

Managing WebSphere Message Broker Resources in a Production Environment

Discover best practices for
administering the Message Broker

Learn about security, backup,
and problem determination

Use sample scenarios
and scripts



Saida Davies
Martin Cernicky
Alywin BC Ching
Brian M McCarty
Gregorio Patino
Amar Shah



International Technical Support Organization

Managing WebSphere Message Broker Resources in a Production Environment

September 2006

Archived

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

First Edition (September 2006)

Microsoft Windows XP Service Pack 1

Version	Release	Modification	Fix pack	Product name	Product number
6	0	0	6.0.0.1	WebSphere Message Broker	
6	0	0	6.0.0.1	WebSphere MQ	
8	2	0	5 b	DB2	

AIX V5.3 Technology Level 04 Service Pack 03

Version	Release	Modification	Fix pack	Product name	Product number
6	0	0	6.0.0.1	WebSphere Message Broker	
6	0	0	6.0.0.1	WebSphere MQ	
8	2	0	5 b	DB2	

© Copyright International Business Machines Corporation 2006. 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

Figures	ix
Tables	xi
Notices	xiii
Trademarks	xiv
Preface	xv
The team that wrote this redbook	xv
Become a published author	xviii
Comments welcome	xviii
Part 1. Introduction	1
Chapter 1. Overview of this book	3
1.1 The scope of the book	4
1.2 Audience for this book	4
1.3 Assumptions and prerequisites	5
1.4 Business relevance	5
Chapter 2. WebSphere Message Broker features	7
2.1 Capability and environments	8
2.1.1 Capabilities description	8
2.1.2 Environments description	9
2.2 Components description and function	11
2.2.1 Broker	11
2.2.2 Configuration Manager	12
2.2.3 User Name Server	13
2.2.4 Toolkit	13
Chapter 3. WebSphere Message Broker environment	15
3.1 Overview of environments	16
3.1.1 Development	17
3.1.2 Test and acceptance	18
3.1.3 Production	18
3.2 Managing broker domain	18
3.2.1 Resource management	19
3.2.2 Security	21
3.2.3 Backup, restore, and recover	22
3.2.4 Monitor and health-check	23

3.2.5 Maintenance strategy	27
3.2.6 Problem determination	30
3.3 Recommended considerations	31
3.3.1 Naming convention	31
3.3.2 Queue manager	33
3.3.3 Broker	36
3.3.4 Database	37
3.3.5 Syslog configurations	39
Part 2. Best practices	41
Chapter 4. Resource management	43
4.1 Managing changes in the domain	44
4.1.1 Versioning	44
4.1.2 Procedures and documentation	47
4.1.3 Automation	48
4.2 Managing problems in the domain	50
4.3 Administering the domain	51
4.3.1 Automation	51
4.4 Implementing reliability, availability, and scalability	53
4.4.1 Reliability	53
4.4.2 High Availability	54
4.4.3 Scalability	58
4.5 Characteristics for message broker components	62
4.5.1 Broker	62
4.5.2 Configuration Manager	69
4.5.3 User Name Server	70
4.5.4 Toolkit	75
Chapter 5. Security	83
5.1 Securing the domain	84
5.1.1 Authentication	84
5.1.2 Authorization	89
5.1.3 Secure Sockets Layer	96
5.1.4 Message protection	106
5.1.5 Complementary options	107
5.2 Characteristics for message broker component	109
5.2.1 Broker	110
5.2.2 Configuration Manager	111
5.2.3 User Name Server	111
5.2.4 Toolkit	113
Chapter 6. Backup, restore, and recover	115
6.1 Backing up the domain	116

6.1.1 Backup options for database	116
6.1.2 Scheduling and expiration	116
6.2 Restoring and recovering the domain	118
6.2.1 Restoring from backups	119
6.2.2 Recovering from disaster	119
6.3 Characteristics for message broker components	119
6.3.1 Broker	119
6.3.2 Configuration Manager	123
6.3.3 User Name Server	127
6.3.4 Toolkit	129
Chapter 7. Monitor and health-check	135
7.1 Basic of monitoring and health-checking	136
7.1.1 Availability	136
7.1.2 Reliability	136
7.1.3 Performance	137
7.1.4 Frequency	137
7.1.5 Archiving	138
7.1.6 Reporting	138
7.1.7 Alerting	138
7.2 Monitoring the domain	139
7.2.1 Monitoring systems for availability and failure events	139
7.2.2 Monitoring systems for performance and reliability	140
7.2.3 Basic parameters	141
7.3 Optimizing resources	144
7.3.1 Tools	145
7.4 Characteristics for message broker components	146
7.4.1 Broker	146
7.4.2 Configuration Manager	150
7.4.3 User Name Server	150
7.4.4 Toolkit	151
Chapter 8. Maintenance strategy	153
8.1 Maintaining the domain updates	154
8.1.1 Fix pack levels	154
8.1.2 Version levels	155
8.1.3 Staging updates	156
8.1.4 Update procedure	158
8.2 Characteristics for message broker components	161
8.2.1 Broker	161
8.2.2 Configuration Manager	161
8.2.3 User Name Server	162
8.2.4 Toolkit	162

Chapter 9. Problem determination	163
9.1 Basic steps	164
9.1.1 Identifying the failing components	164
9.1.2 Categorizing errors	165
9.1.3 Event messages	165
9.1.4 Locating additional information	175
9.1.5 Other useful logs	179
9.1.6 Collecting data for further support	181
9.2 Advanced steps	192
9.2.1 Tracing	192
9.2.2 Analyzing abends and cores	207
9.2.3 Investigating ODBC errors	210
9.2.4 Configuration Manager connection problem	210
9.2.5 Debugging message flows	211
9.3 Characteristics for message broker components	211
9.3.1 Problem determination with the broker component	211
9.3.2 Problem determination with the Configuration Manager	212
9.3.3 Problem determination with the User Name Server	212
9.3.4 Problem determination with the Message Brokers Toolkit	213
Chapter 10. Scenario	217
10.1 Environment setup	218
10.1.1 The logical topology	218
10.1.2 The physical topology	220
10.2 Resource management	222
10.2.1 Change management	222
10.2.2 Creating a repeatable procedure for bar deploys	222
10.3 Security	234
10.3.1 Configuration of the Certificate Authority	234
10.3.2 Implementing HTTPS support for broker	238
10.3.3 Connection encryption in Toolkit	248
10.4 Backup, restore, and recover	266
10.4.1 Backing up broker	267
10.4.2 Restoring broker from backup	268
10.4.3 Re-creating broker when broker backup not available	271
10.5 Problem determination	276
10.5.1 Alerts and logs	281
10.5.2 Failing components	282
10.5.3 Tracing	282
10.5.4 Resolving	283
Appendix A. Message Brokers Toolkit update without internet access	285
Installing an interim fix	286

How to speed-up the process significantly (advanced topic)	286
Installing a fix pack or refresh pack	287
Appendix B. Screen samples from IBM Tivoli Enterprise Portal	289
Main product console window	290
Broker related tasks	292
Queue manager related tasks	296
Web-based portal	298
Appendix C. Additional material	301
Locating the Web material	301
Using the Web material	302
How to use the Web material	302
Abbreviations and acronyms	303
Related publications	305
IBM Redbooks	305
How to get IBM Redbooks	305
Help from IBM	306
Index	307

Figures

2-1	WebSphere Message Broker V6.0 environment	9
3-1	Environment overview	17
4-1	Setting description information for a message flow	45
4-2	Properties of a message flow after deployment	45
4-3	User Defined Properties.	46
4-4	Queue manager failover using shared disks	57
4-5	Execution Group Platform Processor Architecture default.	67
4-6	Selecting Processor Architecture for execution group	68
4-7	Properties view showing Processor Architecture.	68
4-8	Topic Access Control List	73
4-9	Managing policies for the topics.	74
4-10	User Access Control List	75
4-11	Message Brokers Toolkit with many Capabilities enabled.	77
4-12	Message Brokers Toolkit with no Capabilities enabled	79
4-13	Altering the Message Brokers Toolkit shortcut	80
4-14	Managing different broker domains	81
5-1	Authentication configuration.	85
5-2	Protocol handling	86
5-3	Domains view of the Message Brokers Toolkit	89
5-4	ACL example	94
6-1	Selecting all Projects from Resource Navigator view	130
6-2	Saving the export in a zipped file	131
6-3	Selecting the restore option	132
6-4	Selecting the resources to be restored	133
6-5	Resources restored in the project	134
9-1	Message from the Message Brokers Toolkit	166
9-2	Windows Event Viewer - Application log	169
9-3	Searching for diagnostic messages in the Information Center	176
9-4	Successful deploy message	214
9-5	Example of pop-up message from deploy action.	215
9-6	Messages in the Alerts view	215
9-7	Filtering alerts from a broker domain	216
10-1	The logical topology of message broker scenario environment.	218
10-2	The physical topology of message broker scenario environment	220
10-3	WmbDeployTask class diagram	225
10-4	Package Explorer view of the WmbDeployTool Project.	226
10-5	Ant view showing WmbDeployTool.xml	231
10-6	Client's sample message flow	243

10-7	Server's sample message flow	243
10-8	Invoking a Web service using SSL	244
10-9	Configuring SSL in the HTTP Input node.	245
10-10	Invocation of the server message flow from a Web browser	246
10-11	SOAP response from the web service message flow	247
10-12	Response from the client message flow	247
10-13	Opening the IBM Key Management tool	249
10-14	Keystore parameter values	250
10-15	Assigning a password	251
10-16	Password location window	251
10-17	Adding a new CA root certificate	252
10-18	Adding the CA's certificate from a file	253
10-19	Label of the CA certificate	253
10-20	List of CA certificates, including the new added certificate	254
10-21	Creating a new certificate request	255
10-22	Certificate Request form	256
10-23	Location of the certificate request file	256
10-24	Receiving the certificate response from the CA	258
10-25	Certificate response location	258
10-26	The added certificate for the queue manager	259
10-27	Securing the WebSphere MQ channel	262
10-28	Setting the SSL properties in the SYSTEM.BKR.CONFIG channel	263
10-29	Domain configuration properties	265
10-30	Connecting to the Configuration Manager from the Toolkit with SSL	266
10-31	Connecting to Configuration Manager using CMPAPI Exerciser.	273
10-32	Connected to Configuration Manager	273
10-33	Selecting the option to delete the broker	274
10-34	Specify the broker name to be deleted	274
10-35	Output after deleting the broker	275
10-36	TestDB.msgflow	276
B-1	Main IBM Tivoli Enterprise Portal window	290
B-2	Message broker components view	292
B-3	Execution group view	293
B-4	Message flow view	294
B-5	Broker actions	295
B-6	Error Log view	296
B-7	Queue Statistics view	297
B-8	Web-based portal main window	298
B-9	Web-based message flow view	299

Tables

6-1	Three generation backup strategy	117
8-1	Advantages (+) and disadvantages (-) for update alternatives of domain components	158
9-1	Structure of user.log.	172
9-2	Example inserts for a BIP message.	177
10-1	The domain components	219
10-2	The Windows machine 1	221
10-3	The Windows machine 2	221
10-4	The AIX machine	222
10-5	SSL properties	242
10-6	SSL Java system properties	264

Notices

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

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

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

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

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

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

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

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

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

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

developerWorks®

pSeries®

z/OS®

AIX®

Cloudscape™

DB2 Universal Database™

DB2®

HACMP™

IBM®

OMEGAMON®


OS/400®

Parallel Sysplex®

RACF®

Rational®

Redbooks™

Redbooks (logo) ™

SupportPac™

Tivoli Enterprise™

Tivoli®

WebSphere®

The following terms are trademarks of other companies:

Java, JRE, JVM, J2EE, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. 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.

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

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

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

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

Preface

This IBM® Redbook provides guidance on how to manage WebSphere Message Broker V6.0 in a production environment.

It discusses best practices for administering the message broker domain and its components and includes additional information that is useful for WebSphere Message Broker V6.0 architects, designers, and project managers. It discusses security considerations for the message broker domain, how to backup and to restore resources, problem determination steps, and how to gather additional data if further message broker support is required. The book also demonstrates these important management tasks as sample scenarios, which are available to download.

The experts who compiled the information in this book have worked extensively on various versions of WebSphere Message Broker.

The team that wrote this redbook

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

Saida Davies is a Project Leader for the ITSO and has 17 years of experience in IT. She has published several Redbooks™ on WebSphere® Business Integration topics for multiple platforms. Saida has experience in the architecture and design of WebSphere MQ solutions, extensive knowledge of z/OS® operating system and a detailed working knowledge of both IBM and Independent Software Vendors' operating system software. In a customer facing role as a senior IT specialist with IBM Global Services, her responsibilities included the development of services for WebSphere MQ within the z/OS and Windows® platform. This covered the architecture, scope, design, project management and implementation of the software on stand-alone systems or on systems in a Parallel Sysplex® environment. She has received Bravo Awards for her project contributions. Saida has a degree in Computer Studies and her background includes z/OS systems programming. Saida supports Women in Technology activities and contributes and participates in the their meetings.

Martin Cernicky is an IT Specialist from the IBM Software Services in the Czech Republic and has 16 years of experience in IT. He joined IBM in 1995 and is currently working in the pSeries® support team. He has good knowledge of

AIX/pSeries systems and solutions and his additional responsibility for the last five years, is supporting WebSphere MQ middleware messaging and WebSphere Message Broker. Martin has designed and implemented solutions using the WebSphere MQ family products and has experience in the implementation of messaging and broker solutions on distributed platforms, especially on AIX® and Windows. He is a co-author of the IBM Redbook *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198. Martin holds a degree in Automated Technology Systems at Czech Institute of Technology.

Alywin BC Ching is an Advisory IT Specialist working with ITS, IBM Singapore. He has over 15 years of experience in IT. His expertise is in both application programming and systems support. His primary role in IBM is to provide support on IBM messaging products for South East Asia customers. He holds a degree in Computing Science from Staffordshire University, United Kingdom.

Brian McCarty is a Lead Systems Programmer from Texas with USAA. He has ten years of experience with integration and messaging solutions primarily in areas of banking, investment and insurance. His interests reside in developing enterprise scale applications specializing in the use of WebSphere Application Server, MQ and Message Broker. Previously he co-authored the IBM Redbooks *WebSphere MQ Integrator Deployment and Migration*, SG24-6509 and *Messaging Solutions in a Linux Environment*, SG24-6336. Brian holds a Masters Of Science in Computer Information Systems.

Gregorio Patino is a Consulting IT Architect at PRAGMA, in Medellin, Colombia and has 12 years of experience in Information Technology. In the last five years, he has lead the design and development of J2EE™ custom software solutions, software architecture, integration projects and Identity Management solutions for the leading Colombian companies. His areas of expertise include J2EE software development, software architecture, distributed systems, EAI and SOA and Identity Management. He has already authored the IBM Redbook called *WebSphere Adapter Development*, SG24-6387. Gregorio holds a degree of Systems Engineering from EAFIT University, and for the last four years he has been teaching postgraduate courses at this university.

Amar Shah is currently a technical team lead for the Message Broker change team, IBM India Software Lab. His responsibilities include resolving customer problems, defect fixing on Message Broker along with the process improvement and serviceability improvement and working on Proof of Concept. He is also acting as a single point of contact for HSBC GLT, Pune, India for Message Broker. Prior to working for the Message Broker change team, he worked for AIX Operating System change team for three and a half years. He has good knowledge of AIX OS and databases. He has eight years of experience in IT. Amar joined IBM in 1998. He holds a Master of Science (M.S.) degree in software systems from Birla Institute of Technology (BITS), Pilani, India.



*Figure 1 From left, back row: Amar, Brian, and Gregorio
Front row: Aywin, Saida, and Martin*

The team would like to thank the following people for their assistance and contributions to this redbook:

Rachel Alderman with Toolkit update without internet access
IBM Hursley

Mark Hiscock with Toolkit SSL encryption
IBM Hursley

Colin Paice with IP13 Support Pac
BM Hursley

H R Sridhar with SSL configuration
IBM India

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will team with IBM technical professionals, Business Partners, or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbook@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



Part 1

Introduction

This part of the book provides a brief overview of the WebSphere Message Broker product. It discusses the features of Version 6 and important considerations about the environment that you must provide to administer the message broker platform. This part includes the following chapters:

- ▶ Chapter 1, “Overview of this book” on page 3
This chapter defines the scope of the book, the targeted audience for the book, and the business relevance of the topics of this book inside any organization.
- ▶ Chapter 2, “WebSphere Message Broker features” on page 7
This chapter provides an overview of WebSphere Message Broker V6.0 and gives a brief discussion of the components of the product and their functionality.
- ▶ Chapter 3, “WebSphere Message Broker environment” on page 15
This chapter discusses typical considerations of an organization in order to provide a stable environment for administering the life cycle of the WebSphere Message Broker product and its applications. This chapter also provides an introduction to the management areas of message broker and their relevance.

Overview of this book

This chapter provides an overview of how this book is organized and how you can find the information that we present. It also describes the intended audience and gives details on the business relevance for the topics that are covered. This chapter discusses the following topics:

- ▶ The scope of the book
- ▶ Audience for this book
- ▶ Assumptions and prerequisites
- ▶ Business relevance

1.1 The scope of the book

This book provides useful information regarding the management and administration of the WebSphere Message Broker in a production environment.

It focuses on the management and administration of WebSphere Message Broker. Although we know that the life cycle of application integration includes designing, developing, and deploying applications, these topics are beyond the scope of this book. We discuss those topics only in relation to the administrative tasks.

The book also provides information about how to maintain and manage the WebSphere Message Broker middleware platform. How to configure or manage the development, testing, or quality assurance environments is also beyond the scope of this book.

1.2 Audience for this book

Part 1, “Introduction” on page 1 provides a general overview of WebSphere Message Broker and helps you to understand the concepts around the product and the advantages of using this product within an IT environment. It also contains relevant information for architect roles because it presents typical environments that are used by organizations to manage the WebSphere Message Broker product and the applications that run on top of it. It provides a brief introduction about all the aspects an architect or a manager has to think about in order to properly administer the Message broker platform.

Part 2, “Best practices” on page 41 can help message broker administrators understand the best practices in some of the most relevant management areas of the product. It describes the best practices in managing resources, security, backup and restore, monitoring and health checking, maintenance strategy, and problem determination. We provide a sample scenario at the end of the book to illustrate some of the practices that we describe and their implementation.

This book is not intended to provide useful information for beginning users of WebSphere Message Broker and is also not intended to provide information about migrating to WebSphere Message Broker from previous versions of the product. You can find detailed information about how to use WebSphere Message Broker V6.0 in *WebSphere Message Broker Basics*, SG24-7137 and also in the product documentation. For information about how to migrate to WebSphere Message Broker V6.0 from previous versions, see *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198 and the product documentation.

1.3 Assumptions and prerequisites

We make the following assumptions about the readers of this book:

- ▶ You are familiar with the WebSphere Message Broker V6.0 product or at least with a previous version of the product.
- ▶ You are familiar with WebSphere MQ.
- ▶ You have adequate experience on how to configure WebSphere Message Broker.
- ▶ You have all the necessary privileges on all the components that are involved in configuring and administering the Message broker platform, including operating systems privileges or database privileges where appropriate.

1.4 Business relevance

Organizations these days, are always searching for better ways to adapt to the changing conditions of their environments. Business managers are constantly planning and designing new strategies that aim their companies to achieve higher customer satisfaction, to better quality standards, to enhance their process automation, and to lower their operational costs. These strategies can include:

- ▶ Establishing a connection with the customers, partners, and suppliers.
- ▶ Merging or acquiring a new company (normally with different IT infrastructures).
- ▶ Identifying market trends and opportunities and reacting quickly to them.
- ▶ Outsourcing or integrating geographically disperse centers with language and cultural differences.

IBM has a complete portfolio of products that are focused on providing the solution of all the business integration needs. Business integration means coordination and cooperation of all business processes and applications. It involves bringing together the data and process intelligence in an enterprise and harnessing these so that the applications and the users can achieve their business goals.

WebSphere Message Broker is one of the key products in the business integration scenario. It provides a powerful message broker solution that is driven by business rules. It routes and transforms messages between applications, even if these applications handle their own format that is written in different programming languages or is running in distinct operating system

platforms. The applications do not need to know anything except their own conventions and requirements (in terms of information exchange).

WebSphere Message Broker provides easy integration of applications and processes in an organization, using message and data transformations in a centralized place (the broker) without interventions in the code of the remaining applications. It also provides the capability of extending the information technology infrastructure to customers and suppliers by meeting all their requirements inside the broker configuration. In addition, it provides different ways of interacting between applications (point-to-point and publish/subscribe), a centralized administration of all the applications integration and facilities to extend the functionality of the product through the integration with the Tivoli® family of products and application programming interfaces.

WebSphere Message Broker features

This chapter provides a high-level introduction to WebSphere Message Broker and describes the main capabilities and components of the product.

This chapter includes the following topics:

- ▶ Capability and environments
- ▶ Components description and function

2.1 Capability and environments

WebSphere Message Broker is a powerful information broker that allows business data and information in the form of messages to flow between disparate applications across multiple hardware and software platforms. Business rules can be applied to the data flowing through the message broker to route, store, retrieve, and transform the information.

2.1.1 Capabilities description

The primary capabilities of WebSphere Message Broker are message routing, message transformation, message enrichment, and publish/subscribe. WebSphere Message Broker provides connectivity for both standard-based and nonstandard-based applications and services. The routing can be simple point-to-point routing, or it can be based on matching the content of the message to business rules that are defined to the broker.

Transformation and enrichment of in-flight messages is an important capability of WebSphere Message Broker. This capability enables business integration without the need for additional logic in the applications. Messages can be transformed between applications to use different formats (for example, from a Custom Wire Format in an existing system to XML for use with a Web service).

Messages can also be transformed and enriched by integration with multiple sources of data such as databases, applications, and files to perform any type of data manipulation including logging, updating, and merging. Business information from messages flowing through the broker can be stored in databases or can be extracted from databases and files and added to the message for processing in the target applications.

The publish/subscribe is an alternative method to manipulate messages. The simplest method of routing messages is to use point-to-point messaging by sending messages directly from one application to another. With publish/subscribe, messages are sent to all applications that have subscribed to a particular topic. The broker handles the distribution of messages between publishing applications and subscribing applications. Applications can publish on or subscribe to many topics as well as apply more sophisticated filtering mechanisms.

2.1.2 Environments description

WebSphere Message Broker is comprised of two principal parts (Figure 2-1):

- ▶ A runtime environment that contains the components for running the broker applications that are created in the development environment
- ▶ A development environment for the creation of message flows, message sets, and other broker application resources

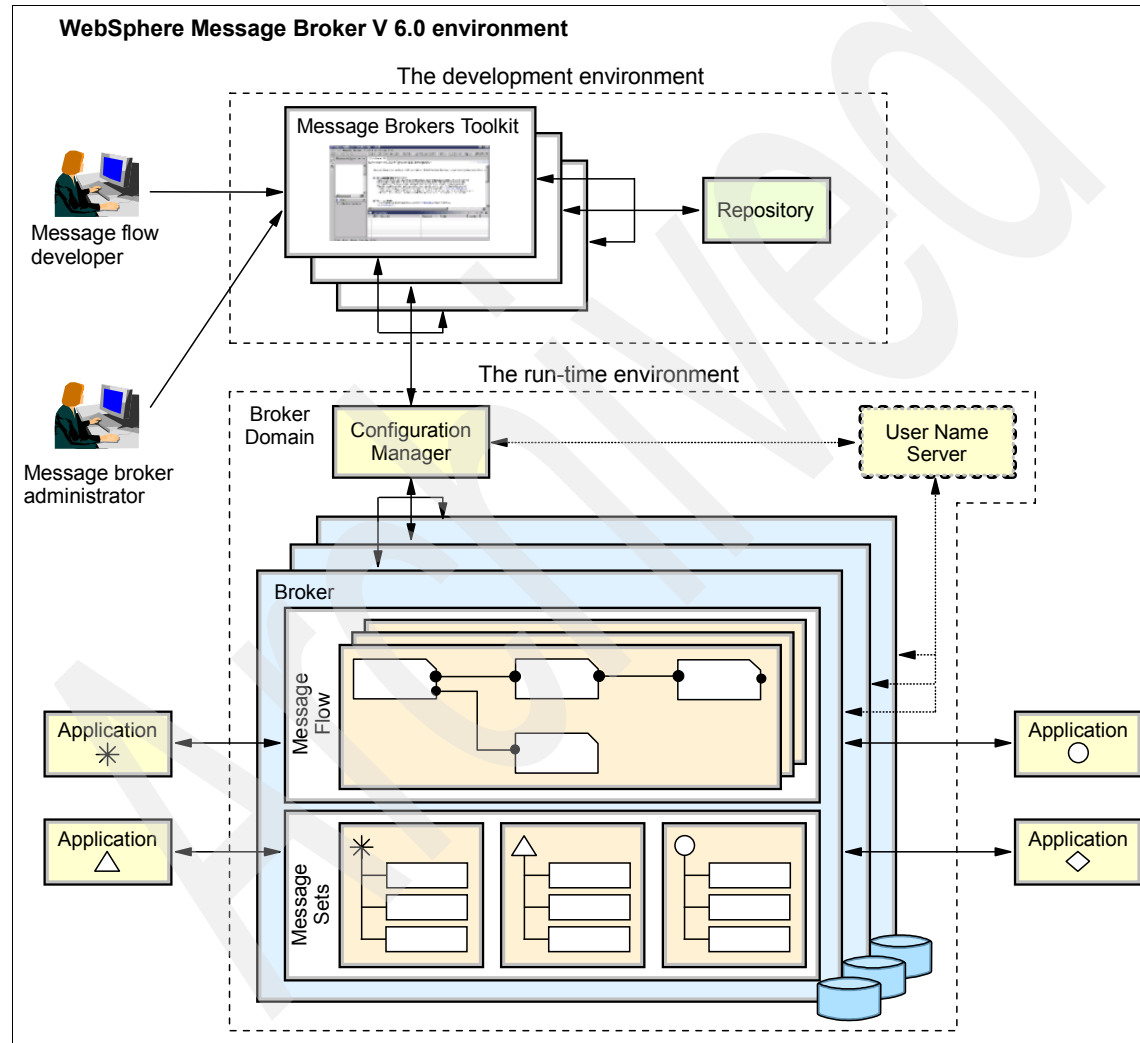


Figure 2-1 WebSphere Message Broker V6.0 environment

The runtime environment is the set of components that are required to deploy and run message flow applications in the broker. The main components of a runtime environment are the *broker*, the *Configuration Manager*, and the *User Name Server*.

Components are grouped together in broker domain. The brokers in a single broker domain share a common configuration that is defined in the Configuration Manager. A broker domain contains one or more brokers, a single Configuration Manager and one or more Message Brokers Toolkit. It can also contain a User Name Server. A broker can only belong to one broker domain. A Message Brokers Toolkit can connect more domains.

All components of WebSphere Message Broker communicate using WebSphere MQ communications protocol. Applications can use either WebSphere MQ protocol or other protocols, such as WebSphere MQ, WebSphere MQ Mobile, JMS, HTTP/HTTPS, Telemetry, Real-time, and Multicast.

The development environment is where the message flow applications that provide the logic to the broker are developed. The broker uses the logic in the message flow applications to process messages from business applications at run-time. In the Message Brokers Toolkit, you can develop both message flows and message sets for message flow applications. The main component of a development environment is Message Brokers Toolkit.

Message flows are applications that provide the logic that the broker uses to process messages from business applications. Message flows are created in the Message Brokers Toolkit using the graphical editor to click and place nodes and connect them. Each node performs some basic logic, and a selection of nodes are provided to perform particular tasks that can be combined to perform complex manipulations and transformations of messages.

A message set is a definition of the structure of the messages that are processed by the message flows in the broker. The message broker must know the structure of a message in order to enable a message flow to manipulate or transform the message. This definition can then be used for verification of the message structure in the message broker, and within the Message Brokers Toolkit to assist with the construction of message flows and mappings.

2.2 Components description and function

WebSphere Message Broker consists of four components:

- ▶ The broker
- ▶ The Configuration Manager
- ▶ The User Name Server
- ▶ The Message Brokers Toolkit

In this section, we describe briefly the functions of each component.

2.2.1 Broker

The broker is a set of execution processes that hosts and runs message flows. When a message arrives at the broker from a business application, the broker processes the message before passing it on to one or more other business applications. The broker routes, transforms, and manipulates messages according to the logic that is defined in its message flows.

Message flows runs in the broker under execution groups. The execution groups enable message flows within the broker to be grouped together. Each broker contains a default execution group, and more execution groups can be created as long as unique names are given to them within the broker. Each execution group is a separate operating system process so that the contents of an execution group remain separate from the contents of another execution group within the same broker. Having each execution group as a separate operating system process can be useful for isolating pieces of information for security. These message flows then execute in separate address spaces or as unique processes. Broker applications such as message flows and message sets are deployed to a specific execution group, but the same message flow and message set can be run in different execution groups in order to enhance performance.

Each broker requires a queue manager to communicate with other components in broker domain. Broker communicates with the Configuration Manager from which it receives configuration information, with other brokers to which it is associated and with User Name Server if used.

Each broker requires a database in which it stores the information in order to process messages at run-time. A connection to the user databases can be configured in the broker to manipulate data and messages between the broker and these databases.

2.2.2 Configuration Manager

The Configuration Manager is the interface between the Message Brokers Toolkit and the brokers in the broker domain. It stores configuration details for the broker domain in an internal repository, providing a central store for resources in the broker domain. The Configuration Manager is responsible for deploying message flow applications to the brokers. It also reports back on the progress of the deployment and the status of the broker.

The user can work with Configuration Manager in the following three ways:

- ▶ Use Message Brokers Toolkit which is the main WebSphere Message Broker tool. It is described in 2.2.4, “Toolkit” on page 13.
- ▶ Use commands from operating system command line.
- ▶ Use Configuration Manager Proxy API which is described in the next section.

Configuration Manager requires a queue manager to communicate with other components in broker domain. Configuration Manager communicates with all the brokers to send configuration information and with User Name Server if used.

Configuration Manager Proxy API

The Configuration Manager Proxy (CMP) is a Java™ application programming interface to the Configuration Manager. It enables programs to be written to automatically create and administer broker domains. Java objects map to domain objects to make the Configuration Manager Proxy API simpler to code. A sample API Exerciser is supplied to demonstrate the capabilities and how to program an application to use them.

You can use the Configuration Manager Proxy API for the following tasks:

- ▶ Viewing domain objects
- ▶ Adding and removing brokers
- ▶ Modifying broker properties
- ▶ Setting up Access Control Lists
- ▶ Backup and restore of domain
- ▶ Editing and deploying topology
- ▶ Editing and deploying topics
- ▶ Querying and deleting active subscriptions
- ▶ Creating and deleting execution groups
- ▶ Deploying bar files
- ▶ Viewing deployed resources
- ▶ Querying logs
- ▶ Starting and stopping message flows
- ▶ Controlling user trace

The available administration functions are comparable to those found in the Message Brokers Toolkit and the command line, and can be a powerful tool for managing administration of the domain if desired.

2.2.3 User Name Server

A User Name Server is an optional component that is only required where publish/subscribe broker applications are run, and where extra security is required for applications to publish or subscribe to topics. The User Name Server provides authentication for topic-level security for users and groups performing publish/subscribe operations.

User Name Server requires a queue manager to communicate with other components in broker domain. User Name Server communicates the Configuration Manager from which it receives configuration information and with the brokers which require extra publish/subscribe security.

2.2.4 Toolkit

The Message Brokers Toolkit is the main GUI for the WebSphere Message Broker. When it connects to the Configuration Manager, the status of the brokers in the domain is derived from the configuration information stored in the Configuration Manager's internal repository.

It is used for:

- ▶ Broker domain administration
- ▶ Message flow and message set development
- ▶ Deploying message flows and message sets to the broker

The Message Brokers Toolkit is built on the Rational® Application Development Platform. This is an Eclipse-based Unified Modeling Language (UML) visualization and Java Development tool that has been extended with tools specifically for WebSphere Message Broker. These tools are grouped into a few views called perspectives.

The Broker Administration perspective is used for the administration of the broker domains defined in the Message Brokers Toolkit. This perspective is also used for the deployment of message flows and message sets to brokers in the defined broker domains.

The Broker Application Development perspective is used to design and develop message flows and message sets. This contains editors to create message flows, transformation code such as ESQL, and message definitions.

The Debug perspective is used for application developer tests and debug message flows. This contains editors to create a debug connection to the brokers and set up message flow breakpoints to check processed messages and data.

WebSphere Message Broker environment

This chapter discusses the considerations that are related to the environments of WebSphere Message Broker inside an organization. It also provides an overview of the main aspects of administering WebSphere Message Broker, which include:

- ▶ Resource management
- ▶ Security
- ▶ Backup, restore, and recover
- ▶ Monitor and health-check
- ▶ Maintenance strategy
- ▶ Problem determination

It includes the following topics:

- ▶ Overview of environments
- ▶ Managing broker domain
- ▶ Recommended considerations

3.1 Overview of environments

WebSphere Message Broker V6.0 delivers an integrated solution for your current application and information mediation needs. It also adds advanced capabilities to your enterprise service bus (ESB) by extending the integration of your service-oriented architecture (SOA) to encompass your entire enterprise requirements. WebSphere Message Broker V6.0 enables you to connect disparate applications and business data across multiple platforms and to provide transformation and intelligent routing capabilities for the business information.

In most organizations, the WebSphere Message Broker environments are divided into development, acceptance, test, and production environments. This section discusses briefly best practices for WebSphere Message Broker in these respective environments.

Figure 3-1 shows a typical WebSphere Message Broker V6.0 environment that spans development, acceptance, and production for an organization.

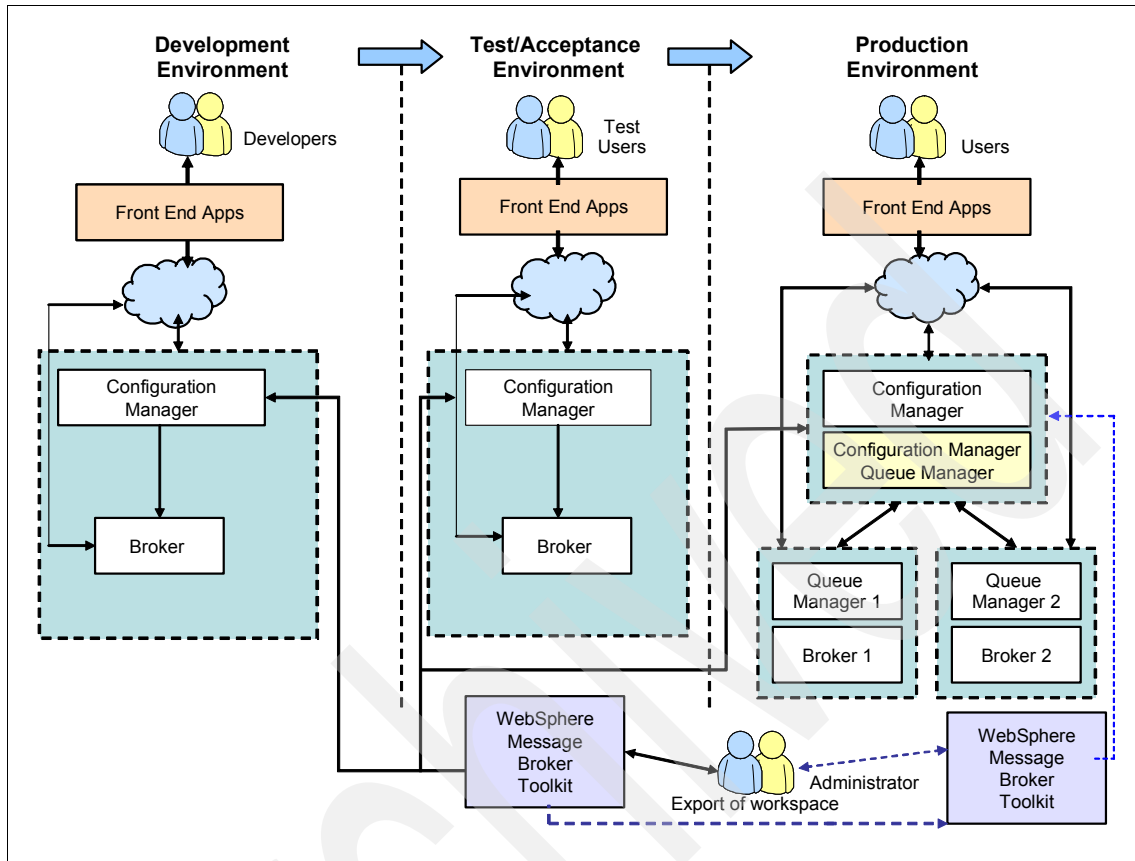


Figure 3-1 Environment overview

3.1.1 Development

In a development environment, it is a common practice for a developer to develop message flows using a toolkit and to perform unit testing on the message flows and message sets. When developing new applications, the developer can use the versioning facility to tag each change with a version. This method of making changes helps in change management activities.

Tip: In the development environment, an application developer is responsible for designing message flows and creating or amending elements that require programming. Alternatively, the system administrator is the person who is responsible for configuring and managing the operation and functionality of the message broker environment. In some small-medium-business (SMB) organizations, a developer and administrator can be the same person. For an organization that has these roles divided, it is very important to have both the developer and the administrator working together when planning for changes or when introducing new applications to the message broker environment. Working together ensures that the entire message broker environment is managed to the optimum to meet the organization requirements.

3.1.2 Test and acceptance

When the development is completed, these message flows and message sets move to the various test and acceptance environments for testing iterations such as systems integration, user acceptance, fault, and performance testing. On the acceptance test server, these message flows and their related message maps and message sets are tested for the functionality and performance and to ensure that they meet business requirements. In the case of test failures, the results are fed back to the developers, who then make the necessary changes to the message flow artifacts. When you have tested these changes for the respective applications and they have passed the in-house quality assurance process, you can move them to the production environment.

3.1.3 Production

With all the requirements met, the message flows and the related message maps and message sets are imported to the production environment for production usage. To better manage the production environment, a number of new administration and management utilities can be used in WebSphere Message Broker V6.0, as discussed in 3.2, “Managing broker domain” on page 18.

3.2 Managing broker domain

In this section, we discuss the following aspects of managing the broker domain:

- ▶ Resource management
- ▶ Security
- ▶ Backup, restore and recover
- ▶ Monitoring and health-check
- ▶ Maintenance strategy
- ▶ Problem determination

3.2.1 Resource management

Management of WebSphere Message Broker resources includes:

- ▶ Managing changes in the domain
- ▶ Managing problems in the domain
- ▶ Administering the domain
- ▶ Implementing RAS

Managing changes in the domain

The change management is an important process in managing the broker domain. The following sections describe briefly some of the important aspects of change management activities that are discussed in this book.

Versioning

The enhanced versioning capabilities that were introduced to the development environment in the last release have been extended to the runtime environment. All resources deployed can be tagged with version, author, and other useful information in addition to the standard compile time and deployment time attributes. The new information is displayed in the administration interface, making it easy to see which resources have been deployed to production environment. For specific examples of its usage, see 4.1.1, “Versioning” on page 44.

User Defined Properties (UDPs) are defined when constructing a message flow using the Message Flow editor. These properties are used by the Extended Structured Query Language (ESQL) program inside a certain type of message flow node (such as a Compute Node). The advantage of UDPs is that they provide the flexibility to change the values at deployment time that can vary between targeted runtime environments. Changes are not needed in the application programs. It is important to track and record these UDPs when using them to provide references to the developers or administrator when needed. Further details about UDP are described in 4.1.1, “Versioning” on page 44.

Note: You can also use this property as a Java property inside the Java code that is generated by a JavaCompute node.

Procedures and Documentation

It is important to document all changes, to communicate the activity to those within the scope of the change, and to archive it for later for audit or for backout purposes. While this activity is highly dependent on the specific organizations usage of change and release management, an introduction to best practices for this activity as it relates to WebSphere Message Broker is included in 4.1.2, “Procedures and documentation” on page 47.

Automation

Automation of broker management processes involve the aspects of version and source code control system, builds, deploys and applying maintenance to the software. It is important that all of these activities are documented for future references. Further details are described in 4.1.1, “Versioning” on page 44.

Managing problems in the domain

It is a good practice to define a problem management process for all the environments as discussed in 3.1, “Overview of environments” on page 16. A formal or standardized problem management process creates a systematic approach to counter an unexpected events when they occur. For more information, see 4.2, “Managing problems in the domain” on page 50.

Administering the domain

This section discusses several ways to manage the WebSphere Message Broker V6.0 resources.

Note: While discussing the various administrative tasks in this chapter and in Chapter 4, “Resource management” on page 43, several references are made to the tools Configuration Manager Proxy API and Command Assistance Wizard.

Configuration Manager Proxy API

The Configuration Manager Proxy API (CMPAPI) is a set of Java classes that are packaged in a single .jar file. The CMPAPI allow Java applications to interact with the WebSphere Message Broker components including the Message Brokers Toolkit and Configuration Manager Proxy API Exerciser. In addition to using the delivered user applications, you can use the CMPAPI to develop custom administrative and management tools.

Tip: Before using the CMPAPI, first ensure that the user ID that you use to connect to the Configuration Manager has the appropriate access or permission to connect to the Configuration Manager queue manager and that you use the underlying WebSphere MQ objects. Also, verify that the user ID has the appropriate authority to manipulate broker domain objects.

Command Assistant Wizard

Command Assistant wizard provides another method for administering the broker runtime, as an alternative to the standard command line utilities. A review of this feature is discussed in “Command Assistant wizard” on page 75.

Automation

Automating, or tooling, to create reusable processes for change management and other administrative tasks such as controlling components is reviewed in detail in Chapter 4, “Resource management” on page 43.

Implementing RAS

RAS is the acronym for reliability, availability, and scalability. The following sections describe briefly the definition for each area.

Reliability

WebSphere Message Broker V6.0 is a business integration products that coordinates and cooperates all your business processes and applications. Thus, the reliability of WebSphere Message Broker V6.0 is critical to your entire business organization. For more information about reliability, see 4.4.1, “Reliability” on page 53.

Availability

Availability as it relates to the discussions in this book is concerned with developing runtime solutions that are available continuously for the desired length of time. Implementing availability solutions in a WebSphere Message Broker environment has many possible solutions. Each of the possibilities has their own benefits and drawbacks. For more information about the aspects of potential solutions and the considerations that you should make, see 4.4.2, “High Availability” on page 54.

Scalability

The inherent ability for WebSphere Message Broker to scale well makes the task of ensuring sufficient capacity and performance one of the simpler tasks for administrators to perform. Several options exist to scale the runtime resources both vertically and horizontally. We discuss these options in detail in 4.4.3, “Scalability” on page 58.

3.2.2 Security

A secure WebSphere Message Broker V6.0 production environment is assessed from three perspectives:

- **Authentication**

Authentication is a security process to determine who or what a service requestor is declared to be. For example, authentication done through a log on screen. In this case, the user is required to use identification and a password. This kind of security is not bullet proof because the password can be lost, stolen, or revealed accidentally. In a WebSphere Message Broker

V6.0 production environment, you can enhance authentication in several ways. We describe the authentication mechanisms that WebSphere Message Broker supports in 5.1.1, “Authentication” on page 84.

► Authorization

Authorization is a security process to find users who are authorized to access the WebSphere Message Broker V6.0 resources after those users are identified. It is the process of granting or denying access to a system resource and ensuring that users who attempt to work with those resources have the necessary authorization to do so.

The main features of WebSphere Message Broker to handle authorization are:

- Groups management
- ACL administration
- Topic based security

These topics are discussed in detail in 5.1.2, “Authorization” on page 89.

► Secure Sockets Layer (SSL) support

WebSphere Message Broker uses SSL to enhance three types of communications:

- Between the message broker and external web services
- Between Java MQ client applications and the message broker
- In the Real-time node processing

The SSL support of the WebSphere Message Broker is discussed in 5.1.3, “Secure Sockets Layer” on page 96.

Other options

In addition to the WebSphere Message Broker security offering, there are complementary options that are important in a production message broker environment. You can find an introduction to these options in 5.1.5, “Complementary options” on page 107.

3.2.3 Backup, restore, and recover

In a WebSphere Message Broker V6.0 production environment, it is very important to maintain a precise backup and restore strategy. This strategy offers the ability to recover the WebSphere Message Broker V6.0 domain quickly and efficiently to ensure continuity and availability.

The following is the list of components that are required to be backed up in a typical WebSphere Message Broker V6.0 production environment:

- ▶ Broker database
- ▶ Configuration Manager
- ▶ User Name Server
- ▶ Message flow files
- ▶ Message set definition files
- ▶ ESQL files
- ▶ Mapping files
- ▶ XML Schema files
- ▶ Broker archive files

Chapter 6., “Backup, restore, and recover” on page 115 further discusses the backup and recovery steps for these components.

3.2.4 Monitor and health-check

Keeping a check on the health of message broker components is an important discipline that you must consider to keep the production system in predictable condition. In the situation where message broker components availability, reliability, and performance results in inadequate operation controls, you need a process to improve controls. It is also necessary to establish a process to ensure that the controls that are in place are in fact effective. Effective controls ensure that the production environment meets the availability, reliability, and performance criteria that is expected by users that are defined by the Service Level Agreement (SLA).

Availability

Application availability is an important requirement in a 24x7, end-to-end solution. In today's business environment, many organization are demanding higher levels of system (application, software, and hardware) availability. Therefore, it is important for an organization to have a proactive measure in place, either to counter, to reduce, or to potentially eliminate unplanned outages. This proactive measure is achieved through monitoring and regular health-checks on the overall system. For this, the WebSphere Message Broker environment is no exception.

Reliability

WebSphere Message Broker V6.0 is a business integration products that coordinate and cooperate all your business processes and applications. It involves bringing together the data and process intelligently in your organization and harnessing these so that your applications and users can achieve their business goals. This objective makes WebSphere Message Broker reliability

critical to your entire business organization. To ensure that all WebSphere Message Broker runtime components are in top condition, a health-check process is a must have on regular basis to ensure no unplanned outages.

Performance

Problems with performance of WebSphere Message Broker is usually a measure against the message throughput. In most production environments, performance is normally the major concern. When WebSphere Message Broker performs well, it can reduce the processing cost significantly. When the performance level message flow falls, it is usually due to one of the following reasons:

- Lack of resources running the message flow

Message broker normally requires CPU resources intensively when processing a message. When a message broker process a message, it involves message parsing, manipulation of message data, data storage, and creation of output data. Message flows with lots of I/O processing need more CPU time (for example read, write, delete, and update from the database). Care has to be taken if there is already a high level of persistent message processing taking place within the broker queue manager. The additional load of persistent message impacts the CPU and memory resources. Therefore, it is important to monitor system resources usage (for example, CPU, memory, and disk I/O activity) of the production environment regularly.

For brokers that use the database as a repository, you must ensure sufficient space in the file system is located in the respective brokers' databases.

- Environment not well tuned

It is important that the health-check is done to ensure that the environment to run WebSphere Message Broker is in optimum condition. If an environment is configured poorly, even the most efficient algorithm and coding within the message flow can still struggle to run efficiently. Message broker and its dependent components, such as broker database engine and queue manager, are no exception. Their activities also need to be monitored and tuned properly.

- Bad application design

The sequence of message flows or how applications interact can effect system performance. You should take note of the response time and the message throughput that is provided by all the components. There are times when it might appear that the message broker components are causing the delay when, in fact, the problem actually might be bad application design and integration. To detect such issues, investigate the runtime performance of the message flows together with a connected applications.

- ▶ Inefficient processing algorithm for message flow

The practice of writing good algorithms for message processing is essential to producing efficient message flows. A bad sequence of processing nodes within message flow leads to a series of significant, necessary processing of the message flow. It is important for a developer to fully understand the facilities that WebSphere Message Broker provides to develop the message flows efficiently.

- ▶ Bad design or excessive ESQL processing within message flow

Take care when implementing a processing algorithm in ESQL within a message flow. Find the most efficient order to get the highest level of performance from a message flow.

Monitoring and health-check tasks

There should be a formal discipline to ensure that monitoring and health-check tasks are in place for the WebSphere Message Broker V6.0 environment. These tasks are:

- ▶ Frequency
- ▶ Archiving
- ▶ Reporting
- ▶ Alerting

These tasks are discussed in detail in 7.1, “Basic of monitoring and health-checking” on page 136.

Monitoring the domain

Basically, monitoring tasks can be categorized into monitoring the systems for failure events and monitoring systems for performance, as discussed in detail in 7.2, “Monitoring the domain” on page 139.

The following discuss briefly the areas that are used to assist the monitoring and health-check activities for a WebSphere Message Broker V6.0 domain.

System and components logs

To ensure application availability and message broker components, availability and reliability are maintained in a production environment, information gathered from the local system error logs, message broker logs, Toolkit logs, queue manager logs and database logs are used as the input to a planned health-check process. When potential problems are identified, you can take proactive steps to correct these issues.

WebSphere MQ component attributes

In a WebSphere Message Broker V6.0 environment, the queue manager serves as the base component to the overall infrastructure. We must ensure the queue manager is in top conditions. One way to monitor how efficient a queue manager is by examining the activities of a queue. For example, you can use a queue attribute such as CURDEPTH to determine the throughput of the specific queue manager.

The WebSphere MQs queue efficiency can also be done programatically. You can use the attributes in the message header, for example PutTime, to determine the turnaround time for one particular message.

Broker

There are several broker processes that you can use to monitor for health-check purposes:

- ▶ bipservice
- ▶ bipbroker
- ▶ biphttplistener
- ▶ DataFlowEngine

For information about how these processes are monitored are described, see 7.4.1, “Broker” on page 146.

Accounting and Statistics

Message flow accounting and statistics data is the information that can be collected by a broker to record performance and operating details of message flow execution. Message flow accounting and statistics data records dynamic information about the runtime behavior of a message flow. For example, it indicates how many messages are processed and how large those messages are as well as CPU usage and elapsed processing times. Data collection is started and stopped dynamically when you issue the **mqsi** **change** **flow** **stats** command. The accounting and statistics data provides information at message flow, thread, and node level.

Accounting and statistics data is collected only for message flows that start with an MQInput, HTTPInput, or user-defined input node. If the data collection is started for a message flow that starts with one of these nodes, the data is collected for all built-in and user-defined nodes, including those in subflows. If the message flow starts with another input node (for example, a Real-timeInput node), no data is collected and no error is reported.

The data collection can be activated on both production and test systems. If the default level of statistics (message flow) are collected then the impact on broker performance is minimal. However, collecting more data than the default message

flow statistics can generate high volumes of report data that might cause a small but noticeable performance overhead.

System resources

It is important to understand the newly developed message flows requirements. Ensure these message flows are not starved from resources, for example CPU, I/O and memory.

You can monitor these resources using the external tool, such as IBM Tivoli OMEGAMON®. IBM Tivoli OMEGAMON is a product that delivers availability monitoring with a GUI interface for the production environment management. It can identify common system problem and automate corrective action using predefined industry best-practice situations. For more information, see 7.3.1, “Tools” on page 145.

3.2.5 Maintenance strategy

Creating a maintenance strategy is an important aspect of a production environment. The strategy should ensure that the environment maintains the availability, reliability, and performance criteria.

The following tasks trigger a maintenance process to take place:

- ▶ New release
- ▶ Fix pack
- ▶ Refresh pack

For each of these, rolling out a maintenance task on a production system requires proper planning. You must consider a feasibility study, plan and manage the installation processes, test these items in the development and test environments, schedule and plan for a production maintenance window, and implement a recovery plan if an unforeseen issue occurs during or after the installation process.

Important of maintenance

It is important for an organization to administer the broker domain and to ensure that all current components are up to date. Before a fix pack or refresh pack is scheduled for production environment, it has to be tested on the development environment and then tested in the test environment, where the all the activities are managed by a methodology. In this topic, for discussion, we use a simple maintenance methodology to describe how the WebSphere Message Broker maintenance process is carried out in a typical production environment.

Reviewing and feasibility study of a fix pack

In some organizations, fix packs are installed only when a problem occurs. This methodology, however, is not recommended because this practice normally causes a great deal of confusion and panic to an organization. With this chaotic situation, the plan to update the fix pack can be quite error prone.

We recommend that you schedule fix pack updates on a regular basis (for example, every three months). It is important to review the readme files that are provided with the fix packs. You should go through all the documented changes and check them against your message broker environment and note issues that you think might be related to your environment.

We recommend that you contact IBM Support if you need further clarification for an issue. We also suggest that you search the Web site to see if there is any issue that spawned from the fix pack itself. We also recommend that an administrator check the WebSphere Message Broker V6.0 Web site regularly for new updates. You can use your archived error logs to determine whether there are errors that can be resolved by the fix pack.

Decision making

With all the data gathered, you should have enough information to decide whether to proceed with the installation of the fix pack. In some organizations, business continuity is critical, and the decision to proceed with the fix pack update must be brought forward to management for approval. In these cases, the documentation that is submitted to management is supported by technical information (for example, the reason for the upgrade, the estimated down time caused by the upgrade, how the upgrade will impact the business, and the recovery plan if one is needed).

Planning for the fix pack installation

When you as an administrator or your management have agreed to move on with an update, the next step to take is to plan for the update activities. In this stage, a few considerations are needed to be taken. For example, you have to decide when and what order to take to update the message broker components. Also, you must derive a strategy to update the components where dependencies among the message broker components are critical. These strategies are discussed in 8.1, “Maintaining the domain updates” on page 154.

Here is a list of considerations for message broker components update:

- Broker

It is recommended that you have a broker, the Configuration Manager, and the User Name Server at the same fix pack level. When planning for broker updates, you must consider Rational Agent Controller, WebSphere MQ, and

the repository database for the broker. Ensure that you include these components when you are updating the broker component with the new fix pack.

- **Configuration Manager**

Performing the update of the fix pack to the Configuration Manager is straight forward because there is only one subcomponent — WebSphere MQ.

- **User Name Server**

Performing the update of the fix pack to the User Name Server is similar to the Configuration Manager. It is very straight forward because it only involves one sub component — WebSphere MQ.

- **Toolkit**

The Message Brokers Toolkit can be upgraded at a more granular level. The Toolkit updates are provided by Rational Product Updater. It is part of Rational Application Developer, which is shipped with Message Brokers Toolkit. One main consideration when performing an update to the Message Brokers Toolkit is you must check for Rational Product Updater updates first before attempting to upgrade the Toolkit.

Fix pack testing

Testing a fix pack is a critical process. The tester has to plan and draft the testing conditions to test against the message broker environment. The tester must have a good knowledge of the organization's hardware infrastructure, installed applications, and in-house applications behaviors.

A good test plan should cover from the fix pack upgrade procedures to application testing. It is always recommended to set the test environment as close as possible to the production environment to ensure that the test results match the production environment conditions.

After the broker has been updated, the users perform unit, integrated, and functional tests to ensure the creditability and reliability of the message broker applications. In some organizations where the environments have failover resources, failover resources are included into the entire system test to ensure everything is in order after the update.

Production fix pack update

It is recommended that you take appropriate backups before every fix pack update in the WebSphere Message Broker domain. The backup strategy is discussed in detail in 6.1, “Backing up the domain” on page 116. This backup strategy ensures a working backup system is available in an event of mishap or unforeseen problems. The details about how to recover the WebSphere

Message Broker components are discussed in 6.2, “Restoring and recovering the domain” on page 118.

3.2.6 Problem determination

Effective and systematic problem determination combines a logical thought process with technical diagnostic skills to discover the root cause of a problem. Using a logical thinking process for problem determination is a skill that is developed more effectively through experience than through formal training.

This section discusses a few of the aspects of logical problem determination. You can develop technical skills for problem determination through formal training. Chapter 9, “Problem determination” on page 163 provides a detailed discussion on several of the technical skills and techniques that are needed for effective problem determination with the message broker.

Logical thought process

You can initiate a logical thought process for WebSphere Message Broker problem determination by asking very specific questions. Some questions that you should pose are:

- ▶ What changes were done to the WebSphere Message Broker components or external resources since the last successful execution?
- ▶ What system administration procedures were carried out before the failure occurred?
- ▶ What changes were done to the message flows, message sets, and other artifacts since the last successful execution?

Answering questions such as these might determine the root cause of the problem. What these types of questions will always do is help to accelerate the problem determination process by narrowing the range of possibilities for applying technical diagnoses.

Technical skills

Technical diagnoses for WebSphere Message Broker problem determination can be aligned into two concept groups: basic and advanced. The skills that are defined as basic problem determination techniques are those that all message broker administrators should have. At a high level, these tasks involve:

- ▶ Identifying failing components or resources
- ▶ Being able to categorize errors
- ▶ Understanding event messages and how to interpret them
- ▶ Understanding where errors and events can be reported per operating systems

- ▶ Locating additional information
- ▶ Gathering diagnostic data for further support by IBM Support

Very specific details about these basic, technical skills are available in 9.1, “Basic steps” on page 164. Advanced techniques are those that take more experience and skill than basic ones. These tasks are implemented more rarely because they are done only when basic tasks fail to identify the root cause or are done at the request of IBM Support. These tasks involve:

- ▶ User tracing
- ▶ Service tracing
- ▶ Configuration Manager Proxy API tracing
- ▶ JVM™ tracing
- ▶ System tracing
- ▶ MQ tracing
- ▶ Analyzing abend and core files
- ▶ Advanced Database and ODBC diagnostics

IBM Support Center

In the event that you need further assistance or that you have applied basic skills and have not discovered the root cause, it might be necessary to open a problem with IBM Support. At this point, you should contact IBM Support Center and provide all the information that you collected during your problem determination process.

It is very important that you provide clear and accurate observations (to the best of your knowledge) during the course of your investigation. When you deliver clear, detailed, and concise information to IBM Support, it increases the problem determination efficiency by narrowing the range of problem possibilities.

3.3 Recommended considerations

In this section, we discuss the recommended areas to consider when configuring and fine tuning WebSphere Message Broker V6.0 environments.

3.3.1 Naming convention

A good naming convention practice for WebSphere Message Broker resources ease the making of customizing, operating, and administering activities. A good naming convention ensures that the given names are unique, where no names

are duplicated, and prevents necessary system setup confusions. This section lists the resources where naming convention activities are applied to a WebSphere Message Broker infrastructure.

Configuration Manager

When you create a Configuration Manager, give it a name that is unique on your system. Names must be unique between Configuration Manager and brokers. Configuration Manager names are case sensitive on UNIX®. It is good to base the Configuration Manager name on the queue manager name. For example, if your Configuration Manager's queue manager is named *QMRAL*, name the Configuration Manager *QMRAL_CFGMGR*.

Broker

For the broker name, give a name that is unique within your broker domain. You must use the same name for that broker when you create it on the system in which it was created initially using the command line. You must also ensure that you use the same name when a reference to that broker in the broker domain topology in the workbench. The latter is a representation of the physical broker in the configuration repository.

Broker names are case-sensitive, except on Windows platforms. Each broker requires its own queue manager, so you should use the queue manager name as the base name. For example, you can append *BRK* to the queue manager name of *QMRAL*, to give the name of *QMRAL_BRK*. With this naming convention, it is easy to associate the broker with the queue manager, because they both begin with the same prefix — *QMRAL*.

Execution groups

Each execution group name must be unique within a broker. One possible naming convention format can be *<broker_name><unique_execution_name>*, for example *QMRALBRK_EXEGRP1*. This convention eases the association between the broker and its respective execution groups and helps greatly in problem determination, especially in the product trace.

Message flows and message processing nodes

Each message processing node must be unique within the message flow that it is assigned to. For example, if you include two MQOutput nodes in a single message flow, provide a descriptive unique name for each (for example, if you have a condition where the message routing flow is set to route sales or customer service departments through a filter node). Use meaningful Output names, such as *Out_Sales* and *Out_CustSvs*. Another point to note is that each message flow name must be unique within the broker domain. Any reference to that name within the broker domain is always to the same message flow. You can assign the same message flow to many brokers.

Message sets and message flows

Each message name must be unique within the message set to which it belongs. Message set names must be unique within the broker domain. Any reference to that name within the broker domain is always to the same message set. You can therefore assign the same message set to many brokers. You can use the similar techniques as described in “Message flows and message processing nodes” when giving names to the message sets and message flow.

User Name Server

When creating a User Name Server, the similar rules naming the resources for a broker are applied. Like broker, the User Name Server requires a queue manager. Use the queue manager name as the base. For example, you can append *USRSVR* to the queue manager name of *QMRAL* to give the name of *QMRAL_USRSVR*. With this naming convention, it is easy to associate the User Name Server with the queue manager, because they both begin with the same prefix — *QMRAL*.

3.3.2 Queue manager

This section describes the important considerations that are needed when implementing and configuring the WebSphere Message Broker V6.0 queue managers.

Topology of WebSphere MQ components

There are three possible configurations where the broker queue manager applications can be configured:

- ▶ Applications and broker queue manager are located on the same system.
- ▶ Applications connect to a queue manager which is remote from the broker queue manager.
- ▶ Input and output messages for the message flow arrive over a WebSphere MQ channel.
- ▶ WebSphere MQ Client applications connect to the broker queue manager.

Each of these configurations require a different amount of message processing. The less queue managers that the application flow needs to traverse, the more efficient and cheaper to process each messages. Most production environments tend to have complex queue managers infrastructures. To measure the efficiency of a production environment, it has to be measure collectively as the performance for every queue manager will determine the overall performance. Therefore, it is important to examine the queue managers' configuration and tuning to achieve good throughput. You should explore other considerations,

such as queue manager clustering, to promote good throughput in the production environment.

Trusted channel and listener

The trusted channel and listener are the possible recommendation to increased the throughput for queue managers. Running both broker queue manager channel and listener in trusted mode increased significantly the message processing performance. It reduces both virtual and real storage requirements by allowing the listener to interact directly with the queue manager instead of via an agent process. The instruction path length is also reduced and this gives CPU saving.

Note: Trusted channel and listener are implemented only on Windows, Solaris™, and HP-UX.

The following show how trusted channel and listener are implemented on Windows and UNIX queue manager:

► On Windows

Trusted channel and listener mode is activate by setting `MQIBindType=FASTPATH` in the channels key of the queue manager's entry in the Windows registry.

► On UNIX

It is set in the respective queue manager `qm.ini` file, for example:

- CHANNEL
- `MQIBindType=FASTPATH`

Note: It is only recommended to implement trusted channel and listener in a stable environment on which there are no testing applications or user exits that might abend or need to be cancelled because trusted listener does not offer any real storage savings over the standard listener.

Hardware capability for queue manager logging

The logging activity for a queue manager affects significantly the rate at which messages can be processed. For example, when processing persistent messages, the queue manager must log the messages as they are created and updated. In addition, it concurrently processes a message flow where parsing of messages could also be done. Processing persistent messages requires CPU and I/O resources processing time. When the volume of transactions for persistent message are high, it might cause performance degradation in terms of queue manager throughput. Therefore, it is important to consider the speed of the disk I/O activities.

We recommend that you use a write cache for the disks where the queue manager log is located. With write cache, the time taken to complete the I/O is equal to the time it writes to the cache. This is achieved where the disk subsystem will copy the data from the cache to the disk at the same time the I/O is performed. Ensure the cache is nonvolatile, this is to protect the log data in the event of a catastrophic failure. This implementation increases your queue manager throughput significantly, even with persistent messages processing, because writing to the cache is faster than writing directly to the disk.

Note: On Windows and AIX, write cache is available through the use of SSA disks with a fast write nonvolatile cache. To achieve the same effect on z/OS, use a device such as the ESS 2105-E20 DASD subsystem with Feature 2121. An alternate approach is to use Solid state disks.

Queue manager log configuration

The configuration of a queue manager log varies by operating system. On Windows, Linux®, and UNIX systems, it is possible to control the following:

- ▶ The maximum size of a write to a log extent
- ▶ The size of a log extent
- ▶ The number of log extents
- ▶ The location of the log
- ▶ Whether to use circular or linear logging

The parameter to control the configuration of the queue manager log are specified within the log stanza in the registry on Windows operating system. For distributed systems such as UNIX and Linux, they are in the `qm.ini` file of a queue manager. You can find the detailed explanations for these in *WebSphere MQ System Administration Guide*, SC34-6584, which is available at:

<http://www.ibm.com/software/integration/wmq/library/library6x.html>

The maximum size of a write to a log extent and where the log is located are the most important considerations in respect of logging performance. The size and location of the file is important because the switching of the log tends to take up longer CPU processing time as it needs to create a new file for logging continuity. To save network connectivity costs, it is also recommended to have the queue manager log located on its own disk.

3.3.3 Broker

This section lists the important considerations when attempting to increase the WebSphere Message Broker V6.0 broker component's performance.

Trusted broker

On Windows, Solaris, and HP-UX, it is possible to run the broker as a trusted application. With this implementation, the WebSphere MQ API calls are processed more efficiently. To define a broker as a trusted application, use the **-t** flag on the **mqscreatebroker** command. This option sets the channels in use to fastpath mode. (We discuss the advantages of trusted channels in 3.3.2, "Queue manager" on page 33.) To modify an existing broker to make it trusted, use the **-t** flag on the **mqschangeproperties** command. To change a trusted broker back to standard mode, use the **-n** flag on the **mqschangeproperties** command.

Multiple instances and execution groups

If you have a broker which is required to run multiple copies of message flows, it is recommended that you use additional instances of the message flows within an execution group. If your environment needed greater message throughput for a message flow, use multiple execution groups. Each additional instance of a message flow is implemented as an operating system thread on Windows and UNIX, or as a Task Control Block (TCB) on z/OS. Each additional execution group has the potential to use an additional processor, which makes the processing of execution groups faster as the system processes the execution groups in parallel. With multiple instances and execution groups implementations, they give significant performance advantage.

Open handle cache

When each execution group runs, it has a cache of opened queue manager's queue handles. It is likely that you might want to increase this parameter, especially when you have many message flows assigned to one execution group. You can increase the size of the cache to a value that is greater than the typical number of queues which are being read and written. To change the cache size, use the **mqschangeproperties** command with **-v** option.

Trace

Running traces such as the product trace, for example user and service, and the ODBC trace cause processing overheads. It is important to ensure that all tracing is disabled when you do not require problem determination.

To disable the product trace, use the **mqschangetrace** command with a level of **none**. Also, make sure that all message flows and execution groups are not in trace mode. Check to confirm that Trace node is not in your message flows when

problem diagnosis is not required. When trace node is connected in a message flow path, it results in a processor and memory overhead even it does not write trace data.

The step to disable ODBC trace varies by operating system:

- ▶ On Windows environment, you control ODBC trace using the tracing tab of the ODBC function, which is available in the Control Panel.
- ▶ On UNIX, you control it through the Trace keyword of the .odbc.ini file.
- ▶ On AIX, the trace file is located in /var/wmqi/odbc/.odbc.ini.
- ▶ On Solaris and HP-UX, it is in /opt/wmqi/merant/odbc.ini.
- ▶ On z/OS, you control ODBC tracing by setting the APPLTRACE keyword in the DSNAOINI file in the broker's file system. Possible values are 0 to disable and 1 to enable.

You should also check for other products or system-related traces, such as WebSphere MQ trace, DB2® trace, or system traces, which might get enabled message flow development or problem determination.

3.3.4 Database

Database tuning can have a significant effect on message throughput depending on the composition of the message flow.

Database trace

It is critical to improve the efficiency of database processing by disabling the trace when not used. Doing so minimizes the read and write I/O time. In normal operation, ensure that the database trace is disabled. Like all other product traces, leaving the trace running on the system results in CPU, memory, and I/O overheads.

Log and data I/O time

Database update operations (such as update, insert, and delete) performance benefits if the database log is located on a fast device, such as a disk with a fast write nonvolatile cache. This location on the faster device is because these update operations require synchronous writing to the database log. The faster the writing takes place, the higher the message throughput is.

In general, when update operations are performed in a relational database to data that is not of type BLOB, the data is buffered and a physical write of the data to disk is deferred until a checkpoint takes place within the database manager.

This is recommended. However, in situation where BLOB data is concern, the data is not buffered but is written straight to disk, making the speed of the disk on which the data is located is important. In this case, using a fast device, such as one with fast write nonvolatile cache, will optimize performance.

Broker database

There is limited potential for tuning the broker database, unless you have one or more of the following situations:

- ▶ Subscribers are registering or de-registering for publish/subscribe.
- ▶ Retained publications are in use.
- ▶ Aggregate nodes are in use.

In all cases, it is recommended that you ensure that the broker database log is located on a fast nonvolatile device, when there will be insertion of data.

Business database

Business database performance is a vital key to the success of your on demand applications. It is essential that you begin with a fundamental knowledge of how to achieve the best possible performance with your business database. Database tuning can improve message flow throughput significantly. The following are some basic recommendations for database tuning.

Create view table and indexes

In a situation where read activity against a table is high from message flows, consider creating a read-only view table. The view table can be referenced in the message flow in the same way as the original table. This read-only view table reduces the amount of locking that the database manager has to perform and result in better read performance.

Keeping indexes in their own buffer pool can significantly improves performance, especially when indexes are heavily used for example, index scan. Consider using of indexes on database tables that have many rows. WebSphere Message Broker V6.0 uses Dynamic SQL to process each of the database calls within a message flow. It is desirable to reuse statements whenever possible. You can use the database advisor utilities or tools to recommend and review indexes for SQL workloads, for example *Design Adviser* in DB2.

Cluster indexes is another implementation that worth mentioning. For database that support clustered indexes, it can be created to order the rows in the table in the same physical order as the desired result set. Clustered indexes allow linear access pattern to data pages. It promotes effective perfecting data, this helps in avoiding unnecessary sorts. Cluster indexes are created using the *CLUSTER* option of the *CREATE INDEX* statement. Do not create clustered indexes on volatile tables, because the index will not be used. For best performance, create

the index over small data types, for example integer and char(10), unique columns, and columns most often used in range searches.

Buffer pools

Proper definition of buffer pools has major effects to have a well performing system. With 32-bit operating systems, you must be aware of the limit on shared memory, which restricts a database's buffers pools from exceeding the restricted limits. For 64-bit systems, there is no shared memory limit. Refer to the respective operating system manuals for respective platform to get the maximum shared memory limit for respective platforms. If your message flows require high read activity against a database table, ensure that buffer pools are of an adequate size to provide a good hit ratio for data and index reads.

Note: Never allocate more memory than your system can accommodate. If you do so, it incurs costly operating system memory paging which will degrade the overall system performance.

3.3.5 Syslog configurations

On Linux and UNIX systems, all WebSphere Message Broker messages, except those generated by the command line utilities, are sent to the syslog. It is recommended that you redirect user messages to another text file to separate the system errors from the WebSphere Message Broker logs. It is also recommended that you do this activity before you create the broker to ensure that you capture all the broker's creation activities.

For detail information about how to configure the syslog daemon, refer to the topic *Linux and UNIX systems: Configuring the syslog daemon* in the WebSphere Message Broker V6.0 Information Center, which is available at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp?topic=/com.ibm.etools.mft.doc/an04230_.htm

The following example shows how to configure syslog to these recommendations:

1. Log on as root.
2. Enter the following commands to create a file called user.log:

On UNIX systems

```
touch /var/adm/user.log
chown root:mqbrkrs /var/adm/user.log
chmod 640 /var/adm/user.log
```

On Linux

```
touch /var/log/user.log
chown root:mqbrkrs /var/log/user.log
chmod 640 /var/log/user.log
```

3. Add the following line to the /etc/syslog.conf file to redirect debug level messages to the file user.log:

On UNIX systems

```
user.info /var/adm/user.log
```

On Linux

```
user.info /var/log/user.log
```

You can use *user.** instead of *user.info* in these examples. The asterisk (*) means that information, notice, warning, and debug messages are caught.

You can add similar lines to direct information, warning, and error messages to user.log.

4. Restart the syslog daemon using the following commands:

On AIX

```
refresh -s syslogd
```

On HP-UX and Solaris

```
kill -HUP `cat /etc/syslog.pid`
```

On Linux

```
/etc/init.d/syslogd restart
```

or (if rc.d is not a soft link)

```
/etc/rc.d/init.d/syslogd restart
```

For other syslog options, see the documentation for your operating system.



Part 2

Best practices

This part of the book includes the best practices for administering WebSphere Message Broker V6.0. These practices and considerations are catalogued in different management areas, and each area is discussed in a separate chapter, as follows:

- ▶ Chapter 4, “Resource management” on page 43
- ▶ Chapter 5, “Security” on page 83
- ▶ Chapter 6, “Backup, restore, and recover” on page 115
- ▶ Chapter 7, “Monitor and health-check” on page 135
- ▶ Chapter 8, “Maintenance strategy” on page 153
- ▶ Chapter 9, “Problem determination” on page 163

At the end of this part, Chapter 10, “Scenario” on page 217 provides a WebSphere Message Broker runtime scenario. This scenario includes some useful Message broker management and administration hints and illustrates some critical aspects that we discuss throughout the chapters in this book.

Resource management

This chapter discusses the concepts of managing resources that WebSphere Message Broker uses in a production environment. It also describes the common areas that are related to resource management.

We cover the following topics in this chapter:

- ▶ Managing changes in the domain
- ▶ Managing problems in the domain
- ▶ Administering the domain
- ▶ Implementing reliability, availability, and scalability
- ▶ Characteristics for message broker components

4.1 Managing changes in the domain

Any WebSphere Message Broker environment, such as production, should define a systematic approach for any change. Commonly, a change to the environment manifests itself in the form of maintenance, configuration, or deployment. For these types of tasks, the recommended approach is to develop *repeatable processes* to enhance change management capabilities. When you use repeatable processes to manage changes, you reduce the possibility of human error and you promote traceability and accountability for actions that are taken.

A significant portion of change management surrounds the use of source management systems and related topics. Because the focus here is on production administration considerations, a detailed discussion on code control is outside the scope of this book. For more information, you can find a detailed SupportPac™ document, *WebSphere Business Integration Message Broker V5 Change Management and Naming Standards*, at:

<ftp://ftp.software.ibm.com/software/integration/support/supportpacs/individual/ic04.pdf>

A large portion of this SupportPac document discusses source code control as it relates to WebSphere Message Broker.

Primarily our discussion here focuses on change management aspects as they relate to WebSphere Message Broker administrators and the repeatable processes that administrators insist developers and builders use or that are used by the administrators themselves.

4.1.1 Versioning

The enhanced versioning capabilities that are introduced in the development environment with the latest release have been extended to the runtime environment. All resources deployed can now be tagged with version, author, and other useful information. In addition, the standard compiled time and deployment time attributes remain available. The new information is displayed in the administration interface, making it easy to see what version message flows have been deployed to production systems. The recommendation is that WebSphere Message Broker administrators advise developers that relevant information should be deployed with artifacts such as message flows and should be kept up-to-date.

Setting the version and description

While developing a message flow, the version of the message flow should be defined along with any other important attributes, such as owner or contact information. Figure 4-1 illustrates one scenario.

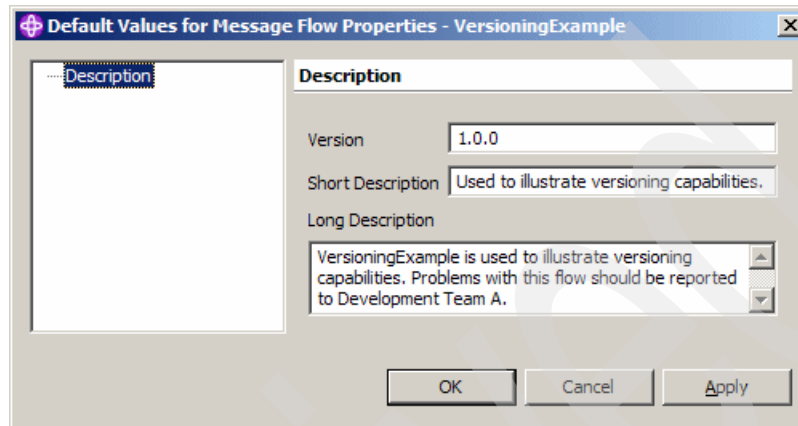


Figure 4-1 Setting description information for a message flow

After the message flow has been deployed, you can view the new attributes. These attributes includes the deployment and modification dates and times (the default information). Additionally, the recently set version attribute is also available. Figure 4-2 shows the *Properties* view of the Message Brokers Toolkit under the Broker Administration perspective with the example message flow details.

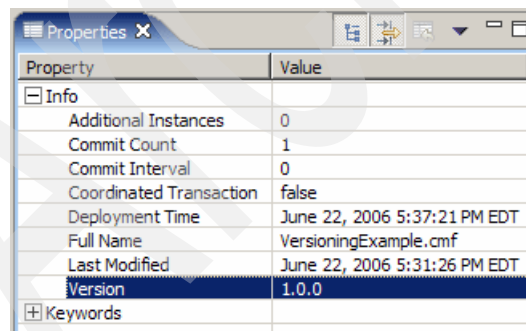


Figure 4-2 Properties of a message flow after deployment

These properties fields are the built-in properties of the message flow. However, you can also define and set *User Defined Properties*.

Setting User Defined Properties

The developer or builder can define and set User Define Properties (UDPs). UDPs provide a customizable hierarchy. This hierarchy allows you to create *Property Groups*. Additionally, fields can have values of types other than String. Currently the types Boolean, Date, Double, Float, Integer, Long, Time, and Timestamp are also available. Figure 4-3 presents a hierarchy using the groups Basic and Advanced with a few properties created.

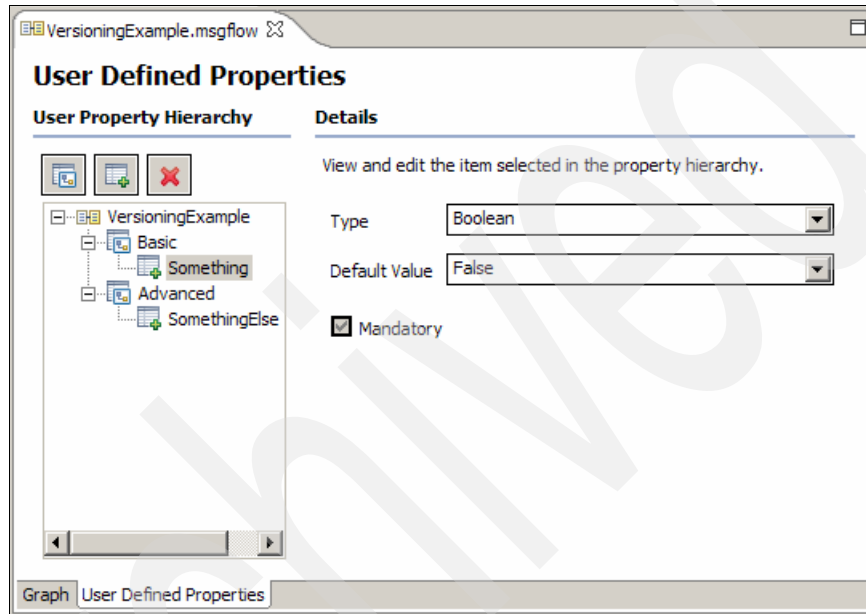


Figure 4-3 User Defined Properties

For detail information about using UDPs, refer to the topic *User Defined Properties* in the WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ac00643_.htm

For detailed information, refer to the topic *Configuring a message flow at deployment time using UDPs* in the WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ac06007_.htm

4.1.2 Procedures and documentation

The major focus of change management is the use of structured documentation to facilitate communication between support teams. As requests for change are created by the WebSphere Message Broker application owners to be done by the administrator, the history of that change must be recorded within a unique log. While this change log can be used as the communication tool between teams, it can also be used to capture the exact steps (the *playbook*) to be performed and for future audit or review.

The WebSphere Message Broker administrator should ensure that several pieces of information are available in the change log before performing the change. This information must include the basic information of why the change is being done, when it should be performed, what steps need to be taken, and who or what can experience an impact.

The deploying of the new version of message flows or new message flow is not about the message flow itself, because the message broker applications are very complex. The message flow depends upon many resources, and the change documentation must contain all relevant information, such as information for the following:

- ▶ Broker
 - Information summary
 - Broker configuration parameters and properties
 - Changes
 - Properties changes
 - Communication protocol changes
- ▶ User Name Server
 - Information summary
 - User Name Server properties
 - Topics and user information
 - Changes
 - Which topics must be deleted, created, or altered
 - Which users must be deleted, added, or altered
- ▶ Execution group
 - Information summary
 - Which message flows the execution group contains
 - Which bar files belong to the execution group

- Changes
 - Which message flow has been canceled and must be deleted from the execution group
 - Which message flows are new and must be deployed
 - Which message flows have a new version and must be redeployed
- ▶ Message flow
 - Information summary
 - WebSphere MQ queues used by particular message flow
 - Databases and database table used by a particular message flow
 - Other sources used by message flow (for example HTTP communication partners, user plug-in specifics)
 - Changes
 - Which WebSphere MQ queue must be deleted, created, or altered
 - Which database table must be deleted, created, or altered
 - Which other sources must be accessed or altered

To make the task of documenting a change simpler as they are needed, maintaining an accurate database of reference information is helpful. For every runtime component or artifact that the administrator supports, there must be a corresponding list of details including:

- ▶ Application owner listings that are used for as contacts when problems occur:
 - Ensure that all message flows have a ownership assigned
 - Ensure that all queues that are used by message flows have ownership assigned
 - Ensure that all databases and database tables that are used by message flows have ownership assigned
- ▶ Service level agreements (SLA):
 - Ensure that all components have at a minimum SLA regarding availability defined
 - Ensure that SLAs include a detailed schedule when maintenance can be performed

4.1.3 Automation

Developing repeatable processes and a good communication system between the WebSphere Message Broker support teams involved are two major focuses of change management. Using automation falls into the category of defining

repeatable processes. You can implement repeatable procedures in the form of one or more procedures that use automation. (Automation, or *tooling*, is also discussed in 4.3.1, “Automation” on page 51.)

Tooling for change management can surround aspects of version and source code control systems, builds, deploys, and applying maintenance to WebSphere Message Broker software and supporting products. The latest release of WebSphere Message Broker expands upon the possibilities of using automation and developing custom tooling to support change management. For example, new and extended command line utilities are provided to allow full automation of deployment of new resources to production environments.

The Configuration Manager Proxy API is a new systems management application interface that allows you to develop full-scale, custom tooling without having to install the Message Brokers Toolkit or the corresponding command line executable.

You can find more information about this interface in *Developing applications that use the Configuration Manager Proxy Java API* in the WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ae33010_.htm

The complete *Configuration Manager Proxy API* reference in the WebSphere Message Broker V6.0 Information Center is available at:

<http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/com/ibm/broker/config/proxy/package-overview.html>

In addition, you can find information about problem determination for the Configuration Manager Proxy API in “Utilizing Configuration Manager Proxy API tracing” on page 199.

A comprehensive example using automation in the form of a deployment tool is presented as a scenario in 10.2, “Resource management” on page 222. In addition, you can use the command line executables such as **mqsdeploy** (UNIX) and **mqsdeploy.bat** (Windows) for scripting a deployment tool. We also discuss these executables in the example scenario.

Note: Windows provides two variants of the **mqsdeploy** command, **mqsdeploy.bat** and **mqsdeploy.exe**. Only **mqsdeploy.bat** can be used in environments that do not have the broker component installed.

4.2 Managing problems in the domain

In a production environment, we recommend that you define a problem management process. Using a formal problem management creates a systematic approach towards handling unexpected events. Creating the problem management process is an activity that can be done throughout the organization and that can involve many groups. The procedures developed within the process include at a high-level problem discovery, root cause analysis, resolution, and return-to-service.

The WebSphere Message Broker administrator translates the high-level procedures into specific tasks that need to be performed. After the individual tasks are completed, the results are reported to a problem manager so that they can be grouped and reviewed by a problem management team before further action is taken.

The broker administrator performs the following steps:

1. *Problem discovery*: Make initial health-checks and review monitored status.
 - Completed prior to performing any formal problem determination for root cause analysis.
 - Obvious causes of the problem or an area of investigation that is likely to give useful results should be studied.
 - Initial findings reported to the component owner.

Follow further the guidelines that are provided in Chapter 7, “Monitor and health-check” on page 135 for health-check and monitoring activities.

2. *Root-cause analysis*: Perform problem determination.
 - Needed if the failure appears to be located within a WebSphere Message Broker component discovered during problem discovery.
 - Report the root-cause to the component owner. If the root-cause remains undetermined, report the steps taken thus far and the plan for further analysis.

For further assistance on problem determination steps, refer to Chapter 9, “Problem determination” on page 163.

3. *Resolution*: Apply the change needed to resolve the problem.
 - Based on the discovery from root-cause analysis and instructions received from the problem manager, initiate the resolution required.
 - Before applying changes, ensure that you have communicate the necessary change to any effected component owners. For further details refer to 4.1, “Managing changes in the domain” on page 44.

- Repeat the problem discovery process to ensure that you have resolved the problem. If you have not resolved the problem, repeat steps 2 and 3.
 - Report that the problem has been resolved to component owners.
4. *Return-to-service*: Notify the user that normal processing has returned.
- After you have completed the previous steps, the administrator and component owners consider a normal production state has been reached (service has been fully restored).
 - Typically, the problem manager or a designated representative notifies the users.
 - The WebSphere Message Broker administrator and component owners must monitor the production environment actively for occurrences of the same report problem or new abnormal events.

4.3 Administering the domain

This section describes best practices for developing reusable processes to manage WebSphere Message Broker runtime components.

4.3.1 Automation

Another way of describing automation in this context is the term *tooling*. (We introduced this concept in 4.1.3, “Automation” on page 48.)

Controlling components

We introduced the concept of automation as it relates to managing a broker domain in 4.2, “Managing problems in the domain” on page 50. Developing automation in the form of operational procedures can take many forms. You can implement tooling to cover procedures that you otherwise need to perform manually.

This section presents a specific type of tooling that you can develop — a shutdown procedure — where a comprehensive tool includes scripts to control all aspects of a shutdown and startup sequence for a broker. The embedded procedures are specific to the environment used.

Common steps for this tool might use the following sequence:

1. Enabling or disabling clustered queue instances on the broker that is started or stopped.
2. Starting or stopping the broker.

3. Starting or stopping the queue manager that is used by the broker.
4. Starting or stopping the database that is used internally by the broker.
5. Setting custom environment variables.
6. Capturing output for historical logging of actions that are taken.

Example 4-1 illustrates a simple script that performs a stop procedure for the AIX-based broker BROKER2.

Example 4-1 A script for a stop procedure: wmbstop.ksh

```
#!/usr/bin/ksh
## Provides a simple example of a shutdown procedure for BROKER2

bn=${0##*/}
bk=BROKER2
qm=QM_ $bk
cq=A.CLUSTERED.QUEUE
log="wmbadmin.log"

## Performs a logging to STDOUT and $log
logIt() {
  timeStamp=`date +"%m%d%C%y-%H%M%S"`
  echo "$bn[$timeStamp]: $1" | tee -a $log
}

## Allows the script to exit early if an error was encountered.
die() {
  if [ $1 -gt 0 ]; then
    logIt "Exiting from error."
    exit $1
  fi
}

logIt "Shutdown procedure initiated"

## Put disable instances of clustered queues owned by $qm.
mqsc="ALTER QL($cq) PUT(DISABLED)"
m=`echo "$mqsc" | runmqsc -e $qm 2>&1`; t=$?
if [ $t -gt 0 ]; then
  logIt "$cq not put disabled! [RC=$t]"
  die $t;
else
  logIt "$cq put disabled. [RC=$t]"
fi
```



```
m=`mqsisstop $bk 2>&1`; t=$?; logIt "$m [RC=$t]"; die $t

m=`endmqm -w $qm 2>&1`; t=$?; logIt "$m [RC=$t]"; die $t

m=`db2stop 2>&1`; t=$?; logIt "$m [RC=$t]"; die $t

logIt "Shutdown complete with no errors reported. [RC=0]"
```

Results are captured in a log file, `wmbadmin.log`, that you can review later if needed, as shown in Example 4-2.

Example 4-2 The `wmbadmin.log` file after `wmbstop` is executed

```
wmb@m106910f:/var/mqsi $ cat wmbadmin.log
wmbstop.ksh[06232006-122603]: Shutdown procedure initiated
wmbstop.ksh[06232006-122603]: A.CLUSTERED.QUEUE put disabled. [RC=0]
wmbstop.ksh[06232006-122612]: BIP8071I: Successful command completion.
[RC=0]
wmbstop.ksh[06232006-122634]: Waiting for queue manager 'QM_BROKER2' to
end.
Waiting for queue manager 'QM_BROKER2' to end.
WebSphere MQ queue manager 'QM_BROKER2' ended. [RC=0]
wmbstop.ksh[06232006-122635]: 06/23/2006 12:26:35    0    0    SQL1064N
DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful. [RC=0]
wmbstop.ksh[06232006-122635]: Shutdown complete with no errors
reported. [RC=0]
```

The administrator can run a shutdown procedure such as the one shown or some other comparable script interactively. In addition, the recommendation is to also have the procedure executed automatically during a server shutdown or called by higher-level automation to apply maintenance automatically.

4.4 Implementing reliability, availability, and scalability

The major areas of runtime support are reliability, availability, and scalability (also known commonly as *RAS*). In this section, we discuss these areas in detail.

4.4.1 Reliability

WebSphere Message Broker is a very reliable runtime platform. However, improper design of message flows can limit this reliability, or *fault-tolerance*.

Administrators should assist developers to ensure that message flows are designed and are built using techniques that allow problems to be self-resolving or to allow for simple problem determination. The more effort that you place on the beginning of a message flow life cycle (design and development time), the less effort you will need to support the application when you release it to production.

The techniques that we describe here reduce the chance for issues in production. Even while using these principles, problem determination can still be necessary. For assistance with problem determination techniques, see Chapter 9, “Problem determination” on page 163.

Bullet proofing message flows

While designing and developing a message flow, review these concepts:

- ▶ Evaluate how *all* potential errors can be handled within the message flow.
- ▶ The design must provide the required functions while also preventing the errors from disrupting normal operations.
- ▶ Always prefer to use the facilities that are provided by the broker to handle any errors that are encountered before creating custom extensions.
- ▶ If basic error processing is not sufficient, developers can take specific action in response to certain error conditions and situations.

While the basic concept — ensuring all errors are handled appropriately — is relatively simple, the application of the concept can become complex in some scenarios. For more information, refer to *WebSphere Message Broker V6, Best Practices Guide: Bullet Proofing Message Flows*, REDP-4043, which is available at:

<http://www.redbooks.ibm.com/abstracts/redp4043.html>

4.4.2 High Availability

This section discusses High Availability (HA) as it relates to WebSphere Message Broker resources.

Concepts

HA is a resource’s ability to withstand a hardware or software failure, dependent on the resource’s reliability. You can achieve HA by using resource duplication to remove single points of failure and adding automatic error detection and

correction features. The availability of the node that is running the WebSphere Message Broker instance is a combination of the availability of several resources:

- ▶ The hardware
- ▶ The operating system
- ▶ The application
 - WebSphere Message Broker
 - WebSphere MQ
 - Database used by the broker
 - External resources
- ▶ The network

To calculate the overall availability of the WebSphere Message Broker environment, the availability of each of these components must be multiplied together. For information about how to calculate the availability, refer to *Planning for Availability in the Enterprise*, which is available at IBM WebSphere Developer Technical Journal Web site:

http://www.ibm.com/developerworks/websphere/techjournal/0312_polozoff/polozoff.html

Hardware

Mainframes deliver HA with characteristics such as error detection and reliable redundant hardware components. While Intel® and Power-based servers provide some improvements in these areas, Intel and Power-based servers still do not have all these types of redundancies built-in.

With these types of systems, HA has to be achieved with installation of additional resources, for example:

- ▶ Redundant network or host adapters
- ▶ Redundant network hardware
- ▶ Redundant power supplies and uninterruptible power supply
- ▶ Hardware RAID (redundant array of inexpensive disks)
- ▶ Redundant servers or server farms
- ▶ Application and operating system HA

WebSphere Message Broker resources

Even with these types of redundancies, you can still experience a server outage. In the event of an outage, the overall services and application should still remain available to the user.

There are several ways to ensure WebSphere Message Broker application based HA, for example:

- ▶ Application and service monitoring and automated responses
- ▶ Application clustering with load balancer
- ▶ Redundant servers or server farms

In a typical setup, you can use a combination of these methods to achieve an acceptable level of HA for the hardware, operating system, and application. The remainder of this section describes specific concerns with each of the major WebSphere Message Broker resources.

HA for databases

For certain types of processing by the broker (including starting and stopping), the database that is used must be available. This need is external to the application databases, which might also be used though the HA principles remains the same. While the actual implementation of an HA database instance is beyond the scope of this book, reference material on this subject should exist for the database product that is used.

Implementing an HA configuration for databases is another task that is also very dependent on the operating systems and hardware that are available. For users that use DB2, the article *DB2 Universal Database and the Highly Available Data Store* is a very concise yet detailed review of scenarios. This article is available at the IBM WebSphere Developer site:

<http://www.ibm.com/developerworks/db2/library/techarticle/0310melnyk/0310melnyk.html>

HA for WebSphere MQ

Before developing an HA solution for WebSphere MQ it is highly recommended that you review the document *Understanding high availability with WebSphere MQ*. This document describes in detail multiple technologies and when you should use them. It discusses topics such as HA clustering, active-active and active-standby, shared disks, and z/OS shared queues. In addition, this document discusses the commonly misunderstood use of WebSphere MQ clustering as an HA solution. The document is available at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0505_hiscock/0505_hiscock.html

You should review the *Appendix A - Available SupportPacs* and the *Resource* chapters in this document. A significant amount of additional resources are available to assist with deciding upon, designing, and implementing an HA solution for WebSphere MQ.

In addition, *Clustering and high availability in an enterprise service bus*, provides architectural guidance on building HA solutions for ESBs that have foundations in WebSphere MQ. This paper is available at:

<http://publibfp.boulder.ibm.com/epubs/pdf/22491360.pdf>

In general, HA clustering with shared disk solutions (active-active or standby) create a method for detecting queue manager failure and then recovering its required files (those on a shared disk) and any other needed resources on a remote server.

Figure 4-4 illustrates HA clustering with shared disk at a high level.

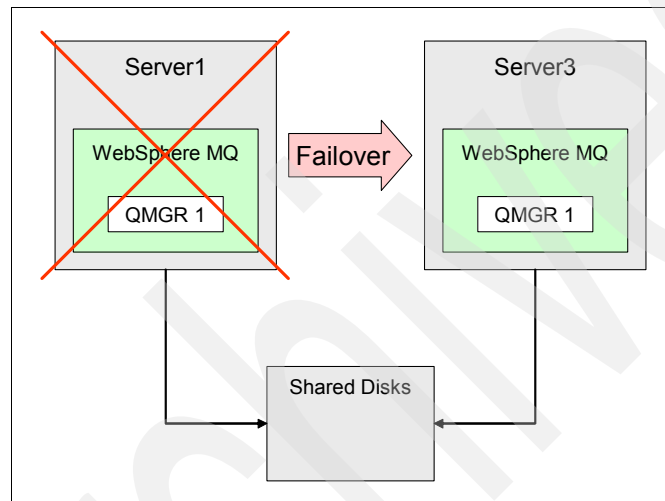


Figure 4-4 Queue manager failover using shared disks

How HA clustering with shared disk is achieved is specific to each environment and is described in detail in the resources that we have mentioned here.

HA for WebSphere Message Broker

For WebSphere Message Broker components to be considered HA, the dependent resources such as WebSphere MQ must also be considered HA. In addition to these prerequisites, you might need to do other work to ensure that broker processing is available continuously.

The methods available for achieving HA brokers is described in the following resources:

- *High Availability z/OS Solutions for WebSphere Business Integration Message Broker V5*, REDP-3894

<http://www.redbooks.ibm.com/abstracts/REDP3894.html>

- ▶ *WebSphere Business Integration Message Broker and high availability environments*
http://www.ibm.com/developerworks/websphere/library/techarticles/0403_humphreys/0403_humphreys.html
- ▶ Each of the following SupportPacs discuss how to implement solutions with specific HA clustering technologies:
 - *IC61: Configuring WebSphere BI Brokers for AIX with HACMP*
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000105&loc=en_US&cs=utf-8&lang=en
 - *IC62: Configuring WebSphere BI Brokers for Sun Solaris with Sun Cluster*
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24005876&loc=en_US&cs=utf-8&lang=en
 - *IC63: Configuring WebSphere BI Brokers for Sun Solaris with Veritas Cluster Server*
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000950&loc=en_US&cs=utf-8&lang=en
 - *IC64: WebSphere BI Brokers for HP-UX - Using with Multi-Computer / ServiceGuard*
http://www.ibm.com/support/docview.wss?rs=171&q1=mAlJ&uid=swg24006311&loc=en_US&cs=utf-8&lang=en
 - *IC71: Configuring WebSphere BI Brokers for Windows with Microsoft Cluster Server*
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24006261&loc=en_US&cs=utf-8&lang=en

4.4.3 Scalability

You can scale WebSphere Message Broker horizontally or vertically by creating runtime topologies that can have multiple servers, brokers, execution groups, or additional instances of message flows. Because there are a large number of possibilities, this section outlines common technologies and illustrates some high-level scenarios.

Vertical

Vertical scalability is the ability to increase the existing capacity of the hardware or software by adding resources or by upgrading the existing resources to newer technologies. Examples of vertical scaling might include adding RAM, CPUs, networking bandwidth, or I/O devices to make the serving component operate faster and with increased capacity.

While increasing hardware based capacity is beyond the scope of this book, the ability to increase processing capacity for a single node running a broker instance is described here. A physical process is created per execution group instance. Within these processes, message flows are executed as threads. For each additional instance of a message flow, an additional thread is created.

The ability to add additional processing power on a single node is possible by simply defining more execution groups or more additional instances of message flows. You should experiment with how many of each type you should create before you enter production, and you should perform this experimentation with similar hardware (physical resources), similar message flow workload, and similar overall workload (non-WebSphere Message Broker applications). All else being equal, we recommend that each thread (message flow instances) have an available processor (threads equal CPUs) for the first experiment. Then, this scenario will be your baseline test.

For further assistance with this task, consult the article *Determining How Many Copies of a Message Flow Should Run Within WebSphere Business Integration Message Broker V5*. This article describes a method for calculation and is available at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0311_dunn/dunn.html

In addition, 4.5.1, “Broker” on page 62 and “Runtime topologies” on page 61 provides more detail on the effects of creating multiple execution groups or message flows.

Horizontal

Horizontal scalability is the ability to connect multiple hardware or software entities so that they work as a single, logical unit. With servers, for example, you can increase the speed or availability of your solution by adding more servers and using clustering and load balancing technologies. The more sophisticated forms of these types of technologies are sometimes referred to as *workload management*.

Horizontal scalability through workload management is one of the ways to increase the amount of processing capacity that is available to WebSphere Message Broker. The next section reviews the use of WebSphere MQ clustering as it relates to broker scalability. In addition, you can find specific examples or runtime topologies in “Runtime topologies” on page 61.

WebSphere MQ clustering

For WebSphere Message Brokers that use WebSphere MQ for submitting work to message flows, you can use WebSphere MQ clustering technology for

workload management. For every broker that runs the same message flow, the input nodes that are used and mapped to instances of local queues could have those queues defined as instances in a cluster. Using this design creates two or more instances of brokers that are essentially clones or copies of each other. These multiple instances of queues can be continually scaled horizontally by placing them on more physical servers.

Attention: This technique of creating what appears to be clones of brokers (and their queue managers) should not be confused with the formal topic termed *cloned brokers*. The scenario that is presented in this chapter deals with making any WebSphere MQ scalable through clustering. The topic of cloned brokers means that for publish/subscribe solutions the subscription table of a cloned broker is replicated to all other brokers with which it is cloned. For further information, refer to *Cloned brokers* in the WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/aq14150_.htm

For architectural guidance with using WebSphere MQ clustering for scalability, see *Providing more processing power through clustered queue managers*, which is available at:

<http://publibfp.boulder.ibm.com/epubs/pdf/p5203200.pdf>

While the details of creating a WebSphere MQ cluster are beyond the scope of this book, there is a large quantity of resources available:

- ▶ *Queue Manager Clusters* in WebSphere MQ Information Center
<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.csqzah.doc/csqzah07.htm>
- ▶ *WebSphere MQ V6 Fundamentals*, SG24-7128
<http://www.redbooks.ibm.com/abstracts/sg247128.html>
- ▶ *Messaging Solutions in a Linux Environment*, SG24-6336, provides very good coverage on the subject that is not operating-system specific
<http://www.redbooks.ibm.com/abstracts/sg246336.html>
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, also has good usage examples
<http://www.redbooks.ibm.com/abstracts/sg246392.html>
- ▶ *WebSphere Application Server V5: Using WebSphere and WebSphere MQ clustering*, TIPS0224
<http://www.redbooks.ibm.com/abstracts/tips0224.html>

- ▶ The IBM developerWorks® article, *Migrating WebSphere MQ queue manager clusters to WebSphere MQ V6.0*

http://www.ibm.com/developerworks/websphere/library/techarticles/0605_vanstone/0605_vanstone.html

Runtime topologies

The inherent ability of WebSphere Message Broker to scale well implies that if message flows are designed to not prevent *N*-way processing, you can run multiple copies on different types of runtime topologies. Some considerations that promote *N*-way processing are designing messages that can be processed on any node or ensuring required downstream resources also promote *N*-way processing.

Determining the exact topology that best fits the runtime is, of course, specific to the type and capacity of the hardware that is available and the work type that is done. In addition, the type of performance that is needed converts to the type of scalability that is required. If the message processing is close to real-time with online users waiting for small pseudo-synchronous (request/reply), non-persistent messages, the runtime topology can be much different from one that uses large messages that are processed in batch mode.

For assistance with designing a runtime topology that best suits the environment and service level requirements, see the following SupportPacs:

- ▶ *WebSphere BI Message Broker: Designing for Performance*, IP04
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24006518&loc=en_US&cs=utf-8&lang=en
- ▶ *WebSphere Message Broker V6.0 for z/OS – Performance Report*, IP14
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010453&loc=en_US&cs=utf-8&lang=en
- ▶ *WebSphere Message Broker V6.0 for HP-UX – Performance Report*, IP6C
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010450&loc=en_US&cs=utf-8&lang=en
- ▶ *WebSphere Message Broker V6.0 for AIX – Performance Report*, IP6D
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010451&loc=en_US&cs=utf-8&lang=en
- ▶ *WebSphere Message Broker V6.0 for Solaris – Performance Report*, IP6E
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010454&loc=en_US&cs=utf-8&lang=en

- ▶ *WebSphere Message Broker V6.0 for Windows – Performance Report*, IP74
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010452&loc=en_US&cs=utf-8&lang=en

4.5 Characteristics for message broker components

This section provides specific resource management information for the following message broker components:

- ▶ Broker
- ▶ Configuration Manager
- ▶ User Name Server
- ▶ Message Brokers Toolkit

4.5.1 Broker

Because the broker role in the IT infrastructure is very important, there are some advanced issues that you must consider to support high demanding environments. These issues are:

- ▶ Handling of large messages
- ▶ Coexistence of versions in the same environment
- ▶ Support for 32-bit and 64-bit architectures

Considerations for the handling of large messages

WebSphere MQ provides the capability to transport large messages. In the context of this section, large messages are those that range from 5 MB to 100 MB. While WebSphere Message Broker is able to parse and write large messages, you should review several considerations before attempting this type of activity in a production environment.

Development

You should first review the design of the message flow parsing or writing of large messages before attempting any configuration changes to the broker environment. Fully discussing ESQL design and development possibilities is not in the scope of this book. However, complete documentation for design development considerations is presented in *Handling large XML messages* in WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ac16740_.htm

You can also find information in *Handling large MRM messages* in WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ac20702_.htm

Runtime

This section provides some possibilities for configuring and tuning a broker and execution group to handle large messages.

There is no single calculation that you can use to gauge the storage requirements accurately for either an execution group or the message flows that use it, because of the significant differences between usage scenarios. Some of the variables include such aspects as populating and retrieving data from user databases, message set dictionaries and internal representation of such parser information, database interactions through message aggregation, or processing related to publish/subscribe.

As such, the recommendation is to execute likely production scenarios early and often throughout the development and testing cycles. After conducting performance tests, if you discover a possibility for storage constraints, applying the recommendations that we discuss here might be useful.

Tip: If predicting storage usage directly through calculations remains necessary, contact IBM Support for further assistance.

WebSphere Message Broker does not have many memory tuning parameters that directly affect how memory management is performed. How large messages are handled within the message flow is where the most noticeable changes are made. From a user perspective, it is not possible to effect the storage that is occupied by the libraries and executable code. However, there are some possibilities for tuning.

The most important task is to ensure that you have configured the operating system to allow the execution group to access as many available resources as possible. This type of configuration is highly dependent on the specific operating system that is used. Before you attempt any other tuning, perform an audit of the environment to ensure that it is compliant with the documentation that is provided for the specific operating system. If the limits are not set to at least the minimum

recommendations, perform the necessary changes. For example, on AIX, the system resource limit for data segment and stack segment should be set to unlimited using the command as shown in Example 4-3.

Example 4-3 Setting resource limits to unlimited on AIX

```
unlimit -d unlimited  
unlimit -s unlimited
```

To locate operating system tuning recommendations, review the *WebSphere Message Broker Installation Guide*, which is available at:

<http://www.ibm.com/software/integration/wbimessagebroker/library>

Detailed guidance is presented primarily for HP-UX and Solaris systems under the section *Checking the kernel configuration*. Additionally, locate the appropriate recommendations for WebSphere MQ and that database that the broker is using (for example DB2).

For operating systems where manual adjustment is crucial for WebSphere MQ, specific documentation on this topic is provided in the following topics, which are available in WebSphere Message Broker V6.0 Information Center:

► AIX

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.amqaac.doc/amqaac0201.htm>

► HP-UX

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.amqcac.doc/amqcac0203.htm>

► Linux

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.amqlac.doc/amqlac0450.htm>

► Solaris

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.amqdac.doc/amqdac0203.htm>

When you ensure that the operating system is configured sufficiently, you can perform a few additional tasks for the WebSphere Message Broker runtime. While you should complete any of the tasks that we have discussed for all brokers, the tasks that follow are specifically for large message tuning.

The first task is to isolate message flows that process large messages to their own execution groups. Additionally, ensure that a minimal amount of supporting artifacts are deployed and resources are created. Artifacts and resources are

normally in the form of message flows, message sets, and databases, respectively. The reason for this approach is that for any message flow to process messages, dependent resources are loaded and then cached for the lifetime of the process or until the message flow is undeployed. These resources are stored in the form of runtime objects that occupy space on the heap (along with administrative objects).

Another tuning option is available, which is isolated to the execution group. Every execution group creates its own Java Virtual Machine (JVM). As such, it is possible to make configuration changes to influence the size of the JVM. In addition, thread stack storage and storage that is occupied by development artifacts, such as message flows and message sets, can also be influenced. For information about JVM problem determination, see “Utilizing JVM tracing” on page 203.

When the JVM is created, the default minimum heap size (for example, 128 MB) is created and reserved for the lifetime of that process. If the message flows in the execution group do not use Java extensively, this default size might be larger than what is actually needed. The possibility for tuning is to reduce the default size to a smaller value. Example 4-4 changes the JVM heap size to 16 MB. The result is that the execution group's main memory heap can now occupy a larger portion of the address space.

Example 4-4 Using mqsichangeproperties to change the JVM minimum heap size

```
mqsichangeproperties BROKER1 -e default -o ComIbmJVMManager -n jvmMinHeapSize -v 16777216
```

Coexistence

WebSphere Message Broker V6.0 can coexist with either WebSphere MQ Integrator V2.1 or WebSphere Business Integration Message Broker V5.0. Coexistence also makes it possible to install multiple versions of the runtime on the same machine. These multiple versions could, for example, have different levels of service applied to them. Message Brokers Toolkit cannot have multiple versions because it can only be installed once on a machine.

Restriction: It is not possible to have both version V2.1 and V5.0 on the same machine. Also, it is not possible to install several instances of either V2.1 or V5.0 on a single machine.

Using a single data source

A version V2.1 or V5.0 broker must use separate sets of database tables. Thus, if a V2.1 or V5.0 broker and a V6.0 broker share a database instance, they must use different database schemas. This technique can be accomplished by using

the same broker service ID and data source name but different data source user IDs. Example 4-5 illustrates this technique.

Example 4-5 Using one data source for two broker versions

```
mqsicreatebroker BK50 -i wmb -a xxxxx -n BKDB -q QM_BK50 -u DBIDA -p yyyy  
mqsicreatebroker BK60 -i wmb -a xxxxx -n BKDB -q QM_BK60 -u DBIDB -p yyyy
```

While this technique is technically possible, the recommendation is that the various levels of brokers have their own database instances if possible.

Additional details

Detailed documentation on coexistence and migration from previous versions to WebSphere Message Broker V6.0 is available in *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247198.html>

Earlier editions of migration redbooks are also available:

- ▶ For version V5.0, *Migration to WebSphere Business Integration Message Broker V5*, SG24-6995

<http://www.redbooks.ibm.com/abstracts/sg246995.html>

- ▶ For version V2.1, *WebSphere MQ Integrator Deployment and Migration*, SG24-6509

<http://www.redbooks.ibm.com/abstracts/sg246509.html>

32-bit and 64-bit support

WebSphere Message Broker V6.0 provides support for 64-bit execution groups from Fix Pack 1 onwards.

Execution groups

The addition of 64-bit execution groups allows support of very large messages and enables WebSphere Message Broker to run applications in fastpath (trusted) mode when using a 64-bit WebSphere MQ V6.0 queue manager.

The execution groups are created and deployed in the workbench that is provided by the Message Brokers Toolkit or by using the command line executable. When creating execution groups, the specification of either running the process as a 32-bit or a 64-bit executable is made. The processor architecture of the target platform determines if a 64-bit executable can be supported. It is also possible for a target broker to contain a mix of 32-bit and 64-bit execution groups.

When using the Message Brokers Toolkit, you can change the default selection for the Execution Group Platform Processor Architecture, normally 32-bit, so that all future execution groups are created with a different default.

Figure 4-5 shows the default for Execution Group Platform Processor Architecture changed to 64-bit.

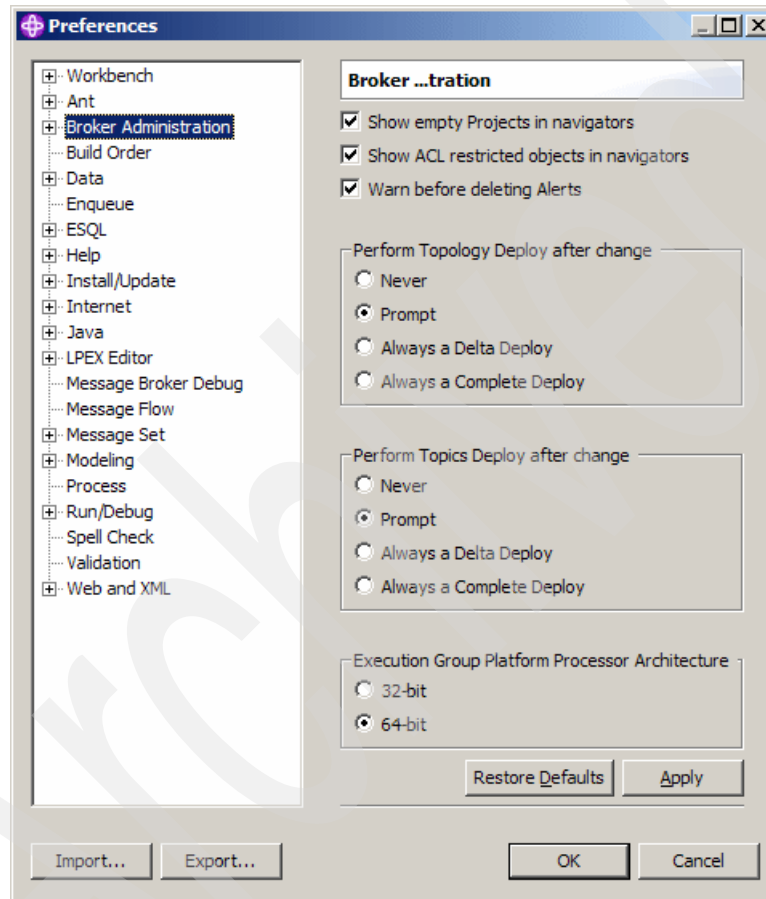


Figure 4-5 Execution Group Platform Processor Architecture default

In addition, you can use the wizard to change the processing architecture when the execution group is created. Figure 4-6 shows a single panel from the Create an Execution Group wizard with a 64-bit processing architecture selected.

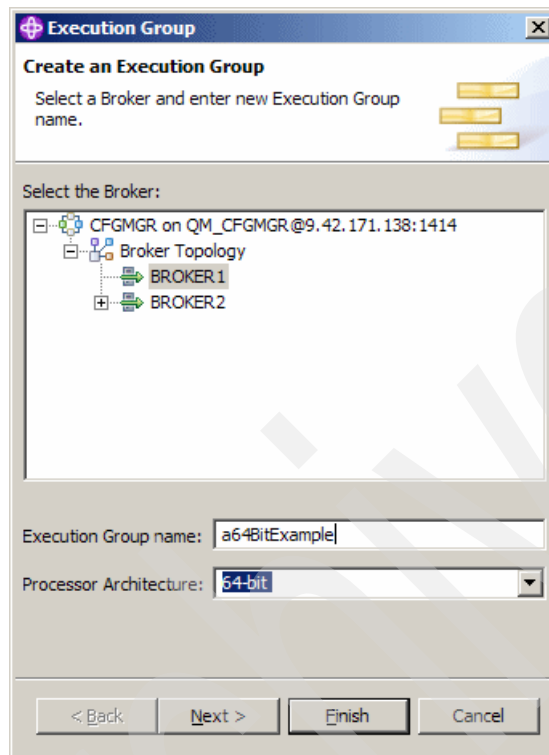


Figure 4-6 Selecting Processor Architecture for execution group

The Properties view shows a reminder of the Processor Architecture in use, as shown in Figure 4-7.

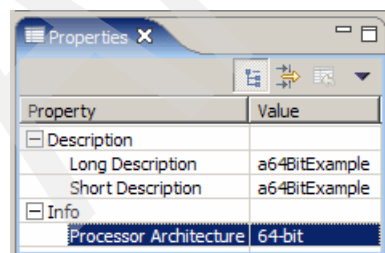


Figure 4-7 Properties view showing Processor Architecture

If you do not use the Message Brokers Toolkit to create the execution group, you can also use the **mqsicreateexecutiongroup** command with the **-l** option.

Databases

In addition to defining the processor architecture for the execution group, the broker database needs both a 32-bit and a 64-bit ODBC data source definition. If you are using 64-bit execution groups and UNIX, ensure that you set the **MQSI_LIBPATH64** environment variable to include the regular 64-bit database libraries.

You can find detailed documentation on these steps in the topic *Configuring a 64-bit ODBC data source on UNIX systems* in WebSphere Message Broker V6.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.eb.doc/ah25530_.htm

When using a 64-bit DB2 instance, ensure that you add the *<DB2 instance directory>/sqllib/lib32* library to the start of the library search path environment variable. Adding this library might prevent the ability to run DB2 commands from the same environment command line. If you want to execute DB2 commands, create a separate environment shell and run the **db2profile** command for the relevant instance without setting **mqsiprofile** first.

When using a 64-bit Oracle instance, ensure that you add the *\$ORACLE_HOME/lib32* library to the start of the library search path environment variable.

4.5.2 Configuration Manager

When you deploy large bar files, you might experience problems do to a lack of real memory (storage). The most likely cause of this problem is that the Configuration Manager's JVM that is used to perform the deployment logic has exceeded the temporary storage space.

To rectify this problem, change the JVM heap size parameter to increase the amount of memory that the Configuration Manager's JVM is allowed to access, as shown in Example 4-6.

Example 4-6 Changing the JVM heap size of the Configuration Manager

```
C:\Program Files\IBM\MQSI\6.0>mqsistop CFGMGR  
BIP8071I: Successful command completion.
```

```
C:\Program Files\IBM\MQSI\6.0>mqsichangeconfigmgr CFGMGR -j 320  
BIP8071I: Successful command completion.
```

```
C:\Program Files\IBM\MQSI\6.0>mqsiservice CFGMGR
BIPV600 en US
    ucnv Console CCSID 437    dft ucnv CCSID 5348
    ICUW ibm-5348_P100-1997    ICUA ibm-5348_P100-1997

Install Path = C:\Program Files\IBM\MQSI\6.0
Working Path = C:\Documents and Settings\All Users\Application
Data\IBM\MQSI
process id = 0
ConfigMgr db userId = wmb
ConfigMgr db password =
jdbc driver name = COM.ibm.db2.jdbc.app.DB2Driver
jdbc connection name = db2
queue manager = QM_CFGMGR
jvm heap size = 335544320
ComponentType = ConfigMgr

BIP8071I: Successful command completion.
```

```
C:\Program Files\IBM\MQSI\6.0>mqsistart CFGMGR
WebSphere MQ queue manager running.
BIP8096I: Successful command initiation, check the system log to ensure
that the
    component started without problem and that it continues to run without
problem.
```

In this example, the Configuration Manager is stopped, and the JVM heap size is changed from the default of 256 MB to a new size of 320 MB (adding 64 MB) with the **mqsichangeconfigmgr** command. To confirm the change, the **mqsiservice** command is used to illustrate that the number of bytes has in fact been changed to 335544320. (You can confirm this change with the calculation $320 \times 1024 \times 1024$.)

4.5.3 User Name Server

If the applications are using the publish/subscribe services of a broker, you can apply an additional level of security to the topics where messages are published and subscribed. This additional level of security is provided by the User Name Server component through access control lists (ACL).

Prior to enabling this feature, ensure that the User Name Server's queue manager has intercommunication established with the broker's queue manager by creating sender/receiver channel pairs. Then, to enable this feature, the

broker must be made aware of the User Name Server by using either of the following methods:

- ▶ When creating the broker specify the option `UserNameServerQueueManagerName` on the **mqsicreatebroker** command.
- ▶ Change an existing broker using the **mqsichangebroker** command.

When the User Name Server starts, an indication is made in the syslog that it has registered with the broker.

Tip: To start the User Name Server on z/OS, a sample BIPUNSP file is provided. Refer to the topic *Customizing the User Name Server component data set* in the WebSphere Message Broker V6.0 Information Center:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp?topic=/com.ibm.etools.mft.doc/ae22270_.htm

Messages similar to that shown in Example 4-7 are sent to the syslog (UNIX) or Application log at Event Viewer (Windows).

Example 4-7 Successful User Name Server registration messages

```
BIP9141W: The component was started.
BIP2001I: The WebSphere Message Broker service has started, process ID 196827.
BIP8201I: User Name Server starting with refresh interval 60.
BIP8204I: User Name Server is registering a client with UUID
12345678-1234-1234-1234-123456789abc, and cache version 0.
```

Prior to adding the first ACL entry, ensure that the User Name Server's queue manager has intercommunication established with the Configuration Manager's queue manager by creating sender/receiver channel pairs. Additionally, the Configuration Manager must be made aware of the User Name Server's queue manager name. Use the **mqsicreateconfigmgr** or **mqsichangeconfigmgr** commands to set this value.

Example 4-8 shows that the relationship has been established.

Example 4-8 Associating the User Name Server with the Configuration Manager

```
C:\>mqsichangeconfigmgr CFGMGR -s QM_UN
BIP8071I: Successful command completion.

C:\>mqsiservice CFGMGR
BIPv600 en US
      ucnv Console CCSID 437      dft ucnv CCSID 5348
      ICUW ibm-5348_P100-1997     ICUA ibm-5348_P100-1997
```

```
Install Path = C:\Program Files\IBM\MQSI\6.0
Working Path = C:\Documents and Settings\All Users\Application
Data\IBM\MQSI
process id = 3508
ConfigMgr db userId = wmb
ConfigMgr db password =
jdbc driver name = COM.ibm.db2.jdbc.app.DB2Driver
jdbc connection name = db2
queue manager = QM_CFGMGR
UNS queue manager = QM_UN$
jvm heap size = 402653184
ComponentType = ConfigMgr
```

BIP8071I: Successful command completion.

The following sequence of steps illustrates how to register the user names and groups with the Configuration Manager:

1. Add the users and groups to the password file `pwgroup.dat`, as shown in Example 4-9.

Example 4-9 pwgroup.dat

```
# Each line contains two or more required tokens delimited by
# commas. The first is a user id and the second that user's
# password. All subsequent tokens specify the set of groups that
# the user belongs to.
```

# USERNAME	PASSWORD	GROUPS
# =====	=====	=====
wmb,	sam605cc,	mqrbrks
testuser,	sam605cc,	mqrbrks,mqm
dummy,	sam605cc,	mqm

For further information about configuring the User Name Server, see 5.2.3, “User Name Server” on page 111.

2. Use the `mqsi changeusernameserver` command to register the groups and password file (Example 4-10).

Example 4-10 Binding the authProtocolDataSource to the User Name Server

```
C:\>mqsisstop CFGMGR
BIP8071I: Successful command completion.
```

```
C:\>mqsisstop usernameserver
```

BIP8071I: Successful command completion.

```
C:\>mqsichangeusernameserver -g "c:\pwgroup.dat"
```

BIP8071I: Successful command completion.

```
C:\>mqsistart usernameserver
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

```
C:\>mqsistart CFGMGR
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

3. Restart the Configuration Manager and User Name Server for the change to take effect.

After binding the name of the data source file (pwgroup.dat), which is required by the authentication protocol to the User Name Server, you can verify the change by using the Topic/Users tab on the Topics panel from the Message Brokers Toolkit. This tab shows the Topic Access Control List information that is sent from the User Name Server. You can also use this tab to ensure that the User Name Server has registered with the Configuration Manager.

Figure 4-8 shows the newly registered topic access controls.

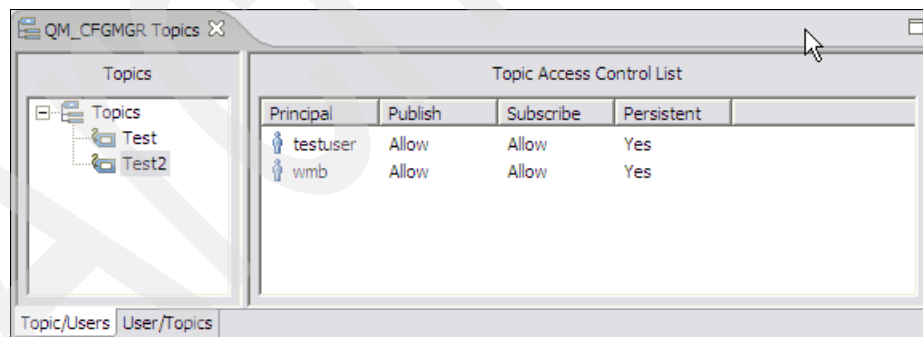


Figure 4-8 Topic Access Control List

You can manage the user and group policies for the topic by right-clicking **Topics** and selecting **Manage Policies** from the menu.

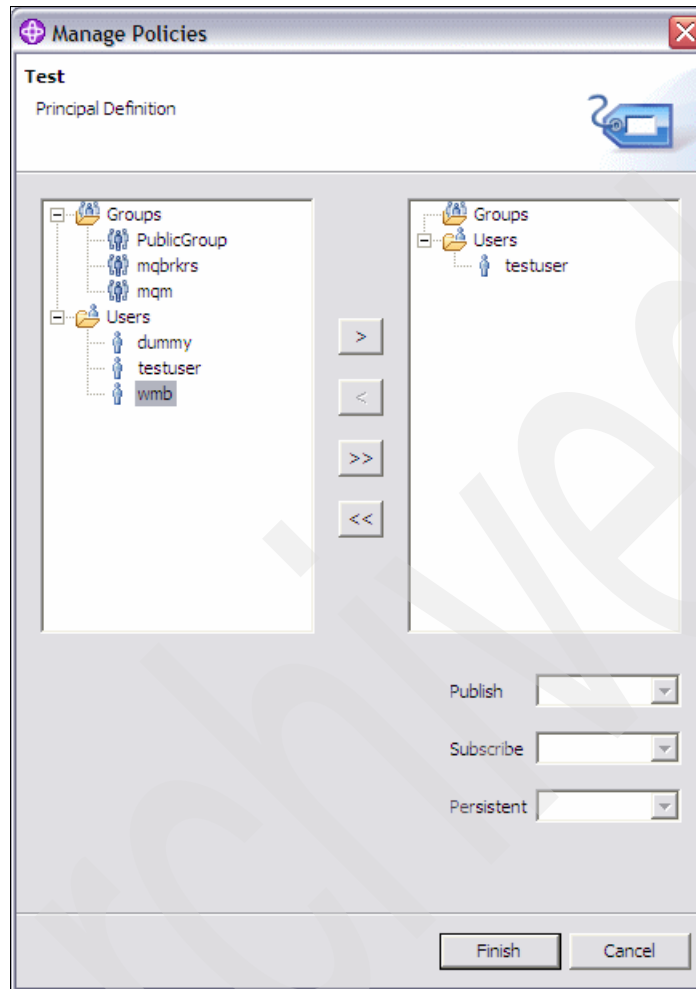


Figure 4-9 Managing policies for the topics

Additionally, the Users/Topics tab shows the User Access Control List. Figure 4-10 shows the newly registered user access controls.

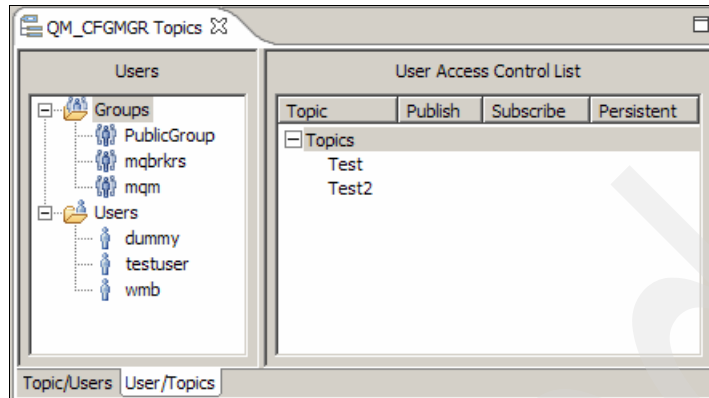


Figure 4-10 User Access Control List

For assistance with problem determination with the User Name Server component, see 9.3.3, “Problem determination with the User Name Server” on page 212.

4.5.4 Toolkit

This section discusses the specific information that is related to the management of resources from the Message Brokers Toolkit perspective.

Command Assistant wizard

The Message Brokers Toolkit on Windows provides a feature called the Command Assistant wizard as another method for administering the WebSphere Message Broker V6.0 runtime. This wizard is an alternative to the standard command line utilities.

Note: The various administration tasks are limited to the local machine on which the Message Brokers Toolkit is installed.

You can use the Command Assistant wizard to create, modify, and delete the following physical runtime components:

- ▶ Brokers
- ▶ Configuration Managers
- ▶ User Name Servers

For more information about the Command Assistant wizard, refer to the topic *Using the Command Assistant wizard* in WebSphere Message Broker V6.0 Information Center:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ae35700_.htm

Changing JVM's footprint

The Message Brokers Toolkit is launched from the executable **wmb** (location on Windows in the C:\Program Files\IBM\MessageBrokersToolkit\6.0 directory). This executable sets properties and launches workbench in a new JVM. You can define the properties that it sets for the JVM can along with the **wmb** command or by specifying properties in the **wmbt.ini** file. In certain situations, altering the JVM startup properties might be a helpful, especially when the footprint of the workbench is needed to be considerably larger or smaller than the default.

JRE heap values

For the IBM Java Runtime Environment (JRE™) that the Message Brokers Toolkit uses, the following values are used to alter the JVM heap size:

► **-Xms<size>**

Sets the initial size of the heap. If this option is not specified, it defaults to 4 MB on Linux and Windows.

► **-Xmx<size>**

Sets the maximum size of the heap. On Windows this value defaults to half of the real storage, but not less than 16 MB or more than 2 GB. On Linux, it defaults to half the real storage but not less than 16 MB or more than 512 MB.

Maximizing the JVM footprint

In situations where the Message Brokers Toolkit is heavily utilized, such as a machine that is used for development where the significant portion of user activity is within the work space, increasing both the minimum and maximum values might be a helpful. An ideal scenario for this option would be one where the developer is using multiple features and plug-ins such as the Message Brokers Toolkit itself, WebSphere MQ Explorer, and other tools in the IBM Rational family. Normally, this type of scenario would be a configuration that has many Capabilities enabled, as shown in Figure 4-11 on page 77.

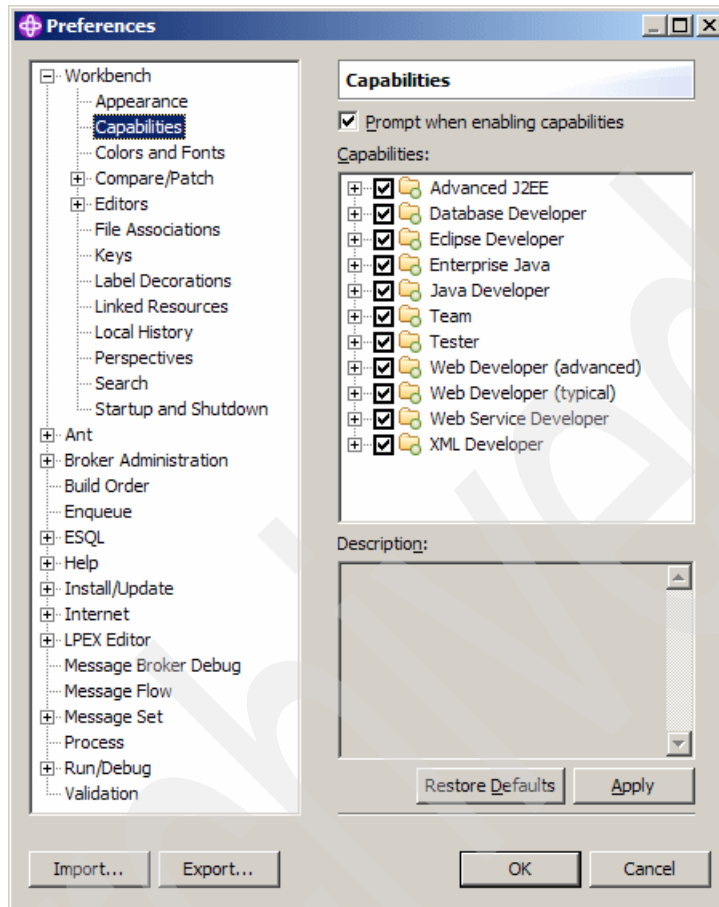


Figure 4-11 Message Brokers Toolkit with many Capabilities enabled

The assumption is that the machine being used has real memory available that exceeds half of real storage that can be used for development. In Example 4-11, the two parameters highlighted were made to the wmbt.ini file such that the minimum heap size is set to 128 MB and the maximum size is 640 MB.

Example 4-11 wmbt.ini with additional JVM arguments

```
; Licensed Materials - Property of IBM
; Component: Rational Software Development Platform Launcher
; Copyright IBM Corporation 2004 All Rights Reserved.
; US Government Users Restricted Rights Use, duplication or disclosure restricted by
; GSA ADP Schedule Contract with IBM Corp.
;-----

; This file controls the behavior of the product launcher

[Settings]
ProductName=WebSphere Message Broker Toolkit
Version=6.0
Full=Yes
KeyName=wmbt60
; Other settings
CustomizationINI=C:\Program
Files\IBM\MessageBrokersToolkit\6.0\eclipse\plugins\com.ibm.etools.msgbroker.tooling_
6.0.0\plugin_customization.ini

VMArgs=-Xj9
VMArgs=-Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse.emf.ecore.impl.
EPackageRegistryImpl
VMArgs=-Dosgi.splashPath=platform:/base/plugins/com.ibm.etools.msgbroker.tooling,plat
form:/base/plugins/com.ibm.etools.msgbroker.tooling.n11,platform:/base/plugins/org.ec
lipse.platform
VMArgs=-Declipse.product=com.ibm.etools.msgbroker.tooling.ide
VMArgs=-Dosgi.instance.area.default=@user.home/IBM/wmbt6.0/workspace
VMArgs=-Xms128M
VMArgs=-Xmx640M
```

Minimizing the JVM footprint

Conversely to increasing the size of the Message Brokers Toolkit, it can sometimes be beneficial to decrease the size to a footprint as small as possible. A situation where this might be necessary is a machine running many applications, such as an administrative desktop with several large monitoring tools. The Message Brokers Toolkit in such an environment would only be used for lightweight administrative tasks or to poll a Configuration Manager manually for domain status.

There are two techniques described here. First, reduce the number of Capabilities for the toolkit such that only the tools really needed are available. Figure 4-12 shows all non-Message Brokers Toolkit Capabilities disabled:

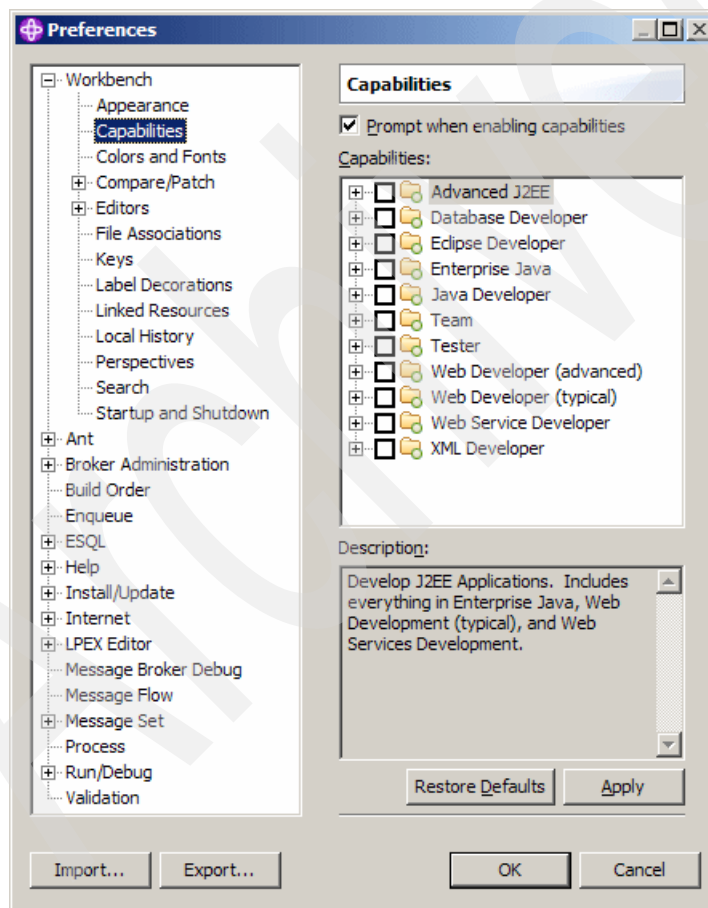


Figure 4-12 Message Brokers Toolkit with no Capabilities enabled

Additionally, specify much smaller values for the JVM's maximum heap size. Figure 4-13 shows that the Windows shortcut used to launch the Message Brokers Toolkit now has three additional parameters specified. The first disables the splash screen because it is not necessary to see it every time and seeing it decreases startup time. The other two parameters ensure that the maximum JVM size does not grow beyond 24 MB. Light testing shows that this value is probably the lowest value that can be used. The value might need to be increased for your environment.

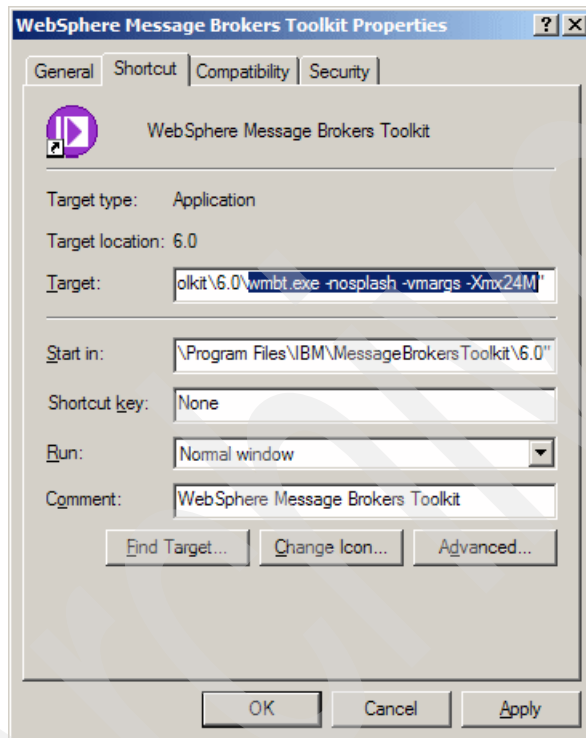


Figure 4-13 Altering the Message Brokers Toolkit shortcut

Managing different broker domains

Figure 4-14 illustrates the capability of the Message Brokers Toolkit to manage more than one message broker domain. The Message Brokers Toolkit can manage different broker domains concurrently. There are two broker domains in this example:

- ▶ Domain *D1* with
 - Configuration Manager *D1_CM* on AIX
 - Brokers *D1_BKA1* and *D1_BKA2* on AIX
 - Broker with Rules and Formatter Extensions *D1_BKW1* on Windows
- ▶ Domain *D2* with
 - Configuration Manager *D2_CM* on z/OS
 - Broker *D2_BKZ1* on z/OS

Both broker domains can be managed by both Message Brokers Toolkits, either from Windows or Linux workstations.

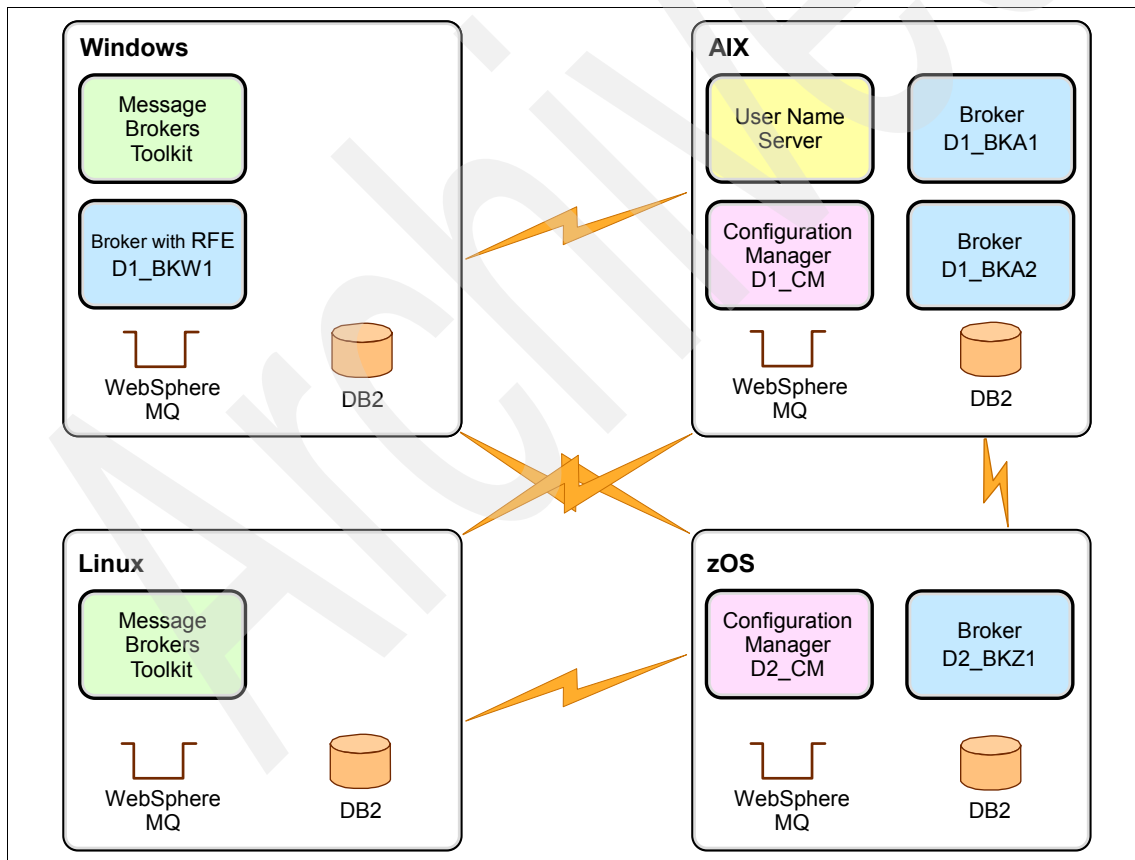


Figure 4-14 Managing different broker domains

Security

This chapter discusses the security enablement features of the WebSphere Message Broker domain. It also provides relevant information about how the message broker supports the authentication, authorization, and encryption processes. It includes the following topics:

- ▶ Securing the domain
- ▶ Characteristics for message broker component

5.1 Securing the domain

The most common approach to securing the WebSphere Message Broker domain is to divide the responsibility of the security in several areas of concern. In our case, the main areas of concern are:

- ▶ Authentication
- ▶ Authorization
- ▶ Securing the transport layer

In this section, we discuss how the message broker supports each of these areas of concern in security.

5.1.1 Authentication

Authentication is the process of verifying the identity of an user of a program inside the message broker platform. In WebSphere Message Broker, the identity of the user of the platform is defined by login and password, and in cases such as the Real-time node and the HTTP listener, you can implement a stronger authentication mechanism. Another mechanism that the message broker provides are the security exits.

Before we describe each of these features, there is an important consideration regarding the name of the principals inside the WebSphere Message Broker middleware platform. Because WebSphere Message Broker is designed to be a platform-independent product and the WebSphere Message Broker environment can be heterogeneous, the name of all the principals must be up to eight characters long. This name length requirement brings adaptability of the product to all possible different scenarios (in terms of operating systems and users repositories). If there is a Windows only environment, the principals' names are allowed up 12 characters long.

Authentication for Real-time node

Authentication services are supported between client applications that use the WebSphere MQ Real-time Transport and WebSphere Message Broker Real-timeInput and Real-timeOptimizedFlow nodes.

The WebSphere Message Broker authentication services verify that a broker and a client application are who they claim they are and can therefore participate in a publish/subscribe session. Each participant in the session uses an authentication protocol to prove to the other that they are who they say they are and are not an intruder impersonating a valid participant.

The WebSphere Message Broker product supports the following four protocols:

- P:** A simple telnet-like password authentication
- M:** Mutual challenge-response password authentication
- S:** Asymmetric Secure Sockets Layer (SSL)
- R:** Symmetric SSL

The protocols vary in strength, in terms of providing protection against participants that are not valid participants in the session. *P* is the weakest, and *R* is the strongest.

Configuring Authentication protocol

You can configure the set of protocols that a specific broker can support in the broker domain using the workbench. Each broker can specify one or more protocols that it can support. In addition to the broker configuration, you can use the workbench to enable or to disable authentication on each Real-timeInput node that is defined for a particular broker. When authentication is enabled at a Real-timeInput node, that node supports the full set of protocols that is specified for its corresponding broker. Figure 5-1 illustrates the configuration options that represents the overview of authentication configuration.

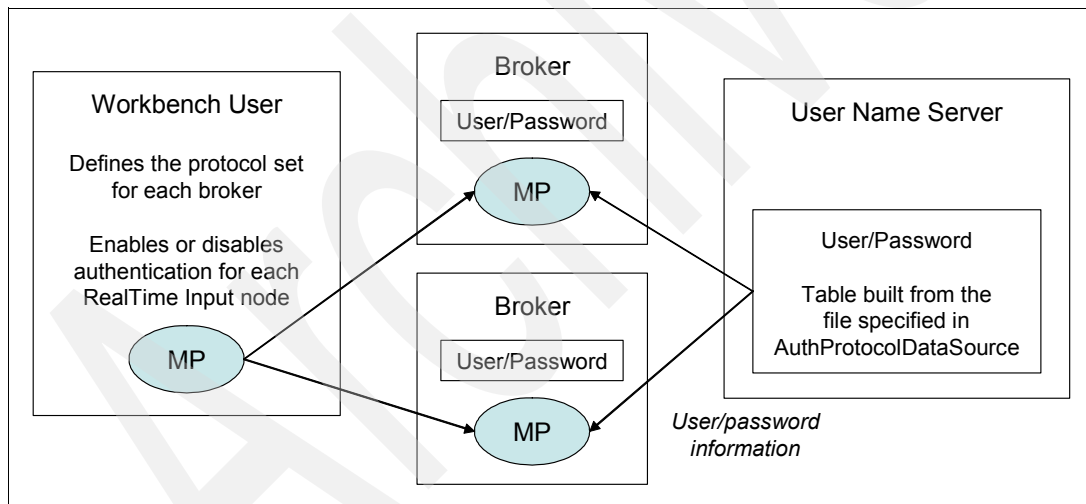


Figure 5-1 Authentication configuration

Figure 5-2 shows how the protocol handles in the Real-time node.

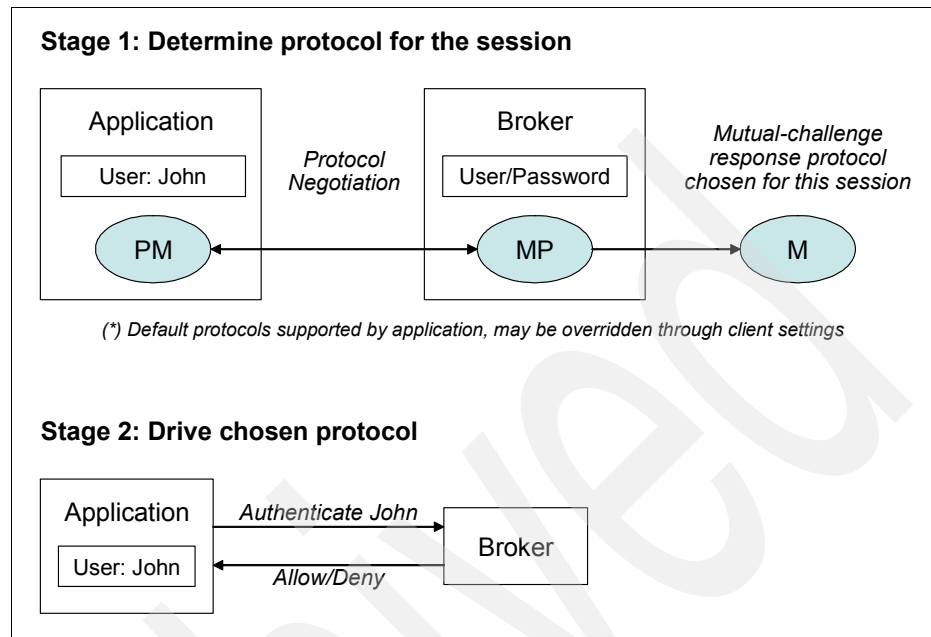


Figure 5-2 Protocol handling

Simple telnet-like password authentication

This protocol can also be described as *password in-the-clear* because the password passes unencrypted over the network. The client application connects to the Real-timeInput node using TCP/IP. The input node requests that the client identify itself, and the client sends its user ID and password.

This simple protocol relies on both the client and the broker knowing the password that is associated with a user ID. In particular, the broker needs access to a repository of user and password information. The user ID and password information is distributed by the User Name Server to all the brokers in a WebSphere Message Broker domain. The User Name Server extracts user and password information from an operating system file.

The User Name Server approach allows for the centralized maintenance of the source of users and passwords, with automatic distribution of the information to brokers, and automatic refreshes of the information if required. It also provides availability benefits, because user and password information is maintained persistently at each broker.

Each client application must know its own user ID and keep its password secret. When creating a connection, a client specifies its credentials as a

name/password combination. In the case where user and password information is stored in a flat file on the User Name Server system, the passwords are stored and distributed in-the-clear.

The disadvantage of this protocol is that it provides relatively weak security. It does not compute a session key. Thus, you should use it only in environments where there are no *eavesdroppers* and no untrusted *middle-men*.

Mutual challenge-response password authentication

This type of authentication is a more sophisticated and secure protocol that involves the generation of a secret session key. Both the client and the server compute this key using the client's password. They prove to each other that they know this secret through a challenge and response protocol.

The client must satisfy the server's challenge before the server satisfies the client's challenge. Thus, an attacker who is impersonating a client can gather no information to mount an *offline* password guessing attack. Both the client and the server prove to each other that they know the password, so this protocol is not vulnerable to *impersonation* attacks.

As in the case of the simple telnet-like password protocol, the broker must have access to user and password information. Information about the user ID and password is distributed by the User Name Server to all the brokers in the domain. The User Name Server extracts user and password information from an operating system file.

Each client application must know its own user ID and keep its password secret. When creating a connection, a client specifies its credentials as a name/password combination.

SSL symmetric and asymmetric authentication

SSL is an industry-standard technology for securing the flow of confidential data across public networks and providing assured identity context. SSL builds on a number of fundamental concepts, including asymmetric and symmetric cryptography, message digests, digital signatures, digital certificates, CipherSuites, and certificate authorities. These concepts are common to all SSL implementations.

Note: For a full description of each of these concepts, refer to *WebSphere MQ Security*, SC34-6588, which is available at:

<http://www.ibm.com/software/integration/wmq/library/library6x.html>

The support of SSL in the authentication of the Real-time node is described in “SSL on Real-time nodes” on page 96.

SSL authentication for the HTTP listener

In addition to Real-time node, another authentication mechanism that WebSphere Message Broker supports is the SSL authentication of the HTTP listener. This mechanism provides confidentiality to the HTTP communications of the message broker. (An example of HTTPS listener configuration is explained in 10.3, “Security” on page 234.)

Security exits

Security exits are special programs that you can use to verify that the partner at the other end of a WebSphere MQ channel is genuine. Security exits are a facility that WebSphere MQ provides. You can use them to authenticate access to the Configuration Manager for different client programs that are trying to access it. When the Message Brokers Toolkit is started, it does not invoke a security exit to monitor its connection to the Configuration Manager by default. From a developer perspective, security exits are standard WebSphere MQ security exits, written in Java, and implemented in the WebSphere MQ exit interface.

To use a security exit in a communication between a Message Brokers Toolkit and a Configuration Manager, you must provide the security exit Java class and the JAR file that contains it when creating or modifying a domain within the Domains view of the Message Brokers Toolkit. Figure 5-3 on page 89 shows the Domains view of the Message Brokers Toolkit.

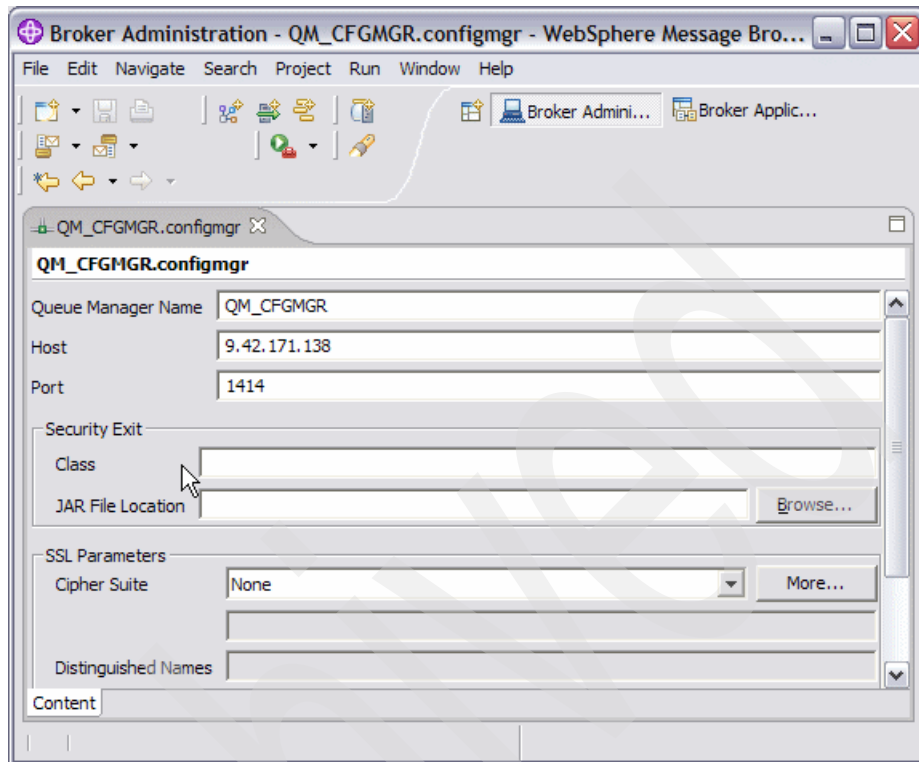


Figure 5-3 Domains view of the Message Brokers Toolkit

The information about how to set up the security exit at the server end of the connection is available in the standard WebSphere MQ documentation.

Note: For more information about Security Exits in WebSphere MQ, refer to *WebSphere MQ Security V6.0*, SC34-6588, or to the IBM WebSphere MQ Information Center at:

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp>

5.1.2 Authorization

Authorization is the process of granting or denying access to system resources. For WebSphere Message Broker, authorization controls who has permission to access WebSphere Message Broker resources and ensures that users who attempt to work with those resources have the necessary authorization to do so.

Examples of tasks that require authorization are:

- ▶ Configuring a broker using, for example, the **mqsicreatebroker** command.
- ▶ Accessing queues, for example, putting a message to the message flow input queue.
- ▶ Taking actions within the workbench, for example, deploying a message flow to an execution group.
- ▶ Publishing topics and subscribing to topics.

We can categorize the WebSphere Message Broker users in the following groups:

- ▶ **Workbench users**

Designated to develop and test the message flows, mostly using the Message Brokers Toolkit to produce artifacts.

- ▶ **Administrative users**

Allowed executing commands of one or all of the components of the WebSphere Message Broker domain (for example, the message brokers or the Configuration Manager).

- ▶ **Service users**

Operating systems accounts are used for controlling and managing the broker domain components.

- ▶ **Database users**

Operating systems accounts or particular database accounts (depends on the database configuration) are used to access the brokers and application databases.

Groups considerations

For services users, the most important considerations are:

- ▶ A local group called **mqbrkrs** must be created on each machine of the WebSphere Message Broker middleware platform, and the broker service user IDs must belong to that group.
- ▶ On Windows, there are additional requirements for a runtime component's service user ID:
 - Because the broker runs as a Windows Service, the service user ID must be allowed to log on as a service. If this access is not granted, you are able to create the component, but the **mqsistart** command fails.
 - No users are allowed to log on automatically as a service (including those in the Administrators group). The permission needs to be granted

separately and is a requirement for all runtime components' service user IDs (Configuration Manager, broker and User Name Server).

To grant permission:

- i. Select **Control Panel** → **Administrative Tools** → **Local Security Policy**.
 - ii. Open **Local Policies** → **User Rights Assignment** and change the setting *Log on as a service*.
 - iii. Add the service user IDs to the list. You only need to do this once for each service ID, but you must do it *locally on each machine* that runs brokers. So, if a domain ID is being used as the service user ID, you must make a change to the security policy *on each machine* in the domain that runs brokers (not only on the domain controller).
- The service user IDs for the Configuration Manager and the broker can be different and in a production environment can even belong to different systems. In order to avoid errors in deployment of message flows and message sets from the Configuration Manager to the broker, we suggest to follow these considerations:
- Ensure that the broker's user ID is a member of the mqm and mqbrkr groups.
 - Define the Configuration Manager's user ID on the system on which the broker is running.
 - Ensure that all IDs are in lowercase letters so that they are compatible between computers.

For database users:

- The user ID intended to access the broker database must follow the limit length consideration (up to eight characters, unless Windows and then up to 12). When you use a DB2 database as a broker database, then use a local user ID to access the database. If the brokers are created in a domain environment where a domain user ID is used for the service user ID, set the database ID to a local ID that is authorized to access databases.
- Authorize the database selected user IDs to access the broker.
- When broker is created, identify the database selected user ID using the **-u** and **-p** options on the **mqsicreatebroker** command.

Runtime objects ACLs

Runtime resources are WebSphere Message Broker objects that exist at runtime environment. Runtime security controls the permission that is necessary for actions taken on these resources from the workbench. For example, the user ID

that is used to deploy a message flow to an execution group must have permission to take that deploy action.

In WebSphere Message Broker, each runtime object has an access control list (ACL). The ACL for an object determines the view and modify permissions that a principal has for that object (see Example 5-4 on page 94). ACLs provide more granularity and give a greater degree of control over the permissions that principals have. ACLs also enable a single Configuration Manager in a single security domain to control brokers with incompatible access control, such as development, system test, and production.

WebSphere Message Broker allows control access by object as opposed to by group. For example, user JUNGLEMPERRY might be given access to modify BROKERA but have no access rights to BROKERB. In a further example the same user might have access to deploy to execution group EXEGRP1 but not to EXEGRP2, even though they are both members of BROKERA.

Access level control is based on the operation requested on any type of WebSphere Message Broker object (for example, brokers and execution groups). Access control enables permission assignment to principals (users and groups), and these permissions specify view and modify rights for deployment configuration resources. In WebSphere Message Broker, administrators can configure ACLs to use their organization's principals.

This model provides full object level access based on the action that needs to be performed, provided that the necessary group and access entry has been created and that the system administrator can nest local and domain groups to help organize security requirements and users.

Security environments can still be operated with any existing group level from earlier releases of WebSphere Message Broker without any migration issues. If there is any existing user role definition groups with members, you can use this model without making any additional changes. The object level security model is only invoked if the user role definition groups are either empty or do not exist. If you implement object level security, the user role definition groups must be empty.

Attention: When you use ACLs in WebSphere Business Integration Message Broker V5.0, you must follow the correct migration steps to WebSphere Message Broker V6.0, because ACL implementation has changed slightly from V5.0 to V6.0. Refer to *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247198.html?Open>

In the workbench, when attempting an operation for which permission is required, the following information is passed to the Configuration Manager:

- ▶ Object type
- ▶ Object name
- ▶ Requested action
- ▶ Your user ID

The Configuration Manager checks the ACL table. If your user ID is included in the ACL for the named object, you have permission to perform the operation.

There are four different access levels that can be set for a user or group ID: full, view, deploy, and edit. The access level is set through the `-x` parameter of the `mqsicreateaclentry` command. You can also manipulate ACL entries using the Java Configuration Manager Proxy API. The `-x` parameter determines which user or group ID can create, delete, or modify the WebSphere Message Broker objects.

Figure 5-4 illustrates how the control traverses to determine the rights that user *John* has over the WebSphere Message Broker resources. We assume that the command entered to set the user's permissions is as follows:

```
mqsicreateaclentry CFGMGR -u JOHN -x F -b BROKER1 -e EXEGRP1
```

Note: For more information about access control lists, refer to the WebSphere Message Broker V6.0 Information Center:

<http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp>

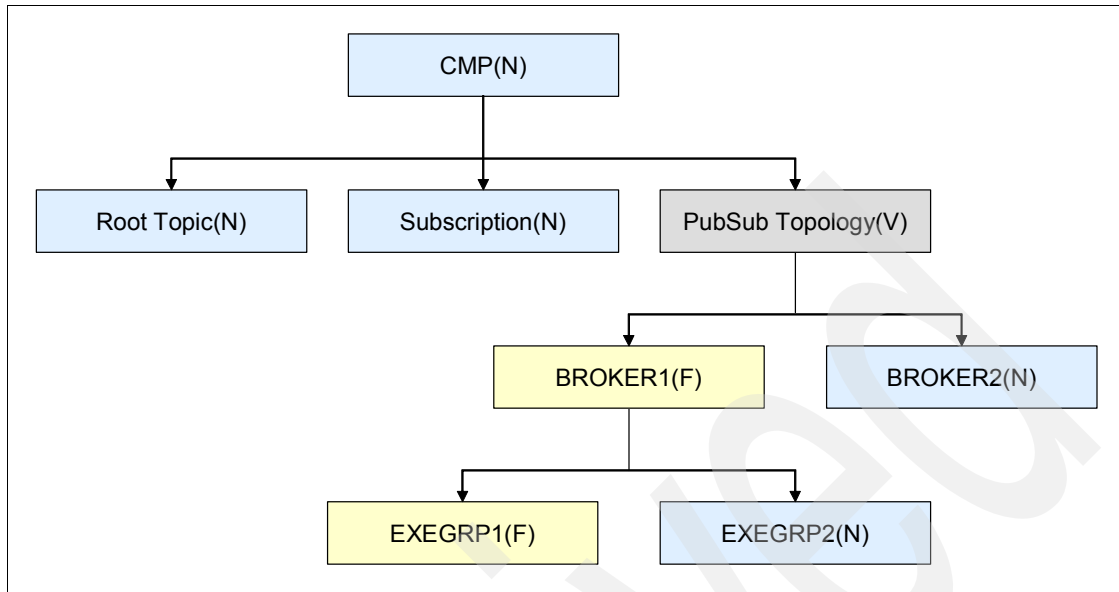


Figure 5-4 ACL example

The control of access for the specified user ID traverses down the tree and follows this path:

- ▶ First, it sets CMP to *None*, which means that the user John cannot manipulate the CMP resource.
- ▶ On the second level, the Root Topic and Subscription are also set to *None*. Topology is set to *view*, which enables John to view the broker topology.
- ▶ When the access control traverses to level 3, the access control for BROKER1 is set to *Full* but BROKER2 access for John is set to *None*.
- ▶ Finally, at execution group level, the EXEGRP1 access control rights for John is set to *Full* access and *None* for EXEGRP2. Thus, John only has access to deploy execution group EXEGRP1 but not to EXEGRP2, although both are members of BROKER1.

Sometimes, some users are in several groups, so the ACLs that are defined to some runtime objects might conflict. When this kind of event occurs, the conflict is solved using the following rules:

- ▶ If the user has an explicit user ACL on the topic of interest, this always takes priority and the broker verifies the current operation on that basis.
- ▶ If the user does not have an explicit user ACL on the topic of interest but has explicit user ACLs against an ancestor in the topic tree, the closest ancestor ACL for that user takes priority, and the broker verifies the current operation on that basis.
- ▶ If there are no explicit user ACLs for the user on the topic of interest or its ancestors, the broker attempts to verify the current operation on the basis of group ACLs:
 - If the user is a member of a group that has an explicit group ACL on the topic of interest, the broker verifies the current operation on the basis of that group ACL.
 - If the user is not a member of a group that has an explicit group ACL on the topic of interest but is a member of a group with explicit group ACLs against an ancestor in the topic tree, the closest ancestor ACL takes priority and the broker verifies the current operation on that basis.
 - If, at a particular level in the topic tree, the user ID is contained in more than one group with an explicit ACL, permission is granted if any of the specifications are positive. Otherwise, it is denied.

Topic-based security

This authorization feature is used when you need to control which applications can access which topics in a publish/subscribe environment. WebSphere Message Broker allows you to specify what user IDs and group IDs have access to the following:

- ▶ Publishing information to a topic
- ▶ Subscribing for information available in a topic
- ▶ Request for persistent delivery of messages
- ▶ Define the level of message protection for a topic

By default, WebSphere Message Broker provides an implicit group, *PublicGroup*, to which all users belong automatically. This implicit group simplifies the specification of ACLs in a topic tree. In particular, this group is used in the specification of the ACL for the topic root. (The default setting of the topic root allows publish/subscribe operations for the *PublicGroup*.) You can change this ACL using the Message Brokers Toolkit, but you cannot remove it. It determines the default permissions for the entire topic tree. You can define the ACLs for the *PublicGroup* elsewhere in the topic tree. When it is used, it defines permissions for all users.

If there is a principal named *Public* that is defined in your existing security environment, you cannot use it for topic-based security. If a principal named *Public* is used within an ACL, it is equated to *PublicGroup* and, therefore, always allows global access.

Another important consideration in topic-based security is that WebSphere Message Broker grants special publish/subscribe access control privileges to members of the *mqbrkrs* group. This group is given implicit privileges so that its members can publish, subscribe, and request the persistent delivery of messages on the topic root and so that all other topics inherit these permissions. If you attempt to change or configure ACL for the *mqbrkrs* group using the workbench, WebSphere Message Broker ignores those changes.

Note: For more information about topic-based security, refer to the WebSphere Message Broker V6.0 Information Center:

<http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp>

5.1.3 Secure Sockets Layer

This section provides information about the implementation of SSL in the WebSphere Message Broker middleware platform. WebSphere Message Broker implements SSL in:

- ▶ Real-time node processing
- ▶ HTTP listener process
- ▶ MQ Java client

SSL on Real-time nodes

WebSphere Message Broker authentication services provide an optional facility that is supported between JMS clients and Real-timeInput nodes of WebSphere Message Broker. To configure the message broker to use the authentication services, follow these steps:

1. Configure and start a User Name Server in a broker domain.
2. Configure each Real-timeInput node to use authentication and set the chosen authentication protocol in each of the brokers that are to use the authentication services.
3. Edit a file that specifies client user IDs and passwords.
4. Specify the names of the files that are required to implement the SSL protocol.

Configuring User Name Server

For information about how to configure the User Name Server, see “Configuring User Name Server” on page 112.

Configuring broker

You configure a broker to support WebSphere Message Broker authentication services. To do so, you specify two authentication and access control parameters and use the workbench to configure the appropriate Real-timeInput nodes and the sets of protocols that are to be supported on the broker.

Follow these steps:

1. Switch to the Broker Administration perspective.
2. For each message flow in the Message Flow Topology:
 - a. Right-click the Real-timeInput node.
 - b. Click **Properties**.
 - c. Select **Authentication**.
3. For each broker in the Broker Topology:
 - a. Right-click the broker.
 - b. Click **Properties**.
 - c. Complete the Authentication Protocol Type field.

Choose any combination of the options *P*, *M*, *S*, and *R*; for example, *S*, *SR*, *RS*, *R*, *PS*, *SP*, *PSR*, *SRM*, *MRS*, and *RSMP* are all valid combinations of these options.

Note: The order in which you specify the options is significant. The broker chooses the first option that the client supports. If you want the broker to always support the strongest protocol that the client supports, choose the combination *RSMP*.

4. If you select *S* or *R* as one of the options in the Authentication Protocol Type field, specify the SSL Key Ring File Name and the SSL Password File Name.
5. Click **OK**.

6. Use the **mqsicreatebroker** or **mqsichangebroker** command, with the following parameters, to configure the broker:

-s UserNameServerQueueManagerName

This parameter specifies the name of the queue manager associated with the User Name Server. You must specify this parameter if authentication services, publish/subscribe access control services, or both types of service is required.

-j Publish/Subscribe Access Control Flag

This flag must be set, in addition to specifying the **-s** parameter, if you want to use publish/subscribe access control services.

Note: There is no corresponding flag for using the authentication services in the broker. It is enabled at the routing input node level.

Sample passwords file

Two sample files are shipped with WebSphere Message Broker:

- ▶ pwgroup.dat is a sample file that you can use when you set the **-j** flag
- ▶ password.dat is a sample file that you can use in the default case

These files are located under the sample\Auth subdirectory of the standard message broker installation. Example 5-1 shows the layout of the password.dat file.

Example 5-1 Layout of the file password.dat

```
# This is a password file.

# Each line contains two required tokens delimited by
# commas. The first is a user ID, the second is that user's
# password.

#USERNAME PASSWORD
=====
subscriber,subpw
admin,adminpw
publisher,pubpw
```

The password.dat file compliments the user and group information that is drawn by the User Name Server from the operating system. User names that are defined in the file, but not the operating system, are treated as unknown by the

broker domain. User names that are defined in the operating system, but are not defined in the password file, are denied access to the system.

In contrast, the `pwgroup.dat` file contains group information as well as user and password information. Each user entry includes a list of group names that specify the groups that contain the user. Example 5-2 shows the layout of the `pwgroup.dat` file.

Example 5-2 Layout of the file `pwgroup.dat`

```
#This is a password file.  
#Each line contains two or more required tokens delimited by  
#commas. The first is a user ID and the second is that user's  
#password. All subsequent tokens  
#specify the set of groups that the user belongs to.  
  
#USERNAME PASSWORD GROUPS  
subscriber,subpw,group1,group2,group3  
admin,adminpw,group2  
publisher,pubpw,group2,group4
```

You can use this file to provide the only source of user, group, and password information for the broker domain.

To deploy updated user and password information to the broker network if this information is drawn from an operating system file, stop the User Name Server and brokers, update the file, and then restart the User Name Server and brokers.

If passwords are drawn from the operating system, updates are distributed automatically to the brokers. Use normal operating system management tools to change users or passwords.

Authentication in JMS client

For client applications that use WebSphere MQ classes for Java Message Service Version 5.3 before CSD4, the client application always has an authentication protocol level of *PM*. The client application and broker negotiate on the choice of protocol for a session. Where the broker supports both protocols (that is, either *PM* or *MP* is set in the workbench definition of a broker), the first protocol specified in the workbench is chosen.

For client applications that use WebSphere MQ classes for Java Message Service Version 5.3, CSD 5 or later, the client application supports two levels of authentication.

A TopicConnectionFactory can be configured to support either a MQJMS_DIRECTAUTH_BASIC authentication mode or a MQJMS_DIRECTAUTH_CERTIFICATE authentication mode. The MQJMS_DIRECTAUTH_BASIC authentication mode is equivalent to a level of *PM* and the MQJMS_DIRECTAUTH_CERTIFICATE authentication mode is equivalent to a level of *SR*.

If authentication services have been successfully configured for a Real-timeInput node, a JMS client application needs to specify its credentials when creating a connection. To do this, the JMS client application supplies a user/password combination to the TopicConnectionFactory.createTopicConnection method; for example:

```
factory.createTopicConnection("user1", "user1pw");
```

If credentials are not specified or are specified incorrectly, the application receives a JMS wrapped exception containing the MQJMS error text.

SSL on HTTP listener

The WebSphere Message Broker provides a facility to enable the communication between message flows and Web services in a secure way. This feature provides the capability for a message flow to communicate with a Web service in two ways:

- ▶ Using an HTTP Input node to encapsulate a message flow inside a Web service invocation.
- ▶ Using an HTTP Request node to invoke an external Web service and an HTTP Reply node to get its answer and pass it to another nodes inside a message flow.

The necessary steps to enable SSL on HTTP listener are:

1. Create a key store file.
2. Configure the broker to use an SSL port.
3. Change the message flows nodes.

Creating a keystore file to store the broker's certificates

WebSphere Message Broker includes a Java Runtime Environment (JRE) that supplies a keystore manipulation program, which is called **keytool**. To invoke the **keytool** command:

1. Select **Start** → **IBM WebSphere Message Brokers 6.0** → **Command Console** to open the broker command console.
2. In the command console, type the following command:

```
"%MQSI_FILEPATH%\jre\bin\keytool"
```


The help options display and validate that the command is working.

3. Use the **keytool** command to create the key store. In the command console, type the following command:

```
"%MQSI_FILEPATH%\jre\bin\keytool" -genkey -keypass <password>  
-keystore <keystore_file> -alias tomcat
```

Where:

- *password* is the keystore password
- *keystore_file* is the fully qualified name of the keystore file. This file is normally called *keystore* and is usually located in the message broker users home directory.

The command then prompts for some personal details that are used to generate the certificates. When you have entered all of the values, the tool should either generate or add (if one already exists) a key store.

Note: You can set these values to any value that is required, but the properties on the broker must be changed to reflect these values. The **-genkey** option generates all the certificate files that are required to get HTTPS working. However, the certificate files are not official certificates, so you should not use them in a production system. You must purchase a real certificate from a certificate organization. Consult your system administrator, because every company has its own policies for creating and using certificates. To import a certificate that is generated by a certificate authority use the **-import** option instead of the **-genkey** option.

The keystore is now created and is ready for use by the broker.

Configuring the broker to use SSL on a particular port

To make use of HTTP over SSL, you must set several properties on the broker. You can change these properties using the **mqsichangeproperties** command as follows:

- Turn on SSL support in message broker by setting a value for `enableSSLConnector`:

```
mqsichangeproperties <broker_name> -b httplistener -o HTTPListener  
-n enableSSLConnector -v true
```
- Choose the keystore file to be used by setting a value for `keystoreFile`:

```
mqsichangeproperties <broker_name> -b httplistener -o HTTPSConnector  
-n keystoreFile -v <fully_qualified_path_to_keystore_file>
```

- ▶ Specify the password for the keystore file by setting a value for keystorePass:
`mqschangeproperties <broker_name> -b httplistener -o HTTPSConnector -n keystorePass -v <password_for_keystore>`
- ▶ Specify the port on which the broker needs to listen for HTTPS requests:
`mqschangeproperties <broker_name> -b httplistener -o HTTPSConnector -n port -v <port_to_listen_on_for_HTTPS>`

Ensure that all of these properties are set with the correct values. You only have to set the `enableSSLConnector` property. The other properties have default values. However, you should change these default values. The `mqschangeproperties` command lists the default values for all the properties.

Configuring HTTPRequest node to use SSL

You can configure the HTTPRequest node to communicate with other applications using HTTP over SSL. Here, we describe how to configure the node for a Windows system, but almost identical steps are required for other platforms. To complete these steps, an HTTPS server application is required. For simplicity, the only details that we give here are for using the HTTPInput node for SSL as the server application. However, the same details also apply when you are using any other server application.

To configure HTTPRequest node to use SSL, follow these steps:

1. Add certificates to the cacerts file.

The certificate for the server application to be called must be added to the cacerts file for WebSphere Message Broker, which is located in the JRE security directory. To find the cacerts file on Windows:

- a. Select **Start → IBM WebSphere Message Brokers 6.0 → Command Console** to open a broker command console.
- b. In the command console, enter the following command to change to the directory to where the cacerts files is located:

```
cd "%MQSI_FILEPATH%\jre\lib\security"
```

c. Use the **keytool** command to modify the cacerts file:

```
"%MQSI_FILEPATH%\jre\bin\keytool" -import -alias mykey -file  
<name_of_certificate_file> -keystore cacerts -keypass changeit
```

Where:

- *name_of_certificate* file is the fully qualified name of the certificates file
This file is normally found in the message broker users home directory.
- *changeit* is the default password for the cacerts file
Change this password as soon as possible. You can use **keytool** to change the password.

It is important to ensure that you have imported the correct certificate into the cacerts file. The correct certificate is the certificate that the HTTP server should use.

2. Create a message flow to make HTTPS requests.

The following message flow creates a generic message flow for converting an WebSphere MQ message into an HTTPRequest:

- Create a message flow with the nodes MQInput → HTTPRequest → Compute → MQOutput.
- For the MQInput node, set the queue name to HTTPS.IN1 and create the WebSphere MQ queue.
- For the MQOutput node, set the queue name to HTTPS.OUT1 and create the WebSphere MQ queue.
- For the HTTPRequest node, set the Web Service URL to point to the HTTP server to call. For calling the HTTPInput task use:
`https://localhost:7083/testHTTPS`
- For the HTTPRequest node, set the *Advance properties* to use *OutputRoot.BLOB* as the *Response location* in tree.
- In the compute node add the ESQL code listed in Example 5-3.

Example 5-3 ESQL example code for test HTTPS

```
CREATE COMPUTE MODULE test_https_Compute  
  CREATE FUNCTION Main() RETURNS BOOLEAN  
  BEGIN  
    -- CALL CopyMessageHeaders();  
    CALL CopyEntireMessage();  
    set OutputRoot.HTTPResponseHeader = null;  
    RETURN TRUE;  
  END;
```

```

CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER;
    DECLARE J INTEGER;
    SET I = 1;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I = I + 1;
    END WHILE;
END;

CREATE PROCEDURE CopyEntireMessage() BEGIN
    SET OutputRoot = InputRoot;
END;
END MODULE;

```

The message flow is now ready to be deployed to the broker and tested.

3. Test your example.

To test that the example works, complete the following steps:

- a. Follow the instructions in steps 1 and 2.
- b. Deploy the HTTPRequest message flow.
- c. Put a message to the WebSphere MQ queue HTTPS.IN1. A message should appear on the output queue. If it fails, an error appears in the local error log (event log on windows).

SSL on WebSphere MQ Java Client

The WebSphere MQ Java Client supports SSL encrypted connections over the SVRCONN channel between the application and the queue manager. This topic describes how to use this SSL support when communicating between the Configuration Manager Proxy (CMP) and the Configuration Manager.

For one-way authentication with the client (Configuration Manager Proxy) authenticating the server (Configuration Manager) only, perform the following steps:

1. Generate or obtain all the appropriate keys and certificates, including a signed pkcs12 certificate for the server and the appropriate public key for the certificate authority that signed the pkcs12 certificate.
2. Add the pkcs12 certificate to the queue manager certificate store and assign it to the queue manager. Use the standard WebSphere MQ facilities (for example, WebSphere MQ Explorer).

3. Add the certificate of the certificate authority to the JSEE truststore of the Java Virtual Machine (JVM) at the Configuration Manager Proxy end using a tool such as **keytool**.
4. Decide which cipher suite to use.
5. Change the properties on the SYSTEM.BKR.CONFIG channel, using WebSphere MQ Explorer, to specify the cipher suite to be used.
6. Add the required parameters (cipher suite, for example) to the Configuration Manager Proxy. If a truststore other than the default is used, its full path must be passed in via the truststore parameter.

When these steps have been performed, the Configuration Manager Proxy connects to the Configuration Manager if it has a valid signed key that has been signed by a trusted certificate authority.

For two-way authentication (with the Configuration Manager also authenticating the Configuration Manager Proxy) perform the following additional steps:

1. Generate or obtain all the appropriate keys and certificates. This includes a signed pkcs12 certificate for the client and the appropriate public key for the certificate authority that signed the pkcs12 certificate.
2. Add the certificate of the certificate authority to the queue manager certificate store; use the standard WebSphere MQ facilities (for example, WebSphere MQ Explorer).
3. Set the SYSTEM.BKR.CONFIG channel to always authenticate. Change the parameter SSLCAUTH(REQUIRED) with WebSphere MQ Explorer runmqsc command.
4. Add the pkcs12 certificate to the JSEE keystore of the JVM at the Configuration Manager Proxy end using a tool such as **keytool**.
5. If not using the default keystore, its full path must be passed into the Configuration Manager Proxy via the keystore parameter.

When you have performed these steps, the Configuration Manager allows the Configuration Manager Proxy to connect only if the Configuration Manager Proxy has a certificate signed by one of the certificate authorities in its keystore.

Further restrictions can be made using the sslPeerName field. For example, connections can be allowed only from certificate holders with a specific company or department name in their certificates. In addition, a security exit can be invoked for communications between the Configuration Manager Proxy and the Configuration Manager. For more information, see “Security exits” on page 88.

5.1.4 Message protection

The only feature inside the WebSphere Message Broker to protect the information from is the *Quality Protection of Messages*.

In Internet deployments, cryptographically based protection of messages enhances security by preventing tampering and eavesdropping by hackers. The authentication services that WebSphere Message Broker provides ensure that only legitimate event broker servers and clients can connect to each other. However, a hacker might still be able to observe messages in transit or tamper with messages on established connections. Message protection provides security against these kinds of attacks.

Message protection consumes processor time and can slow system throughput. However, not all messages are equally sensitive. Message protection is configurable on a per-topic basis, so that you only get the protection that is really needed. For example:

- ▶ Some topics might get no message protection at all, while others might get channel integrity (making it impossible for hackers to insert or delete messages undetected).
- ▶ Message integrity make it impossible for hackers to alter messages undetected.
- ▶ Message privacy makes it impossible for hackers to observe message contents.

The protection levels are cumulative. For example, if there is a request for message privacy, then message integrity and channel integrity are included as well. If there is a request for message integrity, then channel integrity is included as well. The higher levels of protection consume more resources than the lower levels.

Message protection can also be set on internal system topics. Unlike user topics, this message protection must be either enabled on all topics or on none.

Implementing quality of protection

The `enableQoPSecurity` option of the `mqsichangeproperties` command enables quality of protection. By default, quality of protection is enabled if any of the quality of protection settings have been changed from *n* or *none*. The levels of message protection are defined using the `sysQoPLevel` and `isysQoPLevel` options, also in the `mqsichangeproperties` command.

From the Broker properties window, you can set the values for temporary topics using the following:

- ▶ Temporary Topic Quality of Protection
- ▶ Sys Topic Quality of Protection
- ▶ ISys Topic Quality of Protection

The default value is *none*, or you can select one of the following values from each of the lists:

- ▶ *Channel Integrity* (which prevents hackers from inserting or deleting messages without being detected)
- ▶ *Message Integrity* (which prevents hackers from changing the content of a message without being detected)
- ▶ *Encrypted for Privacy* (which prevents hackers from looking at the content of a message)

The value that you select is the same for all users and is independent of the value that the user has selected.

5.1.5 Complementary options

In addition to the security options that WebSphere Message Broker V6.0 provides, there are other complementary options that you can configure to obtain a more secure message broker middleware platform. The most important of these options are:

- ▶ WebSphere MQ SSL
- ▶ WebSphere MQ Object Authority Manager (OAM)
- ▶ WebSphere MQ Extended Security Edition

WebSphere MQ SSL

WebSphere MQ allows channels of all types to be secured using SSL. These types of channels include distributed message channels, cluster message channels, and client channels. The main uses of SSL in WebSphere MQ are:

- ▶ To provide authentication of the identity of a remote application or queue manager
- ▶ To assure confidentiality and data integrity for the communication link after it is established

A channel is configured to require SSL authentication by specifying a single *CipherSpec* string in the SSL cipher specification (SSLCIPH) attribute of the channel object. The name of a CipherSpec identifies the encryption algorithm

and Message Authentication Code (MAC) algorithm that is used after the channel is established.

Each CipherSpec that is available in WebSphere MQ correlates with a *CipherSuite* that is used by other SSL applications, where that CipherSuite uses the RSA key exchange protocol. Different CipherSpecs provide different levels of security and performance.

For more information about the CipherSpecs that are available, refer to *WebSphere MQ Security*, SC34-6588, which is available at:

<http://www.ibm.com/software/integration/wmq/library/library6x.html>

Note: For more information about the WebSphere MQ SSL, refer to *WebSphere MQ V6 Fundamentals*, SG24-7128, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247128.html?Open>

WebSphere MQ Object Authority Manager

The WebSphere MQ Object Authority Manager is a component inside any queue manager of the WebSphere MQ platform. Whenever an entity that is connected to the queue manager attempts to perform an action against a WebSphere MQ object, the Object Authority Manager is queried to determine if that entity has authority to perform the action. Each action that can be performed against an object has an associated authority that can be granted or revoked using the Object Authority Manager. The following actions can be performed against WebSphere MQ objects by entities connected to a queue manager:

- ▶ Connecting to a queue manager
- ▶ Message queue interface (MQI) operations
- ▶ Performing actions under a different user context
- ▶ Performing administrative operations on WebSphere MQ objects

Note: For more information about the WebSphere MQ Object Authority Manager, refer to *WebSphere MQ V6 Fundamentals*, SG24-7128, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247128.html?Open>

WebSphere MQ Extended Security Edition

IBM Tivoli Access Manager for Business Integration is a multi-platform security management solution for WebSphere MQ that upgrades the native security services of WebSphere MQ to those provided by WebSphere MQ Extended Security Edition. It provides application-level data protection for WebSphere MQ based applications, without the need to modify or even recompile them.

Application-level data protection differs from link-level or channel-level data protection in that the integrity and confidentiality of messages can be demonstrated, not just while messages are in transit from system to system but also while they were under the control of WebSphere MQ itself (for example, resident in a queue). This protection is critical for customers who are using WebSphere MQ to process personally identifiable information or other types of sensitive data, such as high value financial transactions.

Tivoli Access Manager for Business integration allows you to remotely manage which applications can put and get messages from specific queues or queue managers. When an application makes a call to the WebSphere MQ interface to put a message in the queue, Tivoli Access Manager for Business Integration intercepts and analyzes the call to verify whether the sending application is authorized to access the requested queue. If the call is authorized, it determines (based on a policy you define) whether the data in the transaction should be digitally signed, signed and encrypted, or passed on unchanged before placing the message in the requested queue.

Administration of these security policies is done remotely, using a Web-based tool that replaces the need for administrators to visit each physical system. Tivoli Access Manager for Business Integration is designed to support applications that are written to the WebSphere MQ native programming application program interface (API) and the Java Messaging Service API (bindings mode on distributed servers and 100% Java mode on distributed clients). It supports systems running WebSphere MQ queue manager services as well as systems running the WebSphere MQ client on a variety of platforms.

Note: For more information about WebSphere MQ Extended Security Edition, refer to *WebSphere MQ Security in an Enterprise Environment*, SG24-6814, which is available at:

<http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SG24-6814>

5.2 Characteristics for message broker component

This section discusses specific security information, not covered in 5.1, “Securing the domain”, that are related directly with some of the message broker components.

5.2.1 Broker

We discuss the following specifics for the broker:

- ▶ Authorization
- ▶ Setup considerations

Authorization

When the service user ID is set with the **-i** option on the **mqsicreatebroker** or **mqsichangebroker** command, it determines the user ID under which the broker component process runs. If you will have the service user ID as the message broker administrator, it is recommended to include the user ID as part of the **mqbrkrs** group ID. Doing so gets the user ID access authority to the following message broker queues:

- ▶ SYSTEM.BROKER.ADMIN.QUEUE
- ▶ SYSTEM.BROKER.CONTROL.QUEUE
- ▶ SYSTEM.BROKER.INTERBROKER.QUEUE
- ▶ SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
- ▶ SYSTEM.BROKER.EXECUTIONGROUP.REPLY
- ▶ SYSTEM.MODEL.BROKER.QUEUE

By assigning the **mqbrkrs** group ID to the user ID, it also gives access to the Message Broker commands (for example **mqsistart**, **mqsistop**, **mqsichangebroker**, **mqsideletebroker** and other **mqsi** commands).

Setup considerations

The following lists some of the issues that you should consider when managing and maintaining a broker component in a production environment:

- ▶ If a work path other than the default has been set for any component, ensure that the services user ID has appropriate access to this location.
- ▶ All Message Brokers Toolkit users need read access to the WebSphere MQ Java \lib subdirectory of the WebSphere MQ home directory (the default is *X:\Program Files\WebSphere MQ*, where *X*: is the operating system disk). This access is restricted to users in the local group *mqm* by WebSphere MQ. WebSphere Message Broker installation overrides this restriction and gives read access for this subdirectory to all users.
- ▶ If there are WebSphere Message Broker components running on different queue managers, it is necessary to set the permissions for the transmission queues defined to handle the message traffic between the queue managers. Grant put and setall authority to the local **mqbrkrs** group, or to the service user ID of the component supported by the queue manager on which is the transmission queue defined. Use the Object Authority Manager, for example

setmqaut utility to assign relevant ACL entry for the WebSphere MQ manager resources.

- ▶ As for access control over the resources such as message flows, you must ensure that the broker's service user ID has authority to:
 - put and get messages from each input queue defined in an MQInput and MQOutput node included in a message flow
 - put and get messages to any output, reply, and failure queues, for example MQOutput, MQReply and MQFailure included in a message flow
- ▶ As for the user ID used to run the application that interacts with the broker, you must ensure the user ID has authority for the followings:
 - get authority to each output queue identified in an MQOutput node or an MQReply node to the user ID under which a receiving or subscribing client application runs
 - put authority to each input queue identified in an MQInput node to the user ID under which a sending or publishing client application runs

5.2.2 Configuration Manager

When setting up the Configuration Manager, you need to decide which user ID you want to use (for example create, change, list, delete, start, and stop a Configuration Manager and also display, retrieve, and change trace information). The command to assign an user ID when you create the Configuration Manager is **mqsi createconfigmgr**. This command sets the user ID as the service ID to run the Configuration Manager. When you execute the command, it grants put and get authority on your behalf to the group mqbrkrs for the following queues:

- ▶ SYSTEM.BROKER.CONFIG.QUEUE
- ▶ SYSTEM.BROKER.CONFIG.REPLY
- ▶ SYSTEM.BROKER.ADMIN.REPLY
- ▶ SYSTEM.BROKER.SECURITY.REPLY
- ▶ SYSTEM.BROKER.MODEL.QUEUE
- ▶ SYSTEM.MODEL.BROKER.QUEUE

5.2.3 User Name Server

We discuss the following specifics for the User Name Server:

- ▶ Configuration of the User Name Server
- ▶ Authorization

Configuring User Name Server

The User Name Server distributes to the brokers the information (specifically, passwords) that is required to support these authentication protocols. To configure the User Name Server to support authentication, two parameters are provided for the `mqsi createusernameserver` and `mqsi changeusernameserver` commands:

- ▶ The first parameter, the `-s` flag, `AuthProtocolDataSource`, describes the location of an operating system file that contains the information that is required to support the authentication protocols.
- ▶ The second parameter, the `-j` flag, indicates whether the file that is pointed to by the `AuthProtocolDataSource` parameter contains group and group membership information as well as password information.

Set the `-j` flag if you need to support both authentication and publish/subscribe access control in the broker domain. User and group information need to be drawn from a flat file rather than from the operating system.

Some comments about these parameters:

- ▶ Use the `AuthProtocolDataSource` parameter to specify the source of any protocol-related information. For example, you can specify the name of a file that contains user ID and password information. The user ID and password information in this file must mirror exactly the operating system user ID and password definitions. Make sure that the appropriate file system security is set for this password file.
- ▶ The default location of this file is the WebSphere Message Broker home directory. If you store the file elsewhere, give the full path name of the file.
- ▶ For the change to the User Name Server to take effect, the User Name Server must be stopped and then restarted.

The `mqsi changeusernameserver` command also supports the `-d` flag to integrate the information directly with a Windows domain.

Authorization

When you execute the command `mqsi createusernameserver`, it grants put and get authority on your behalf to the `mqbrkrs` group for the following queues:

- ▶ `SYSTEM.BROKER.SECURITY.QUEUE`
- ▶ `SYSTEM.BROKER.MODEL.QUEUE`

If the User Name Server is required to access the Configuration Manager's or broker's queues, there is necessary to ensure each group or user included in the create access control lists must have access authority to the

SYSTEM.BROKER.CONFIG.QUEUE and SYSTEM.BROKER.CONFIG.REPLY for the respective components.

5.2.4 Toolkit

The specific security feature that can be enabled in the Message Brokers Toolkit configuration is the domain awareness. Enabling domain awareness means that access to the Message Brokers Toolkit is not restricted to users from one domain. With domain awareness enabled, users from different domains who are either trusted by the primary domain or in a Windows 2000 transitive trust relationship can perform Message Brokers Toolkit tasks, provided that they are in the correct user role definition groups or object level security groups.

When using domain awareness:

- ▶ Domain information is retrieved by the Configuration Manager
- ▶ Membership of user role definition or object level security groups is not restricted to users from one domain
- ▶ Nesting in user role definition or object level security groups is supported

Note: It is recommended that you run the Message Brokers Toolkit with domain awareness enabled. With this option, the domain information for a workbench user is flowed with the user ID to the Configuration Manager for increased security.

To set up the domain awareness, let us assume that the Configuration Manager is running on a computer named WKSTN1, which is a member of a domain named DOMAIN1, and that users from DOMAIN2 also want to use the workbench. To configure the domain awareness, perform the following steps:

1. Add any domain users or groups to the local group names that need to be used in your ACLs:
 - If users using workbench are from local domain, then add these users to the local groups that are used on the workbench ACLs.
 - If users using workbench are from another domains, the best way to secure the access is to make the other domain a trusted domain of the Configuration Manager's computer and then add the remote groups or users to the local groups that being used on the workbench ACLs.
2. When creating the Configuration Manager, use the **-m** option on the **mqsicreateaclentry** command to ensure that the domain is considered when verifying the user.

When the workbench is started, it sends the domain information for your user ID to the Configuration Manager automatically.

Attention: You can disable the domain awareness, but running with this feature means that the domain information for the workbench user is not flowed with the user ID information, thus reducing security. It is, therefore, recommended to run with domain awareness enabled. Use the **-a** option on the **mqscreateaclentry** command to allow a user to be verified without considering the domain.

Backup, restore, and recover

Data backup and restore is an essential part of operations for any business system. As with any production-critical programs, you must plan backup and recovery for data in the event of a system crash. When considering backup and recovery policies, you should evaluate, identify, and prioritize business critical data and ask the following questions:

- ▶ Are there any existing procedures in place?
- ▶ What procedures must be defined and used to backup business data?
- ▶ What must be the frequency of the backups?
- ▶ What are the trade-offs between online and offline backups?
- ▶ What effect does the deployment, for example, the number of users and the amount and importance of data have on the backup and recovery strategy?

In this chapter, we discuss the basic backup and restore strategies for the WebSphere Message Broker V6.0 components, which include:

- ▶ Broker
- ▶ Configuration Manager
- ▶ User Name Server
- ▶ Message Brokers Toolkit

6.1 Backing up the domain

Brokers rely on a database engine to maintain and control their configuration data. Brokers, the Configuration Manager, and the User Name Server rely on WebSphere MQ to transport and guarantee messages between components. Ensure that a backup process is established that includes these sources of information to preserve the integrity and consistency of the broker domain. In addition to backing up the broker domain, you must also back up any development resources, such as message flow files, message set definition files, ESQL files, mapping files, XML Schema files and broker archive files.

Details about how to perform backups are described in 6.3, “Characteristics for message broker components” on page 119 for each message broker component.

6.1.1 Backup options for database

The broker supports online backups for its database. However, all the brokers that share the database must be stopped before taking the database backup. Stopping the broker ensures that the broker is in consistent state after the database is restored. We refer to this type of backup as an *online backup* because the database instance is still in service while the backup is taken. If the deployment is large or if you need to maintain continuous availability, then an online backup is a better choice. For more information about online and offline database backup considerations, refer to the DB2 information center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp>

Because the Configuration Manager uses its own built-in database engine, there is a special backup facility using the command `mqsibackupconfigmgr`.

6.1.2 Scheduling and expiration

You must define the backup scheduling policy in a production environment so that in the event of a failure, you can restore the message broker environment to its most recent status. Consider the following points while deciding your backup scheduling policy:

- ▶ Migrating to a newer version of product or fix pack.
- ▶ Deploying newer versions of message broker resources (message flows and any other deployable objects).
- ▶ Periodic backups (for example, weekly or monthly).

You must take the backups before and after you change the broker domain. The backup before the change is important for restore and recovery in case of error,

and the backup after the change is the first regular backup for the new version of environment.

Similar to the backup scheduling policy, you should put in place an expiration policy. Consider the following points when determining an expiration policy:

- ▶ Is the message broker environment migrated successfully to the next fix pack or newer version of product? If yes, take the backup of this new environment and the previous backups can be expired or discarded.
- ▶ Keep the last two successful backups and delete the older backups.
- ▶ If the backups are very old and are obsolete to the current production environment, you can discard them.

We describe here an example backup scheduling and expiration strategy known as *three generations strategy* (also known as *grandfather, father, and son* strategy). A multi-generation backup copies technique is adopted commonly in a production environment.

Table 6-1 describes a multi-level backup strategy that is practiced by most organizations (how they maintain and how long they keep their backups) for a production environment. Note that the frequency in term of days, weeks, or months depends on your organization requirements and business needs.

Table 6-1 Three generation backup strategy

Frequency	Daily	Weekly	Monthly
Week 1			
Monday	Tape_D_1		
Tuesday	Tape_D_2		
Wednesday	Tape_D_3		
Thursday	Tape_D_4		
Friday	Tape_D_5	Tape_W_1	Tape_M_1
Week 2			
Monday	Tape_D_6		
Tuesday	Tape_D_7		
Wednesday	Tape_D_8		
Thursday	Tape_D_9		

Frequency	Daily	Weekly	Monthly
Friday	Tape_D_10	Tape_W_2	
Week 3			
Monday	Tape_D_11		
Tuesday	Tape_D_12		
Wednesday	Tape_D_13		
Thursday	Tape_D_14		
Friday	Tape_D_15	Tape_W_3	
Week 4			
Monday	Tape_D_1		
Tuesday	Tape_D_2		
Wednesday	Tape_D_3		
Thursday	Tape_D_4		
Friday	Tape_D_5	Tape_W_1	Tape_M_2

Table 6-1 shows a metric that describes how long the backup resources are retained. It gives you an idea of how the three generations backup strategy is practiced. The daily column has the backup of data that changes on a daily basis, for example Toolkit workspace and broker database. The weekly column backup includes data that does not change so frequently, for example Configuration Manager and WebSphere MQ configurations. Finally, the monthly column backup includes the entire broker environment.

6.2 Restoring and recovering the domain

In this section, we discuss various restore and recover scenarios for the message broker components. This section discusses only the difference between the terms *restoring* and *recovering*. For information about how to perform these procedures, see 6.3, “Characteristics for message broker components” on page 119.

6.2.1 Restoring from backups

Here we discuss how to restore the broker and the Configuration Manager from the available database and filesystems backups. When restored, the components are at their state when the last backup was taken.

6.2.2 Recovering from disaster

Recovering from disaster is attributed to the situation where we do not have the backup of database or filesystems in which are the message broker components and queue managers stored. It mainly involves re-creating the components.

6.3 Characteristics for message broker components

In this section, we describe the steps for backing up and restoring the message broker components.

6.3.1 Broker

This section describes backing up, restoring and recovering the broker. This is further illustrated with examples in 10.4.1, “Backing up broker” on page 267.

Backing up broker

Perform the backup of the broker using the following steps:

1. Choose a safe storage location, such as a tape drive or DVD, to store the backup information.
2. Before taking the backup, ensure that no deploy process is running to the broker and then stop the broker issuing the **mqsistop** command.
3. Backup the broker database using database backup commands.
4. Backup the registry information for the broker. The registry holds the vital information about each runtime components UUID values (this is useful while restoring the components from the failures). Use the following commands:

UNIX:

```
/var/mqsi/registry/<broker_name>
```

Windows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphereMQIntegrator\2\  
<broker_name>
```

z/OS:

<broker_directory>/registry/<broker_name>

5. Along with the registry backup, take broker service information using following command:

```
mqsIService <broker_name>
```

For future reference and restore operation, save the output from this command. The Component UUID field denotes the UUID of the broker.

6. The queue manager backup is also important part of the broker component. Refer to WebSphere MQ 6.0 Information Center for the backup instructions:

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/basto.htm>

Restoring broker from backup

If the database is not corrupted but you cannot correct the problem using problem determination techniques, perform the following sequence of operations to re-create the broker:

1. Ensure that no deploy process is running.
2. Stop the broker using the **mqsistop** command
3. Take a backup of the broker database. Individual tables cannot be backed up or restored.
4. Delete the broker using the following command.

On Windows and UNIX systems:

```
mqsidedeletebroker <broker_name> -w
```

Specify the **-w** parameter to delete all files from the broker workpath.

On z/OS systems:

```
mqsidedeletebroker <broker_name> 1 3
```

Specify the **1** parameter to delete the broker registry and the **3** parameter to delete the broker DB2 tables and indexes.

5. Re-create the broker with the same name using the **mqsicreatebroker** command.
6. Restore the broker database tables, either from the backup that you have just taken or from a previously successful backup version.
7. Run the **mqsIService** command for the broker using the **-r** option to set the BrokerUUID to the value noted during backup (step 5 above).
8. Start the broker using the **mqsistart** command.

9. Redeploy the domain configuration to ensure that the configuration across the broker domain is consistent.

Recovering broker when backup not available

If the broker database fails, cannot be corrected using problem determination techniques, and the broker backup is not available, perform the following sequence of operations to re-create the broker:

1. Ensure that no deploy is running to the broker.
2. If the broker processes are still running, stop the broker using the **mqsistop** command.
3. Delete the broker using the following command.

On Windows and UNIX systems:

```
mqsdeletebroker <broker_name> -w
```

Specify the **-w** parameter to delete all files from the broker workpath.

On z/OS systems:

```
mqsdeletebroker <broker_name> 1 3
```

Specify the **1** parameter to delete the broker registry and the **3** parameter to delete the broker DB2 tables and indexes.

4. Use the Configuration Manager Proxy API to remove all references to the broker:
 - a. Start the *Configuration Manager Proxy API Exerciser* sample (which is available on the **Start** menu on Windows).
 - b. Connect to the Configuration Manager.
 - c. Right-click the topology object, and then click **Remove references to a previously deleted broker**.
5. Re-create the broker using the **mqsicreatebroker** command.
6. Start the broker using the **mqsistart** command.
7. Add the broker to the Configuration Manager domain topology.
8. Redeploy the broker configuration.

Recovering after broker queue manager fails

Important: If the failed queue manager is shared between a broker, User Name Server, and Configuration Manager, re-create all three components, because each component creates different queues.

If the broker queue manager fails and you cannot correct the problem using problem determination techniques, restore the queue manager from its backup and then restart the broker. If the queue manager backup is not available, perform the following sequence of operations to re-create the broker:

1. Ensure that no deploy is running to the broker.
2. Stop the broker using **mqsistop** command, if running.
3. Take a backup of the broker database.
4. Run **mqsiservice** command for the broker and note the broker UUID value.

```
mqsiservice <broker_name>
```

The Component UUID field denotes the UUID of the broker.

5. Delete the broker using the following command.

On Windows and UNIX systems:

```
mqsdeletebroker <broker_name> -q
```

Specify the **-q** parameter to delete the broker queue manager.

On z/OS systems:

```
mqsdeletebroker <broker_name> 1 2 3
```

Specify the **1** parameter to delete the broker registry, the **2** parameter to delete the broker queues, and the **3** parameter to delete the broker DB2 tables and indexes.

6. Re-create the broker using the **mqsicreatebroker** command. The **mqsicreatebroker** command creates the queue manager and default queues automatically.
7. Restore the broker database tables from the backup that was just taken in step **3**.
8. Run **mqsiservice** command for the broker using the **-r** option to set the BrokerUUID to the value noted during backup (step **4**).

```
mqsiservice <broker_name> -r BrokerUUID=<'broker_uuid'>
```
9. Start the broker.
10. Redeploy the domain configuration.

6.3.2 Configuration Manager

This section describes backing up, restoring, and recovering the Configuration Manager.

Backing up Configuration Manager

The command **mqsibackupconfigmgr** backs up the Configuration Manager repository and uses the Configuration Manager Proxy to communicate with the Configuration Manager.

The default name of the backup image is in the form:

`<ConfigMgr_name>_<date>_<time>.zip`

Perform the backup of Configuration Manager using the following steps:

1. Choose a safe storage location, such as a tape drive or DVD, to store the backup information.
2. Before taking the backup, ensure that no deploy process is running to any broker and then stop the Configuration Manager issuing the **mqsi stop** command.
3. Backup the Configuration Manager repository using the following command (Example 6-1):

On Windows and UNIX systems:

```
mqsibackupconfigmgr <ConfigMgr_name> -d <backup_directory_name>
```

On z/OS systems:

```
mqsibackupconfigmgr <ConfigMgr_name> d=<backup_directory_name>
```

The **-d** or **d=** parameters specify the directory where the archive is placed.

Example 6-1 Backing up the Configuration Manager

```
C:\Program Files\IBM\MQSI\6.0>mqsibackupconfigmgr CFGMGR -d c:\backup
```

```
BIP1075I: Creating backup archive for Configuration Manager 'CFGMGR' in  
directory 'c:\backup' ...
```

```
BIP1017I: A backup archive called 'CFGMGR_060615_140958.zip' was  
created successfully.
```

```
BIP8071I: Successful command completion.
```

```
C:\Program Files\IBM\MQSI\6.0>
```

4. The queue manager backup is also important part of the Configuration Manager component. Refer to WebSphere MQ 6.0 Information Center for the backup instructions:

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/basto.htm>

Note: If you plan to restore the Configuration Manager data repository on a platform other than z/OS, take a copy of the file below:

```
<ConfigMgr_directory>/components/<component_name>/  
<directory_name>/service.properties
```

This file needs to be kept with the zipped file that is produced by the **mqsibackupconfigmgr** command and must be copied to the equivalent place under the Configuration Manager location after restoring the Configuration Manager repository upon issuing the **mqsirestoreconfigmgr** command.

Restoring Configuration Manager from backup

If the Configuration Manager repository is not corrupted but you cannot correct the problem using problem determination techniques, perform the following sequence of operations to re-create the Configuration Manager:

1. Ensure that no deploy is running to any brokers.
2. Stop all brokers that are using the Configuration Manager using the **mqsistop** command.
3. Stop the Configuration Manager using the **mqsistop** command, if running.
4. Re-create the Configuration Manager repository from a backup using the following command on the same machine as the Configuration Manager (Example 6-2 on page 125):

On Windows and UNIX systems:

```
mqsirestoreconfigmgr <ConfigMgr_name> -d <backup_directory_name>  
-a <backup_file_name>.zip
```

On z/OS systems:

```
mqsirestoreconfigmgr <ConfigMgr_name> d=<backup_directory_name>  
a=<backup_file_name>.zip
```

The **-d** or **d=** parameters specify the directory where the archive is placed and the **-a** or **a=** parameters specify the name of backup file.

Example 6-2 Restoring the Configuration Manager

```
C:\Program Files\IBM\MQSI\6.0>mqsirestoreconfigmgr CFGMGR -d c:\backup  
-a CFGMGR_060615_140958.zip
```

```
BIP1079I: Replacing repository for 'CFGMGR' with archive  
'c:\backup\CFGMGR_060615_140958.zip'...  
BIP1171I: Verifying the restored repository...  
BIP8071I: Successful command completion.
```

```
C:\Program Files\IBM\MQSI\6.0>
```

5. Start the Configuration Manager using the **mqsistart** command.

Recovering Configuration Manager when backup not available

Attention: After the Configuration Manager is re-created, adding the existing brokers to the Configuration Manager domain and deploying to such brokers can result in UUID mismatch errors. This mismatch is because the Configuration Manager creates new UUIDs for the brokers that are added to its domain, which do not match with the existing brokers UUIDs. To avoid this situation, delete and re-create the broker from the broker machine before adding to the newly created Configuration Manager domain.

If the Configuration Manager repository fails, cannot be corrected using problem determination techniques and the repository backup is not available, perform the following sequence of operations to re-create the Configuration Manager:

1. Stop the Configuration Manager using the **mqsistop** command, if running.
2. Delete the Configuration Manager using the following command.

On Windows and UNIX systems:

```
mqsdeleteconfigmgr <ConfigMgr_name> -w -n
```

Specify the **-w** parameter to delete all files from the Configuration Manager workpath and the **-n** parameter to delete Configuration Manager repository.

On z/OS systems:

```
mqsdeleteconfigmgr <ConfigMgr_name> 1 n=
```

Specify the **1** parameter to delete the Configuration Manager registry and the **n=** parameter to delete the Configuration Manager repository.

3. Re-create the Configuration Manager using the **mqsicreateconfigmgr** command. The **mqsicreateconfigmgr** command creates the fresh

configuration repository automatically (any of the data that was previously in the repository are lost).

4. Start the Configuration Manager using the **mqsistart** command.
5. Redefine the brokers in domain topology and redeploy the configuration.

Recovering after Configuration Manager queue manager fails

Important: If the failed queue manager is shared between a broker, User Name Server, and Configuration Manager, re-create all three components, because each component creates different queues.

If the Configuration Manager queue manager fails and you cannot correct the problem using problem determination techniques, restore the queue manager from its backup and then restart the Configuration Manager. If the queue manager backup is not available, perform the following sequence of operations to re-create the Configuration Manager:

1. Ensure that no deploy is running to any brokers.
2. Stop all brokers that are using the Configuration Manager using the **mqsistop** command.
3. Stop the Configuration Manager using the **mqsistop** command, if running.
4. Delete the Configuration Manager using the following command.

On Windows and UNIX systems:

```
mqsdeleteconfigmgr <ConfigMgr_name> -q
```

Specify the **-q** parameter to delete the Configuration Manager queue manager.

On z/OS systems:

```
mqsdeleteconfigmgr <ConfigMgr_name> 1 2
```

Specify the **1** parameter to delete the Configuration Manager registry and the **2** parameter to delete the Configuration Manager queues.

Do not use the **-n** or **n=** options in order to preserve the Configuration Manager repository. Alternatively, take backup of the Configuration Manager repository before deleting the Configuration Manager issuing the **mqsibackupconfigmgr** command.

5. Re-create the Configuration Manager using the **mqsicreateconfigmgr** command. This command creates the queue manager and default queues automatically.
6. Start the Configuration Manager using the **mqsistart** command.

7. Start the brokers using the **mqsistart** command.
8. Redeploy the domain configuration.

6.3.3 User Name Server

This section describes backing up, restoring, and recovering the User Name Server.

Backing up User Name Server

Perform the backup of User Name Server using the following steps:

1. Choose a safe storage location, such as a tape drive or DVD, to store the backup information.
2. Before taking the backup, ensure that no deploy process is running to any broker and then stop the User Name Server issuing the **mqsistop** command.
3. Backup the registry information for the User Name Server. The registry holds vital information about runtime components UUID values, which are useful while restoring the components from the failures. Use the following commands:

On UNIX:

```
/var/mqsi/registry/UserNameServer
```

On Windows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphereMQIntegrator\2\  
UserNameServer
```

On z/OS:

```
<broker_directory>/registry/UserNameServer
```

4. Along with the registry backup, also take User Name Server service information using following command:

```
mqsiservice UserNameServer
```

For future reference and restore operation, save the output from this command. The Component UUID field denotes the UUID of the User Name Server.

5. The queue manager backup is also important part of the User Name Server component. Refer to WebSphere MQ 6.0 Information Center for the backup instructions:

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/basto.htm>

Recovering after User Name Server fails

If the User Name Server fails and you cannot correct the problem using problem determination techniques, perform the following sequence of operations to re-create the User Name Server:

1. Ensure that no deploy is running to any brokers.
2. Stop the User Name Server using the **mqsistop** command, if running.
3. Stop the User Name Server queue manager using the **endmqm** command.
4. Delete the User Name Server using the following command.

On Windows and UNIX systems:

```
mqsdeleteusenameserver -w
```

Specify the **-w** parameter to delete all files from the User Name Server workpath.

On z/OS systems:

```
mqsdeleteusenameserver c=<ComponentDirectory> 1
```

Specify the **c=** parameter for name of the component directory and the **1** parameter to delete the User Name Server registry.

5. Re-create the User Name Server using the **mqsicreateusenameserver** command.
6. Start the User Name Server using the **mqsistart** command.

Recovering after User Name Server queue manager fails

If the User Name Server queue manager fails and you cannot correct the problem using problem determination techniques, restore the queue manager from its backup and then restart the User Name Server. If the queue manager backup is not available, perform the following sequence of operations to re-create the User Name Server:

1. Ensure that no deploy is running to any brokers.
2. Stop the User Name Server using the **mqsistop** command, if running.
3. Delete the User Name Server using the following command.

On Windows and UNIX systems:

```
mqsdeleteusenameserver -q
```

Specify the **-w** parameter to delete the User Name Server queue manager.

On z/OS systems:

```
mqsdeleteusenameserver c=<ComponentDirectory> 1 2
```

Specify the **c=** parameter for name of the component directory, the **1** parameter to delete the User Name Server registry, and the **1** parameter to delete the User Name Server queues.

4. Re-create the User Name Server using the **mqsicreateusernameserver** command. This command creates the queue manager and default queues automatically.
5. Start the User Name Server using the **mqsistart** command.
6. Redeploy the domain configuration.

6.3.4 Toolkit

This section describes backing up, restoring, and recovering the Message Brokers Toolkit. When the Message Brokers Toolkit is launched, a dialog box prompts the user to select where the Workspace should be located. The Workspace is the directory where user work is stored. Message Brokers Toolkit workspace holds all the message broker resources (message flows, message sets, Plug In Node Projects, configuration for the broker connections, and broker archive files).

Backing up Toolkit resources

The following choices are available to backup the Toolkit resources:

- Copy the entire Workspace directory and save it to a safe location.
You can move the Workspace to another Toolkit environment and re-imported to restore the message broker resources.

- Export the projects from the Toolkit to the zipped files:
 - a. Select all projects in the Message Brokers Toolkit Resource Navigator view as shown in Figure 6-1.

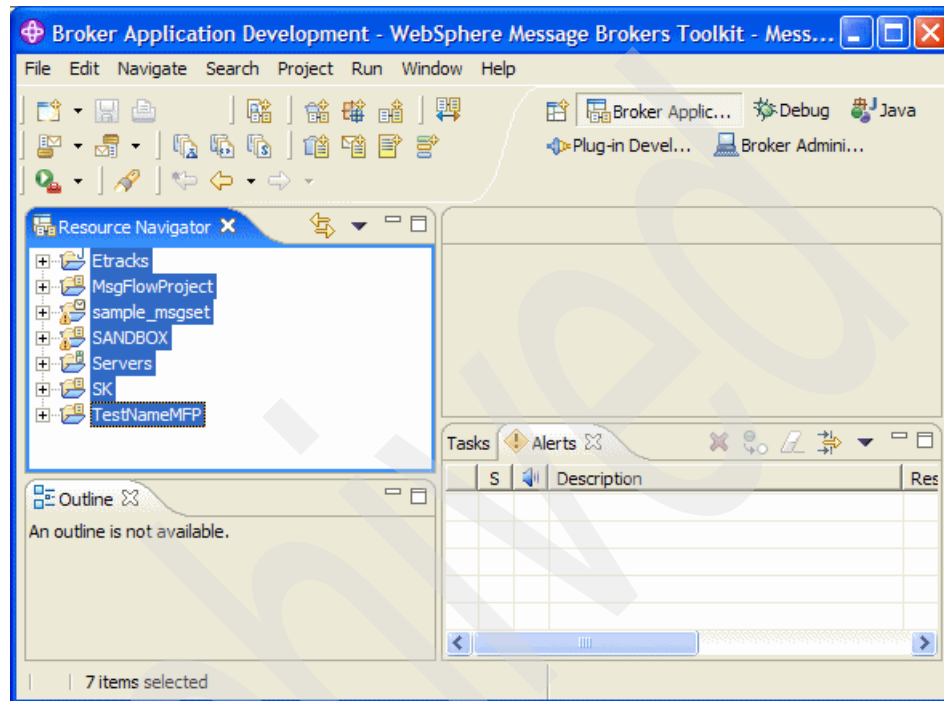


Figure 6-1 Selecting all Projects from Resource Navigator view

- b. Select **File** → **Export** and choose the **Zip file** option. Click **Next**.
 - c. Enter the export destination and file name to **To zip file** entry or use the **Browse** button to select it.
 - d. Ensure that **Create directory structure for files** is selected, as illustrated in Figure 6-2. Then click **Finish**.

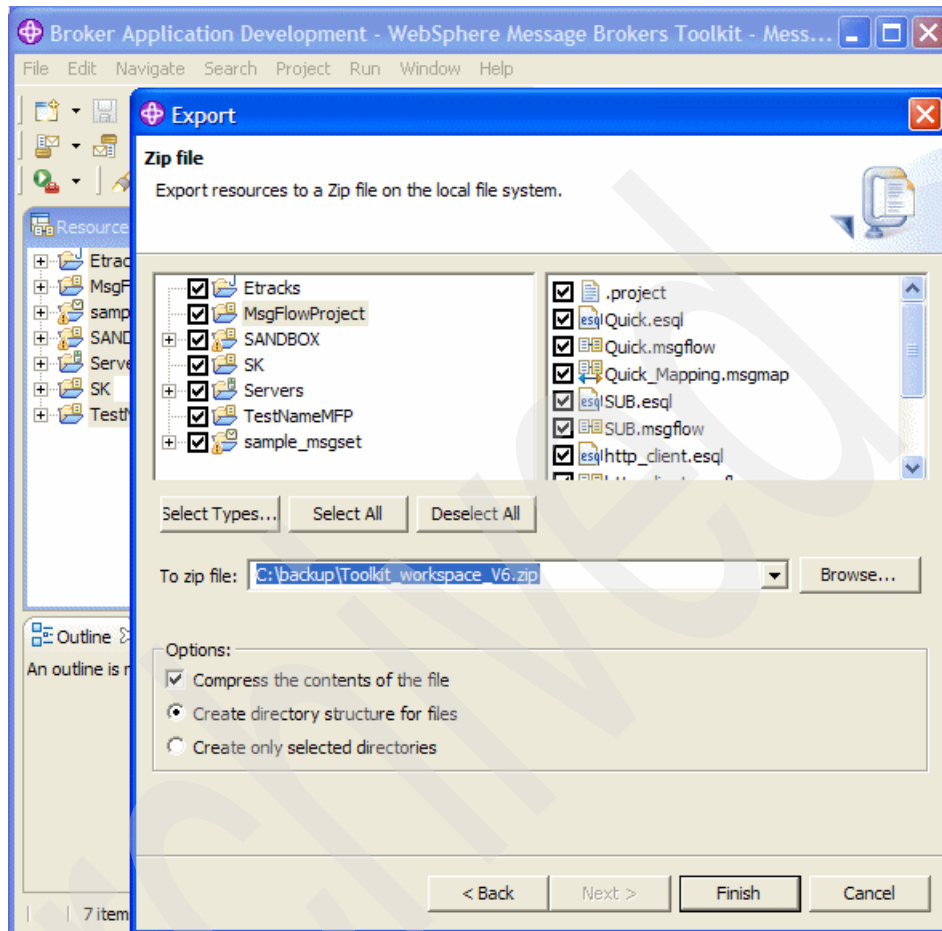


Figure 6-2 Saving the export in a zipped file

Restoring Toolkit resources from backup

Because the Toolkit backup has two choices, the corresponding choices are available for restore:

- ▶ Restore the entire Workspace directory from the backup to the same or another Toolkit location.
- ▶ Import the projects to the Toolkit from the zipped files as follows:
 - a. Extract the appropriate zipped file that contains your project into a temporary directory.
 - b. Select **File** → **Import** and choose the **Existing Project into Workspace** option from the Message Brokers Toolkit menu. Click **Next**.

- c. Enter this temporary directory to **Project contents** entry or use the button **Browse** to select it. Click **Finish**.

Restoring Toolkit resources from Local History

You can restore the deleted resources from the Toolkit workspace projects from the Local History as follows:

1. In the Navigator view, select the folder or project into which you want to restore a local history state. Right-click the folder and from the resource's menu, select **Restore from Local History**, as shown in Figure 6-3.

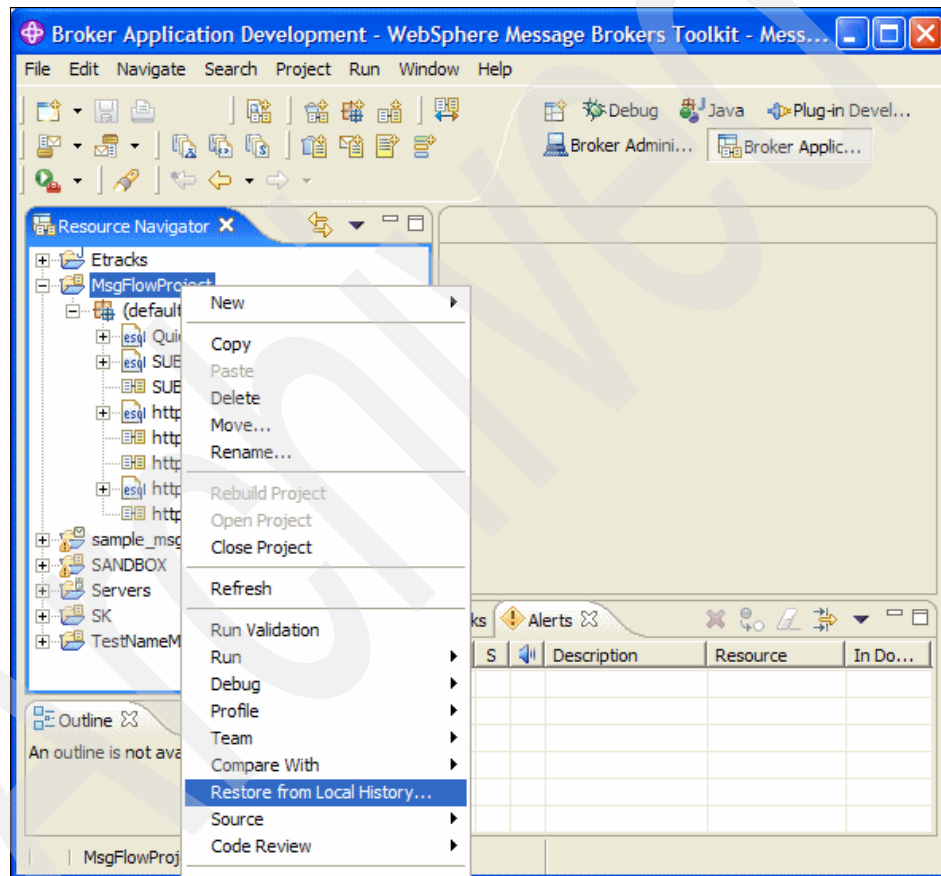


Figure 6-3 Selecting the restore option

2. The **Restore From Local History** dialog opens and shows all files that were previously contained in the selected folder or project and all of their sub-folders. Check the files that you want to restore.

If you do not want to restore just the last state of a file, you can select any other state of the file from the Local History list on the right-hand side of the dialog (Figure 6-4). The bottom pane of the dialog shows the contents of the state.

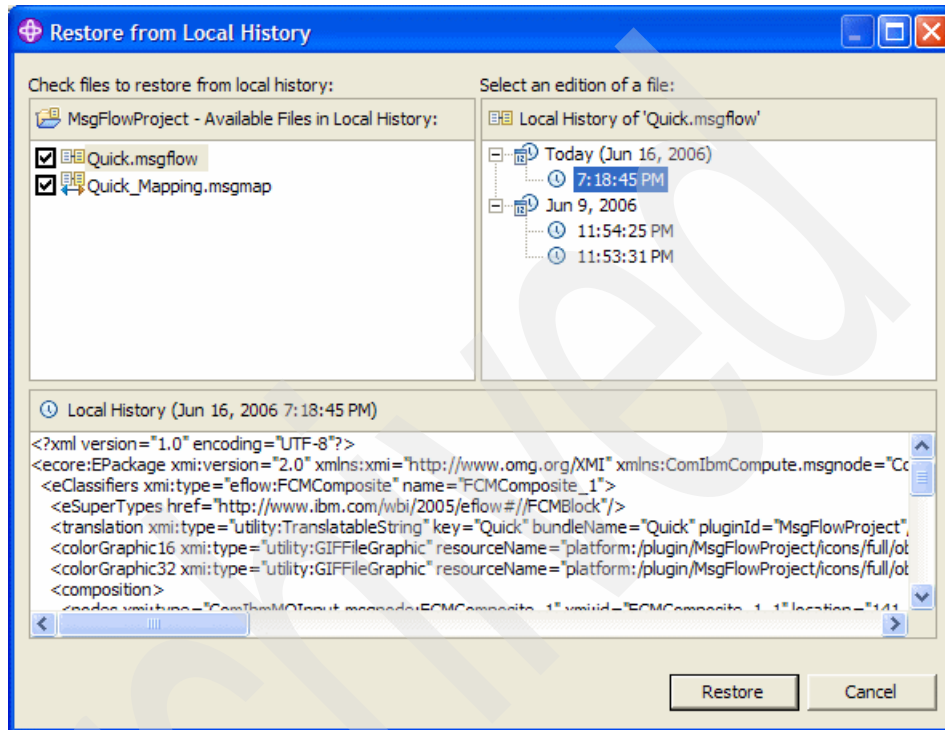


Figure 6-4 Selecting the resources to be restored

Tip: You can configure your Workbench preferences to specify how many days to keep files, how many entries per file you want to keep, or the maximum file size for files to be kept. Select **Window** → **Preferences** → **Workbench** → **Local History**.

3. If you are done with all files, click **Restore**. In Figure 6-5, the restored objects are highlighted.

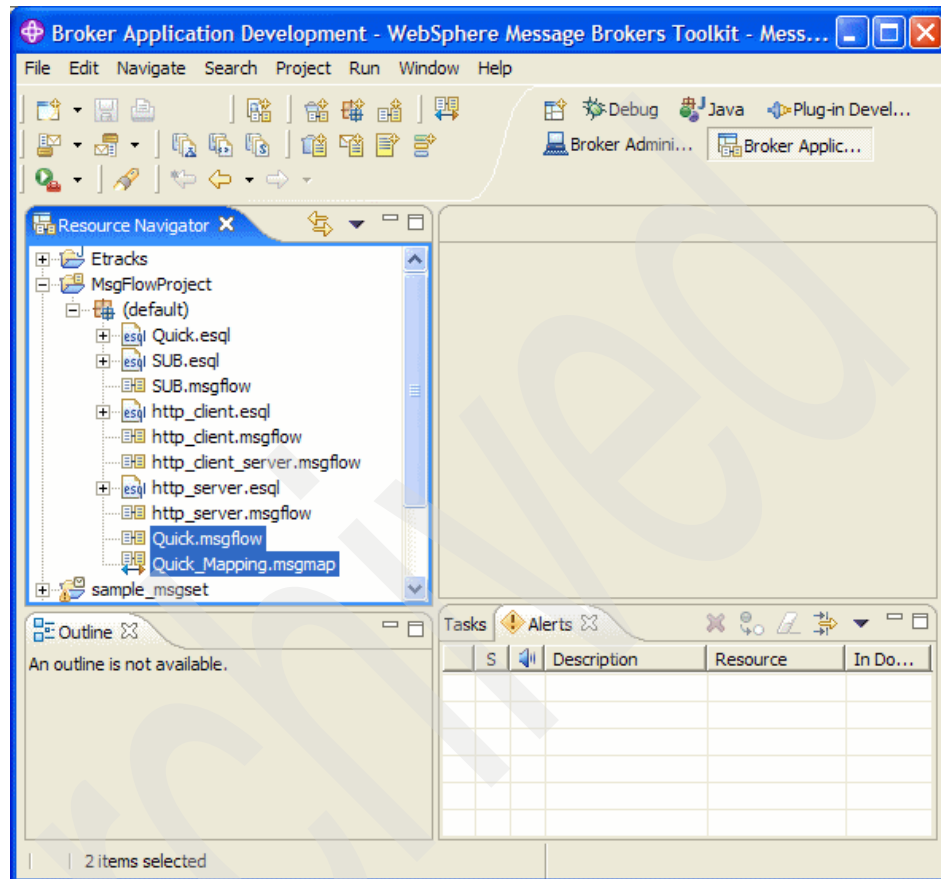


Figure 6-5 Resources restored in the project

Monitor and health-check

Health-checks of message broker components is an important discipline that you must perform to keep the production systems in a predictable condition. Monitoring and health-checks ensure that the production environment meets the appropriate Service Level Agreements (SLA) for availability, reliability, and performance.

In this chapter, we discuss the monitoring and health-checking aspects of WebSphere Message Broker V6.0 components and their importance to the production environment. It includes the following topics:

- ▶ Basic of monitoring and health-checking
- ▶ Monitoring the domain
- ▶ Optimizing resources
- ▶ Characteristics for message broker components

7.1 Basic of monitoring and health-checking

In any production environment, operational staff must ensure the continued monitoring and health checking process in long-term operation processes as part of on-going operations. You should consider the following aspects as a critical part of monitoring and health-checking:

- ▶ Availability
- ▶ Reliability
- ▶ Performance

In addition, you should define a process for the following:

- ▶ Frequency of performing health-check
- ▶ Archiving of monitored data
- ▶ Reporting of data for analysis
- ▶ Altering mechanism

This section discusses in detail the basics of monitoring and health-checking.

Note: The technical articles that we reference in this chapter were written for previous versions of the product. These articles are still valid for WebSphere Message Broker V6.0.

7.1.1 Availability

Availability is the measure of time that a service is functioning normally. It also serves as a measure of the time that the recovery process requires after the service fails. The necessary availability requirements are defined by Service Level Agreements (SLAs) specific to each organization. However, in today's business environment, many organizations demand higher levels of system availability. Therefore, it is important for an organization to have proactive measures to reduce unplanned outages that result in service interruptions. Promoting availability can be accomplished through monitoring and regular health-checking of the message broker and the dependent resources. For an additional review of availability as it relates to message broker, see 4.4.2, "High Availability" on page 54.

7.1.2 Reliability

WebSphere Message Broker is a business integration product that coordinates and integrates the business processes and applications. It brings together the data and process intelligence in the enterprise and harnesses these assets so that the applications and the users can achieve their business goals. The

reliability of the message broker components is critical to gauge and to respond to faults that occur in production efficiently. To ensure that all the message broker runtime components remain reliable, you should perform a health-check process on routine basis. For an additional review of reliability as it relates to message broker, see 4.4.1, “Reliability” on page 53.

7.1.3 Performance

You can measure the performance of a service in terms of throughput and latency. Effective monitoring and health-checking helps you tune performance with the overall goal of reducing transaction costs and reaching performance SLAs. You accomplish this goal by measuring throughput and latency against a baseline when the message broker is configured accurately for the specific operating environment. Continuous monitoring and routine health-checks identifies bottlenecks in the processing that are outside of the established baseline. Using the metrics that you capture, you can make accurate changes to increase throughput and reduce latency. For an additional review of performance as it relates to scalability, see 4.4.3, “Scalability” on page 58.

Important: The default settings for the message broker or its dependent resources are not optimized for specific configurations. Instead, the default values are set to allow commonality between all environments. For tuning recommendations under specific scenarios, see “Runtime topologies” on page 61.

7.1.4 Frequency

You should monitor message broker automatically using a set of procedures that execute without an operator being present and that send alerts automatically. Thus, the frequency of monitoring as we describe in this book is continuous.

It is recommended that you perform a manual health-check on a routine basis where the frequency is determined by the specific environment, culture, and business needs of your organization. For example, if you add new components or change existing components, you should perform health-check procedures more frequently. If you make few changes, the time between health-checks is longer.

7.1.5 Archiving

You can archive monitoring data for the historical analysis and then use that analysis for capacity planning activities. Data areas that are useful for capacity planning are:

- ▶ Is the message broker increasingly consuming more resources (such as CPU, storage, Input/Output) continuously or only during various processing periods?
- ▶ Does message broker generate error types at regular intervals or only during certain processing periods?
- ▶ How does the broker perform during the peak hours over the period of time?

The trend analysis from the history data can be useful when deciding for further enhancements to resources (software and hardware).

7.1.6 Reporting

You can use the performance and accounting information that you collect for the brokers for the following purposes:

- ▶ To enable efficient and effective tuning of message flows design.
- ▶ To enable infrastructure teams to charge users accurately for their use of message broker facilities (charge-back).
- ▶ To create performance and unplanned service interruptions reports for SLA reports for management review.

WebSphere Message Broker V6.0 lets operational staff gather statistics data down to the message flow level. Developers of message flows do not need to create the designs to capture performance metrics. After the administrator enables statistics collection, the broker measures a broad range of metrics, including CPU and I/O consumption. It also measures metrics such as number of messages processed, maximum message size, and average elapsed time.

Using the message broker metrics in conjunction with other statistical data such as those available to WebSphere MQ, databases, or operating systems provides a comprehensive sampling of data for analysis by capacity planning teams.

7.1.7 Alerting

Sending alerts automatically is a crucial aspect of a monitoring solution for WebSphere Message Broker. The alerts can be in the form of an e-mail, a page, or a message to a z/OS console operator. A simple monitoring tool provides alerts directly to the administrator. A comprehensive solution allows aggregation

of events from multiple monitors and sends alerts for open or close status based on rules to one or more component owners. Some resources provide alerting functions internal to the product, such as DB2. For more information about this functionality, see 7.2.1, “Monitoring systems for availability and failure events” on page 139.

7.2 Monitoring the domain

After the deployment process is complete, ensure that the operations staff continues the monitoring and health checking process in their long-term operations processes as part of the on-going operations.

7.2.1 Monitoring systems for availability and failure events

You detect availability and failure events by reviewing the error logs that all of the WebSphere Message Broker components and the underlying resources, such as WebSphere MQ and databases, generate.

The following list of error reporting sources can be useful as input to the health-check process:

- ▶ System logs:
 - System Event log on Windows
 - Syslog on UNIX systems
 - System console and started task logs on z/OS
- ▶ WebSphere Message Broker domain logs:
 - Eclipse logs
 - Message Brokers Toolkit Event log
 - Message Brokers Toolkit Task views
 - message broker abends files
- ▶ WebSphere MQ error logs:
 - *AMQERRXX.log* files
 - First Failure Symptom logs (FDCs)
- ▶ Database log and configurations, for example DB2 provides:
 - db2diag.log
 - database manager configuration (DBMCFG)
 - database configuration (DBCFCG)
- ▶ You must monitor the databases in the event that a fault is detected. IBM DB2 provides monitoring and health-check tools that you can configure for alert settings and notification based on the event.

The DB2 Health Center assists database administrators by alerting them to potential problems and by providing recommendations to resolve problems. Database administrators can monitor an instance remotely using the Web Health Center and can view alert details and recommendations. The health monitoring process generates e-mail notifications of alerts, warnings, or both. For more information about the Health Center, refer to standard DB2 product documentation, which is available at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/welcome.htm>

We provide specific details on interpreting messages in 9.1, “Basic steps” on page 164. You can find information about each error that reports to the various components in 9.3, “Characteristics for message broker components” on page 211.

7.2.2 Monitoring systems for performance and reliability

Monitoring database performance is as important as monitoring WebSphere Message Broker components performance because the brokers performance (and also performance of message flows that perform database operations) rely on database performance. Resources that are available for use to monitor database, operating system, and broker performance include:

- ▶ There is a facility in message broker statistics and accounting that provides broker runtime performance in terms of CPU usage, message throughput, cost per message, and so forth. For more information about this tool, see 7.3.1, “Tools” on page 145.
- ▶ IBM DB2 provides tools for monitoring the performance of database components, such as Activity Monitor. Activity Monitor helps monitor application performance, application concurrency, resource consumption, and SQL statement usage. Activity Monitor can assist in diagnosing database performance issues such as lock waiting situations and in tuning for optimal utilization of the database resources. For a database engine other than DB2, refer to the corresponding documentation for the monitoring tasks.
- ▶ You must monitor operating system resources such as file system space, storage I/O, CPU utilization, physical memory, and paging space use that use operating system utilities or commands. For example, on AIX, utilities such as **topas**, **vmstat**, and **svmon** provide statistics for various operating system parameters.

7.2.3 Basic parameters

This section discusses some of the basic parameters — from the broker domain perspective — that you must include in your health-checking and monitoring.

Queue manager

WebSphere MQ provides several different approaches for monitoring queue managers and their ability to intercommunicate. You can find the product documentation about this subject in the *Monitoring WebSphere MQ*, which is available at:

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.csqzax.doc/csqzax05.htm>

There are several functions categorized by the groupings:

- ▶ Event monitoring, which is the process of detecting occurrences of instrumentation events in a queue manager network.
- ▶ Message monitoring, which is the process of identifying the route a message has taken through a queue manager network.
- ▶ Accounting and statistics monitoring, which is generated intermittently by queue managers to record information about the operations that are performed by applications or to record information about the activities occurring with the system.
- ▶ Real-time monitoring, which is a technique that allows determination of the state of queues, channels, listeners, and services within a queue manager.

In addition to the embedded monitoring features of WebSphere MQ, you need to monitor a few additional physical resources. For example, you need to monitor the available physical storage (disk) that the queue manager uses. This task is especially important with the use of persistent messages because the complete message payloads are stored physically on the disk. In addition, on Windows and UNIX systems, it is possible to choose between circular or linear logging. For more information about calculating the size of log and the amount of available physical storage required, refer to *WebSphere MQ System Administration Guide*, which is available at:

<http://www.ibm.com/software/integration/wmq/library/library6x.html>

For logging options on z/OS, refer to the standard WebSphere MQ documentation for z/OS, which is available at:

<http://www.ibm.com/software/integration/wmq/library/library6x.html>

Database

You should monitor the file system where databases are created for sufficient space because as they can grow as the data gets populated. You must also monitor the parameters that are relevant to database performance. For example, in some cases you need to tune DB2 to query larger amounts of data (as in the broker resources table). The DB2 parameters that you should consider for monitoring are:

- ▶ **APPLHEAPSZ:** To increase the maximum amount of memory pages that can be allocated for the application heap.
- ▶ **APP_CTL_HEAP_SZ:** To increase the maximum amount of memory pages that can be allocated for the application control heap.
- ▶ **DBHEAP:** There is one database heap per database, and the database manager uses it on behalf of all applications that are connected to the database. This database heap contains control block information for tables, indexes, table spaces, and buffer pools.

For a database engine other than DB2, refer to the corresponding documentation for the performance monitoring tasks.

Broker

You can monitor the following parameters on regular basis:

- ▶ **Broker processes.**

The following processes are spawned when the broker is started:

- `bipservice`
- `bipbroker`
- `biphttplistener`
- `DataFlowEngine`

The `bipservice` process is a very small process that starts and then starts and monitors the `bipbroker` process. The `bipbroker` process is designed to monitor and administrate the execution groups. It performs such functions as handling deploy requests and communicating with the Configuration Manager. It does not process any message flows, but its size is governed by the size of the deploy requests that are received.

The `biphttplistener` process is only used for Web service functionality (when the `HTTPInput`, `HTTPRequest`, and `HTTPReply` nodes are in use). Therefore, if the deployed message flows do not contain such resources, then `biphttplistener` does not consume any significant amount of resources.

- ▶ A WebSphere Message Broker execution group is implemented as one operating system process. The name of the executable for this process starts

with the DataFlowEngine keyword following the broker name, execution group UUID, and execution group name.

Ensure that this process is not reaching its limit, which is defined by the operation system. For example, the execution group process on AIX (DataFlowEngine) allocates four memory segments by default. This default allocation makes the DataFlowEngine process grow up to 1 GB in size. If you observe that the DataFlowEngine process is nearing this mark, then the administrator can increase the number of segments for the DataFlowEngine.

However, before doing so, the administrator should check whether the increased requirement for memory is not a result of one of the following:

- The large deployment configuration and large message processing
- A memory leak
- Inefficient message flow design

You can monitor the DataFlowEngine process by writing a script to execute a process listing. Example 7-1 shows the process listing output on AIX.

Example 7-1 Process listing of default execution group

```
$ ps -elf | head -1 ; ps -elf | grep -v "grep" | grep DataFlowEngine
  F S UID  PID  PPID  C PRI NI ADDR  SZ  WCHAN  STIME  TTY  TIME CMD
242001 A wmb 39482 38198  0  60 20 1c0b9c 50180      * 17:33:48      -  0:11
DataFlowEngine BROKER2 f2b328ee-0b01-0000-0080-9eecf1db91dc default 1
```

In Example 7-1, the highlighted fields are:

- UID: Process ID of DataFlowEngine process.
- SZ: The size in pages of the core image of the process.
- CMD: Full name of DataFlowEngine process.

For more information, refer to the article *How to monitor system and WebSphere Business Integration Message Broker V5 resource use*, which is available at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0407_dunn/0407_dunn.html

For message flow tuning considerations, refer to 7.4, “Characteristics for message broker components” on page 146 and “Broker” on page 142.

Monitor the syslog for any deploy timeout errors or broker startup errors. For example, if you see the BIP2080E or BIP2066E errors, tune the timeout parameters for broker (such as configurationTimeout and configurationDelayTimeout) using the **mqs i changebroker** command. Also, check whether the system is running low on resources during a deploy or if there are any long running message flows that are not responding to the deploy request. These issues are just some of the pointers towards

approaching the cause of the issue discussed here. In practice, there might be other situations where these errors can occur.

- ▶ If you use aggregation flows, monitor messages and queue depths on `SYSTEM.BROKER.AGGR.*` queues. WebSphere Message Broker now internally uses `SYSTEM.BROKER.AGGR.*` queues to store the request, response, control, and timeout messages for each aggregation group.
 - ▶ You should monitor the stdout and stderr messages. WebSphere Message Broker captures stdout and stderr to a file so that you can view this information later. Components such as Java write useful information to stdout and stderr that is not captured within the service trace or other message broker files. The default location for the files are:
 - `<WorkPath>/components/<broker_Name>/<EG_UUID>/stdout`
 - `<WorkPath>/components/<broker_Name>/<EG_UUID>/stderr`
- You must monitor these files to ensure that there are no error messages.

7.3 Optimizing resources

It is important that you understand the needs of the message flows that are developed so that they adequately achieve the SLA. For example, you need to determine if they are CPU bound or I/O bound. If a CPU bound message flow is starved for CPU, the user might not get the required throughput. Similarly, insufficient I/O configuration for message flows can result in significant processing delays by logging activities. Thus, knowing whether a message flow is CPU bound or I/O bound is key to knowing how to increase message throughput. It is not profitable to add processors to a system that is I/O bound. Instead, you should use a storage system that is faster, such as a solid state disk. For other scalability considerations, refer to 4.4.3, “Scalability” on page 58.

In many message flows, the parsing and manipulation of messages is common and is known as CPU-intensive activity, because such message throughput is sensitive to processor speed. Brokers benefit from the use of faster processors. Thus, as a general rule, it is better to allocate fewer faster processors than many slower processors.

It is also important to ensure that software levels are as current as possible. The latest levels of WebSphere Message Broker and WebSphere MQ software offer the best levels of performance.

7.3.1 Tools

In this section, we discuss on some of the tools and features of the product that you can use for monitoring and health-checking tasks.

IBM Tivoli OMEGAMON

IBM Tivoli OMEGAMON XE for WebSphere Business Integration is a product in Tivoli portfolio that manages WebSphere MQ and WebSphere Message Broker environments from a single console. IBM Tivoli OMEGAMON XE for WebSphere Business Integration supports distributed and mainframe systems and provides an end-to-end view across all systems. It analyzes application performance and identifies bottlenecks and monitors message rates, brokers, message flows, and sub-flows. You can also use IBM Tivoli OMEGAMON XE for WebSphere Business Integration for administration tasks. For more information, refer to *Implementing IBM Tivoli OMEGAMON XE for WebSphere Business Integration V1.1*, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg246768.html?Open>

Using WebSphere Message Broker Accounting and Statistics

There are various tasks that you can use when monitoring WebSphere Message Broker:

- ▶ Monitoring long-term use of resources to ensure that there are enough CPU resources available to meet the current and the future needs.
- ▶ Monitoring business application throughput to ensure that expected performance statistics are met. Typical information includes:
 - Number of messages that are processed per day or week
 - Number of messages that are processed during peak hour in a week
 - Amount of CPU that is used per week
 - Peak hourly CPU and memory that are used during a week
 - Overall cost per message
- ▶ Comparing the cost per message of the current production message flows to the improved flows currently developed and evaluating whether the change makes a significant difference to the production systems.
- ▶ From the throughput of one execution group or instance, calculating how many execution groups or instances are required to meet your throughput requirements.
- ▶ Identifying message flows and nodes within flows where there are significant delays or bottlenecks. For example, performing a DB2 update produces I/O load to the log data sets.

- ▶ Identifying message flows and nodes within flows that have a relatively high cost per message. When looking to reduce the CPU that is used by a broker, start with the nodes that use a lot of CPU (or similar for system memory).
- ▶ Identifying flow through error paths (for example, if message flow through error processing nodes, then determine why this is occurring).

WebSphere Message Broker provides snapshot and archive data. Snapshot is for online monitoring of the broker; archive is for long term monitoring.

For further information about how to analyze and interpret the WebSphere Message Broker statistics and accounting data, refer to SupportPac *IS11: WebSphere Business Integration using Statistics and accounting with V5 Broker*, which is available at:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24007228&loc=en_US&cs=utf-8&lang=en

Using test programs

You can run test programs on your systems to determine quickly if there are any major issues with the setup of the operating systems, WebSphere MQ, or WebSphere Message Broker. The following SupportPac provides a sniff test:

WebSphere Business Integration Broker - Sniff test and Performance on z/OS, IP13

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24006892&loc=en_US&cs=utf-8&lang=en

This SupportPac is designed specifically for z/OS. You can develop comparable tests for any WebSphere Message Broker platform.

7.4 Characteristics for message broker components

In this section, we discuss specific details for message broker components in health-checking and monitoring areas.

7.4.1 Broker

You can perform the basic health-checking of the broker environment by ensuring that:

- ▶ All broker processes are running properly.
- ▶ Broker queue manager is running properly.
- ▶ There are no undesired error messages in syslog or event log.

- ▶ There are no error messages in WebSphere Message Broker stdout and stderr files.
- ▶ The database is running and there are no database related errors in either database log or the broker log.

Problems with the performance of a WebSphere Message Broker message flow usually take one of two forms. The first is that the required message throughput cannot be achieved. The second is that, although the required level of message throughput is being achieved, the CPU cost is far greater than expected. When the expected level of performance from a message flow is not met, it is usually because of one of the following reasons:

- ▶ A lack of resources with which to run the message flow.
- ▶ A poorly configured environment.
- ▶ Poor response time from a dependent application.
- ▶ An inefficient processing algorithm for the message flow.
- ▶ Inefficient or excessive ESQL processing within the message flow.

Broker tuning considerations

The configuration of the message broker and the manner in which message flows are assigned can have a noticeable effect on message throughput. You can perform the following tuning options in a production environment:

- ▶ Trusted broker

On supported platforms, with the exception of z/OS, you can run the broker as a *trusted application*. When the broker runs as a trusted application, WebSphere MQ API calls are processed more efficiently. Trusted applications are a WebSphere MQ concept. By default, the broker does not run as a trusted application.
- ▶ Multiple instances

If there is a requirement to run multiple copies of message flow within a broker, use additional instances of the message flow within an execution group.
- ▶ Multiple execution groups

It is possible to assign a message flow to one or more execution groups. An execution group provides separation at the process level on the Windows and UNIX systems and at the address space level on z/OS. Each additional execution group has the potential to use an additional processor.
- ▶ Trace

Because the overhead to run trace is significant, it is important to ensure that all tracing is turned off when not required. The primary trace features are the

product trace (user and service) and ODBC trace. For complete examples on using traces, see Chapter 9, “Problem determination” on page 163.

To ensure that all tracing within message broker is switched off, use the **mqsi changetrace** command with a level of none. Remember to do this for all execution groups.

To verify that traces are turned off, use **mqsi reporttrace** command for all execution groups (Example 7-2).

Example 7-2 Verifying user trace and service tracing level

```
$ mqsi reporttrace BROKER2 -t -e default
```

```
BIP8098I: Trace level: none, mode: safe, size: 4096 KB.
```

```
BIP8071I: Successful command completion.
```

```
$
```

```
$ mqsi reporttrace BROKER2 -u -e default
```

```
BIP8098I: Trace level: none, mode: safe, size: 4096 KB.
```

```
BIP8071I: Successful command completion.
```

You can write automated scripts to query the tracing options for all the brokers, execution groups, message flows, and the level of tracing (user level and service level). Also, you should ensure that there are no trace nodes in the processing paths of your message flows.

ODBC trace is turned off using the tracing tab of the ODBC function as follows:

- On Windows, it is available in the Control Panel.
- On UNIX, it is controlled through the Trace keyword in the `odbc.ini` file. Possible values are 0 (off) and 1 (on), as shown in Example 7-3.
- On z/OS, it is controlled through the `APPLTRACE` keyword under the stanza `[COMMON]` in the `BIPDSNAO` file. Possible values are 0 (off) and 1 (on).

Example 7-3 ODBC stanza on UNIX system

```
[ODBC]
Trace=0
TraceFile=/traces/odbctrace.out
TraceDll=/opt/IBM/mqsi/6.0/merant/lib/odbctrac.so
InstallDir=/opt/IBM/mqsi/6.0/merant
UseCursorLib=0
IANAAppCodePage=4
```

You should check for other product or system-related traces, such as DB2 trace or operating system trace.

For more information, refer to the article *Tuning a WebSphere MQ Integrator Broker*, which is available at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0301_dunn/dunn.html

Database tuning considerations

The extent to which you use the broker database depends upon which facilities you use in the message flows. For example, using of publication nodes can lead into increasing the broker database use. In this case, it is necessary to ensure that the broker database is well tuned, especially to handle insert or delete activity.

The extent to which you use the business database is entirely dependent on the business requirements of the message flow. With simple routing flows, it is unlikely that there is any database access. With complex transformations, there might be involved database processing that involve a mixture of read, update, insert, and delete activity. So, it is important to ensure that the database engine is tuned for required application performance.

Queue manager tuning considerations

The WebSphere Message Broker is dependent on the performance of its associated queue manager for the efficiency of the MQ API requests, which are issued within a message flow. There are a number of factors that determine the performance of the queue manager. You can effect these factors by tuning the WebSphere MQ components: channels, listeners, and queue manager logging.

In practice, the broker queue manager is likely to be connected to other queue managers. So, it is important to examine the configuration of these other queue managers as well to ensure that they are well tuned.

For more information about tuning considerations for the broker queue manager, refer to the following resources:

- ▶ *WebSphere BI Message Broker: Designing for Performance*, IP04:
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24006518&loc=en_US&cs=utf-8&lang=en
- ▶ *Improve the performance of your WebSphere Business Integration Message Broker V5 message flow*
http://www.ibm.com/developerworks/websphere/library/techarticles/0406_dunn/0406_dunn.html

- *Tuning a WebSphere MQ Integrator Broker*

http://www.ibm.com/developerworks/websphere/library/techarticles/0301_dunn/dunn.html

7.4.2 Configuration Manager

The Configuration Manager is the interface between the Message Brokers Toolkit and the brokers in the broker domain. Thus, you should ensure that the Configuration Manager and the queue manager processes are running properly when a deploy operation is planned. Also, you should ensure that the channels between the Configuration Manager queue manager and the broker queue manager are up and running. Otherwise, the deploy message might not reach the broker.

The Configuration Manager stores configuration details for the broker domain in an internal repository. The internal repository is stored in the file system. So, you should make sure that the file system that holds the repository information has sufficient space. You can find these repositories at following locations:

- On UNIX: `/var/mqsi/components/<configmgr>`
- On Windows: `C:\Documents and Settings\All Users\Application Data\ibm\MQSI\common\dbinstmgr\{DATABASE_NAME}`
- On z/OS: `++componentdirectory++/components/<configmgrName>`

You should also check the Event log and syslog to ensure that there are no errors reported for the Configuration Manager.

The Configuration Manager requires a queue manager to communicate with other components in broker domain. Thus, ensure that:

- The Configuration Manager queue manager is running by using the **dspmq** command.
- The channels between the Configuration Manager queue manager and other domain components, brokers, and User Name Server are running.

7.4.3 User Name Server

You only need a User Name Server component where publish/subscribe broker applications are running and where extra security is required for applications to publish or subscribe topics. Thus, when you use topic-based security in a broker domain, make sure that the User Name Server process `bipuns` `UserNameServer` is running.

The User Name Server requires a queue manager to communicate with other components in broker domain. So, you need to ensure that:

- ▶ The User Name Server queue manager is running using the **dspmq** command.
- ▶ The channels between the User Name Server queue manager and other domain components, brokers, and Configuration Manager are running.

7.4.4 Toolkit

In the Message Brokers Toolkit, you should review the domain Event log after each deployment to the broker domain to ensure that the deployment was successful. You should also review the Eclipse log if problems arise while using the Message Brokers Toolkit.

Maintenance strategy

Well executed maintenance tasks and a proper maintenance strategy for WebSphere Message Broker are important keys to running message broker safely and in high performance in the production environment. In this chapter, we discuss the main considerations and aspects about a maintenance strategy for WebSphere Message Broker.

This chapter includes the following topics:

- ▶ Maintaining the domain updates
- ▶ Characteristics for message broker components

8.1 Maintaining the domain updates

Keeping software components up-to-date is an important task of administering the broker domain. So, you need plan for and maintain updates for WebSphere Message Broker and its components, which include Rational Agent Controller, WebSphere MQ, and databases.

Good planning is a critical part of properly and safely maintaining updates. A number of decisions and steps, and their correct order, are required to update the message broker successfully. For example, you need to:

- ▶ Decide which fix pack levels you need to install
- ▶ Decide when and in which order to update message broker components
- ▶ Check dependencies among message broker components and product levels by:
 - Reviewing product and fix pack readme files and installation documentation
 - Determining whether the updates of Rational Agent Controller, WebSphere MQ, and the database are required
- ▶ Carefully plan all steps that are needed for a successful update

In this section, we discuss the dependencies among the different product levels of message broker components, the differences between fix pack and version levels of the message broker, and the key stages for performing updates.

8.1.1 Fix pack levels

WebSphere Message Broker V6.0 products requires three types of updates:

- ▶ *Refresh packs*, which includes minor new features and fixes, such as 6.0.1.0
- ▶ *Fix packs*, which includes fixes only, such as 6.0.0.1
- ▶ *Fixes*, which is a single published emergency fix, such as SA79582

We recommend that you install refresh packs and fix packs as proactive maintenance during the scheduled maintenance window, which is described in 8.1.3, “Staging updates” on page 156.

We do not recommend that you install single emergency fixes as part of a general installation. These types of fixes are only intended to solve a particular problem in the effected environment.

Important: Normally, emergency fixes are created to solve a specific issue or issues for a particular environment. You should always consult the emergency fix readme file prior to installing the fix in a production environment.

Note: The details about maintenance strategy for WebSphere Message Broker and WebSphere MQ are described on the Web at:

<http://www.ibm.com/support/docview.wss?rs=171&uid=swg21223595>

In the context of the message broker domain run-time and development environments, there are different requirements between fix pack levels of components. The following rules are recommended:

1. It is strongly recommended to have all run-time components (brokers, the Configuration Manager, and the User Name Server) at the same fix pack level.
2. It is generally recommended to have Message Brokers Toolkit at the same fix pack level as the run-time components.

The InstallShield Multi-Platforms installation program is provided for all fix packs and refresh packs as part of the installation package.

8.1.2 Version levels

WebSphere Message Broker V6.0 can coexist with previous versions of the product, such as WebSphere MQ Integrator V2.1 or WebSphere Business Integration Message Broker V5.0. There are two types of coexistence:

- ▶ Coexistence with previous versions of components in broker domain V6.0
- ▶ Coexistence with previous versions of the product that are installed on the same computer as the V6.0 product

There are restrictions and recommendations for both type of coexistence. So, for more information, refer to the WebSphere Message Broker Information Center, under the topic **Migrating** → **Coexistence**, at:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp?topic=/com.ibm.etools.mft.doc/ah23930_.htm

Although the previous versions of components have different fix pack numbering, you can use the same general update policy as we have described in this chapter for V6.0 updates.

Important: Carefully check the readme files and product documentation about coexistence for information about which fix pack level for a previous version of the component is compactible with which fix pack of the V6.0 component.

8.1.3 Staging updates

The main maintenance strategy question is when and how often to perform updates. Because each environment is unique, there is no single answer. You usually have a general maintenance strategy for the entire IT environment. However, the basic recommendations for updates are as follows:

- ▶ Install new fix packs or refresh packs as soon as they are available, and start your installation process immediately (because the update procedure can take time). See 8.1.4, “Update procedure” on page 158 for information about the recommended steps that you need to take.
- ▶ If your environment does not allow the frequent updates (fix packs for WebSphere Message Broker and WebSphere MQ should be announced once every three months), the maximum update period for WebSphere Message Broker and WebSphere MQ should be every six months.
- ▶ You do not need to perform database updates regularly from a WebSphere Message Broker point of view. Install updates in case of a problem solution or refer to maintenance strategy for specific database.
- ▶ It is not recommended that you perform WebSphere Message Broker, WebSphere MQ, or database updates at the same time. Use the separate maintenance cycle and window for each software product.

Note: Exceptions to these recommendations include the following:

- Install an update out of regular order if new functionality included in the update is required.
- Install an update for more than one software product at the same time if the fix pack for one product is dependent upon the fix pack for another (for example, the broker fix pack requires a specific fix pack for WebSphere MQ). In this case, more accurate testing is recommended.

There is another update question that you should think over carefully: should you update all WebSphere Message Broker components at the same time or update the components sequentially, step-by-step? The method that you choose depends upon your environment and business requirements.

If you update all the WebSphere Message Broker components at the same time, the typical steps are as follows:

1. Ensure that no deploy is in process.
2. Stop all broker domain components.
3. Update all broker domain components.
4. Start all broker domain components.
5. Check all available logs.
6. Check functionality of whole broker domain.

If you update all the WebSphere Message Broker components sequentially, step-by-step, the recommended steps are as follows:

1. Ensure that no deploy is in process.
2. Perform the update process of the Configuration Manager:
 - a. Stop the Configuration Manager.
 - b. Update the Configuration Manager.
 - c. Start the Configuration Manager.
 - d. Check all available logs and the Configuration Manager functionality.
3. Perform the update process of the User Name Server:
 - a. Stop the User Name Server.
 - b. Update the User Name Server.
 - c. Start the User Name Server.
 - d. Check all available logs and the User Name Server functionality.
4. Perform the update process for each broker:
 - a. Stop the broker.
 - b. Update the broker.
 - c. Start the broker.
 - d. Check all available logs and the broker functionality.
5. Perform the update process of the Toolkit:
 - a. Stop the Toolkit.
 - b. Update the Toolkit.
 - c. Start the Toolkit.
 - d. Check all available logs and the Toolkit functionality.
6. Check the functionality of entire broker domain.

Table 8-1 discusses the advantages and disadvantages to both types of updates.

Table 8-1 Advantages (+) and disadvantages (-) for update alternatives of domain components

Updating all WebSphere Message Broker components at the same time	Updating WebSphere Message Broker components step-by-step
+ Easier to manage.	- The correct order of steps are required.
+ Whole maintenance window is shorter.	- Entire maintenance window is longer.
+ Components start and run at the same level.	- The broker runs for short time at a different level than other run-time components, which is potentially a problem.
- The broker is not available for the entire maintenance window.	+ The broker downtime is shorter.
- Possible errors are discovered after all you have updated the components; problem determination can be more difficult.	+ Possible errors are discovered immediately after a particular component update, which makes problem determination easier.

8.1.4 Update procedure

The proper update procedure must contain the software update and a backup of broker domain components. It must also check the product documentation and fix pack readme files. The same procedure must run in the development production first and then in the test and production environment.

The recommended steps for an entire update cycle are as follows:

1. Read the fix pack readme file immediately when it is released. Check the file carefully for new functionality, limitations, requirements, and product dependencies for each of the following:
 - WebSphere Message Broker components
 - Rational Agent Controller
 - WebSphere MQ
 - Databases
2. Make initial planning in the following areas:
 - Choose the type of components update (at the same time or step-by-step)
 - Check if an update of underlying components (Rational Agent Controller, WebSphere MQ, or databases) is required
 - Document detailed updating steps
 - Plan for a backup of broker domain components and their underlying resources (Rational Agent Controller, WebSphere MQ, databases, and operating systems)

Refer to Chapter 6, “Backup, restore, and recover” on page 115 for further backup information.

- Document detailed steps for recreating your environment to the previous version in case of any errors or problems
 - Estimate the required time for the update
 - Download or order any required software packages
3. Start the *freezing* period, that is the time after which no changes are allowed in the broker domain, message flows, and message sets.
 4. Perform the update process in the development environment. Use your documented updating steps and note any differences, errors, and real-time duration. In general, follow these steps:
 - a. Plan the maintenance window.
 - b. Back up the broker domain components and their underlying resources.
 - c. Update underlying components, if required.
 - d. Update the broker domain components.
 - e. Perform functional tests of message flows and broker domain.
 - f. Back up the broker domain components and their underlying resources.
 - g. Prepare new version of the updating steps according to information from this update to use as a base for the test environment update.

Important: Making notes during the update process of the development environment and then using these notes to review the process can be very helpful for the production environment update. It can minimize troubles and short cut outages.

5. Perform the update process in the test environment. Use the reviewed updating steps from and note any differences, errors, and real time duration. In general, follow these steps:
 - a. Review the time duration and plan the maintenance window.
 - b. Back up the broker domain components and their underlying resources.
 - c. Update underlying components, if required.
 - d. Update the broker domain components.
 - e. Perform complex tests of the entire broker domain (stress, acceptance, and other non-functional tests).

- f. (Optional) Restore the broker domain components to the previous level to ensure that it is working and that the steps are correct. Then, write the actual time of duration.
- g. Back up the broker domain components and their underlying resources.
- h. Prepare a new version of the updating steps according to the information from this update and use this new version as the base for the production environment update.

Important: Making notes during the update process of the test environment and then using these notes to review the process can be very helpful for the production environment update. It can minimize troubles and short cut outages.

- 6. Perform the update process in production environment. Use the reviewed updating steps and note any differences, errors, and real-time duration. In general, follow these steps:
 - a. Review the time duration and plan the maintenance window.
 - b. Back up the broker domain components and their underlying resources.
 - c. Update underlying components if required.
 - d. Update the broker domain components.
 - e. Check proper functionality of the broker domain.
 - f. Backup the broker domain components and their underlying resources.
 - g. Prepare a new version of the updating steps according to the information from this update as useful information source for the next message broker update.

Important: Making notes during the update process and then using these notes to review the process can be very helpful for the next WebSphere Message Broker update. It can shorter preparation time and it is especially useful in case IT personnel changes.

- 7. After appropriate running time without claims in the production environment, you can declare the update successful and can commit software updates if applicable (for example on AIX).
- 8. Stop the freezing period. Now, the new functionality from fix pack or refresh pack can be used.

If you have more than three typical environments as we have described here, for example several testing environments, simply repeat these steps to conform to your requirements.

Note: This section covers only fix pack and refresh pack update procedures for WebSphere Message Broker V6.0. If you are upgrading from previous versions of the product, refer to *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198, or to the standard product documentation, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247198.html?Open>
<http://www.ibm.com/software/integration/wbimessagebroker/library>

8.2 Characteristics for message broker components

In this section, we describe specific characteristics for each of message broker components in the maintenance strategy area.

8.2.1 Broker

The broker should be at same level as the Configuration Manager and the User Name Server. The broker has three underlying components:

- ▶ Rational Agent Controller
- ▶ WebSphere MQ queue manager
- ▶ Databases for broker repository and optionally for message flow data processing

So, check carefully before you perform a product update to be sure whether actual levels of Rational Agent Controller, WebSphere MQ, or the database product are supported or whether their update is required.

8.2.2 Configuration Manager

The Configuration Manager should be at the same level as the brokers and the User Name Server. The Configuration Manager has only one underlying component: WebSphere MQ queue manager. So, check carefully before you perform a product update whether the actual level of the WebSphere MQ product is supported or whether its update is required.

8.2.3 User Name Server

The User Name Server should be at the same level as the brokers and the Configuration Manager. The User Name Server has only one underlying component — WebSphere MQ queue manager. So, check carefully before you perform a product update whether the actual level of WebSphere MQ is supported or whether its update is required.

8.2.4 Toolkit

Although the main updates for Message Brokers Toolkit are delivered in the same manner (fix packs and refresh packs), you can further update the Toolkit to the more granular level. The Message Brokers Toolkit updates are provided via Rational Product Updater, which is bundled in the Message Brokers Toolkit product.

You can perform the update directly from the IBM Support site through the Internet. Alternatively, you can download the update packages to another computer for an offline update. For more information, refer to Appendix A, “Message Brokers Toolkit update without internet access” on page 285.

The following rules for Message Brokers Toolkit updates are recommended:

- ▶ Find for the latest updates via Rational Product Updater immediately after the Toolkit installation or fix pack or refresh pack update.
- ▶ Find for the latest updates via Rational Product Updater regularly.
- ▶ Look to the Message Brokers Toolkit documentation and readme files for coexistence of the Message Brokers Toolkit with other Rational Application Developer products.

Problem determination

This chapter provides basic and advanced techniques for diagnosing problems with WebSphere Message Broker components. The tools that are available for problem determination provide guidance for determining root cause for issues that you might encounter. This chapter provides specific details about how to collect diagnostic information and data for further support from IBM. It includes the following topics:

- ▶ Basic steps
- ▶ Advanced steps
- ▶ Characteristics for message broker components

9.1 Basic steps

This section provides the basic administrator WebSphere Message Broker knowledge that you needed to identify the failing components. It details the locations and layouts for errors that WebSphere Message Broker and its components generate. You can find useful information to assist with problem determination in the Message Brokers Toolkit, by using the operating system tools, and within the directory structure of the system that is running the product.

Important: This chapter is supplemental to the product documentation regarding problem determination. You should review the following sections:

Testing and debugging message flow applications:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/ag03640_.htm

Troubleshooting and support:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/au03830_.htm

Troubleshooting Reference:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/au09090_.htm

In addition, there is an extensive list of technotes that are available for the message broker at:

<http://www.ibm.com/support/search.wss?tc=SSKM8N+SS3GH2,SSKMAB&rs=849&dc=DB520&rank=8>

9.1.1 Identifying the failing components

If the administrator understands the WebSphere Message Broker components and their relationships as illustrated throughout this book, responding to a problem (or a potential problem) in a production environment is easier. Understanding the interaction between the components allows you to triage, diagnose, and restore service quickly. In addition, if you cannot determine the root cause of the problem or cannot resolve the problem within the service level objective, it is possible that you might be able to restore service through a workaround. If you cannot resolve the problem through internal support, you can gather the appropriate information for the IBM support team.

9.1.2 Categorizing errors

The relationship between the WebSphere Message Broker components and all other functions (such as queue managers, databases, and operating systems) can be complex. If errors are reported, they must be attributed to the offending component or subsystem. A simple scenario to illustrate this concept might be a database that is used by the broker has an expired password. The root cause and what needs to be rectified exists between the database and the operating system, although the error is reported by the broker logging. The ability to recognize and categorize these kinds of errors is important.

The remainder of this section provides additional insight on the best way to recognize and categorize errors.

9.1.3 Event messages

Event messages that show errors are usually the first indication of where a problem might exist and must be resolved in some part of the WebSphere Message Broker's configuration or deployed applications. These messages are displayed in the Message Brokers Toolkit, in the operating system logs, or on the command line in response to a command or a command-line utility.

Event message structure

Event messages that the WebSphere Message Broker generates have the following identification:

BIP8081E

In this identification:

- ▶ BIP is the three-character product family identity, the same as in previous versions of the product.
- ▶ The four-character number following BIP is the unique identifier of the message. This number indicates the condition that generated the message in the product. You can look up this number in the product help or message catalog for further information. (For more information about messages, see "Messages on command line" on page 167.)
- ▶ The final character indicates the event message type: I for an information message, W for a warning message, or E for an error message.

An event message that the WebSphere Message Broker produces that has this structure is known as a *BIP message*. The BIP message code is usually accompanied by a description of the condition that generated the message and sometimes a reason for the condition or a suggestion on how to fix it.

Event message types

Event messages that WebSphere Message Broker generates might be classified as one of three types:

- Information messages

Information messages provide information to the user and do not represent any problem or concern. Success conditions are classed as information messages. For example, BIP8096I is a message that indicates successful command initiation for `mqsistart` on the command line, and BIP0892I is the message for a successful response that is received from the Configuration Manager in the Message Brokers Toolkit (Figure 9-1).

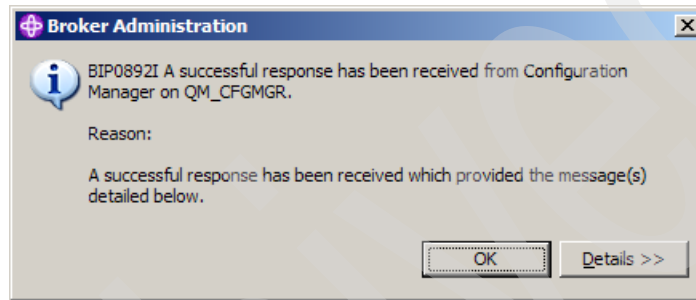


Figure 9-1 Message from the Message Brokers Toolkit

- Warning messages

Warning messages are less severe than error messages and do not represent an immediate problem. However, they indicate situations that might need investigation. These messages are less likely to be seen on the command line or as messages in the Message Brokers Toolkit. You will more likely see these messages in the system log of the machine where the warning occurred.

The Message Brokers Toolkit generates warning messages without a BIP code, and you can see these messages in the *Problems* view and in the *Alerts* view in the Broker Administration perspective of the Message Brokers Toolkit.

- Error messages

The most severe and important messages are the error messages that WebSphere Message Broker generates. You can see these messages on the command line, in the Message Brokers Toolkit, and in the system log of the machine where the product is running. These messages are either generated immediately in response to a failed action, such as trying to start a broker that does not exist, or as a response to a failure during the running of the product.

Messages seen on the command line or as messages in the Message Brokers Toolkit as failure responses to an action from the user.

Messages that are found in the system log or in the Message Brokers Toolkit Event Log usually report events that occur in running components in the WebSphere Message Broker run time. An example would be an error message that is produced when a message flow attempts to access a WebSphere MQ queue that does not exist. These errors do not occur as a result of a user action.

Messages on command line

Messages are displayed on the command line in the same format as the BIP messages in the Message Broker Toolkit when commands have run. Example 9-1 shows a typical command line message. This is a BIP8018 that is generated by attempting to start a broker that is already started. The BIP code is displayed, followed by the description in the first sentence. The remainder of the message gives the reason and a suggestion for solving the problem.

Example 9-1 A BIP message displayed on the command line

```
D:\MQSI\6.0>
```

```
D:\MQSI\6.0> mqsistart BROKER1
```

```
BIP8018E: Component running.
```

```
This component is running, the command issued cannot be run against  
a running component.
```

```
Stop the component and reissue the command.
```

```
D:\MQSI\6.0>
```

For many of the WebSphere Message Broker commands, syntax assistance is displayed in addition to the message if the parameters that are used in the command are incorrect. This information is a useful way to determine the syntax that must be used with these commands. Example 9-2 shows the syntax of the **mqsistop** command and the specific details as to the meanings or requirements of the parameters, followed by the BIP8007E error message, which indicates that a mandatory argument is missing.

Example 9-2 Syntax help for the mqsistop command

```
D:\MQSI\6.0>
```

```
D:\MQSI\6.0> mqsistop
```

```
BIP8117W: Stops a component.
```

Syntax:

```
mqsistop componentName [-q] [-i]
```

Command Options:

'componentName' broker name, Configuration Manager name or
'UserNameserver'

'-q' stops the WebSphere MQ queue manager used by the component

'-i' forces the broker to stop immediately (use with caution).

BIP8007E: Mandatory argument missing.

When using this command interface the user should supply the mandatory argument.

Correct and reissue the command.

D:\MQSI\6.0>

Windows Event Viewer

Messages that the WebSphere Message Broker run time generates are recorded in the local error log. On Windows, this is displayed in the Windows Event Viewer, on UNIX it is the syslog, and on z/OS it is the system console. These messages are all BIP messages that include information and warning messages as well as error messages. Where an error condition has occurred, there might be multiple messages to describe the problem that are generated by different components or parts of the run time.

This section provides information about locating and viewing messages using the Windows Event Log.

The exact location of the Windows Event Log depends upon the version of Windows that you have installed. Typically, it is found under Administrative Tools in the Windows Control Panel. You can also access it through the Computer Management option (right-click **My Computer** then **Manage** → **System Tools** → **Event Viewer**).

Event Viewer Application log

In the Windows Event Viewer, the BIP runtime messages from WebSphere Message Broker are recorded in the Application log (Figure 9-2). For example, messages that the run time produces have a source of *WebSphere Broker v6001*, but messages that other programs, such as WebSphere MQ and DB2 Universal Database, generate are also in the Application log. When you update the version of WebSphere Message Broker, a corresponding change is seen in the source name to indicate from which version the message came because multiple versions can be installed on the same machine. Thus, both versions can generate messages in the Application log.

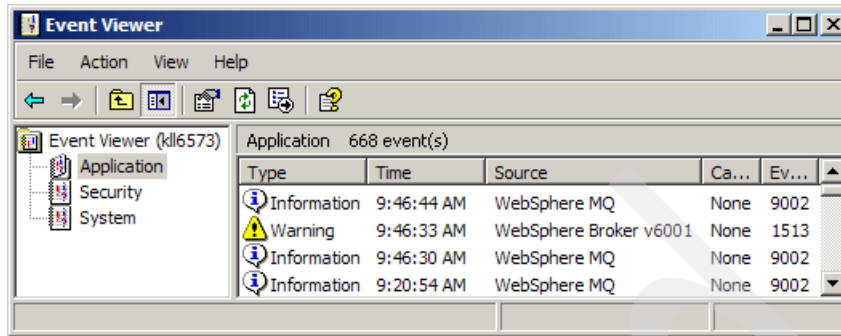


Figure 9-2 Windows Event Viewer - Application log

The Event column of the BIP code in the Application log (for example, the BIP2648E message is an error type). To view more details about any message in the Event Viewer, double-click the message to open it in a window (called Event Properties).

When the WebSphere Message Broker run time sends error messages to the Application log, there is usually a set of messages to indicate what problems have occurred. There is often a sequence of messages that can be followed to determine the cause of the error.

Tip: When developing message flows, disconnect nodes connect to Failure and Catch terminals. When an error occurs the message flows can put errors into the Application log. If the Failure and Catch terminals are connected, then the flow handles the errors and, therefore, there are no error messages in the Application log.

Event Viewer System log

The System Log in the Windows Event Viewer Information displays information about the starting, running, and stopping of WebSphere Message Broker components. Components such as brokers, the Configuration Manager, and User Name Server are created as Windows Services. So, when commands are sent to these components to start or to stop them, messages are written to the System Log to indicate the status of the command.

Event Viewer log properties

It is important to ensure that you have set up the properties for the Application log and System log correctly in the Windows Event Viewer. Otherwise, messages could fail to be recorded, which can lead to difficulties in troubleshooting problems in the run time. The Application log in particular can fill up very quickly if there are problems in the system and either can become full (in

which case no new messages are written) or can begin overwriting earlier messages. Either way, the original cause of a problem can be lost.

To check the properties of the Application log:

1. In the Windows Event Viewer, right-click the **Application** log in the left-side pane and select **Properties** from the context menu. Increase the *Maximum log size* to determine the number of messages that the log can detain. The size of log that is required depends on the amount of activity that is likely to be recorded over time.
2. Select an action under **When maximum log sized is reached**. Selecting the first option, *Overwrite events as needed*, is recommended, because this prevents the log from becoming full and the loss of recent messages. However, the option that is selected must take into account local working practices and how long event messages can be retained. Clearing the log manually means that no old messages are lost, but it does run the risk of the log becoming full and new messages not being recorded.

You can clear the log manually at any time by clicking **Clear Log**. You can save the Application log by right-clicking the Application log and selecting **Save log file as** from the context menu.

UNIX syslog

Comparable messages to those found in the Windows Event Viewer are located on UNIX, routed by the syslog daemon. It is very important that the UNIX system be configured to capture the logging output in a location that the broker administrator can access quickly. You should begin diagnostic activity here. WebSphere Message Broker produces minimal logging even at the level of debug and the contents are easy to understand.

UNIX syslog configuration

To configure syslog:

1. Log on as root.
2. Enter the following commands to create a file called user.log.

On UNIX systems:

```
touch /var/adm/user.log
chown root:mqbrkrs /var/adm/user.log
chmod 640 /var/adm/user.log
```

On Linux:

```
touch /var/log/user.log
chown root:mqbrkrs /var/log/user.log
chmod 640 /var/log/user.log
```

3. Add the following line to the `/etc/syslog.conf` file to redirect the debug level messages to the file `user.log`.

On UNIX systems:

```
user.info /var/adm/user.log
```

On Linux:

```
user.info /var/log/user.log
```

You can use `user.*` instead of `user.info` in the preceding examples. The asterisk (*) means that the information, notice, warning, and debug messages are caught.

You can add similar lines to direct information, warning, and error messages to the `user.log`.

4. Restart the syslog daemon and enter the following command:

On AIX:

```
refresh -s syslogd
```

On HP-UX and Solaris:

```
kill -HUP 'cat /etc/syslog.pid
```

On Linux (if `rc.d` is not a soft link):

```
/etc/init.d/syslogd restart or /etc/rc.d/init.d/syslogd restart
```

Check that the syslog has been configured to route messages of at least the level of *info* to a file accessible to the broker administrator. Example 9-3 shows the level set to *info* on AIX.

Example 9-3 File syslog.conf example

```
root@m106910f:/etc # tail syslog.conf
# example:
# "mail messages, at debug or higher, go to Log file. File must exist."
# "all facilities, at debug and higher, go to console"
# "all facilities, at crit or higher, go to all users"
# mail.debug          /usr/spool/mqueue/syslog
# *.debug             /dev/console
# *.crit              *
# *.debug             /tmp/syslog.out      rotate size 100k files 4
# *.crit              /tmp/syslog.out      rotate time 1d
user.info /var/adm/user.log
```

After making changes to the syslog configuration, be sure to use the refresh or restart process that is defined to the version of UNIX that you used. Example 9-4 shows how to refresh the syslog daemon on AIX.

Example 9-4 Refreshing syslog daemon on AIX

```
root@m106910f:/etc # refresh -s syslogd
0513-095 The request for subsystem refresh was completed successfully.
```

You can find specific documentation for the syslog process by searching for *Linux and UNIX systems: Configuring the syslog daemon* in the WebSphere Message Broker product documentation, which is available at:

<http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp>

Understanding UNIX syslog events

This section reviews the structure of a syslog based event message that WebSphere Message Broker components, such as a broker or User Name Server, generate. Example 9-5 is a complete message that notifies a broker startup.

Example 9-5 Informational message in user.log

```
Jun 14 09:30:39 m106910f user:info WebSphere Broker v6001[37822]:
(BROKER2)[1]BIP2001I: The WebSphere Message Brokers service has started
at version 6001; process ID 33418. : BROKER2.service:
/build/S600_P/src/AdminAgent/ControlProcess/rios_aix_4/ImbControlServic
e.cpp: 338: ImbControlService::StartNewAA: :
```

Table 9-1 shows a breakdown of the structure of Example 9-5.

Table 9-1 Structure of user.log

Usage	Example
Timestamp	Jun 14 09:30:39
Hostname	m106910f
Message Priority	user:info
Version	WebSphere Broker v6001
bipservice process ID (PID)	[37822]:
Component name	(BROKER2)
Detailed message showing bipbroker PID	BIP2001I: The WebSphere Message Brokers service has started at version 6001; process ID 33418.

Although the detailed message varies for each entry in the syslog, the overall structure remains consistent. Example 9-6 shows a message that is generated by a shutdown of the broker. Notice that the level has changed to *warn|warning*.

Example 9-6 Warning message user.log

```
Jun 14 10:42:49 m106910f user:warn|warning WebSphere Broker
v6001[39492]: (BROKER2)[1]BIP2002W: The WebSphere Message Brokers
service has stopped. : BROKER2.service:
/build/S600_P/src/AdminAgent/ControlProcess/rios_aix_4/ImbControlService.cpp: 518: ImbControlService::Cleanup:
```

Note: It is likely that relevant messages might be shown at the priority level of *warn|warning* but that they might not indicate problems or give cause for concern. For example, in Example 9-6, the administrator requested that the broker be shutdown. The broker stopping was an expected result of the command, although messages were produced at the *warn|warning* level. Being able to distinguish between unexpected and expected alerts is a necessary technique for effective problem determination.

Example 9-7 shows a problem that is reported at the *err|error* level. To simulate this condition, we stopped the database while the broker was still running. Database issues are a common errors that are captured by the syslog. In most cases, there are two messages immediately together — one for the ODBC return message and one for the SQL State message, as shown in Example 9-7.

Example 9-7 Error messages in user.log

```
Jun 14 11:08:31 m106910f user:err|error WebSphere Broker v6001[28510]:
(BROKER2)[1]BIP2321E: Database error: ODBC return code '-1'. :
BROKER2.agent: /build/S600_P/src/DataFlowEngine/ImbOdbc.cpp: 227:
ImbOdbcHandle::checkRcInner: :
```

```
Jun 14 11:08:31 m106910f user:err|error WebSphere Broker v6001[28510]:
(BROKER2)[1]BIP2322E: Database error: SQL State '08001'; Native Error
Code '-30081'; Error Text '[IBM][CLI Driver] SQL30081N A communication
error has been detected. Communication protocol being used: "TCP/IP".
Communication API being used: "SOCKETS". Location where the error was
detected: "9.42.171.249". Communication function detecting the error:
"connect". Protocol specific error code(s): "79", "*", "*'.
SQLSTATE=08001 '. : BROKER2.agent:
/build/S600_P/src/DataFlowEngine/ImbOdbc.cpp: 355:
ImbOdbcHandle::checkRcInner: :
```

You should review the syslog before and after you make any changes to a component as standard practice.

AIX error log

A comparable location to the Windows System Log on AIX is accessed through the **errpt** command. This command generates a report of logged errors from the AIX error log. Certain types of events, such as a WebSphere Message Broker process crash that creates a core file, can generate debugging information that you can retrieve with this command.

To simulate this effect, the PID of the bipservice running BROKER2 was queried so a SIGQUIT signal was sent to it, as shown in Example 9-8.

Example 9-8 Sending a SIGQUIT to the bipservice generating a core file

```
wmb@m106910f:/var/mqsi $ ps -ef | grep "bipservice BROKER2" | grep -v  
grep | awk '{print $2}' | xargs kill -SIGQUIT $1
```

```
wmb@m106910f:/var/mqsi $ ls -l core  
-rw-rw---- 1 wmb mqbrkrs 4404658 Jun 16 11:13 core
```

You can run a simple test on the core file to determine its origin by using the **file** command, as shown in Example 9-9.

Example 9-9 Testing the core file's origin with the file command

```
wmb@m106910f:/var/mqsi $ file core  
core: AIX core file fulldump largedata 32-bit, bipbroker
```

Running the **errpt** command with the **-a** option shows the detailed information for this failure condition, as shown in Example 9-10.

Example 9-10 Command output for errpt -a

LABEL:	CORE_DUMP
IDENTIFIER:	A63BEB70
Date/Time:	Fri Jun 16 11:13:36 EDT 2006
Sequence Number:	109
Machine Id:	0006910F4C00
Node Id:	m106910f
Class:	S
Type:	PERM
Resource Name:	SYSPROC

Description

SOFTWARE PROGRAM ABNORMALLY TERMINATED

Probable Causes
SOFTWARE PROGRAM

User Causes
USER GENERATED SIGNAL

Recommended Actions
CORRECT THEN RETRY

Failure Causes
SOFTWARE PROGRAM

Recommended Actions
RERUN THE APPLICATION PROGRAM
IF PROBLEM PERSISTS THEN DO THE FOLLOWING
CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
SIGNAL NUMBER
3
USER'S PROCESS ID:
43714
FILE SYSTEM SERIAL NUMBER
13
INODE NUMBER
2
PROCESSOR ID

9.1.4 Locating additional information

In addition to the information that accompanies event messages, there are two other locations that you can use to find more information:

- ▶ Information Center
- ▶ Messages.html
- ▶ IBM Software Support

Finding diagnostic messages in the Information Center

You can use the Information Center in the Message Brokers Toolkit (or other source of product help) to locate information about a BIP message, as follows:

1. Open the Message Brokers Toolkit Help by selecting **Help → Help Contents**.
2. Click **WebSphere Message Broker V6.0** in the list of contents.
3. Select **Diagnostic messages** from the bottom of the WebSphere Message Broker V6.0 list of contents. In the Diagnostic messages page on the right side, enter a four-character number in the box labelled *Enter message number*, and click **Search** to locate the BIP message description with that number. Figure 9-3 shows an example for BIP5285.



WebSphere Message Broker V6.0

Diagnostic messages

Enter message number:

BIP5285

Severity

20 : Error

Message

Message Translation Interface Parsing Errors have occurred:
Message Set Name : '<insert_0>'
Message Set Level : '<insert_1>'
Message Format : '<insert_2>'
Message Type Path : '<insert_3>'

Explanation

Review further error messages for an indication to the cause of the errors.

Figure 9-3 Searching for diagnostic messages in the Information Center

4. For each message, the following information is returned:

- The BIP code
- Severity
- Message
- Explanation
- Response

Finding diagnostic messages in messages.html file

Another place to find information about BIP messages is in an HTML file called messages.html. You can find this file in the messages directory in the installation path. In a typical Windows installation, this path is C:\Program Files\mqsi\6.0\messages. Several directories exist here for translated messages and in each directory is a messages.html file.

The messages.html file contains a list of BIP messages in order, in a table with links to further information at the top of the file. Locate the required BIP message and click the link to view the information about it. This information is the same as in the Help system, except for the addition of an Inserts table for each message, which details the variable information for that message. The message displayed with a BIP2623 is as follows:

Unable to open queue '&2' on WebSphere MQ queue manager '&1':
completion code &3; reason code &4.

Using the table provided in the messages.html file enables you to determine the source of variable information in the error message. Replace the ampersand (&) variables in the messages.html file with the descriptions from the table to understand the message. For example, if you received the error message in this example, then the next step is to look up the WebSphere MQ reason code to help determine the cause of the error, as shown in Table 9-2.

Table 9-2 Example inserts for a BIP message

Insert number	Description	Datatype
&1	MQ queue manager name	CHARACTER
&2	MQ queue name	CHARACTER
&3	MQ return code	CHARACTER
&4	MQ reason code	CHARACTER

Finding information through IBM Software Support

Embedded diagnostic information that is delivered with the products can contain sufficient information for problem determination. If you cannot discover the root cause for an error, it might be necessary to broaden the search to the online resources that are available at the IBM Software Support Web site. The primary site for IBM Software Support is located at:

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

The three main software products that we use in this book include WebSphere Message Broker, WebSphere MQ, and DB2 Universal Database. Each product has a primary support page:

- ▶ WebSphere Message Broker
<http://www.ibm.com/software/integration/wbimessagebroker/support>
- ▶ WebSphere MQ
<http://www.ibm.com/software/integration/wmq/support>
- ▶ DB2 Universal Database
<http://www.ibm.com/software/data/db2/udb/support>

You should visit each of these sites regularly to check for new product support information and to stay up-to-date with report problems.

IBM Software Support RSS feeds

As an alternative to continually visiting Web sites for current information, IBM Software Support provides Really Simple Syndication (RSS) 2.0 feeds for each of the products. RSS feeds enable real-time updates to be streamed to the administrator. The primary site for IBM Software Support RSS feed is available at:

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

The three main software products that we use this book each have their own RSS feeds that you can access through any RSS reader. While many RSS feeds exist, the following are most relevant for message broker:

- ▶ WebSphere Message Broker
<http://www.ibm.com/software/support/rss/websphere/849.xml>
- ▶ WebSphere MQ
<http://www.ibm.com/software/support/rss/websphere/171.xml>
- ▶ DB2 Universal Database™ for Linux
<http://www.ibm.com/software/support/rss/db2/71.xml>

9.1.5 Other useful logs

A variety of other logs that record information are created when using WebSphere Message Broker. This section gives a brief description of some of the other logs that are generated.

Installation logs

Each product produces a log during installation. If any errors occur during the installation process, the details are recorded in the installation logs.

WebSphere Message Broker runtime

Installing the base product and maintenance packages of WebSphere Message Broker on Windows creates the file `mqsib6_install.log` in the home directory of the user who is performing the installation. The home directory is usually `C:\Documents and Settings\UserID`, where *UserID* is the login ID of the user who installed the product.

On UNIX, the directory would be the one defined by `$HOME` of root, as illustrated in Example 9-11.

Example 9-11 Locating status message in an installation log

```
root@m106910f:/ # grep BIP862 $HOME/mqsib6_install.log
(Jun 8, 2006 5:42:58 PM), mqsib6.Setup,
com.ibm.ismp.beans.wizard.actions.WriteToLogWizardAction, dbg, BIP8621:
Install of IBM WebSphere Message Broker 6.0 has started
(Jun 8, 2006 5:51:42 PM), mqsib6.Setup,
com.ibm.ismp.beans.wizard.actions.WriteToLogWizardAction, dbg, BIP8622:
Install of IBM WebSphere Message Broker 6.0 has finished successfully
```

The installation log is a useful tool for verifying applied maintenance. Ensuring a successful installation is a recommended practice before using the product and can also be revised in the event of a suspected problem. Therefore, you should archive this log between maintenance updates.

Reviewing the first line of the installation log shows the location of a file called `vpd.properties`. WebSphere Message Broker uses an installation wrapper procedure that hides the operating system specific details. The `vpd.properties` file contains the specific location of installed files, including version information. For a better understanding of what files are altered or changed during the application of maintenance, you can use this file for educational purposes.

Example 9-12 from the vpd.properties on Windows shows that only the base product for the broker component was installed and that no maintenance was applied.

Example 9-12 Example of vpd.properties file on Windows

```
mqsib60.brokercomp|6|0|0|0| |1=Broker Component|Broker Component|Broker  
Component| | |5724-J04|C:\Program  
Files\IBM\MQSI\6.0|0|0|1|mqsib60.brokerf|6|0|0|0| |1|0|false|  
|true|3|mqsib60.brokercomp|6|0|0|0| |1
```

Because the AIX installation in this example has maintenance applied, the vpd.properties file shows the change in version for the broker component, as shown in Example 9-13.

Example 9-13 vpd.properties file showing broker component on AIX

```
root@m106910f:/ # cat /usr/lib/objrepos/vpd.properties | grep  
mqsib60.brokercomp  
mqsib60.brokercomp|6|0|1|0| |1=Broker Component|Broker Component|Broker  
Component| | |5724-J04|/opt/IBM/mqsi/6.0|0|0|1|mqsib60.brokerf|6|0|1|0|  
|1|0|false| |true|3|mqsib60.brokercomp|6|0|1|0| |1
```

Message Brokers Toolkit

You can find the Message Brokers Toolkit installation log in the installation directory of the Message Brokers Toolkit, for example:

C:\MessageBrokers\logs\wmbt_install.log

You can find the log for the uninstallation of the Message Brokers Toolkit in the home directory, for example:

C:\Documents and Settings\UserID\wmbt_uninstall.txt

WebSphere MQ

The installation log for WebSphere MQ is created in the temp directory. This directory varies from machine to machine. To find the location of this directory type the following on a command line:

```
set temp
```

This command displays the location of the temp directory. The WebSphere MQ installation log file also includes a time and date stamp in the name. The following shows an example WebSphere MQ install log location and name for WebSphere MQ V6.0:

```
C:\Documents and Settings\wmbuser\Local  
Settings\Temp\MQv6_Install_2005-10-06T12-48-54.log
```


ODBC Drivers for Cloudscape or DB2 Universal Database

You can find the installation log for any DB2 Universal Database component, including the ODBC Drivers for Cloudscape™, in the home directory. The following show the name and an example location for the logs that these DB2 products produce:

► On Windows:

C:\Documents and Settings\UserID\My Documents\DB2LOG\db2.log
C:\Documents and Settings\UserID\My Documents\DB2LOG\db2wi.log

► On UNIX:

/tmp/db2setup.err, /tmp/db2setup.his, /tmp/db2setup.log

The db2.log or db2setup.his log file contains all of the installation information for every DB2 Universal Database component that is installed, while the db2wi.log or db2setup.log files contain information for just the last DB2 Universal Database component that was installed on the machine. On UNIX, there is also the db2setup.err file, which contains all errors from the DB2 Universal Database installation.

MRM logs

A number of logs are generated in Message set Projects during the import or export of message definitions, such as XML schema. A log is also created by the runtime if deployment of a message set with a TDS layer fails. This type of error is indicated by a BIP1836 message in the Message Brokers Toolkit Event Log.

9.1.6 Collecting data for further support

Normally, errors encountered with one of the products reviewed in this chapter are the result of simple configuration issues or non-supported use of a component. Typically, these types of errors can be solved by simple review of the error, configuration, and what changes have been applied to the environment prior to the error occurring. If it does appear that there might be a product problem, then you should open a Problem Management Record (PMR) through standard support channels.

Before opening the PMR, it is important that you gather certain information early so that the problem and potential software defect can be determined. There is a very detailed technote *MustGather: Read first for all WebSphere Message Broker Versions*, available at:

<http://www-1.ibm.com/support/docview.wss?rs=959&uid=swg21209857>

If you collect data early, even before you open the PMR, it can help IBM Support determine quickly whether:

- ▶ Symptoms match known problems (rediscover).
- ▶ There is non-defect problem that can be identified and resolved.
- ▶ There is a defect that identifies a workaround to reduce severity.
- ▶ Locating the root cause can speed development of a code fix.

Before opening a PMR, it is also important to gather all software levels for the accessed products. For example, the products that need to be reported on for the broker component would be the operating system, WebSphere Message Broker, WebSphere MQ, and the databases that is used. A simple script can be created to gather this information so that the results are consistent. A secondary benefit is that the commands that are required for each platform are documented and accessed easily.

Example 9-14 shows a script to query for software levels on AIX.

Example 9-14 Software installation report script

```
#!/usr/bin/ksh
## wmbinstlprt.ksh
## Produces a simple report of the software levels related to WMB.

osver=`uname -a`
dbver=`su - wmb "-c db2level | tr -d \"\n\""`
mqver=`su - mqm "-c dspmqver -p 7 | grep -v "AMQ8351" | sed 's/^/ /g'`
mbver=`su - wmb "-c mqsIService -v | grep -v 8071I"`
pcver=`lslpp -aL | egrep -i 'websphere|db2'`

echo "\n -- WebSphere Message Broker Install Report --\n"
echo " $osver\n"
echo " $dbver\n"
echo "$mqver"
echo "$mbver"
echo "$pcver\n"
```

When executed as root, the report should produce output similar to that shown in Example 9-15.

Example 9-15 Software installation report output

```
-- WebSphere Message Broker Install Report --

AIX m106910f 3 5 0006910F4C00
```

DB21085I Instance "wmb" uses "32" bits and DB2 code release "SQL08025" with level identifier "03060106".Informational tokens are "DB2 v8.1.1.112", "s060429", "U807381", and FixPak "12".Product is installed at "/usr/opt/db2_08_01".

```
Name:      WebSphere MQ classes for Java
Version:   6.0.1.1
CMVC Level: j600-101
Build Type: Production
Name:      WebSphere MQ classes for Java Message Service
Version:   6.0.1.1
CMVC Level: j600-101
Build Type: Production
Name:      WebSphere MQ
Version:   6.0.1.1
CMVC level: p600-101-060504
BuildType: IKAP - (Production)
```

db2_08_01.ca	8.1.1.64	C	F	Configuration
Assistant				
db2_08_01.cc	8.1.1.64	C	F	Control Center
db2_08_01.ch.en_US.iso88591				
db2_08_01.cj	8.1.1.64	C	F	Java Common files
db2_08_01.client	8.1.1.64	C	F	Base Client Support
db2_08_01.cnvucs	8.1.1.64	C	F	Code Page Conversion
Tables -				
db2_08_01.conn	8.1.1.64	C	F	Connect Support
db2_08_01.conv	8.1.1.64	C	F	Code Page Conversion
Tables				
db2_08_01.cs.rte	8.1.1.64	C	F	Communication
Support - TCP/IP				

The remainder of this section illustrates some of the common procedures that are outlined in the *MustGather* technotes. Each example introduces some of the possible ways that system type tracing might be gathered. The procedures that we describe here might be slightly different between operating systems, so study the notes from the *MustGather* technotes in detail before you begin.

MustGather: Procedure for taking startup trace of a broker

To take a startup trace of a broker, perform the commands in the following steps:

1. Stop the broker as shown in Example 9-16.

Example 9-16 Stopping the broker

```
wmb@m106910f:/var/mqsi $ mqsistop BROKER2
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

2. Enable the broker agent startup trace, as shown in Example 9-17.

Example 9-17 Enabling mqsiservice trace

```
wmb@m106910f:/var/mqsi $ mqsiservice BROKER2 -r Trace=debug
```

```
BIPv600 en US
```

```
ucnv Console CCSID 819 dft ucnv CCSID 819
```

```
ICUW ISO-8859-1 ICUA ISO-8859-1
```

```
Install Path = /opt/IBM/mqsi/6.0
```

```
Working Path = /var/mqsi
```

```
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08
```

```
process id = 19122
```

```
broker db name = MQSITCP
```

```
broker db userId = wmb
```

```
broker db password =
```

```
queue manager = QM_BROKER2
```

```
pubsub migration = no
```

```
fastpath Queue Manager = no
```

```
configuration timeout = 300
```

```
configuration delay timeout = 60
```

```
statistics major interval = 60
```

```
Trace, bipservice, bipbroker, bipconfigmgr, bipuns = debug
```

```
ComponentType = Broker
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

```
wmb@m106910f:/var/mqsi $ mqsiservice BROKER2 -r TraceSize=1073741824
```

```
BIPv600 en US
```

```
ucnv Console CCSID 819 dft ucnv CCSID 819
```

```
ICUW ISO-8859-1 ICUA ISO-8859-1
```

```
Install Path = /opt/IBM/mqsi/6.0
```

```
Working Path = /var/mqsi
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08
process id = 19122
broker db name = MQSITCP
broker db userId = wmb
broker db password =
queue manager = QM_BROKER2
pubsub migration = no
fastpath Queue Manager = no
configuration timeout = 300
configuration delay timeout = 60
statistics major interval = 60
Trace, bipservice, bipbroker, bipconfigmgr, bipuns = debug
TraceSize = 1073741824
ComponentType = Broker
```

```
BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
```

Important: Make sure to follow the instructions properly, using the specific attributes. In some instances, the command will accept a non-valid attribute but it will not report an error. For example, setting the trace attribute to notavalidattribute in Example 9-18 was accepted without warning.

Example 9-18 Non-valid attribute without an error

```
wmb@m106910f:/var/mqsi $ mqsiservice BROKER2 -r
Trace=notavalidattribute
```

```
BIPv600  en US
      ucnv Console CCSID 819      dft ucnv CCSID 819
      ICUW ISO-8859-1      ICUA ISO-8859-1
```

```
Install Path = /opt/IBM/mqsi/6.0
Working Path = /var/mqsi
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08
process id = 34006
broker db name = MQSITCP
broker db userId = wmb
broker db password =
queue manager = QM_BROKER2
pubsub migration = no
fastpath Queue Manager = no
configuration timeout = 300
```

```
configuration delay timeout = 60
statistics major interval = 60
Trace, bipservice, bipbroker, bipconfigmgr, bipuns = notavalidattribute
ComponentType = Broker
```

BIP8071I: Successful command completion.

3. Start the broker, as shown in Example 9-19.

Example 9-19 Starting the broker

```
wmb@m106910f:/var/mqsi $ mqsistart BROKER2
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

```
wmb@m106910f:/var/mqsi $
```

At this point in the procedure, the problem should be re-created.

4. Stop the broker. Verify that the broker is stopped before continuing, as shown in Example 9-20.

Example 9-20 Stopping the broker

```
wmb@m106910f:/var/mqsi $ mqsistop BROKER2
```

BIP8071I: Successful command completion.

```
wmb@m106910f:/var/mqsi $
```

5. Retrieve the trace log for the specified component, as shown in Example 9-21.

Example 9-21 Retrieving the trace log

```
wmb@m106910f:/var/mqsi $ mqsireadlog BROKER2 -t -b agent -f -o
agent.xml
```

BIP8071I: Successful command completion.

```
wmb@m106910f:/var/mqsi $
```

6. Format the XML log file, as shown in Example 9-22.

Example 9-22 Formatting the trace log

```
wmb@m106910f:/var/mqsi $ mqsiformatlog -i agent.xml -o agent.txt
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

7. Turn off the trace, as shown in Example 9-23.

Example 9-23 Disabling mqsiservice trace

```
wmb@m106910f:/var/mqsi $ mqsiservice BROKER2 -r Trace=""
```

```
BIPv600 en US
```

```
ucnv Console CCSID 819 dft ucnv CCSID 819
```

```
ICUW ISO-8859-1 ICUA ISO-8859-1
```

```
Install Path = /opt/IBM/mqsi/6.0
```

```
Working Path = /var/mqsi
```

```
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08
```

```
process id = 34006
```

```
broker db name = MQSITCP
```

```
broker db userId = wmb
```

```
broker db password =
```

```
queue manager = QM_BROKER2
```

```
pubsub migration = no
```

```
fastpath Queue Manager = no
```

```
configuration timeout = 300
```

```
configuration delay timeout = 60
```

```
statistics major interval = 60
```

```
TraceSize = 1073741824
```

```
ComponentType = Broker
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

```
wmb@m106910f:/var/mqsi $ mqsiservice BROKER2 -r TraceSize=""
```

```
BIPv600 en US
```

```
ucnv Console CCSID 819 dft ucnv CCSID 819
```

```
ICUW ISO-8859-1 ICUA ISO-8859-1
```

```
Install Path = /opt/IBM/mqsi/6.0
```

```
Working Path = /var/mqsi
```

```
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08
```

```

process id = 34006
broker db name = MQSITCP
broker db userId = wmb
broker db password =
queue manager = QM_BROKER2
pubsub migration = no
fastpath Queue Manager = no
configuration timeout = 300
configuration delay timeout = 60
statistics major interval = 60
ComponentType = Broker

```

```

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $

```

Tip: Before opening the PMR, check that the trace logs were populated. It might not be possible to ensure that the specific error that you intended to capture is present in the trace logs. However, it is possible to determine whether the correct component was picked up by the settings. In our example, the agent.txt file indicates that BROKER2 was accessed as follows:

```
wmb@m106910f:/var/mqsi $ head agent.txt
```

Timestamps are formatted in local time, 240 minutes before GMT.
Trace written by version 6001; formatter version 6001

```

2006-06-13 10:49:00.672043      1 { ImbAdminAgent::main  ,
'BROKER2'
2006-06-13 10:49:00.676558      1  ImbAdminAgent::main 'About to
drive ImbParserManager constructor'
2006-06-13 10:49:00.676588      1  {
ImbParserManager::ImbParserManager
2006-06-13 10:49:00.676764      1  }
ImbParserManager::ImbParserManager
2006-06-13 10:49:00.676776      1  ImbAdminAgent::main 'About to
drive ImbAdminAgent constructor'
2006-06-13 10:49:00.676795      1  { ImbNamedMutex::ImbNamedMutex
, 'mqsiV6ListenerConfigFileMutex'
2006-06-13 10:49:00.676804      1      ImbNamedMutex::ImbNamedMutex
'Creating local pthread mutex'

```

8. After you open a PMR, submit the agent.txt file to IBM Support.

MustGather: Procedure to trace mqsi* command executables

To trace mqsi* command executables, perform the commands in the following steps:

1. Set the environment variable MQSU_UTILITY_TRACE, as shown in Example 9-24.

Example 9-24 Export MQSI_UTILITY_TRACE variable

```
wmb@m106910f:/var/mqsi $ export MQSI_UTILITY_TRACE=debug
wmb@m106910f:/var/mqsi $
```

2. Run the command that you want to trace, as shown in Example 9-25.

Example 9-25 Running mqsilist to trace

```
wmb@m106910f:/var/mqsi $ mqsilist

BIP8099I: Broker: BROKER2 - QM_BROKER2
BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
```

3. Retrieve the trace log for the specified command, as shown in Example 9-26.

Example 9-26 Retrieving the trace log

```
wmb@m106910f:/var/mqsi $ mqsireadlog utility -t -b mqsilist -f -o
mqsilist.xml

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
```

4. Format the XML trace file, as shown in Example 9-27.

Example 9-27 Formatting the trace log

```
wmb@m106910f:/var/mqsi $ mqsiformatlog -i mqsilist.xml -o mqsilist.txt

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
```

Tip: Before you open the PMR, check that the trace logs were populated. It might not be possible to ensure that the specific error that you intended to capture is present in the trace logs. However, it is possible to determine whether the correct command was picked up by the settings. In our example, the mqsilist.txt file indicates that the **mqsilist** command was accessed as follows:

```
wmb@m106910f:/var/mqsi $ head mqsilist.txt
```

Timestamps are formatted in local time, 240 minutes before GMT.
Trace written by version 6001; formatter version 6001

```
2006-06-13 11:16:33.723339      1 {  
  ImbPlatformSupport::getEnvironmentVariable ,  
  'MQSI_UTILITY_TRACESIZE'  
2006-06-13 11:16:33.723407      1 }  
  ImbPlatformSupport::getEnvironmentVariable , ''  
2006-06-13 11:16:33.723419      1 { mqsi::main , 'mqsilist'  
2006-06-13 11:16:33.723438      1   mqsi::main 'current locale  
  is' , 'en_US en_US en_US en_US en_US en_US en_US'  
2006-06-13 11:16:33.723449      1 {  
  ImbCmdBase::processCommandString  
2006-06-13 11:16:33.723464      1 {  
  ImbCmdmqsilist::processCommand  
2006-06-13 11:16:33.728199      1 {  
  ImbCmdLibUnix::getNextBroker , '', ''
```

5. After you open a PMR, submit the mqsilist.txt file to IBM Support for problem determination.

MustGather: Procedure for initiating an ODBC trace

To initiate an ODBC trace, perform the commands in the following steps:

1. For UNIX platforms, edit the .odbc.ini file to initiate trace for ODBC activity. The location of the file is referenced by the environment variable \$ODBCINI (or \$ODBCINI64). Ensure that you edit the correct file (Example 9-28).

Example 9-28 Location of .odbc.ini file

```
wmb@m106910f:/var/mqsi $ echo $ODBCINI  
/var/mqsi/odbc/.odbc.ini  
wmb@m106910f:/var/mqsi $
```

2. Under the stanza entry [ODBC] change Trace=0 to Trace=1. Set TraceFile to *<a directory with free space>/odbctrace.out*. In Example 9-29, a file system that is dedicated to ad-hoc logging and tracing mounted to a directory called /traces is used.

Attention: On a critical systems such as those in production, the best practice is to create a file system that is dedicated to tracing and secondary or ad-hoc logging activity. Ideally, this file system should be an expandable file system such as on a Storage Area Network (SAN). For example, pointing the ODBC tracing activity to a mission critical file system (such as /var/mqsi, /var/mqm, or /tmp on UNIX) can be risky. If these file systems became full, then any application that uses them could become unavailable. Some tracing options such as those accessed through .odbc.ini cannot have their size controlled or limited. Therefore, you should use this type of tracing with caution and should monitor it closely.

Example 9-29 Edited attributes in .odbc.ini file

```
wmb@m106910f:/opt/IBM/mqsi/6.0/merant $ cat $ODBCINI | grep Trace
Trace=1
TraceFile=/traces/odbc.trc
TraceDll=/opt/IBM/mqsi/6.0/merant/lib/odbctrac.so
wmb@m106910f:/opt/IBM/mqsi/6.0/merant $
```

3. Re-create the problem or perform the activity that requires database tracing. For example, if the broker is failing to establish a connection with the database, tracing activity during **mqsisstart** might be necessary. Notice that the odbctrace.out file grows rapidly even with limited activity (Example 9-30).

Example 9-30 Example of starting the broker with odbc trace

```
wmb@m106910f:/traces $ mqsisstart BROKER2;sleep 1;ls -s;sleep 10;ls -s
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

total 488	488 odbc.trc
total 7240	7240 odbc.trc

4. After you open the PMR, submit the odbc.trc file to IBM Support for problem determination.

9.2 Advanced steps

The topics that we cover in this section provide very detailed information about extracting diagnostic information for problems that are difficult to solve. We also discuss further suggestions for identifying components or functions that might contribute to a failure scenario.

9.2.1 Tracing

You can use the following traces for problem determination:

- ▶ User tracing
- ▶ Service tracing
- ▶ Configuration Manager Proxy API tracing
- ▶ JVM tracing

Utilizing user tracing

Creating user traces is likely to be the most common tracing activity. There is a considerable amount of functionality available; however, it is simple to use. You can use these types of traces to solve application development issues (message sets and message flows problems) and to help solve administrative problems such as configuration issues with an execution group or message flow.

There are two significant forms of user tracing:

- ▶ Changing user trace attributes with the **mqsi changetrace** command.
- ▶ Injecting detail into the user trace from trace nodes in message flows.

The first option can be preformed in a production environment with no prior preparation, although the data presented is arbitrarily determined by WebSphere Message Broker. The second option gives more granular control over data presented but requires preparation while the message flow is being developed.

Attention: Using trace nodes causes significant overhead, even when the trace nodes themselves are not accessed by the message flow path (the code path). For this reason, you should not use trace nodes in production until you have exhausted other means such as **mqsi changetrace**. For problems with message parsing that occur in production, attempt to capture an instance of the offending message and to reproduce the problem in a test environment that uses trace nodes.

Changing any attributes in trace nodes requires new deployments and cannot be controlled on an ad-hoc basis. Therefore, changing an attribute is likely to be a function accomplished by development and not production support. Most likely,

you will use the `mqsi changetrace` command in production, so for the remainder of this section, we discuss its usage.

Altering user tracing from the `mqsi changetrace` command

Controlling the user trace from the `mqsi changetrace` command is simple. There are only three significant decisions that you need to make:

- ▶ Should the trace occur at the execution group or at the message flow level. If you need to study results from multiple executions from one high-level activity, choose the execution group level. If you know that the issue is isolated to only one message flow, choose that option.
- ▶ To what size will you allow the user trace to grow before it rolls over. It is important to note that these types of traces are stored in binary format and cannot be viewed readily without running the post-processing. Therefore, a considerable amount of data can be stored, yet you cannot monitor the file directly for the occurrence of the problem.

Tip: A simple solution to determining size is to create a simple script to execute the test case between the setting and queried of the user trace. In this manner the process can be executed repeatedly until the test condition is narrowed into a very granular procedure until only the problem being worked is captured.

An alternative method is to initiate the user trace and then continually view the system log until the first report of the problem as occurred (such as `tail -f` on UNIX). After the first notification that the problem has occurred is manifested in the system log the user trace can be queried.

- ▶ At what level, either *normal* or *debug*, should the trace run. To know the best option for various situations requires some practice interpreting the generated trace file. A guideline is that tracing at a normal level shows overall activity. The debug level shows specific variables or conditions that are accessed and their values or results.

Tip: WebSphere Message Broker generates only limited information automatically (through the event message process). Thus, it is reasonable to allow the execution group to be traced at the normal level throughout the life span of the running process. Using this technique allows the administrator to poll the cyclical trace on an ad-hoc basis to better understand what type and level of activity is taking place within the execution group.

Errors that occur in production can sometimes be difficult to reproduce because they rely on very specific conditions. If an error does occur in production, having minor tracing active can be beneficial. The root cause of a problem might not be discovered by tracing at the normal level; however, it will most likely isolate the issue to the point that you can reproduce the error condition more rapidly.

Any level of tracing consumes processing resources and reduces throughput. Thus, you should perform testing in an isolated environment to determine if the overhead that tracing creates at the normal level is acceptable. You should perform this evaluation before making your decision on using this technique in production.

Example 9-31 shows tracing at the normal level. The user trace is reset with the **-r** flag for the execution group **default** on **BROKER2** (additionally, you can use the **-c** option to specify the size the trace log).

Example 9-31 User trace at the normal level

```
wmb@m106910f:/var/mqsi $ mqsichangetrace BROKER2 -u -e default -l
normal -r

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $ mqsireadlog BROKER2 -u -e default -f -o
default.xml
BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ mqsiformatlog -i default.xml -o default.trc

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
```

Example 9-32 shows that while the trace was running a configuration change was initiated and that it was completed successfully.

Example 9-32 Results of user trace at the normal level

```
wmb@m106910f:/var/mqsi $ cat default.trc
```

Timestamps are formatted in local time, 240 minutes before GMT.
Trace written by version 6001; formatter version 6001

```
2006-06-14 12:23:42.132965      5655  UserTrace  BIP4040I: The
Execution Group 'default' has processed a configuration message
successfully.
```

A configuration message has been processed successfully. Any configuration changes have been made and stored persistently.

No user action required.

```
2006-06-14 12:23:42.133575      5655  UserTrace  BIP2638I: The MQ
output node 'ConfigurationMessageFlow.outputNode' attempted to write a
message to queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' connected to
queue manager 'QM_BROKER2'. The MQCC was '0' and the MQRC was '0'.
```

```
2006-06-14 12:23:42.133598      5655  UserTrace  BIP2622I: Message
successfully output by output node
'ConfigurationMessageFlow.outputNode' to queue
'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager 'QM_BROKER2'.
```

Threads encountered in this trace:
5655

Having a prior understanding of WebSphere MQ applications makes the output of traces at the normal level very intuitive and takes little analysis to determine its meaning. For example, the keywords MQCC and MQRC indicate a WebSphere MQ *Completion Code* and *Reason Code* respectively. The formatted trace in Example 9-33 shows a very simple configuration problem that is caused by a queue name that is used by an OutputNode that is not resolved by the broker's queue manager.

Example 9-33 Unresolved queue name in user trace at the normal level

```
wmb@m106910f:/var/mqsi $ tail -n 13 default.trc
```

```
2006-06-14 12:46:50.646915      5398  Error      BIP2623E: Unable to
open queue 'TESTOUT' on WebSphere MQ queue manager '': completion
code 2; reason code 2085.
```

A message flow node failed to open the indicated WebSphere MQ message queue. The error codes relate to the MQOPEN call.

Check the WebSphere MQ completion and reason codes in the WebSphere MQ Application Programming Reference manual to establish the cause of the error, taking any appropriate action. It may be necessary to restart the message broker after you have performed this recovery action. If the open failed because the queue manager or queue did not exist, define these objects to WebSphere MQ. If the failure of the open was because incorrect object names were specified, correct the message flow configuration and attempt to redeploy the message broker.

2006-06-14 12:46:51.667243 5398 Error BIP2648E: Message backed out to a queue; node 'MessageFlow1.MQInput'.

Node 'MessageFlow1.MQInput' has received a message which has previously been backed out one or more times because of a processing error in the main path of the message flow. The failure terminal is not attached, so the message broker is putting the message directly to the requeue or dead letter backout queue associated with this node. The MQMD 'backoutCount' of the message now equals the 'backoutThreshold' defined for the WebSphere MQ input queue.

Examine previous messages and the message flow to determine why the message is being backed out. Correct this situation if possible. Perform any local error recovery processing required.

2006-06-14 12:46:51.668430 5398 UserTrace BIP2638I: The MQ output node 'MessageFlow1.MQInput' attempted to write a message to queue 'DLQ' connected to queue manager 'QM_BROKER2'. The MQCC was '0' and the MQRC was '0'.

2006-06-14 12:46:51.668487 5398 UserTrace BIP2615I: The WebSphere MQ input node 'MessageFlow1.MQInput' has backed out the message to the backout requeue or the dead letter queue.

Message backout processing has been invoked, and the message has either been backed out by being written to the backout requeue or dead letter queue, as determined by the WebSphere MQ queue manager and queue configuration.

No user action required.

Notice that the text in Example 9-33 describes exactly what error occurred and what took place to resolve the error automatically. In this example, the message could not be placed on the output queue because it does not exist.

Because the input queue did not have a backout threshold defined and there was no failure terminal defined, the broker placed the message on the queue managers Dead Letter queue automatically. Because the normal trace was running, determining what took place because of the error is discovered easily. The text states that the original message is on the Dead Letter queue.

In contrast, the event messages in Example 9-34 were extracted from the user.log. These messages show that there was a problem, but they do not contain the full explanation. Namely, these messages do not indicate to where the original message was backed out. The problem was solved by configuring the queue manager correctly by creating the output queue name requested by the message flow.

Example 9-34 Unresolved queue name in user.log

```
Jun 14 12:46:50 m106910f user:err|error WebSphere Broker v6001[30682]:
(BROKER2.default)[5398]BIP2623E: Unable to open queue '
TESTOUT' on WebSphere MQ queue manager '': completion code 2; reason
code 2085. : BROKER2.bf8ce7b9-0b01-0000-0080-f393237b0c08
: /build/S600_P/src/DataFlowEngine/ImbMqManager.cpp: 600:
ImbMqConnection::getQueue: ComIbmMQConnectionManager: ComIbmMQConne
ctionManager
Jun 14 12:46:51 m106910f user:err|error WebSphere Broker v6001[30682]:
(BROKER2.default)[5398]BIP2648E: Message backed out to
a queue; node 'MessageFlow1.MQInput'. :
BROKER2.bf8ce7b9-0b01-0000-0080-f393237b0c08:
/build/S600_P/src/DataFlowEngine/ImbMqIn
putNode.cpp: 1859: ImbCommonInputNode::eligibleForBackout:
ComIbmMQInputNode: MessageFlow1#FCMComposite_1_1
Jun 14 13:14:41 m106910f user:info WebSphere Broker v6001[30682]:
(BROKER2.default)[1]BIP2204I: Execution group using process
'30682' thread '1' stopped. :
BROKER2.bf8ce7b9-0b01-0000-0080-f393237b0c08:
/build/S600_P/src/DataFlowEngine/ImbMain.cpp: 886:
main: :
```

Unfortunately, some error conditions are not determined as easily as these simple configuration problems. For example, in many cases, WebSphere Message Broker is used to route messages intelligently to a variety of output queues based on content.

A scenario might be where a message arrives that contains the *valid* content that causes processing to begin; however, the routing logic did not produce the expected results. Ideally, these types of scenarios would be discovered during design and development time. In this kind of event, the analysis in production

with deploying debugging code (such as trace nodes) or attaching the debugger might not be realistic. Fortunately, user tracing at the debug level should allow these types of problems to be studied with very minimal impact.

Example 9-35 illustrates the user trace at the debug level, including the conditions found and the value of variables used. The trace attributes were changed by performing a reset and specifying a level of debug by executing the following command:

```
mqsichangetrace BROKER2 -u -e default -l debug -r
```

After the test message was put, post-processing was the same as with the user trace at the normal level.

Example 9-35 User trace at the debug level

```
wmb@m106910f:/var/mqsi $ tail default.trc
```

```
2006-06-14 13:15:03.658636      5398  UserTrace  BIP2537I: Node
'MessageFlow1.Compute': Executing statement  ''SET OutputRoot =
InputRoot;'' at ('.MessageFlow1_Compute.CopyEntireMessage', '2.3').
2006-06-14 13:15:03.658686      5398  UserTrace  BIP2539I: Node
'MessageFlow1.Compute': Evaluating expression ''InputRoot'' at
('.MessageFlow1_Compute.CopyEntireMessage', '2.20'). This resolved to
''InputRoot''. The result was ''ROW... Root Element Type=16777216
Namespace='' Name='Root' Value=NULL''.
2006-06-14 13:15:03.658736      5398  UserTrace  BIP2568I: Node
'MessageFlow1.Compute': Copying sub-tree from ''InputRoot'' to
''OutputRoot''.
2006-06-14 13:15:03.658819      5398  UserTrace  BIP2537I: Node
'MessageFlow1.Compute': Executing statement  ''RETURN TRUE;'' at
('.MessageFlow1_Compute.Main', '5.3').
2006-06-14 13:15:03.658884      5398  UserTrace  BIP4007I: Message
propagated to 'out' terminal of node 'MessageFlow1.Compute'.
2006-06-14 13:15:03.659092      5398  UserTrace  BIP2638I: The MQ
output node 'MessageFlow1.MQOutput' attempted to write a message to
queue ''TESTOUT'' connected to queue manager ''. The MQCC was '0' and
the MQRC was '0'.
2006-06-14 13:15:03.659109      5398  UserTrace  BIP2622I: Message
successfully output by output node 'MessageFlow1.MQOutput' to queue
''TESTOUT'' on queue manager ''.
```

Threads encountered in this trace:

```
5398 5655
```

Utilizing service tracing

It is unlikely that you will use the service trace on a routine basis. Typically, service tracing is done only at the request of IBM Support after user trace attempts to find a root cause for a problem have failed. Because service tracing is different from the procedures used under MustGather, we review them here for clarity.

You can run the service trace procedure through the **mqsichangetrace**, **mqsireadlog**, and **mqsiformatlog** commands similarly to the user trace; however there are a few key differences. Primarily, you can use the service trace procedure to trace components other than an execution group, such as a broker, Configuration Manager, or User Name Server. Example 9-36 shows services trace for User Name Server.

Example 9-36 Service trace for User Name Server

```
wmb@m106910f:/var/mqsi $ mqsichangetrace UserNameServer -t -b
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

```
wmb@m106910f:/var/mqsi $ mqsireadlog UserNameServer -t -f -b agent -o  
UserNameServer.agent.xml
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

```
wmb@m106910f:/var/mqsi $ mqsiformatlog -i UserNameServer.agent.xml -o  
UserNameServer.agent.trc
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

Utilizing Configuration Manager Proxy API tracing

The Configuration Manager Proxy API provides a powerful tool for developing custom applications to manage and to monitor WebSphere Message Broker. A complete solution should include the ability to enable the proxy tracing when needed.

Normally, you need to enable a trace of this type only at the request of IBM Support. Sometimes, it is also useful to enable the trace when applications that use the proxy are experiencing unexpected results do to misuse or misconfiguration of the API.

Example 9-37 shows a functioning application where the trace is enabled to capture the interaction between the client and the Configuration Manager during a connect and disconnect.

Example 9-37 ConfigurationManagerProxyApiTestHarness.java

```
import com.ibm.broker.config.proxy.ConfigManagerProxy;
import com.ibm.broker.config.proxy.ConfigManagerProxyLoggedException;
import com.ibm.broker.config.proxy.MQConfigManagerConnectionParameters;

public class ConfigurationManagerProxyApiTestHarness {

    public static void main(String[] args) {
        final String trc = "cmpapi.trc";
        final String host = "k116573";
        final int port = 1414;
        final MQCMCP mqcmcp = new MQCMCP(host, 1414, null);
        ConfigManagerProxy cmp = null;
        try {
            ConfigManagerProxy.enableConfigManagerProxyTracing(trc);
            cmp = ConfigManagerProxy.getInstance(mqcmcp);
        } catch (ConfigManagerProxyLoggedException e) {
            e.printStackTrace();
        } finally {
            if (cmp != null) {
                cmp.disconnect();
            }
            ConfigManagerProxy.disableConfigManagerProxyTracing();
        }
    }

    static class MQCMCP extends MQConfigManagerConnectionParameters {
        public MQCMCP(String hostname, int port, String qmgr) {
            super(hostname, port, qmgr);
            this.disableDomainAwareness(); // helps limit trace output
        }
    }
}
```

Important: In Example 9-37, the use of the method `ConfigManagerProxy.enableConfigManagerProxyTracing(String filename)` enables Configuration Manager Proxy API service tracing. This method is *not* equivalent to `start SystemTrace()`. The former method provides only the interactions between the client and the Configuration Manager. The latter enables the tracing of the Configuration Manager comparative to executing the following command:

```
mqsischangetrace CFGMGR -t -b -l normal
```

Providing the ability to enable the Configuration Manager service trace might be a useful function of a custom application, but it should be done with caution because it causes more overhead for the component.

Executing the application above produce trace results in the `cmpapi.trc` file in the local directory. Example 9-38, Example 9-39, and Example 9-40 show select example entries from the trace to illustrate what information can be useful to a developer who uses the Configuration Manager Proxy API.

Example 9-38 Determining the version of Configuration Manager Proxy API

```
Configuration Manager Proxy SCCS Version =  
Config/com/ibm/broker/config/proxy/ConfigManagerProxy.java,  
Config.Proxy, S000, S000-L50831.3 1.86
```

Example 9-39 Determining properties that are used by Configuration Manager Proxy API

```
ip=kl16573, port=1414, qmgr=null, mqseClassname=null, mqseURL=null,  
channelName=SYSTEM.BKR.CONFIG, maxRetries=3,  
requestQueueName=SYSTEM.BROKER.CONFIG.QUEUE,  
responseQueueName=SYSTEM.BROKER.CONFIG.REPLY, retryWaitMillis=3000,  
sessionID=null, userId=wmb@klchl5w, deployID=null
```

Example 9-40 Successfully opening request and reply queues

```
MQSender successfully opened queue 'SYSTEM.BROKER.CONFIG.QUEUE' with  
open options 8208  
MQReceiver successfully opened queue 'SYSTEM.BROKER.CONFIG.REPLY' with  
open options 8193
```

For illustrative purposes in our scenario, after we ran the `ConfigurationManagerProxyApiTestHarness.java` application successfully, we stopped the queue manager that was supporting the Configuration Manager to

show the results of a failure to connect, which provided several interesting results.

First, the stack trace of the connection attempt was written to the console (Example 9-41).

Example 9-41 Stack trace of failed connection

```
com.ibm.broker.config.proxy.ConfigManagerProxyLoggedMQException: The
queue manager 'null' is not running, or there is no related listener
running on port 1414 (MQ reason code 2059 while trying to connect)
    at
com.ibm.broker.config.proxy.MQConnectionHelper.connectToMQ(MQConnection
Helper.java:391)
    at com.ibm.broker.config.proxy.MQSender.send(MQSender.java:300)
    at
com.ibm.broker.config.proxy.SendManager.send(SendManager.java:163)
    at
com.ibm.broker.config.proxy.AdministeredObjectPool.registerWithConfigMa
nager(AdministeredObjectPool.java:1686)
    at
com.ibm.broker.config.proxy.AdministeredObjectPool.registerAdministered
Object(AdministeredObjectPool.java:1487)
    at
com.ibm.broker.config.proxy.ConfigManagerProxy.<init>(ConfigManagerProx
y.java:224)
    at
com.ibm.broker.config.proxy.ConfigManagerProxy.getInstance(ConfigManag
erProxy.java:291)
    at
ConfigurationManagerProxyApiTestHarness.main(ConfigurationManagerProxyA
piTestHarness.java:16)
```

Without interrogating the Configuration Manager Proxy API trace, determining what caused the failure did not take much analysis. The first line indicates that the underlying WebSphere MQ for Java transport could not establish a connection with the queue manager, causing a reason code of 2059, otherwise known as MQRC_Q_MGR_NOT_AVAILABLE. However, after taking a look at the cmpapi.trc file that was created, although a connection was never establish, shows some additional interesting details. While these details were not exceptionally beneficial for this error condition, it is discussed to illustrate that further information is captured by the proxy layer though never reported to the higher level API user.

For example, the `disconnect()` method on the `ConfigManagerProxy` class does not throw checked exceptions such as `ConfigManagerProxyLoggedException`. The detail trace shows that in fact errors were encountered, but a design decision was made by the Configuration Manager Proxy API developer to not report it to the using application. In the `cmpapi.trc`, this message was found within the disconnect sequence as shown in Example 9-42.

Example 9-42 Failure reported during disconnect sequence

w: MQSender could not connect to the MQ Queue Manager; the Configuration Manager Proxy caught MQException reason code 2059

Utilizing JVM tracing

There are several functions within WebSphere Message Broker that use Java including:

- ▶ Java User plug-in nodes
- ▶ Pub/Sub functionality including Pub/Sub nodes
- ▶ XSLT node
- ▶ HTTPRequest node
- ▶ Realtime node

In the event of issues with any of these components or the underlying JVM that is created to execute these functions, IBM Support can request tracing or debugging. Also, user developed applications that use Java might be able to diagnose runtime problems through some of the techniques presented in this section.

There are two significant ways that parameters can be used during development, testing, or production to help determine the root cause for a problem — setting operating system environment variables or passing additional attributes to the `mqsichangeproperties` command.

Setting environment variables using MQSIJVERBOSE

You can set two environment variables to the operating system before starting the broker debug process (as shown in Example 9-43).

Example 9-43 Setting the environment variables

```
export MQSI_RUN_ATTACHED=1
export MQSIJVERBOSE=-verbose:class,gc,jni
mqsistart <broker_name>
```

The JVM verbose options cause JVM information to be written for class and object garbage collection and also write information for JNI events. Because this

information is directed at SYSOUT, you need to set the MQSI_RUN_ATTACHED environment so that this information is written to the console.

If used on Windows, then you need to edit the DataFlowEngine service so that it runs under the system user ID and so that it can output information to a window. You can set these by following these steps:

1. Go to the **Services** panel. Right-click and select **Properties**.
2. Click the **Log on** tab. Select **Local System Account** and **Allow service to interact with desktop**.

Because MQSIJVERBOSE is an environment variable, it applies to every broker that is started in the environment on which it is set. Therefore, if it should only apply to one broker, then you need to set it just before that broker is started and apply it to every execution group for that broker.

Example 9-44 shows these commands executed on an AIX environment that houses BROKER2.

Example 9-44 Using MQSIJVERBOSE on AIX

```
wmb@m106910f:/var/mqsi $ export MQSI_RUN_ATTACHED=1
wmb@m106910f:/var/mqsi $ export MQSIJVERBOSE=-verbose:class,gc,jni
wmb@m106910f:/var/mqsi $ mqsistop BROKER2
```

```
BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ mqsistart BROKER2
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

```
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ Redirecting stdout to
/var/mqsi/components/BROKER2/1d1220d8-0b01-0000-0080-b59c0566a828//stdo
ut
Redirecting stderr to
/var/mqsi/components/BROKER2/1d1220d8-0b01-0000-0080-b59c0566a828//stde
rr
```

In this example, something very different happened than at a normal broker startup. Because the processes are now running in console mode, additional output was shown which states to where standard output and error is directed:

```
/var/mqsi/components/BROKER2/1d1220d8-0b01-0000-0080-b59c0566a828//stdout  
  
/var/mqsi/components/BROKER2/1d1220d8-0b01-0000-0080-b59c0566a828//stderr
```

You should review these files for additional JVM logging. Further analysis shows that the stdout file did not contain any relevant information, but the stderr file did capture a wealth of information. This file now describes the class loading that took place, the JNI activity and garbage collection results.

Example 9-45 shows a few entries for illustration purposes.

Example 9-45 Example of verbose output a broker JVM

```
wmb@m106910f:/var/mqsi/components/BROKER2/1d1220d8-0b01-0000-0080-b59c0566a828 $ head stderr
```

```
2006-06-15 14:35:06.845804 Execution group started.  
2006-06-15 15:28:40.498856 Execution group started.  
JVMST080: verbose:gc is enabled  
JVMST082: -verbose:gc output will be written to stderr  
[Opened /opt/IBM/mqsi/6.0/jre/lib/core.jar in 2 ms]  
[Opened /opt/IBM/mqsi/6.0/jre/lib/graphics.jar in 100 ms]  
[Opened /opt/IBM/mqsi/6.0/jre/lib/security.jar in 0 ms]  
[Opened /opt/IBM/mqsi/6.0/jre/lib/server.jar in 1 ms]  
[Opened /opt/IBM/mqsi/6.0/jre/lib/xml.jar in 2 ms]  
[Opened /opt/IBM/mqsi/6.0/jre/lib/charsets.jar in 20 ms]
```

Any support personnel with previous knowledge or experience debugging applications usage of a JVM will find this output very familiar.

Using mqsichangeproperties

An alternative technique to using environment variables that has some benefits is to use the **mqsichangeproperties** command. Using the **mqsichangeproperties** command has the advantage that you can change these attributes for a specific execution group using the **-e** parameter. You can change all execution groups for a broker by omitting the **-e** parameter. For example, the commands in Example 9-46 trace the JVM usage under only the execution group called default. Before performing these steps, the procedure using environment variables was undone by exiting and re-entering the shell, and then restarting BROKER2.

Example 9-46 Using mqsichangetrace to set JVM properties

```
wmb@m106910f:/var/mqsi $ mqsichangeproperties BROKER2 -e default -o
ComIbmJVMMManager -n jvmVerboseOption -v all

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ mqsistop BROKER2

BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ mqsistart BROKER2

WebSphere MQ queue manager running.
BIP8096I: Successful command initiation, check the system log to ensure
that the component started without problem and that it continues to run
without problem.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ mqsireadlog BROKER2 -t -e default -f -o
default.jvm.xml
BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ mqsiformatlog -i default.jvm.xml -o
default.jvm.txt
BIP8071I: Successful command completion.
wmb@m106910f:/var/mqsi $
wmb@m106910f:/var/mqsi $ head default.jvm.txt

Timestamps are formatted in local time, 240 minutes before GMT.
Trace written by version 6001; formatter version 6001

2006-06-15 15:59:01.136510      1 handleJVMOutput , 'JVM output',
'[Opened /opt/IBM/mqsi/6.0/jre/lib/core.jar in 2 ms]
'
2006-06-15 15:59:01.140135      1 handleJVMOutput , 'JVM output',
'[Opened /opt/IBM/mqsi/6.0/jre/lib/graphics.jar in 4 ms]
'
2006-06-15 15:59:01.140552      1 handleJVMOutput , 'JVM output',
'[Opened /opt/IBM/mqsi/6.0/jre/lib/security.jar in 0 ms]
'
2006-06-15 15:59:01.140903      1 handleJVMOutput , 'JVM output',
'[Opened /opt/IBM/mqsi/6.0/jre/lib/server.jar in 0 ms]
```

Specific diagnostic techniques that are available for the IBM JRE are beyond the scope of this book. However, you can find comprehensive information in the *Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2*, which is available at:

<http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/diagnosis/diag142.pdf>

9.2.2 Analyzing abends and cores

If a dump or abend occurs on your system, an error message is produced.

- ▶ On Windows

BIP2111 error message (message broker internal error). The error message contains the path to the MiniDump file in your errors directory.

- ▶ On UNIX

BIP2060 error message (execution group terminated unexpectedly). Look in the directory where the broker was started or in the service user ID's home directory to find the core dump file.

Abend files

When a process ends abnormally (an abend or dump), an entry is made in the syslog or the Windows Event log. The BIP message in the syslog contains the location of the abend file. The typical information contained in an abend file is:

- ▶ Broker operating environment details

- Product details
- Operating system details
- Deployment details
- Failure location

- ▶ A stack of the core file

- ▶ Process Environment Variables

Example 9-47 shows a typical abend file structure from an AIX system.

Example 9-47 Sample abend file contents

```
+-----+
|
| First Failure Symptom Report
| =====
|
|
```

Proc start time :- Tue Jun 20 10:34:32 2006

Product Details
+++++

Vendor :- IBM
Product Name :- WebSphere Message Brokers
Program ID :- 5724-J04
Version :- 6001

OS Information
+++++

Operating System :- AIX
Version :- 5
Release :- 3
Node Name :- remwbisvil
Machine ID :- 00C7AD5F4C00

Environment
+++++

Installation Path :- /opt/IBM/mqsi/6.0
Service User ID :- wmb
Work Path :- /var/mqsi
Executable Name :- DataFlowEngine
Process ID :- 237816

Deployment
+++++

Component Name :- BROKER2
Component UUID :- 6ce92169-0a01-0000-0080-96cda04e8604
Queue Manager :- QM_BROKER2
Data Source Name :- MQSITCP
DB User ID :- wmb
Execution Group :- default
EG UUID :- 6de92169-0a01-0000-0080-96cda04e8604
Main EG :- yes
User trace :- 0
Service trace :- 0
Trace size :- 0

Build Information
+++++

```

Backing build      :-
Sandbox           :- /build/S600_P
CMVC Level        :- S600-L60322.2
Build type        :- Production
64 Bit Build      :- no

Failure Location
+++++

Time of Report     :-
Message Flow       :- GENERAL.CodecPIC
Thread ID         :- 0x00000C2

```

```

-----
abend record for pid 237816 tid 3104 time in seconds since 01/01/1970: 1144230668
File: /build/S600_P/src/CommonServices/Unix/ImbAbend.cpp
Line: 888
Function: signal received
---- Inserts ----
6
@(#) 1.33.3.12 CommonServices/Unix/ImbAbend.cpp, CommonServices, S600, S600-L60320.2
06/03/10 12:49:47 [3/20/06 17:49:53]
1035611552
-----
----- Stack dump for current thread ( 3104)
(0x00000000) <invalid code address>
(0xd0126558+0x00000058) _p_raise [/usr/lib/libpthreads.a(shr_xpg5.o)]
(0xd033657c+0x00000030) raise [/usr/lib/libpthreads.a(shr_xpg5.o)]
(0xd03643a0+0x000000b8) abort [/usr/lib/libpthreads.a(shr_xpg5.o)]
(0xd0801458+0x00000000) <no name available> [/usr/lib/libC.a(shrcore.o)]
(0xd7bf7700+0x00000084) __ct__Q2_6ImbLog13FastEntryItemFPC12ImbLogSourcePCcb
[/usr/opt/mqsi/6.0/lib/libCommonServices.a(libCommonServices.a.so)]
(0xd7bf767c+0x00000020) writeEntry__6ImbLogFPC12ImbLogSourcePCc
[/usr/opt/mqsi/6.0/lib/libCommonServices.a(libCommonServices.a.so)]
.....
.....

```

9.2.3 Investigating ODBC errors

Message broker interacts with the databases (the broker database or user database) using ODBC drivers. Thus, when the ODBC errors are received at run time or at component creation time, you first need to verify that the ODBC environment is set up correctly. The following are the typical check points:

- ▶ Ensure that the \$ODBCINI and \$ODBCINI64 environment variables are pointing to the correct ODBC configuration files.
- ▶ The ODBC configuration files to which these environment variables point must comply with the product supplied sample template, which are available in `<product_install_dir>/merant` and `<product_install_dir>/DD64` respectively.
- ▶ Make sure that the data source name stanza is set up correctly in the `odbc.ini` file.
- ▶ Ensure that the library path (`LIBPATH` and `LD_LIBRARY_PATH`) are set correctly in the broker service user environment.

9.2.4 Configuration Manager connection problem

If a user is having trouble connecting to the Configuration Manager from Toolkit, try to determine if there is a MQ connection problems or if is being rejected by the Configuration Manager. You need to look at the error message because it might be self explaining. You also need to determine if requests are getting to `SYSTEM.BROKER.CONFIG.QUEUE`.

If the connection problem is related to an MQ connection, it is likely a setup problem. In this case, you need to verify the setup and, if necessary, take an MQ trace.

If the Configuration Manager is rejecting the connection, ensure that there is a correct ACL entry for the correct user ID. Doing so should allow the user to perform the operation.

If you still cannot resolve the problem, take a Configuration Manager and a Configuration Manager Proxy trace and send them to IBM Support Center for further assistance.

9.2.5 Debugging message flows

You can use the flow debugger in Message Brokers Toolkit to debug the message flows. This is explained further with an example in the article *Debugging message flows using WebSphere Message Broker*, which is available at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0510_tan/0510_tan.html

9.3 Characteristics for message broker components

In this section, we discuss specific details for message broker components in problem determination areas.

9.3.1 Problem determination with the broker component

Problem determination with the broker component typically manifests itself through alerts from monitoring or through errors that are reported by the user. After initial notification of a potential problem is received, problem determination and problem resolution can begin. Most of the techniques that we introduce in this chapter involve runtime diagnostics, such as the event messages, log analysis, and tracing.

When you begin problem determination with the broker component, the absolute first place to look is the Windows Event Viewer or syslog — the master log for the WebSphere Message Broker product. Rarely does a serious situation occur that does not present itself in some way to event log sources.

The broker component is very complex environment. It has dependencies on WebSphere MQ intercommunication, internally used data sources, application or business oriented data sources, and operating system resources. One of the most important tasks an operations team can accomplish to assist with problem determination is to develop a complete inventory of dependencies to the broker environment. This inventory can be transformed easily into a diagnostic checklist.

In addition, encouraging application development teams to document how their application operates, defining service level agreements in detail, and applying operations rigorously can make problem resolution in a production environment more realistic.

9.3.2 Problem determination with the Configuration Manager

Problems with the Configuration Manager component are normally reported by a WebSphere Message Broker developer or administrator and not by users of the broker applications. (You can find details on monitoring the Configuration Manager component in 7.1, “Basic of monitoring and health-checking”.) If you discover communication problems between the Configuration Manager and the brokers in the Configuration Manager’s domain, use the following initial procedures to discover the root cause:

- ▶ Ensure that the Configuration Manager component is running.
- ▶ Interrogate the event log sources, such as the Windows Event Viewer or the UNIX syslog.
- ▶ Verify communication between the broker’s queue managers and the Configuration Manager queue manager through standard WebSphere MQ intercommunication problem determination.

This component has no facilities for user tracing and relies only on service tracing to assist with deep problem determination. Therefore, if you find a problem that you cannot resolve, IBM Support might request that you perform tracing. (For more information, review 9.1.6, “Collecting data for further support” on page 181.)

9.3.3 Problem determination with the User Name Server

As with the Configuration Manager component, problems with the User Name Server component are normally only reported by a WebSphere Message Broker developer or administrator. The difference is that problems can be determined by publish/subscribe applications that detect failures. You can find further details about monitoring in 7.4.3, “User Name Server” on page 150.

If you discover communication problems between the User Name Server and the brokers in the same domain, use the following initial procedures to discover the root cause:

- ▶ Ensure that the User Name Server component is running.
- ▶ Interrogate the event log sources, such as the Windows Event Viewer or the UNIX syslog.
- ▶ Verify that communication between the broker’s queue managers and the User Name Server queue manager through standard WebSphere MQ intercommunication problem determination.

This component has no facilities for user tracing and relies only on service tracing to assist with deep problem determination. Therefore, if you find a

problem that you cannot resolve, IBM Support might request that you perform tracing. (For more information, review 9.1.6, “Collecting data for further support” on page 181.)

9.3.4 Problem determination with the Message Brokers Toolkit

The Message Brokers Toolkit displays two basic types of messages:

- ▶ Pop-up messages that are generated as instant feedback to an action.
- ▶ Messages that result from the running of the product and that are displayed in a log or a view and that are:
 - The results of a deploy operation in the Event Log
 - An error produced when a message flow that contains an error is saved

This section discusses the following locations where messages are displayed in the Message Brokers Toolkit:

- ▶ Message Brokers Toolkit Event Log
- ▶ Pop-up messages
- ▶ Alerts view

The most important of these locations is the Message Brokers Toolkit Event Log, because this location is where the results of administration operations on the broker domain are displayed.

Message Brokers Toolkit Event Log

The Event Log editor in the Message Brokers Toolkit is the primary source of messages that relate to deployment actions from the Message Brokers Toolkit to the Configuration Manager and the broker topology. The messages that the Event Log displays are *not* recorded in the Windows Event Viewer. Do not confuse the two logs.

Whenever a deployment action is initiated through the Message Brokers Toolkit, the responses that are received are stored in the Configuration Manager's internal repository. These messages are then displayed in the Event Log editor in the Message Brokers Toolkit.

To access the Event Log, double-click **Event Log** (marked with a flag icon) in the Domains view of the Broker Administration perspective in the Toolkit.

When you select a message that is displayed in the event log, its contents are displayed in the Details section. (These are BIP messages with the same structure as those seen in the Windows Application log and on the command line.) You see a deploy response message for each broker that is involved in a

deploy action in the Event Log. You also see the name of the broker in the Source column under Logs.

Figure 9-4 shows a successful deploy messages for BROKER1.

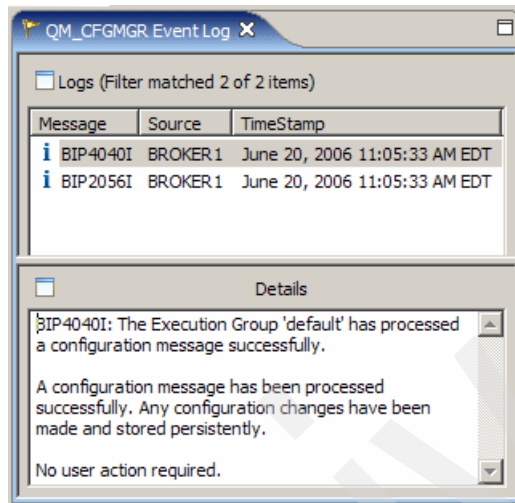


Figure 9-4 Successful deploy message

When the Toolkit Event Log displays an error message, it might be necessary to look for other, related error messages in the Windows Event Viewer or a remote systems local logs to determine the cause of the failure.

The Event Log updates itself automatically as messages , but right-clicking in the Event Log and selecting **Revert** refreshes the information in the log. Using this context menu, you can also filter and clear the log. Using **Clear** removes the messages from the Configuration Manager's repository. Thus, you must use this option only if you do not want to keep the messages. When messages are present in the log, then you can save the log from the context menu as text files.

Pop-up messages

Figure 9-5 is an example of a pop-up message that notifies you of a successful response from the Configuration Manager after a deploy action.

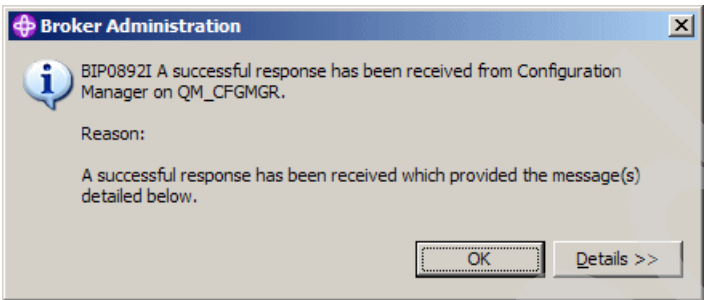


Figure 9-5 Example of pop-up message from deploy action

This message also shows the typical structure of a BIP message with the BIP code followed by a description and then a *reason* section that details why the message is being displayed. In this pop-up message there is also a *details* section that the user can make visible or hidden by clicking **Details**. The details section can provide more information about the message and, in an error message, can contain suggestions about how to correct the problem.

The Message Brokers Toolkit can also display pop-up messages in that are not BIP messages and frequently are some kind of progress or status indicator, such as when messages have been added to a broker archive file.

Alerts view

The Alerts view in the Broker Administration perspective looks similar to the Problems view in the way that it displays messages. However, the messages that it displays are the statuses of runtime components within a broker domain. This information is refreshed regularly so that if connections to the brokers in the domain are running correctly, then any changes to the brokers or execution group states are displayed. Figure 9-6 shows an example of the messages that are displayed in the Alerts view.

	S	Description	Resource	In Domain
	S	Broker is not running	BROKER2	QM_CFGMGR@9.42.171.138:1414
	S	Message Flow is not running	BROKER2/default/http_client	QM_CFGMGR@9.42.171.138:1414

Figure 9-6 Messages in the Alerts view

The Alerts view shows that a deploy operation is in progress (waiting for an answer from the Configuration Manager), BROKER2 is not running, and the execution group and message flows are deployed to it. These messages are displayed because the broker is stopped. If a component is stopped unexpectedly, then this display can alert you to a problem.

Running components are not displayed in this view, because they are displayed in the Domains view. You cannot sort messages in the Alerts view in the same way as the messages in the Problems view (from the Broker Application Development perspective), but you can filter them. The Alerts Filter dialog enables you to select the alert sources. You can display the Alerts Filter dialog by clicking the **Filter** button (the button with the three arrows).

You can clear alert sources, such as individual brokers and execution groups, so that no alerts relating to these are displayed in the Alerts view. Doing so is useful if the Message Brokers Toolkit is connected to more than one broker domain. You can use the Filter Alerts to view alerts from just one domain at a time. Figure 9-7 shows the selectable nodes.

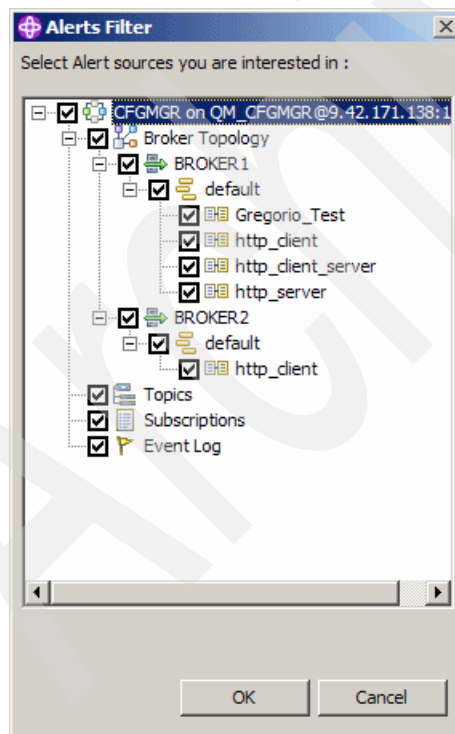


Figure 9-7 Filtering alerts from a broker domain

Scenario

This chapter contains the message broker scenario that we used for this book. The scenario illustrates important message broker managing tasks in practice. First, we describe the message broker environment that the scenario uses. Then, we described the following tasks which the scenario implements:

- ▶ Resource management
 - Operational procedures
 - Change management
- ▶ Security
 - Hyper Text Transfer Protocol Secure support for the broker
 - Connection encryption in the Toolkit
- ▶ Backup, restore, and recover
 - Backing up the broker
 - Restoring the broker
 - Recreating the broker
- ▶ Problem determination
 - Alerts and logs
 - Failing components
 - Tracing
 - Resolving

10.1 Environment setup

In this section, we describe the environment that we use for our message broker scenario.

10.1.1 The logical topology

Figure 10-1 shows the logical topology of the message broker scenario environment.

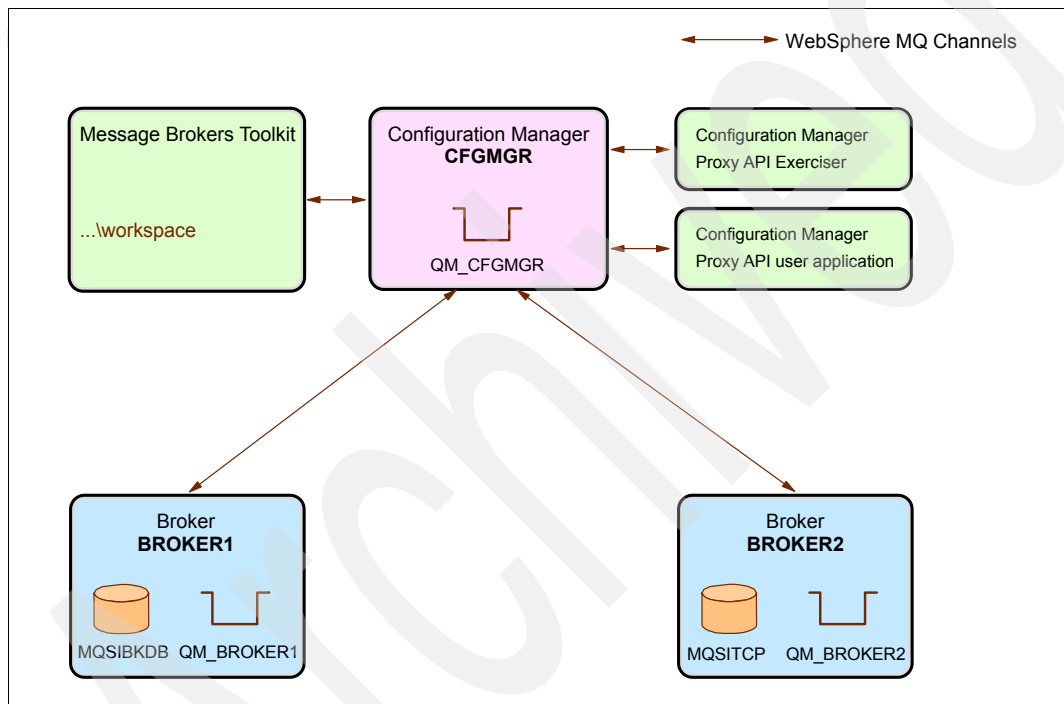


Figure 10-1 The logical topology of message broker scenario environment

The environment consists of one broker domain with the Configuration Manager, the Message Brokers Toolkit, and two brokers in the domain. Table 10-1 lists the components and their required resources.

Table 10-1 The domain components

Component type	Component name	Queue manager	Database
Message Brokers Toolkit	N/A	N/A	N/A
Configuration Manager	CFGMGR	QM_CFGMGR	N/A
Broker	BROKER1	QM_BROKER1	MQSIBKDB ^a
Broker	BROKER2	QM_BROKER2	MQSITCP ^{bb}

- a. There are user databases for message flow interactions on each broker as well. However, we do not document that in this table so that the table and figures are easier to read.
- b. We use the TCP/IP loopback communication instead of shared memory to enable more than 10 sessions from the broker into the database at the same time.

10.1.2 The physical topology

Figure 10-2 shows the physical topology of the message broker scenario environment.

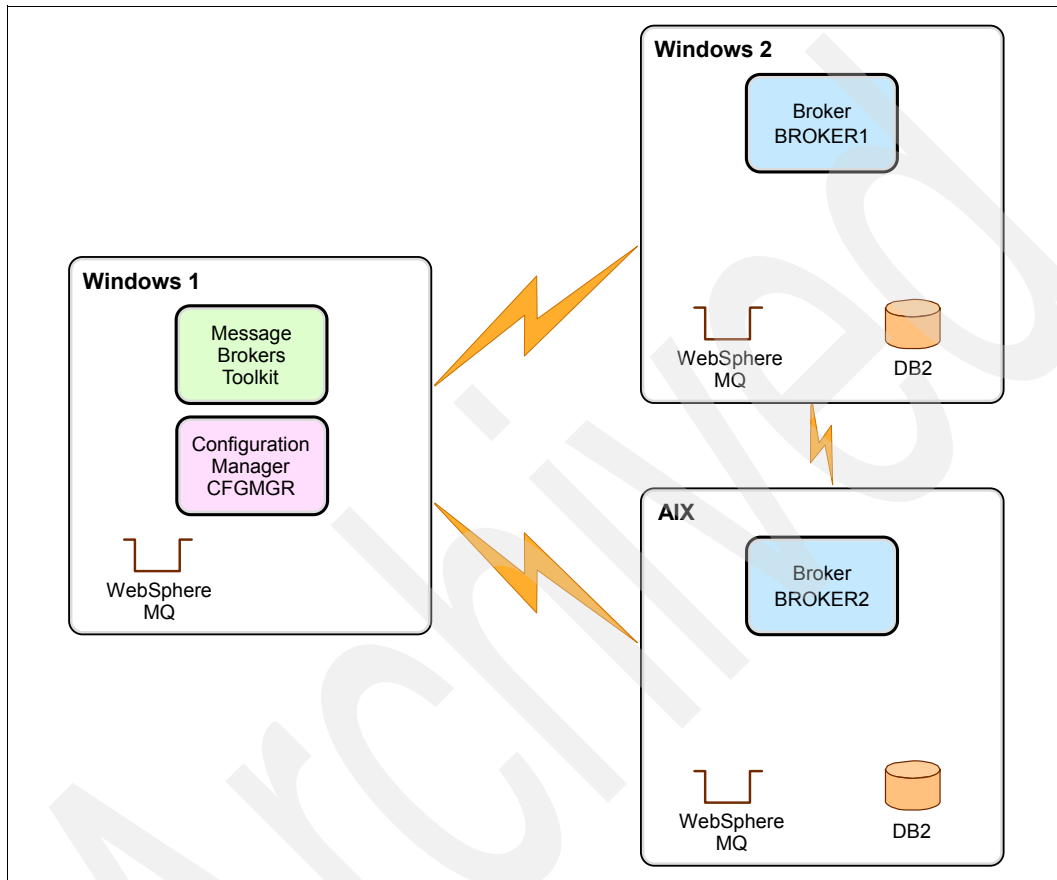


Figure 10-2 The physical topology of message broker scenario environment

The physical environment consists of three machines that run Windows and AIX operating systems. The details of these machines are as follows:

- ▶ The Message Brokers Toolkit and the Configuration Manager are on the first Windows machine
- ▶ One broker is on the second Windows machine
- ▶ One broker is on the AIX machine

Table 10-2, Table 10-3, and Table 10-4 list the software products and their versions that we use on these machines.

Table 10-2 The Windows machine 1

Software	Installed level
Operating system	Microsoft® Windows XP Service Pack 1
WebSphere Message Broker	V6.0 Fix Pack 6.0.0.1
Message Brokers Toolkit	V6.0 Fix Pack 6.0.0.1 Build id 20060106_1130 ^a
WebSphere MQ	V6.0 Fix Pack 6.0.1.1
DB2	N/A

- a. We performed the online updates from the IBM Support Web site using the IBM Rational Product Updater after we installed Fix Pack 6.0.0.1.

Table 10-3 The Windows machine 2

Software	Installed level
Operating system	Microsoft Windows XP Service Pack 1
WebSphere Message Broker	V6.0 Fix Pack 6.0.0.1
WebSphere MQ	V6.0 Fix Pack 6.0.1.1
DB2	V8.2 Fix Pack 5 ^a

- a. DB2 Universal Database V8.2 Fix Pack 5 is equivalent to DB2 Universal Database V8.1 with Fix Pack 12.

Table 10-4 The AIX machine

Software	Installed level
Operating system	V5.3 Technology Level 04 Service Pack 03
WebSphere Message Broker	V6.0 Fix Pack 6.0.0.1
WebSphere MQ	V6.0 Fix Pack 6.0.1.1
DB2	V8.2 Fix Pack 5 ^a

- a. DB2 Universal Database V8.2 Fix Pack 5 is equivalent to DB2 Universal Database V8.1 with Fix Pack 12.

Before you run the scenario that we describe here, we assume that you have:

- ▶ Installed the WebSphere Message Broker V6.0 components according to the *WebSphere Message Broker V6.0 Installation Guide* that is supplied with the product, which is available at:
<http://www.elink.ibm.link.ibm.com/public/applications/publications/cgi-bin/pbi.cgi?CTY=US&FNC=SRX&PBL=GC34-6621>
- ▶ Used the latest available fix pack versions for WebSphere Message Broker V6.0 and WebSphere MQ V6.0.

10.2 Resource management

Resource management presents several opportunities for using repeatable processes and procedures for to manage and to maintain a WebSphere Message Broker environment. The scenario that we describe here shows one possible solution to enhance the change management process.

10.2.1 Change management

This scenario illustrates only a few tasks that you could use to facilitate change management. The recommendation is to review change management policies and guidelines that you use at your location. After this review, you should create processes and procedures for maintaining WebSphere Message Broker that fit within those policies and guidelines.

10.2.2 Creating a repeatable procedure for bar deploys

This scenario illustrates one way to create a repeatable procedure to deploy a bar file. The goal of this scenario is to illustrate that you should create controls for changes in production so that procedures are reliable, repeatable, auditable, and

allow for backing out changes. This scenario applies these goals through a procedure to retrieve a bar file from a build server, to deploy the file to a chosen broker, and to create and to archive the file after a successful deploy.

Scripting ensures that if valid parameters are passed, the execution results will remain the same and, therefore, the results will be reliable and repeatable. In addition, the ability to restore previous versions of bar files from the archive allows for backing out of changes. Finally, during the execution, results are logged to document both positive and negative results, thus creating an audit trail.

Utilizing Ant

In this scenario, we use *Ant* to facilitate scripting of the deployment procedure. Ant is commonly known as a build tool, but it has been used in recent years for software deployments and configuration changes where extensibility is needed. The tooling is extensible because features that are not present can be easily built with Java classes that extend Ant's Task class. Although you can create extensions, you can still access all of the built-in functionality.

This *deploytool* scenario also presents a portable Java solution. As such, it should perform equally as well on any environment with a Java Runtime Environment (JRE). Also, this solution requires that the ConfigManagerProxy.jar file be on the classpath along with the Ant and the new class that we create. This benefit means that the solution is portable with a very small footprint. This solution does not require that the entire Message Brokers Toolkit be installed on the deploy server.

Using Ant or any other Java-based solution is not required for creating a repeatable deployment procedure. Command line utilities are available with the WebSphere Message Broker product. Using these tools would work equally as well wrapped with a scripting language, such as shell on UNIX or Windows batch. The drawback is that you would need to install WebSphere Message Broker, and the specific scripting language would not be platform independent.

This scenario is suitable for sites that have multiple operating systems such as Windows, UNIX, z/OS, and so on for their development, testing, and production environments.

Complete instructions about using Ant are available at:

<http://ant.apache.org>

Complete instructions about creating custom Ant tasks are available at:

<http://ant.apache.org/manual/develop.html#writingowntask>

Ant is delivered with the Message Brokers Toolkit. Thus, you could develop a solution by creating a new Java project to work.

Designing and developing the WmbDeployTask.java

The WebSphere Message Broker installation delivers several examples that use the Configuration Manager Proxy API, which is located in the following directory on the scenarios development server:

`C:\Program Files\IBM\MQS1\6.0\sample\ConfigManagerProxy\cmp`

This directory includes a Java class called `DeployBar.java` that you can use as a reference. For this scenario, we created a new Java class to enhance the `DeployBar.java` sample.

While the overall goal of the sample program remained intact, we made a design enhancement to breakdown the functions into separate methods. Using separate methods and try or catch blocks allows us to track the root cause of a failure more quickly and improves reusability.

Figure 10-3 illustrates the design of the relevant portion of the class. It shows the inner private class and all methods, but the fields are hidden.

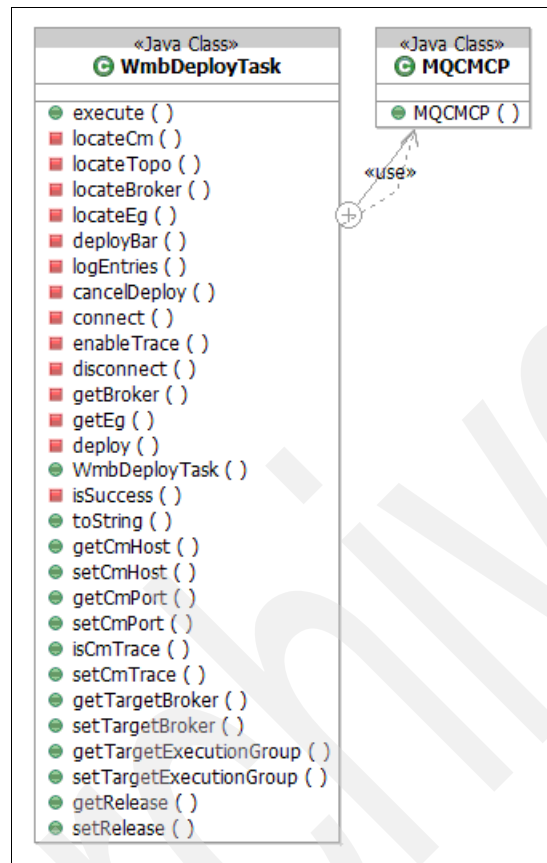


Figure 10-3 WmbDeployTask class diagram

To extend the `org.apache.tools.ant.Task` class and to use `org.apache.tools.ant.BuildException`, the `ant.jar` file had to be located and added to the Java projects build path. In addition, we used multiple classes from the Configuration Manager Proxy API from the `com.ibm.broker.config.proxy` package (such as `BrokerProxy` and `LogEntry`). Therefore, we also added the `ConfigManagerProxy.jar`.

For an overview, see the Package Explorer view in the Message Brokers Toolkit's Java Perspective, as illustrated in Figure 10-4 for the WmbDeployTool project.

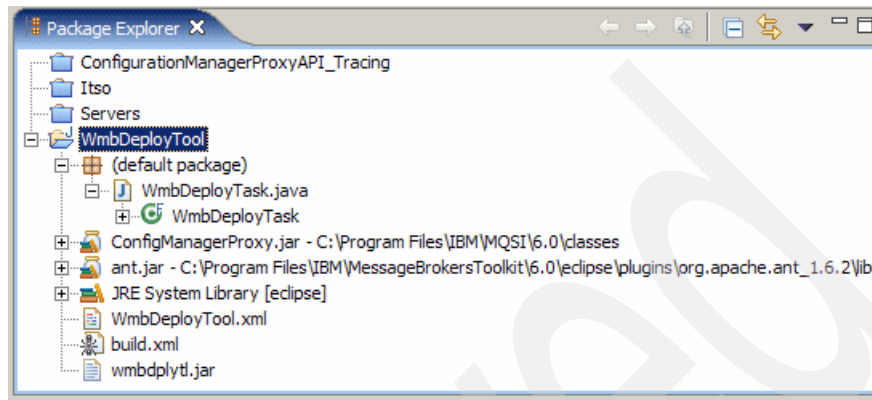


Figure 10-4 Package Explorer view of the WmbDeployTool Project

Tip: While the figure does not show the complete WmbDeployTask.java, the complete source code is available in Appendix C, “Additional material” on page 301 and Example 10-1 includes a snippet of the code.

An aspect of this class that is different than most user applications is that all exceptions are wrapped with Ant's un-checked exception called *BuildException*. Ant conveniently stops processing when this exception is encountered. Because of this feature, calls made to the Configuration Manager Proxy API that generate checked exceptions are wrapped as new exceptions and are re-thrown. This function allows the script to be stopped immediately when problems occur and the details reported.

Example 10-1 WmbDeployTask.java

```
public final class WmbDeployTask extends Task {

    public static final long TIME_TO_WAIT_MS = 30 * 1000;

    public void execute() {
        log(this.toString());
        ConfigManagerProxy cmp = null;
        try {
            cmp = locateCm();
            final TopologyProxy topo = locateTopo(cmp);
            final BrokerProxy broker = locateBroker(topo);
```

```

        final ExecutionGroupProxy eg = locateEg(broker);
        final DeployResult dr = deployBar(eg);
        logEntries(dr);
        // Example assumes anything but 'success' is a 'failure'!
        if (!isSuccess(dr.getCompletionCode())) {
            cancelDeploy(cmp);
            throw new BuildException(notSuccess);
        }
    } finally {
        disconnect(cmp);
    }
}

private ConfigManagerProxy locateCm() {
    try {
        final ConfigManagerProxy cmp = connect();
        if (cmp.hasBeenUpdatedByConfigManager(true)) {
            log("Connected to '" + cmp.getName() + "'.");
            return cmp;
        }
        throw new BuildException(cmError);
    } catch (ConfigManagerProxyLoggedException e) {
        throw new BuildException(cmError, e);
    } catch (ConfigManagerProxyPropertyNotInitializedException e) {
        throw new BuildException("Error locating CM name.", e);
    }
}

private TopologyProxy locateTopo(final ConfigManagerProxy cmp) {
    try {
        final TopologyProxy topo = cmp.getTopology();
        if (topo.getName() != null) {
            String topoName = topo.getName();
            log("Topology '" + topoName + "' has been found.");
            return topo;
        }
        throw new BuildException(topoError);
    } catch (ConfigManagerProxyPropertyNotInitializedException e) {
        throw new BuildException(topoError, e);
    }
}

private BrokerProxy locateBroker(final TopologyProxy topo) {
    try {
        final BrokerProxy bp = getBroker(topo);

```

```

        if (bp.getName() != null) {
            log("Broker '" + bp.getName() + "' has been found.");
            return bp;
        }
        throw new BuildException(brokerError);
    } catch (ConfigManagerProxyPropertyNotInitializedException e) {
        throw new BuildException(brokerError, e);
    }
}

private ExecutionGroupProxy locateEg(final BrokerProxy bp) {
    try {
        final ExecutionGroupProxy eg = getEg(bp);
        if (eg.getName() != null) {
            String egNm = eg.getName();
            log("Execution group '" + egNm + "' has been found.");
            return eg;
        }
        throw new BuildException(egError);
    } catch (ConfigManagerProxyPropertyNotInitializedException e) {
        throw new BuildException(egError, e);
    }
}

private DeployResult deployBar(final ExecutionGroupProxy eg) {
    log("Deploying bar '" + this.release + "...");
    try {
        final DeployResult dr = deploy(eg);
        if (dr == null) {
            String dpError = "Deploy results were not received.";
            throw new BuildException(dpError);
        }
        log("-- CC[" + dr.getCompletionCode() + "] --");
        return dr;
    } catch (ConfigManagerProxyLoggedException e) {
        throw new BuildException("Error during bar deploy!", e);
    } catch (IOException e) {
        throw new BuildException(this.release + " not found.", e);
    }
}

private void logEntries(final DeployResult dr) {
    final Enumeration enum = dr.getLogEntries();
    while (enum.hasMoreElements()) {
        log("-----");
    }
}

```



```

        log(((LogEntry) enum.nextElement()).getDetail());
        log("-----");
    }
}

private void cancelDeploy(final ConfigManagerProxy cmp) {
    log("Cancelling deployment...");
    try {
        cmp.cancelDeployment();
    } catch (ConfigManagerProxyLoggedException e) {
        throw new BuildException("Error during cancellation!", e);
    }
}

/* Remaining source not shown in text example. */

```

Scripts that are created for Ant use XML documents to define the execution steps. This scenario uses a simple XML file to define several attributes, including the URL of the build server where the bar file that is targeted for release is located. In addition, log, archive, and temp directories are also set. Finally, the common parameters for accessing the Configuration Manager, broker, and the targeted execution group are also specified.

Example 10-2 presents the complete XML-based script, `WmbDeployTool.xml`. The example illustrates that the Configuration Manager's server (k116573) is accessed at port 1414. The target broker is `BROKER1`, and the execution group is `testDeploy`. The bar file that is deployed is `testDeploy.bar`.

Example 10-2 WmbDeployTool.xml

```

<project
  name="WmbDeployTool"
  default="deploy">
  <target name="deploy">
    <description>
      This is an example of using a script
      to create a repeatable procedure for
      performing WebSphere Message Broker
      bar file deploys. This script will
      retrieve a targeted release of a bar
      file from the build server, deploy it
      and archive if successful. Logging is
      used to the console and to a
      timestamped file.
    </description>
  </target>
</project>

```

```

<tstamp>
  <format
    property="time"
    pattern="MMddyyyy-hhmm" />
</tstamp>
<taskdef
  name="deploy"
  classname="WmbDeployTask" />
<property
  name="deployName"
  value="testDeploy" />
<property
  name="deployLog"
  value="c:\DeployLogs\${deployName}-${time}.log" />
<property
  name="deployServer"
  value="c:/Documents and
Settings/builder/IBM/wmbt6.0/workspace/Servers" />
<!-- Simulates remote URL of build server. -->
<property
  name="deploySource"
  value="file:${deployServer}/${deployName}.bar" />
<property
  name="deployTemp"
  value="c:\DeployTemp" />
<property
  name="deployRelease"
  value="${deployTemp}\${deployName}.bar" />
<property
  name="deployArchive"
  value="c:\DeployArchive\${deployName}.bar.${time}" />
<echo
  message="Logging will be written to '${deployLog}'. " />
<record
  name="${deployLog}"
  logLevel="verbose" />
<!-- Retrieve the targeted release from the build server. -->
<mkdir dir="${deployTemp}" />
<get
  src="${deploySource}"
  dest="${deployRelease}"
  verbose="true" />
<!-- Deploy the release to the Broker/Execution Group -->
<deploy
  cmHost="k116573"

```

```

        cmPort="1414"
        cmTrace="true"
        targetBroker="BROKER1"
        targetExecutionGroup="${deployName}"
        release="${deployRelease}" />
<!-- Archive the release if the deployment was a success. -->
<echo
    message="Archiving release to '${deployArchive}'..." />
<copy
    file="${deployRelease}"
    toFile="${deployArchive}" />
<delete dir="${deployTemp}" />
<echo message="Done!" />
</target>
</project>

```

You can validate the XML file from the Ant view of the Java perspective, as shown in Figure 10-5.

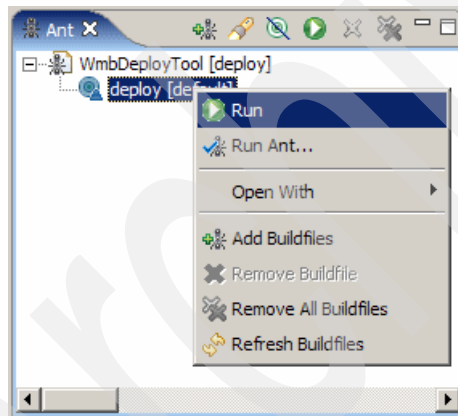


Figure 10-5 Ant view showing WmbDeployTool.xml

Executing deploy procedure

Example 10-3 illustrates a successful deploy. It was executed directly from the Message Brokers Toolkit, but you could easily execute this directly from the command line where Ant is installed.

Example 10-3 Output of successful deploy

```
Buildfile: C:\Documents and
Settings\builder\IBM\wmbt6.0\workspace\WmbDeployTool\WmbDeployTool.xml
build:
    [echo] Logging will be written to
    'c:\DeployLogs\testDeploy-06212006-0236.log'.
    [mkdir] Created dir: C:\DeployTemp
    [get] Getting: file:/c:/Documents and
    Settings\builder\IBM\wmbt6.0\workspace\Servers/testDeploy.bar
    [get] .
    [deploy] WmbDeployTask:Object{cmHost=k116573, cmPort=1414,
    cmTrace=true, targetBroker=BROKER1, targetExecutionGroup=testDeploy,
    release=c:\DeployTemp\testDeploy.bar}
    [deploy] Traces will be written to cmpapi.trc.
    [deploy] Connected to 'CFGMR'.
    [deploy] Topology '' has been found.
    [deploy] Broker 'BROKER1' has been found.
    [deploy] Execution group 'testDeploy' has been found.
    [deploy] Deploying bar 'c:\DeployTemp\testDeploy.bar'...
    [deploy] -- CC[success] --
    [deploy] -----
    [deploy] BIP1520I: The Configuration Manager has initiated a
    deployment operation.
    [deploy]
    [deploy] The Configuration Manager received a request to deploy
    configuration data and has consequently asked the following brokers to
    change their configuration: BROKER1
    [deploy]
    [deploy] The receipt of this message does not necessarily mean that
    deployment was successful; view the Event Log Editor in the Message
    Brokers Toolkit to check the outcome of the deployment. If you are
    deploying programmatically using the Config Manager Proxy, check the
    returned DeployResult object or LogProxy. There will be a separate set
    of log messages for each broker.
    [deploy] -----
    [deploy] Disconnecting...
    [echo] Archiving release to
    'c:\DeployArchive\testDeploy.bar.06212006-0236'...
    [copy] Copying 1 file to C:\DeployArchive
```

```
[delete] Deleting directory C:\DeployTemp
[echo] Done!
BUILD SUCCESSFUL
Total time: 14 seconds
```

Notice in Example 10-4 that the BIP event message that is generated is exactly the same as those returned when using the deploy feature of the Message Brokers Toolkit directly. If you encounter problems, you can diagnose them in the same way as any deploy problem. For assistance with problem determination, refer to Chapter 9, “Problem determination” on page 163.

For example purposes, we executed the script without a valid host name for the Configuration Manager that we used. The DeployLogs directory contained the log file that included significant details about the problem. Primarily, it contains the full stack trace of the exception thrown by the Configuration Manager Proxy API and wrapped as a BuildException by WmbDeployTask. Example 10-4 is a snippet from the stack trace. The root error and where it was first encountered by the sample application is identified.

Example 10-4 Stack trace from failed deploy

```
Caused by:
com.ibm.broker.config.proxy.ConfigManagerProxyLoggedMQException: The
queue manager 'null' is not running, or there is no related listener
running on port 1414 (MQ reason code 2059 while trying to connect)
    at
com.ibm.broker.config.proxy.MQConnectionHelper.connectToMQ(MQConnection
Helper.java:391)
    at com.ibm.broker.config.proxy.MQSender.send(MQSender.java:300)
    at
com.ibm.broker.config.proxy.SendManager.send(SendManager.java:163)
    at
com.ibm.broker.config.proxy.AdministeredObjectPool.registerWithConfigMa
nager(AdministeredObjectPool.java:1686)
    at
com.ibm.broker.config.proxy.AdministeredObjectPool.registerAdministered
Object(AdministeredObjectPool.java:1487)
    at
com.ibm.broker.config.proxy.ConfigManagerProxy.<init>(ConfigManagerProx
y.java:224)
    at
com.ibm.broker.config.proxy.ConfigManagerProxy.getInstance(ConfigManag
erProxy.java:291)
    at WmbDeployTask.connect(WmbDeployTask.java:143)
    at WmbDeployTask.locateCm(WmbDeployTask.java:44)
```

This version of the tool bypasses any additional processing when it encounters a completion code of anything other than a success or an exception. You could be made inject different processing based on various completion codes. For a description of all the completion code possibilities and their meaning, refer to *CompletionCodeType*, which is available at:

<http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/topic/com.ibm.etools.mft.doc/com.ibm/broker/config/proxy/CompletionCodeType.html>

10.3 Security

Our security scenario covers mainly the aspect of SSL encryption for WebSphere Message Broker security configuration. To illustrate the implementation of this aspect in this section, we provide two approaches to this implementation:

- ▶ The implementation of SSL in the HTTP server of the WebSphere Message Broker
- ▶ The implementation of SSL between the Message Brokers Toolkit and the Configuration Manager as a model for implementing SSL between WebSphere MQ Java clients with a message broker

Before presenting this information, we describe the configuration of our Certification Authority as one of the fundamental components of any SSL implementation.

10.3.1 Configuration of the Certificate Authority

Before implementing our security scenario, we have to set up all the environment related with a SSL scenario. Basically, we install and configure an internal Certificate Authority (CA) to support the role of validation of all the certificates that are emitted in our scenario.

It is very important to know that there are many options to get X509 Server Certificates:

- ▶ Requesting the certification from a public root CA (Verisign and others)
- ▶ Requesting the certification from a private CA. Doing so implies that you must administer your own CA inside your intranet. Your CA can be a root CA or a subsidiary CA. A certificate from a public root CA is required for using a root CA in order to adhere to the standards and an internet connection is required in order to use a subsidiary CA.

Some options that you can use to set up an internal CA are:

- A public CA
 - Microsoft Windows Certificate Services
 - OpenSSL
 - OS/400® DCM
 - z/OS RACF®
- For simplicity of management, installing your own internal CA and choosing to use OpenSSL for Windows. General documentation and information for OpenSSL is available at:

<http://www.openssl.org>.

Configuration of the CA using OpenSSL

The steps to set up the OpenSSL Certificate Authority are:

1. Go to the c:\openssl\bin directory and create the root certificate, as shown in Example 10-5.

Example 10-5 Creation of root certificate

```
E:\OpenSSL\bin>perl ca.pl -newca
CA certificate filename (or enter to create)
```

```
Making CA certificate ...
```

```
Loading 'screen' into random state - done
```

```
Generating a 1024 bit RSA private key
```

```
.....+++++
```

```
.....+++++
```

```
writing new private key to './demoCA/private/cakey.pem'
```

```
Enter PEM pass phrase: itso2006
```

```
Verifying - Enter PEM pass phrase: itso2006
```

```
-----
```

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

```
-----
```

```
Country Name (2 letter code) [AU]:US
```

```
State or Province Name (full name) [Some-State]:North Carolina
```

```
Locality Name (eg, city) []:Raleigh
```

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IBM
```

```
Organizational Unit Name (eg, section) []:ITS0
```

```
Common Name (eg, YOUR name) []:ITS0 Administrator
```

Email Address []:itsoadmin@ibm.com

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []:itso2006

An optional company name []:IBM-ITSO

Using configuration from e:\OpenSSL\bin\openssl.cnf

Loading 'screen' into random state - done

Enter pass phrase for ./demoCA/private/cakey.pem:

Check that the request matches the signature

Signature ok

Certificate Details:

Serial Number:

d9:da:ec:6f:03:df:fc:26

Validity

Not Before: Jun 20 14:21:12 2006 GMT

Not After : Jun 19 14:21:12 2009 GMT

Subject:

countryName = US

stateOrProvinceName = North Carolina

organizationName = IBM

organizationalUnitName = ITS0

commonName = ITS0 Administrator

emailAddress = itsoadmin@ibm.com

X509v3 extensions:

X509v3 Subject Key Identifier:

5D:09:D6:1A:02:80:F5:4C:8D:16:9F:BC:FF:F4:46:E5:E3:73:18:43

X509v3 Authority Key Identifier:

keyid:5D:09:D6:1A:02:80:F5:4C:8D:16:9F:BC:FF:F4:46:E5:E3:73:18:43

DirName:/C=US/ST=North Carolina/O=IBM/OU=ITS0/CN=ITS0 Administrator

/emailAddress=itsoadmin@ibm.com

serial:D9:DA:EC:6F:03:DF:FC:26

X509v3 Basic Constraints:

CA:TRUE

Certificate is to be certified until Jun 19 14:21:12 2009 GMT (1095 days)

Write out database with 1 new entries

Data Base Updated

2. Create a self-signed certificate from the CA and the private key, as shown in Example 10-6.

Example 10-6 Creation of self-signed certificate and private key

```
E:\OpenSSL\bin>perl ca.pl -newreq
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
...+++++
.....+++++
writing new private key to 'newkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:North Carolina
Locality Name (eg, city) []:Raleigh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IBM
Organizational Unit Name (eg, section) []:ITSO
Common Name (eg, YOUR name) []:ITSO Administrator
Email Address []:itsoadmin@ibm.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:itso2006
An optional company name []:IBM-ITSO
Request is in newreq.pem, private key is in newkey.pem
```

3. Self sign the certificate, as shown in Example 10-7.

Example 10-7 Self-signing the CA root's certificate

```
E:\OpenSSL\bin>perl ca.pl -sign
Using configuration from e:\OpenSSL\bin\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for ./demoCA/private/akey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
```

```
Serial Number:
d9:da:ec:6f:03:df:fc:27
Validity
Not Before: Jun 20 14:33:55 2006 GMT
Not After : Jun 20 14:33:55 2007 GMT
Subject:
countryName           = US
stateOrProvinceName   = North Caroline
localityName          = Raleigh
organizationName       = IBM
organizationalUnitName = ITS0
commonName            = ITS0 Administrator
emailAddress          = itsoadmin@ibm.com
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
D1:4D:56:DE:B3:35:BC:31:B0:14:18:CC:9A:D6:9C:2D:68:BF:E1:DD
X509v3 Authority Key Identifier:
keyid:5D:09:D6:1A:02:80:F5:4C:8D:16:9F:BC:FF:F4:46:E5:E3:73:18:43
```

Certificate is to be certified until Jun 20 14:33:55 2007 GMT (365 days)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y

Write out database with 1 new entries

Data Base Updated

Signed certificate is in newcert.pem

4. Copy the newcert.pem to ITS0_cert.pem.

5. Copy the newkey.pem to ITS0_key.pem.

10.3.2 Implementing HTTPS support for broker

Implementing SSL encryption in the HTTP server of the WebSphere Message Broker is more important when the interaction in SOA environments demands the most secure connection to invoke third-party Web services or to expose some of the message flows as Web services. In our scenario, we designed two simple message flows that illustrate these two needs:

- ▶ Invoking an external Web service
- ▶ Exposing a message flow as a Web service.

To implement the HTTPS for a broker, you need to:

- ▶ Install the CA certificate in the message broker keystore
- ▶ Modify the message broker properties
- ▶ Modify the HTTP message flows
- ▶ Test the flows

Install the CA certificate in the message broker keystore

After you the set up of the CA and create the root certificate, you need to configure the environment of the message broker. You must install the CA certificate in the JRE, and you must create a new certificate for the HTTP server process. Follow these steps:

1. Export the CA root's certificate as an X509 certificate, using the following command:

`C:\OpenSSL\bin>openssl x509 -inform PEM -outform DER -in demoCA/cacert.pem -out demoCA/cacertX509.der -trustout`
2. Create a keystore in the JRE. The keystore is the place where all the certificates are stored. Example 10-8 shows how to create a keystore.

Example 10-8 Creating the keystore

```
C:\IBM\MQSI\6.0\jre\bin>keytool -genkey -v -keypass itso2006 -alias itso
```

```
Enter keystore password: itso2006
```

```
What is your first and last name?
```

```
[Unknown]: ITSO Administrator
```

```
What is the name of your organizational unit?
```

```
[Unknown]: ITSO
```

```
What is the name of your organization?
```

```
[Unknown]: IBM
```

```
What is the name of your City or Locality?
```

```
[Unknown]: Raleigh
```

```
What is the name of your State or Province?
```

```
[Unknown]: North Carolina
```

```
What is the two-letter country code for this unit?
```

```
[Unknown]: US
```

```
Is CN=ITSO Administrator, OU=ITSO, O=IBM, L=Raleigh, ST=North Carolina, C=US correct?  
(type "yes" or "no") [no]: yes
```

```
Generating 1,024 bit DSA key pair and self-signed certificate (SHA1WithDSA)
```

```
for: CN=ITSO Administrator, OU=ITSO, O=IBM, L=Raleigh, ST=North Carolina, C=US
```

```
[Saving c:\Documents and Settings\patino\keystore]
```

3. Store the CA root's certificate in the cacerts file as a trusted root certificate, as shown in Example 10-9.

Example 10-9 Storing the CA root's certificate in the cacerts file

```
C:\IBM\MQSI\6.0\jre\bin>keytool -import -file cacertX509.der -keystore
..\lib\security\cacerts -alias itsoCA -trustcacerts

Enter keystore password: changeit
Owner: EMAILADDRESS=itsoadmin@ibm.com, CN=ITSO Administrator, OU=ITSO, O=IBM,
      ST=North Carolina, C=US
Issuer: EMAILADDRESS=itsoadmin@ibm.com, CN=ITSO Administrator, OU=ITSO, O=IBM,
      ST=North Carolina, C=US
ST=North Carolina, C=US
Issuer: EMAILADDRESS=itsoadmin@ibm.com, CN=ITSO Administrator, OU=ITSO, O=IBM,
ST=North Carolina, C=US
Serial number: ab7674bd2f1c4672
Valid from: 6/16/06 11:21 AM until: 6/15/09 11:21 AM
Certificate fingerprints:
    MD5:  C5:70:FB:6E:79:CB:8D:48:E7:AA:9A:60:0D:13:ED:85
    SHA1: 8B:2A:0C:9D:03:43:76:0E:19:95:8C:4A:75:72:3C:F1:B3:E2:60:79
Trust this certificate? [no]: yes
Certificate was added to keystore
```

4. Create a certificate request file from the keystore using the following command:

```
C:\IBM\MQSI\6.0\jre\bin>keytool -certreq -alias itso -file
broker1.crs
```

Enter keystore password: itso2006

This certificate request file is used to generate the certificate of the message broker HTTP listener.
5. Approve the certificate request file by the CA and generate the new message broker's certificate, as shown Example 10-10.

Example 10-10 Approval of message broker's certificate by the CA

```
C:\OpenSSL\bin>openssl ca -in broker1.crs -out broker1.pem -keyfile demoCA/private/cakey.pem -cert demoCA/cacert.pem
```

```
Using configuration from C:\OpenSSL\bin\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
```

```
Serial Number:
  ab:76:74:bd:2f:1c:46:74
Validity
  Not Before: Jun 16 19:57:44 2006 GMT
  Not After : Jun 16 19:57:44 2007 GMT
Subject:
  countryName           = US
  stateOrProvinceName   = North Caroline
  organizationName       = IBM
  organizationalUnitName = ITS0
  commonName             = ITS0 Administrator
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    0B:CD:8A:E2:96:4B:25:0E:51:64:AD:72:16:B0:30:64:31:6F:0F:3E
  X509v3 Authority Key Identifier:
    keyid:9D:51:4F:9F:3E:6F:68:DE:E1:FC:D9:7A:C4:65:44:3F:1E:C2:9D:E5
```

Certificate is to be certified until Jun 16 19:57:44 2007 GMT (365 days)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y

Write out database with 1 new entries

Data Base Updated

-
6. Export the message broker's certificate to an X509 format using the following command:

```
C:\OpenSSL\bin>openssl x509 -inform PEM -outform DER -in broker1.pem
-out broker1_X509.der
```

7. Import the message broker's certificate in the keystore, as shown in Example 10-11.

Example 10-11 Importing the certificates message broker in the keystore

```
C:\IBM\MQSI\6.0\jre\bin>keytool -import -alias itso -file broker1_X509.der
-trustcacerts
```

Enter keystore password: itso2006

Certificate reply was installed in keystore

Modifying the message broker properties

Now that you have installed the CA and have stored the root certificate and the broker certificate in the keystore file, you need to change the properties of the message broker configuration to allow SSL communications in the HTTP server. To change the properties of the broker, use the **mqsichangeproperties** command.

To enable the SSL configuration on the message broker in our scenario, we must change at least three properties of the message broker, as described in Table 10-5.

Table 10-5 SSL properties

Object	Name	Value
HTTPListener	enableSSLConnector	true
HTTPSConnector	keystoreFile	c:\Documents and Settings\patino\keystore
HTTPSConnector	keystorePass	itso2006

Example 10-12 shows the execution of the commands to change the properties that are listed in Table 10-5.

Example 10-12 Execution of mqsichangeproperties command

```
C:\IBM\MQSI\6.0\bin> mqsichangeproperties BROKER1 -b httplistener -o HTTPListener -n enableSSLConnector -v true
```

BIP8071I: Successful command completion.

```
C:\IBM\MQSI\6.0\bin> mqsichangeproperties BROKER1 -b httplistener -o HTTPSConnector -n keystoreFile -v "c:\Documents and Settings\patino\keystore"
```

BIP8071I: Successful command completion.

```
C:\IBM\MQSI\6.0\bin> mqsichangeproperties BROKER1 -b httplistener -o HTTPSConnector -n keystorePass -v itso2006
```

BIP8071I: Successful command completion.

Note: For more information and reference about the **mqsichangeproperties** command and the SSL properties, refer to *WebSphere Message Broker V6.0 Infocenter*, which is available at:

<http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp>

Modifying the HTTP message flows

Now, you are ready to implement message flows using the SSL HTTP server configuration. In our scenario, we implement two sample message flows — one that acts as a server and the other that acts as a client (from the HTTP perspective). Figure 10-6 shows the client message flow and represents a sample process where:

- ▶ An input queue receives data from some other process
- ▶ The flow needs to invoke a Web service in a secure way
- ▶ The flow uses the Web service response to send it to an output queue

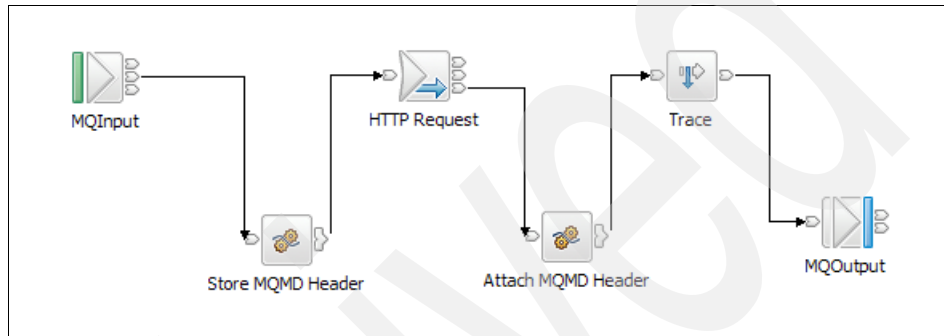


Figure 10-6 Client's sample message flow

Figure 10-7 illustrates the server message flow and represents a simple server flow where:

- ▶ It provides two entry points (one with SSL and the other without SSL).
- ▶ It computes the input from the HTTP input
- ▶ It answers back in a HTTP reply

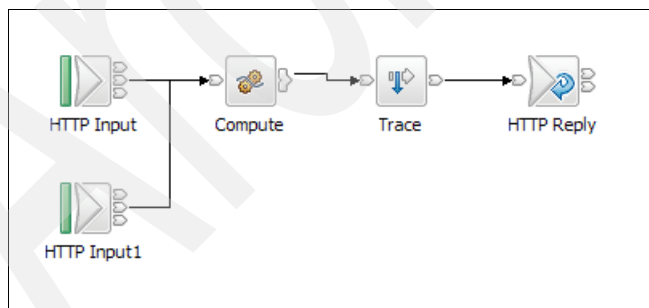


Figure 10-7 Server's sample message flow

To provide SSL encryption in an HTTP request, the Web Service URL contains the correct information, as shown in Figure 10-8.

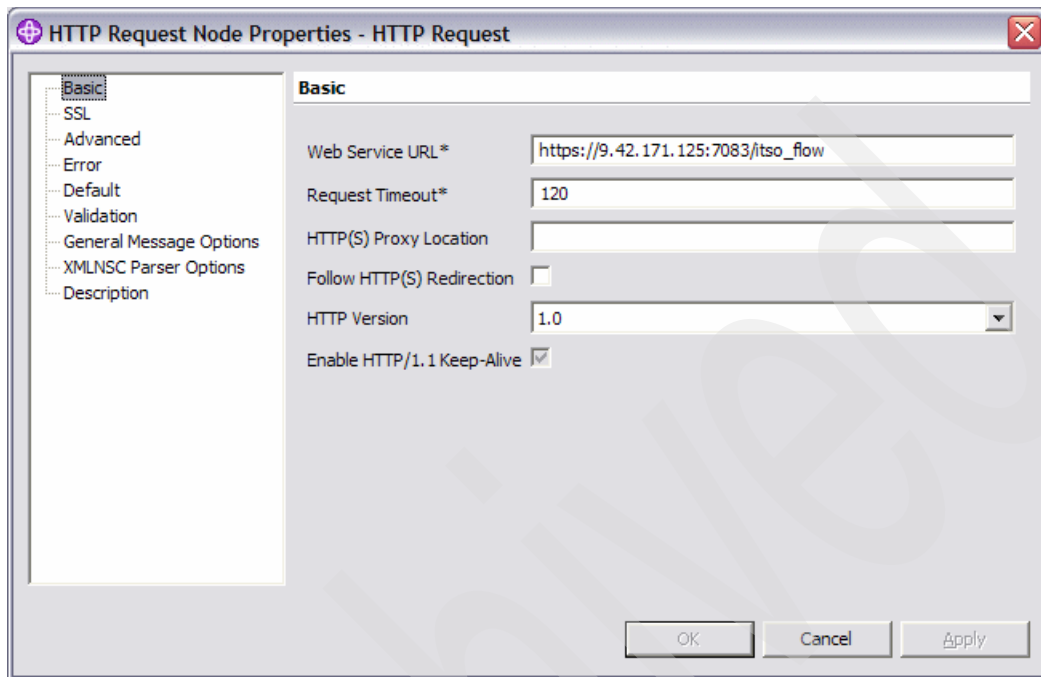


Figure 10-8 Invoking a Web service using SSL

Note: The default port of the SSL HTTP listener is 7083. Use the `mqsi changeproperties` command to change its value.

The next step in our scenario is to configure the SSL encryption in the server message flow. To do so, select **Use HTTPS** and assure that the URL Selector has the correct value, as shown in Figure 10-9.

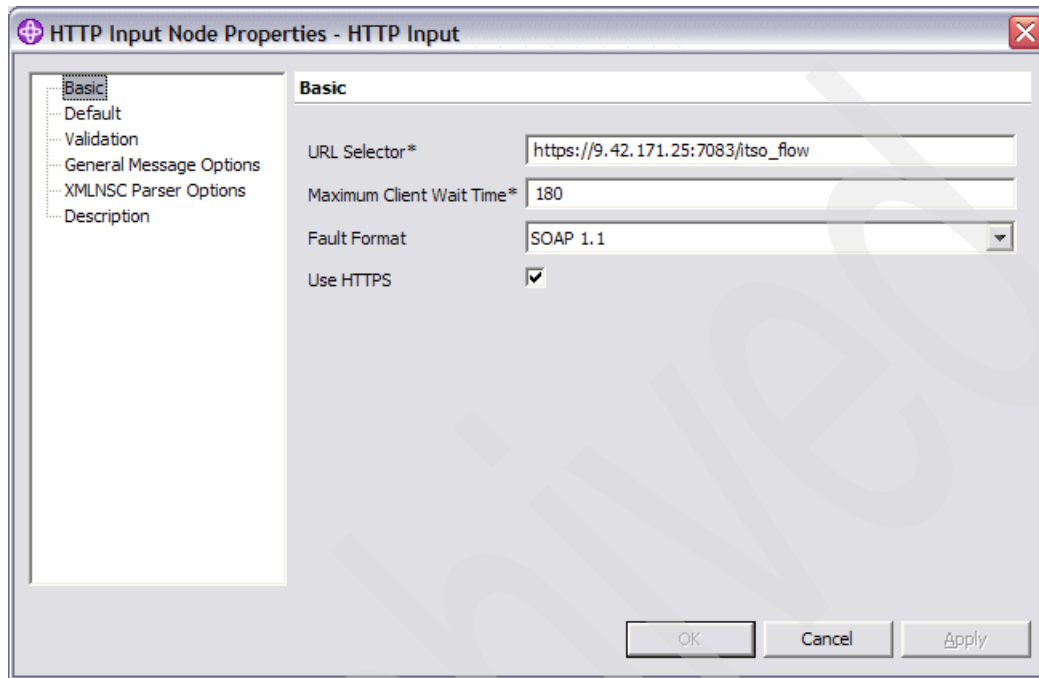


Figure 10-9 Configuring SSL in the HTTP Input node

Testing the flows

The final step is to test the configurations. One way to test the server message flow is using a Web service client. In our scenario, we use the Web browser for simplicity.

To probe the server message flow, you must follow these steps:

1. Open a Web browser window and go to the URL that is specified in the HTTP input node as shown in Figure 10-8 on page 244 and Figure 10-9 on page 245.

Figure 10-10 shows a sample of this invocation. In our scenario, you can ignore the warning that the browser shows only if it informs a mismatch error between the name of the certificate and the name of the server. Otherwise, you must check the procedure for generating and signing the SSL keys.

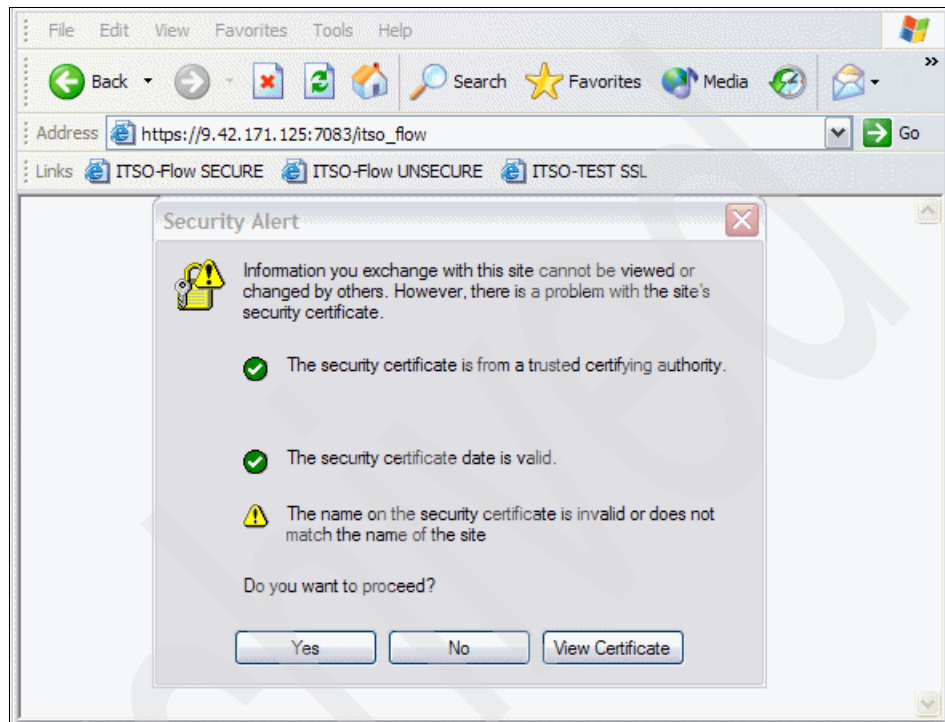


Figure 10-10 Invocation of the server message flow from a Web browser

Note: This warning (which indicates that the certificate is trusted but the name of the server does not match with the Web server that was requested) is generated because we are not using fully qualified domain names in our scenario. To avoid this warning, you must define the common name of your certificate identically to the FQDN of the message broker Web server.

2. The server answers the request with the message that is shown in Figure 10-11.

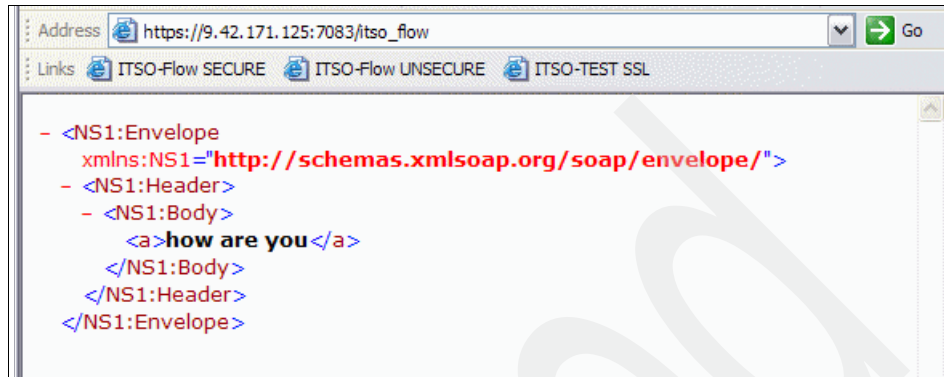


Figure 10-11 SOAP response from the web service message flow

To test the client message flow, follow these steps:

1. Create a test XML message and save it in a file (for example `c:\temp\message.xml`).
2. Load the file and send it to the queue `QM_BROKER1.IN`, using a WebSphere MQ tool, such as `RFHUtil`.

If everything works properly, you find a message in the `QM_BROKER1.OUT` queue (Figure 10-12).

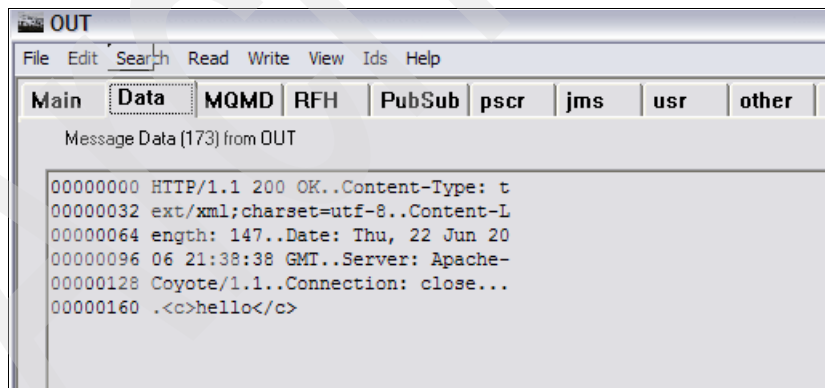


Figure 10-12 Response from the client message flow

If this is not the response that you obtain from the queue or if there are messages on the Dead Letter queue, you must review your queue manager configuration.

Refer to Appendix C, “Additional material” on page 301, and look for the file named Ch10.3_HTTPS_MessageFlows.zip. It contains the message flows that we used in this security scenario in a Project Interchange zipped file.

10.3.3 Connection encryption in Toolkit

Another important example of connection encryption inside a message broker environment is the implementation of SSL between the Message Brokers Toolkit and the Configuration Manager. Because the Message Brokers Toolkit is an Eclipse-based application and it uses the WebSphere MQ Java client technology, you can use this example as a model to implement SSL encryption between any WebSphere MQ Java client and a message broker.

The initial considerations in this implementation are:

- ▶ The SSL configuration of the Message Brokers Toolkit is based on the standard Java security **keytool** utility.
- ▶ The SSL configuration of the message broker (in this case, the message broker or the Configuration Manager) is based on the implementation of SSL in the queue manager of this broker.
- ▶ The use of a CA is highly recommended (in our scenario, we use OpenSSL to configure our own CA).

The main tasks to implement a 2-way SSL communication between the Message Brokers Toolkit and the Configuration Manager are:

- ▶ Creating the keystore of the queue manager
- ▶ Adding the CA certificate to the queue manager's keystore
- ▶ Generating the queue manager's certificate
- ▶ Generating the certificate for the Message Brokers Toolkit
- ▶ Enabling the SSL encryption on the queue manager
- ▶ Enabling the SSL encryption on the Java client
- ▶ Testing the SSL communication

Creating the keystore of queue manager

To create the keystore of the queue manager in our scenario, follow these steps:

1. In the Configuration Manager machine, open the WebSphere MQ Explorer, right-click the **IBM WebSphere MQ** icon, and select **Manage SSL Certificates** option of the menu, as shown in Figure 10-13. This option opens the IBM Key Management Tool.

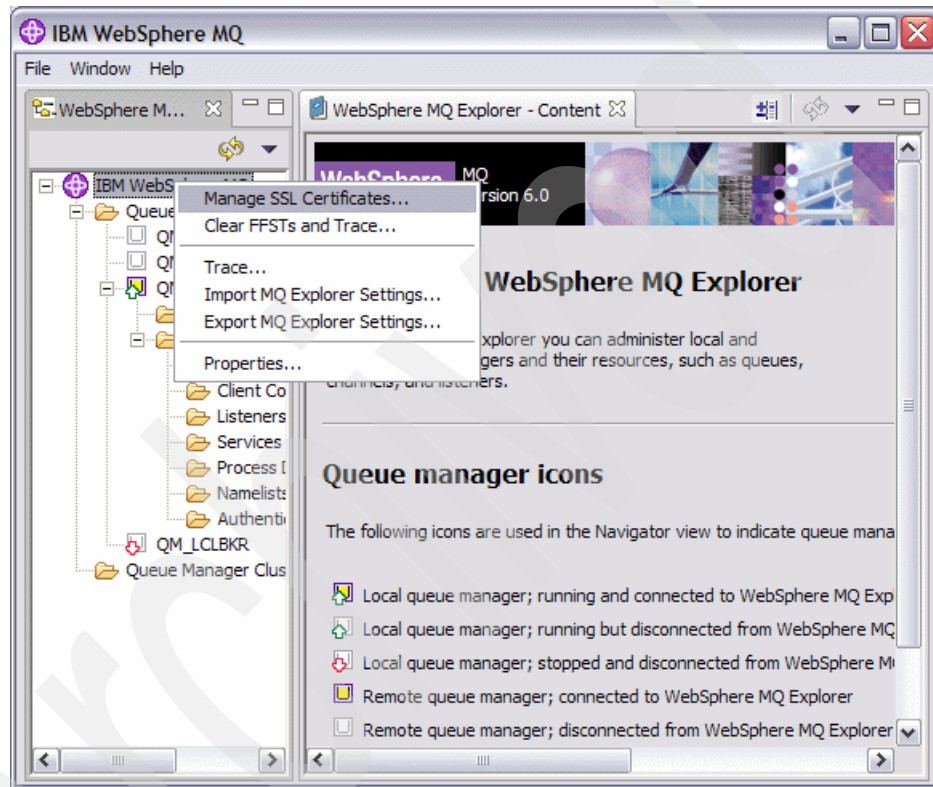


Figure 10-13 Opening the IBM Key Management tool

2. In the IBM Key Management tool, choose **Key Database File** → **New** and create a keystore using the parameter values shown in Figure 10-14. Click **OK**.

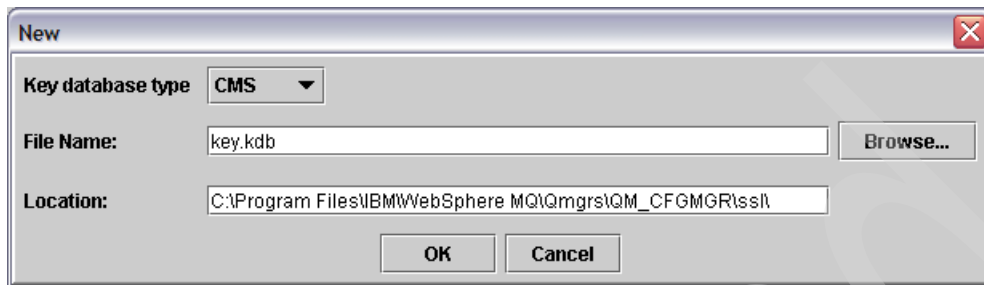


Figure 10-14 Keystore parameter values

Note: The name of the keystore file must be *key.kdb*, and the location of this file must be the *ssl* directory under the queue manager directory inside the WebSphere MQ directory structure.

3. Assign a password for this keystore, and select **Stash the password to a file** (Figure 10-15). Click **OK**.

Tip: In our scenario, we use the password `its02006`. However, in a production environment, we suggest that you use a stronger password.

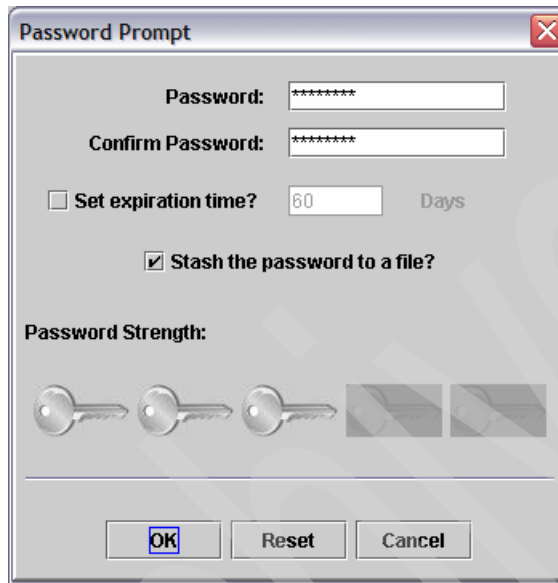


Figure 10-15 Assigning a password

4. After you create a password, an information window displays that shows the final location of the password files (Figure 10-16). The WebSphere MQ components use this password file to access the keystore and the certificates that are stored there.

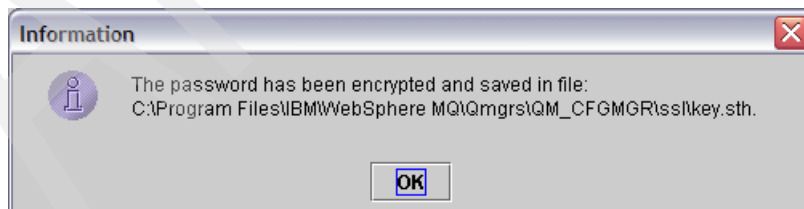


Figure 10-16 Password location window

Adding the CA certificate on queue manager

After you create the keystore repository, you must add the CA root certificate. This certificate helps the queue manager to trust the Message Brokers Toolkit's certificate. To add this root certificate, follow these steps:

1. In the IBM Key Management tool, open the key.db file that you just created (in “Creating the keystore of queue manager” on page 249) if the file is not already open.
2. Select **Signer Certificates** in the **Key database content** panel, and then click **Add** (Figure 10-17).

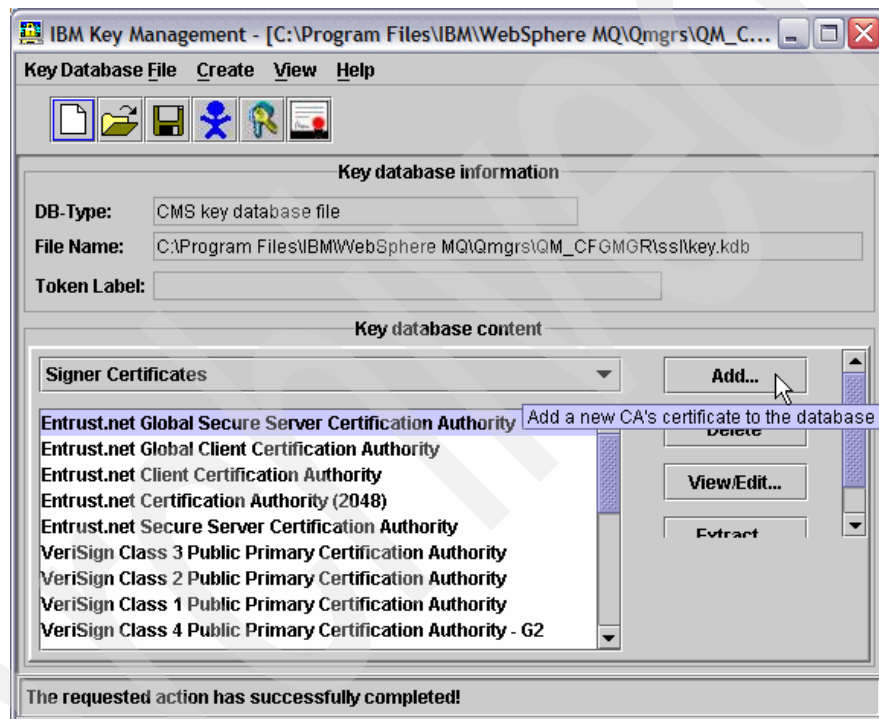


Figure 10-17 Adding a new CA root certificate

3. Enter the information about the CA certificate location in the dialog box, as shown in Figure 10-18, and then click **OK**.

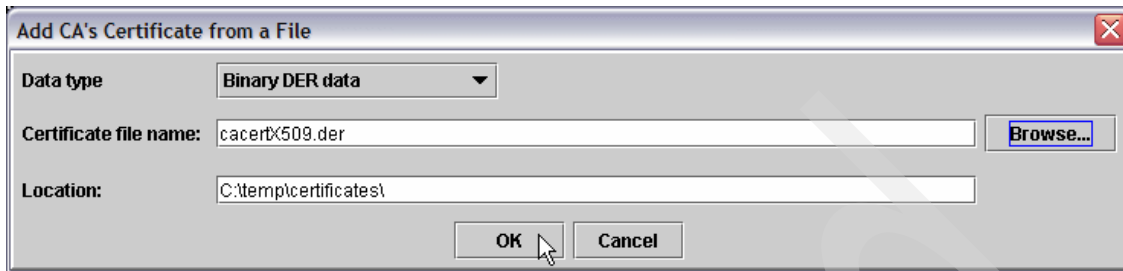


Figure 10-18 Adding the CA's certificate from a file

4. Enter a label name for this certificate. In our scenario we use the label of ITS0 CA Certificate, as shown in Figure 10-19. Click **OK**.

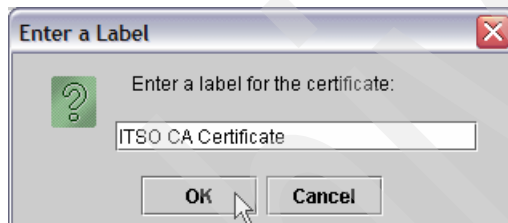


Figure 10-19 Label of the CA certificate

The new added CA root certificate should now display under the Signer Certificates option of the Key database content (Figure 10-20).

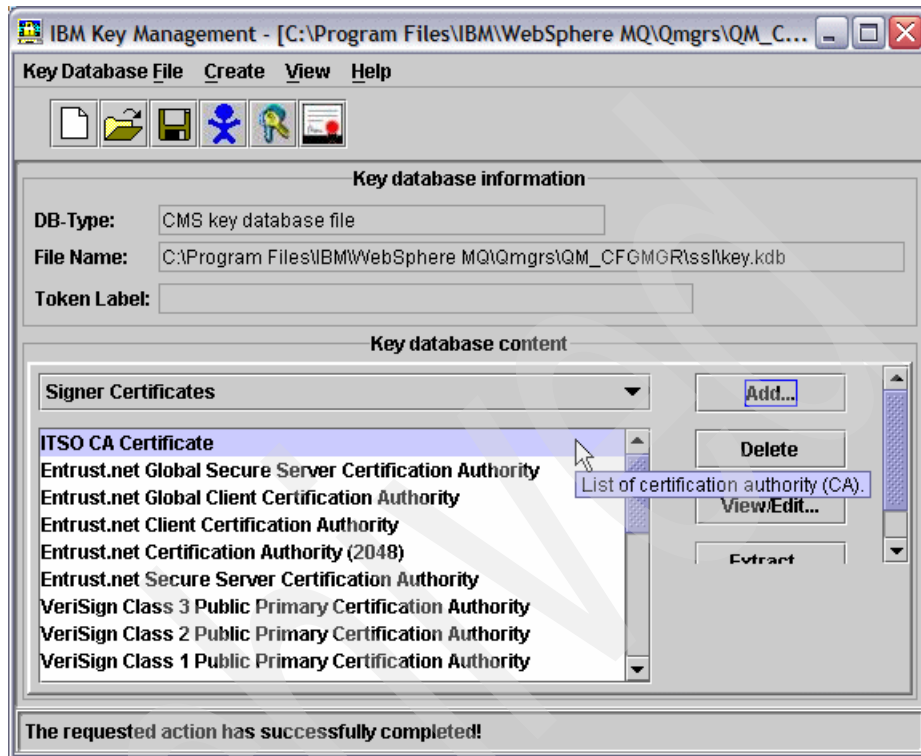


Figure 10-20 List of CA certificates, including the new added certificate

Generating the queue manager's certificate

Now, you have to create the queue manager's certificate. This certificate is used to validate the authenticity of the queue manager to the Java client. Follow these steps:

1. In the IBM Key Management tool, select **Create** → **New Certificate Request**, as shown in Figure 10-21.

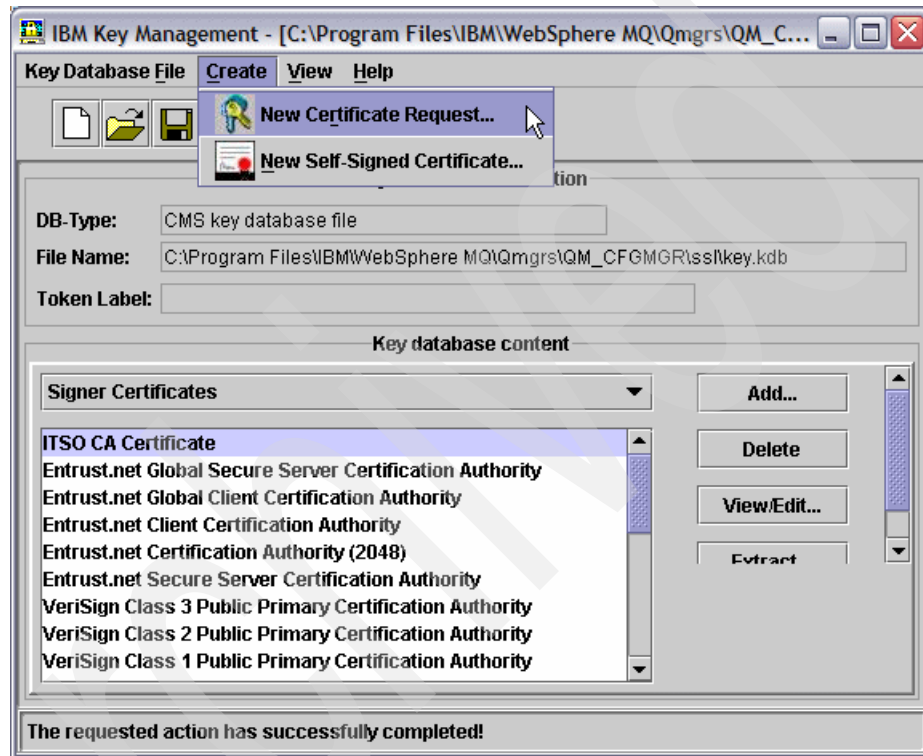
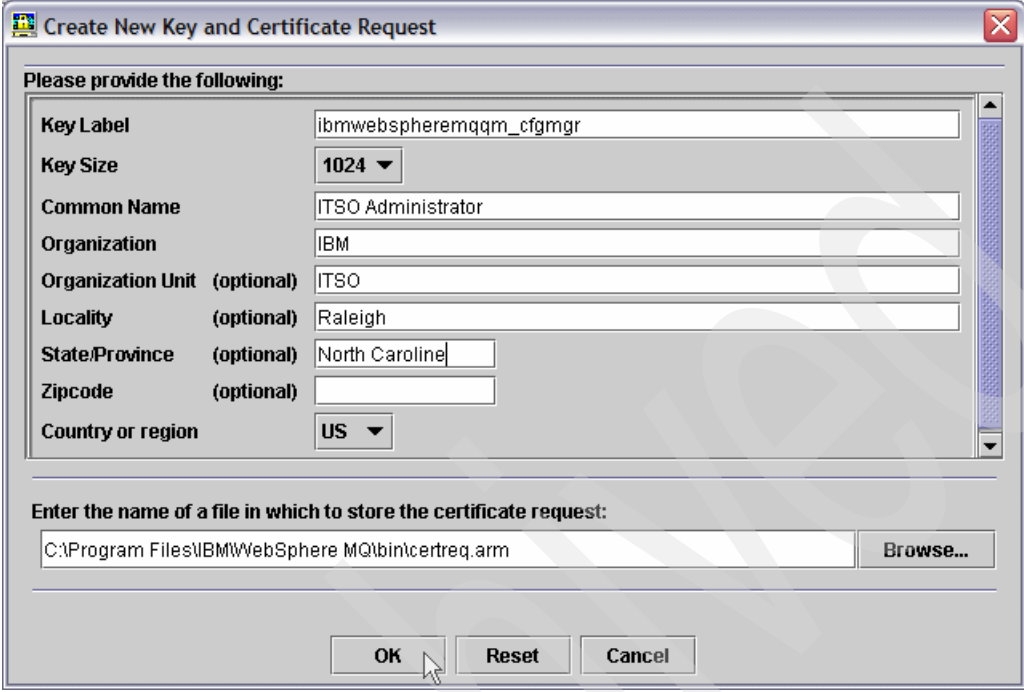


Figure 10-21 Creating a new certificate request

2. Complete the Certificate Request form according to the following rules:
 - The key label must conform with the syntax `ibmwebspheremqXXX`, where XXX is the name of the queue manager.
 - The key size must be 1024.
 - Remember the path that you used to store the certificate request.

Figure 10-22 on page 256 shows an example of the Certificate Request form. When the form is complete, click **OK**.



The dialog box is titled "Create New Key and Certificate Request". It contains a section "Please provide the following:" with several input fields: "Key Label" (text box with "ibmwebspheremqqm_cfgmgr"), "Key Size" (dropdown menu with "1024"), "Common Name" (text box with "ITSO Administrator"), "Organization" (text box with "IBM"), "Organization Unit (optional)" (text box with "ITSO"), "Locality (optional)" (text box with "Raleigh"), "State/Province (optional)" (text box with "North Carolina"), "Zipcode (optional)" (text box), and "Country or region" (dropdown menu with "US"). Below this section is a section "Enter the name of a file in which to store the certificate request:" with a text box containing "C:\Program Files\IBM\WebSphere MQ\bin\certreq.arm" and a "Browse..." button. At the bottom are "OK", "Reset", and "Cancel" buttons.

Figure 10-22 Certificate Request form

3. A confirmation window, such as that shown in Figure 10-23, indicates the final location of the certificate request file. Click **OK** in this information window.



Figure 10-23 Location of the certificate request file

4. Sign the certificate request using the certification authority tools. In our scenario, we use OpenSSL and its tools to operate our own CA. Example 10-13 on page 257 shows the script to sign the certificate request and to generate the response in a X509 format.

Example 10-13 Signing and formatting the certificate response to X509

```
C:\OpenSSL\bin>openssl ca -in certreq.arm -out qm_cfgmgr.pem -keyfile demoCA/private/cakey.pem -cert demoCA/cacert.pem
```

```
Using configuration from C:\OpenSSL\bin\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for demoCA/private/cakey.pem: itso2006
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number:
        ab:76:74:bd:2f:1c:46:77
    Validity
        Not Before: Jun 21 18:53:55 2006 GMT
        Not After : Jun 21 18:53:55 2007 GMT
    Subject:
        countryName             = US
        stateOrProvinceName     = North Caroline
        organizationName        = IBM
        organizationalUnitName  = ITS0
        commonName              = ITS0 Administrator
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            49:B9:5B:5C:D3:91:A9:60:FE:A4:04:56:8B:C0:FC:19:B7:93:54:EB
        X509v3 Authority Key Identifier:
            keyid:9D:51:4F:9F:3E:6F:68:DE:E1:FC:D9:7A:C4:65:44:3F:1E:C2:9D:E5
```

```
Certificate is to be certified until Jun 21 18:53:55 2007 GMT (365 days)
```

```
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
```

```
Write out database with 1 new entries
```

```
Data Base Updated
```

```
C:\OpenSSL\bin>openssl x509 -inform PEM -outform DER -in qm_cfgmgr.pem -out qm_frgmgr_X509.der
```

5. In the IBM Key Management Tool (Figure 10-24), click **Receive** to receive the certificate response from the certificate authority and store it in the *Personal Certificates* content of the keystore.

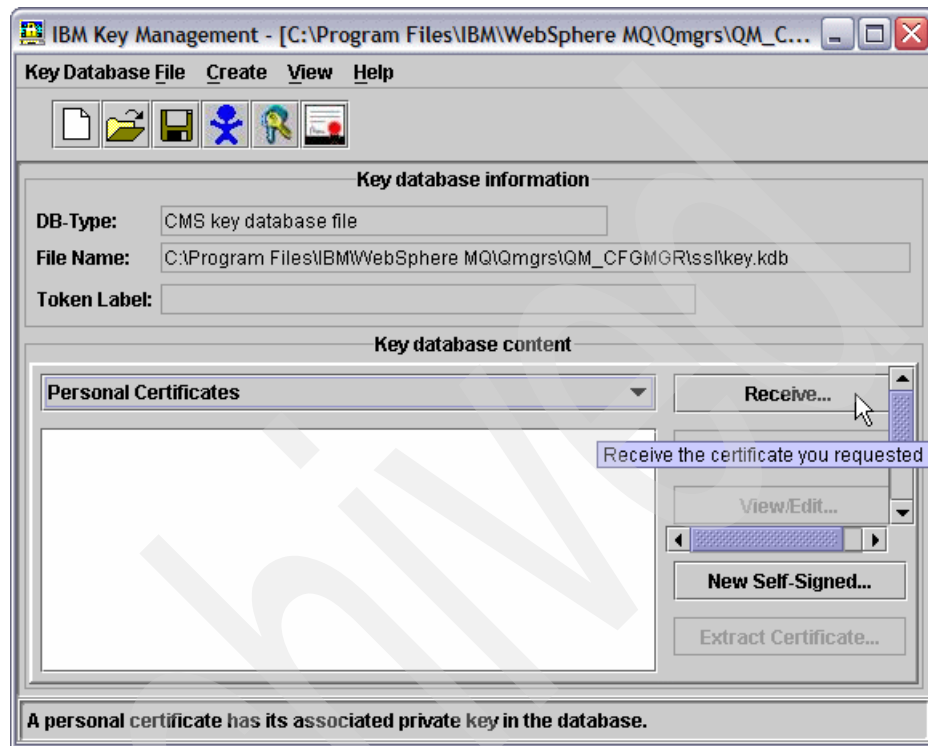


Figure 10-24 Receiving the certificate response from the CA

6. Complete the form with the appropriated information to get the response file, and be sure that the data type of the certificate is Binary DER data, as shown in Figure 10-25. Click **OK**.



Figure 10-25 Certificate response location

The new personal certificate appears in the keystore file (Figure 10-26).

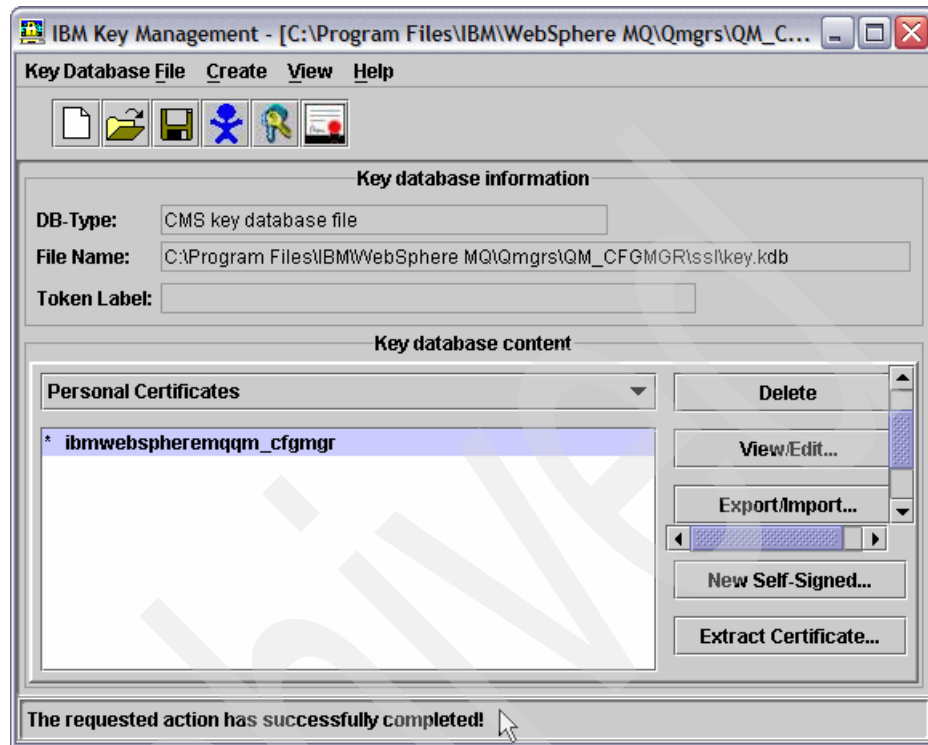


Figure 10-26 The added certificate for the queue manager

Generating the certificate for Message Brokers Toolkit

After configuring the queue manager side of the communication, you need to configure and set up the Java side. Follow these steps:

1. Generate a private key using a keysize of 2048 and the RSA algorithm using the Java **keytool** command, as shown in Example 10-14.

Example 10-14 Generating the private key of the Message Brokers Toolkit

```
C:\IBM\MQSI\6.0\bin>keytool -genkey -alias mbtoolkit -keyalg RSA -keysize 2048
```

```
Enter keystore password: itso2006
What is your first and last name?
  [Unknown]: Message Brokers Toolkit
What is the name of your organizational unit?
  [Unknown]: ITS0
What is the name of your organization?
  [Unknown]: IBM
```

What is the name of your City or Locality?
[Unknown]: Raleigh
What is the name of your State or Province?
[Unknown]: North Caroline
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=Message Brokers Toolkit, OU=ITSO, O=IBM, L=Raleigh, ST=North Caroline, C=US correct? (type "yes" or "no")
[no]: yes

Enter key password for <mbtoolkit>
(RETURN if same as keystore password): itso2006

2. Generate a certificate request for the Message Brokers Toolkit, as shown in Example 10-15.

Example 10-15 Generation of the certificate request

```
C:\IBM\MQSI\6.0\bin>keytool -certreq -alias mbtoolkit -file mbtoolkit.crs
```

Enter keystore password: itso2006

3. Send the certificate request of the Message Brokers Toolkit to the CA to be signed. The signed certificate must be converted to the X509 format, as shown in Example 10-16.

Example 10-16 Signing of the certificate request and formatting of the response

```
C:\OpenSSL\bin>openssl ca -in mbtoolkit.crs -out mbtoolkit.pem -keyfile demoCA/private/cakey.pem -cert demoCA/cacert.pem
```

```
Using configuration from C:\OpenSSL\bin\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for demoCA/private/cakey.pem: itso2006
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number:
        ab:76:74:bd:2f:1c:46:78
    Validity
        Not Before: Jun 21 19:30:06 2006 GMT
        Not After : Jun 21 19:30:06 2007 GMT
    Subject:
        countryName           = US
        stateOrProvinceName   = North Caroline
        organizationName      = IBM
```



```
organizationalUnitName    = ITSO
commonName                = Message Brokers Toolkit
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    30:7D:B0:E9:99:D0:9B:FE:9A:AE:E5:79:10:CB:10:22:9B:8E:0C:91
  X509v3 Authority Key Identifier:
    keyid:9D:51:4F:9F:3E:6F:68:DE:E1:FC:D9:7A:C4:65:44:3F:1E:C2:9D:E5
```

Certificate is to be certified until Jun 21 19:30:06 2007 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

```
C:\OpenSSL\bin>openssl x509 -inform PEM -outform DER -in mbtoolkit.pem -out mbtoolkit_X509.der
```

-
4. Import the certificate response into the Java keystore file using the **keytool** command, as shown in Example 10-17.

Example 10-17 Storing the certificate in the Java keystore file

```
C:\IBM\MQSI\6.0\bin>keytool -import -alias mbtoolkit -file mbtoolkit_X509.der -trustcacerts
```

```
Enter keystore password: itso2006
Certificate reply was installed in keystore
```

Enabling SSL encryption in Configuration Manager

The Configuration Manager uses WebSphere MQ as its transport layer. The Configuration Manager implies that you must enable the SSL encryption in this layer. Every SSL encryption in WebSphere MQ must secure both ends of the channel that are involved in the communication between the queue managers. In our case, because we are trying to secure the communication between the Message Brokers Toolkit and the Configuration Manager, we have to be aware of the fact that the WebSphere MQ end channel that is used to receive the messages from the toolkit in the Configuration Manager side is SYSTEM.BKR.CONFIG.

To secure the SYSTEM.BKR.CONFIG channel, you can change the properties according to the WebSphere MQ documentation. In our scenario, we use the WebSphere MQ Explorer for simplicity. Figure 10-27 shows how to get the properties dialog of a WebSphere MQ channel in the WebSphere MQ Explorer.

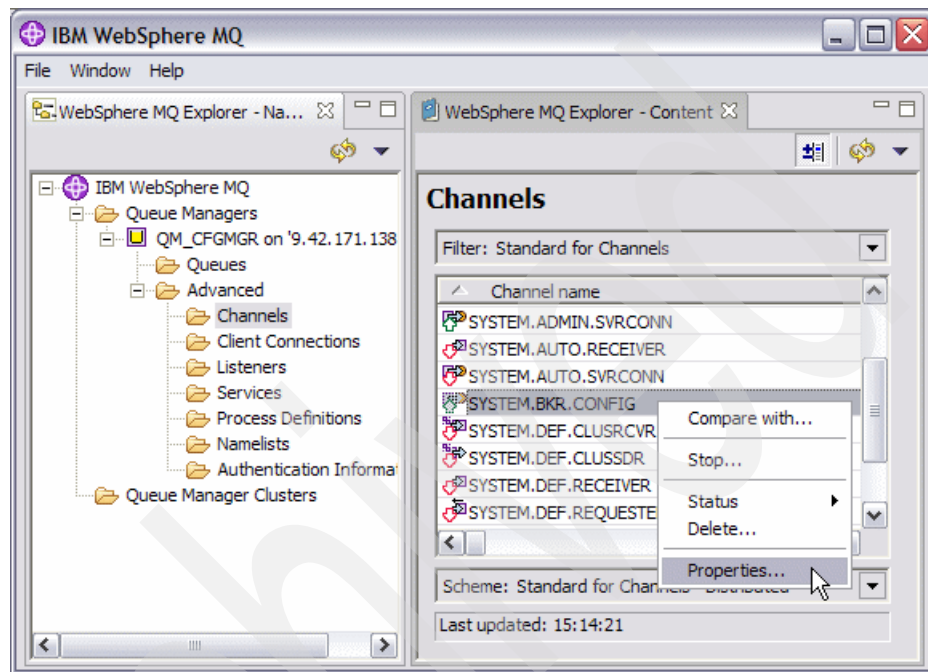


Figure 10-27 Securing the WebSphere MQ channel

In the Properties dialog, change the SSL properties to adjust the value of the SSL CipherSpec attribute. This attribute identifies the combination of encryption algorithm and Message Authentication Code (MAC) algorithm that the SSL connection uses. In our scenario, we select the NULL_MD5 value, as shown in Figure 10-28.

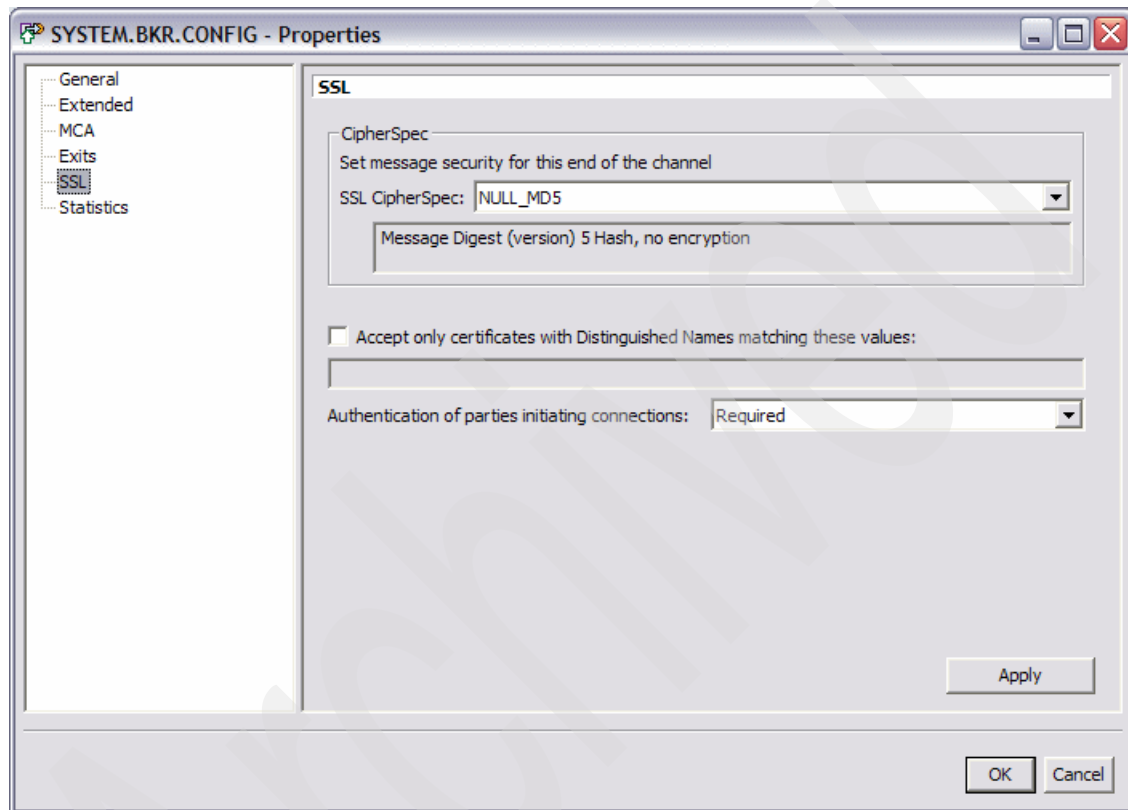


Figure 10-28 Setting the SSL properties in the SYSTEM.BKR.CONFIG channel

Attention: Both ends of a SSL connection must agree the same CipherSpec to be able to communicate.

Enabling SSL in Message Brokers Toolkit

The Message Brokers Toolkit is an Eclipse-based application, and it uses the WebSphere MQ Java client technology to connect to the Configuration Manager. The best way to enable the use of SSL is to set up the Java system properties related to SSL.

Table 10-6 lists the properties that we use in our scenario.

Table 10-6 SSL Java system properties

Property	Description
<code>javax.net.ssl.keyStore</code>	Specifies the full path of the Java keystore file.
<code>javax.net.ssl.keyStorePassword</code>	Defines the password of the Java keystore.
<code>javax.net.ssl.trustStore</code>	Specifies the full path of the store of the trusted CA root certificates.
<code>javax.net.ssl.trustStorePassword</code>	Specifies the password of the trust store.

To set these properties, follow the directions in the Message Brokers Toolkit documentation. In our scenario, we change the command line invocation of the program from `wmbt.exe` to that shown in Example 10-18.

Example 10-18 Changing SSL Java system properties

```
wmbt.exe -vmargs -Djavax.net.ssl.keyStore="c:\Documents and  
Settings\wmb\keystore" -Djavax.net.ssl.keyStorePassword=itso2006  
-Djavax.net.ssl.trustStore="c:\IBM\MQSI\6.0\jre\lib\security\cacerts"  
-Djavax.net.ssl.trustStorePass=changeit
```

We also set the properties of the domain configuration in the Toolkit Broker Administration Navigator view to indicate:

- ▶ The path of the Key Store and the Trust Store attributes.
- ▶ The value of the Cipher Suite attribute. (Because it must match the value set in the other channel end, in our scenario we set it to `SSL_RSA_WITH_NULL_MD5`.)

At the time of the writing of this book, you cannot set the password information in the domain configuration window.

Figure 10-29 shows the domain configuration window with the attribute values for our scenario.

The screenshot shows a Java Swing window titled "QM_CFGMGR.configmgr". The window has a standard Mac OS X title bar with three buttons (red, yellow, green) and a close button. The window contains several input fields and buttons:

- Queue Manager Name:** A text field containing "QM_CFGMGR".
- Host:** A text field containing "9.42.171.138".
- Port:** A text field containing "1414".
- Security Exit:** A section containing:
 - Class:** An empty text field.
 - JAR File Location:** A text field with a "Browse..." button to its right.
- SSL Parameters:** A section containing:
 - Cipher Suite:** A dropdown menu showing "SSL_RSA_WITH_NULL_MD5" and a "More..." button to its right.
 - Distinguished Names:** An empty text field.
 - CRL Name List:** An empty text field.
 - Key Store:** A text field containing "c:\Documents and Settings\wmb\keystore" with a "Browse..." button to its right.
 - Trust Store:** A text field containing "c:\IBM\MQSI\6.0\jre\lib\security\cacerts" with a "Browse..." button to its right.

Figure 10-29 Domain configuration properties

Testing the SSL connection

To test the configuration changes, start the Message Brokers Toolkit and then establish a connection with the Configuration Manager, as shown in Figure 10-30.

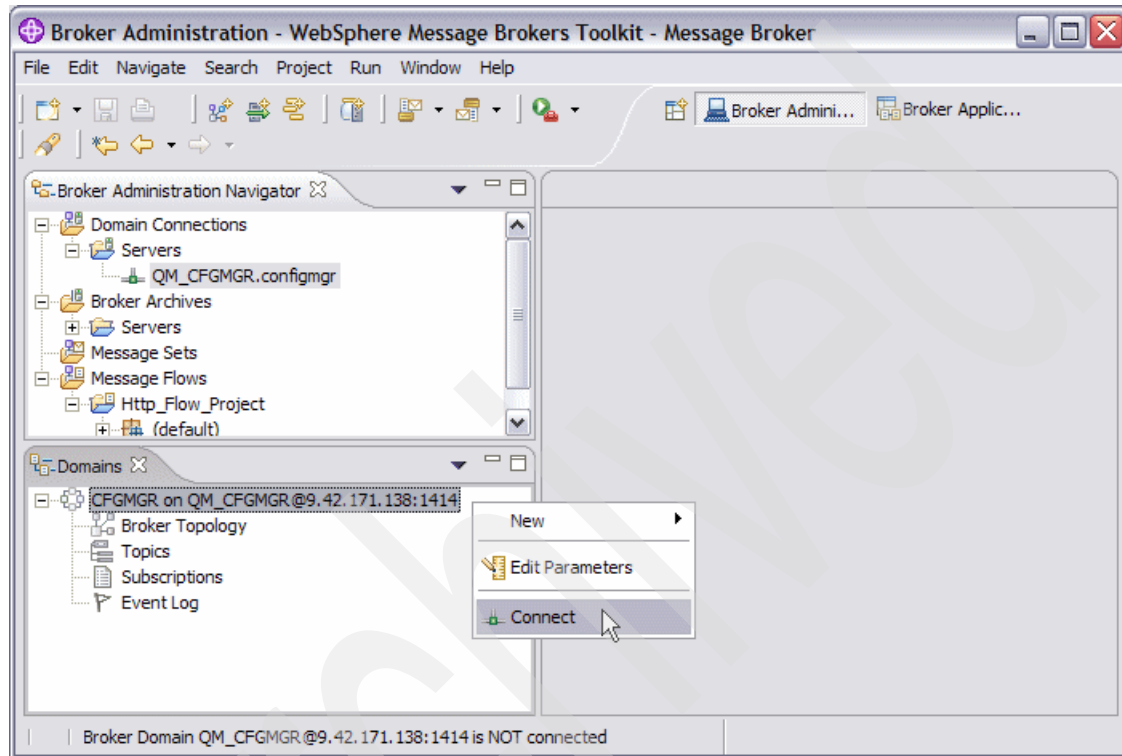


Figure 10-30 Connecting to the Configuration Manager from the Toolkit with SSL

If everything is configured properly, the connection success.

10.4 Backup, restore, and recover

As introduced in Chapter 6, “Backup, restore, and recover” on page 115, this section illustrates backup, restore, and recover operations for the broker. We use the IBM DB2 Universal Database V8.2 in this scenario. For a database engine other than DB2, refer to the corresponding documentation for the backup and restore tasks.

10.4.1 Backing up broker

We do not discuss queue manager backup here because it is not needed for our scenarios. For our discussion, we describe only how to restore and recover the broker if its repository database fails.

To back up the broker component:

1. Before taking the backup, ensure that no deploy process is running to the broker and then stop the broker using the following command:
`mqsisstop <broker_name>`
2. Verify that all the broker processes are stopped, as shown in Example 10-19.

Example 10-19 Checking for the broker processes

```
$ ps -ef |grep bip
wmb 37242 28086 0 10:20:48 pts/3 0:00 grep bip
$
```

3. Take the backup of broker database as shown in Example 10-20. In this example, MQSITCP is the broker database.

Example 10-20 Backing up the broker database

```
$ db2 backup database MQSITCP user wmb using wmb to /backup/BROKERDB
with 2 buffers buffer 1024 parallelism 1 without prompting
```

```
Backup successful. The timestamp for this backup image is :
20060614102423
$
```

Attention: Make a note of the time stamp that displays after the successful completion of a backup. The time stamp is unique for each backup. You use the time stamp as a parameter for the command to restore the database. If you do not know the value for the time stamp when you perform a restore, then only one backup image must be available on the source media. Thus, noting the time stamp ensures that you use the correct backup in a restore operation, if it is required at a later stage. For more information, refer to corresponding documentation for your database engine.

4. Take the backup of the broker registry, using the command in Example 10-21.

Example 10-21 Backing up the broker registry

```
$ tar -cf /backup/BROKER_REGISTRY/registry.tar /var/mqsi/registry
```

5. Along with the registry backup, also take broker service information using following command:

```
mqsiservice <broker_name>
```

For future reference and restore operation, save the output from this command. The Component UUID field denotes the UUID of the broker, as shown in Example 10-22.

Example 10-22 Storing the broker UUID information

```
$ mqsiservice BROKER2 > /backup/BROKER2_serviceinfo.txt
```

```
$ cat /backup/BROKER2_serviceinfo.txt
```

```
BIPV600 en US
```

```
ucnv Console CCSID 819 dft ucnv CCSID 819
```

```
ICUW ISO-8859-1 ICUA ISO-8859-1
```

```
Install Path = /opt/IBM/mqsi/6.0
```

```
Working Path = /var/mqsi
```

```
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08
```

```
process id = 34276
```

```
broker db name = MQSITCP
```

```
broker db userId = wmb
```

```
broker db password =
```

```
queue manager = QM_BROKER2
```

```
pubsub migration = no
```

```
fastpath Queue Manager = no
```

```
configuration timeout = 300
```

```
configuration delay timeout = 60
```

```
statistics major interval = 60
```

```
ComponentType = Broker
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $
```

10.4.2 Restoring broker from backup

To restore the broker component from backup, do the following:

1. Ensure that no deploy process is running and stop the broker, if it is running, using the following command:

```
mqsisistop <broker_name>
```

2. Delete the broker using following command:

```
mqsideletebroker <broker_name> -w
```


Example 10-23 Deleting the broker

```
$ mqsideletebroker BROKER2 -w
```

```
WebSphere MQ queue manager 'QM_BROKER2' starting.  
5 log records accessed on queue manager 'QM_BROKER2' during the log  
replay phase.  
Log replay for queue manager 'QM_BROKER2' complete.  
Transaction manager state recovered for queue manager 'QM_BROKER2'.  
WebSphere MQ queue manager 'QM_BROKER2' started.  
BIP8071I: Successful command completion.  
$
```

3. Verify that the broker is deleted, as shown in Example 10-24.

Example 10-24 Verifying the deletion of broker

```
$ mqsilist
```

```
BIP8099I: UserNameServer: UserNameServer - QM_UN$
```

```
BIP8071I: Successful command completion.  
$
```

4. Re-create the broker with the same name, as shown in Example 10-25.

Example 10-25 Recreating the broker

```
$ mqsicreatebroker BROKER2 -i wmb -a wmb -n MQSITCP -q QM_BROKER2
```

```
AMQ8110: WebSphere MQ queue manager already exists.  
WebSphere MQ queue manager running.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.
```

```
The setmqaut command completed successfully.  
BIP8071I: Successful command completion.  
$
```

5. Restore the broker database, as shown in Example 10-26.

Example 10-26 Restoring the broker database

```
$ db2 RESTORE DATABASE MQSITCP user wmb using wmb FROM  
"/backup/BROKERDB" TAKEN AT 20060614102423 WITH 2 BUFFERS BUFFER 1024  
REPLACE EXISTING PARALLELISM 1 WITHOUT ROLLING FORWARD WITHOUT  
PROMPTING
```

```
SQL2540W Restore is successful, however a warning "2539" was  
encountered during Database Restore while processing in No Interrupt  
mode.  
$
```

Note: Message 2539 means that an existing database is being replaced with a new one.

6. Run the `mqsiservice` command for the broker using the `-r` option to set BrokerUUID to the value that you noted during backup:

```
mqsiservice <broker_name> -r BrokerUUID='<broker_uuid>'
```

Example 10-27 Resetting the broker UUID

```
$ mqsiservice BROKER2 -r BrokerUUID='be8ce7b9-0b01-0000-0080-f393237b0c08'
```

```
BIPv600 en US  
ucnv Console CCSID 819 dft ucnv CCSID 819  
ICUW ISO-8859-1 ICUA ISO-8859-1  
  
Install Path = /opt/IBM/mqsi/6.0  
Working Path = /var/mqsi  
Component UUID = be8ce7b9-0b01-0000-0080-f393237b0c08  
broker db name = MQSITCP  
broker db userId = wmb  
broker db password =  
queue manager = QM_BROKER2  
pubsub migration = no  
fastpath Queue Manager = no  
configuration timeout = 300  
configuration delay timeout = 60  
statistics major interval = 60
```

ComponentType = Broker

BIP8071I: Successful command completion.

7. Start the broker using the **mqsisstart** command and verify that the execution groups and message flows are available, as shown in Example 10-28.

Example 10-28 Starting and verifying the broker after restore operation

```
$ mqsistart BROKER2
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

```
$ mqsilist BROKER2
```

BIP8130I: Execution Group: default - 28800

BIP8071I: Successful command completion.

```
$ mqsilist BROKER2 -e default
```

BIP8131I: MessageFlow: MessageFlow1

BIP8071I: Successful command completion.

```
$
```

8. Redeploy the domain configuration to ensure the configuration across the broker domain is consistent.

10.4.3 Re-creating broker when broker backup not available

If the broker database fails and cannot be corrected, perform the following sequence of operations to re-create the broker:

1. Ensure that no deploy is running to the broker.
2. If the broker processes are still running, stop the broker using the **mqsisstop** command.
3. Delete the broker using the following command:

```
mqsideletebroker <broker_name> -w
```

Example 10-29 Deleting the broker

```
$ mqsideletebroker BROKER2 -w
```

WebSphere MQ queue manager 'QM_BROKER2' starting.

```
5 log records accessed on queue manager 'QM_BROKER2' during the log
replay phase.
Log replay for queue manager 'QM_BROKER2' complete.
Transaction manager state recovered for queue manager 'QM_BROKER2'.
WebSphere MQ queue manager 'QM_BROKER2' started.
BIP8071I: Successful command completion.
$
```

4. Verify that the broker is deleted, as shown in Example 10-30.

Example 10-30 Verifying the deletion of broker

```
$mqsilist
```

```
BIP8099I: UserNameServer: UserNameServer - QM_UN$
BIP8071I: Successful command completion.
```

```
$ ls /var/mqsi/registry
LOCK          UserNameServer
$
```

5. Use Configuration Manager Proxy API to remove all references to the broker:

- a. Start the Configuration Manager Proxy API Exerciser sample.

On Windows:

**Start → All Programs → IBM WebSphere Message Broker 6.0 → Java
Programming APIs → Configuration Manager Proxy API Exerciser**

On UNIX:

```
<product install dir>/sample/ConfigManagerProxy/  
StartConfigManagerProxyExerciser
```

- b. Connect to the Configuration Manager.

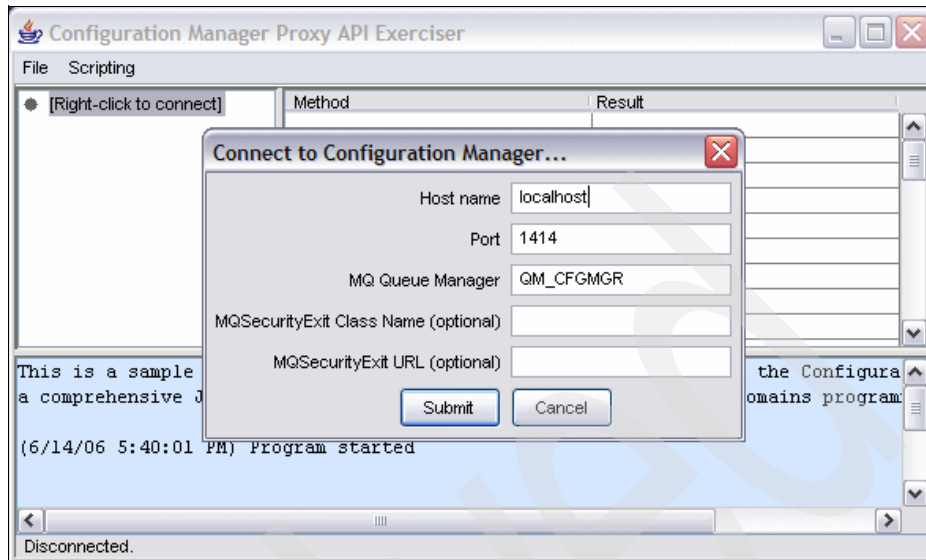


Figure 10-31 Connecting to Configuration Manager using CMPAPI Exerciser

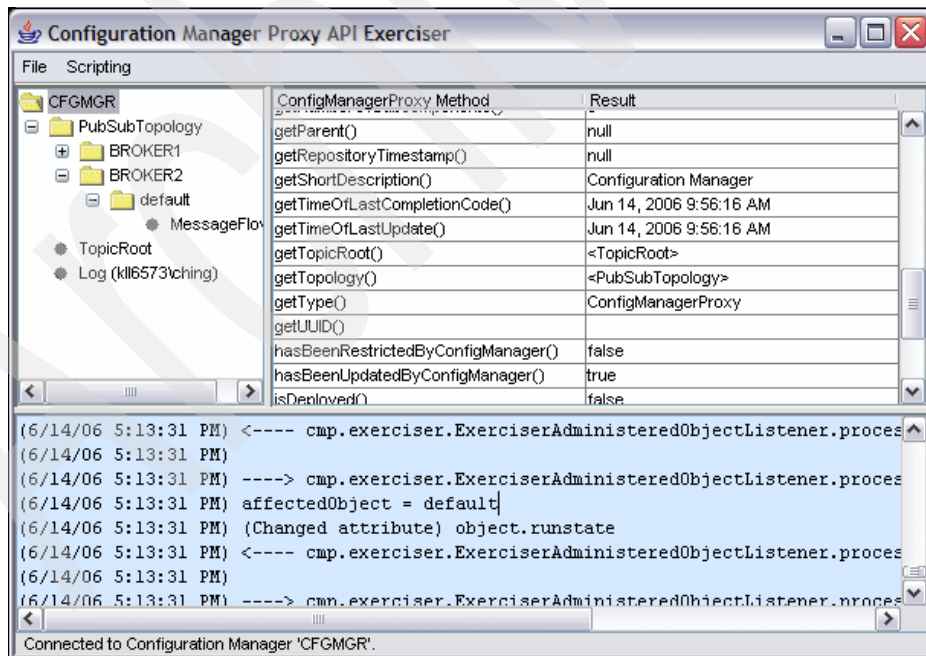


Figure 10-32 Connected to Configuration Manager

- c. Right-click the topology object, then click **Remove references to a previously deleted broker** (Figure 10-33).

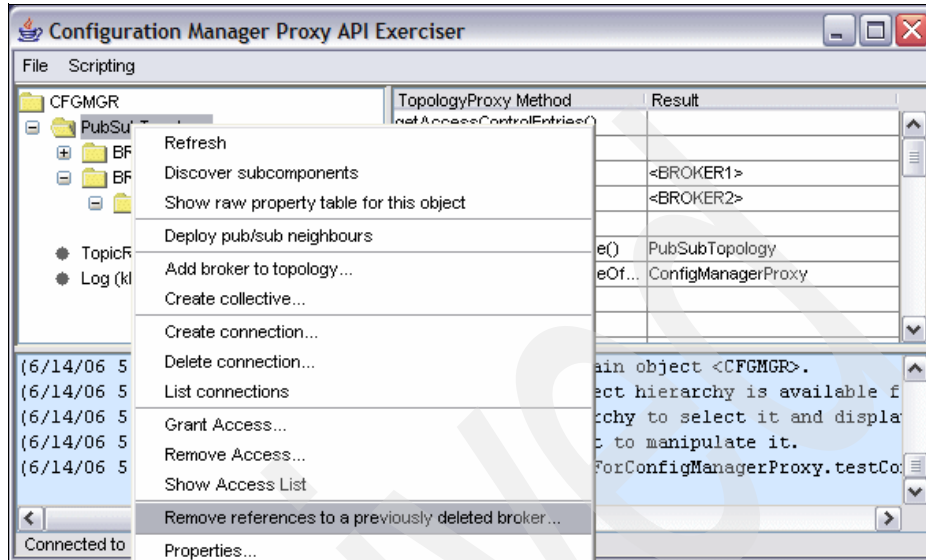


Figure 10-33 Selecting the option to delete the broker

- d. Enter the name of the broker to be removed and click **Submit** (Figure 10-34).

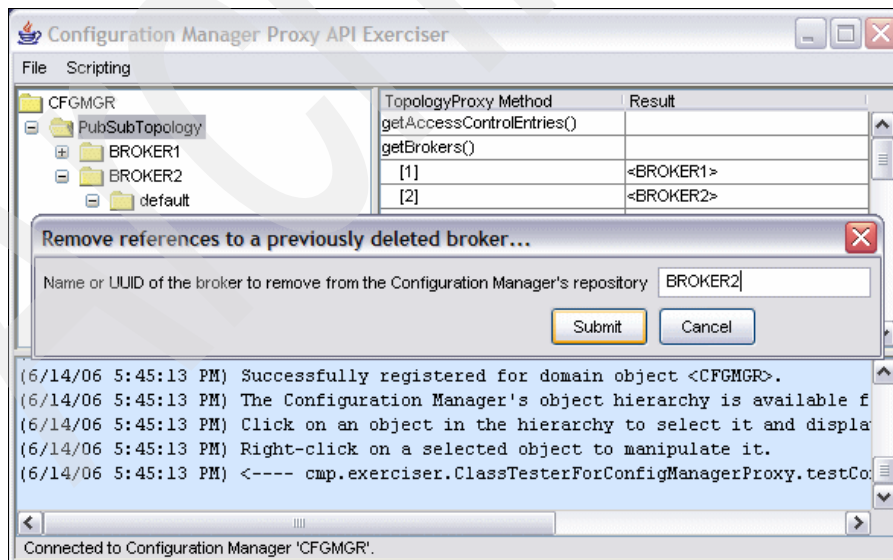


Figure 10-34 Specify the broker name to be deleted

e. Refer to the information in the message window (Figure 10-35).

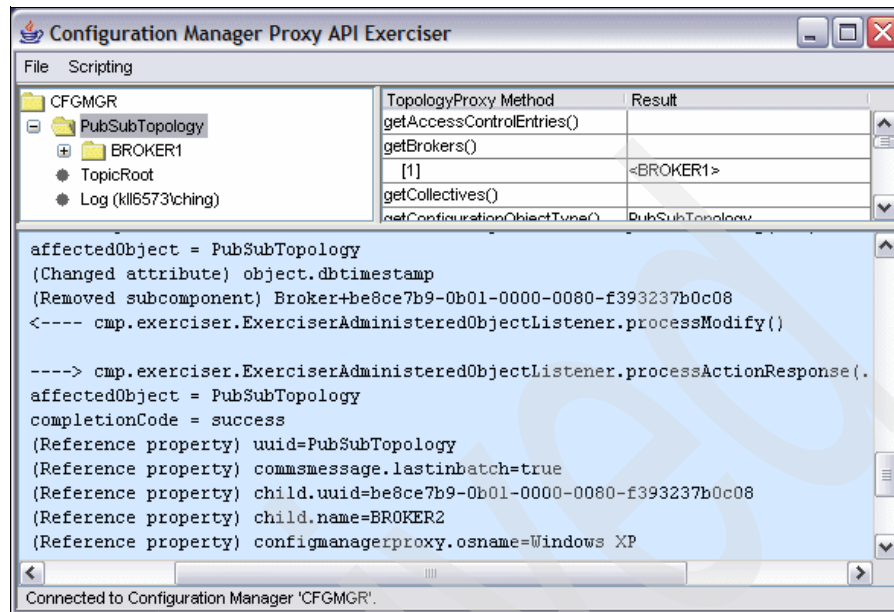


Figure 10-35 Output after deleting the broker

6. Re-create the broker using the **mqsicreatebroker** command, as shown in Example 10-31.

Example 10-31 Recreating the broker

```
$ mqsicreatebroker BROKER2 -i wmb -a wmb -n MQSITCP -q QM_BROKER2
```

AMQ8110: WebSphere MQ queue manager already exists.

WebSphere MQ queue manager running.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

The setmqaut command completed successfully.

```
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
The setmqaut command completed successfully.  
BIP8071I: Successful command completion.  
$
```

7. Start the broker using the **mqsi**start command.
8. Add the broker to the Configuration Manager domain topology.
9. Redeploy the broker configuration.

10.5 Problem determination

This section provides a problem determination scenario. Several of the tools and techniques introduced in Chapter 9, “Problem determination” on page 163. This scenario simulates a problem that could occur with an application databases used by a message flow.

For WebSphere Message Broker to be able to access application databases a valid user ID and password for the database must be available. The user ID and password used to access the database is set through the **mqsi**setdbparms command and must remain consistent with the actual user ID and password.

In this scenario, the user ID used to access the database unfortunately had its password changed without the WebSphere Message Broker administrator being notified. The remainder of these section describes the alert received, problem determination process and service restoration procedure used for this type of event. The message flow used in this scenario reads from input queue TESTDB.IN, performs an insert into database table TESTDB, then passes the message to output queue TESTDB.OUT, as shown in Figure 10-36.

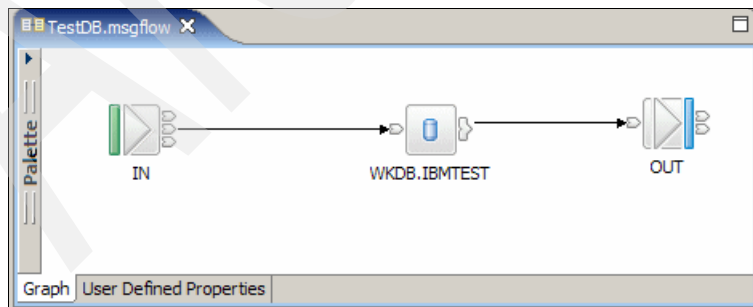


Figure 10-36 TestDB.msgflow

Example 10-32 shows the ESQL that we used to perform the database insert.

Example 10-32 ESQL to insert into database

```
CREATE DATABASE MODULE TestDB_Database
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN
    INSERT INTO Database.TESTDB (MSG,ARRIVE_DATE,ARRIVE_TIME) VALUES
    (Root.XML.Test,CURRENT_DATE,CURRENT_TIME);
  END;
END MODULE;
```

We deployed the message flow to the AIX-based BROKER2 and checked to ensure successful deployment. Before we caused the failure, we created a simple script called `startmessagetraffic.ksh` to simulate an application writing to the input queue and another one reading from the output queue (Example 10-33).

Example 10-33 Script startmessagetraffic.ksh

```
#!/usr/bin/ksh
## startmessagetraffic
## Sends a simple XML message to an input Q and then reads an output Q.

bn=`basename $0`
usage="Usage: $bn -i <input queue> -o <output queue> [-h]"

input_queue=""
output_queue=""
seconds=2

message() {
  echo "<Test><Timestamp>`date +%T`</Timestamp></Test>"
}

while getopts :i:o:h opt; do
  case $opt in
    :) print "$bn: Option -$OPTARG requires a value."
      print $usage; exit 1;;
    \?) if [[ -n ${OPTARG=""} ]]; then
        print "$bn: Unknown option -$OPTARG encountered."
        print $usage; exit 1;
      fi;;
    h) print $usage; exit 0;;
    i) input_queue="$OPTARG";;
    o) output_queue="$OPTARG";;
```

```

    esac
done

if [ "$input_queue" = "" ] || [ "$output_queue" = "" ]; then print
$usage; exit 1; fi

while [ 0 == 0 ]; do
    m=`message`
    echo "Sending..."
    echo "$m"
    echo "$m" | q -o $input_queue >/dev/null 2>&1
    echo "Received..."
    q -I $output_queue -w $seconds 2>/dev/null
done

```

Tip: To simulate a user application putting messages to TESTDB.IN, we used the SupportPac MA01: WebSphere MQ - Q program. This program is robust, easy-to-use, and freely available for download from:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000647&loc=en_US&cs=utf-8&lang=en

Alternatively, you could use the sample programs **amqspu**t and **amqsge**t with small modifications to the **startmessagetraffic.ksh** script. These sample programs are located in the **/usr/mqm/samp/bin** directory on AIX. For information about other operating systems, refer to the WebSphere Message Broker documentation.

We executed the test script to ensure success prior to the failure condition being generated. Executing the program on any UNIX system should produce results similar to those shown in Example 10-34.

Example 10-34 Successful output from startmessagetraffic

```
wmb@m106910f:/var/mqsi $ startmessagetraffic -i TESTDB.IN -o TESTDB.OUT
```

```

Sending...
<Test><Timestamp>13:34:13</Timestamp></Test>
Received...
<Test><Timestamp>13:34:13</Timestamp></Test>
Sending...
<Test><Timestamp>13:34:15</Timestamp></Test>
Received...
<Test><Timestamp>13:34:15</Timestamp></Test>
Sending...

```

```
<Test><Timestamp>13:34:17</Timestamp></Test>
Received...
<Test><Timestamp>13:34:17</Timestamp></Test>
Sending...
<Test><Timestamp>13:34:19</Timestamp></Test>
Received...
<Test><Timestamp>13:34:19</Timestamp></Test>
```

To cause the failure condition, we changed the password for the user ID dbuser with the commands in Example 10-35 (passwords are not displayed).

Example 10-35 Changing the password of dbuser

```
dbuser@m106910f:/home/dbuser $ passwd
```

```
Changing password for "dbuser"
dbuser's Old password:
dbuser's New password:
Enter the new password again:
```

```
dbuser@m106910f:/home/dbuser $
```

Even after we changed the password for the database, the script continued to run successfully. Not until we restarted the broker did the error condition present itself.

Important: This particular scenario illustrates a simple problem that could be very misleading. Brokers in some environments are restarted infrequently. If the exact scenario presented here occurred in production, the problem might not manifest itself for days, weeks, or months until a broker is restarted.

Imagine further a situation where a broker is only restarted after maintenance to WebSphere Message Broker is applied. Then, when the message flows are started, they begin to fail. The lesson here is to apply a problem determination methodology in all situations. In this scenario, making the assumption that the maintenance was the cause of the failure would have been a mistake. Before attempting problem resolution, be reasonably sure that the fix that you use (such as backing out maintenance or using `mqsisetdbparms`) is the correct solution.

To cause the failure simulating a restart of the broker in the future, we stopped the script and restarted the brokers. Notice in Example 10-36 that even after we started the broker, errors were still not received because the message was not yet being sent. WebSphere Message Broker does not attempt to access data sources until they are needed by a message flow.

Example 10-36 Restarting BROKER2 and startmessagetraffic.ksh

```
wmb@m106910f:/var/mqsi $ mqsistop BROKER2
```

```
BIP8071I: Successful command completion.
```

```
wmb@m106910f:/var/mqsi $ mqsistart BROKER2
```

```
WebSphere MQ queue manager running.
```

```
BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.
```

```
wmb@m106910f:/var/mqsi $ tail ../adm/user.log
```

```
Jun 19 15:30:51 m106910f user:info WebSphere Broker v6001[32270]:  
(BROKER2)[1]BIP2001I: The WebSphere Message Brokers service has started  
at version 6001; process ID 20290. : BROKER2.service:  
/build/S600_P/src/AdminAgent/ControlProcess/rios_aix_4/ImbControlService.cpp: 338: ImbControlService::StartNewAA: :  
Jun 19 15:30:52 m106910f user:info WebSphere Broker v6001[18822]:  
(BROKER2.default)[1]BIP2201I: Execution Group started: process '18822';  
thread '1'; additional information: brokerName 'BROKER2';  
executionGroupUUID '1d1220d8-0b01-0000-0080-b59c0566a828';  
executionGroupLabel 'default'; defaultExecutionGroup 'true';  
queueManagerName 'QM_BROKER2'; trusted 'false'; dataSourceName  
'MQSITCP'; userId 'wmb'; migrationNeeded 'false'; brokerUUID  
'1c1220d8-0b01-0000-0080-b59c0566a828'; filePath '/opt/IBM/mqsi/6.0';  
workPath '/var/mqsi'; ICU Converter Path '' :  
BROKER2.1d1220d8-0b01-0000-0080-b59c0566a828:  
/build/S600_P/src/DataFlowEngine/ImbMain.cpp: 351: main: :
```

```
wmb@m106910f:/var/mqsi $ startmessagetraffic -i TESTDB.IN -o TESTDB.OUT  
Sending...
```

```
<Test><Timestamp>15:37:40</Timestamp></Test>
```

```
Received...
```

```
Sending...
```

```
<Test><Timestamp>15:37:44</Timestamp></Test>
```

```
Received...
```

```
Sending...
```

```
<Test><Timestamp>15:37:47</Timestamp></Test>
```

```
Received...
```

10.5.1 Alerts and logs

Our scenario above presented a situation where a parameter on a remote resource was changed (the application database's password). The broker did not detect a problem readily, and it did not effect the environment until sometime in the future when the broker was restarted. Because the database was only used by the message flow on an as-needed basis, the problem was not reported until an alert was received.

There are at least four possible alerts that could be received from a scenario such as this:

- ▶ A current depth alert can be received on the input queue TESTDB.IN.
- ▶ The syslog reports BIP messages ending in the letter *E*.
- ▶ The application using the output queue TESTDB.OUT can report that incoming message traffic is not being received.
- ▶ The current depth alert can be received for the message flow's backout queue. In this configuration the input queue TESTDB.IN had a BOQNAME defined as TESTDB.BACKOUT (or depth alert on Dead Letter Queue of the broker's queue manager can be received when backout queue does not exist).

Monitoring detected one of the conditions above and distributed an alert. The broker administrator received the alert and started problem determination. The administrator should review the syslog immediately because it is the primary record of events for WebSphere Message Broker on UNIX.

Note: Review Event log for Windows or system console log for z/OS.

In the syslog, four error lines were found for every message that was attempted to be processed. Example 10-37 shows the relevant lines of the first occurrence of the error condition found.

Example 10-37 Relevant lines from syslog for failed connection to datasource

```
(BROKER2.default)[6426]BIP2628E: Exception condition detected on input
node 'TestDB.IN'. : BROKER2.1d1220d8-0b01-0000-0080-b59c0566a828:
/build/S600_P/src/DataFlowEngine/ImbCommonInputNode.cpp: 1070:
ImbCommonInputNode::run: ComIbmMQInputNode: TestDB#FCMComposite_1_1
(BROKER2.default)[6426]BIP2230E: Error detected whilst processing a
message in node 'TestDB.WKDB.IBMTTEST'. :
BROKER2.1d1220d8-0b01-0000-0080-b59c0566a828:
/build/S600_P/src/DataFlowEngine/ImbDatabaseNode.cpp: 295:
ImbDatabaseNode::evaluate: ComIbmDatabaseNode: TestDB#FCMComposite_1_3
```

```
(BROKER2.default)[6426]BIP2321E: Database error: ODBC return code '-1'.
: BROKER2.1d1220d8-0b01-0000-0080-b59c0566a828:
/build/S600_P/src/DataFlowEngine/ImbOdbc.cpp: 227:
ImbOdbcHandle::checkRcInner: :
(BROKER2.default)[6426]BIP2322E: Database error: SQL State '08001';
Native Error Code '-30082'; Error Text '[IBM][CLI Driver] SQL30082N
Attempt to establish connection failed with security reason "24"
("USERNAME AND/OR PASSWORD INVALID)". SQLSTATE=08001 '. :
BROKER2.1d1220d8-0b01-0000-0080-b59c0566a828:/build/S600_P/src/DataFlow
Engine/ImbOdbc.cpp: 355: ImbOdbcHandle::checkRcInner: :
(BROKER2.default)[6426]BIP2648E: Message backed out to a queue; node
'TestDB.IN'. : BROKER2.1d1220d8-0b01-0000-0080-b59c0566a828:
/build/S600_P/src/DataFlowEngine/ImbMqInputNode.cpp: 1859:
ImbCommonInputNode::eligibleForBackout: ComIbmMQInputNode:
TestDB#FCMComposite_1_1
```

10.5.2 Failing components

Determining the cause of the failure is easy from the `USERNAME AND/OR PASSWORD INVALID`. However, one error might not describe the entire problem. For example, there are at least two possible scenarios for the event in Example 10-37 on page 281:

- ▶ Either the password for the requested datasource was changed but the brokers configuration was not updated.
- ▶ The data source alias was changed to point to different remote database instance using a different password.

Before resolving which of these issues (or others) is the root cause, you must first determine the name of the data source that is accessed. Unfortunately, the name of the data source is not immediately apparent because it is not presented in the syslog.

If the message flow source (TestDB.msgflow) that is currently deployed is available, then you can review the node called WKDB.IBMTEST. If the version of the message flow that is running currently is not 100% reliable or for further confirmation, you can find the exact data source name through tracing.

10.5.3 Tracing

Using the procedure from “MustGather: Procedure for initiating an ODBC trace” on page 190, you can attempt to discover all the details of the data source connection, as shown in Example 10-38.

Example 10-38 Relevant lines from ODBC trace for failed connection to datasource

EXIT	SQLConnect with return code -1 (SQL_ERROR)
HDBC	0x3b90cc68
UCHAR *	0x300c832e [-3] "USERTCP"
SWORD	-3
UCHAR *	0x300c844e [-3] "dbuser"
SWORD	-3
UCHAR *	0xf177dd70 [-3] "*****"
SWORD	-3

The ODBC trace has searched to locate the first occurrence of a failed SQLConnect attempt. The first UCHAR is the data source that is accessed, in this example USERTCP. The second UCHAR verifies that the user ID used to access the data source is dbuser.

10.5.4 Resolving

After you determine that the dbuser password has changed recently, you can use the `mqsisetdbparms` command to alter the password for dbuser to the new value, as shown in Example 10-39.

Example 10-39 Resolving changed database password with mqsisetdbparms

```
wmb@m106910f:/var/mqsi $ mqsistop BROKER2
```

BIP8019E: Component stopped.

A previous command was issued to stop this component or it has never been started.

This component may be started, changed or deleted.

```
wmb@m106910f:/var/mqsi $ mqsisetdbparms BROKER2 -n USERTCP -u dbuser -p password
```

BIP8071I: Successful command completion.

```
wmb@m106910f:/var/mqsi $ mqsistart BROKER2
```

WebSphere MQ queue manager running.

BIP8096I: Successful command initiation, check the system log to ensure that the component started without problem and that it continues to run without problem.

Message Brokers Toolkit update without internet access

This appendix describes how to update Message Brokers Toolkit V6.0 when you do not have Internet on the Toolkit machine. This type of update might be useful for an environment (especially for a production environment) where Internet access is not allowed for security reasons.

This procedure consists of two main steps:

- ▶ Downloading update packages from the Internet to the allowed computer
- ▶ Installing updates from a local file system to the Toolkit computer

We describe two types of updates in this chapter:

- ▶ Installing an interim fix
- ▶ Installing a fix pack or refresh pack

Installing an interim fix

Follow these installation steps to apply interim fixes to your Message Brokers Toolkit:

1. On your local drive, create a temporary empty folder (*wbimbv60*) which will contain all the interim fixes for Message Brokers Toolkit V6.0 that are sitting on the IBM Support site.
2. Connect to the update FTP site using a Web browser:
<ftp://ftp.software.ibm.com/software/mqseries/fixes/wbimbv60>
3. Download the content of the *wbimbv60* folder to the local *wbimbv60* folder.
4. Note the presence of the *policy_messagebroker.xml* and *policy_eventbroker.xml* files in this folder.
5. Use appropriate local procedure (FTP, CD, Memory Key, or other protocol) to transfer your local copy of the *wbimbv60* folder to the workstation that you want to upgrade.
6. Shutdown the Toolkit and start the IBM Rational Product Updater.
7. From the Rational Product Updater, apply fixes for the Toolkit as follows:
 - a. From the menu, select **Preferences** → **Update Sites**.
 - b. Browse for the *wbimbv60* folder, select the appropriate policy file (for WebSphere Message Broker or WebSphere Event Broker), and click **OK**.
 - c. Select the Installed Products tab and click **Find Updates**.
 - d. Select the displayed items and click **Install Updates**.
 - e. Wait until installation of the fixes completes.
 - f. Use the same dialog (**Preferences** → **Update Sites**) and select **Restore Default**.
8. Shutdown the Rational Product Updater and start the Toolkit.

How to speed-up the process significantly (advanced topic)

Each fix specifies independently whether an Eclipse initialization is required. The Rational Product Updater has no capability to perform a single initialization after applying multiple fixes. You can workaroud this limitation — and save about 20 minutes per interim fix — by modifying a value in the fix properties file. Follow these steps:

1. On your local copy of the *wbimbv60* folder, locate all *update.properties* files. If the file contains the line `initWorkbench=true`, simply remove this line with a standard text editor.

2. Proceed with applying the fixes by continuing with step 6 on page 286. Rational Product Updater will not cause an Eclipse initialization for each fix.
3. You have to use the **-clean** option for the first Toolkit start after you install the fix. Otherwise, the fix will not take effect. Locate the desktop shortcut that you use to start the Toolkit, copy it, and add the **-clean** option.

Installing a fix pack or refresh pack

Note: The following steps are written for Message Brokers Toolkit Fix Pack 6.0.0.1, because it was the latest fix pack at the time of writing. However, you can use similar steps for new fix packs or refresh packs. Refer to online documentation *Install instructions for Message Brokers Toolkit only* under the actual Toolkit fix pack bookmark on the standard IBM Support site:

<http://www.ibm.com/support/docview.wss?rs=849&uid=swg27006041>

To apply Message Brokers Toolkit Fix Pack 6.0.0.1 via Rational Product Updater, you need to first update your Rational Product Updater to 6.0.1 level, because Message Brokers Toolkit Fix Pack 6.0.0.1 is based on Rational Application Developer 6.0.1.

Important: The procedure that we describe here is intended for computers where only Message Brokers Toolkit V6.0 is installed.

If you have a shell-sharing environment (your Message Brokers Toolkit shares the computer with either WebSphere Integration Developer, Rational Application Developer, or Rational Software Architect), you have to do additional steps and updates before applying Message Brokers Toolkit Fix Pack 6.0.0.1. In this case, refer to the online documentation *Install instructions for toolkit plus other products* and *Instructions for creating a local RPU mirror* on the IBM Support site:

<http://www.ibm.com/support/docview.wss?rs=849&uid=swg24011391>

Follow these installation steps to update your Message Brokers Toolkit to Fix Pack 6.0.0.1:

1. Refer to online documentation *Installing IBM WebSphere Message and Event Broker Toolkit Fix Pack 6.0.0.1* on the IBM Support site:

<http://www.ibm.com/support/docview.wss?rs=849&uid=swg27007325>

2. Download the Rational Product Updater fix pack 6.0.1 from:
ftp://ftp.software.ibm.com/software/rationalsdp/updater/60/zips/rpu_601.zip
3. Follow the installation instructions from step 1 and update Rational Product Updater to 6.0.1 level:
 - a. Edit the policy_601.xml file and write the correct path into URL location.
 - b. From the menu, select **Preferences** → **Update Sites**.
 - c. Browse for the policy_601.xml in the extracted package, and click **OK**.
 - d. Select the Installed Products tab and click **Find Updates**.
 - e. Select **OK** to continue installation.
 - f. Wait until the installation of the Rational Product Updater Fix Pack completes.
4. Download the Message Brokers Toolkit Fix Pack 6.0.0.1 from the IBM Support site:
<http://www.ibm.com/support/docview.wss?rs=959&uid=swg27006041>
5. Extract the saved zipped file to a directory on the local system.
6. Ensure that the Message Brokers Toolkit is shutdown.
7. From the Rational Product Updater, apply the update for the Toolkit:
 - a. From the menu, select **Preferences** → **Update Sites**.
 - b. Browse for the policy.xml file in the extracted package, and click **OK**.
 - c. Select the Installed Products tab and click **Find Updates**.
 - d. Select the displayed items and click **Install Updates**.
 - e. Wait until the installation of the Message Brokers Toolkit Fix Pack completes.

When you have completed all of these steps, you should have the following entries in your RPU Installed Products panel:

- ▶ IBM WebSphere Message Brokers Toolkit 6.0.0.1
- ▶ J2EE Connector Tools 6.0.1.2

For detailed instructions and the required prerequisites and how to obtain them, refer to the *WebSphere Message Broker V6.0 Installation Guide*, which is available at:

<http://www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US&FNC=SRX&PBL=GC34-6621Th>

Screen samples from IBM Tivoli Enterprise Portal

The figures in this appendix illustrate the IBM Tivoli Enterprise™ Portal user interface and explain briefly the IBM Tivoli Omegamon functionalities that you can use for managing and monitoring WebSphere Message Broker and WebSphere MQ.

For further information about how to use IBM Tivoli Omegamon in a WebSphere Business Integration environment, refer to *Implementing IBM Tivoli OMEGAMON XE for WebSphere Business Integration V1.1*, SG24-6768, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg246768.html?Open>

Main product console window

Figure B-1 shows the main IBM Tivoli Enterprise Portal window.

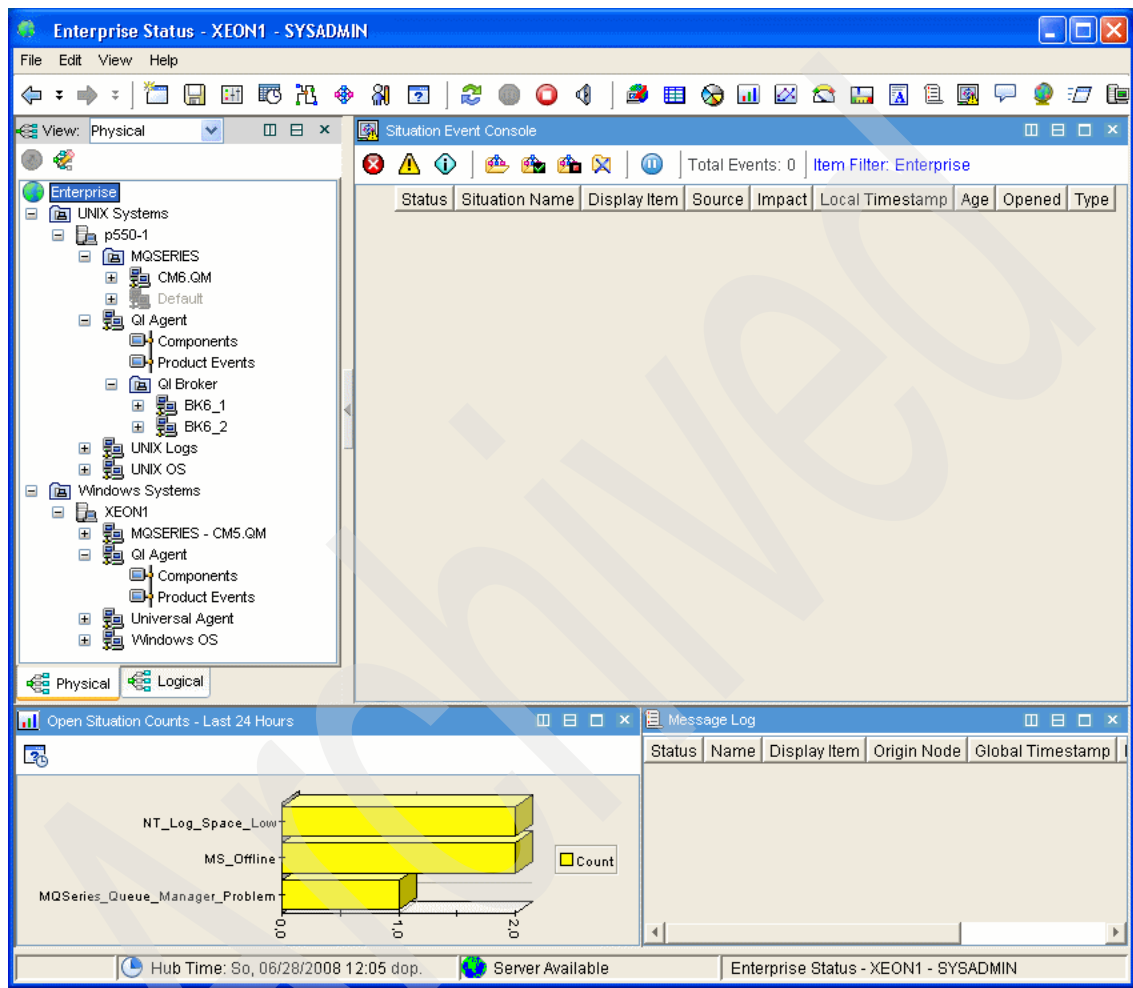


Figure B-1 Main IBM Tivoli Enterprise Portal window

This main window has four standard views:

► Navigator view (upper left)

The Navigator view shows all managed systems and components and is used to change window content for a selected component.

In our example, the navigator view include the following components:

- AIX system *p550-1*
- Windows system *XEON1*
- queue managers *CM6.QM* and *CM5.QM*
- brokers *BK6_1* and *BK6_2*

In this view, you can see the operating system logs and agents also (UNIX Logs, UNIX, Universal Agent, Windows).

Note: The items *QI Agent* are the Tivoli Omegamon monitoring components for WebSphere Message Broker. You can find all broker domain components under these *QI Agent* items.

► Situation Event Console (upper right)

This view lists each alert for situations that are associated with a Navigator item. A situation must be associated with a Navigator item before an alert can show here. Association is automatic for predefined situations in the Navigator Physical view and for situations that are edited when the Situation editor was accessed through the Navigator item menu with the current Navigator item and any other Navigator items on this branch. The console has a toolbar for filtering the view to show only the alerts that you want to see and a menu with items for managing alerts.

► Open Situation Counts (lower left)

This view displays the summary statistics for the latest alerts.

► Message Log (lower right)

This view provides an overview of changes in a situation status on your monitored network. It displays a row of data for each status change of situations distributed to managed systems.

Broker related tasks

To see the managed brokers, click the *QI Agent* item. Figure B-2 shows three message broker components that run on the AIX system: the Configuration Manager *CM6* and the brokers *BK6_1* and *BK6_2*. The *QI Agent* displays useful information such as the status, Queue manager name, product version, operating system version, process name, date when the component was started, and UUID. The Product Events view shows configured alerts that are related to the message broker components on this AIX system.

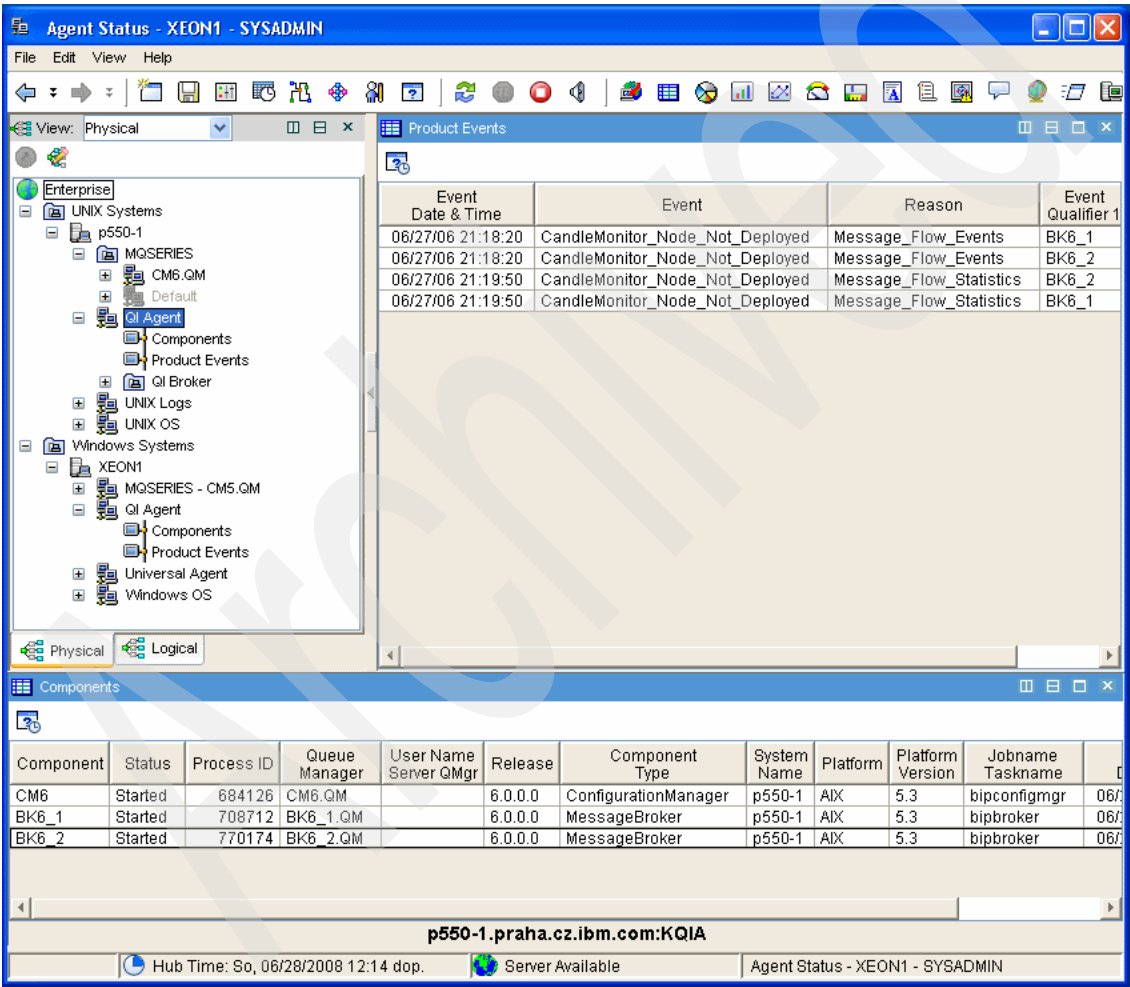


Figure B-2 Message broker components view

Figure B-3 shows the Execution Group Information view. Information about parameters for execution groups of the selected broker (*BK6_1* in our example) are displayed in the lower section, including trace informations. The Enterprise Event Log displays the alerts, if any, for these execution groups.

The screenshot displays the 'Execution Group Information - XEON1 - SYSADMIN' window. The left pane shows a tree view with 'BK6_1' selected under 'Product Events'. The right pane shows the 'Enterprise Event Log' with columns: Status, Name, Display Item, Origin Node, Global Ti. The bottom pane shows the 'Execution Group Information' table.

Execution Group	Total Msg Flows	Started Msg Flows	User Trace Level	User Trace Filter	User Trace Log File Size	User Trace Log File Mode	Trace Level	Trace Filter	Trace Log File Size	Trace Log File Mode	Event Fil
default	4	3	None	none	4194304	Safe	None	none	4194304	Safe	

Below the table, the selected broker is identified as **BK6_1::KQIB**. The status bar at the bottom shows 'Hub Time: So, 06/28/2008 12:10 dop.', 'Server Available', and the window title 'Execution Group Information - XEON1 - SYSADMIN'.

Figure B-3 Execution group view

Figure B-4 shows the Message Flow Information view. Information about the parameters for message flows of selected broker (*BK6_1* in our example) are displayed in the lower section, including transaction and trace informations. The Enterprise Event Log displays the alerts, if any, for these message flows.

Message Flow Information - XEON1 - SYSADMIN

View: Physical

Enterprise Event Log

Execution Group	Message Flow	Msg Flow Type	Status	Processing Nodes	Additional Instances	Commit Count	Commit Interval	Coordinated Transaction	User Trace Level	User
default	TestDB	User	Started	3	0	1	0	No	None	nc
default	Test_basic	User	Stopped	2	0	1	0	No	None	nc
default	ConfigurationMessageFlow	System	Started	4	0	1	5	No	None	nc
default	PubSubControlMsgFlow	System	Started	6	0	1	5	No	None	nc

BK6_1::KQIB

Hub Time: So, 06/28/2008 12:11 dop. Server Available Message Flow Information - XEON1 - SYSADMIN

Figure B-4 Message flow view

Figure B-5 illustrates how to perform an action for a selected message broker component . The steps are the same for all managed objects, either WebSphere Message Broker or other components (WebSphere MQ, operating systems, or logs):

1. Choose the component or object for the action (in any available views).
2. Right-click the chosen component.
3. Click **Take Action** → **Select** to display the Take Action window.
4. Choose the action to perform on the component.

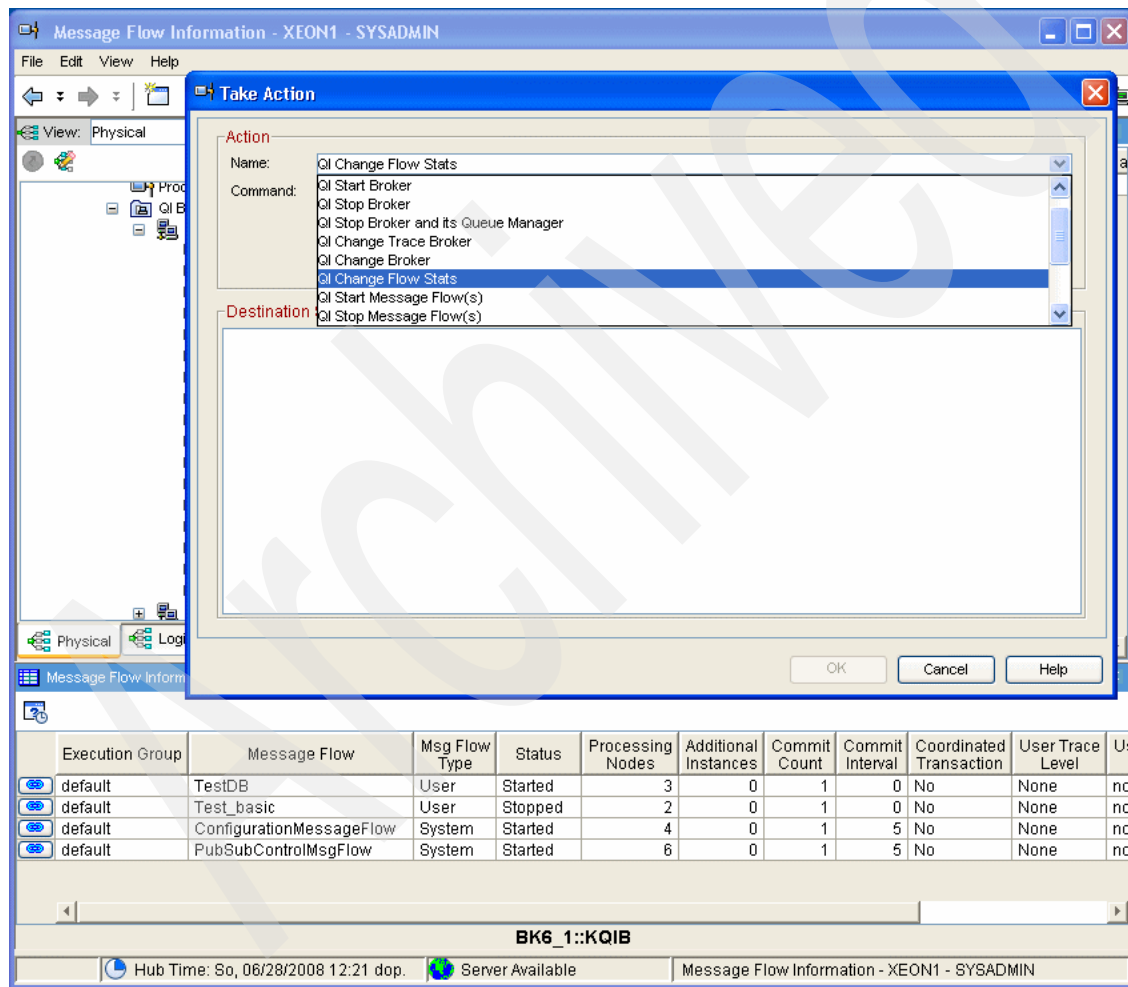


Figure B-5 Broker actions

Queue manager related tasks

The IBM Tivoli Omegamon is also able to monitor WebSphere MQ components. With the Error Log view, you can show and process queue manager error log records (Figure B-6).

Error Log - XEON1 - SYSADMIN

View: Physical

Enterprise

- UNIX Systems
 - p550-1
 - MQSERIES
 - CM6.QM
 - Channel Definitions
 - Channel Performance
 - Cluster Queue Manager
 - Dead-Letter Queue Messages
 - Error Log**
 - MQSeries Events
 - Queue Definitions
 - Queue Manager Status
 - Queue Statistics
- Default
 - QI Agent
 - Components

Physical Logical

Take Action

Action

Name: <Select Action>

Command:

Arguments...

Destination Systems

Run

Error Log

Log Date & Time	Message ID	Message Text	Explanation	User Action
06/27/06 21:12:53	AMQ90...	Channel 'BK6_2.TO.CM6' is starting.	Channel 'BK6_2.TO.CM6' is starting.	None. BK
06/27/06 21:12:53	AMQ90...	Channel 'BK6_1.TO.CM6' is starting.	Channel 'BK6_1.TO.CM6' is starting.	None. BK
06/27/06 22:52:53	AMQ95...	Disconnect interval expired.	Channel 'BK6_2.TO.CM6' closed because no messages arr...	None. BK
06/27/06 22:52:53	AMQ90...	Channel 'BK6_2.TO.CM6' ended normally.	Channel 'BK6_2.TO.CM6' ended normally.	None. BK
06/27/06 22:52:53	AMQ95...	Disconnect interval expired.	Channel 'BK6_1.TO.CM6' closed because no messages arr...	None. BK
06/27/06 22:52:53	AMQ90...	Channel 'BK6_1.TO.CM6' ended normally.	Channel 'BK6_1.TO.CM6' ended normally.	None. BK

CM6.QM::MQ

Hub Time: So, 06/28/2008 12:16 dop. Server Available Error Log - XEON1 - SYSADMIN

Figure B-6 Error Log view

Figure B-7 shows the Queue Statistics view which displays, for example, queue open informations, queue depth, triggering information, put and get statistics, and queue creation date.

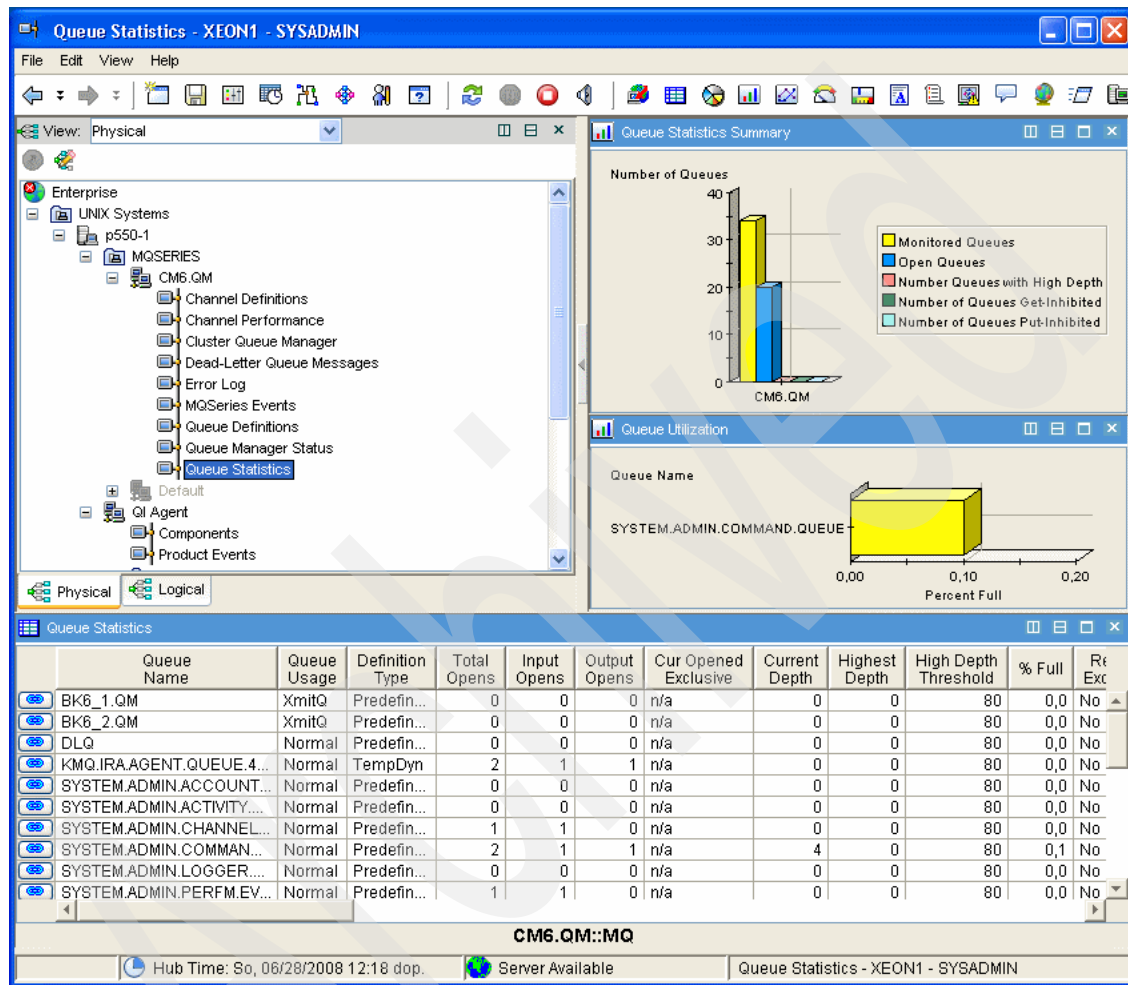


Figure B-7 Queue Statistics view

Web-based portal

You can also access the Tivoli Enterprise Portal from a Web browser. You can view the same information in the Web-based portal window (Figure B-8) as in the Web-based message flow view (Figure B-9).

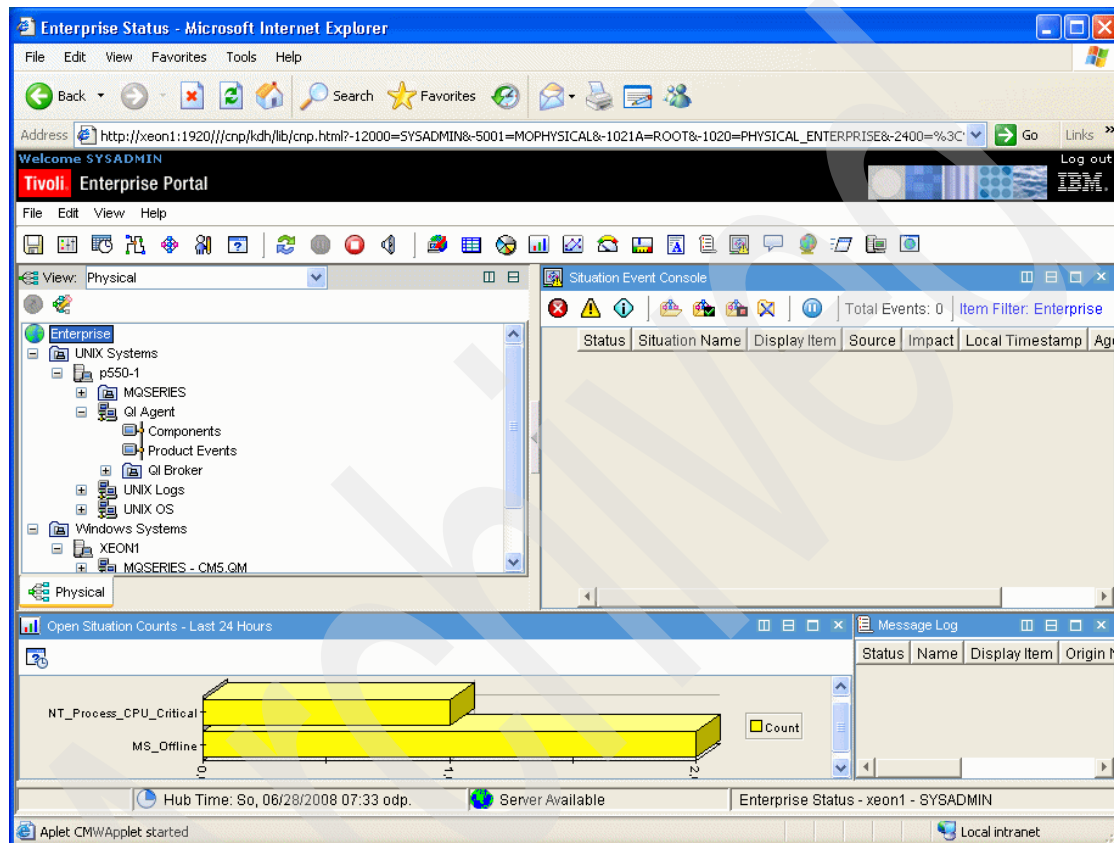


Figure B-8 Web-based portal main window

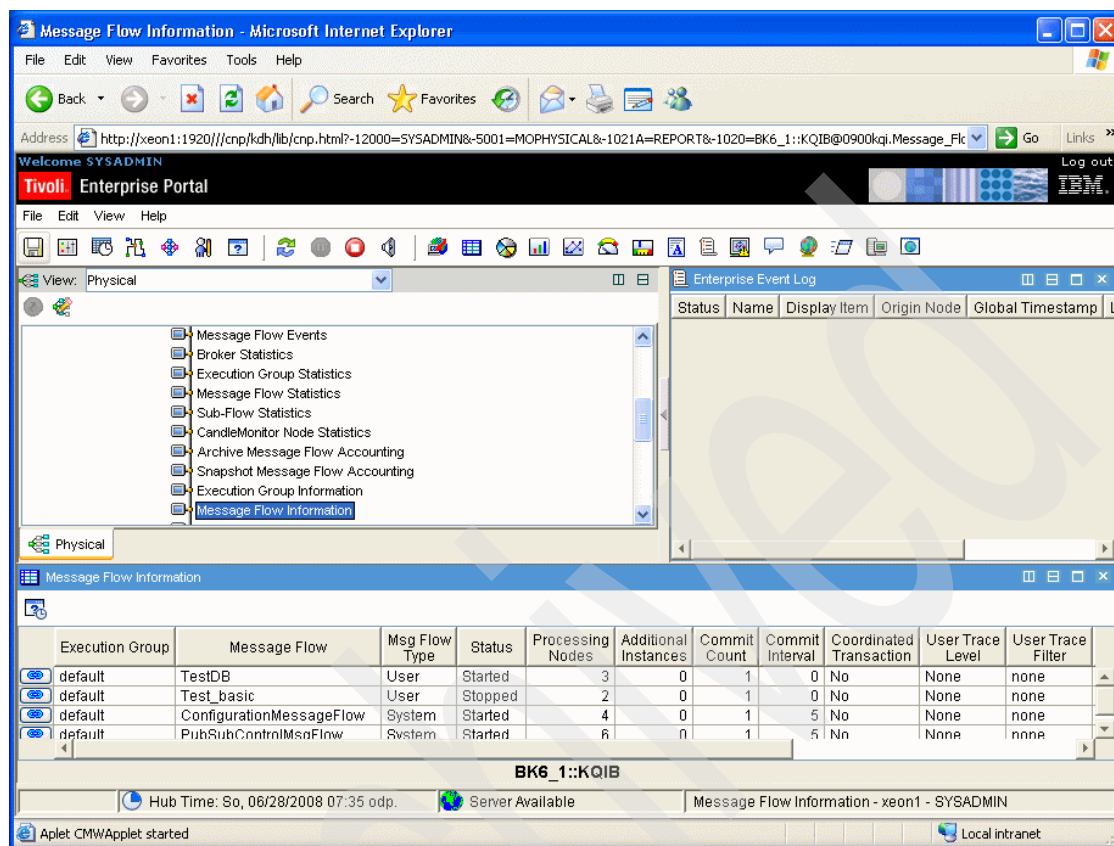


Figure B-9 Web-based message flow view

Additional material

This appendix describe the additional material to which this redbook refers that you can download from the Internet.

Locating the Web material

The Web material that is associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247283>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the redbook form number, SG247283.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
Ch10 all.zip	All chapter 10 scenario zipped samples
Ch10.2_WmbDeployTool.zip	Chapter 10.2 scenario zipped samples
Ch10.3_HTTPS_MessageFlows.zip	Chapter 10.3 scenario zipped samples
Ch10.5_DBflows.zip	Chapter 10.5 scenario zipped samples

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zipped file into this folder.

Abbreviations and acronyms

ACL	Access Control Lists
API	application program interface
BITS	Birla Institute of Technology
CA	Certification Authority
CMP	Configuration Manager Proxy
CMPAPI	Configuration Manager Proxy API
CWF	Custom Wire Format
ESB	Enterprise Service Bus
FDCs	First Failure Symptom logs
GUI	Graphical User Interface
HA	High availability
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MAC	Message Authentication Code
MQI	Message queue interface
OAM	Object Authority Manager
PMR	Problem Management Record
RAD	Rational Application Developer
RSA	Rational Software Architect
SAN	Storage Area Network
SLA	Service level agreements
SOA	service-oriented architecture
SSL	Secure Sockets Layer
TCB	Task Control Block
UDPs	User define properties
UML	Unified Modeling Language

Related publications

In this section, we list the publications that we consider particularly suitable for a more detailed discussion of the topics that we cover in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 305. Note that some of the documents that we reference here might be available in softcopy only.

- ▶ *WebSphere Message Broker Basics*, SG24-7137
- ▶ *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198
- ▶ *WebSphere MQ Integrator Deployment and Migration*, SG24-6509
- ▶ *Messaging Solutions in a Linux Environment*, SG24-6336
- ▶ *WebSphere Adapter Development*, SG24-6387
- ▶ *WebSphere MQ V6 Fundamentals*, SG24-7128
- ▶ *High Availability z/OS Solutions for WebSphere Business Integration Message Broker V5*, REDP-3894

To locate operating system tuning recommendations, review the *WebSphere Message Broker Installation Guide*, which is available at:

<http://www.ibm.com/software/integration/wbimessagebroker/library>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

32-bit and 64-bit support 66

A

abend files 207
 core files, analyzing 31, 207
acceptance environment 18
access control list 70
accounting and statistics 26
advanced database diagnostics 31
advanced steps 192
AIX 64
AIX error log 174
alerting 138
alerts and logs 281
Alerts view 166
alerts view 215
algorithm, Message Authentication Code 263
Ant
 ant.jar file 225
 using 223, 225–226, 229
archiving 138
audience for this book 4
authentication 84
 JMS client 99
 Real-time node 84
authentication protocol
 configuring 85
authorization 84, 89
automation 20–21, 48, 51
availability 21, 23
availability and failure events
 monitoring systems 139

B

backing up a broker 119
backing up the Configuration Manager 123
backup 22, 115, 217
basic parameters 141
basic steps 164
best practices 41
BIP message 165

broker 10–11, 26, 28, 32, 36, 47, 62, 110, 119, 142, 146, 161
 archive files 23
 backing up 119
 configuring 97
 configuring to use SSL on a particular port 101
 database 23, 38
 implementing HTTPS support 238
 problem determination 211
 recovering after queue manager fails 122
 recovering when backup not available 121
 re-creating when backup not available 271
 related tasks 292
 restoring from backup 120, 268
 tuning considerations 147
broker domain
 managing 18
 managing different domains 81
buffer pools 39
bullet proofing message flows 54
business database 38
business relevance 5

C

capability and environments
 7–8
Certificate Authority 234
 adding on queue manager 252
 configuring using OpenSSL 235
 installing the certificate 239
change management 222
Cloudscape 181
clustering 59
CMPAPI Exerciser 273
coexistence 65
collecting data 181
command
 db2profile 69
 dspmq 150
 errpt 174
 keytool 100
 mqsdchangebroker 71, 98, 110, 143
 mqsdchangeconfigmgr 70–71

- mqsicchange flowstats 26
- mqsicchange properties 36, 101, 106, 205, 242
- mqsicchange trace 36, 148, 192
- mqsicchange usernameserver 72, 112
- mqsiccreate aclentry 93, 113
- mqsiccreate broker 71, 90–91, 98, 110, 275
- mqsiccreate configmgr 71
- mqsiccreate executiongroup 69
- mqsiccreate usernameserver 112
- mqsideploy (UNIX) 49
- mqsideploy.bat (Windows) 49
- mqsilist 190
- mqsi reporttrace 148
- mqsiservice 70
- mqsisetdbparms 276, 283
- mqsisstart 90, 166, 271, 276
- mqsisstop 167
- wmb 76
- Command Assistant wizard 20, 75
- command line
 - messages 167
- complementary options 107
- components
 - controlling 51
 - description and function 7, 11
- components update 158
- Configuration Manager 10–12, 23, 29, 32, 69, 111, 123, 150, 161, 261
 - backing up 123
 - connection problem 210
 - enabling SSL encryption 261
 - problem determination 212
 - Proxy (CMP) 12
 - Proxy API 12, 20, 49
 - Proxy API tracing 31
 - recovering after queue manager fails 126
 - recovering when backup not available 125
 - restoring from backup 124
 - using Proxy API tracing 199
- configuring a broker 97, 101
- configuring the Certificate Authority 234
- connection problem, Configuration Manager 210
- controlling components 51
- Create an Execution Group wizard 68

D

- data for further support
 - collecting 181

- database 37, 69
 - backup options 116
 - trace 37
 - tuning considerations 149
- database, broker 23, 38
- DB2 Universal Database 181
- decision making 28
- deploy procedure
 - executing 232
- description information, message flow 45
- development environment 17
- diagnostic messages
 - in messages.html file 177
 - in the Information Center 176
- diagnostics
 - database and ODBC 31
- disaster recovery 119
- domain
 - administering 20
 - backing up 116
 - components 219
 - managing changes 19, 44
 - managing problems 20, 50
 - monitoring 25, 139
 - restoring and recovering 118
 - securing 84
- domain updates
 - maintaining 154

E

- enterprise service bus 16
- environment
 - acceptance 18
 - development 17
 - production 18, 50
 - testing 18
- environment variables
 - setting using MQSIJVERBOSE 203
- environments
 - overview 16
- errors
 - categorizing 165
 - investigating ODBC 210
 - problem determination 30, 163, 217, 276
- ESQL 19
- ESQL design 62
- ESQL files 23
- event message 165

- structure 165
- types 166
- event viewer
 - application log 168
 - log properties 169
 - system log 169
- execution groups 32, 47, 66
 - multiple instances 36
- Extended Structured Query Language. *See* ESQL.

F

- failing components 282
 - identifying 164
- fault-tolerance 53
- files
 - mapping 23
- fix packs 154
 - levels 154
 - planning for installation 28
 - production fix pack update 29
 - refresh pack installing 287
 - reviewing and feasibility study 28
 - testing 29
- fixes 154
- freezing period 159

G

- groups considerations 90

H

- HA 54
 - for databases 56
 - for WebSphere Message Broker 57
 - for WebSphere MQ 56
- health-check 23, 25, 135–136
- high availability. *See* HA.
- horizontal scalability 59
- HP-UX 64
- HTTPRequest node to use SSL
 - configuring 102
- HTTPS support, broker 238

I

- IBM
 - Contact us xviii
 - Redbooks Web site 305
 - Software Support 178

- RSS feeds 178
- Support Center 31
- Tivoli Enterprise Portal 289
- Tivoli OMEGAMON 145
- IBM Java Runtime Environment 76
- Information Center 176
- installation logs 179
- interim fix
 - installing 286

J

- Java
 - keystore 261
 - keytool 259
- Java Virtual Machine. *See* JVM.
- JRE heap values 76
- JVM 78
- JVM footprint
 - changing 76
 - maximizing 76
 - minimizing 79
- JVM heap size parameter 69
- JVM maximum heap size 80
- JVM minimum heap size 65
- JVM tracing 31
 - using 203

K

- key store file
 - creating 100
- key store of queue manager
 - creating 249
- keystore 261
- keytool 259

L

- large messages
 - considerations for handling 62
- Linux 64, 81
- log and data I/O time 37
- logical thought process 30

M

- main product console window 290
- maintaining domain updates 154
- maintenance 27
- maintenance strategy 27, 153

- maintenance window 159
 - managing different broker domain 81
 - managing problems in the domain
 - problem discovery 50
 - resolution 50
 - return-to-service 51
 - root-cause analysis 50
 - managing the broker domain 18
 - mapping files 23
 - Message Authentication Code (MAC) algorithm 263
 - message broker
 - modifying properties 242
 - message broker component
 - characteristics 109
 - message broker components
 - characteristics 62, 119, 146, 161, 211
 - message broker keystore
 - installing the CA certificate 239
 - Message Brokers Toolkit 11, 13, 29, 69, 180, 259, 266
 - enabling SSL 264
 - event log 213
 - generating the certificate for 259
 - problem determination 213
 - updating without Internet access 285
 - message enrichment 8
 - message flow 48
 - bullet proofing 54
 - debugging 211
 - files 23
 - message processing nodes 32
 - modifying 243
 - properties 45
 - setting description information 45
 - message protection 106
 - message routing 8
 - message sets
 - definition files 23
 - message flows 33
 - message transformation 8
 - monitor and health-check 23, 25, 135
 - monitoring and health-checking 136
 - monitoring the domain 25, 139
 - MQSIJVERBOSE 203
 - MRM logs 181
 - MustGather
 - procedure for taking startup trace of a broker 184
 - procedure to trace mqsi* command executables 189
 - mutual challenge-response password authentication 87
- N**
- naming conventions 31
- O**
- ODBC
 - diagnostics 31
 - drivers for Cloudscape 181
 - drivers for DB2 Universal Database 181
 - investigating errors 210
 - open handle cache 36
 - Open SSL 235
 - optimizing resources 144
- P**
- password authentication 86
 - performance and reliability
 - monitoring systems 140
 - performance tuning 137
 - pop-up messages 215
 - prerequisites 5
 - problem determination 30, 163, 217, 276
 - problem determination, broker 211
 - problem determination, Configuration Manager 212
 - problem discovery 50
 - Problems view 166
 - production environment 18, 50
 - Properties view 68
 - properties, message flow 45
 - properties, modifying message broker 242
 - Proxy API 12, 20, 31, 49, 199
 - publish/subscribe 8
 - pwgroup.dat file 72–73
- Q**
- queue manager 33, 126, 141
 - log configuration 35
 - related tasks 296
 - tuning considerations 149
 - queue manager failure, recovering 122
 - queue manager logging
 - hardware capability 34
 - queue manager's certificate

generating 255

R

RAS

implementing 21, 53

Rational Application Development platform 13

Rational Product Updater 162

recommended considerations 31

recover 22, 115, 217

recovery

from disaster 119

from queue manager failure 122

when backup not available 121

Redbooks Web site 305

refresh packs 154

reliability, availability, and scalability. *See* RAS.

repeatable procedure for bar deploys

creating 222

reporting 138

resolution 50

resource management 19, 43, 217, 222

restore 22, 115, 217, 268

a broker from backup 120, 268

Configuration Manager 124

from backups 119

the domain 118

return-to-service 51

root-cause analysis 50

RSA

algorithm 259

runtime 63

objects ACLs 91

topologies 61

S

sample passwords file 98

sample programs

amqsget 278

amqspout 278

sample script for stop procedure 52

scalability 21, 58

horizontal 59

vertical 58

scenario

environment setup 218

scheduling and expiration 116

securing the domain 84

security 21, 83, 217, 234

exits 88

topic-based 95

Service Level Agreement (SLA) 23

Service Level Agreements (SLA) 136

service level agreements (SLA) 48

service tracing

using 199

service-oriented architecture (SOA) 16

simple telnet-like password authentication 86

single data source

using 65

SOA 16

Solaris 64

SSL

authentication for the HTTP listener 88

encryption 261

Java system properties 264

on HTTP listener 100

on Real-time nodes 96

on WebSphere MQ Java Client 104

properties 242

Secure Sockets Layer (SSL) 96

symmetric and asymmetric authentication 87

testing the connection 266

SSL (Secure Sockets Layer) 96

SSL encryption 261

staging updates 156

stop procedure script 52

strategy

maintenance 27, 153

SupportPac MA01

WebSphere MQ - Q 278

Syslog configurations 39

system and components logs 25

system log 169

system tracing 31

T

technical skills 30

test and acceptance 18

test and acceptance environment 18

test programs

using 146

testing the flows 245

Toolkit 13, 75, 113, 129, 151, 162

backing up resources 129

connection encryption 248

restoring resources from backup 131

- restoring resources from local history 132
- tools 145
- topology
 - logical 218
 - physical 220
 - WebSphere MQ components 33
- tracing 36, 192, 282
 - JVM 31
 - service 31
 - system 31
 - user 31
- transport layer
 - securing 84
- trusted broker 36
- trusted channel and listener 34
- tuning 137, 147, 149
- tuning considerations 149
- tuning considerations, broker 147

U

- UDPs 19, 46
- UML (Unified Modeling Language)
 - Eclipse-based 13
- Unified Modeling Language (UML) 13
- UNIX 71
- UNIX syslog 170
 - configuration 170
 - understanding events 172
- update procedure 158
- updates, staging 156
- updating the domain 154
- User Defined Properties. *See* UDPs.
- User Name Server 10–11, 13, 23, 29, 33, 47, 70, 111, 127, 150, 162
 - backing up 127
 - configuring 97, 112
 - problem determination 212
 - recovering after failure 128
 - recovering after queue manager fails 128
- user tracing 31
 - altering from the mqsischangetrace command 193
 - using 192
- user.log
 - structure 172

V

- version and description

- setting 45
- version levels 155
- versioning 19, 44
- vertical scalability 58
- view table and indexes
 - creating 38

W

- Web-based portal 298
- WebSphere Message Broker 1, 5
 - Accounting and Statistics
 - using 145
 - capabilities description 8
 - environment 9, 15, 17
 - features 7
 - HA 57
 - Information Center 63, 71
 - resources 55
 - runtime 179
 - V6.0 5, 16, 46
 - V6.0 environment 9
- WebSphere MQ 64, 261
 - 180
 - clustering 59
 - component attributes 26
 - Extended Security Edition 108
 - Object Authority Manager 108
 - SSL 107
- Windows 71, 81
- Windows event viewer 168
- wmbadmin.log file 53
- WmbDeployTask.java
 - designing and developing 224

X

- XML schema files 23



Redbooks

Managing WebSphere Message Broker Resources in a Production Environment

Discover best practices for administering the Message Broker

Learn about security, backup, and problem determination

Use sample scenarios and scripts

This IBM Redbook provides guidance on how to manage WebSphere Message Broker V6.0 and its resources in a production environment.

It begins with an overview of the WebSphere Message Broker V6.0 architecture, components, and features. These components play a key role in the integration of disparate applications and platforms by providing functional and transport capabilities that support and facilitate enterprise-level business integration. The book describes how to manage and to administer WebSphere Message Broker V6.0 appropriately to achieve maximum stability, availability, and performance in a production environment.

This book provides extensive guidance and instructions for various administration tasks for the message broker domain, such as, resource management, security, backup and recover, monitor and health check, maintenance strategy, and problem determination. It covers all the main message broker domain components, such as the Configuration Manager, the broker, the User Name Server, and the Message Brokers Toolkit. It also discusses underlying components such as WebSphere MQ queue manager and the database that is required to run the message broker and notes all generic- and platform-specific considerations.

Finally, this book includes sample scenarios that demonstrate important tasks that complement the discussions included in this book.

**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



Managing WebSphere Message Broker Resources in a Production Environment

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages

