uS)  SRB(uS)  Tot(uS) (%) /tran /tran /tran
----- /----- /----- /----- /----- /----- /----- /-----
CSQ1MSTR.008F  00000000 00000000 00000000 0.1
CSQ1CHIN.008D  00000000 00000000 00000000 0.0
CSQ1BRK*       00000000 00000000 00000000 0.0
Total CPUmicrosecs/tran          0
-----
Ending loop at 2012-10-11 12:44:38.328985
OEMPUTX Normal Exit: End of program
Exiting at 2012-10-11 12:44:38.330825

```

Return code 2192: The return code 2192 has multiple meanings. In OEMPUTX, it is always interpreted as a pageset full condition. It can also mean that the coupling facility structure is full or that shared message data set is full.

The entry to element ratio, which helps determine the capacity of the list structure, was initially at 1:4 (152836:689615). Immediately following the structure definition or rebuild, the ratio is typically at 1:6. Both the size of the structure and the entry to element ratio were adjusted over time. The messages were of a consistent size, so the ratio did not vary after the first adjustment.

In this test, the number of messages that were successfully put expanded from 114,914 to 240,805. The size doubled from 266 M to 512 M, and the message capacity increased slightly more. Table 22-1 shows the test results for the no offloading scenario, 1 K message tests.

Table 22-1 Test results, no offload 1 K messages

Test Number	Total Messages Put	Number of Entries	Number of Elements	Ratio
1	114914	152836	689615	1:4
2	148433	148582	890731	1:5
3	182332	182408	1094125	1:5
4	201825	201961	1211080	1:5
5	223325	223729	1340078	1:5
6	240805	241096	1444961	1:5
7	240805	241096	1444961	1:5

No offloading scenario: 4 K message tests

For the 4 K messages, the entry-to-element ratio started at 1:5. The structure rebuild that was performed between the test series used the last value from the previous runs. The ratio was automatically altered to 1:17. The structure expanded from the 266 M to 512 M, as expected over the test series. The message count went from 40,945 to 91,736, or an increase of 2.25 times the original message count. Table 22-2 shows the test results for the no offloading scenario, 4 K message tests.

Table 22-2 Test results, no offload 4 K messages

Test Number	Total Messages Put	Number of Entries	Number of Elements	Ratio
1	40945	122871	737147	1:5
2	51255	51316	922727	1:17
3	56717	51316	922727	1:17
4	69536	63020	1133191	1:17
5	76947	69853	1251791	1:17
6	85112	77155	1385175	1:17
7	91736	91954	1651388	1:17
8	91736	91954	1651388	1:17

No offloading scenario: 8 K message tests

For the 8 K messages, the entry to element ratio started at 1:5. The ratio was automatically altered to 1:33. The structure expanded from the 266 M to 512 M, as expected over the test series. The message count went from 21,484 to 48,871, or an increase of 11.65 times the original message count. For this size message, the more efficient storage use that was provided by the ratio change enhanced the number of messages that could be held in the structure. Table 22-3 shows the test results for the no offloading scenario, 8 K message tests.

Table 22-3 Test results, no offload 8 K messages

Test Number	Total Messages Put	Number of Entries	Number of Elements	Ratio
1	21484	119954	719445	1:5
2	27286	27664	927881	1:33
3	30188	30616	102654	1:33
4	37036	41312	1393505	1:33
5	45344	49080	1661773	1:33
6	48871	49080	1661773	1:33
7	48871	49080	1661773	1:33

No offloading scenario: 16 K message tests

For the 16 K messages, the entry to element ratio started at 1:33, as it was copied from the last series of tests during the structure rebuild. The ratio was automatically altered to 1:64. As seen in the previous tests, the structure expanded from 266 M to 512 M, as expected over the test series. The message count went from 12,938 to 25,834, or an increase of 1.91 times the original message count. Table 22-4 shows the test results for the no offloading scenario, 16 K message tests.

Table 22-4 Test results, no offload 16 K messages

Test Number	Total Messages Put	Number of Entries	Number of Elements	Ratio
1	12938	25341	854066	1:33
2	14490	14803	956510	1:64
3	17745	18071	1171340	1:64
4	21708	20127	1296382	1:64
5	23929	21963	1432866	1:65
6	25834	25994	1705173	1:65
7	25834	25994	1705173	1:65

No offloading scenario: 32 K message tests

For the 32 K messages, the entry to element ratio started at 1:63, as it did not yet adjust down completely from the previous series of test. The ratio was automatically altered to 1:125. As seen in the previous tests, the structure expanded from 266 M to 512 M, as expected over the test series. The message count went from 6,666 to 13,217, or an increase of 1.98 times the original message count. Table 22-5 shows the test results for the no offloading scenario, 32 K message tests.

Table 22-5 Test results, no offload 32 K messages

Test Number	Total Messages Put	Number of Entries	Number of Elements	Ratio
1	6666	13580	866733	1:63
2	8221	7779	964138	1:123
3	10052	8498	1068979	1:125
4	11108	10501	1306914	1:124
5	12249	11453	1444282	1:126
6	13217	13679	1718394	1:125
7	13217	13679	1718394	1:125

No offloading scenario: 64 K message tests

As expected, this test failed because the messages were too large to fit into the CF space that was available. The result of the test run can be seen in Example 22-6.

Example 22-6 No offload allowed - messages too large for coupling facility

```
Compiled Sep 13 2006 11:23:57.
buffer:-qN0OFF.64K      * request queue
buffer:-fileDD:MSGIN
buffer:-dJTEST4
parm: -mCSQ1 -n500000 -x -l1 -c1 -s65536
Message file: DD:MSGIN
OEMPUTX about to MQCONN to QMgr CSQ1.
OEMPUTX about to MQOPEN request queue: N0OFF.64K
CPU type 3A3BD52817
Date Time 2012/10/15 15:12:22.
Description JTEST4.
Entering PUT only loops...
Preload the queue with 0 messages...
  Message size      : 65536
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages       : 500000
  Syncpoints          : NO-SYNCPOINT
Starting loop at 2012-10-15 15:12:22.397849
OEMPUTX: MQPUT failed cc=2, rc=2366  MQRC_WRONG_CF_LEVEL
<<< MESSAGE PUT TO REQUEST QUEUE ** 0 ** >>>
MQMD header size is 324 bytes
StrucId      :MD Version      :1
Report        :00000000MsgType      :00000008
Expiry        :FFFFFFFFFeedback    :00000000
Encoding       :00000311CodedCharSetId :00000000 (0)
Format        :MQSTR
Priority       :FFFFFFFPPersistence  :00000000
MsgId         :.....
              <00000000000000000000000000>
              <00000000000000000000000000>
CorrelId       :.....
              <00000000000000000000000000>
              <00000000000000000000000000>
BackoutCount   :00000000
ReplyToQ       :
ReplyToQMgr    :
UserIdentifier :
AccountingToken
              <0000000000000000000000000000000000000000000000000000000000000000>
ApplIdentityData:
PutApplType    :00000000
PutApplName    :
PutDate        :PutTime      :
ApplOriginData :
** Message size 65536

OEMPUTX: MQGET failed cc=2, rc=2366  MQRC_WRONG_CF_LEVEL
Workload manager data
```

```

          Samples %idle %unknown(MQ?) %using CPU %doing I/O %Wait for CPU
CSQ1CHIN.0022      15    100           0       0       0       0
CSQ1MSTR.0089      15    100           0       0       0       0
-----
Total Transactions : 0
Elapsed Time       : 0.000 seconds
Application CPU Time: 0.000 seconds (84.8%)
Transaction Rate   : 0.000 trans/sec
-----
Round trip per msg : 387 microseconds
Avg App CPU per msg : 0 microseconds
-----
Jobname.ASID TCB(uS) SRB(uS) Tot(uS) (%)
/tran /tran /tran
-----
CSQ1MSTR.0089 00000000 00000000 00000000 0.0
CSQ1CHIN.0022 00000000 00000000 00000000 0.0
CSQ1BRK*       00000000 00000000 00000000 0.0
Total CPUmicrosecs/tran           0
-----
Ending loop at 2012-10-15 15:12:22.398753
OEMPUTX Normal Exit: End of program
Exiting at 2012-10-15 15:12:22.401072

```

No offload test conclusions

The use of the ALLOWAUTOALT capability on the coupling facility policy definition was recommended for many years to let the system more efficiently use the storage available. In these tests, the structure was also allowed to expand. However, as you can see from the series, the expansion takes much longer than the entry-to-element ratio reconfiguration. For some message sizes, the ratio change is beneficial.

If the message sizes are constant (or fairly constant), the use of ALLOWAUTOALT(YES) can improve the storage use within the structure, and from an WebSphere MQ perspective, give greater capacity.

22.3 SMDS offload: Using the default offload rules

In this series of tests, the message capacity of a coupling facility structure was tested by using a small, shared message data set and the default offload rules as shown in Figure 22-2 on page 344.

Rebuilding the CF structure is done between the message size changes to keep the tests consistent.

The CF structure is defined to WebSphere MQ with a CFLEVEL of 5 and the SMDS is defined at 100 cylinders. This structure is a small SMDS data set. In a production environment, this structure typically is much larger. This size was used to demonstrate effects that might be observed in a test environment on which resources are often less available.

SMDS offloading scenario: 1 K message tests

There were seven tests run in this series; tests one through six showed a continued increase in the number of messages that could be put on the queue before the OEMPUTX program terminated because of the 2192 storage medium full reason code. The final test, test seven, did not show any improvement.

Before the test, the coupling facility structure included the characteristics that are shown in Example 22-7.

Example 22-7 Coupling facility SMDOFF before tests

```
STRNAME: IBM1SMDOFF
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE      : 512 M
  POLICY INITSIZE: 272144 K
  POLICY MINSIZE : 204108 K
  FULLTHRESHOLD  : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX          : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF03      CF02      CF04
ENFORCEORDER    : NO
EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/16/2012 09:04:08
CFNAME         : CF04
COUPLING FACILITY: 002097.IBM.02.00000001DE50
                  PARTITION: 0D CPCID: 00
ACTUAL SIZE     : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL      CHANGED %
ENTRIES:        122865      39      0
ELEMENTS:       737154      144      0
EMCS:           94584       42      0
LOCKS:          1024
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CA53D145 42E03AB1
LOGICAL VERSION: CA38844B 24F7FD2A
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME    : IXCL0057
DISPOSITION     : KEEP
ACCESS TIME     : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS   : 3

CONNECTION NAME ID VERSION SYSNAME  JOBNAME ASID STATE
-----
CSQEIBM1CSQ101 01 00010010 SC61      CSQ1MSTR 0089 ACTIVE
CSQEIBM1CSQ202 02 00020005 SC62      CSQ2MSTR 0060 ACTIVE
```

The WebSphere MQ CFSTRUCT definition is shown in Figure 22-1 and Figure 22-2 on page 344.

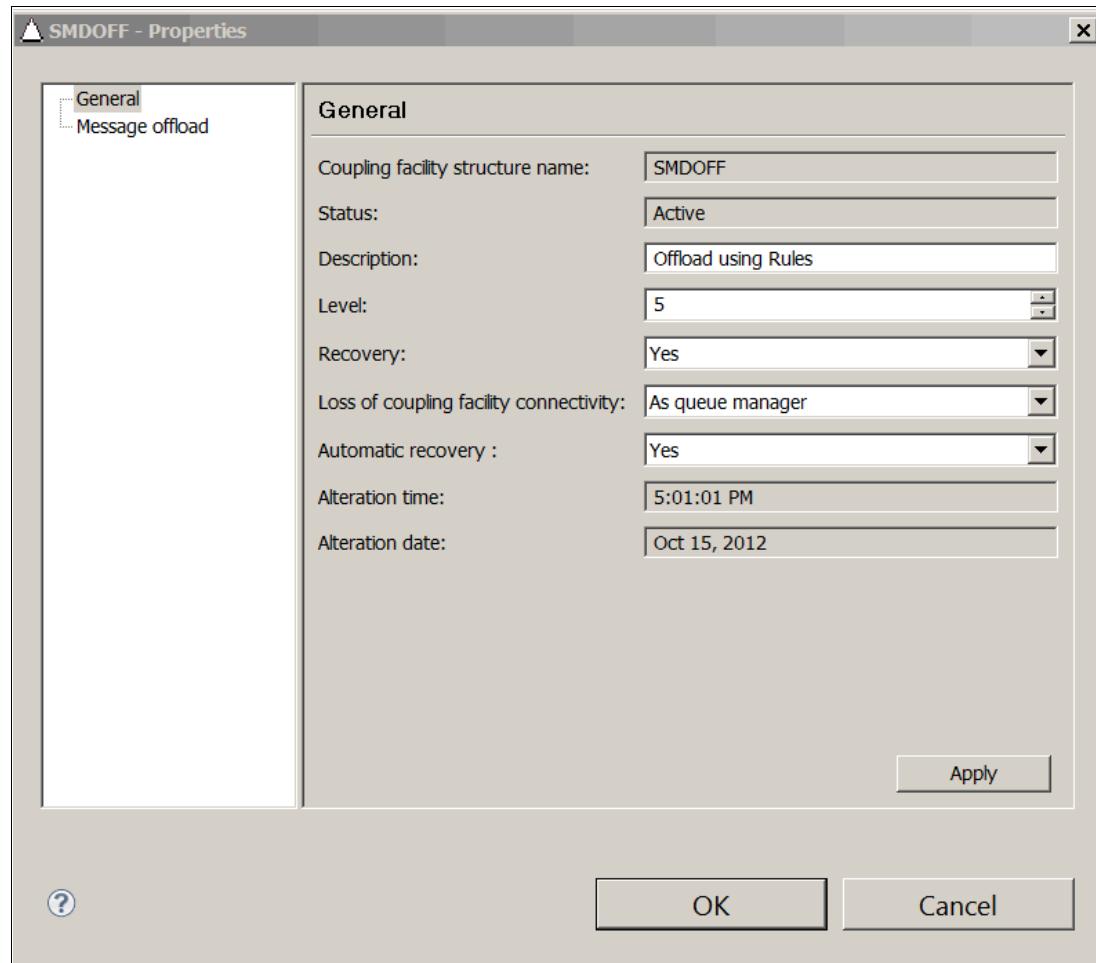


Figure 22-1 SMDOFF CFSTRUCT definition first panel

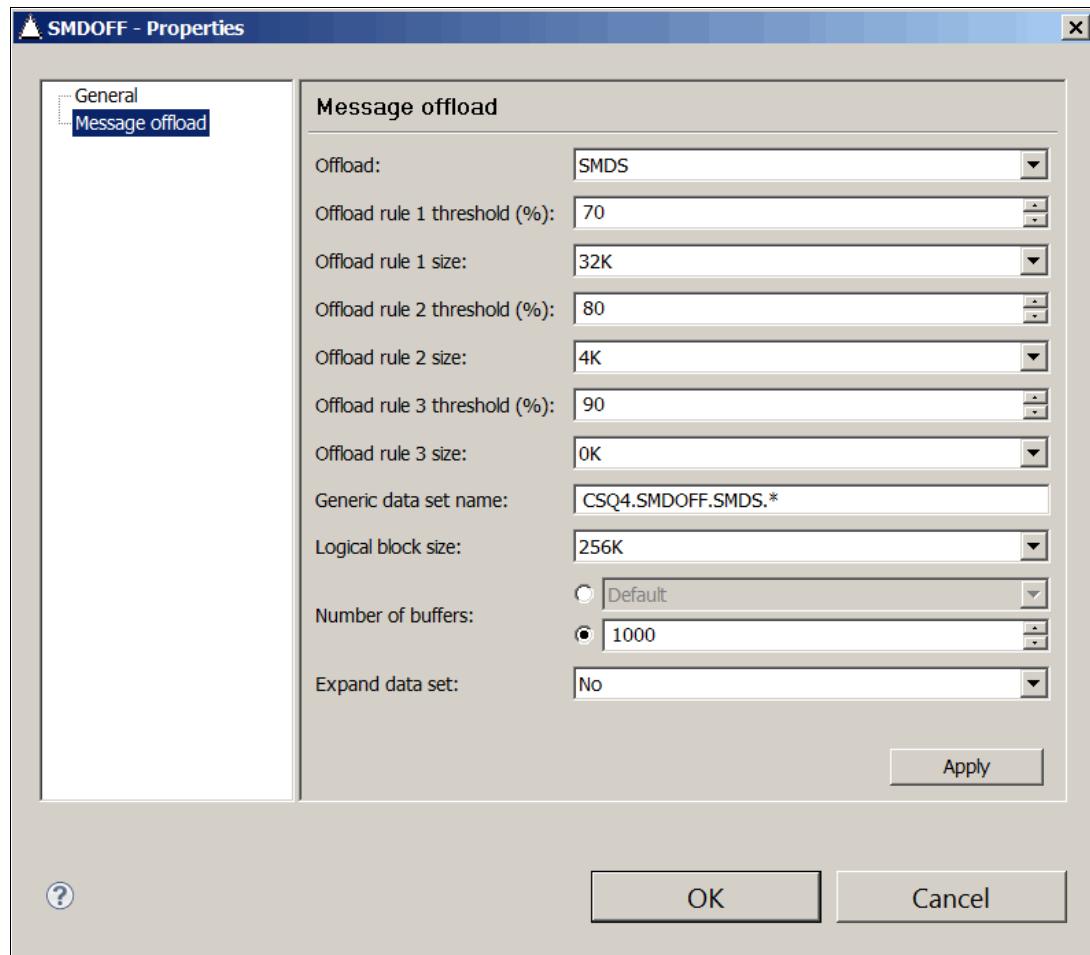


Figure 22-2 SMDOFF CFSTRUCT definition second panel

The cluster definition that was used for the shared message data set is shown in Example 22-8.

Example 22-8 SMDS VSAM definition

```
DEFINE CLUSTER
  (NAME(CSQ4.SMDOFF.SMDS.CSQ1)
   CYLINDER(100 5)
   LINEAR
   VOLUME(TOTMQA)
   SHAREOPTIONS(2 3) )
DATA
  (NAME(CSQ4.SMDOFF.SMDS.CSQ1.DATA) )
```

In the first test series (the 1 K message size), the results showed the increases in capacity from message offloading, the coupling facility size increased, and a minor change to the entry to element ratio adjustments. The SMDS data set also filled.

The shared message data set usage was displayed after the queue was filled and the number of messages that were offloaded is also reported in the results, as shown in Table 22-6. The number of messages that were offloaded reached the maximum in the second test run (17,596), but the CF structure continued to expand until it reached the limit of 516 M. As was observed in the no offload tests, the entry to element ratio did not alter after the initial test because the messages were all the same size. The number of entries and elements continued to expand. As the ratio generally remained constant after the first test, that information is not in the table.

Table 22-6 Test results, SMDS offload 1K messages, default rule

Test Number	Total Messages Put	Offloaded Messages	Number of Entries	Number of Elements
1	126202	17814	129101	722744
2	150273	17856	178016	982339
3	165183	17856	195378	1087227
4	180917	17856	214985	1204128
5	198452	17856	236413	1333768
6	217898	17856	253918	1438211
7	233564	17856	253918	1438211
8	233564	17856	253918	1438211

As expected, the structure is expanded in size and internal configuration, as shown in Example 22-9. Only the size information is shown.

Example 22-9 Coupling facility SMDOFF size and configuration that followed the test series

```

ACTUAL SIZE      : 512 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL     CHANGED   %
ENTRIES:        253918    233603   91
ELEMENTS:       1438211   1330104  92
EMCS:           189168      42      0
LOCKS:          1024

```

SMDS offloading scenario: 4 K message tests

This series showed a surprising result at first glance. The three tests run gave the same results, a total of 41,115 messages were put to the queue, with a consistent 8,928 messages that were offloaded. One of the advantages of the use of message body offloading that was observed is the ability to have deeper queues. The result seems to refute that claim, until the parameters of the system coupling facility structure definition and the WebSphere MQ CFSTRUCT definition are compared, and the messages that were issued by WebSphere MQ and XCF were examined. Table 22-7 shows the test results.

Table 22-7 Test results, SMDS offload 4 K messages, default rules

Test Number	Total Messages Put	Offloaded Messages	Number of Entries	Number of Elements
1	41115	8928	129101	722744
2	41115	8928	127480	724365
3	41115	8928	127480	724365

In all previous tests, the no offload, 1 K message size SMDS offloads, and the JES log contained a number autoalter messages, as shown in Example 22-10.

Example 22-10 Autoalter messages example.

```
IXC530I START ALTER REQUEST FOR STRUCTURE IBM1SMDOFF ACCEPTED.  
IXC590I AUTOMATIC ALTER PROCESSING FOR STRUCTURE IBM1SMDOFF 80  
COMPLETED. TARGET ATTAINED.  
CURRENT SIZE:          512 M   TARGET:      512 M  
CURRENT ENTRY COUNT:  254830    TARGET:  254830  
CURRENT ELEMENT COUNT: 1449989   TARGET: 1449989  
CURRENT EMC COUNT:    189168    TARGET: 189168
```

The decision to automatically alter a structure size, or the allocation of the structure contents (entries, elements, event queue controls, and event monitor controls) is made by the sysplex, which is based on the policy definition and the status of the sysplex.

During the first test run in this series, there was some adjustment of the structure and one instance of the IXC messages. The 4 K tests had no other alter activity on the structure. The reason for this difference in behavior was simple. The autoalter full threshold on the structure is set to 85% while the offload threshold that initiates the offloads is set at 80% and offloads 4 K messages. Messages are offloaded to the capacity of the shared message data set, which was reached at the 8,928 messages mark. Because there was no program that was pulling messages from the queue, the automatic structure expansion never had an opportunity to start.

During the tests, the SMDS filled and messages shown in Example 22-11 were displayed.

Example 22-11 Shared message data set full messages

```
CSQE213I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 899
data set CSQ4.SMDOFF.SMDS.CSQ1 is now 90% full
CSQE213I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 900
data set CSQ4.SMDOFF.SMDS.CSQ1 is now 92% full
CSQE213I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 901
data set CSQ4.SMDOFF.SMDS.CSQ1 is now 94% full
CSQE213I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 902
data set CSQ4.SMDOFF.SMDS.CSQ1 is now 96% full
CSQE213I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 903
data set CSQ4.SMDOFF.SMDS.CSQ1 is now 98% full
CSQE239I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 904
data set CSQ4.SMDOFF.SMDS.CSQ1 has become full so new large messages
can no longer be stored in it
```

SMDS offloading scenario: 8 K message tests

As with the other test, the 8 K message tests showed similar results to the 4 K tests. The total number of messages that could be put is 23,132, with 5,952 messages offloaded, as shown in Table 22-7.

Table 22-7 Test results, SMDS offload 8K messages, default rules

Test Number	Total Messages Put	Offloaded Messages
1	23132	5952
2	23132	5952
3	23132	5952

The size of the structure and the ratios were not altered, as shown in the before and after coupling facility display results in Example 22-12.

Example 22-12 Coupling facility display before and after 8K test

Before the tests (just after structure rebuild):

```
ACTUAL SIZE      : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL    CHANGED   %
ENTRIES:        128339     39       0
ELEMENTS:       730323     144      0
EMCS:           94584      42       0
LOCKS:          1024
```

After the tests:

```
ACTUAL SIZE      : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL    CHANGED   %
ENTRIES:        128339     23171    18
ELEMENTS:       730323     596168   81
EMCS:           94584      42       0
LOCKS:          1024
```

SMDS offloading scenario: 16 K, 32 K, and 64 K message tests

As with the 4 K and 8 K message size tests, the results from the remaining message sizes show that the small size of the shared message data set affected the number of messages that could be put on the queue. After messages are directed to the SMDS, it fills before the CF structure is resized or reconfigured.

Table 22-8 shows the total depth and offloaded messages.

Table 22-8 Test results, SMDS offload 16 K, 32 K, and 64 K messages, default rules

Message size	Depth	Offload
16K	12350	3571
32K	5826	1970
64K	1035	1035

SMDS offloading scenario conclusions

Message body offloading can provide deeper queue depths, with some caveats. As captured, the benefits are based on the message length and the size of the shared message data set. When message sizes are small, a small SMDS did provide more queue capacity.

Although no one expects such a small SMDS in production (especially when message sizes might not be as predictable as a controlled test environment), the effects that were captured in this series shows that under some circumstances, the addition of an SMDS for offloading negatively impacts the maximum number of messages that can be stored.

22.4 SMDS offload: Large SMDS data sets

In this scenario, a new shared message data set is allocated for each queue manager with a dramatic size increase. The initial allocation is 5,000 cylinders (up from 100) and the data sets can expand. The expectation is that during the series of tests, the volume of messages should increase based on the expansion that is allowed in the coupling facility and the shared message data set.

In this series of tests, messages are put from applications that are connected to two of the three queue managers in the queue sharing group: CSQ1 and CSQ2. CSQ1 is on logical partition (LPAR) SC61 and CSQ2 is on SC62.

The allocation job stream for the CSQ3 queue manager is shown in Example 22-13.

Example 22-13 Shared message data set allocation

```
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
      DELETE 'CSQ4.SMDOFF.SMDS2.CSQ3' ERASE CLUSTER
      SET MAXCC=0

      DEFINE CLUSTER          -
        (NAME(CSQ4.SMDOFF.SMDS2.CSQ3)  -
         CYLINDER(5000 25)           -
         LINEAR                      -
         VOLUME(TOTMQT TOTMQU)       -
         SHAREOPTIONS(2 3)           -
         DATA                         -
        (NAME(CSQ4.SMDOFF.SMDS2.CSQ3.DATA) )

/*

```

Because the SMDOFF CFSTRUCT was active previously, the change to the new data sets was implemented by using the following process:

1. All queues that use the SMDOFF structure were cleared and deleted.
2. The structure was deleted by using the following delete command;
-csq1 delete CFSTRUCT(SMDOFF)
3. The structure was redefined by using the following command:
**-csq1 define CFSTRUCT(SMDOFF) cflevel(5)
offload(SMDS) DSGROUP('CSQ4.SMDOFF.SMDS2.*')**

Large shared message data set offloading scenario: 1 K messages

There were seven tests run in this series. Tests one through five showed a continued increase in the number of messages that could be put on the queue before the OEMPUTX program terminated because of the 2192 storage medium full reason code. The final tests, six and seven, did not show any change in the volume.

One observation that was made during these tests is that although there were two active jobs putting messages to each queue during the tests, the WebSphere MQ Explorer shows only one connection to the queue from any of the WebSphere MQ Content panels. This fact was true whether the queue is observed from the individual queue managers or from the queue sharing group perspective, as shown in Figure 22-3.

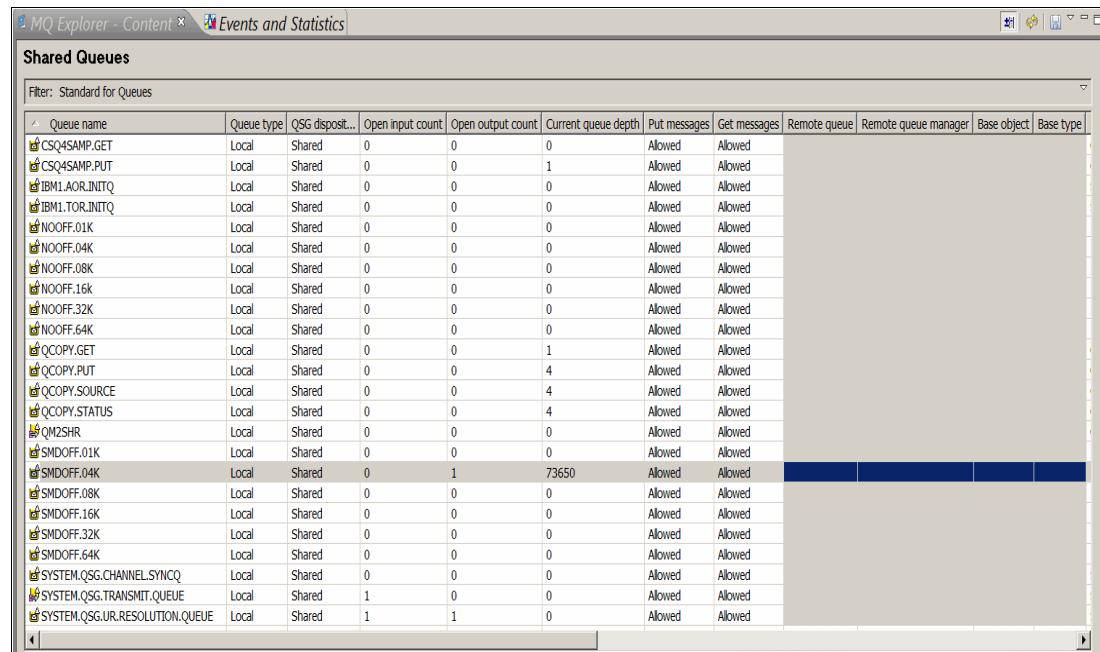


Figure 22-3 Output process count on a shared queue

The number of messages successfully put from the two queue managers are captured in Table 22-9. The increase in the number of messages between test two and three, and between four and five coincided with the structure and the shared message data set expanding successfully.

Table 22-9 Large SMDS 1K message test results

Test Number	Total Messages	Messages Put from CSQ1	CSQ1 Offloaded messages	Messages Put from CSQ2	CSQ2 Offloaded messages
1	148002	74998	18203	73004	18268
2	148002	74872	18224	73130	18246
3	179398	90696	21133	88702	21137
4	196378	99905	23179	99182	23311
5	199087	100234	23185	98853	23305
6	199087	99782	23202	99305	23288
7	199087	99945	23152	99142	23338

Large shared message data set offloading scenario: 4 K messages

There were eight tests run in this series and the pattern was not a continued upward curve as was seen in some of the other tests. There was substantial adjustment in the coupling facility structure size over the test runs. However, allowing the system to adjust the structure size and entry to element ratio can also result in capacity reduction, especially when there are competing workloads. In this test series and with others in this category, there are some examples of the system recapturing storage temporarily. The CF structure information captured before the tests and after all were completed is shown in Example 22-14.

Example 22-14 SMDOFF information

SMDOFF Display before the 4K tests:

```
ACTUAL SIZE      : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL      CHANGED    %
ENTRIES:        157812     34         0
ELEMENTS:       727498     64         0
EMCS:           7882       12         0
LOCKS:          1024
```

SMDOFF Display After the 4K tests:

```
ACTUAL SIZE      : 512 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL      CHANGED    %
ENTRIES:        163085     163085    100
ELEMENTS:       1642253    1493958   90
EMCS:           7882       12         0
LOCKS:          1024
```

This test series clearly show that capacity predictions can be more difficult when multiple storage repositories are used.

Message groups from the SMDS expansion also showed up many times during the test series. Although all were successful, the messages were arriving so rapidly that the allocation and formatting could not keep pace. A larger secondary allocation helps smooth out this behavior. Some of the messages from the CSQ1 log are shown in Example 22-15.

Example 22-15 Shared message data set expansion and format messages

```
CSQE213I -CSQ1 CSQEDSS2 SMDS(CSQ1) CFSTRUCT(SMDOFF) 206
Data set CSQ4.SMDOFF.SMDS2.CSQ1 is now 90% full
CSQE211I -CSQ1 CSQEDSI1 Formatting is in progress for 207
4500 pages in SMDS(CSQ1) CFSTRUCT(SMDOFF) data set
CSQ4.SMDOFF.SMDS2.CSQ1
CSQE212I -CSQ1 CSQEDSI1 Formatting is complete for 208
SMDS(CSQ1) CFSTRUCT(SMDOFF) data set CSQ4.SMDOFF.SMDS2.CSQ1
CSQE217I -CSQ1 CSQEDSI1 Expansion of SMDS(CSQ1) 209
CFSTRUCT(SMDOFF) data set CSQ4.SMDOFF.SMDS2.CSQ1 was successful,
pages added, total pages 949500
```

The test results for the 4 K message sizes are shown in Table 22-10. After the coupling facility structure and the shared message data set reached their full size and the optimal entry to element ratio for the message size, the total number of messages remained constant.

Table 22-10 Large SMDS 4 K message test results

Test Number	Total Messages	Messages Put from CSQ1	CSQ1 Offloaded messages	Messages Put from CSQ2	CSQ2 Offloaded messages
1	105069	53312	36110	51757	36628
2	128477	62480	42879	62967	43967
3	125535	62521	40715	62814	41683
4	120551	60510	33261	60041	34100
5	150061	74913	44262	75148	45415
6	131308	74913	32145	65121	32926
7	163051	81435	44323	81616	45741
8	163051	81787	44341	81264	45723

Large shared message data set offloading scenario: 8 K messages

There were seven tests run in this series and the pattern is more expected, though there is a slight drop between test two and three. Messages from shared message data set expansion and formatting also are in both queue managers JESMSGLG output. The test results are shown in Table 22-11.

Table 22-11 Large SMDS 8K message test results

Test Number	Total Messages	Messages Put from CSQ1	CSQ1 Offloaded messages	Messages Put from CSQ2	CSQ2 Offloaded messages
1	88899	41634	34610	47265	35140
2	101273	51740	39355	49533	40776
3	94597	46666	36355	47931	37101
4	115124	63647	42950	51477	42908
5	135115	72626	50612	62489	52324
6	135115	67227	50661	67888	52275
7	135115	66200	50905	68915	52031

Although the last three tests had the same total number of messages, it is interesting that the distribution of the messages is not the same. The CF information is shown in Example 22-16. The number of entries is slightly more than the number of messages.

Example 22-16 Coupling facility SMDOFF allocation after the 8K tests

ACTUAL SIZE	:	432 M
STORAGE INCREMENT SIZE:	1 M	
USAGE INFO	TOTAL	CHANGED %
ENTRIES:	135150	135150 100
ELEMENTS:	1367651	1300038 95
EMCS:	7882	18 0
LOCKS:	1024	

Large shared message data set offloading scenario: 16 K messages

There were five tests run in this series. There was an initial drop in volume between tests one and two, then the volume stabilized. The test results are shown in Table 22-12.

Table 22-12 Large SMDS 16K message test results

Test Number	Total Messages	Messages Put from CSQ1	CSQ1 Offloaded messages	Messages Put from CSQ2	CSQ2 Offloaded messages
1	85150	41954	36950	43196	38286
2	75136	37935	32196	37201	33026
3	87722	43781	37686	43941	38966
4	87722	43926	37724	43796	38928
5	87722	43407	37676	44315	38976

Large shared message data set offloading scenario: 32 K messages

There were 12 tests runs in this series before stabilizing for two runs. This series of tests also showed the effects of storage recapture. The test results are shown in Table 22-13.

Table 22-13 Large SMDS 32K message test results

Test Number	Total Messages	Messages Put from CSQ1	CSQ1 Offloaded messages	Messages Put from CSQ2	CSQ2 Offloaded messages
1	81701	40978	38351	40723	38925
2	85667	43054	40294	42613	40948
3	104241	51964	49402	52277	50018
4	124322	61890	58811	62432	60220
5	145154	72479	68822	72675	70516
6	166547	82553	78908	83994	81244
7	175693	87436	83246	88457	84855
8	177099	87355	82924	89744	85659
9	195461	97120	92263	98341	94619
10	161615	79642	75306	81973	77730
11	182747	89883	85348	92864	88745
12	182747	90720	86377	92027	87716

Large shared message data set offloading scenario: 64 K messages

There were five tests run in this series and the results are interesting. The total number of messages that could be held stabilized quickly because there was little adjustment required by the system. These messages always required offloading of the message body because of their size, so the coupling facility only held the message header and control information and the message bodies are offloaded.

The results are shown in Table 22-14.

Table 22-14 Large SMDS 64K message test results

Test Number	Total Messages	Messages Put from CSQ1	CSQ1 Offloaded messages	Messages Put from CSQ2	CSQ2 Offloaded messages
1	100297	49973	49973	50324	50324
2	100297	49998	49998	50299	50299
3	123294	61647	61647	61647	61647
4	123294	61647	61647	61647	61647
5	123294	61647	61647	61647	61647

22.5 SMDS offload: Effects of offload rule changes

In the previous tests, the default offload rules were used to control when the shift was made between the use of the coupling facility and the shared message data set for the message body. In this test series, the offload rule was changed to offloading the entire message body after the coupling facility application structure reaches 50% full.

These tests continue to take advantage of the ALLOWAUTOALT feature.

Important: If the structure or the offload storage is full, adjusting the offload rule does not affect the ability to put messages on the shared queues until some messages are read.

Changing the rule for the SMDOFF structure is done by using WebSphere MQ Explorer, as shown in Figure 22-4.

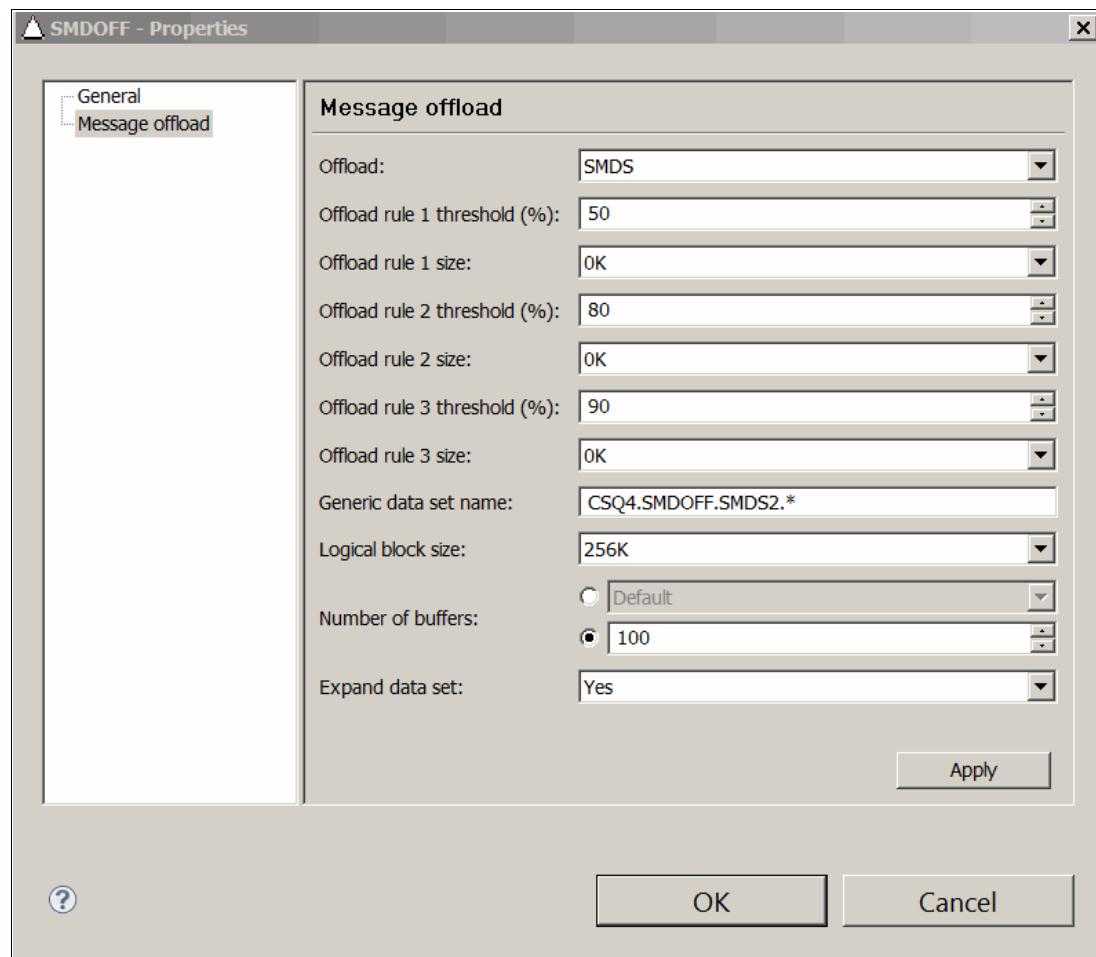


Figure 22-4 SMDOFF change in offload rule

Adjusting the offload rule to put more message bodies onto external storage has two primary benefits: it conserves the precious CF space and provides more storage capacity during a message consumer slowdown (or disappearance). This test series compares the number of messages that are stored when the 50% rule is used rather than the default graduated rule.

50% Offload rule: 1 K messages

The test results that are shown in Table 22-15 compare the total number of messages put from the graduated rule and the 50% rule, and the number of messages offloaded on each queue manager. The total number of messages put on each queue manager is not included, only to conserve space.

Table 22-15 50% Offload rule: 1 K message test results

Test Number	Total Messages Default Rule	Total Message 50% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 50% Rule	CSQ2 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 50% Rule
1	148002	217216	18203	81494	18268	81432
2	148002	241284	18224	90992	18246	89984
3	179398	267981	21133	99386	21137	101613
4	196378	329319	23179	123249	23311	123716
5	199087	310537	23185	114615	23305	113568
6	199087	359538	23202	134903	23288	132892
7	199087	359538	23152	134163	23338	133632

For messages this small, 50% offloading made a substantial difference in the maximum message depth over the default offload rule that was used for SMDS.

Finding: The maximum queue depth when the 50% message body offload was used was 1.80 times higher than the default values.

The comparison of the 50% message body off load rule with the message depths that were achieved when there was no offloading at all is shown in Table 22-16.

Table 22-16 50% Offload rule compared to no offloading: 1K messages

Test Number	Total Message 50% Rule	Total Messages No offloading
1	217216	114914
2	241284	148433
3	267981	182332
4	329319	201825
5	310537	223325
6	359538	240805
7	359538	240805

50% Offload rule: 4 K message tests

The test results compare the maximum message depth that was reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages offloaded on each queue manager. These results are shown in Table 22-17.

Table 22-17 50% Offload rule: 4 K message test results

Test Number	Total Messages Default Rule	Total Message 50% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 50% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 50% Rule
1	105069	184478	36110	82275	36628	84158
2	128477	214222	42879	94182	43967	96763
3	125535	251754	40715	113538	41683	115732
4	120551	251754	33261	113999	34100	116009
5	150061	254783	44262	116087	45415	117773
6	131308	280622	32145	124574	32926	127989
7	163051	280622	44323	124964	45741	127599

As with the 1 K message test, the 4 K message showed a substantial increase in the depth when an across the board 50% message body offload was used. The maximum queue depth when the 50% message body offload was used was 1.72 times higher than when the default values were used.

The comparison of the 50% message body offload rule with the message depths that were achieved when there was no offloading at all is shown in Table 22-18.

Table 22-18 50% Offload rule compared to no offloading: 4 K messages

Test Number	Total Message 50% Rule	Total Messages No offloading
1	184478	40945
2	214222	51255
3	251754	56717
4	251754	69536
5	254783	76947
6	280622	85112
7	280622	91736

50% Offload rule: 8 K message tests

The test results compare the maximum message depth that was reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages that were offloaded on each queue manager. These results are shown in Table 22-19.

Table 22-19 50% Offload rule: 8 K message test results

Test Number	Total Messages Default Rule	Total Message 50% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 50% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 50% Rule
1	88899	162608	34610	75396	35140	77145
2	101273	185360	39355	91038	40776	92886
3	94597	195360	36355	90910	37101	93014
4	115124	183270	42950	85091	42908	86743
5	135115	216126	50612	100758	52324	102658
6	135115	271777	50661	126270	52275	129997
7	135115	271777	50905	126868	52031	129399

The trend of much deeper queues continued with the 8 K messages when the 50% offload rule was used.

Finding: The maximum queue depth when the 50% message body offload was used was two times higher than when the default values were used.

The comparison of the 50% message body offload rule with the message depths that were achieved when there was no offloading at all is shown in Table 22-20.

Table 22-20 50% Offload rule compared to no offloading: 8 K messages

Test Number	Total Message 50% Rule	Total Messages No offloading
1	162608	21484
2	185360	27286
3	195360	30188
4	183270	37036
5	216126	45344
6	271777	48871
7	271777	48871

50% Offload rule: 16 K message tests

The test results compare the maximum message depth that was reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages that were offloaded on each queue manager. These results are shown in Table 22-21.

Table 22-21 50% Offload rule: 16 K message test results

Test Number	Total Messages Default Rule	Total Message 50% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 50% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 50% Rule
1	85150	178929	36950	86049	38286	87616
2	75136	175240	32196	84218	33026	85701
3	87722	218653	37686	105112	38966	107012
4	87722	212595	37724	101398	38928	104668
5	87722	241337	37676	115880	38976	118203
6		269342		129616		131715
7		269342		129455		131876

The 16 K message showed a substantial increase in the depth when an across the board 50% message body offload was used. In addition, the saturation point was not reached as quickly.

Finding: The maximum queue depth when the 50% message body offload was used was slightly more than three times higher than when the default values were used.

The comparison of the 50% message body offload rule with the message depths that were achieved when there was no offloading at all is shown in Table 22-22.

Table 22-22 50% Offload rule compared to no offloading: 16 K messages

Test Number	Total Message 50% Rule	Total Messages No offloading
1	178929	21484
2	175240	27286
3	218653	30188
4	212595	37036
5	241337	45344
6	269342	48871
7	269342	48871

50% Offload rule: 32 K message tests

The test results compare the maximum message depth that was reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages that were offloaded on each queue manager. These results are shown in Table 22-23.

Table 22-23 0% Offload rule - 32K message test results

Test Number	Total Messages Default Rule	Total Message 50% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 50% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 50% Rule
1	81701	177803	38351	86318	38925	88790
2	85667	177803	40294	86652	40948	88456
3	104241	219299	49402	105428	50018	110545
4	124322	236986	58811	116451	60220	116450
5	145154	237472	68822	116444	70516	116444
6	166547	237531	78908	116445	81244	116445
7	175693	238028	83246	116444	84855	116444
8	177099	238030	82924	116445	85659	116445
9	195461	238028	92263	116444	94619	116444

The 32 K message showed an increase in the depth when an across the board 50% message body offload was used. In addition, the saturation point was not reached as quickly.

Finding: The maximum queue depth when the 50% message body offload was used was just over 1.2 times higher than when the default values were used.

The comparison of the 50% message body offload rule with the message depths that were achieved when there was no offloading at all is shown in Table 22-24.

Table 22-24 50% Offload rule compared to no offloading: 32 K messages

Test Number	Total Message 50% Rule	Total Messages No offloading
1	177803	6666
2	177803	8221
3	219299	10052
4	236986	11108
5	237472	12249
6	237531	13217
7	238028	13217
8	238030	13217
	238028	

50% Offload rule: 64 K message tests

The test results compare the maximum message depth reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages that were offloaded on each queue manager. These results are shown in Table 22-25.

Table 22-25 50% Offload rule: 64 K message test results

Test Number	Total Messages Default Rule	Total Message 50% Rule	CSQ1 Offloaded messages - Default Rule	CSQ1 Offloaded messages - 50% Rule	CSQ2 Offloaded messages - Default Rule	CSQ2 Offloaded messages - 50% Rule
1	100297	123295	49973	61648	50324	61647
2	100297	123297	49998	61650	50299	61647
3	123294	123294	61647	61647	61647	61647

For the 64 K messages, there was no increase in the maximum queue depth, which is expected because messages over 63 K always are offloaded. There is no comparison chart against no offloading for that reason.

50% Offload rule conclusion

If conserving coupling facility space or deeper queues are a requirement, tuning the offload rules and the use of shared message data sets helps to meet that requirement when messages are less than 63 K. The benefits vary based on the following factors:

- ▶ Size of the coupling facility structure
- ▶ Whether ALLOWAUTOALT is enabled
- ▶ The shared message data set size
- ▶ Whether the SMDS can expand
- ▶ The message body size
- ▶ The offload rules that are in place

22.6 SMDS offload: Effects of 100% message body offload

In the previous test series, the default offload rule and a 50% coupling facility structure full rule were compared. In this series, the offload rule is set to offload all (100%) of the message bodies all of the time.

These tests still take advantage of the ALLOWAUT0ALT feature.

The offload rule change was made by using WebSphere MQ Explorer. The offload rules for those tests are shown in Figure 22-5.

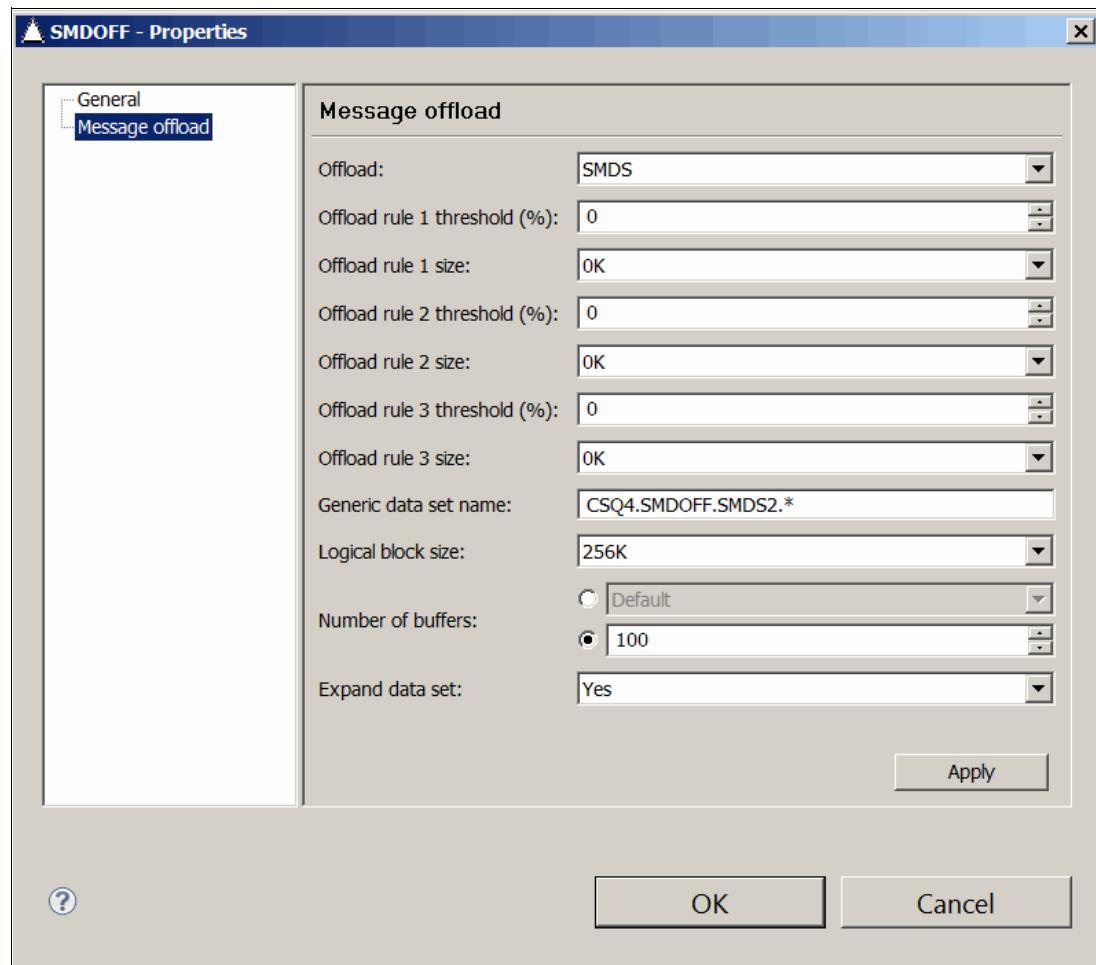


Figure 22-5 100% message body offload rule

Adjusting the offload rule to put all message bodies onto external storage is useful in an environment where the coupling facility structure size is constrained, or the getting application is known to be slower than the putting application. This test series compares the number of messages that are stored when the 100% rule is used with the default graduated rule.

100% Offload rule: 1 K messages

The test results that are shown in Table 22-26 compare the total number of messages that are put from the graduated rule and the 100% rule, and the number of messages that are offloaded on each queue manager. The total number of messages put on each queue manager is not included.

Table 22-26 100% Offload rule: 1 K message test results

Test Number	Total Messages Default Rule	Total Message 100% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 100% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 100% Rule
1	148002	279210	18203	152999	18268	126211
2	148002	244970	18224	172130	18246	172840
3	179398	424971	21133	212264	21137	212707
4	196378	424971	23179	212219	23311	212752
5	199087	471180	23185	236009	23305	235171
6	199087	563850	23202	281270	23288	282580
7	199087	563850	23152	281244	23338	282606

For messages this small, 100% offloading made even more difference in the maximum message depth over the default, graduated offload rule that was used for SMDS.

Finding: The use of 100% message body offloading increased the capacity by 2.8 times over the default rule.

The comparison of the 100% message body offload rule with the message depths that were achieved when there was no offloading and at 50% at all is shown in Table 22-27.

Table 22-27 100% Offload rule compared to 50% and no offloading: 1 K messages

Test Number	Total Message 100% Rule	Total Message 50% Rule	Total Messages No offloading
1	279210	217216	114914
2	244970	241284	148433
3	424971	267981	182332
4	424971	329319	201825
5	471180	310537	223325
6	563850	359538	240805
7	563850	359538	240805

100% Offload rule: 4 K messages

The test results that are shown in Table 22-28 compare the total number of messages put from the graduated rule and the 100% rule, and the number of messages that were offloaded on each queue manager. The total number of messages put on each queue manager is not included.

The saturation point for the 100% offload test was reached in fewer runs than the deafly or 50% offload rule. In Table 22-28, the message counts are given for the tests that are the most similar.

Table 22-28 100% Offload rule: 4 K message test results

Test Number	Total Messages Default Rule	Total Message 100% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 100% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 100% Rule
1	105069	385204	36110	157763	36628	196781
2	128477	475428	42879	232935	43967	240486
3	125535	573544	40715	282564	41683	290980
4	120551		33261		34100	
5	150061	573544	44262	280812	45415	292732
6	131308		32145		32926	
7	163051	573544	44323	280812	45741	293190

The comparison of the 100% message body offload rule with the message depths that were achieved when there was no offloading and at 50% at all is shown in Table 22-29.

Finding: Using 100% message body offloading increased the capacity by 3.5 times over the default rule.

Table 22-29 100% Offload rule compared to 50% and no offloading: 4 K messages

Test Number	Total Message 100% Rule	Total Message 50% Rule	Total Messages No offloading
1	385204	217216	114914
2	475428	241284	148433
3	573544	267981	182332
4		329319	201825
5	573544	310537	223325
6		359538	240805
7	573544	359538	240805

100% Offload rule: 8 K messages

The test results that are shown in Table 22-30 compare the total number of messages put from the graduated rule and the 100% rule, and the number of messages that were offloaded on each queue manager. The total number of messages put on each queue manager is not included.

Table 22-30 100% Offload rule: 8 K message test results

Test Number	Total Messages Default Rule	Total Message 100% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 100% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 100% Rule
1	88899	279226	34610	157763	35140	126211
2	101273	310380	39355	154648	40776	155732
3	94597	344968	36355	171700	37101	173268
4	115124	383255	42950	190690	42908	192565
5	135115	471179	50612	234373	52324	236806
6	135115	563855	50661	280041	52275	283814
7	135115	563855	50905	280570	52031	283285

For 8 K messages, 100% offloading made even more difference in the maximum message depth over the default, graduated offload rule that was used for SMDS.

Finding: The use of 100% message body offloading increased the capacity by 4.1 times over the default rule.

The comparison of the 100% message body offload rule with the message depths that were achieved when there was no offloading and at 50% at all is shown in Table 22-31.

Table 22-31 100% Offload rule compared to 50% and no offloading: 8 K messages

Test Number	Total Message 100% Rule	Total Message 50% Rule	Total Messages No offloading
1	279226	162608	21484
2	310380	185360	27286
3	344968	195360	30188
4	383255	183270	37036
5	471179	216126	45344
6	563855	271777	48871
7	563855	271777	48871

100% Offload rule: 16 K messages

The test results compare the maximum message depth that was reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages that were offloaded on each queue manager. These results are shown in Table 22-32.

Table 22-32 100% Offload rule: 16 K message test results

Test Number	Total Messages Default Rule	Total Message 100% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 100% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 100% Rule
1	85150	419201	36950	209600	38286	209601
2	75136	419202	32196	209600	33026	209612
3	87722	419200	37686	209600	38966	209600

The 16 K message showed another substantial increase in the depth when an across the board 100% message body offload was used over the default rule.

Finding: The use of 100% message body offloading increased the capacity by 4.8 times over the default rule.

Comparing the 100%, 50%, and no offloading message depth is shown in Table 22-33. Because the saturation point is reached much more quickly, there are only three 100% totals in the table. The difference between the three tests was only two messages.

Table 22-33 100% Offload rule compared to no offloading: 16 K messages

Test Number	Total Message 100% Rule	Total Message 50% Rule	Total Messages No offloading
1	419201	178929	21484
2		175240	27286
3	419202	218653	30188
4		212595	37036
5		241337	45344
6	419200	269342	48871
7		269342	48871

100% Offload rule: 32 K messages

The test results compare the maximum message depth that was reached for the default, graduated offload rule (see 22.4, “SMDS offload: Large SMDS data sets” on page 349) and the 50% rule, and the number of messages that were offloaded on each queue manager. These results are shown in Table 22-34.

Table 22-34 100% Offload rule: 32 K message test results

Test Number	Total Messages Default Rule	Total Message 100% Rule	CSQ1 Offloaded messages: Default Rule	CSQ1 Offloaded messages: 100% Rule	CSQ2 Offloaded messages: Default Rule	CSQ2 Offloaded messages: 100% Rule
1	161615	232902	75306	116451	77730	116451
2	182747	232902	85348	116451	88745	116451
3	182747	232890	86377	116445	87716	116445

The selection of the default tests was made from the final tests. The 100% offload rule did not show the growth and variation in the number of messages and offloads that were observed in the default of 50% offload rules.

The 32 K message showed an increase in the maximum depth when an across the board 100% message body offload over the default rule was used.

Finding: The use of 100% message body offloading increased the capacity by 1.2 times over the default rule.

Comparing the 100%, 50%, and no offloading message depth is shown in Table 22-35. Because the saturation point is reached much more quickly, there are only three 100% totals in the table. The difference between the three tests was only two messages.

Table 22-35 100% Offload rule compared to 50% no offloading: 32 K messages

Test Number	Total Message 100% Rule	Total Message 50% Rule	Total Messages No offloading
1	232902	177803	6666
2		177803	8221
3		219299	10052
4	232902	236986	11108
5		237472	12249
6	232890	237531	13217
7		238028	13217

100% Offload rule: 64 K messages

The results are the same as shown in, “50% Offload rule: 64 K message tests” on page 361 because 64 K messages are always offloaded.

100% Offload rule conclusion

If conserving coupling facility space or deeper queues are a requirement, putting the entire message body onto the shared message data set is a good solution. This configuration is especially useful when there is a great deal of space available for the SMDS. The benefit varies based on the following factors:

- ▶ Size of the coupling facility structure
- ▶ Whether ALLOWAUTOALT is enabled
- ▶ The shared message data set size
- ▶ Whether the SMDS can expand
- ▶ The message body size
- ▶ The offload rules that are in place

22.7 SMDS offload: Mixed message sizes

These series of tests compared the maximum message depths and offloads for the default offloading rule, the 50% rule, and the 100% message body rule. Each series was run eight times. The results show that the wide distribution of message sizes reduces the ability to predict the queue depth benefit of offloading at all levels.

All the queues that were used in these tests were defined on the SMDOFF structure. Although the messages are targeted towards different queues, they resolve to the same storage areas: the coupling facility and the shared message data set. The shared message data set remains allocated at 5,000 cylinders in primary storage.

SMDS Offload: Mixed message sizes, default offload rule

In this series of tests, the default offload rule was used. Between tests 7 and 8, the coupling facility structure that is used (SMDOFF) was resized downwards. The ALLOWAUTOALT(YES) on the structure definition allows the system to reclaim unused CF storage when other areas might need it.

In Figure 22-6, the number of messages of each size, the total number of messages, and the offload count for each test are shown.

Default Rule	1K	4K	8K	16K	32K	64K	Total Msgs	Offload count
Test 1	18098	21410	18762	16272	13355	10034	97931	31035
Test 2	24483	31108	27817	23439	19198	14462	140507	46426
Test 3	27789	34570	30673	26023	21194	16054	156303	51443
Test 4	49321	32911	28748	23948	19487	7412	161827	47506
Test 5	28928	35552	31747	26947	22168	16485	161827	53019
Test 6	28927	35713	31920	26865	22001	16401	161827	53013
Test 7	48338	33725	29837	25009	9848	14980	161737	46943
Test 8	42491	24824	22360	18740	13673	7608	129696	31275

Figure 22-6 Results of the mixed message sizes with default offload rule test

SMDS Offload: Mixed message sizes, 50% offload rule

In this series of tests, all message bodies are offloaded when the coupling facility structure becomes 50% full. The structure size was increased during this series and there was no decrease in the size. Overall queues deepened through the test series.

One interesting note is the first test showed fewer messages of mixes sizes that were stored than the default offload rule. After the first test, the expansion of the coupling facility structure size provided more space, gradually allowing more messages to be stored in the available space. The results are shown in Figure 22-7.

50% Rule	1K	4K	8K	16K	32K	64K	Total Msgs	Offload count
Test 1	10934	17099	15381	13265	10746	6681	74106	28728
Test 2	16052	25724	22778	18820	15315	9866	108555	42645
Test 3	22235	31745	29268	23818	19469	12913	139448	55979
Test 4	26260	42245	37215	31455	25649	16768	179592	73662
Test 5	29897	50285	44997	37671	31174	20442	214466	88394
Test 6	33755	57748	51297	43287	35862	23517	245466	102888
Test 7	41438	72581	64769	53410	43604	28874	304676	127617
Test 8	49881	86247	77153	64862	53650	35543	367336	154791

Figure 22-7 Results of the mixed message sizes with the 50% offload rule test

SMDS Offload: Mixed message sizes, 100% offload rule

The offload rule was altered to 100% of all message bodies before this test series. As in other tests, the structure expanded and the queues deepened. From a queue depth perspective, the 100% offload when there are mixed message sizes does not provide the same message depths as the 50% offload rule. If more direct access storage devices (DASDs) for the linear data sets was available, the results are shown in Figure 22-8.

100% Rule	1K	4K	8K	16K	32K	64K	Total Msgs	Offload count
Test 1	18561	32659	30146	26347	22678	16183	146574	60060
Test 2	23424	40115	37352	32590	28358	20211	182050	74806
Test 3	25902	45077	41792	35978	31362	22048	202159	82332
Test 4	27956	50006	45981	40386	35180	25172	224681	96838
Test 5	38799	64023	56811	48287	39670	26337	273927	114935
Test 6	39192	63871	57230	48076	39573	25985	273927	114571
Test 7	40338	64044	57180	48249	38548	25568	273927	113379
Test 8	37867	64593	57386	48458	39371	25889	273564	115262

Figure 22-8 Results of the mixed message sizes with the 100% offload rule test

Mixed message sizes conclusion

As with fixed size messages, mixed message workload can achieve greater message depths when shared message data sets and offloading rules are used. The effects are difficult to predict, which makes resource planning more challenging.

22.8 DB2 to SMDS offload migration

This section includes information about how to migrate the WebSphere MQ structure definition to use SMDS and the effect on message offloading.

Migrating the CFSTRUCT

Complete the following steps to migrate an existing WebSphere MQ CFSTRUCT definition to use SMDS for message body offloading:

1. Define the shared message data set for each queue manager in the queue sharing group, as shown in Figure 22-9. For more information, see 11.1.4, “Migration from DB2” on page 166.

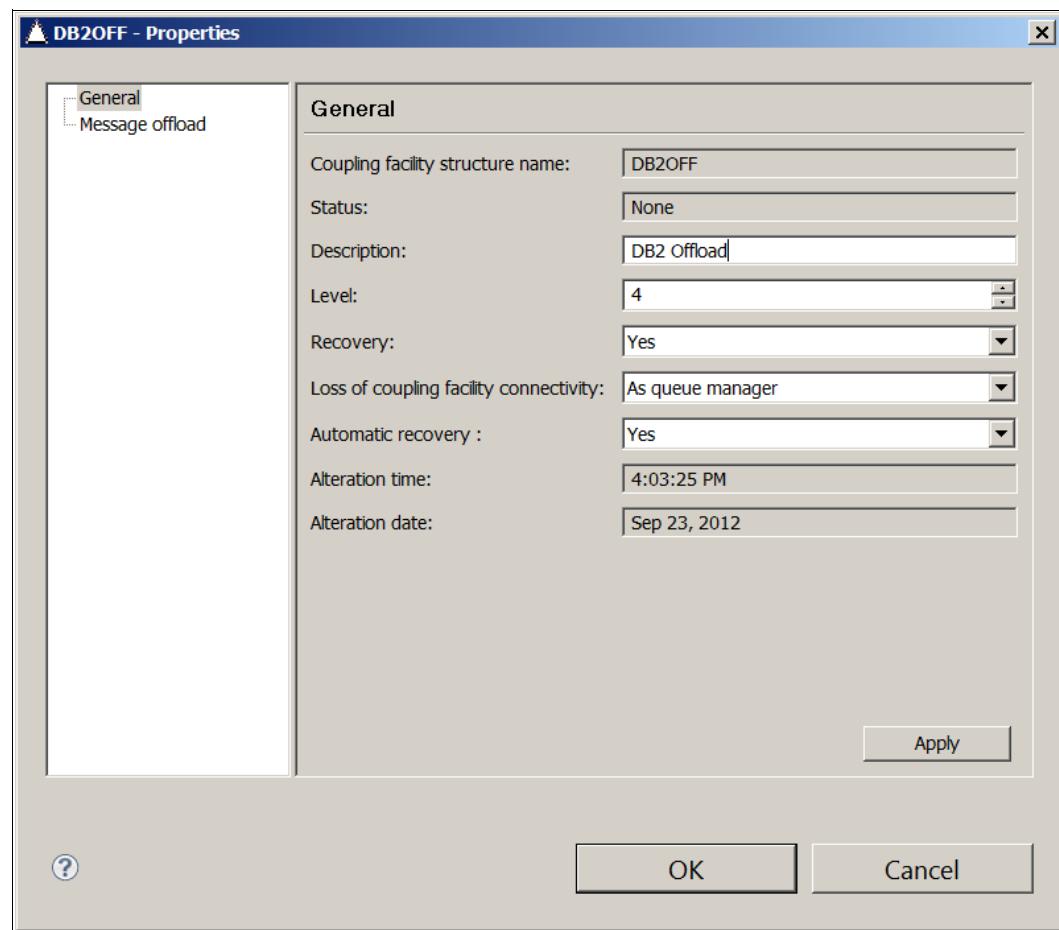


Figure 22-9 Original WebSphere MQ CFSTRUCT definition

2. Complete the following steps to migrate the structure from an older CFLEVEL to level 5:
 - a. Using the WebSphere MQ Explorer, display the CFSTRUCT, as shown in Figure 22-9.

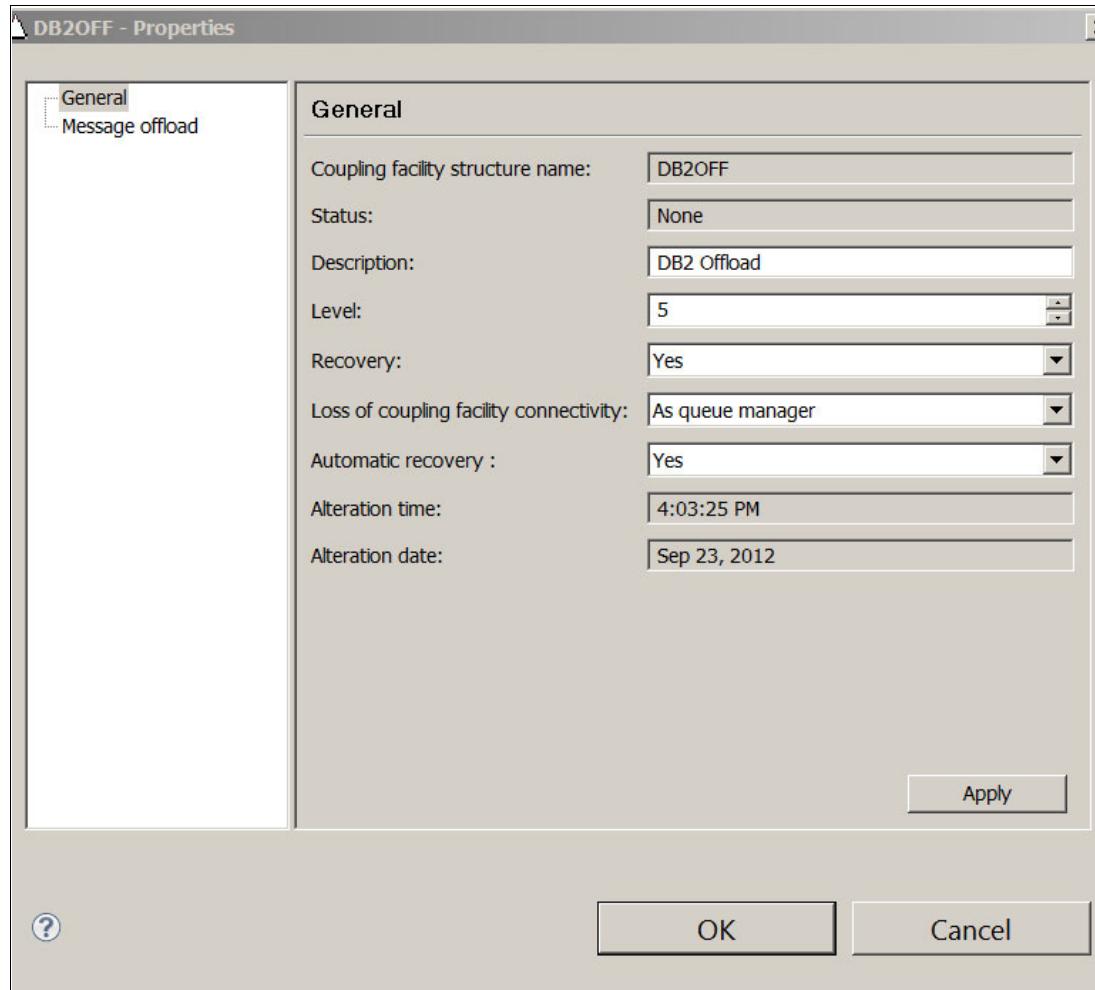


Figure 22-10 Setting the CFLEVEL to 5

2. Set the level to 5, as shown in Figure 22-10 on page 371.

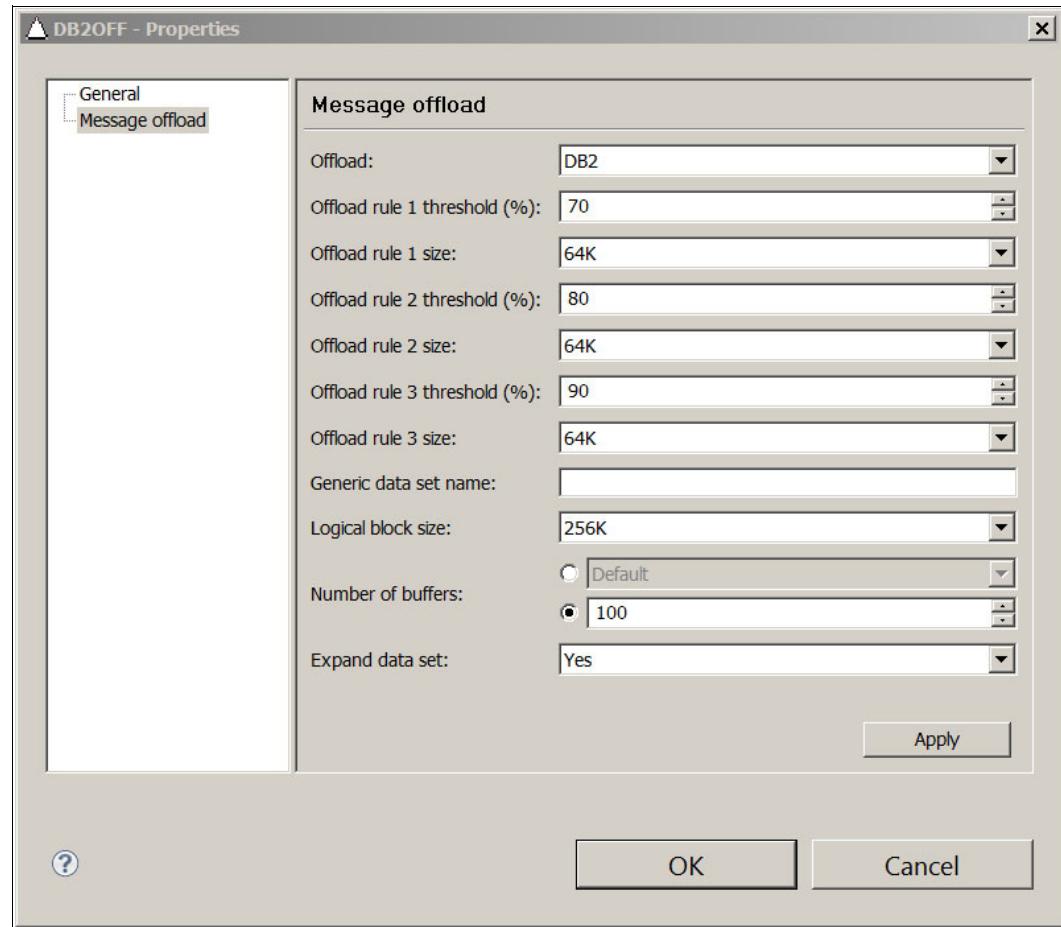


Figure 22-11 Original WebSphere MQ CFSTRUCT Message offload definition

- c. Optionally change the description.
- d. Click **Apply**.
3. Complete the following steps to change the offload attributes:
 - a. Select the **Message offload** folder, as shown in Figure 22-12 on page 373.

Important: The values displayed for the offload attributes vary. They are based on the original CFLEVEL of the structure.

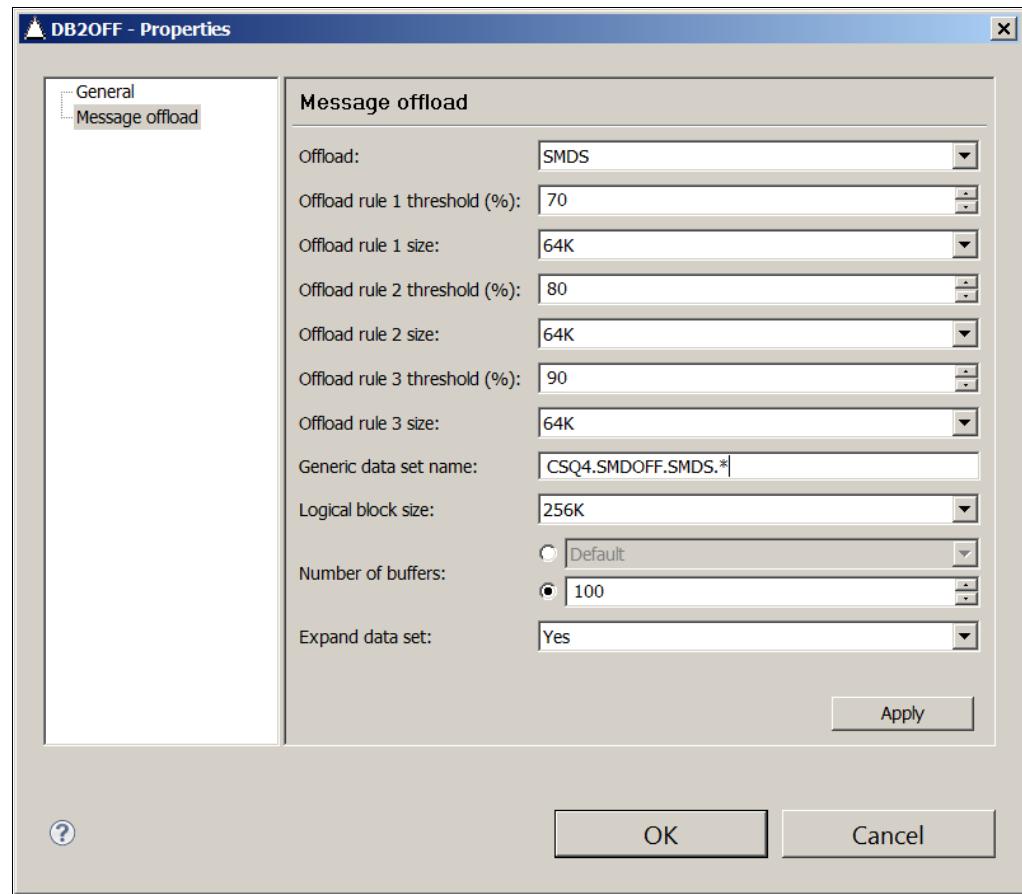


Figure 22-12 Setting the offload properties for SMDS

- b. Change the following OFFLOAD attributes from the original values as shown in Figure 22-11 on page 372:
 - i. Set the OFFLOAD attribute to SMDS.
 - ii. Add the Generic data set name and click **OK** or **Apply**, as shown in Figure 22-12 on page 373.

Message body storage during migration

Control information about the physical location of the message body is kept within the coupling facility structure for each message. This ability provides a great deal of flexibility for moving between the use of DB2 and SMDS as the storage medium.

If the offload target changes from DB2 to SMDS, or SMDS to DB2, and messages were offloaded to each target; the offload use indicates both. The results of the **DISPLAY CFSTATUS TYPE(SUMMARY)** command are shown in Figure 22-13.

```
SDSF OUTPUT DISPLAY CSQ1MSTR STC06813 DSID      2 LINE 464      COLUMNS 02- 81
COMMAND INPUT ===> _                                     SCROLL ==> CSR
18.34.41 STC06813 CSQM201I -CSQ1 CSQMDRTC  DISPLAY CFSTATUS DETAILS  413
 413          CFSTATUS (DB2OFF)
 413          TYPE (SUMMARY)
 413          CFTYPE (APPL)
 413          STATUS (ACTIVE)
 413          OFFLDUSE (BOTH)
 413          SIZEMAX (272384)
 413          SIZEUSED (1)
 413          ENTSMAX (119954)
 413          ENTSUSED (43)
 413          FAILTIME( )
 413          FAILDATE( )
 413          END CFSTATUS DETAILS
18.34.41 STC06813 CSQ9022I -CSQ1 CSQMDRTC ' DISPLAY CFSTATUS' NORMAL COMPLETION
***** BOTTOM OF DATA *****
```

Figure 22-13 *DISPLAY CFSTATUS* summary

Complete the following steps to re-create the offload use:

1. While the CF structure is using DB2 as the offload storage, put messages on the queue that are offloaded. (In the example the message bodies are over 64K).
2. Alter the CF structure definition to use SMDS for message offloads, see, “Migrating the CFSTRUCT” on page 370.
3. Put messages on the queue that are offloaded.
4. Display the summary information for the structure.

In this scenario, five messages were put on the queue while the message bodies were put to DB2 and five were put on the queue while the message bodies were put to the shared message data set. All the messages are fully available, as shown in Figure 22-14 on page 375.

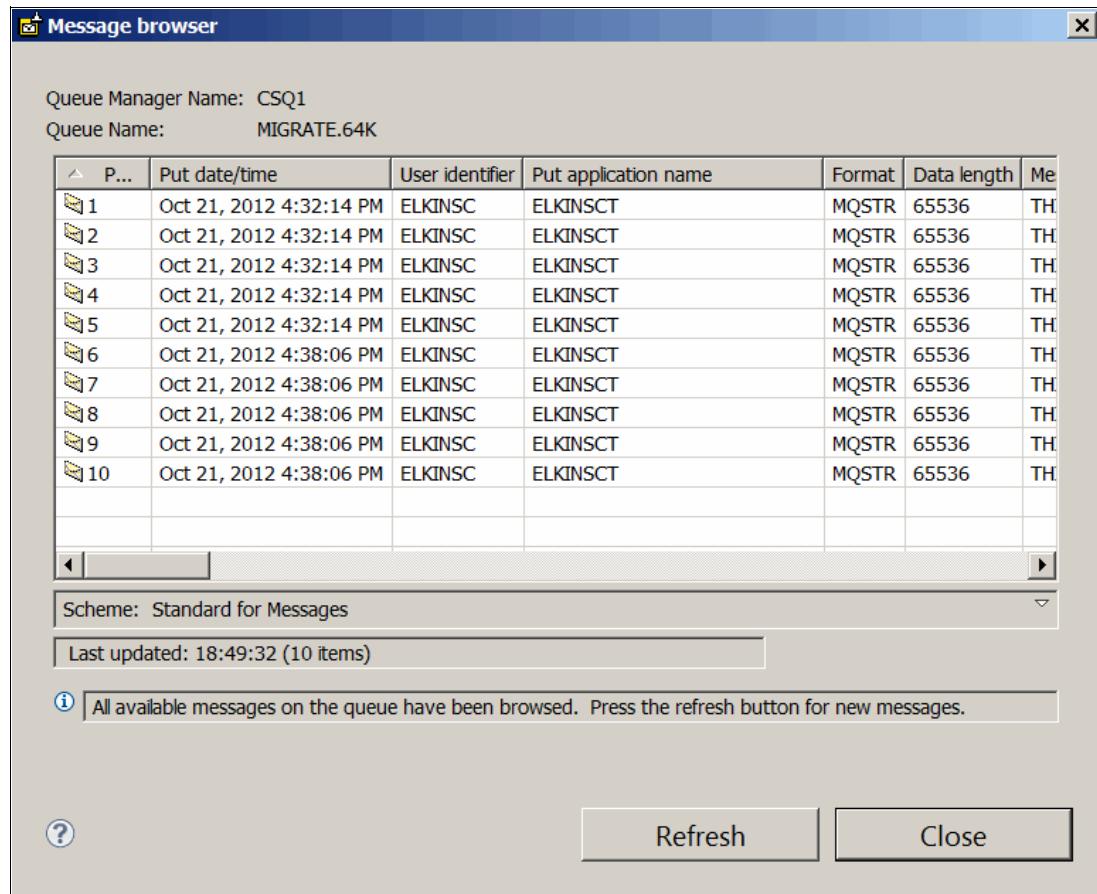


Figure 22-14 Browse of messages with migrated offload

22.9 SMDS offload: Message body availability

Messages that are stored in a shared message data set are available even when the queue manager that owns the SMDS is unavailable. The behavior from an application perspective is no different from messages that are stored in the coupling facility entirely or offloaded to DB2.

In this scenario, message body availability is tested by using the following process:

1. Put messages from queue manager CSQ1.
2. Stop queue manager CSQ1.
3. Get the messages from CSQ2.

Putting the messages

In Example 22-17, the output of the OEMPUTX program shows the 10 messages that are put on the queue from queue manager CSQ1.

Example 22-17 Messages are put to the shared queue

```
buffer:-qMIGRATE.64K      * request queue
buffer:-fileDD:MSGIN
buffer:-dJTEST4
parm: -mCSQ1 -n10. -x -l1 -c1 -s65536
Message file: DD:MSGIN
OEMPUTX about to MQCONN to QMgr CSQ1.
OEMPUTX about to MQOPEN request queue: MIGRATE.64K
CPU type 3A3BD52817
Date Time 2012/10/21 23:30:13.
Description JTEST4.
Entering PUT only loops...
Preload the queue with 0 messages...
  Message size      : 65536
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages       : 10
  Syncpoints          : NO-SYNCPOINT
Starting loop at 2012-10-21 23:30:13.151175
  Workload manager data
  Samples %idle %unknown(MQ?) %using CPU %doing
    CSQ1CHIN.008A     10   100        0        0
    CSQ1MSTR.0078     10   100        0        0
-----
  Total Transactions : 10
  Elapsed Time       : 0.025 seconds
  Application CPU Time: 0.015 seconds (57.7%)
  Transaction Rate   : 393.778 trans/sec
-----
  Round trip per msg : 2539 microseconds
  Avg App CPU per msg : 1464 microseconds
-----
  Jobname.ASID   TCB(uS)   SRB(uS)  Tot(uS)  (%)
                /tran     /tran    /tran
-----
  CSQ1MSTR.0078 00000000 00000002 00000002 0.1
  CSQ1CHIN.008A 00000000 00000000 00000000 0.0
  CSQ1BRK*       00000000 00000000 00000000 0.0
```

Stopping the queue manager

The queue manager is stopped by using the **-CSQ1 STOP QMGR** command.

The queue manager display from WebSphere MQ Explorer when the queue manager is down is shown in Figure 22-15.

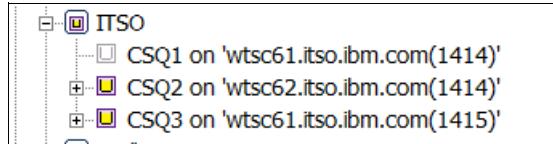


Figure 22-15 Display of queue managers in the QSG

Retrieving the messages from another queue manager

In Example 22-18, the output of the OEMPUTX program shows the 10 messages that are put on a different shared queue, while the original 10 messages on the MIGRATE.64K queue are read. The messages are retrieved without difficulty even though the original queue manager is down.

Example 22-18 Messages are retrieved from the shared queue

```
buffer:-qSMDOFF.64K      * request queue
buffer:-rMIGRATE.64K      * reply queue
buffer:-fileDD:MSGIN
buffer:-dJTEST4
parm: -mCSQ2 -n10. -x -l1 -c1 -s65536
Message file: DD:MSGIN
OEMPUTX about to MQCONN to QMgr CSQ2.
OEMPUTX about to MQOPEN request queue: SMDOFF.64K
OEMPUTX about to MQOPEN reply queue: MIGRATE.64K
CPU type 3B3BD52817
Date Time 2012/10/21 23:51:53.
Description JTEST4.
Entering PUT/GET loops (MQGET_Wait=60 seconds)...
Preload the queue with 0 messages...
Message size : 65536
Message persistence: NON-PERSISTENT
Messages per loop : 1
Total messages : 10
Syncpoints : NO-SYNCPOINT
MQGET replies by : Any message
Starting loop at 2012-10-21 23:51:53.520880
Workload manager data
          Samples %idle %unknown(MQ?) %using CPU %doing I/O
CSQ2CHIN.0066      7   100           0       0       0
CSQ2MSTR.004E      7   100           0       0       0
-----
Total Transactions : 10
Elapsed Time : 0.037 seconds
Application CPU Time: 0.005 seconds (13.2%)
Transaction Rate : 273.090 trans/sec
-----
Round trip per msg : 3661 microseconds
Avg App CPU per msg : 483 microseconds
-----
Jobname.ASID TCB(uS) SRB(uS) Tot(uS) (%)
/tran    /tran    /tran
```

There were no messages that indicated this activity was taking place.



GROUPUR: Using Group units of recovery with CICS

This chapter provides a working example with instructions on how to enable Group units of recovery between CICS and a z/OS WebSphere MQ queue manager in a queue-sharing group.

This chapter contains the following sections:

- ▶ Preparing the scenario
- ▶ Enabling indoubt units of work before Group units of recovery are enabled
- ▶ Configuring the environment to support Group units of recovery
- ▶ Enabling indoubt units of work after Group units of recovery are enabled

23.1 Preparing the scenario

In this chapter, we review a scenario in which a CICS region loses its WebSphere MQ connection before a CICS transaction can commit the messages it PUT to a queue.

This scenario is run before and after the Group units of recovery are enabled between CICS and the WebSphere MQ queue manager. The goal of the scenario is to show how enabling groups units of recovery between CICS and a queue-sharing group improves availability.

To create indoubt units of work, the CICS supplied transaction CIND is used. For more information about the use of the CIND transaction, see the CICS Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.ts.systemprogramming.doc/transactions/cind/dfha74e.html>

When a transaction that is monitored by the CIND tool is identified, it is abended at a point that causes an indoubt of work to be created.

The transaction MVPT is used to generate the indoubt units of work.

The following resources are needed to install the MVPT transaction are provided as samples within the WebSphere MQ product installation libraries:

- ▶ CICS required RDO entries are found in member CSQ4S100 in the WebSphere MQ th1q.SCSQPROC library.
- ▶ The program load modules are found in the WebSphere MQ th1q.SCSQCICS library.

Important: Replace th1q with the high-level qualifier of the WebSphere MQ libraries.

- ▶ Installation instructions are found in the WebSphere MQ Information Center at this website:
http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/fg18160_.htm

To activate Group units of recovery for CICS, the following prerequisites must be met:

- ▶ WebSphere MQ must be version 7.1 or greater.
- ▶ A WebSphere MQ queue-sharing group must be present.
- ▶ CICS TS version must be 4.2 or greater.
- ▶ The CICS region must attach to the queue-sharing group, not an individual queue manager.

The following environment setup was used to test the scenarios:

- ▶ LPAR:
 - SC61
 - SC62
- ▶ WebSphere MQ queue-sharing group: IBM1
- ▶ WebSphere MQ queue-sharing group members:
 - CSQ1 - SC61
 - CSQ2 - SC62
 - CSQ3 - SC61

- ▶ CICS Regions:
 - CICSPAZ5 - SC61
 - CICSPAZ7 - SC62
- ▶ Software versions:
 - CICS TS V4.2
 - WebSphere MQ V7.1
- ▶ z/OS 1.13

Figure 23-1 shows the environment on which the scenarios were run.

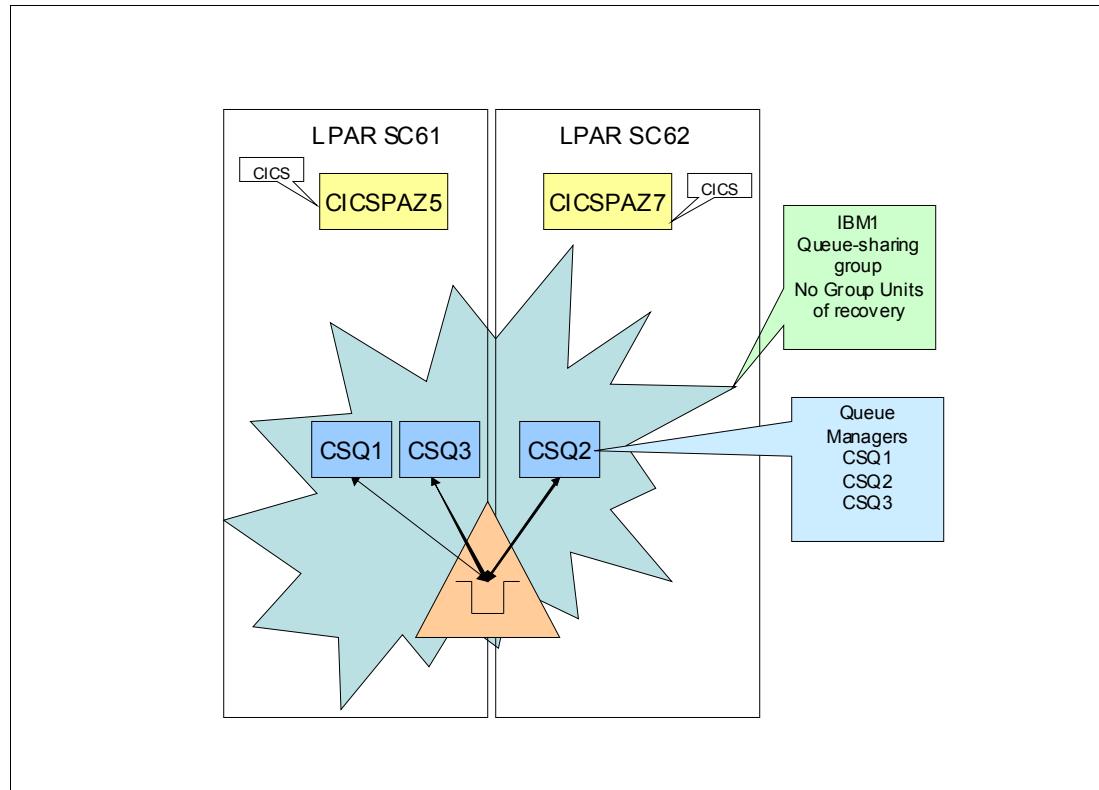


Figure 23-1 Environment for scenarios

23.2 Enabling indoubt units of work before Group units of recovery are enabled

In this scenario, we show that if group units of work are not enabled and there are WebSphere MQ indoubt units of work that must be recovered between CICS and WebSphere MQ, the CICS region must reconnect to the same WebSphere MQ queue manager to recover the lost units of work.

We verify that group units of work are not enabled by examining the CICS and WebSphere MQ informational displays.

Complete the following steps to prepare the environment for Group units of recovery testing:

1. Sign in to the CICS region and display the CICS to WebSphere MQ connection parameters.
2. Run the **CEMT INQUIRE MQCONN** command to verify that the Resyncmember parameter is not set to Groupresync.

Figure 23-2 shows the results of the Resyncmember parameter.

```
I MQCONN
STATUS: RESULTS - OVERTYPE TO MODIFY
Mqname( IBM1 )
Mqmgr(CSQ3)
Mqrelease(0710)
Connectst( Connected )
Resyncmember( Noresync )
Tasks(0001)
Trigmontasks(0001)
Installtime(10/17/12 12:40:46)
Installusrid(CICSUSER)
Installagent(Grplist)
Definesource(WMQ)
Definetime(09/18/12 16:04:35)
Changetime(10/17/12 12:19:30)
Changeusrid(CICSUSER)
Changeagent(Csdapi)
Changeagrel(0670)

SYSID=WRKS APPLID=CICSPA25
TIME: 12.47.00 DATE: 10/17/12
RESPONSE: NORMAL
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 23-2 CICS CEMT INQUIRE MQCONN display results

Important: The Noresync setting of the Resyncmember parameter means the CICS region attempts to connect to the same queue manager once. If that attempt fails, the region connects to any available member in the queue-sharing group.

3. Press PF3 to exit.
4. Clear the panel.

5. Enter CICS transaction CKQC to display the WebSphere MQ connection information.

Figure 23-3 shows the CKQC transaction.

CKQC

Figure 23-3 CKQC transaction

6. Press Enter.

Figure 23-4 shows the CKQC panel.

_Connection

CKTI

Task

--

CKQCM0 CICS Adapter Control -- Initial panel

Select menu bar item using Tab key. Then press Enter.

F1=Help F3=Exit

Figure 23-4 Display for results of initial CKQC command

7. Place the cursor on Connection at the top of the panel and press Enter.

Figure 23-5 shows the Select an action menu.

Connection	CKTI	Task
+-----+ -- Select an action. apter Control -- Initial panel 4 1. Start... sing Tab key. Then press Enter. 2. Stop... 3. Modify... 4. Display +-----+ F1=Help F12=Cancel +-----+		

F1=Help F3=Exit

Figure 23-5 CKQC panel to select an action

8. Enter 4 to select Display.
 9. Press Enter to display the WebSphere MQ connection information.

Figure 23-6 shows the Display Connection panel.

CKQCM2 Display Connection panel							
Read connection information. Then press F12 to cancel.							
CICS Applid = CICSPAZ5 Connection Status = Connected QMgr name= CSQ3 Mqname = IBM1 Tracing = On API Exit = Off Initiation Queue Name = IBM1.AOR.INITQ							
----- STATISTICS -----							
Number of in-flight tasks = 1 Total API calls = 1 Number of running CKTI = 1 APIs and flows analysis Syncpoint Recovery							
----- Run OK 1 MQINQ 0 Tasks 0 Indoubt 0 Futile 0 MQSET 0 Backout 0 UnResol 0 MQOPEN 1 ----- Flows ----- Commit 0 Commit 0 MQCLOSE 0 Calls 6 S-Phase 0 Backout 0 MQGET 1 SyncComp 5 2-Phase 0 GETWAIT 1 SuspReqd 0 MQPUT 0 Msg Wait 1 MQPUT1 0 Switched 0							

F1=Help F12=Cancel Enter=Refresh							

Figure 23-6 WebSphere MQ connection panel

Important: CICS regions is connected to queue manager CSQ3 and queue-sharing group IBM1. The Mqname field shows the queue-sharing group name.

Figure 23-7 shows the connection between CICS and the queue manager.

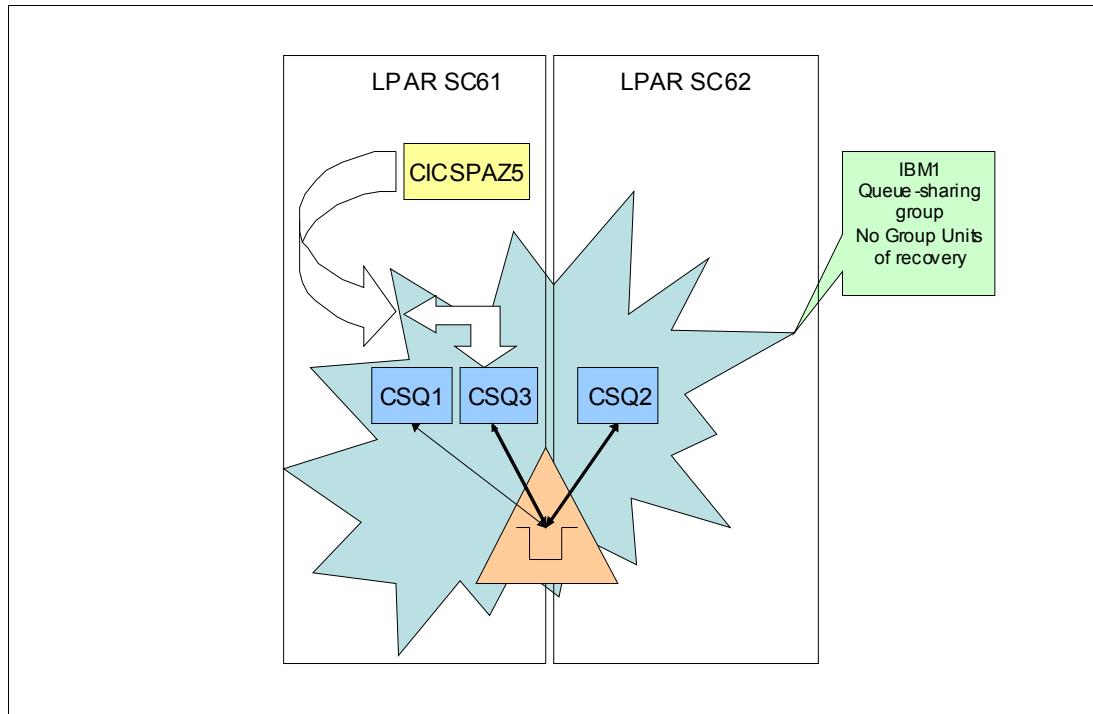


Figure 23-7 CICSPAZ5 connects CSQ3 within the IBM1 queue-sharing group

10. Complete the following steps to exit the CKQC panels:

- a. Press PF12.
- b. Press PK12.
- c. Press PF3.

11. Start WebSphere MQ Explorer and display the queue manager Extended panel, as shown in Figure 23-8.

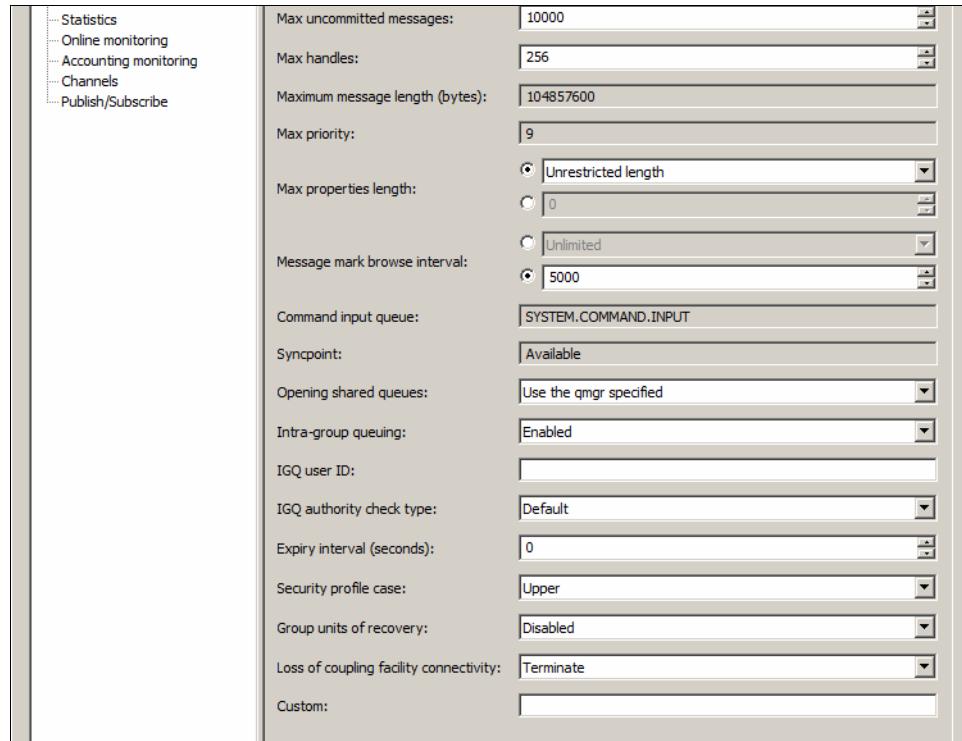


Figure 23-8 WebSphere MQ Explorer queue manager Extended panel

The Group units of recovery should be set to Disabled.

Complete the following steps to conduct the Group units of recovery tests:

1. Determine the CICS transaction that is used and display the transaction definition within CICS where the transaction runs. Use the **CICS CEMT Inquire TRANS(MVPT)** command to allow the Transaction Class to be changed.

Figure 23-9 shows the CEMT INQUIRE command.

```
cemt inquire transaction(MVPT)
```

Figure 23-9 CICS CEMT transaction inquiry

Important: In these scenarios, the sample transaction MVPT is used.

2. Press Enter.

Figure 23-10 shows the results of the command.

```
Inquire TRANSACTION(MVPT)
STATUS: RESULTS - OVERTYPE TO MODIFY
Tra(MVPT) Pri( 001 ) Pro(CSQ4CVK1) Tcl( DFHTCL00 ) Ena Sta
           Prf(DFHCICST) Uda Any             Bac Wai

SYSID=WRKS APPLID=CICSPA25
TIME: 12.49.01 DATE: 10/17/12
RESPONSE: NORMAL
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 23-10 CICS transaction definition display

To allow the CIND transaction to function properly with the MVPT transaction, the Transaction Class (Tcl) must be changed to DFHTCIND.

3. Change the Tcl to DFHTINTD by entering it on the panel and pressing Enter.

The bottom of the panel should read: RESPONSE: NORMAL.

Important: If the response is not shown as NORMAL, you might lack the authority to modify the transaction definition.

Figure 23-11 shows the Tcl was changed.

```
I TRAN(MVPT)
STATUS: RESULTS - OVERTYPE TO MODIFY
Tra(MVPT) Pri( 001 ) Pro(CSQ4CVK1) Tcl( DFHTCIND ) Ena Sta NORMAL
           Prf(DFHCICST) Uda Any                                Bac Wai

SYSID=WRKS APPLID=CICSPA25
TIME: 12.49.43 DATE: 10/17/12
RESPONSE: NORMAL
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 23-11 CICS transaction definition display

4. Press PF3 to exit.
5. Clear the window and enter CIND ON to enable the tool so that indoubt units of work for the MVPT transaction are created.

Figure 23-12 shows the command to start the CIND tool.

```
cind on
```

Figure 23-12 Turn on CIND processing

6. Press Enter.

Figure 23-13 shows that the tool is now active.

```
DFHIN1001 10/17/2012 12:50:12 CICSPA25 TCDD CICSUSER The indoubt tool is now
active for DFHTCIND tranclass transactions.
```

Figure 23-13 The resulting display shows that the indoubt tool is active

The MVPT CICS transaction is initiated from a 3270 terminal by using the following comma-separated parameters:

- The number of messages to PUT
- The fill character
- Message size
- Persistence
- Output Queue name

Important: The following sample transaction and parameters were used to test whether the CIND transaction is functioning properly:

```
- mvpt,1,* ,4096,p,CSQ4SAMP.PUT
```

The output from the MVPT transaction is written to the CSQ4SAMP.PUT queue, as shown in Figure 23-14.

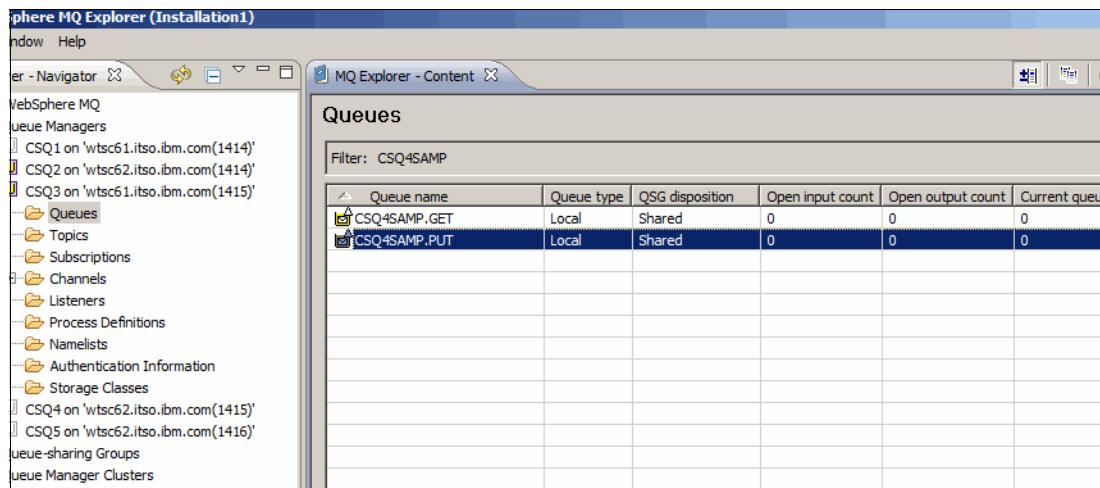


Figure 23-14 Panel showing the Queues that are used for the scenarios

7. Initiate an MVPT transaction to PUT a message to the CSQ4SAMP.PUT queue, as shown in Figure 23-15.

```
mvpt,1,* ,4096,p,CSQ4SAMP.PUT
```

Figure 23-15 Command that is entered into CICS

Figure 23-16 shows that the MVPT was intercepted by CIND and the unit of work did not complete.

DFHAC2201 13:06:12 CICSPAZ5 Transaction MVPT has lost contact with its coordinator system during syncpoint processing and has abended with code ASP1. The unit of work is shunted until contact is restored.

Figure 23-16 Reply when transaction was entered

8. Run the MVPT transaction several times to create multiple indoubt units of work.

The WebSphere MQ Explorer display now shows messages in the queues. CSQ4SAMP.PUT contains messages because CICS abended the MVPT transaction after the MQPUT to create indoubt units of work.

Figure 23-17 shows the current queue depth.

Figure 23-17 Showing current depth of messages in the queues

9. Check the queue status to determine whether there are uncommitted messages. In this case, the status indicates an unresolved unit of work.
 10. Right-click the queue folder and select **Status**, as shown in Figure 23-18.

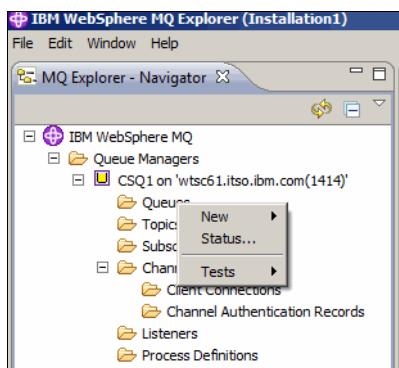


Figure 23-18 Show panel to allow queue status to be displayed

11. Scroll the resulting display to show the uncommitted message count, as shown in Figure 23-19.

Figure 23-19 The queue status panel shows uncommitted units of work

12. Change the CICS region to connect to a different queue manager to show that another manager cannot resolve the indoubt units of work.

Important: In this test, the CSQ3 queue manager was stopped after the CICS connection was broken to force the CICS to connect to a different queue manager.

13. Use the CKQC CICS transaction and use option 2, as shown in Figure 23-20.

Connection	CKTI	Task
		Select an action.
		1. Start... Enter Control -- Initial panel
		2. Stop... sing Tab key. Then press Enter.
		3. Modify...
		4. Display
		F1=Help F12=Cancel

Figure 23-20 Select 2 to stop the connection

14. Select **1 Quiesce** and press Enter, as shown in Figure 23-21.

Connection	CKTI	Task
Select an action.	apter Control -- Initial panel	
2 1. Start...	sing Tab key. Then press Enter.	
2. Stop...		
3. Modify...		
4. Display	Stop Connection	
F1=Help F12=Cancel	Select stop type. Then press Enter.	
	1 1. Quiesce 2. Force	
	F1=Help F12=Cancel	

F1=Help F3=Exit

Figure 23-21 The Stop Connection panel

The resulting message shows that the connection was issued a stop, as shown in Figure 23-22.

CKQCM1	CICS adapter messages panel
Read messages. Then press F4 to retry if appropriate, or F12 to cancel.	
DFHMQ0323 I CICSPAZ5 STOP received from TERMID=TCDD TRANID=CKSD USERID=CICSUSER.	

Figure 23-22 Stop Connection that is requested

15. Verify the results of the command by displaying the connection status in the CKQC display.

The QMgr name= field should be blank to indicate that the CICS region is not connected to a queue manager, as shown in Figure 23-23.

```
CKQCM2          Display Connection panel

Read connection information. Then press F12 to cancel.

CICS Applid = CICSPA5  Connection Status = Disconnected  QMgr name=
Mqname = IBM1      Tracing       = On                  API Exit = Off
Initiation Queue Name = IBM1.AOR.INITQ

----- STATISTICS -----
Number of in-flight tasks = 0      Total API calls      =
Number of running CKTI   = 0
APIs and flows analysis           Syncpoint           Recovery
----- -----
Run OK      0 MQINQ      0 Tasks      0 Indoubt     0
Futile      0 MQSET      0 Backout    0 UnResol     0
MQOPEN      0 ----- Flows ----- Commit      0 Commit      0
MQCLOSE     0 Calls       0 S-Phase    0 Backout     0
MQGET       0 SyncComp    0 2-Phase    0
GETWAIT     0 SuspReqd   0
MQPUT       0 Msg Wait   0
MQPUT1     0 Switched   0

F1=Help F12=Cancel Enter=Refresh
```

Figure 23-23 CKQC connection panel

16. Enter the command to stop the queue manager CSQ3.

In our test environment, the **-csq3 stop qmgr** command was entered from within the TSO SDSF panel.

Important: The format of the command is the subsystem command prefix followed by STOP QMGR. The subsystem command prefix is assigned as one of the steps when a queue manager is created.

Figure 23-24 shows the TSO SDSF job panel.

```
Display Filter View Print Options Search Help

-----
SDSF DA SC61      SC61      PAG 0 CPU/L/Z 2/ 2/ 0 LINE 1-4 (4)
COMMAND INPUT ===> /-csq3 stop qmgr                      SCROLL ===> CSR
NP   JOBNAME StepName ProcStep JobID   Owner    C Pos DP Real Paging   SIO
CSQ1MSTR CSQ1MSTR PROCSTEP STC06867 STC      NS  FE 14T  0.00  0.00
CSQ3MSTR CSQ3MSTR PROCSTEP STC05440 STC      NS  FE 15T  0.00  0.00
CSQ3CHIN CSQ3CHIN PROCSTEP STC05448 STC      IN   F6 7149  0.00  0.00
CSQ1CHIN CSQ1CHIN PROCSTEP STC06868 STC      IN   F6 5157  0.00  0.00

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=IFIND      F6=BOOK
F7=UP        F8=DOWN       F9=SWAP      F10=LEFT       F11=RIGHT     F12=RETRIEVE
```

Figure 23-24 The stop command was entered to stop the queue manager

Important: You might not have authority to issue the STOP QMGR command.

Figure 23-25 shows that indoubt units of work are created.

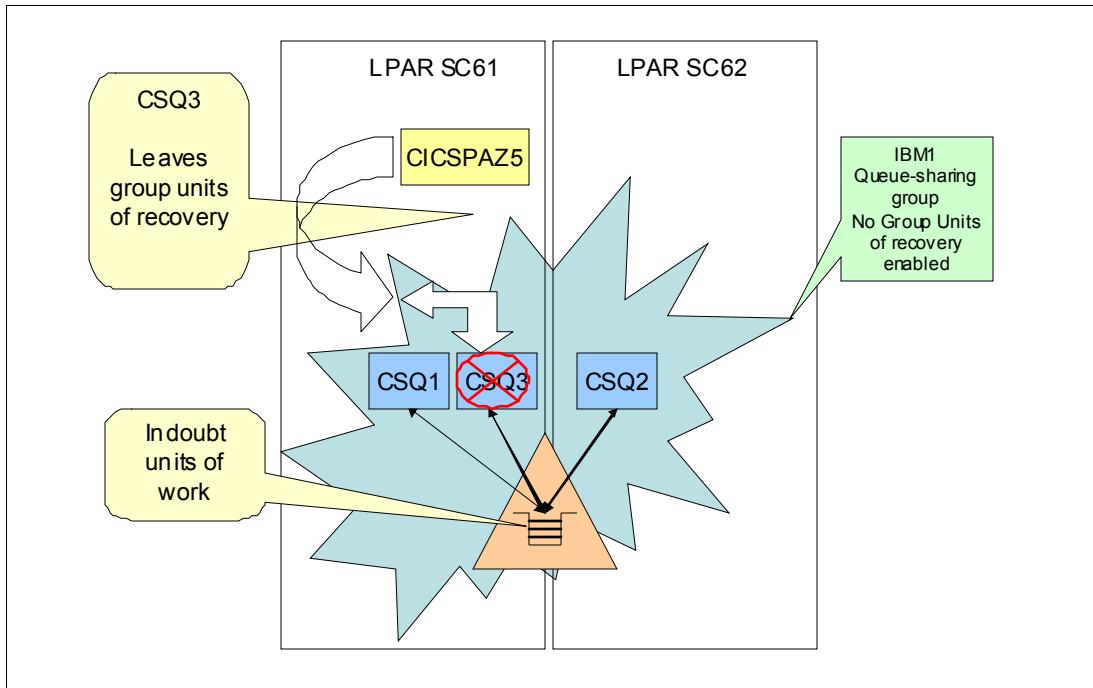


Figure 23-25 Indoubt units of work are created

17. Use the CIND transaction to allow the CICS region to back out the indoubt units of work, as shown in Figure 23-26.

CIND RESYNC BACKOUT

Figure 23-26 CIND command to resolve indoubt units of work

Figure 23-27 shows that the units of work that were held by CIND are now released by the tool.

```
DFHIN1007 10/17/2012 13:16:26 CICSPAZ5 TCDD CICSUSER Initiation of  
resynchronization for units of work awaiting coordinator DFHINDSP is now  
complete.
```

Figure 23-27 Results of CIND RESYNC BACKOUT command

18. On the CICS CKQC panel, press PF12 to go back and then select **1 START** to reconnect to queue manager, as shown in Figure 23-28.

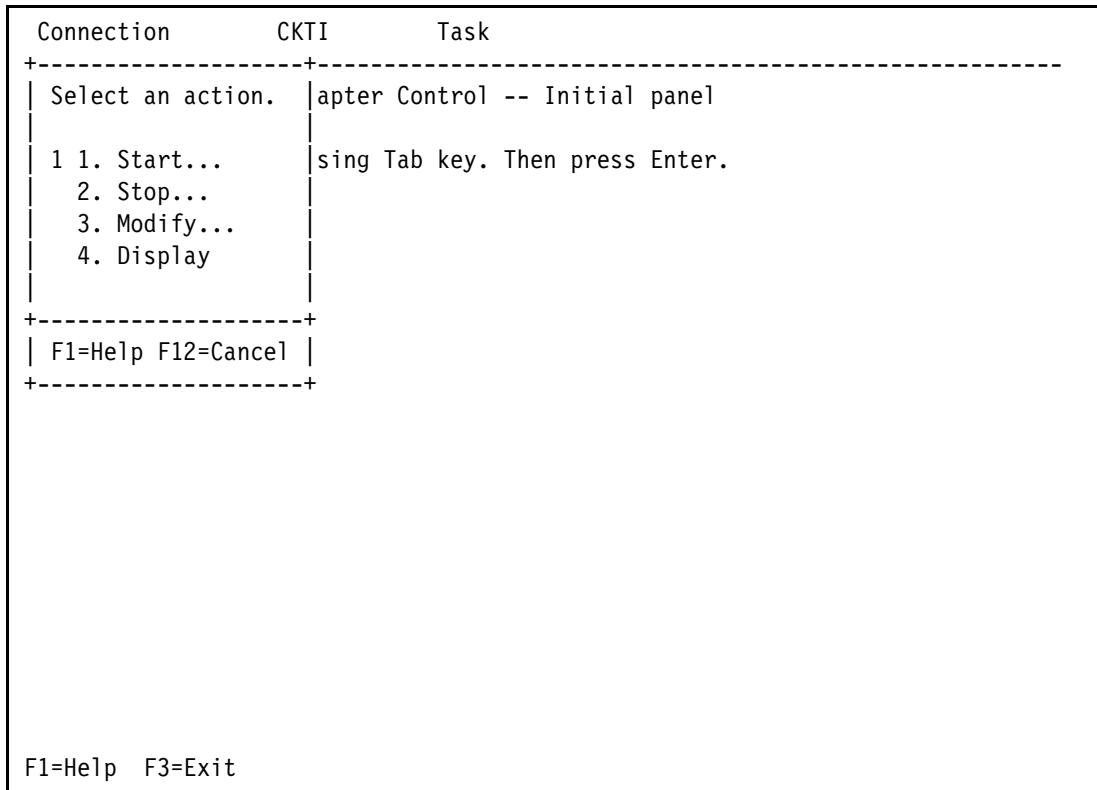


Figure 23-28 Select Start

19. Press Enter.

Figure 23-29 shows the parameters that are used to start the connection.

Connection	CKTI	Task
Select an action.		apter Control -- Initial panel
1 1. Start...		sing Tab key. Then press Enter.
2. Stop...		
3. Modify...		
4. Display		
		+-----+ Start a Connection +-----+
		Type parameters. Then press Enter.
F1=Help F12=Cancel		1. Queue manager name (SN) . . IBM1
		2. Initiation Queue Name (IQ)
		IBM1.AOR.INITQ
		+-----+ F1=Help F12=Cancel +-----+
		F1=Help F3=Exit

Figure 23-29 Start Connection panel

20.Press Enter.

Figure 23-30 shows the connection to queue manager CSQ1 and that there are some indoubt units of work that require synchronization.

CKQCM1	CICS adapter messages panel
Read messages. Then press F4 to retry if appropriate, or F12 to cancel.	
DFHMQ0323 I CICSPA25 CONNECT received from TERMID=TCDD TRANID=CKCN USERID=CICSUSER.	
DFHMQ0351 I CICSPA25 Unable to LOAD API exit CSQCAPX. Program is disabled.	
DFHMQ2064 CICSPA25 Resynchronization outstanding for queue manager CSQ3 after CICS-MQ group attach has connected to queue manager CSQ1.	
F4=Retry F12=Cancel	

Figure 23-30 Connection messages: Connection to CSQ1 and resynchronization outstanding to CSQ3

The resulting display shows a connection to CSQ1 and indoubt units of work for CSQ3.

21. Exit the CKQC displays by pressing the following keys:

- a. PF12
- b. PF12
- c. PF3

Figure 23-31 shows that the CICS region is now connected to the CSQ1 queue manager.

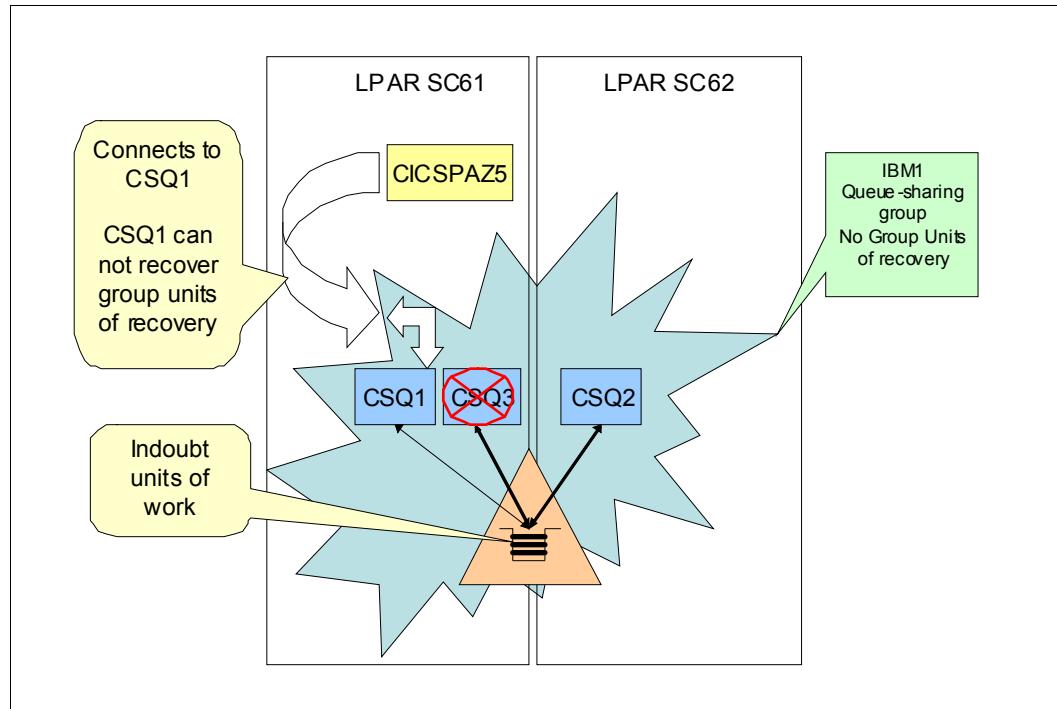


Figure 23-31 CICS region is connected to CSQ1 queue manager

22. Restart queue manager CSQ3.

Figure 23-32 shows that CSQ3 has indoubt units of work at start.

TIME	T	CON-ID	THREAD-XREF	S	URID
601	-	-	-	-	-
601	C	CICSPAZ5	2693A2A0D4E5D7E30000120C	D0000001088B8	2012-10-17
601	C	CICSPAZ5	2693A2A0D4E5D7E30000121C	D000000109C91	2012-10-17
601	C	CICSPAZ5	2693A2A0D4E5D7E30000122C	D00000010B086	2012-10-17
601	C	CICSPAZ5	2693A2A0D4E5D7E30000123C	D00000010C45F	2012-10-17
13.19.28	STC04475	CSQE140I	-CSQ3	CSQEENFR Started listening for ENF	35 602

Figure 23-32 At start CSQ3 shows indoubt units of work

23. Change CICS to connection from CSQ1 to CSQ3 by using the CKQC command.

24. Stop the current connection by using action 2, as shown in Figure 23-33.

Connection	CKTI	Task
Select an action.	apter Control -- Initial panel	
2 1. Start...	sing Tab key. Then press Enter.	
2. Stop...		
3. Modify...		
4. Display		
+-----+		
F1=Help F12=Cancel		
+-----+		
F1=Help F3=Exit		

Figure 23-33 CKQC panel

25. Stop the connection by selecting 1 Quiesce, as shown in Figure 23-34.

Connection	CKTI	Task
Select an action.	apter Control -- Initial panel	
2 1. Start...	sing Tab key. Then press Enter.	
2. Stop...		
3. Modify...	+-----+	
4. Display	Stop Connection	
+-----+		
F1=Help F12=Cancel		
+-----+		
Select stop type.		
Then press Enter.		
+-----+		
1 1. Quiesce		
2. Force		
+-----+		
F1=Help F12=Cancel		
+-----+		

Figure 23-34 Stop Connection

26.Press Enter.

Figure 23-35 shows the result of the stop request.

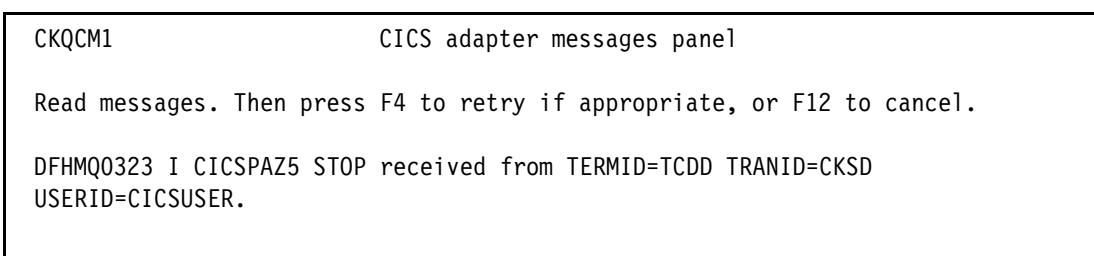


Figure 23-35 Results of stop connection

27.Press PF12 to go back and connect to a queue manager.

Figure 23-36 shows the Select an action menu.

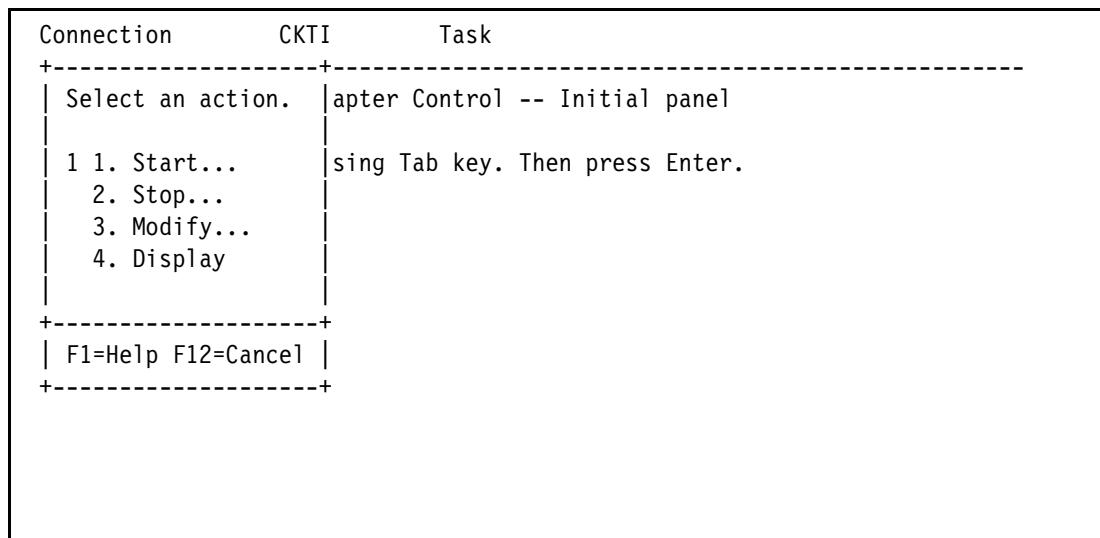


Figure 23-36 Start a queue manager connection

28.Select 1 and press Enter.

Figure 23-37 shows the parameters that are used to start the connection.

Connection	CKTI	Task
Select an action.		apter Control -- Initial panel
1 1. Start...		sing Tab key. Then press Enter.
2. Stop...		
3. Modify...		
4. Display		
		+-----+ Start a Connection +-----+
		Type parameters. Then press Enter.
F1=Help F12=Cancel		
		1. Queue manager name (SN) . . . IBM1
		2. Initiation Queue Name (IQ)
		IBM1.AOR.INITQ
		+-----+ F1=Help F12=Cancel +-----+

Figure 23-37 Start a Connection menu

29. Press Enter.

Figure 23-38 shows messages are displayed as a result of the connection to the queue manager.

CKQCM1	CICS adapter messages panel
	Read messages. Then press F4 to retry if appropriate, or F12 to cancel
	DFHMQ0323 I CICSPA25 CONNECT received from TERMID=TCDD TRANID=CKCN USERID=CICSUSER.
	DFHMQ0351 I CICSPA25 Unable to LOAD API exit CSQCAPX. Program is disabled
	DFHMQ0318 I CICSPA25 UOWID=CICSPA25.X'CA554980BB53AB2E' created by Transaction MVPT Taskid 123 is in doubt.
	DFHMQ0318 I CICSPA25 UOWID=CICSPA25.X'CA55497D2989592E' created by Transaction MVPT Taskid 122 is in doubt.
	DFHMQ0318 I CICSPA25 UOWID=CICSPA25.X'CA554979BA26E02E' created by Transaction MVPT Taskid 121 is in doubt.
	DFHMQ0318 I CICSPA25 UOWID=CICSPA25.X'CA55493DC100692C' created by Transaction MVPT Taskid 120 is in doubt.
	DFHMQ0307 I CICSPA25 Successful connection to queue manager CSQ3 released group IBM1

Figure 23-38 Connect to queue manager results

30. Review results of connection and indoubt units of work that needed recovery.

31. Display the queue manager connection. It connected back to the original queue manager.

Figure 23-39 shows the connection to CSQ3 queue manager.

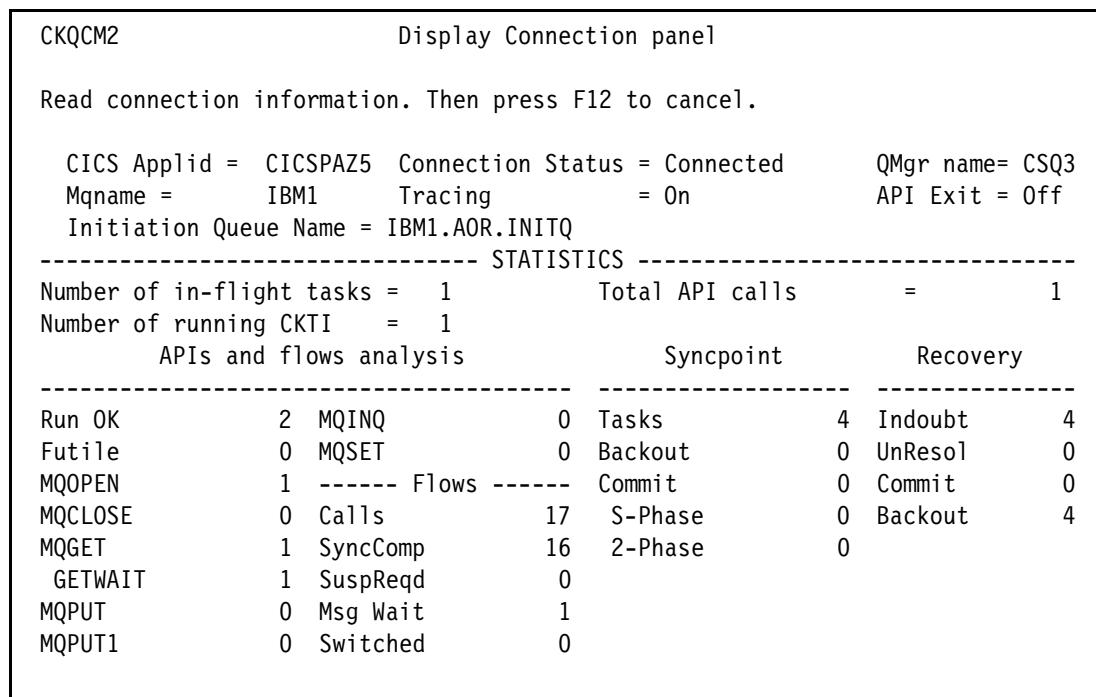


Figure 23-39 Connection display shows reconnected to CSQ3

Important: It might be necessary to stop queue manager CSQ1 to force the connection back to CSQ3.

32. Refresh the WebSphere MQ Explorer queue status display to show that the indoubt units of work were resolved.

Figure 23-40 shows the queue depths for the queue that had the uncommitted messages.

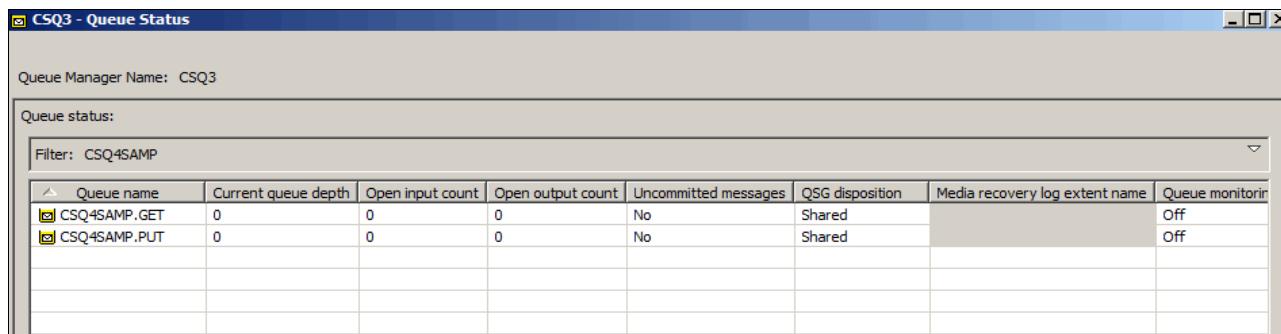


Figure 23-40 WebSphere MQ Explorer queue status display

The display shows that there are no more uncommitted messages.

23.3 Configuring the environment to support Group units of recovery

When Group units of recovery are enabled, the queue managers in the queue-sharing group should be enabled first.

The CICS region parameters can be changed for Group units of recovery after the queue-sharing group member to which it attaches is changed.

Important: The change to the CICS region affects only new connections. Any current connections are not affected by the change.

Enabling Group units of recovery helps eliminate errors that might result from changing the CICS region parameters before the connected queue manager is set up for Group units of recovery.

Complete the following steps to enable Group units of recovery for each queue manager:

1. In WebSphere MQ Explorer, right-click the queue manager name and select **Properties**, as shown in Figure 23-41.

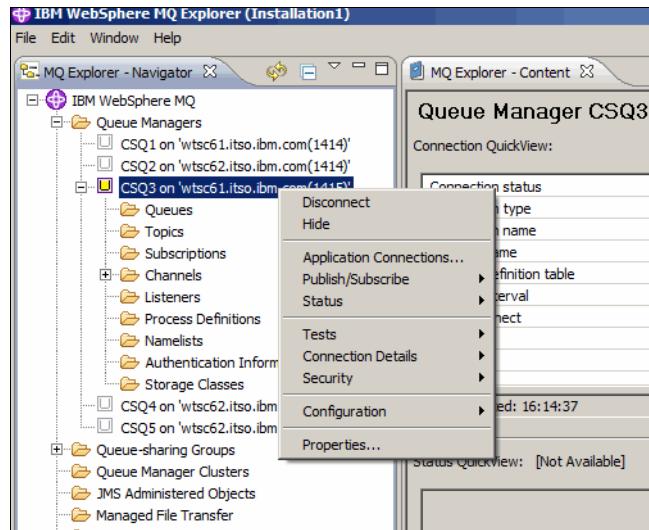


Figure 23-41 Select queue manager properties

2. Select the Extended panel and change the Groups units of recover to Enabled, as shown in Figure 23-42.

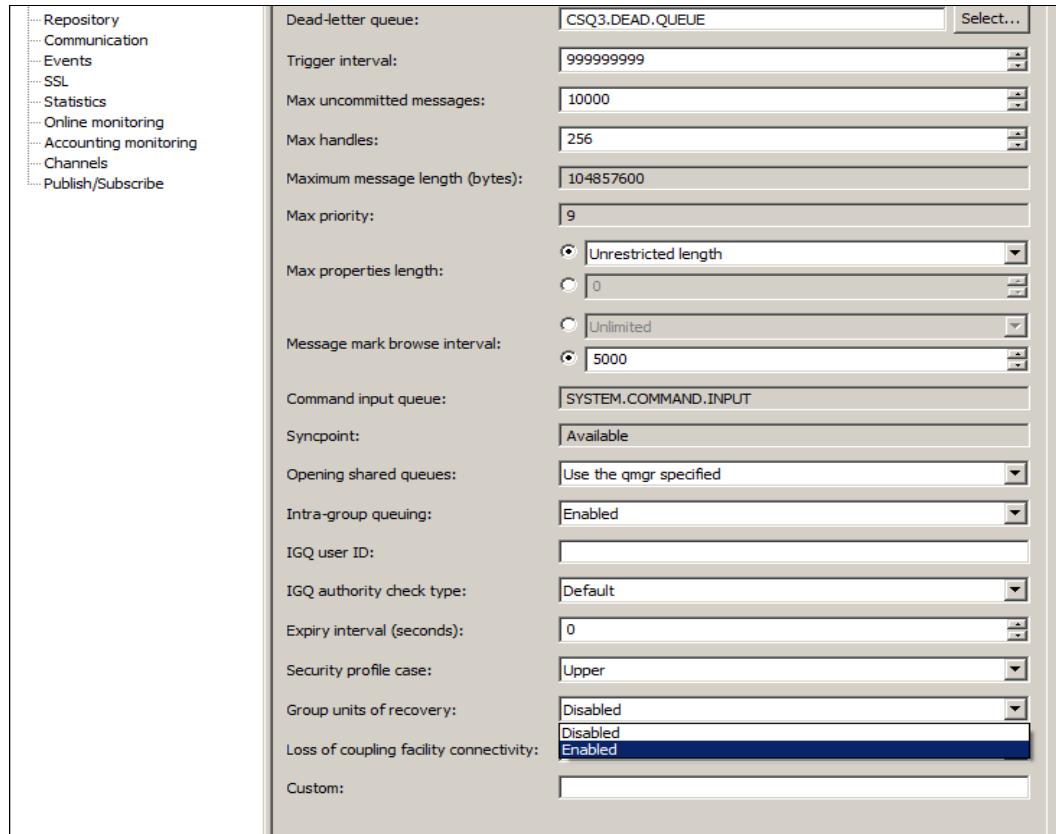


Figure 23-42 Change Groups of Recovery

3. Click **Apply**.
4. Check the JES log for the queue manager for the following message:
CSQM504I -qmgr CSQMCAMM GROUPUR support enabled
Repeat these steps for each queue manager member in the queue-sharing group for which you want to enable Group units of recovery.
5. Change each MQCONN definition in each CICS region in which you want to enable Group units of recovery.
6. Sign in to the CICS and alter the MQCONN definition to enable Group units of recovery.
7. Use CICS CEDA transaction to display the MQCONN definition from a blank window.
8. Press **Enter**.

Figure 23-43 shows the command to expand the resource definition for MQCONN.

```
CEDA EXPAND GROUP(WMQ) MQCONN
```

Figure 23-43 CICS CEDA command

Important: The group name of WMQ is where the MQCONN resolution was located. This location is different in other CICS environments.

9. Enter A to alter the definition, as shown in Figure 23-44.

```
ex GR(WMQ) MQCONN
ENTER COMMANDS
NAME      TYPE        GROUP          LAST CHANGE
IBM1     MQCONN      WMQ           10/16/12 13:06:00

SYSID=WRKS APPLID=CICSPA25
RESULTS: 1 TO 1 OF 1          TIME: 13.26.16   DATE: 10/17/12
F 1 HELP 2 SIG 3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 23-44 CEDA display that allows the alter to be entered

10. Change the following parameters:

- Mqname: queue-sharing group name
- Resyncmember to Groupresync

11. Press **Enter**.

Important: The Mqname must be changed to the queue-sharing group name. It cannot be a queue manager name.

12. You can verify that the ALTER was completed by confirming that the ALTER SUCCESSFUL message appears at the bottom of the panel, as shown in Figure 23-45.

```
OVERTYPE TO MODIFY CICS RELEASE = 0670
CEDA ALter MQconn( IBM1      )
MQconn      : IBM1
Group       : WMQ
DEScription ==> WMQ CONNECTION FOR QSG
Mqname     ==> IBM1
Resyncmember ==> Groupresync      Yes | No | Groupresync
Initqname   ==> IBM1.AOR.INITQ
DEFINITION SIGNATURE
DEFinetime   : 09/18/12 16:04:35
CHANGETime   : 10/17/12 13:27:49
CHANGEUsrid  : CICSUSER
CHANGEAGEnt   : CSDApi           CSDApi | CSDBatch
CHANGEAGRel   : 0670

SYSID=WRKS APPLID=CICSPA25
ALTER SUCCESSFUL TIME: 13.27.49 DATE: 10/17/12
PF 1 HELP 2 COM 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 23-45 Results ALTER successful

13. Press PF3 to exit.

Exiting the panel by using PF3 changes only the resource definition. The active connection in the CICS region is not changed.

Important: CICS SYSTEMS programmers normally make these CICS changes.

Use one of the following options to force the change of the connection between the CICS region and queue-sharing group:

- ▶ Recycle the CICS region with COLD start to enable the modified CICS MQCONN definition.
The CEDA alter that was done previously only altered the definition source. It did not install the changed definition.
- ▶ Use CEMT INQUIRE MQCONN to stop, change, and restart the connection, by completing the following steps:
 - a. Change Connectst to Notconnected.
 - b. Change Resyncmember to Groupresync.
 - c. Change Connectst to Connected.
- ▶ Use the **CQKC** command to stop the current connection by completing the following steps:
 - a. Use CEDA to install the new entry.
 - b. Use the **CKQC** command to start the connection

23.4 Enabling indoubt units of work after Group units of recovery are enabled

In this scenario, we show what happens when a CICS region includes WebSphere MQ indoubt units of work and the queue manager to which it is connected breaks the connection before they are resolved.

Important: This scenario is the same as the first scenario in this chapter. However, in this scenario, the Group units of recovery are enabled before the scenario is started

This scenario requires that the Group units of recovery are activated to CICS and the queue-sharing group to which it attaches.

Complete the following steps to show what happens when a CICS region features WebSphere MQ indoubt units of work and the queue manager to which it is connected breaks the connection before they are resolved:

1. Create indoubt units of work by using the CIND CICS transaction as in the first scenario.
2. Change the transaction class of the MVPT transaction to DFHTCIND.

Figure 23-46 shows CEMT transaction to inquire on the MVTP transaction.

```
cemt inquire transaction(mvpt)
```

Figure 23-46 Show transaction definition

3. Change the transaction class to DFHTCIND.

Example 23-47 shows that the Tcl was changed.

```
INQUIRE TRANSACTION(MVPT)
STATUS: RESULTS - OVERTYPE TO MODIFY
Tra(MVPT) Pri( 001 ) Pro(CSQ4CVK1) Tcl( DFHTCIND ) Ena Sta
           Prf(DFHCICST) Uda Any             Bac Wai

SYSID=WRKS
APPLID=CICSPAZ5
RESPONSE: NORMAL
TIME: 15.43.29 DATE: 10/17/12
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 23-47 Modify transaction class to DFHCIND

4. Press PF3 to exit.
5. Clear the panel and enter CIND ON (as shown in Figure 23-48) so that indoubt units of work can be created for the MVPT transaction.

```
cind on
```

Figure 23-48 Turn on CIND processing

6. Press Enter.

Figure 23-49 shows the tool is now active.

```
DFHIN1001 10/17/2012 12:50:12 CICSPAZ5 TCDD CICSUSER The indoubt tool is now
active for DFHTCIND tranclass transactions.
```

Figure 23-49 Results of CIND ON command that show the tool is active

The MVPT CICS transaction is initiated from a 3270 terminal by using the following comma-separated parameters:

- The number of messages to PUT
- The fill character
- Message size
- Persistence
- Queue name

Important: The following sample transaction and parameters were used to test whether the CIND transaction is functioning properly:

```
mvpt,1,* ,4096,p,CSQ4SAMP.PUT
```

The CSQ4SAMP.PUT queue is used by the MVPT CICS transaction and is the destination queue for the copied messages.

7. Initiate an MVPT transaction to PUT multiple a message to the CSQ4SAMP.PUT queue, as shown in Figure 23-50.

```
mvpt,1,* ,4096,p,CSQ4SAMP.PUT
```

Figure 23-50 Command that is entered into CICS

Figure 23-51 shows that an indoubt of work was created, the transaction that was abended, and the units of work are outstanding.

DFHAC2201 13:06:12 CICSPA5 Transaction MVPT has lost contact with its coordinator system during syncpoint processing and has abended with code ASP1. The unit of work is shunted until contact is restored.

Figure 23-51 Reply when transaction was entered

8. Put several messages in the QUEUE by using the MVPT transaction.

The WebSphere MQ Explorer display now shows messages in the queues. The CSQ4SAMP.PUT queue contains messages because of the CICS abending the MVPT transaction to create indoubt units of work.

Figure 23-52 shows the current queue depths.

Queue Manager Name: CSQ3						
Queue status:						
Filter: CSQ4SAMP						
Queue name	Current queue depth	Open input count	Open output count	Uncommitted messages	QSG disposition	Media recovery log extent name
CSQ4SAMP.GET	0	0	0	No	Shared	
CSQ4SAMP.PUT	4	0	0	Yes	Shared	

Figure 23-52 Queue display that shows message depths

9. Check the queue status to determine whether there are uncommitted messages. In this case, the status indicates that an unresolved unit of work exists.
10. Right-click the queue folder and select **Status**.

Figure 23-53 shows the menu that is used to select the status.

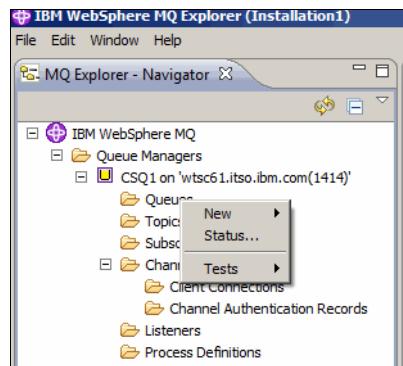


Figure 23-53 Select queue status

11. Scroll the window to show the uncommitted message count.

Figure 23-54 shows there are uncommitted messages.

Queue Manager Name: CSQ3						
Queue status:						
Filter: CSQ4SAMP						
Queue name	Current queue depth	Open input count	Open output count	Uncommitted messages	QSG disposition	Media recovery log extent name
CSQ4SAMP.GET	0	0	0	No	Shared	
CSQ4SAMP.PUT	4	0	0	Yes	Shared	

Figure 23-54 Show uncommitted messages

12. Change the queue manager to which the CICS region is connected to show that another manager can now resolve the indoubt units of work.

Important: In this test, the CSQ3 queue manager was stopped to show that CICS can connect to a different member of the queue-sharing group.

13. Stop queue manager CSQ3, as shown in Figure 23-55.

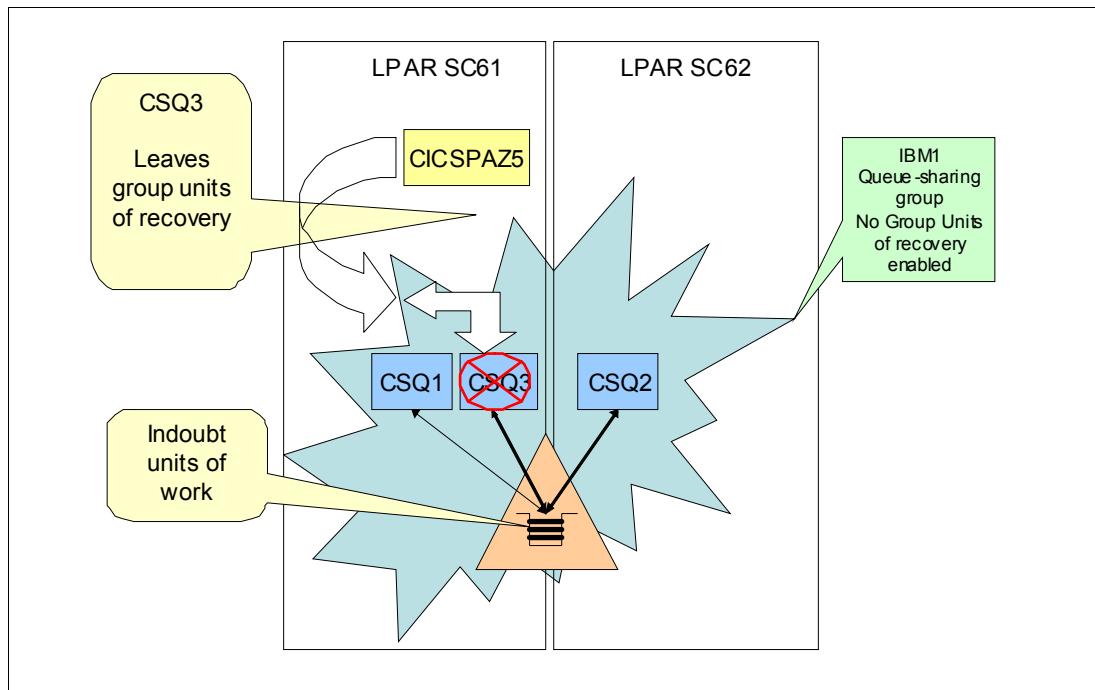


Figure 23-55 CSQ3 queue manager is stopped

14. Use CKQC to display the current queue manager connection.

Figure 23-56 on page 414 shows that the connection switched over to CSQ1 automatically.

Important: The connection automatically changes to another member in the queue-sharing group.

```

CKQCM2                               Display Connection panel

Read connection information. Then press F12 to cancel.

CICS Applid = CICSPAZ5  Connection Status = Connected      QMgr name= CSQ1
Mqname = IBM1      Tracing       = On                      API Exit = Off
Initiation Queue Name = IBM1.AOR.INITQ

----- STATISTICS -----
Number of in-flight tasks = 1          Total API calls = 0
Number of running CKTI   = 1
APIs and flows analysis           Syncpoint        Recovery
----- -----
Run OK      2 MQINQ      0 Tasks      1 Indoubt    8
Futile     0 MQSET      0 Backout    0 UnResol    1
MQOPEN     1 ----- Flows ----- Commit      0 Commit     0
MQCLOSE    0 Calls       7 S-Phase    0 Backout    0
MQGET      1 SyncComp    6 2-Phase    0
GETWAIT    1 SuspReqd   0
MQPUT      0 Msg Wait   1
MQPUT1    0 Switched   0

```

Figure 23-56 Automatically connection switched to a different queue manager

15. Use the CIND transaction to allow the CICS region to back out the indoubt units of work.

Figure 23-57 shows the CIND command to resolve the indoubt units of work by backing them out.

```
CIND RESYNC BACKOUT
```

Figure 23-57 CIND command to resolve indoubt units of work

Figure 23-58 shows the result of running the command.

```
DFHIN1007 10/17/2012 13:16:26 CICSPAZ5 TCDD CICSUSER Initiation of
resynchronization for units of work awaiting coordinator DFHINDSP is now
complete.
```

Figure 23-58 Results of CIND RESYNC BACKOUT command

CICS resolves the indoubt units of work.

Figure 23-59 shows that the indoubt units of work are resolved by CSQ1.

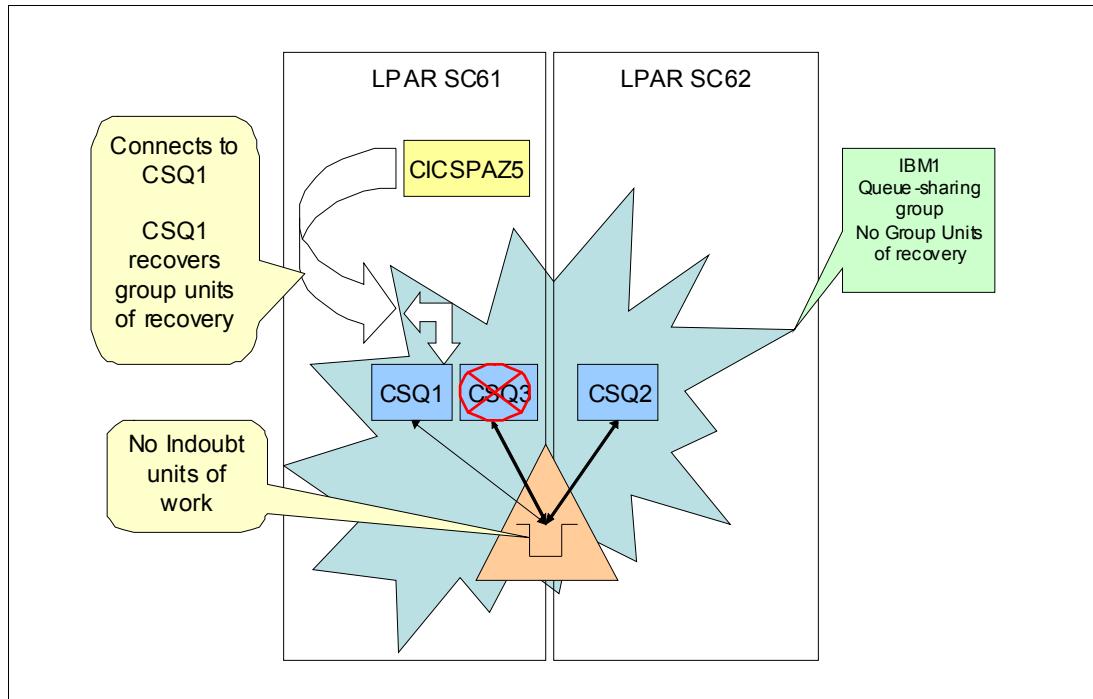


Figure 23-59 Automatic connection to a different queue manager and recovery indoubt units of work

CICS JES log shows the recovery of the indoubt messages.

Figure 23-60 shows the CSQ1 pJES log entries that CSQ1 performs the recovery.

CSQ3002I -CSQ1 INDOUBT RECOVERY BY CICSPAZ5 STILL IN PROGRESS

Figure 23-60 Message indication that indoubt units of work are being resolved

16. Check Queue status for uncommitted messages, as shown in Figure 23-61.

CSQ1 - Queue Status								
Queue Manager Name: CSQ1								
Queue status:								
Filter: CSQ4SAMP								
Queue name	Current queue depth	Open input count	Open output count	Uncommitted messages	QSG disposition	Media recovery log extent name	Queue monitorin	
CSQ4SAMP.GET	0	0	0	No	Shared		Off	
CSQ4SAMP.PUT	1	0	0	No	Shared		Off	

Figure 23-61 Shows that uncommitted messages are now committed

Figure 23-61 also shows that the previous uncommitted messages are now committed.

When indoubt units of work that were created by the loss of connection to the queue manager where CICS and queue-sharing group enabled group units of recover, the following events occur:

- CICS automatically connects to another member of the queue-sharing group if another member is available.
- The newly connected queue manager resolves the indoubt units of work for the disconnected queue manager.

Important: A quick test was performed by using a transaction that PUT several messages to a shared queue and then waited before the queue is closed. While the transaction was waiting, the queue manager was taken down and the CICS region connected to another member of the queue-sharing group. After the running transaction came out of the wait, the newly connected to queue manager completed the processing. The transaction did not abend because of the loss of the connection to the original queue manager.

17. Now start CSQ3 queue manager.

Messages in the CSQ3 JES logs show that the indoubt units of work were recovered by a different queue manager.

Figure 23-62 shows the messages that were issued by CSQ3 that shows the indoubt units of work were resolved by a different member of the queue-sharing group.

```
CSQM502I -CSQ3 CSQMCRRGG Processed BACKOUT request 332
from CSQ1 for indoubt UOW, URID=0000001517CE, CONNECTION
NAME=CICSPAZ5
CSQM502I -CSQ3 CSQMCRRGG Processed BACKOUT request 333
from CSQ1 for indoubt UOW, URID=00000015D0E6, CONNECTION
NAME=CICSPAZ5
```

Figure 23-62 CSQ3 JES log at start: Indoubt units of work that is recovered by a different queue manager



Resiliency: Improving availability

This chapter provides a working example with instructions on how to improve WebSphere MQ resiliency by changing parameters that are associated with loss of connectivity to coupling facility and coupling facility structures.

This chapter contains the following sections:

- ▶ Preparing the scenario
- ▶ Coupling facility application structure issues
- ▶ Coupling facility administration structure failure
- ▶ Loosing connectivity to coupling facility administration structure

24.1 Preparing the scenario

In this chapter, we describe scenarios in which WebSphere MQ members in a queue-sharing group have partial loss of connectivity to a coupling facility or one of its structures in a coupling facility.

As you review the information that is presented in this chapter, it is important to remember the following terms that relate to loss of connectivity:

- ▶ *Resiliency* refers to the ability of a WebSphere MQ queue manager in a queue sharing group to tolerate the loss of a coupling facility structure without terminating.
- ▶ *Partial loss of connectivity* is defined as the condition where connectivity is lost to a coupling facility structure or by some, but not all, logical partitions (LPARs) in the sysplex.
- ▶ *Total loss of connectivity* is defined as the condition in which all LPARs in the sysplex lose connectivity to a coupling facility.

The CFCONLOS attribute determines the queue manager action that is pursued when connectivity is lost to the administration or an application coupling facility structure that is defined with CFLEVEL(5).

The CFCONLOS attribute on the queue manager definition relates to the loss of connectivity to the administration structure.

The CFCONLOS attribute on the CFSTRUCT object relates to the loss of connectivity to that application structure.

The RECAUTO attribute on the CFSTRUCT object determines whether the **RECOVER CFSTRUCT** command should be run automatically for the application structure if it is necessary to restore access to shared queues in the corresponding application structure.

The **RESET CFSTRUCT ACTION(FAIL)** command can be used to manually generate a coupling facility structure failure or force a structure to be rebuilt by using the **RECOVER CFSTRUCT** command.

The following environment was used to test the scenarios:

- ▶ LPARs:
 - SC61
 - SC62
- ▶ WebSphere MQ queue-sharing group: IBM1
- ▶ WebSphere MQ queue-sharing group members:
 - CSQ1 - SC61
 - CSQ2 - SC62
 - CSQ3 - SC61
- ▶ Coupling facility:
 - CF01
 - CF04

24.2 Coupling facility application structure issues

This section describes the actions that were taken to create a loss of connectivity to coupling facility application structures condition.

It also shows the actions of the queue managers as a result of the different scenarios.

24.2.1 Partial loss of connectivity for one queue manager

In this section, connectivity to the N00FF application structure is interrupted on one LPAR by varying the path for the coupling facility offline to the SC62 LPAR.

This configuration causes a partial loss of connectivity, although the queue managers on SC61 retain connectivity.

Also, the CFSTRUCT N00FF application is defined in coupling facility CF01.

In this scenario, there is a single application structure in CF01 and CF01 is not used by any LPARs other than SC61 and SC62.

This configuration allows CF01 to be varied offline to SC62 and only the N00FF application structure are affected.

The following queues were used:

- ▶ CQS4SAMP.PERSISTENT: Contained persistent messages.
- ▶ CSQ4SAMP.NONPERSIST: Contained non-persistent messages.

Figure 24-1 shows the WebSphere MQ Explorer shared queue display.

Shared Queues					
Filter: CSQ4SAMP					
Queue name	Queue type	QSG disposition	Open input count	Open output count	Current queue depth
CSQ4SAMP.GET	Local	Shared	0	0	0
CSQ4SAMP.NONPERSIST	Local	Shared	0	0	1000
CSQ4SAMP.PERSISTENT	Local	Shared	0	0	1000
CSQ4SAMP.PUT	Local	Shared	0	0	0

Figure 24-1 WebSphere MQ Explorer showing queue depths at the start of the test

The N00FF CFSTRUCT CFCONLOS attribute was set to Tolerate to show that the queue managers do not abend when connectivity was lost.

Display the current N00FF coupling facility structure to show where it is defined. Figure 24-2 shows the command to display the coupling facility structure.

Structure name: The actual coupling facility structure name is the concatenation of the queue-sharing group name and CFSTRUCT name.

```
/D XCF,STR,STRNM=IBM1N00FF
```

Figure 24-2 Entering the xcf display to show current connectivity

The CFNAME description in the ACTIVE STRUCTURE section of the resulting display shows that it currently is defined in CF01.

Figure 24-3 shows the output from the display command.

```
D XCF,STR,STRNM=IBM1N0OFF
IXC360I 10.46.04 DISPLAY XCF 365
STRNAME: IBM1N0OFF
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE      : 512 M
  POLICY INITSIZE: 272144 K
  POLICY MINSIZE : 204108 K
  FULLTHRESHOLD  : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX          : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF01      CF04
  ENFORCEORDER    : NO
  EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/19/2012 10:17:51
CFNAME        : CF01
COUPLING FACILITY: 002817.IBM.02.0000000B3BD5
                  PARTITION: 32 CPCID: 00
ACTUAL SIZE      : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO       TOTAL     CHANGED   %
ENTRIES:         7038      78        1
ELEMENTS:        873273    312       0
EMCS:            94584     72        0
LOCKS:           1024
ACTUAL NOTIFYDELAY: 5000
PHYSICAL VERSION: CA57A758 1037862C
LOGICAL VERSION: CA528BD7 BA16522C
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME     : IXCL0058
DISPOSITION      : KEEP
ACCESS TIME      : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS   : 3

CONNECTION NAME ID VERSION SYSNAME  JOBNAM  ASID STATE
-----
CSQEIBM1CSQ101  01 0001000E SC61    CSQ1MSTR 0090 ACTIVE
CSQEIBM1CSQ202  02 00020008 SC62    CSQ2MSTR 0060 ACTIVE
CSQEIBM1CSQ303  03 0003000E SC61    CSQ3MSTR 0093 ACTIVE
```

Figure 24-3 NOOFF structure display

Figure 24-4 shows the test environment configuration.

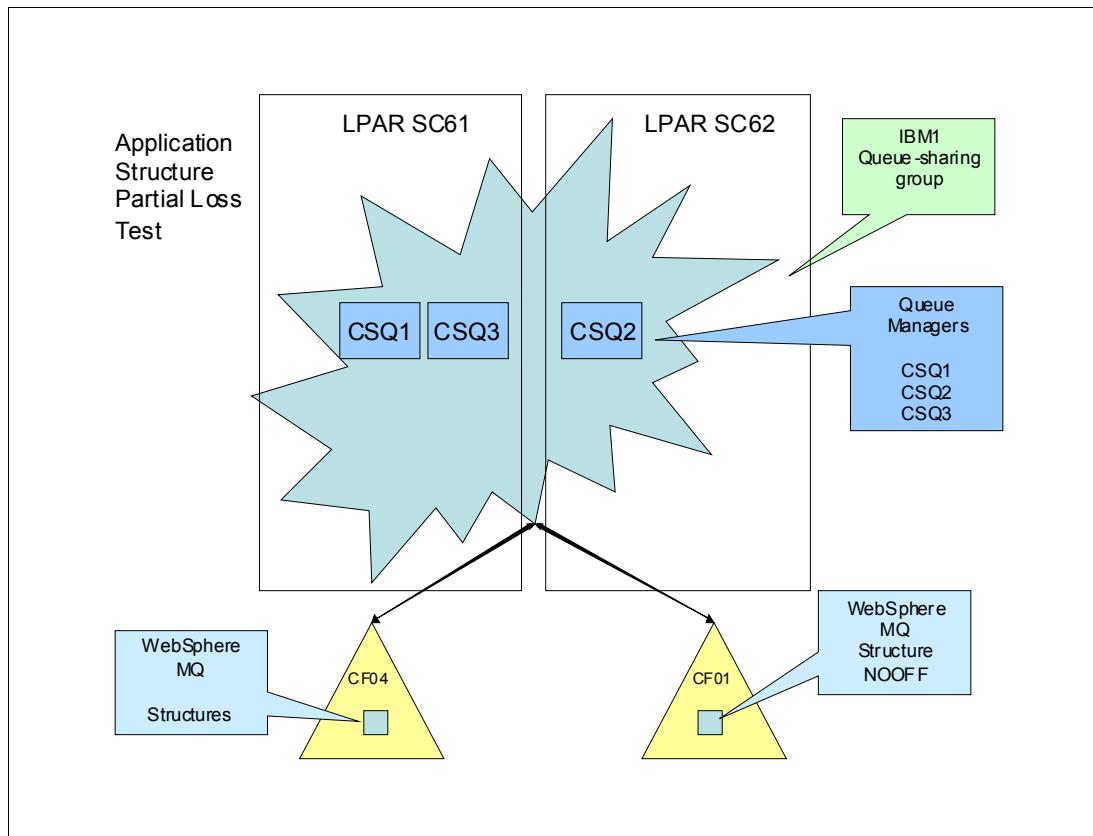


Figure 24-4 Configuration at start of the test

Coupling facility CF01 is varied offline to SC62 by the z/OS systems programmer.

Figure 24-5 shows that CF01 disconnects from the SC62 LPAR.

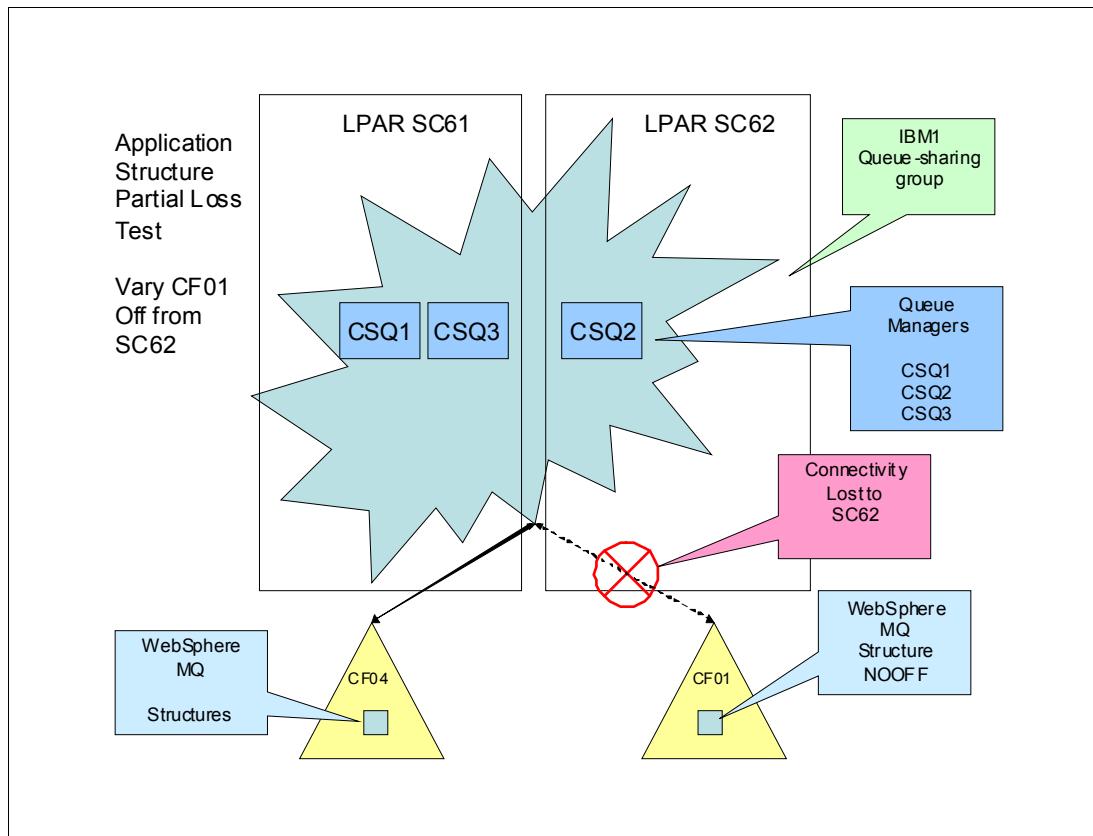


Figure 24-5 CF01 varied offline to SC62

The CSQ2 queue manager notifies the other members of the queue-sharing group to disconnect and asks for a rebuild of the application structure.

Figure 24-6 shows the messages from the CSQ2 JES log when the connection is lost.

10.50.46	STC03609	CSQE007I -CSQ2 CSQESTE EEPLOSSCONN event received for structure N00FF connection name CSQEIBM1CSQ202
10.50.46	STC03609	CSQE148I -CSQ2 CSQESTE Loss of connectivity processing for structure N00FF has been deferred
10.50.55	STC03609	CSQE007I -CSQ2 CSQESTE EEPLXESRECOMMENDACTION event received for structure N00FF connection name CSQEIBM1CSQ202
10.50.55	STC03609	CSQE143I -CSQ2 CSQESFDP Partial loss of connectivity reported for structure N00FF
10.50.55	STC03609	CSQE140I -CSQ2 CSQENFR Started listening for ENF events for structure N00FF
10.50.55	STC03609	CSQE006I -CSQ2 CSQECLOS Structure N00FF connection name CSQEIBM1CSQ202 disconnected
10.50.55	STC03609	CSQE149I -CSQ2 CSQERBLD Waiting for other queue managers to disconnect from structure N00FF
10.50.56	STC03609	CSQE144I -CSQ2 CSQERBLD System-managed rebuild initiated for structure N00FF
10.51.03	STC03609	IXL014I IXLCNN REQUEST FOR STRUCTURE IBM1N00FF WAS SUCCESSFUL. JOBNAME: CSQ2MSTR ASID: 0060
10.51.03	STC03609	CSQE141I -CSQ2 CSQENFR Stopped listening for ENF events for structure N00FF CONNECTOR NAME: CSQEIBM1CSQ202 CFNAME: CF04
10.51.03	STC03609	CSQE005I -CSQ2 CSQECONN Structure N00FF connected as CSQEIBM1CSQ202, version=CA57AEBDAD04992C 00020009
10.51.03	STC03609	CSQE007I -CSQ2 CSQESTE EEPLOSSFAILCONNECTION event received for structure N00FF connection name CSQEIBM1CSQ202
10.51.03	STC03609	CSQE011I -CSQ2 CSQESTE Recovery phase 1 started for structure N00FF connection name CSQEIBM1CSQ202
10.51.03	STC03609	CSQE013I -CSQ2 CSQERWI1 Recovery phase 1 completed for structure N00FF connection name CSQEIBM1CSQ202
10.51.03	STC03609	CSQE012I -CSQ2 CSQERWI2 Recovery phase 2 started for structure N00FF connection name CSQEIBM1CSQ202
10.51.03	STC03609	CSQE014I -CSQ2 CSQERWI2 Recovery phase 2 completed for structure N00FF connection name CSQEIBM1CSQ202
10.51.03	STC03609	CSQE006I -CSQ2 CSQECLOS Structure N00FF connection name CSQEIBM1CSQ202 disconnected
10.51.03	STC03609	CSQE140I -CSQ2 CSQENFR Started listening for ENF events for structure N00FF
10.51.04	STC03609	IXL014I IXLCNN REQUEST FOR STRUCTURE IBM1N00FF
10.51.04	STC03609	CSQE141I -CSQ2 CSQENFR Stopped listening for ENF events for structure N00FF WAS SUCCESSFUL. JOBNAME: CSQ2MSTR ASID: 0060 CONNECTOR NAME: CSQEIBM1CSQ202 CFNAME: CF04
10.51.04	STC03609	CSQE005I -CSQ2 CSQECONN Structure N00FF connected as CSQEIBM1CSQ202, version=CA57AEBDAD04992C 0002000A

Figure 24-6 CSQ2 JES log, which shows the sequence of events

CSQ2 CFCONLOS attributes are set to tolerate the loss of connectivity to the structure so it stays up and continues to service requests. Service requests for non-shared queues and shared queues that are not in structures in CF01 continue to be serviced.

The z/OS operating system performs a system-managed rebuild of the NOOFF application structure in CF04. A display of the IBM1NOOFF structure shows that IBM1NOOFF moved to CF04.

Figure 24-7 shows the command to display the coupling facility structure.

```
/D XCF,STR,STRNM=IBM1NOOFF
```

Figure 24-7 Display command for coupling facility structure IBM1NOOFF

Figure 24-8 shows the results of running the command.

```
D XCF,STR,STRNM=IBM1NOOFF
IXC360I 11.09.01 DISPLAY XCF 417
STRNAME: IBM1NOOFF
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE      : 512 M
  POLICY INITSIZE: 272144 K
  POLICY MINSIZE : 204108 K
  FULLTHRESHOLD  : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX          : DISABLED
  ALLOWREALLOCATE: YES
PREFERENCE LIST: CF01      CF04
  ENFORCEORDER    : NO
  EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/19/2012 10:50:57
CFNAME          : CF04
COUPLING FACILITY: 002097.IBM.02.00000001DE50
                      PARTITION: 0D CPCID: 00
ACTUAL SIZE        : 266 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO        TOTAL     CHANGED    %
  ENTRIES:       7148       2065      28
  ELEMENTS:     898838     36298      4
  EMCs:         94584       62        0
  LOCKS:         1024
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CA57AEBD AD04992C
LOGICAL VERSION: CA528BD7 BA16522C
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME       : IXCL0058
DISPOSITION       : KEEP
ACCESS TIME       : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS   : 3

CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME ASID STATE
-----
CSQEIBM1CSQ101  01 0001000E SC61    CSQ1MSTR 0090 ACTIVE
CSQEIBM1CSQ202  02 0002000A SC62    CSQ2MSTR 0060 ACTIVE
CSQEIBM1CSQ303  03 0003000E SC61    CSQ3MSTR 0093 ACTIVE
```

Figure 24-8 Display of NOOF shows that it was rebuilt in CF04

The members of the queue-sharing group delay servicing requests for queues in the NOOFF structure until the rebuild is complete.

The queue managers perform a peer recovery of CSQ2's connection to the NOOFF structure.

Figure 24-9 shows the CSQ1 JES during recovery.

```
10.50.55 STC04655 CSQE007I -CSQ1 CSQESTE EEPLDISCFAILCONNECTION event  
received for structure NOOFF connection name CSQEIBM1CSQ202  
10.50.55 STC04655 CSQE008I -CSQ1 CSQESTE Recovery event from CSQ2  
received for structure NOOFF  
10.50.55 STC04655 CSQE011I -CSQ1 CSQESTE Recovery phase 1 started for  
structure NOOFF connection name CSQEIBM1CSQ202  
10.50.55 STC04655 CSQE013I -CSQ1 CSQERWI1 Recovery phase 1 completed  
for structure NOOFF connection name CSQEIBM1CSQ202  
10.50.55 STC04655 CSQE008I -CSQ1 CSQESTE Recovery event from CSQ2  
received for structure NOOFF  
10.50.55 STC04655 CSQE012I -CSQ1 CSQERWI2 Recovery phase 2 started for  
structure NOOFF connection name CSQEIBM1CSQ202  
10.50.56 STC04655 CSQE014I -CSQ1 CSQERWI2 Recovery phase 2 completed  
for structure NOOFF connection name CSQEIBM1CSQ202
```

Figure 24-9 CSQ1 JES log showing the request to recover the structure

Figure 24-10 shows the coupling facility is rebuilt in CF04.

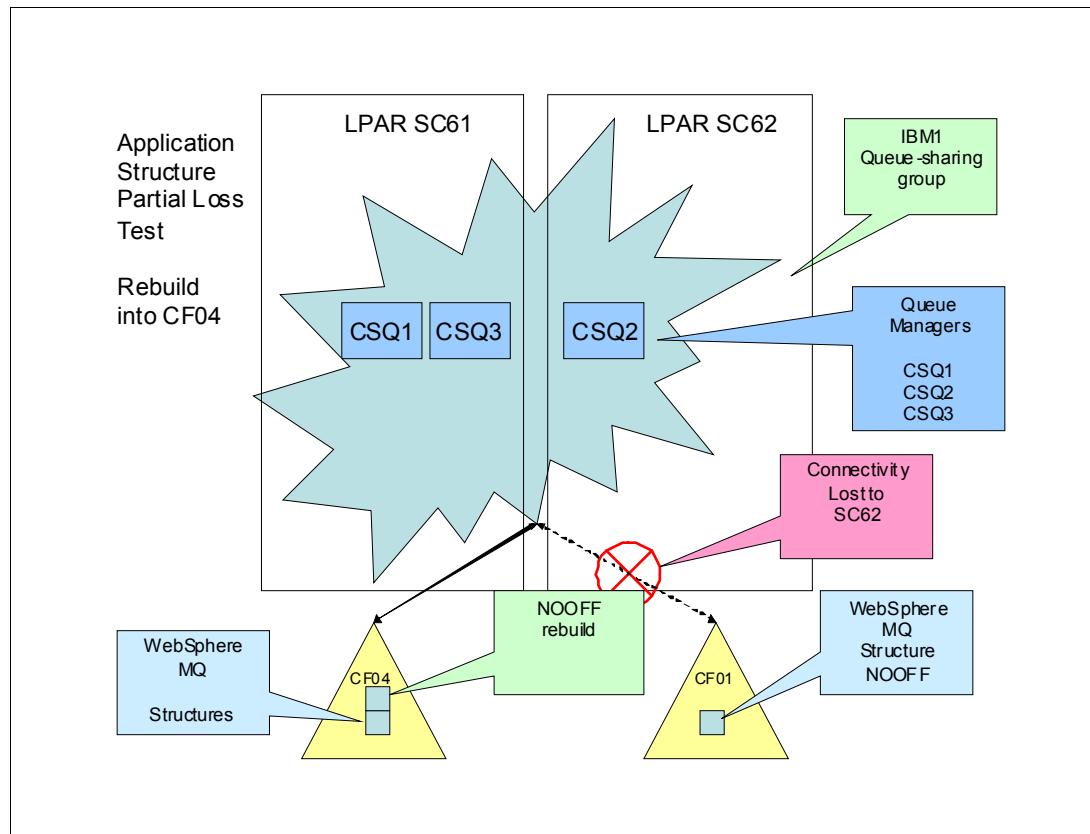


Figure 24-10 NOOFF structure that is rebuilt into CF04

The queue managers resume processing requests for queues in the NOOFF application structure.

Because some queue managers still had connectivity, the non-persistent and persistent messages were recovered, as shown in Figure 24-11, and the messages were retained.

Shared Queues						
Filter: CSQ4SAMP						
Queue name	Queue type	QSG disposition	Open input count	Open output count	Current queue depth	
CSQ4SAMP.GET	Local	Shared	0	0	0	
CSQ4SAMP.NONPERSIST	Local	Shared	0	0	1000	
CSQ4SAMP.PERSISTENT	Local	Shared	0	0	1000	
CSQ4SAMP.PUT	Local	Shared	0	0	0	

Figure 24-11 Current queue depths after the rebuild

24.2.2 Coupling facility structure failure

The test that is described in this section differs from the previous test in that all queue managers lose connectivity to the NOOFF structure.

The IBM sample program INJERROR is used to force an error in the NOOFF structure. The failure of the NOOFF structure is seen as a communications loss by all three queue managers.

Because the structure is failed, non-persistent messages are not available for the structure rebuild and recovery.

The INJERROR program is run to fail the IBM1NOOFF structure.

Figure 24-12 shows the output of the INJERROR program.

```
11:49:49.55 JOB05389 00000090 INJERROR: PROCESSING STARTED - VERSION 3.5 - 08/0
11:49:49.55 JOB05389 00000090 INJERROR: INPUT RECEIVED - STR=IBM1NOOFF      ,
STRTYPE=N/A
,
11:49:49.55 JOB05389 00000090 INJERROR: STR FAILURE INITIATED AGAINST STR=IBM1NOOFF
STRTYPE=N/A
11:49:49.55 JOB05389 00000090 INJERROR: PROCESSING COMPLETE
```

Figure 24-12 Output of the INJERROR program for IBM1NOOFF

Figure 24-13 shows the coupling facility structure failure.

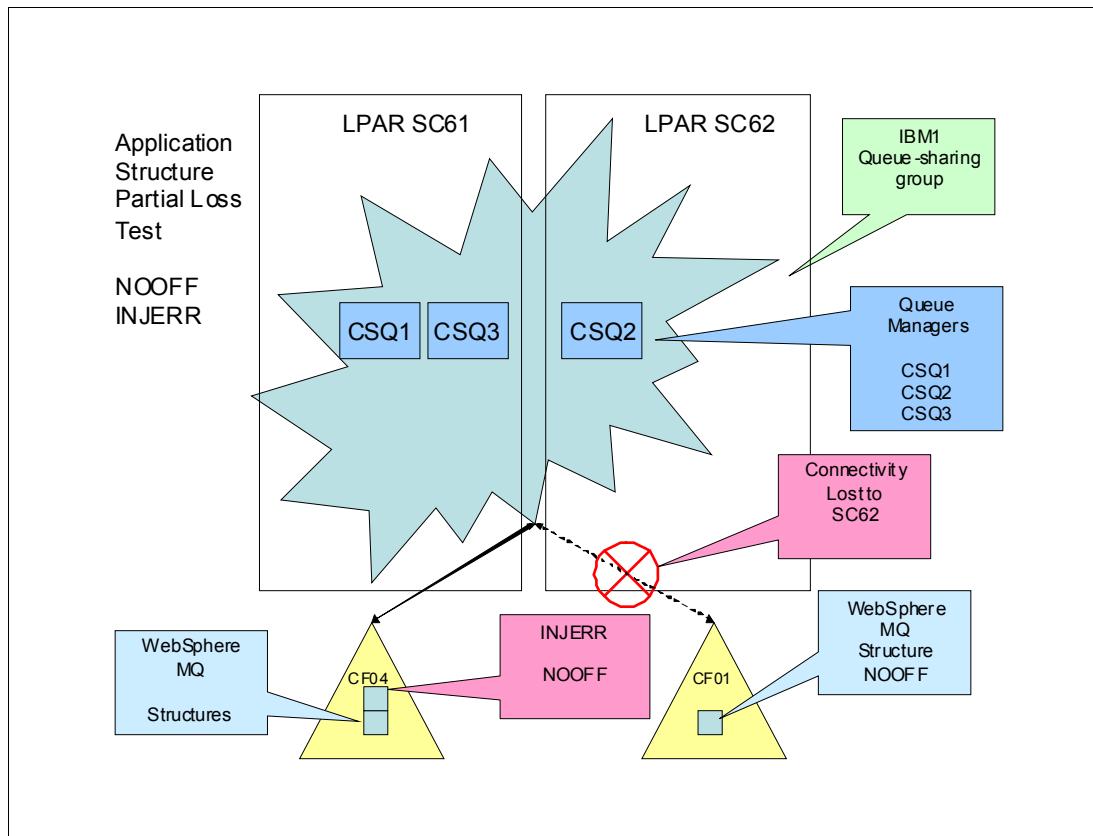


Figure 24-13 Structure failure

The queue managers receive notification of the structure failure, as shown in Figure 24-14.

```
11:50:00.29 STC04655 00000090 CSQE007I -CSQ1 CSQESTE EEPLSTRFAILURE event received  
00000090 for structure N0OFF connection name CSQEIBM1CSQ10  
11:50:00.29 STC04497 00000090 CSQE007I -CSQ3 CSQESTE EEPLSTRFAILURE event received  
00000090 for structure N0OFF connection name CSQEIBM1CSQ30  
11:50:00.39 STC03609 00000090 CSQE007I -CSQ2 CSQESTE EEPLSTRFAILURE event received  
00000090 for structure N0OFF connection name CSQEIBM1CSQ20
```

Figure 24-14 JES log messages that show the queue manager notification of the failure

A system-wide rebuild of the structure is requested by a queue manager. Because only CF04 is available, the N0OF structure is rebuilt in CF04. A RECOVER CFSTRUC command is initiated and the N0OFF structure is recovered.

Figure 24-15 shows that the structure was rebuilt in CF04.

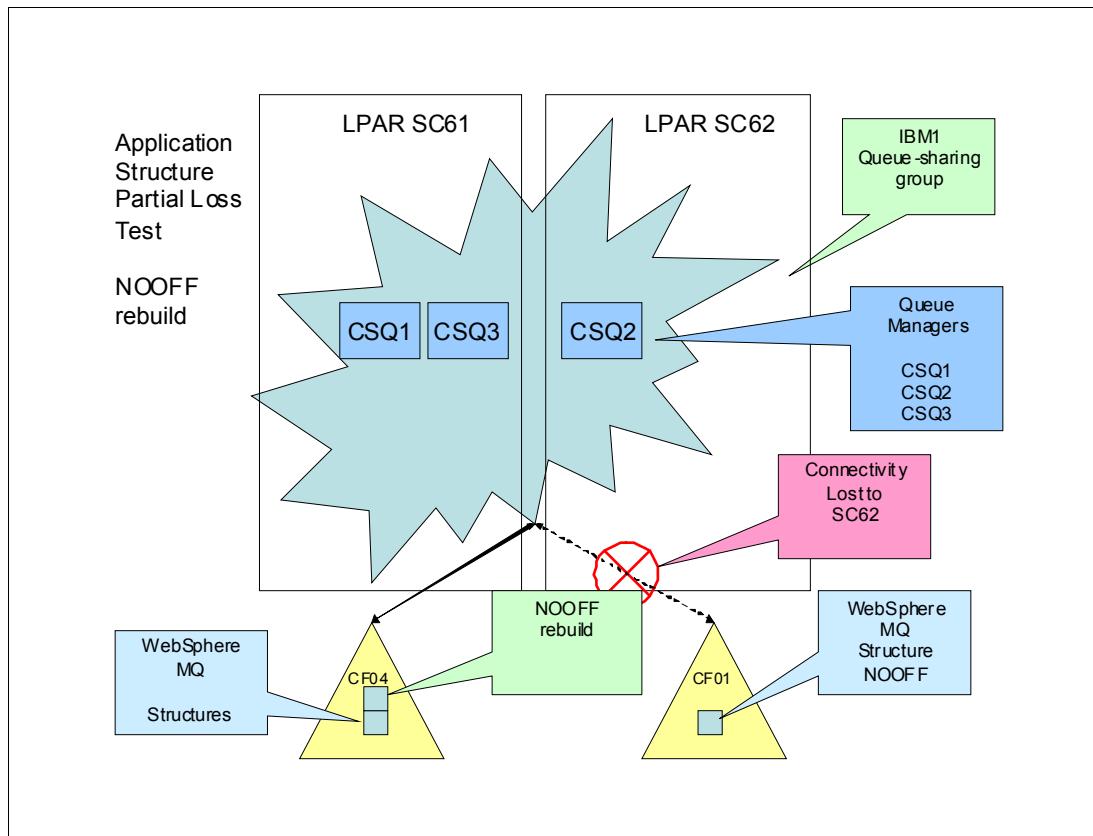


Figure 24-15 Structure that is rebuilt in same CF

The queue managers resume processing requests for the queue in the **NOOFF** application structure.

Because the INJERROR program caused the structure to fail, the non-persistent messages were not recovered.

INJERROR program: The INJERROR program causes a structure to fail and the data within it cannot be retrieved. Because the data is invalid, no messages can be retrieved and the queue managers use their logs for the recovery of messages. No non-persistent messages are contained in the log files.

Figure 24-16 shows that because of the structure failure, the non-persistent messages were not recovered.

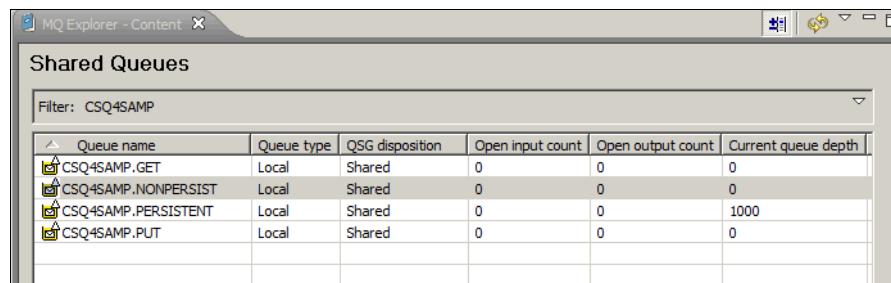


Figure 24-16 Display of the queues that shows the loss of the non-persistent messages

24.2.3 CFSTRUCT CFCONLOS attribute set to TERMINATE

For this test, the CFCONLOS attribute of the N00FF CFSTRUCT was set to terminate.

The test consisted of the following steps:

1. Coupling facility CF01 was varied online to SC61 and SC62.
2. The N00FF CFSTRUCT was rebuilt on CF01.
3. A recovery of the structure was performed and all queue managers connected.
4. The CRFM policy was changed so the N00FF structure did not include an alternative coupling facility where it can rebuild.
5. The previous test was rerun by using INJERROR to force the N00FF structure to fail.
6. Because the N00PT structure can be rebuilt in the same coupling facility, it was treated as a partial loss of connectivity.
7. A recovery of the structure was performed in CF01.
8. All queue managers are connected and resumed processing messages for queues in the N00FF application structure.

The CF01 coupling facility then was varied offline to SC62. Queue manager CSQ2 on SC62 terminated with REASON=00C510AB because of the following factors:

- ▶ Lost connectivity to the N00FF structure.
- ▶ The N00F CFSTRUCT CFCONLOS attribute was set to Terminate.

Figure 24-17 shows that the CSQ2 queue manager abended with a reason code of 00C510AB.

```
13.49.21 STC03609 CSQE007I -CSQ2 CSQESTE EEPLOSSCONN event received  
for structure N00FF connection name CSQEIBM1CSQ202  
13.49.21 STC03609 CSQE006I -CSQ2 CSQECLOS Structure CSQ_ADMIN  
connection name CSQEIBM1CSQ202 disconnected  
13.49.22 STC03609 *CSQV086E -CSQ2 QUEUE MANAGER ABNORMAL TERMINATION REASON=00C510AB  
13.49.22 STC03609 IEA794I SVC DUMP HAS CAPTURED:  
DUMPID=001 REQUESTED BY JOB (CSQ2MSTR)  
DUMP TITLE=CSQ2, QUEUE MANAGER TERMINATION REQUESTED, REASON=00  
2 C510AB  
13.49.36 STC03609 IEF450I CSQ2MSTR CSQ2MSTR - ABEND=S6C6 U0000 REASON=00C510AB  
TIME=13.49.36
```

Figure 24-17 JES log for CSQ2, which shows abend

Queue managers CSQ1 and CSQ3 on SC61 did not terminate because they still had connectivity to the N00FF structure.

CF01 was then varied offline to SC61.

CSQ1 and CSQ3 terminated with REASON=00C510AB because of the following factors:

- ▶ Connectivity to the N00FF structure was lost.
- ▶ The N00F CFCONLOS attribute was set to Terminate.

24.3 Coupling facility administration structure failure

In this scenario, the administration structure CSQ_ADMIN is made to fail by using the INJERROR program.

The administration structure CSQ_ADMIN does include alternative coupling facilities that are specified where it can be rebuilt.

Figure 24-18 shows the administration structure that is defined in CF04.

```
STRNAME: IBM1CSQ_ADMIN
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE    : 64 M
  POLICY INITSIZE: 45 M
  POLICY MINSIZE : 34560 K
  FULLTHRESHOLD : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX        : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF02      CF03      CF04
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/13/2012 16:24:08
CFNAME          : CF04
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
                  PARTITION: OF CPCID: 00
ACTUAL SIZE     : 45 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL    CHANGED   %
ENTRIES:        38151      14       0
ELEMENTS:       76410      179      0
EMCS:           7882       0       0
LOCKS:          256
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CA506E06 6E06AFA4
LOGICAL VERSION: CA506E06 6E06AFA4
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME     : IXCL01EF
DISPOSITION     : KEEP
ACCESS TIME     : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS   : 3

CONNECTION NAME ID VERSION SYSNAME  JOBNAME ASID STATE
-----
CSQEIBM1CSQ101  01 00010010 SC61      CSQ1MSTR 0090 ACTIVE
CSQEIBM1CSQ202  02 00020008 SC62      CSQ2MSTR 0060 ACTIVE
CSQEIBM1CSQ303  03 0003000C SC61      CSQ3MSTR 0093 ACTIVE
```

Figure 24-18 Display of the CSQ_ADMIN coupling facility structure

Figure 24-19 shows the start of the test.

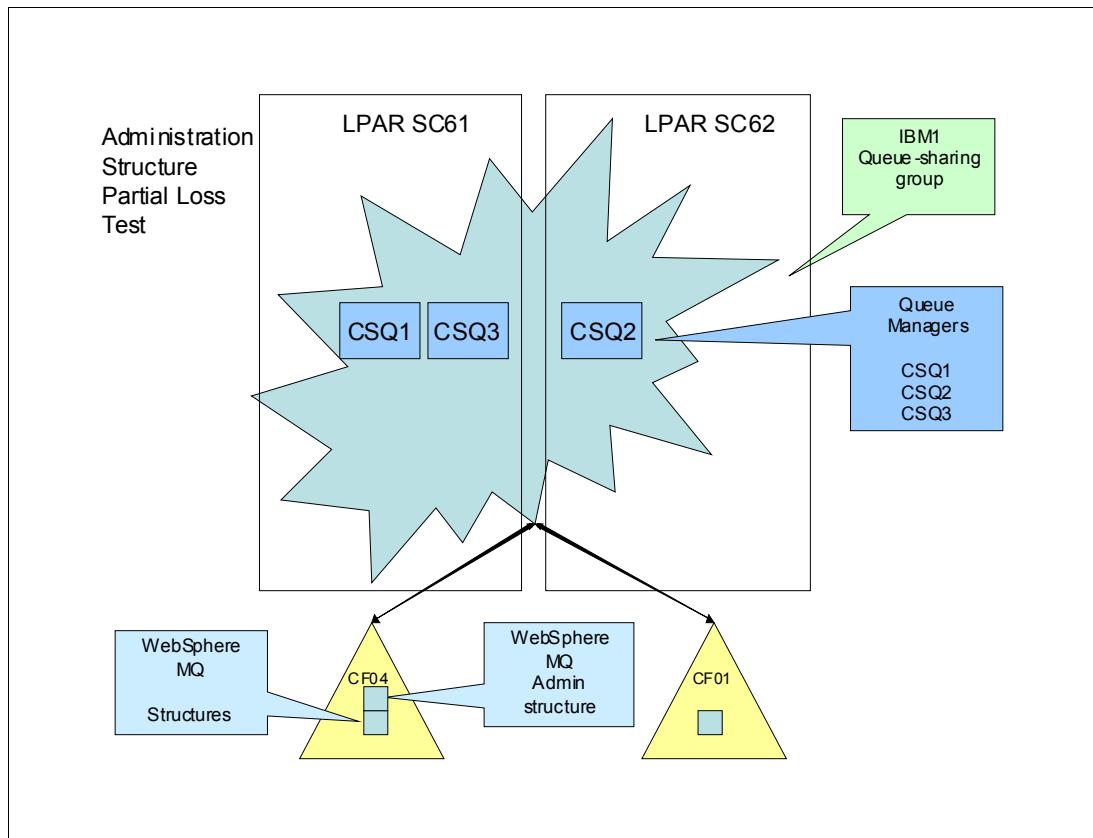


Figure 24-19 Start of test

The CFCONLOS attribute of the CSQ3 queue manager was set to terminate; the other attributes remain set to tolerate.

Figure 24-20 shows the WebSphere MQ Explorer extended page with CFCONLOS set to terminate for CSQ3.

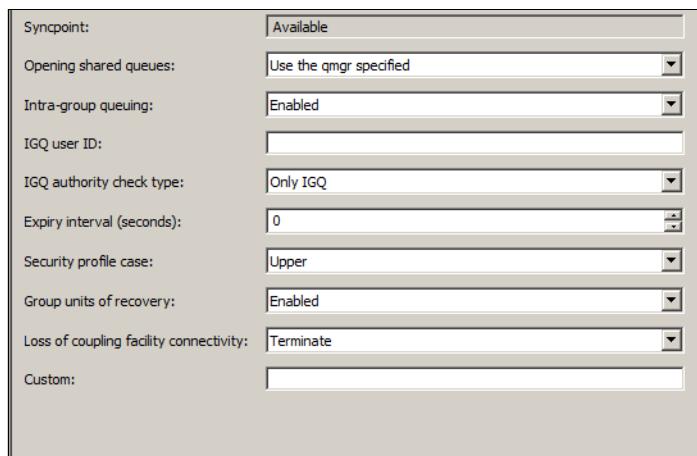


Figure 24-20 CFCONLOS set to terminate

The INJERROR program was run to fail the CSQ_ADMIN administration structure.

Figure 24-21 shows the output from the INJERROR program.

```
INJERROR: PROCESSING STARTED - VERSION 3.5 - 08/09/11
INJERROR: INPUT RECEIVED - STR=IBM1CSQ_ADMIN , STRTYPE=N/A
INJERROR: STR FAILURE INITIATED AGAINST STR=IBM1CSQ_ADMIN , STRTYPE=N/A
```

Figure 24-21 Results of the INJERROR program

The queue managers received notification of the structure failure.

Figure 24-22 shows the JES log when the coupling facility structure fails.

```
13.16.40 STC04655 CSQE007I -CSQ1 CSQESTE EEPLSTRFAILURE event received
772          for structure CSQ_ADMIN connection name CSQEIBM1CSQ101
13.16.40 STC04655 CSQE006I -CSQ1 CSQECLOS Structure CSQ_ADMIN 773
773          connection name CSQEIBM1CSQ101 disconnected
13.16.40 STC04655 CSQE140I -CSQ1 CSQEENFR Started listening for ENF 35 774
774          events for structure CSQ_ADMIN
```

Figure 24-22 Queue managers receive notification of failure

A rebuild of CSQ_ADMIN structure was requested, as shown in Figure 24-23, and the structure was rebuilt in the same coupling facility.

```
13.16.41 STC04655 IXL014I IXLCONN REQUEST FOR STRUCTURE IBM1CSQ_ADMIN
WAS SUCCESSFUL.  JOBNAME: CSQ1MSTR ASID: 0090
CONNECTOR NAME: CSQEIBM1CSQ101 CFNAME: CF02
```

Figure 24-23 Shows that the CSQ_ADMIN structure was rebuilt

The queue managers began rebuilding their entries for the CSQ_ADMIN administration structure.

Figure 24-24 shows the CSQ2 JES log of the status of rebuilding the administration structure.

```
13.16.41 STC03609 CSQE141I -CSQ2 CSQEENFR Stopped listening for ENF 35 743
    743      events for structure CSQ_ADMIN
    741      WAS SUCCESSFUL.  JOBNAME: CSQ2MSTR ASID: 0060
    741      CONNECTOR NAME: CSQEIBM1CSQ202 CFNAME: CF02
13.16.41 STC03609 IXL015I STRUCTURE ALLOCATION INFORMATION FOR 742
    742      STRUCTURE IBM1CSQ_ADMIN, CONNECTOR NAME CSQEIBM1CSQ202,
    742      CONNECTIVITY=DEFAULT
13.16.41 STC03609 CSQE005I -CSQ2 CSQECONN Structure CSQ_ADMIN connected 744
    742      CFNAME ALLOCATION STATUS/FAILURE REASON
    744      as CSQEIBM1CSQ202, version=CA57CF5062107F32 00010011
    742      -----
    742      CF02      STRUCTURE ALLOCATED AC001800
    742      CF03      PREFERRED CF ALREADY SELECTED AC001800
    742      PREFERRED CF HIGHER IN PREFLIST
    742      CF04      PREFERRED CF ALREADY SELECTED AC001800
    742      PREFERRED CF HIGHER IN PREFLIST
13.16.41 STC03609 CSQE031I -CSQ2 CSQECSR1 Admin structure data from CSQ1 incomplete
13.16.41 STC03609 CSQE031I -CSQ2 CSQECSR1 Admin structure data from CSQ3 incomplete
13.16.41 STC03609 CSQE018I -CSQ2 CSQERAD2 Admin structure data building started
13.16.41 STC03609 CSQE019I -CSQ2 CSQERAD2 Admin structure data building completed
13.16.41 STC03609 CSQE031I -CSQ2 CSQECSR1 Admin structure data from CSQ1 incomplete
13.16.41 STC03609 CSQE031I -CSQ2 CSQECSR1 Admin structure data from CSQ3 incomplete
```

Figure 24-24 CSQ1 JES log that shows rebuild and recovery activity

Queue managers complete recovery of their entries.

Figure 24-25 shows the CSQ1 JES log of rebuilding its portion of the administration structure.

```
13.16.41 STC03609 CSQE018I -CSQ1 CSQERAD2 Admin structure data building started
13.16.41 STC03609 CSQE019I -CSQ1 CSQERAD2 Admin structure data building completed
```

Figure 24-25 CSQ1 entries rebuild completed

Figure 24-26 shows the CSQ2 JES log of rebuilding its portion of the administration structure.

```
13.16.41 STC04497 CSQE018I -CSQ2 CSQERAD2 Admin structure data building started
13.16.41 STC04497 CSQE019I -CSQ2 CSQERAD2 Admin structure data building completed
```

Figure 24-26 CSQ2 entries rebuild completed

Figure 24-27 shows the CSQ3 JES log of rebuilding its portion of the administration structure.

```
13.16.41 STC04927 CSQE018I -CSQ3 CSQERAD2 Admin structure data building started
13.16.41 STC04927 CSQE019I -CSQ3 CSQERAD2 Admin structure data building completed
```

Figure 24-27 CSQ3 entries rebuild completed

The queue managers resume normal operation.

The result showed that the queue managers tolerated the loss of a failed administration structure.

Each queue manager completed updating the administration structure with its own entries.

24.4 Loosing connectivity to coupling facility administration structure

In this scenario, the connectivity to the coupling facility that contains the administration structure is lost.

The following sequence occurs:

1. The CFCONLOS attribute of the queue managers is set to terminate.
2. The coupling facility where the administration structure is defined (CF01) is varied offline to SC62. CSQ2 should abend because of the CFCONLOS attribute of the queue manager.
3. After the CSQ2 abends, CF01 is varied online to SC62 and CSQ2 is restarted. The recovery messages are monitored.
4. CF01 is varied offline from SC62 and SC61. Queue managers CSQ1, CSQ2, and CSQ3 should abend because of the CFCONLOS attribute of the queue manager.
5. The administration structure is altered to allow it to rebuild in an alternative coupling facility.
6. The queue managers are restarted to show the recovery of the administration structure.

Figure 24-28 shows that the administration structure is defined in CF01.

```
STRNAME: IBM1CSQ_ADMIN
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE    : 64 M
  POLICY INITSIZE: 45 M
  POLICY MINSIZE : 34560 K
  FULLTHRESHOLD : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX        : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF01
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/13/2012 16:24:08
CFNAME      : CF01
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
                  PARTITION: OF CPCID: 00
ACTUAL SIZE   : 45 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO     TOTAL     CHANGED   %
ENTRIES:       38151      14        0
ELEMENTS:      76410      179       0
EMCS:          7882       0         0
LOCKS:         256
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CA506E06 6E06AFA4
LOGICAL VERSION: CA506E06 6E06AFA4
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME    : IXCL01EF
DISPOSITION    : KEEP
ACCESS TIME    : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS  : 3

CONNECTION NAME ID VERSION SYSNAME  JOBNAME ASID STATE
-----
CSQEIBM1CSQ101 01 00010010 SC61    CSQ1MSTR 0090 ACTIVE
CSQEIBM1CSQ202 02 00020008 SC62    CSQ2MSTR 0060 ACTIVE
CSQEIBM1CSQ303 03 0003000C SC61    CSQ3MSTR 0093 ACTIVE
```

Figure 24-28 Display of the CSQ_ADMIN coupling facility structure

The CFCONLOS attribute of the queue managers was set to terminate by using the TSO iSPF WebSphere MQ management panels.

Figure 24-29 shows the CFCONLOS attribute setting for CSQ1 by using the TSO ISPF MQ Administration panels. In the panels, the value is identified as CF connectivity loss action.

```
Queue manager name . . . . : CSQ1

Monitoring
  Queues . . . . . 0 N=None, L=Low, M=Medium, H=High, 0=Off
  Channels . . . . . 0 N=None, L=Low, M=Medium, H=High, 0=Off
  Auto-defined
    cluster-senders . . . . Q Q=Qmgr, L=Low, M=Medium, H=High, 0=Off
  Queue accounting . . . . E E=Enabled, D=Disabled, N=None

Trace route recording . . . . M M=Message, Q=Queue, D=Disabled
Activity recording . . . . M M=Message, Q=Queue, D=Disabled

Shared queue puts . . . . U U=Use queue manager name, I=Ignore
CF connectivity loss action E E=Terminate, T=Tolerate
```

Figure 24-29 Display of the CSQ1 CFCONLOS attribute

Figure 24-30 shows the CFCONLOS attribute setting for CSQ2.

```
Queue manager name . . . . : CSQ2

Monitoring
  Queues . . . . . 0 N=None, L=Low, M=Medium, H=High, 0=Off
  Channels . . . . . 0 N=None, L=Low, M=Medium, H=High, 0=Off
  Auto-defined
    cluster-senders . . . . Q Q=Qmgr, L=Low, M=Medium, H=High, 0=Off
  Queue accounting . . . . E E=Enabled, D=Disabled, N=None

Trace route recording . . . . M M=Message, Q=Queue, D=Disabled
Activity recording . . . . M M=Message, Q=Queue, D=Disabled

Shared queue puts . . . . U U=Use queue manager name, I=Ignore
CF connectivity loss action E E=Terminate, T=Tolerate
```

Figure 24-30 Display of the CSQ2 CFCONLOS attribute

Figure 24-31 shows the CFCONLOS attribute setting for CSQ3.

```
Queue manager name . . . . : CSQ3

Monitoring
  Queues . . . . . 0 N=None, L=Low, M=Medium, H=High, 0=Off
  Channels . . . . . 0 N=None, L=Low, M=Medium, H=High, 0=Off
  Auto-defined
    cluster-senders . . . . Q Q=Qmgr, L=Low, M=Medium, H=High, 0=Off
  Queue accounting . . . . E E=Enabled, D=Disabled, N=None

Trace route recording . . . . M M=Message, Q=Queue, D=Disabled
Activity recording . . . . M M=Message, Q=Queue, D=Disabled

Shared queue puts . . . . U U=Use queue manager name, I=Ignore
CF connectivity loss action E E=Terminate, T=Tolerate
```

Figure 24-31 Display of the CSQ3 CFCONLOS attribute

Coupling facility CF01 was varied offline to LPAR SC62, which contains the CSQ3 queue manager.

The CF03 queue manager received notification of the loss of connectivity and abended with 00C510AB because of the CFCONLOS attribute setting.

Figure 24-32 shows the CSQ3 JES log messages that were generated.

```
12.23.35 STC16223 CSQM508E -CSQ2 CSQMCRRG GROUPUR agent ended abnormally. Restating
12.33.25 STC16223 CSQY220I -CSQ2 CSQSCTL Queue manager storage usage: 341
            local storage: used 339MB, free 1115MB: above bar: used 329MB, free
            341
            1GB
12.33.38 STC16223 CSQE007I -CSQ2 CSQESTE EEPLOSSCONN event received
            359
            for structure CSQ_ADMIN connection name CSQEIBM1CSQ202
12.33.38 STC16223 CSQE006I -CSQ2 CSQECLOS Structure CSQ_ADMIN
            367
            connection name CSQEIBM1CSQ202 disconnected
12.33.38 STC16223 *CSQV086E -CSQ2   QUEUE MANAGER ABNORMAL TERMINATION REASON=00C510AB
12.33.38 STC16223 IEA794I SVC DUMP HAS CAPTURED: 410
            410
            DUMPID=001 REQUESTED BY JOB (CSQ2MSTR)
            410
            DUMP TITLE=CSQ2, QUEUE MANAGER TERMINATION REQUESTED, REASON=00
            410
            C510AB
12.33.51 STC16223 IEF450I CSQ2MSTR CSQ2MSTR - ABEND=S6C6 U0000 REASON=00C510AB
```

Figure 24-32 Display of CSQ3 JES log

Queue managers CSQ1 and CSQ2 continued to process messages because SC61 still was connected to CF01.

CF01 was varied back online to SC62 and CSQ2 queue manager was restarted.

Figure 24-33 shows the CSQ3 JES log messages that were generated as it reconnected to the administration structure during startup.

```
12.37.19 STC19705 IXL014I IXLCNN REQUEST FOR STRUCTURE IBM1CSQ_ADMIN
12.37.19 STC19705 CSQE141I -CSQ2 CSQEENFR Stopped listening for ENF 35
    677 events for structure CSQ_ADMIN
    676 WAS SUCCESSFUL.  JOBNAME: CSQ2MSTR ASID: 004E
    676 CONNECTOR NAME: CSQEIBM1CSQ202 CFNAME: CF01
12.37.19 STC19705 CSQE005I -CSQ2 CSQECONN Structure CSQ_ADMIN connected
    678 as CSQEIBM1CSQ202, version=CA878F5E8FD54968 00020010
    676 ADDITIONAL STATUS INFORMATION:
12.37.19 STC19705 CSQE021I -CSQ2 CSQECONN Structure CSQ_ADMIN 679
    676 CONNECTION HAS BEEN REESTABLISHED
    679 connection as CSQEIBM1CSQ202 warning, RC=00000004 reason=0201
    679 codes=00000000 00000000 00000000
```

Figure 24-33 Display of CSQ3 JES log

CF01 was varied offline to SC61 and SC62 and queue managers CSQ1, CSQ2, and CSQ3 abended.

Figure 24-34 shows the CSQ1 JES log messages that were generated.

```
12.41.14 STC16096 CSQE007I -CSQ1 CSQESTE EEPLOSSCONN event received
    682 for structure CSQ_ADMIN connection name CSQEIBM1CSQ101
12.41.15 STC16096 CSQE006I -CSQ1 CSQECLOS Structure CSQ_ADMIN
    685 connection name CSQEIBM1CSQ101 disconnected
12.41.15 STC16096 *CSQV086E -CSQ1 QUEUE MANAGER ABNORMAL TERMINATION REASON=00C510AB
12.41.27 STC16096 IEF450I CSQ1MSTR CSQ1MSTR - ABEND=S6C6 U0000 REASON=00C510AB
    096 TIME=12.41.27
```

Figure 24-34 Display of CSQ1 JES log

Figure 24-35 shows the CSQ2 JES log messages that were generated.

```
12.23.35 STC16223 CSQM508E -CSQ2 CSQMCRRG GROUPUR agent ended abnormally. Restating
12.33.25 STC16223 CSQY220I -CSQ2 CSQSCTL Queue manager storage usage: 341
    341 local storage: used 339MB, free 1115MB: above bar: used 329MB, free
    341 1GB
12.33.38 STC16223 CSQE007I -CSQ2 CSQESTE EEPLOSSCONN event received
    359 for structure CSQ_ADMIN connection name CSQEIBM1CSQ202
12.33.38 STC16223 CSQE006I -CSQ2 CSQECLOS Structure CSQ_ADMIN
    367 connection name CSQEIBM1CSQ202 disconnected
12.33.38 STC16223 *CSQV086E -CSQ2 QUEUE MANAGER ABNORMAL TERMINATION REASON=00C510AB
12.33.38 STC16223 IEA794I SVC DUMP HAS CAPTURED: 410
    410 DUMPID=001 REQUESTED BY JOB (CSQ2MSTR)
    410 DUMP TITLE=CSQ2, QUEUE MANAGER TERMINATION REQUESTED, REASON=00
    410 C510AB
12.33.51 STC16223 IEF450I CSQ2MSTR CSQ2MSTR - ABEND=S6C6 U0000 REASON=00C510AB
    583 TIME=12.33.51
```

Figure 24-35 Display of CSQ2 JES log

Figure 24-36 shows the CSQ3 JES log messages that were generated.

```
12.41.14 STC19581 CSQE007I -CSQ3 CSQESTE EEPLOSSCONN event received  
681      for structure CSQ_ADMIN connection name CSQEIBM1CSQ303  
12.41.14 STC19581 CSQE006I -CSQ3 CSQECLOS Structure CSQ_ADMIN  
683      connection name CSQEIBM1CSQ303 disconnected  
12.41.15 STC19581 *CSQV086E -CSQ3   QUEUE MANAGER ABNORMAL TERMINATION REASON=00C510AB  
12.41.15 STC19581 IEA794I SVC DUMP HAS CAPTURED:  
792      DUMPID=001 REQUESTED BY JOB (CSQ3MSTR)  
792      DUMP TITLE=CSQ3, QUEUE MANAGER TERMINATION REQUESTED, REASON=  
792      C510AB  
12.41.28 STC19581 IEF450I CSQ3MSTR CSQ3MSTR - ABEND=S6C6 U0000 REASON=00C510AB
```

Figure 24-36 Display of CSQ3 JES log

The administration structure was altered to allow the administration structure to rebuild in another coupling facility.

Figure 24-37 shows the administration structure preference list, which contains multiple coupling facilities.

```
STRNAME: IBM1CSQ_ADMIN
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
  POLICY SIZE    : 64 M
  POLICY INITSIZE: 45 M
  POLICY MINSIZE : 34560 K
  FULLTHRESHOLD : 85
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: 5
  DUPLEX         : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF02      CF03      CF01
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
-----
ALLOCATION TIME: 10/13/2012 16:30:27
CFNAME          : CF02
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
                  PARTITION: OF CPCID: 00
ACTUAL SIZE     : 45 M
STORAGE INCREMENT SIZE: 1 M
USAGE INFO      TOTAL    CHANGED %
ENTRIES:        38151      14      0
ELEMENTS:       76410      179     0
EMCS:           7882       0      0
LOCKS:          256
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CA506E06 6E06AFA4
LOGICAL VERSION: CA506E06 6E06AFA4
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME     : IXCL01EF
DISPOSITION     : KEEP
ACCESS TIME     : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS   : 0
```

Figure 24-37 Display of administration structure

The queue managers were restarted to show the recovery of the administration structure. Queue manager CSQ1 was started first and built its entries. The entries for CSQ2 and CSQ3 also were built.

Figure 24-38 shows CSQ1 rebuilding the administration structure.

```
12.49.12 STC19743 CSQE140I -CSQ1 CSQEENFR Started listening for ENF 35
171 events for structure CSQ_ADMIN
12.49.12 STC19743 IXL013I IXLCNN REQUEST FOR STRUCTURE IBM1CSQ_ADMIN FAILED.
173 JOBNAME: CSQ1MSTR ASID: 008D CONNECTOR NAME: CSQEIBM1CSQ101
173 IXLCNN RETURN CODE: 0000000C, REASON CODE: 02010C09
173 CONNECTIONS TO THE STRUCTURE ARE BEING PREVENTED AT THIS TIME
173 CONADIAGO: 00000002
173 CONADIAG1: 00000008
173 CONADIAG2: 00020C09
173 CONADIAG10: 80000000

12.49.14 STC19743 IXL014I IXLCNN REQUEST FOR STRUCTURE IBM1CSQ_ADMIN
12.49.14 STC19743 CSQE141I -CSQ1 CSQEENFR Stopped listening for ENF 35
183 events for structure CSQ_ADMIN
181 WAS SUCCESSFUL. JOBNAME: CSQ1MSTR ASID: 008D
12.49.14 STC19743 CSQE005I -CSQ1 CSQECONN Structure CSQ_ADMIN connected
181 CONNECTOR NAME: CSQEIBM1CSQ101 CFNAME: CF02
184 as CSQEIBM1CSQ101, version=CA879D76F68062EB 00010016
12.49.14 STC19743 IXL015I STRUCTURE ALLOCATION INFORMATION FOR
182 STRUCTURE IBM1CSQ_ADMIN, CONNECTOR NAME CSQEIBM1CSQ101,
182 CONNECTIVITY=DEFAULT
182 CFNAME ALLOCATION STATUS/FAILURE REASON
182 -----
182 CF01 NO CONNECTIVITY A8001800
182 CF02 STRUCTURE ALLOCATED AC001800
182 CF03 PREFERRED CF ALREADY SELECTED AC001800
182 PREFERRED CF HIGHER IN PREFLIST
12.49.14 STC19743 CSQE032I -CSQ1 CSQECSR1 Admin structure data from CSQ2 unavailable
12.49.14 STC19743 CSQE032I -CSQ1 CSQECSR1 Admin structure data from CSQ3 unavailable
12.49.14 STC19743 CSQE018I -CSQ1 CSQERAD2 Admin structure data building started
12.49.14 STC19743 CSQE036I -CSQ1 CSQEPRAD Admin structure data building started for
CSQ2
12.49.14 STC19743 CSQE019I -CSQ1 CSQERAD2 Admin structure data building completed
12.49.14 STC19743 CSQE032I -CSQ1 CSQECSR1 Admin structure data from CSQ2 unavailable
12.49.14 STC19743 CSQE032I -CSQ1 CSQECSR1 Admin structure data from CSQ3 unavailable
12.49.14 STC19743 CSQE037I -CSQ1 CSQEPRAD Admin structure data building 200
200 completed for CSQ2
12.49.14 STC19743 CSQE036I -CSQ1 CSQEPRAD Admin structure data building started for
CSQ3
12.49.14 STC19743 CSQE037I -CSQ1 CSQEPRAD Admin structure data building
205 completed for CSQ3
```

Figure 24-38 Display of CSQ1 JES log

Figure 24-39 shows CSQ2 reconnecting to the administration structure.

```
12.53.09 STC19748 CSQE140I -CSQ2 CSQEENFR Started listening for ENF 35
  170           events for structure CSQ_ADMIN

12.53.10 STC19748 IXL014I IXLCNN REQUEST FOR STRUCTURE IBM1CSQ_ADMIN
12.53.10 STC19748 CSQE141I -CSQ2 CSQEENFR Stopped listening for ENF 35
  177           events for structure CSQ_ADMIN
  176           WAS SUCCESSFUL.  JOBNAME: CSQ2MSTR ASID: 004D
  176           CONNECTOR NAME: CSQEIBM1CSQ202 CFNAME: CF02
12.53.10 STC19748 CSQE005I -CSQ2 CSQECONN Structure CSQ_ADMIN connected
  178           as CSQEIBM1CSQ202, version=CA879D76F68062EB 00030016
```

Figure 24-39 Display of CSQ2 JES log

Figure 24-40 shows CSQ3 reconnecting to the administration structure.

```
12.50.59 STC19745 CSQE140I -CSQ3 CSQEENFR Started listening for ENF 35
  503           events for structure CSQ_ADMIN
12.51.00 STC19745 IXL014I IXLCNN REQUEST FOR STRUCTURE IBM1CSQ_ADMIN
12.51.00 STC19745 CSQE141I -CSQ3 CSQEENFR Stopped listening for ENF 35
  510           events for structure CSQ_ADMIN
  509           WAS SUCCESSFUL.  JOBNAME: CSQ3MSTR ASID: 008F
  509           CONNECTOR NAME: CSQEIBM1CSQ303 CFNAME: CF02
12.51.00 STC19745 CSQE005I -CSQ3 CSQECONN Structure CSQ_ADMIN connected
  511           as CSQEIBM1CSQ303, version=CA879D76F68062EB 00020011
```

Figure 24-40 Display of CSQ3 JES log

The ability of a queue manager to recover another queue manager's administration entries if that queue manager is unavailable is shown in Figure 24-40.



A

MQSC scripts for the Coexistence scenario

This appendix provides the MQSC commands to define the required WebSphere MQ objects when the scenario that is described in Chapter 19, “Coexistence: A staged migration on Windows, UNIX, and Linux” on page 259 is prepared.

Defining the objects for queue manager QM1

The commands that are used to define the required objects for queue manager QM1 are shown in Example A-1. Lines that begin with an asterisk are comments.

Example A-1 MQSC commands to define the required objects for queue manager QM1

```
* -----
* Define a listener to receive network communications from queue manager QM2.
* -----



DEFINE LISTENER(QM1.LISTENER) +
    TRPTYPE(TCP)      +
    CONTROL(QMGR)     +
    PORT(1614)        +
    REPLACE



* -----
* Start the listener.
* -----



START LISTENER(QM1.LISTENER)



* -----
* Define the transmission queue to hold messages waiting to be sent to queue
* manager QM2 via the sender channel.
* -----



DEFINE QLOCAL(QM2)  +
    USAGE(XMITQ)   +
    REPLACE



* -----
* Define the remote request queue that is located on queue manager QM2.
* -----



DEFINE QREMOTE(REQUEST.QUEUE) +
    RNAME(REQUEST.QUEUE)  +
    RQMNAME(QM2)         +
    XMITQ(QM2)          +
    REPLACE



* -----
* Define the local reply queue where response messages from the Echo
* application should be sent back to.
* -----



DEFINE QLOCAL(REPLY.QUEUE) +
    REPLACE



* -----
* Define the sender channel that is used to transmit requests from the Request
* application to queue manager QM2, so they can be processed by the Echo
* application.
```

```

* -----
DEFINE CHANNEL(QM1.TO.QM2)      +
    CHLTYPE(SDR)          +
    XMITQ(QM2)           +
    CONNAME('localhost(1615)') +
    REPLACE

* -----
* Define the receiver channel that is used to receive reply messages sent from
* the Echo application that is connected to queue manager QM2.
* -----



DEFINE CHANNEL(QM2.TO.QM1) +
    CHLTYPE(RCVR)        +
    REPLACE

```

Defining the objects for queue manager QM2

The commands that are used to define the required objects for queue manager QM2 are shown in Example A-2. Lines beginning with an asterisk are comments.

Example A-2 MQSC commands to define the required objects for queue manager QM2

```

* -----
* Define a listener to receive network communications from queue manager QM1.
* -----



DEFINE LISTENER(QM2.LISTENER) +
    TRPTYPE(TCP)          +
    CONTROL(QMGR)         +
    PORT(1615)            +
    REPLACE

* -----
* Start the listener.
* -----



START LISTENER(QM2.LISTENER)

* -----
* Define the process that specifies how to start the Echo application.
*
* The value of the APPLICID attribute specifies the fully-qualified path
* to the Echo application.
*
* If you are using Windows this needs to be:
*
* APPLICID('C:\samples\amqsech.exe')
*
* or the equivalent path if you copied the Echo sample to an alternative
* location.

```

```

*
* If you are using UNIX or Linux this needs to be:
*
* APPLICID('/samples/amqsech')
*
* or the equivalent path if you copied the Echo sample to an alternative
* location.
* -----
DEFINE PROCESS(ECHO)          +
    APPLICID('/samples/amqsech') +
    REPLACE

* -----
* Define the transmission queue to hold messages waiting to be sent to queue
* manager QM1 via the sender channel.
* -----
DEFINE QLOCAL(QM1)  +
    USAGE(XMITQ) +
    REPLACE

* -----
* Define the initiation queue where WebSphere MQ should put trigger event
* messages to start the Echo application.
* -----
DEFINE QLOCAL(REQUEST.INITQ) +
    REPLACE

* -----
* Define the request queue to hold messages sent by the Request application
* before they are processed by the Echo application.
*
* This queue is configured so that WebSphere MQ generates a trigger message
* to start the Echo application when the number of messages on the queue
* changes from zero to one.
* -----
DEFINE QLOCAL(REQUEST.QUEUE) +
    TRIGGER          +
    TRIGTYPE(FIRST)  +
    PROCESS(ECHO)     +
    INITQ(REQUEST.INITQ) +
    REPLACE

* -----
* Define the sender channel that is used to transmit replies from the Echo
* application to queue manager QM1 so they can be received by the Request
* application.
* -----
DEFINE CHANNEL(QM2.T0.QM1)      +
    CHLTYP(SDR)        +
    XMITQ(QM1)         +

```

```
CONNAME('localhost(1614)') +
REPLACE

* -----
* Define the receiver channel that is used to receive request messages sent
* from the Request application that is connected to queue manager QM1.
* -----
```

```
DEFINE CHANNEL(QM1.T0.QM2) +
CHLTYPE(RCVR)      +
REPLACE
```




B

WebSphere MQ for z/OS 7.1 System Management Facility changes

This appendix provides the new system management facility (SMF) fields that were added for WebSphere MQ V7.1. These fields support the changes that were made to support most shared message data sets.

SMF 115 data changes

WebSphere MQ V7.1 added new SMF information to support the shared message data set functionality. The information in this appendix is from the Assembler definitions of the new areas and fields.

New shared message data sets SMF115 information: QESD record

In WebSphere MQ V7.1, a new self-defining area was added to hold the statistics information for shared message data sets. This record and the field meanings are shown in Table B-1.

New self-defining SMF 115 information was added to report on the SMDS activity. The data layout for Assembler can be found in the h1q.SCSQMACS(CSQQESD) library. The fields that are of interest from a monitoring perspective are highlighted in **bold** text in Table B-1.

Table B-1 SMF 115 record QESD

Name	Description
QESDID	Control block identifier
QESDLL	Control block length
QESDEYEC	QESD eye catcher
QESDSTR	Application structure name
QESDSTRN	Structure number
QESDSM	SMDS space map statistics
QESDSMU	Space map usage information
QESDSMBT	Total logical blocks for SMDS
QESDSMBS	Logical blocks for space map
QESDSMBD	Logical blocks for message data
QESDSMBU	Data blocks in use
QESDSMBF	Data blocks that are free
QESDSMMC	Count of messages in data set
QESDSMC	Request and result counts
QESDSMAR	Allocated space for message
QESDSMFR	Freed space
QESDSMRR	Reallocated space (for restart)
QESDSMCR	Cleaned up space (normal close)
QESDSMAP	Allocated pages
QESDSMFP	Freed pages
QESDSMRP	Reallocated pages

Name	Description
QESDSMCP	Clean up pages released
QESDSMFL	Allocate failed, data set full
QESDSMM	Maximum and minimum values
QESDSMMM	Maximum message count
QESDSMMU	Maximum used blocks
QESDSMMF	Minimum free blocks
QESDBF	SMDS buffer pool statistics
QESDBFU	Usage values
QESDBFSZ	Buffer size in bytes
QESDBFTO	Total buffers in pool
QESDBFUS	Used shared buffers
QESDBFUP	Used private buffers
QESDBFFS	Free saved buffers
QESDBFFE	Free empty buffers
QESDBFPW	Pool waiters (for any buffer)
QESDBFBW	Buffer waiters (buffer busy)
QESDBFC	Request counts and times
QESDBFGB	Number of times got a buffer
QESDBFGV	Got valid buffer
QESDBFGM	Got matching buffer but empty
QESDBFGF	Got a free empty buffer
QESDBFGL	Got (stole) LRU saved buffer
QESDBFGN	Got no buffer (conditional)
QESDBFRR	Number of read requests
QESDBFRS	Number of times read saved
QESDBFRP	Number of times read partial
QESDBFWR	Number of write requests
QESDBFFB	Freed buffer with valid data
QESDBFDB	Discarded buffer as empty
QESDBFWP	Waited for pool (free buffer)
QESDBFWB	Waited for buffer (when busy)
QESDBFPT	Total pool wait time
QESDBFBT	Total buffer wait time
QESDBFM	Maximum and minimum values

Name	Description
QESDBFMU	Maximum used buffers
QESDBFMF	Minimum free buffers
QESDBFMP	Max number of pool waiters
QESDBFMB	Max number of buffer waiters
QESDIO	MDS I/O statistics
QESDIOU	Data set usage values
QESDIOHA	High allocated control interval
QESDIOHU	High used control interval
QESDIOC1	Control interval size
QESDIOCA	Control area size
QESDIOC	Request counts, pages, and times
QESDIOF	Format write statistics
QESDIOFR	Format write (extend) requests
QESDIOFP	Format total pages
QESDIOFT	Format I/O time
QESDIOFW	Format wait time
QESDIOW	Normal write statistics
QESDIOWR	Write requests
QESDIOWP	Write total pages
QESDIOWT	Write I/O time
QESDIOWW	Write wait time
QESDIOR	Reads from owned data set
QESDIORR	Local read requests
QESDIORP	Local read total pages
QESDIORT	Local read I/O time
QESDIORW	Local read wait time
QESDIOO	Reads from other data sets
QESDIOOR	Other read requests
QESDIOOP	Other read total pages
QESDIOOT	Other read I/O time
QESDIOOW	Other read wait time

New SMF115 fields in existing structures

Some existing structures were modified to include information about the shared message data set support. All fields were added to the end of existing structures, except where noted.

CSQDQSST: Storage manager statistics structure

The following fields were added to the existing structure to hold information about above the bar (64-bit addressing) storage use:

- ▶ QSSTCN64: The count of contractions in the 64-bit storage area during the current SMF interval.
- ▶ QSSTCR64: The count of short on 64-bit storage conditions during the SMF interval.

CSQDQWS1: Mapping macro for SMF 115 subtype 2

The following fields were added to provide the offset, length, and number of the new QESD entries in the SMF 115 subtype 2 records:

- ▶ QWS10R8O: The offset to the QESD area in the SMF115 record.
- ▶ QWS10R8L: The length of the QESD area.
- ▶ QWS10R8N: The number of QESD sections that are present.

CSQDWQ - queue statistics mapping macro

The following fields were added to the queue statistics information structure:

- ▶ PUTDSQ: The number of puts to a shared queue.
- ▶ PUTIGQ: The number of puts to the intra-group queuing transmission queue.
- ▶ PUT1IGQ: The number of MQPUT1s to the intra-group queuing transmission queue.

CSQDWTAS: Task-related statistics mapping macro

The following fields were added to the thread-related accounting structure (an SMF116 structure) to support shared message data sets. The fields are intended for the task that is associated with this record:

Warning: These fields were not added to the end of the structure. Instead, they were added before the interval data. Although there were 24 bytes of reserved space before the interval information, it seems as though 28 bytes were inserted. If programs are used that parse the SMF data, the offsets of a number of fields change, which affect the results.

- ▶ WTASSMRS: The count of when a read from an SMDS is satisfied from the buffers for this task.
- ▶ WTASSMRB: The number of message holding blocks that are read from SMDS.
- ▶ WTASSMWB: The number of message holding blocks that are written to SMDS.
- ▶ WTASSMRP: The number of pages that are read from SMDS.
- ▶ WTASSMWP: The number of pages that are written to SMDS.
- ▶ WTASSMWT: The cumulative time that is spent waiting on SMDS I/O.



C

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server at this website:

<ftp://www.redbooks.ibm.com/redbooks/SG248087>

Alternatively, you can go to the IBM Redbooks website at:

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

Select **Additional materials** and open the directory that corresponds with the IBM Redbooks form number SG248087.

Using the web material

The web material that accompanies this book includes the file SMDS_Test_JCL.zip, which is a compressed file of job code language (JCL) samples.

The JCL samples include the following files:

- DISPLAY_JCL.txt: JCL to display the shared message data set usage and specific queue depths. This JCL was used in the tutorial about migrating from DB2 to SMDS that accompanies this book.
- PUT2DB2_JCL.txt: JCL to put messages onto queues that use DB2 as the offload storage. This JCL was used in the tutorial about migrating from DB2 to SMDS that accompanies this book.
- PUT2SMDS_JCL.txt: JCL to put messages onto queues that use SMDS (or were migrated to use SMDS) as the offload storage.shared message data set. This JCL was used in the tutorial about migrating from DB2 to SMDS that accompanies this book.
- MultipleMsgSizeTestJCL.txt: JCL to submit multiple jobs that use OEMPUTX (from SupportPac IP13) to put messages on queues. This sample JCL is used to test multiple message sizes when SMDS is used for message offloading.

System requirements for downloading the web material

The web material requires the following system configuration:

- ▶ Hard disk space: 3 K
- ▶ Operating System: z/OS

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation and extract the contents of the web material compressed file (SG248087.zip) into this folder. The JCL then can be transferred to z/OS by using normal transfer methods.

Glossary

API exit A user-written program that monitors or modifies the function of an MQI call. For each MQI call that is issued by an application, the API exit is invoked before the queue manager starts to process the call and again after the queue manager completes processing the call. The API exit can inspect and modify any of the parameters on the MQI call.

Application server A managed environment within which applications are deployed and run, with access to a defined set of functionality that can include messaging facilities, such as WebSphere MQ.

Asynchronous Actions that occur without any constraints on timing; for example, sending a message to an independent party that might or might not be active. Compare with Synchronous.

ASCII American National Standard Code for Information Interchange. By using a coded character set that consists of 7-bit coded characters (8 bits including parity check), the code that is used for information interchange among data processing systems, data communication systems, and associated equipment.

Broker In a Publish/Subscribe messaging model, a broker maintains information about topics and the subscribers to those topics. When a publisher publishes information about a topic to the broker, the broker distributes that information to all registered subscribers.

Browse In terms of WebSphere MQ, to examine the contents of a message without destructively removing it from the queue.

Business critical data Data that is not stored elsewhere in the system. If this data is lost, important information or a change in state within the system also is lost.

Callback A software technique that allows a function to be called when a defined event occurs independent of other threads of the program.

Certificate In computer security, a technical evaluation that is made as part of and in support of the accreditation process that establishes the extent to which a particular computer system or network design and implementation meet a pre-specified set of requirements.

Channel exits A security exit forms a secure connection between two security exit programs. One program is for the sending message channel agent (MCA), and one is for the receiving MCA.

Channel A network communications link between two queue managers over which messages flow, or a network communications link between an application and a queue manager over which Message Queuing Interface (MQI) commands flow.

Client application An application that connects to a WebSphere MQ queue manager over a network.

Cluster See queue manager cluster.

Cluster message channel A message channel between two queue managers within the same queue manager cluster.

Co-existence In terms of WebSphere MQ for z/OS, the ability to install and run more than one version of the product in one operating system image. Co-existence does not exist on any other platforms that are supported by WebSphere MQ.

Common Object Request Broker Architecture (CORBA) A standard for software calls that enables programs to communicate across multiple languages and computers.

Conversation In terms of network communication, an established conduit for the exchange of information between two computer programs.

Coupling facility A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

Coupling facility structures Is used to store the messages that are on shared queues. Each coupling facility structure that is used by WebSphere MQ is dedicated to a specific queue-sharing group, but a Coupling Facility can hold structures for more than one queue-sharing group.

Cross-system coupling facility (XCF) A component that provides functions to support cooperation between authorized programs that are running within a sysplex.

Cross-system extended services (XES) A component that provides functions to support cooperation between software products on the z/OS platform.

Customer Information Control System

(CICS) IBM proprietary transaction server that runs on z/OS systems.

Data conversion The process of converting the binary representation of characters and numbers from that of one environment to that of another.

DB2 An IBM relational database management system.

Distributed message channel A message channel between two queue managers in which all messages are transferred from a single transmission queue on one queue manager to destination queues on the remote queue manager.

Distributed queuing Using particular types of WebSphere MQ channels to interconnect queue managers, namely SENDER, RECEIVER, SERVER, and REQUESTOR. Compare with queue manager cluster.

EBCDIC Extended binary-code decimal interchange code. A code character set of 256 8-bit characters.

Eclipse An open-source extensible software platform for integrated development and management of applications that primarily are written in Java. Led by the Eclipse Foundation.

Engine An internal software component that performs common functions that are used throughout a product; for example, publish/subscribe in WebSphere MQ.

Enterprise Service Bus (ESB) A software architecture in which messaging integration technologies are used, generally as part of a SOA implementation.

Exactly once delivery An assurance that is provided by a messaging infrastructure that a message arrives at its destination and that it arrives one time, and only one time.

File Transfer Edition (FTE) IBM WebSphere family product that is used to transfers files between systems.

Full duplex A communication system involving two connected parties where information can be sent in either direction at any time. Compare with Half duplex.

Global unit of work A unit of work which includes actions upon multiple different resources, which can include WebSphere MQ and database products. This unit of work is coordinated by a transaction manager.

Half duplex A communication system involving two connected parties in which information can be sent in only one direction at a time. Compare with Full duplex.

Handle A binary number that represents an active context or reference to an entity, such as a connection, open queue, or message in WebSphere MQ.

Heartbeat In WebSphere MQ, a flow that is periodically sent over channels when there are no WebSphere MQ messages to send, which allows channel failures to be detected earlier.

High Availability Refers to a system, component, or resource that is continuously operational even though there was a hardware or software failure.

Hub and Spoke architecture A WebSphere MQ infrastructure architecture in which services are provided by a small number of hub queue managers, and access to those services is extended through many intermediate spoke queue managers that are interconnected with those hub queue managers.

Hyper Text Transport Protocol (HTTP) A protocol for client-server communication that was popularized by Internet for web servers to deliver content to web browser clients.

IBM Message Service Client (XMS) An application programming interface for the C and C++ programming languages and the .NET environment, which is consistent with the Java Message Service application programming interface.

In-doubt unit of recover The status of a unit of recovery for which a sync point is requested but not yet confirmed.

Information Management System (IMS) Any of several system environments that are available with a database manager and transaction processing that are capable of managing complex databases and terminal networks.

IMSpex A set of systems in an IMS network that work together as a unit. Typically, these systems share resources, run in a Parallel Sysplex® environment, and include a Common Service Layer (CSL).

Java Message Service (JMS) An industry standardized application programming interface for the Java programming language, which is part of the Java 2 platform Enterprise Edition standard.

Message A piece of information with addressing or other meta information associated that can be passed between software components.

LPAR A virtualized computing environment that is abstracted from all physical devices. On IBM mainframes, LPARs can form a Sysplex or Parallel Sysplex.

Library A file or a set of related files; for example, a collection of functions, calls, subroutines, or other data.

Heartbeat In WebSphere MQ, a flow that is periodically sent over channels when there are no WebSphere MQ messages to send, which allows channel failures to be detected earlier.

Message channel A network communications link between two queue managers over which messages flow.

Message Channel Agent (MCA) A component of a WebSphere MQ queue manager (or a WebSphere MQ client product) that forms one half of a channel and establishes network communications with, or responding to network communications from, a partner MCA.

Message Descriptor See WebSphere MQ Message Descriptor (MQMD).

Message Data A data structure that is associated with each WebSphere MQ message and contains application information that is associated with that message.

Message property A name and associated value that is stored a WebSphere MQ message, which is separated from the application message data.

Message queuing A middleware technique that allows unlike software components to interact asynchronously through a queue.

Message queuing interface (MQI) The core application interface that is used to interact with a WebSphere MQ infrastructure.

Message queuing interface (MQI) channel A network communications link between an application and a queue manager over which MQI commands flow.

MQSC WebSphere MQ script commands to Human readable commands, uniform across all platforms, that are used to manipulate WebSphere MQ objects.

Message selector A logical combination of message property names and values that defines criteria for selecting messages from a queue.

Middleware A software infrastructure layer between applications and the infrastructure components they interact with, which is common to multiple nodes in a system and simplifies interaction between the unlike software and hardware components that reside on those nodes.

Non-persistent message A WebSphere MQ message that is not recovered if a queue manager fails, is restarted, or if there is a failure in the underlying infrastructure or operating system. Compare with Persistent message.

Object Authority Manager (OAM) A component of a WebSphere MQ queue manager that performs authority checking.

Open Transaction Manager Access (OTMA) A component of IMS that implements a transaction-based, connectionless client/server protocol in an MVSTTM sysplex environment. The domain of the protocol is restricted to the domain of the z/OS cross-system coupling facility (XCF). OTMA connects clients to servers so that the client can support a large network (or many sessions) while maintaining high performance.

Performance The time taken between submitting a request for a service and completion of that service. How the start and end points of a service are determined are specific to the function that is performed by the service.

Persistent message A WebSphere MQ message that can be fully recovered if a queue manager fails, is restarted, or if there is a failure in the underlying infrastructure or operating system. Persistent messages are required for guaranteed message delivery in WebSphere MQ. Compare with Non-persistent message.

Personal certificate A public certificate that can be used to identify an entity, combined with the private key for that certificate.

Point to point messaging The sending of messages from one location to a single destination that is determined based upon addressing information provided by the sender of the message.

Polling Repeatedly requesting a piece of information at regular intervals to detect changes in that information.

Process An executing instance of a computer program that consists of one or more threads and various resources that is managed by an operating system.

Production environment An environment through which real services are made available within or outside of the business.

Program communication block (PCB) A control block that contains pointers to Information Management System (IMS) databases.

Proxy An interface between an existing service, usually with a proprietary interface and a middleware layer that is used by other nodes in the system to access that service.

Publish/Subscribe messaging A model of messaging in which the producers of information do not have direct knowledge of the consumers of that information, which might be zero or many.

Publisher In a Publish/Subscribe messaging model, a publisher produces information about a particular topic that is distributed to registered subscribers on that topic by a broker.

Query data Transient data that is sent through a system, derived from data that is stored safely within the system.

Queue A container for messages, from which messages are usually retrieved in first-in-first-out order, which can be used as an asynchronous buffer between two software components.

Queue manager Queue managers are the interconnected nodes within a WebSphere MQ infrastructure that maintain the messages and queues and provide data integrity and applications with access to the infrastructure to send and receive messages.

Queue manager cluster A mechanism that is provided by WebSphere MQ to interconnect queue managers in a flexible way by using CLUSSDR and CLUSRCVR channels, which simplifies administration and provides workload balancing facilities for scalability and service availability. Compare with Distributed queuing.

Queue name resolution The action that is performed by a queue manager whenever an application or channel attempts to open a queue to put a message on a queue that is hosted by that queue manager, or to send a message through that queue manager.

Queue sharing group A feature of WebSphere MQ for z/OS that allows applications that are connected to multiple queue managers running on different z/OS systems within a sysplex to get and put messages to the same queue.

Remote Procedure Call (RPC) Calling a software component from a program as though it were local, but the called procedure might be executed on another computer. The calling program waits for the called procedure to complete execution before continuing.

Request/reply messaging Asynchronous communication between two software components, in which a request message is sent and a reply message is returned following processing of the request.

Repackage The process to recreate the original MQ code package to install WebSphere MQ code in multi-version on Linux.

Unit of work (UOW) A logical grouping of actions, which must all succeed or all fail.

Unit of recovery A recoverable sequence of operations within a single resource manager, such as an instance of DB2 for z/OS.

Resiliency The ability of a WebSphere MQ queue manager in a queue sharing group to tolerate the loss of a coupling facility structure without terminating.

Resource Access Control Facility (RACF) IBM proprietary security management service that runs on z/OS systems.

Resource manager A component that, under the control of a transaction manager, manages an individual resource that is participating in a global unit of work.

Scalability The ease with which the capacity of the system can be increased to cope with increased load, and how this increase affects performance.

Secure Sockets Layer (SSL) An industry standardized technology to provide authentication and secure communication.

Send and forget messaging The sending of messages without requiring a reply upon processing of those messages; hence, relying on the exactly once delivery assurance of the message queuing infrastructure to deliver the message.

Service-oriented architecture (SOA) A software architecture in which business processes are grouped and interoperate as loosely coupled services. SOA includes well-defined guiding principles and architectural constructs.

Servlet A Java object in a web server that receives requests and generates replies, typically involving HTML content.

Shared queue In WebSphere MQ for z/OS, a type of local queue. The messages on the queue are stored in the coupling facility and can be accessed by one or more queue managers in a queue-sharing group. The definition of the queue is stored in the shared repository.

Shared message data sets (SMDS) A data set that is used by a queue manager to store offloaded message data for shared messages that are stored in a coupling facility structure.

Socket A logical end-point in a connection between two processes over the TCP/IP network communications protocol.

State information Information that changes over time, but only has one value at any point in time.

Subscriber In a Publish/Subscribe messaging model, a subscriber registers with a broker to receive all information that is published on a particular topic.

SupportPac A package of other functionality or documentation for the WebSphere MQ product, which is distributed through the IBM SupportPacs web page.

Synchronous Actions that occur with constraints on timing; for example, invoking a Remote Procedure Call where the other program must be available to immediately service the call. Compare with Asynchronous.

Thread A single independent sequence of executing instructions in a Process.

Topic In a Publish/Subscribe messaging model, a topic uses group information so that publishers that produce information about a topic can be loosely coupled with subscribers that consume information about that topic.

Transaction The mechanism by which multiple actions, possibly upon multiple resources, can be grouped together in a unit of work.

Transaction manager The component that manages the resources participating in a global unit of work.

Transaction instance block (TIB) In OTMA, is associated with a CM1 input/output message or a CM0 input message.

Transport Layer Security (TLS) An industry standardized technology to provide authentication and secure communication.

Unit of work A logical grouping of actions, which must all succeed or all fail.

VSAM Virtual Storage Access Method. A multifunction, all-purpose IBM data access method.

Web services A standardized way to describe and invoke services.

WebSphere Java Client Part of a WebSphere MQ product that can be installed on a system without installing the full queue manager. The WebSphere MQ Java client is used by Java applications (both WebSphere MQ classes for Java and WebSphere MQ classes for JMS) and communicates with a queue manager on a server system.

WebSphere Message Broker Is an ESB built for universal connectivity and transformation in heterogeneous IT environments. It builds on the WebSphere MQ messaging layers with ESB capabilities that add transformation, intelligent routing, and information flow modeling.

WebSphere MQ Client The software components of WebSphere MQ that allow client applications to connect to a local or remote queue manager over client connection channels.

WebSphere MQ-IMS bridge The component of WebSphere MQ for z/OS that allows direct access from WebSphere MQ applications to applications on your IMS system.

WebSphere MQ Message Descriptor (MQMD) A data structure that is associated with each WebSphere MQ message and contains meta information that is associated with that message, such as identifying and type information.

WebSphere MQ object model A defined set of classes, methods, and properties to interact with WebSphere MQ, which are implemented for multiple object oriented programming languages, including Java and C++, and build upon the facilities that are provided by the Message Queuing Interface.

WebSphere MQ Server The software components of WebSphere MQ that allow a queue manager instance to be run on the local machine. Server also includes support for local Clients.

WebSphere MQ Telemetry Client WebSphere MQ Telemetry provides small client libraries that can be embedded into smart devices that are running on a number of different device platforms. Applications that are built with the clients use MQ Telemetry Transport (MQTT) and the WebSphere MQ Telemetry service to publish and subscribe messages reliably with WebSphere MQ.

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this book. Some publications that are referenced in this list might be available in softcopy only:

- ▶ *IBM WebSphere MQ Primer*, REDP0021
- ▶ *Universal Messaging with IBM WebSphere MQ*, TIPS0925
- ▶ *WebSphere MQ V6 Fundamentals*, SG24-7128-00
- ▶ *High Availability in WebSphere Messaging Solutions*, SG24-7839
- ▶ *Secure Messaging Scenarios with WebSphere MQ*, SG24-8069
- ▶ *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, SG24-8054

You can search for, view, download, or order these publications and other Redbooks, Redpapers, Web Docs, draft, and other materials at the following website:

<http://ibm.com/redbooks>

Other publications

These publications also are relevant as further information sources:

- ▶ WebSphere MQ Managed File Transfer introduction
http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=%2Fcom.ibm.wmqfte.doc%2Fwmqfte_intro.htm
- ▶ WebSphere MQ Managed File Transfer Diagnostic Messages
http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=%2Fcom.ibm.wmqfte.messages.doc%2Fmessages_main.htm
- ▶ Message Service Client for .NET
http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=%2Fcom.ibm.mq.msdoc%2Fxms_cgetstd_intronet.html
- ▶ WebSphere MQ Hypervisor Editions
http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp?topic=%2Fcom.ibm.mq.hypervisor.doc%2Fhy00100_.htm

Online resources

These websites also relevant as further information sources:

- ▶ IBM WebSphere MQ for Multiplatforms Sales Manual
http://www-01.ibm.com/common/ssi>ShowDoc.wss?docURL=/common/ssi/rep_sm/2/897/EN_US5724-H72/index.html&lang=en&request_locale=en
- ▶ WebSphere MQ Product Information
<http://www.ibm.com/support/docview.wss?uid=swg27007431>
- ▶ WebSphere MQ System Requirements
<http://www.ibm.com/software/integration/wmq/requirements>
- ▶ WebSphere MQ Recommended Fixes page
<http://ibm.co/WMQRecommendedFixes>
- ▶ Fixes by version for WebSphere MQ
<http://www.ibm.com/support/docview.wss?uid=swg21254675>
- ▶ WebSphere MQ V7.5 Information Center
<http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/index.jsp>
- ▶ WebSphere MQ V7.1 Information Center
<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/index.jsp>
- ▶ WebSphere MQ V7.0.1 Information Center
<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp>
- ▶ WebSphere MQ - SupportPacs by Product
<http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg27007197#1>
- ▶ Business Integration - WebSphere MQ SupportPacs
<http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg27007205>
- ▶ MSOP: WebSphere MQ Explorer - Configuration and Display Extension Plug-in
http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24011617&loc=en_US&cs=utf-8&lang=en
- ▶ MQC75: WebSphere MQ V7.5 Clients
<http://www-304.ibm.com/support/docview.wss?uid=swg24032744>
- ▶ Cryptography Card List for WebSphere MQ v7.1 and v7.5
<http://www-01.ibm.com/support/docview.wss?uid=swg21516803>
- ▶ IBM Support Portal: Support home MQ
http://www-947.ibm.com/support/entry/portal/overview//software/websphere/websphere_mq
- ▶ z/OS V1R12.0 information center
<http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp>
- ▶ z/OS V1R13.0 information center
<http://publib.boulder.ibm.com/infocenter/zos/v1r13/index.jsp>
- ▶ CICS TS Version 4.1
<http://pic.dhe.ibm.com/infocenter/cicsts/v4r1/index.jsp>

- ▶ IMS Version 10 information center
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.ims10.doc%2Fimshome.htm>
- ▶ DFSMS Storage Administration Reference
<http://publib.boulder.ibm.com/infocenter/zos/v1r12/topic/com.ibm.zos.r12.ida/ida.htm>
- ▶ MP1H: Performance report, WebSphere MQ for z/OS version 7.1.0
<http://www-01.ibm.com/support/docview.wss?uid=swg24031663>
- ▶ Service Oriented Architecture (SOA)
<http://www.ibm.com/soa>
- ▶ MQTT Org
<http://www.mqtt.org>

Help from IBM

IBM Support and downloads

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

IBM Global Services

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



Redbooks

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements



Redbooks

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

(1.5" spine)
1.5" <-> 1.998"
789 <-> 1051 pages



Redbooks

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages

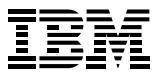


Redbooks

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

(0.5" spine)
0.475" <-> 0.873"
250 <-> 459 pages

(0.2" spine)
0.17" <-> 0.473"
90 <-> 249 pages



Redbooks

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

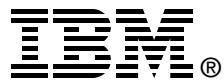
(2.0" spine)
2.0" <-> 2.498"
1052 <-> 1314 pages



Redbooks

IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements

(2.5" spine)
2.5" <-> nnn.n"
1315 <-> nnnn pages



IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements



Maximize your investment in WebSphere MQ

Discover new features that bring value to your business

Learn from scenarios with sample configurations

This IBM® Redbooks® publication is divided into four parts:

Part 1 introduces message-oriented middleware and the WebSphere® MQ product. It explains how messaging technologies are implemented in WebSphere MQ and shows how to get started with configuring a WebSphere MQ environment. This part briefly lists the new features of WebSphere MQ V7.1 and V7.5.

Part 2 introduces the enhancements to WebSphere MQ in Version 7 Release 1. It provides a description of the new features, their business value, and usage examples. It describes enhancements to WebSphere MQ for multiplatforms and z/OS®. Examples of features that are discussed in this part include multiple installation support for multiplatforms, enhanced security with channel authentication records, enhanced clustering, improved availability and scalability on z/OS, and more.

Part 3 introduces the enhancements to WebSphere MQ in Version 7 Release 5 for multiplatforms. It provides a description of the new features, their business value, and usage examples. Examples of enhancements that are discussed in this part include new installation options, such as the bundling of WebSphere MQ Advanced Message Security and WebSphere MQ Managed File Transfer.

Part 4 contains practical scenarios that demonstrate how the new features and enhancements work and how to use them.

In summary, the introduction gives a broad understanding of messaging technologies and WebSphere MQ. It helps you understand the business value of WebSphere MQ. It provides introductory information to help you get started with WebSphere MQ. No previous knowledge of the product and messaging technologies is assumed.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**