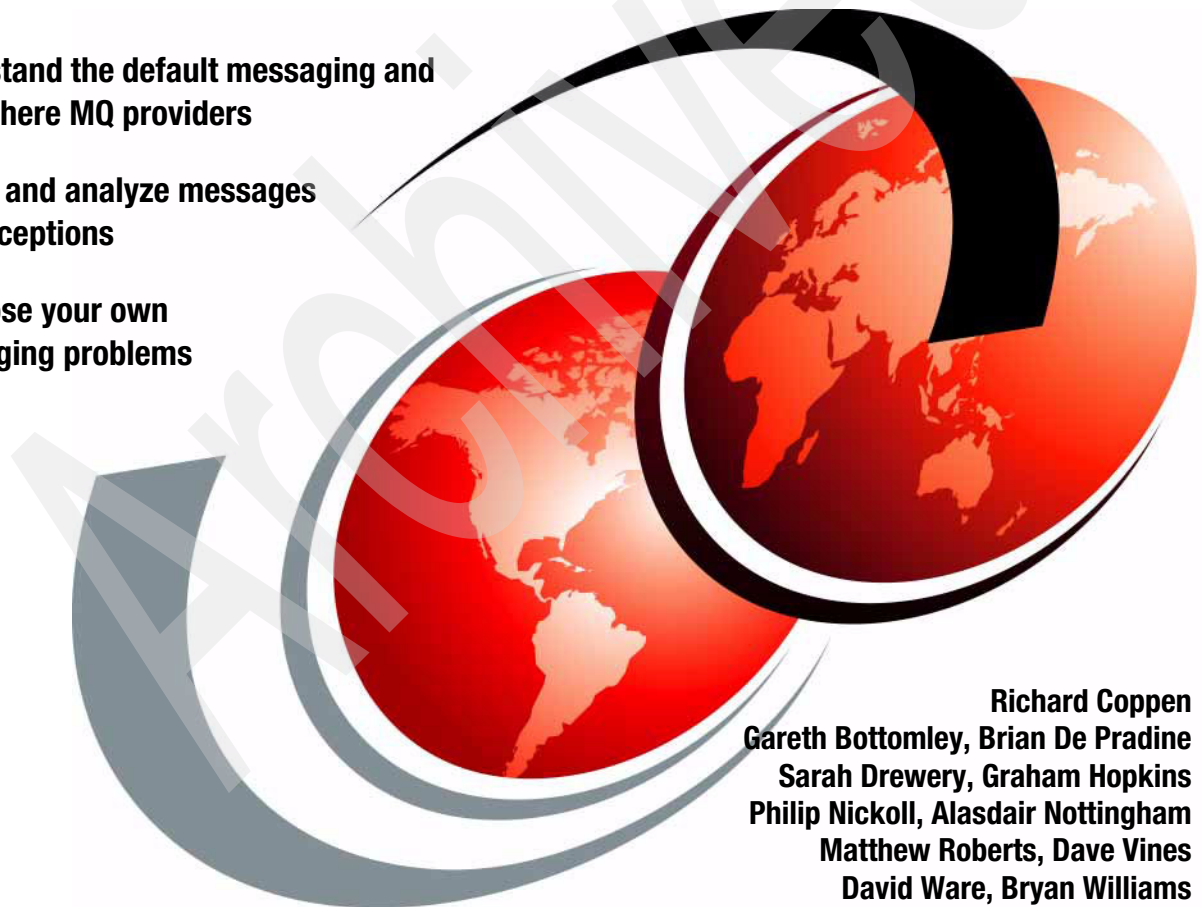


# WebSphere Application Server V6.1: JMS Problem Determination

Understand the default messaging and WebSphere MQ providers

Locate and analyze messages and exceptions

Diagnose your own messaging problems



Richard Copen  
Gareth Bottomley, Brian De Pradine  
Sarah Drewery, Graham Hopkins  
Philip Nickoll, Alasdair Nottingham  
Matthew Roberts, Dave Vines  
David Ware, Bryan Williams





International Technical Support Organization

**WebSphere Application Server V6.1: JMS Problem  
Determination**

September 2007

Archived

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

**First Edition (September 2007)**

This edition applies to WebSphere Application Server V6.1.

**© Copyright International Business Machines Corporation 2007. 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

<b>Notices</b> .....	.xi
Trademarks .....	xii
<b>Preface</b> .....	xiii
The team that wrote this paper.....	xiii
Become a published author .....	xv
Comments welcome .....	xv
<b>Chapter 1. Introduction to JMS application problem determination</b> .....	1
1.1 Identify your messaging provider type.....	2
1.1.1 Default messaging provider .....	2
1.1.2 WebSphere MQ as the JMS provider .....	2
1.1.3 Default messaging provider interoperation with WebSphere MQ .....	2
1.2 Default messaging provider problem determination .....	3
1.2.1 Identify your messaging engine topology .....	3
1.2.2 Determine the messaging engine connected to an application.....	7
1.2.3 Determine your messaging engine status .....	8
1.2.4 Finding queued messages .....	9
1.2.5 Finding lost messages .....	17
1.3 Collect diagnostics .....	19
1.3.1 Collecting JVM logs.....	20
1.4 Guide to chapters .....	21
1.5 Guide to chapters by message .....	22
1.6 Guide to chapters by exception.....	28
1.7 Guide to chapters by symptom .....	30
<b>Chapter 2. Messaging engine problem determination</b> .....	35
2.1 Introduction .....	36
2.1.1 Symptoms of a messaging engine problem .....	36
2.2 Verify the messaging engine is started .....	36
2.3 Analyze diagnostics.....	37
2.3.1 Analyze SystemOut log for the messaging engine .....	38
2.3.2 Analyze the application logs .....	40
2.4 Causes and solutions .....	41
2.4.1 Service integration port conflicts .....	42
2.4.2 Incorrect Control Region Adjunct (CRA) ownership (z/OS) .....	42

2.5 Validate the solution . . . . .	44
<b>Chapter 3. Message store problem determination . . . . .</b>	<b>45</b>
3.1 Determine message store type . . . . .	46
3.2 Where to go from here . . . . .	47
<b>Chapter 4. Message store with data store persistence . . . . .</b>	<b>49</b>
4.1 Introduction to data stores. . . . .	50
4.1.1 Symptoms of a data store problem . . . . .	50
4.2 Verify system integrity . . . . .	51
4.2.1 Verify database status. . . . .	51
4.2.2 Verify database configuration . . . . .	51
4.2.3 Verify connectivity to the database . . . . .	52
4.2.4 Verify the messaging engine is started . . . . .	52
4.3 Messaging engine fails to start . . . . .	52
4.3.1 Analyze the SystemOut log for the messaging engine . . . . .	52
4.3.2 Data source not found. . . . .	55
4.3.3 Data source ClassNotFoundException . . . . .	57
4.3.4 Data source connection pool constraint . . . . .	57
4.3.5 Messaging engine fails to obtain a lock . . . . .	58
4.3.6 Messaging engine fails to obtain lock on failover . . . . .	58
4.3.7 Messaging engine unique ID does not match data store . . . . .	59
4.3.8 Data source exception: SQLCODE -471 . . . . .	61
4.3.9 Data source exception: unavailable resource . . . . .	62
4.3.10 Data source exception: failure to load native library . . . . .	63
4.3.11 Data source exception: storage allocation error . . . . .	64
4.4 Messaging engine not accepting work . . . . .	65
4.4.1 Analyze SystemOut. . . . .	65
4.4.2 Active messaging engine loses the lock on the data store . . . . .	66
4.5 Validate solution . . . . .	67
<b>Chapter 5. File store problem determination . . . . .</b>	<b>69</b>
5.1 Introduction to using file stores . . . . .	70
5.1.1 Symptoms of a file store problem . . . . .	70
5.1.2 Sizing your file store files . . . . .	70
5.1.3 File system requirements for file store . . . . .	72
5.2 Collect diagnostics . . . . .	74
5.3 Analyze diagnostics. . . . .	74
5.4 Causes and solutions . . . . .	76
5.4.1 Incorrect path . . . . .	76
5.4.2 User account not authorized . . . . .	77
5.4.3 File store has a space constraint during startup . . . . .	78
5.4.4 File store has a space constraint during runtime . . . . .	78
5.4.5 Validate solution . . . . .	80

<b>Chapter 6. JMS application problem determination</b> . . . . .	81
6.1 Symptoms of a JMS application problem . . . . .	82
6.2 Analyze diagnostics . . . . .	82
6.2.1 Verify the messaging engine is started . . . . .	82
6.2.2 Analyze application exception messages . . . . .	83
6.3 JMS application outside the server container cannot connect to bus . . . . .	84
6.3.1 Symptoms of a connection problem . . . . .	84
6.3.2 Analyze the exception in the application log . . . . .	85
6.3.3 Unable to contact the bootstrap server . . . . .	88
6.3.4 No messaging engine matches the target properties . . . . .	90
6.3.5 Redirect to messaging engine fails . . . . .	91
6.3.6 Validate the solution . . . . .	92
6.4 JMS application in server container cannot connect to the bus . . . . .	92
6.4.1 Symptoms of a connection problem . . . . .	93
6.4.2 Analyze the exception in the application log . . . . .	93
6.4.3 No messaging engine matches the target properties . . . . .	94
6.4.4 Bus name does not exist . . . . .	95
6.4.5 No messaging engines active . . . . .	95
6.4.6 Matching messaging engine found but not started . . . . .	96
6.4.7 Security authorization failure . . . . .	96
6.4.8 Validate the solution . . . . .	97
6.5 JMS application unable to produce messages . . . . .	97
6.5.1 Symptoms that an application is unable to produce messages . . . . .	97
6.5.2 Analyze diagnostics . . . . .	97
6.5.3 Causes and solutions . . . . .	99
6.5.4 Cannot attach to the requested destination . . . . .	99
6.5.5 Insufficient authorization to create a producer for the queue . . . . .	100
6.5.6 Application not closing producer objects . . . . .	100
6.5.7 Current system state problems . . . . .	101
6.5.8 Validate the solution . . . . .	101
6.6 Message produced but does not arrive at destination . . . . .	102
6.6.1 Verify the transaction was committed . . . . .	102
6.6.2 Verify the message was sent to the correct destination . . . . .	102
6.6.3 Check for the message on the exception destination . . . . .	103
6.6.4 Find the message . . . . .	103
6.7 JMS application unable to consume messages . . . . .	103
6.7.1 Symptoms that an application is unable to consume messages . . . . .	103
6.7.2 Verify integrity . . . . .	104
6.7.3 Analyze diagnostics . . . . .	105
6.7.4 Cannot attach to the destination . . . . .	106
6.7.5 Not authorized for the requested queue . . . . .	107
6.7.6 Application not closing consumer objects . . . . .	107
6.7.7 Validate the solution . . . . .	107

<b>Chapter 7. Messaging in a multiple messaging engine environment . . .</b>	<b>109</b>
7.1 Symptoms of problems in a multiple messaging engine environment . . .	110
7.2 Analyze diagnostics. . . . .	110
7.2.1 Analyze the application log . . . . .	110
7.2.2 Analyze SystemOut for the messaging engines . . . . .	111
7.3 Messages are lost or are slow to arrive. . . . .	111
7.4 Messages are not consumed or consumed slowly . . . . .	113
7.5 SIMPLimitExceededException failure while sending messages to a queue .	114
7.5.1 Examine the queue point message depths . . . . .	115
7.6 SIMPLimitExceededException failure while sending messages to a topic	116
space . . . . .	116
7.6.1 Examine the publication point message depths . . . . .	116
<b>Chapter 8. Clustering problem determination . . . . .</b>	<b>119</b>
8.1 Introduction to clustering for messaging . . . . .	120
8.1.1 Symptoms of problems specific to clustering . . . . .	120
8.2 Analyze diagnostics. . . . .	121
8.2.1 Determine if the messaging engine started normally . . . . .	121
8.3 Clustering configuration problems. . . . .	125
8.3.1 JDBC provider or data source configuration error. . . . .	125
8.3.2 Environment variables in the classpath are scoped incorrectly . . .	130
8.3.3 The file store directories were created incorrectly. . . . .	132
8.3.4 Table schema names are not unique . . . . .	133
8.4 Cluster runtime problems . . . . .	134
8.4.1 Core group policy is incorrect or no preferred server defined . . . .	134
8.4.2 A permanent reply queue is in use . . . . .	138
8.4.3 Data store is locked. . . . .	138
8.4.4 Orphaned messages after failover . . . . .	139
8.4.5 Core group policy is not correct or no preferred server defined . . .	140
8.4.6 Orphan messages on a partitioned destination. . . . .	142
8.4.7 Cluster weights incorrectly specified. . . . .	143
<b>Chapter 9. Message-driven beans problem determination . . . . .</b>	<b>145</b>
9.1 Introduction to message-driven beans . . . . .	146
9.1.1 Symptoms of a messaging-driven bean problem . . . . .	147
9.2 Verify integrity . . . . .	147
9.2.1 Verify the MDB application is started . . . . .	147
9.2.2 Verify that messages can be sent to the target destination . . . . .	148
9.2.3 Verify the MDB is consuming messages. . . . .	148
9.3 Analyze the SystemOut log for the MDB server . . . . .	149
9.4 Causes and solutions . . . . .	149
9.4.1 Activation specification destination name incorrect. . . . .	150



9.4.2	Unable to locate the activation specification . . . . .	151
9.4.3	Invalid activation specification . . . . .	151
9.4.4	Listener port used for default messaging resources . . . . .	152
9.4.5	MDB unable to connect to a messaging engine . . . . .	153
9.4.6	MDB does not handle exception appropriately . . . . .	153
9.4.7	MDB started but inactive: No errors in the logs . . . . .	154
9.5	Validate the solution . . . . .	155
<b>Chapter 10.</b>	<b>WebSphere MQ and MDBs . . . . .</b>	<b>157</b>
10.1	Symptoms of problems with MDBs and listener ports . . . . .	158
10.2	Verify system integrity . . . . .	158
10.3	Analyze the application server log . . . . .	158
10.4	Validate the solution . . . . .	159
<b>Chapter 11.</b>	<b>WebSphere MQ server problem determination . . . . .</b>	<b>161</b>
11.1	Introduction to WebSphere MQ server configurations . . . . .	162
11.2	Verify system integrity . . . . .	162
11.3	Symptoms of problems . . . . .	162
11.4	WebSphere MQ server not enabled . . . . .	164
11.4.1	Analyze SystemOut for the messaging engine . . . . .	164
11.4.2	MQ JMS client not detected . . . . .	165
11.4.3	WebSphere MQ client version not supported . . . . .	165
11.4.4	Previous version registered . . . . .	166
11.4.5	Validate the solution . . . . .	166
11.5	Unable to produce messages to a WebSphere MQ server-hosted destination . . . . .	166
11.5.1	Collect diagnostics . . . . .	167
11.5.2	Analyze the application log . . . . .	167
11.6	Unable to consume messages from a WebSphere MQ destination . . . . .	168
11.6.1	Collect diagnostics . . . . .	168
11.6.2	Analyze diagnostics . . . . .	168
11.7	JMS messages arrive but appear corrupt . . . . .	170
11.7.1	Collect diagnostics . . . . .	170
11.7.2	Analyze diagnostics . . . . .	170
11.7.3	WebSphere MQ RFH2 headers not enabled . . . . .	170
11.8	JMSXUserID field padded with spaces or truncated . . . . .	171
11.8.1	JMSCorrelationID field padded with spaces or truncated . . . . .	172
11.8.2	Validate the solution . . . . .	172
<b>Chapter 12.</b>	<b>WebSphere MQ configuration . . . . .</b>	<b>173</b>
12.1	Identify symptoms . . . . .	174
12.2	WebSphere MQ messaging provider not enabled . . . . .	174
12.2.1	Analyze SystemOut . . . . .	175
12.2.2	MQ JMS client not detected . . . . .	175

12.2.3	WebSphere MQ Client version not supported. . . . .	176
12.2.4	Previous version registered. . . . .	176
12.2.5	Validate the solution . . . . .	177
12.3	Messaging client fails to resolve the WebSphere MQ libraries . . . . .	177
12.3.1	Examine the application messages. . . . .	177
<b>Chapter 13.</b>	<b>WebSphere MQ link problem determination . . . . .</b>	<b>179</b>
13.1	Introduction to WebSphere MQ link . . . . .	180
13.2	Verify integrity . . . . .	180
13.2.1	Verify the WebSphere MQ link is running . . . . .	180
13.2.2	Verify the WebSphere MQ link sender channel is running . . . . .	181
13.2.3	Verify the WebSphere MQ link receiver channel is running . . . . .	181
13.3	Identify symptoms . . . . .	182
13.4	Sender channel fails to start . . . . .	183
13.4.1	Collect diagnostics . . . . .	183
13.4.2	Analyze SystemOut. . . . .	184
13.4.3	Cannot contact listener . . . . .	184
13.4.4	Channel not available . . . . .	185
13.4.5	Attempt to send message failed . . . . .	185
13.4.6	Message sequence number error. . . . .	186
13.4.7	Validate the solution . . . . .	187
13.5	Receiver channel fails to start . . . . .	187
13.5.1	Collect diagnostics . . . . .	187
13.5.2	Analyze the WebSphere MQ queue manager logs. . . . .	187
13.5.3	Channel not defined remotely . . . . .	187
13.5.4	Remote channel not available. . . . .	188
13.5.5	Remote host not available. . . . .	188
13.5.6	Validate the solution . . . . .	189
13.6	Encountering a message flow problem from the service integration bus to WebSphere MQ . . . . .	189
13.6.1	Determine if the message has reached the WebSphere MQ link . . . . .	189
13.6.2	Collect diagnostics . . . . .	190
13.6.3	Analyze diagnostics . . . . .	191
13.6.4	Message is too large. . . . .	192
13.6.5	WebSphere MQ errors . . . . .	193
13.6.6	Validate the solution . . . . .	195
13.7	Encountering a message flow problem from WebSphere MQ to the service integration bus . . . . .	195
13.7.1	Verify integrity . . . . .	195
13.7.2	Collect diagnostics . . . . .	195
13.7.3	Analyze diagnostics . . . . .	196
13.7.4	Target queue manager is not known. . . . .	197
13.7.5	Validate the solution . . . . .	198

13.8	Messages flow across the WebSphere link but appear corrupt . . . . .	198
13.8.1	Collect diagnostics . . . . .	199
13.8.2	Analyze diagnostics . . . . .	199
13.8.3	MQRFH not included. . . . .	199
13.9	Validate the solution . . . . .	201
<b>Chapter 14. JMS application with WebSphere MQ problem determination.</b>		
203		
14.1	Identify symptoms . . . . .	204
14.2	Analyze the application log . . . . .	204
14.3	JMS application is unable to create a connection . . . . .	205
14.4	JMS application is unable to produce messages . . . . .	206
14.5	JMS application is unable to consume messages. . . . .	207
14.6	Validate the solution . . . . .	208
<b>Chapter 15. Foreign bus problem determination . . . . .</b>		
209		
15.1	Introduction to foreign buses . . . . .	210
15.1.1	Identify symptoms . . . . .	210
15.2	Verify system integrity . . . . .	210
15.2.1	Verify the status of the foreign bus link . . . . .	210
15.2.2	Verify the messaging engine has started . . . . .	211
15.3	Analyze diagnostics. . . . .	211
15.3.1	Analyze SystemOut for the messaging engines . . . . .	211
15.3.2	Analyze the application log . . . . .	212
15.4	Validate the solution . . . . .	213
15.5	Foreign bus link usage examples . . . . .	213
15.5.1	Establishing a foreign bus . . . . .	213
15.5.2	Configuring the JMS queue to point to the correct bus. . . . .	216
15.5.3	Configuring topic space mappings . . . . .	219
15.5.4	Securing foreign bus environments. . . . .	222
<b>Chapter 16. Mediation problem determination . . . . .</b>		
229		
16.1	Introduction to mediation . . . . .	230
16.2	Identify symptoms . . . . .	230
16.3	Messages are not being consumed by the application . . . . .	231
16.3.1	Determine if messages are queued on the mediation point . . . . .	231
16.3.2	Determine if messages are queued at the wrong destination . . . . .	231
16.4	Messages are being consumed, but are unmediated . . . . .	232
16.5	Mediation is not performed . . . . .	232
16.5.1	Verify the destination is mediated . . . . .	232
16.5.2	Verify the message is being routed to the mediated destination. . . . .	233
16.6	Messages are waiting to be mediated. . . . .	233
16.6.1	Verify the mediation point is operational . . . . .	234
16.6.2	Verify the mediation application is started. . . . .	234

16.6.3 Analyze SystemOut. . . . .	235
16.6.4 Mediation is not configured correctly. . . . .	237
16.6.5 Invalid mediation authentication alias . . . . .	237
16.6.6 Validate the solution . . . . .	238
16.7 Messages are mediated incorrectly. . . . .	238
16.7.1 Collect diagnostics . . . . .	238
16.7.2 Analyze the trace . . . . .	239
16.7.3 Validate the solution . . . . .	241
16.8 Messages are mediated, but slowly . . . . .	241
16.8.1 Configure for performance . . . . .	241
16.8.2 Validate the solution . . . . .	242
16.9 Messages are arriving at the wrong destination . . . . .	243
16.9.1 Examine the messages. . . . .	243
16.10 Search online support . . . . .	244
<b>Chapter 17. Default messaging provider security . . . . .</b>	<b>245</b>
17.1 Introduction . . . . .	246
17.2 Identify symptoms . . . . .	246
17.3 Analyze diagnostics. . . . .	247
17.4 Authentication problems connecting to the bus. . . . .	250
17.4.1 Solution for an application in an application server. . . . .	250
17.4.2 Resolving the problem in a client application . . . . .	255
17.4.3 Validate solution . . . . .	255
17.5 Authorization problems connecting to the bus . . . . .	255
17.6 Authorization problems accessing a destination . . . . .	258
17.6.1 Validate the solution . . . . .	260
<b>Chapter 18. The next step. . . . .</b>	<b>261</b>
18.1 Online resources . . . . .	262
18.1.1 WebSphere MQ resources . . . . .	262
18.2 Contact IBM. . . . .	262
<b>Related publications . . . . .</b>	<b>265</b>
IBM Redbooks publications . . . . .	265
Online resources . . . . .	265
How to get IBM Redbooks publications . . . . .	266
Help from IBM . . . . .	266
<b>Index . . . . .</b>	<b>269</b>

# 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:

Redbooks (logo) ®  
i5/OS®  
z/OS®  
DB2®

GDDM®  
IBM®  
LANDP®  
MQSeries®

Redbooks®  
RACF®  
Visualization Data Explorer™  
WebSphere®

The following terms are trademarks of other companies:

EJB, Java, JDBC, JVM, J2EE, J2SE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. 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 paper provides problem determination information for IBM® WebSphere® Application Server Version 6.1 administrators and Java™ Message Service (JMS) application developers. It helps you diagnose problems in JMS applications that use either the WebSphere Application Server default messaging provider, the WebSphere MQ provider, or both providers working together.

This information applies to WebSphere Application Server V6.1 on distributed platforms and on z/OS®.

## The team that wrote this paper

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

**Richard Coppen** is a Test Architect at the IBM Hursley laboratory, United Kingdom. Since joining IBM in 1997, Richard has been involved with the development of several IBM software products, specializing in the WebSphere product family's Messaging and Web services technologies. Richard received his Masters degree in Software Engineering from the University of Oxford in 2007.

**Gareth Bottomley** is a Software Engineer in the United Kingdom. He has worked on the WebSphere Application Server family of products for the last six years. He has worked at IBM for nine years. His areas of expertise include J2EE™, JMS data persistence, and transaction processing.

**Brian De Pradine** is a Developer working for IBM Software Group in Hursley, England. He has worked for IBM for nine years. For the past five years, he has worked in WebSphere Application Server development, specifically in the areas of messaging and Web services support. He has a Bachelor of Science degree in Applied Physics from Columbia University, and is currently working towards a Master of Science degree in Software Engineering from Oxford University.

**Sarah Drewery** is a Software Engineer with WebSphere Enterprise Service Bus (ESB) Foundation Technologies in IBM Hursley, UK. She has eight years experience in testing technologies related to messaging. Her areas of expertise include the WebSphere platform and WebSphere messaging, specifically in Network Deployment and clustering. She has a Bachelor of Engineering degree

in Integrated Systems Engineering from Manchester Metropolitan University, and a Master of Science in Bioengineering from the University of Strathclyde.

**Graham Hopkins** is a Software Engineer in the United Kingdom. He has seven years experience in the messaging field. He has worked for IBM for seven years. His areas of expertise include application servers and messaging.

**Philip Nickoll** is a Software Engineer in the United Kingdom. He has three years experience in WebSphere and has worked for IBM for ten years. His area of expertise is the JMS resource adapter in WebSphere.

**Alasdair Nottingham** is a Software Engineer with the IBM WebSphere ESB Foundation Technologies group based in the United Kingdom. He has six years experience in the IT industry working in many fields including J2EE, messaging, security, and SOA, specifically with WebSphere Application Server and WebSphere MQ. He has a Bachelor of Science degree in Computer Science from the University of Southampton (UK).

**Matthew Roberts** is a Senior Developer in the Service Integration Bus development team, working at the IBM Hursley Lab in the United Kingdom. He has six years experience working for IBM in the enterprise messaging field, initially on the Java Message Service (JMS) component of WebSphere MQ and WebSphere Application Server V5.0, and subsequently on the Service Integration Bus component of WebSphere Application Server V6.0 and above. Matt has technical and team leadership responsibility for a range of Service Integration Bus components including JMS, topology, and routing management, as well as publish/subscribe interoperation with WebSphere Message Broker over the MQLink. Most recently, Matt has been an active member of the OASIS technical committee for WS-Notification, a Web service open standard for publish/subscribe messaging, leading to architectural and development responsibility for implementation of WS-Notification in WebSphere Application Server V6.1.

**Dave Vines** is an IBM Master Inventor and Software Engineer at the IBM Hursley Laboratory in the United Kingdom working in WebSphere MQ and ESB Development. He joined the laboratory in 1984 and has worked on a wide variety of IBM (we never put an apostrophe on IBM per branding) program products including GDDM®, MQSeries®, LANDP®, and, for the past decade, Component Broker and WebSphere Application Server. He received a Bachelor of Science degree from the University of Exeter, England, in 1984.

**David Ware** is a Senior Software Engineer in the United Kingdom. He has worked for IBM for twelve years. Most of those years, he worked in the field of asynchronous messaging, starting with WebSphere MQ, and, for the last five years, on the default messaging provider in WebSphere Application Server. He is



an expert in many aspects of messaging and is responsible for the design of the core technologies of the service integration bus messaging engine.

**Bryan Williams** is a Software Engineer at the IBM Hursley Laboratory in the United Kingdom; he works in WebSphere MQ and ESB Test. He joined the laboratory in 1985 and has worked on a wide variety of IBM program products including SDF, GDDM, MQSeries, IBM Visualization Data Explorer™, WebSphere System Business Integration, and, for the past few years, WebSphere Application Server. He has a Bachelor of Science degree in Applied Physics from Coventry University, which he received in 1977. He has worked in various areas of IT for 30 years.

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

Carla Sadtler  
International Technical Support Organization, Raleigh Center

William A Reichert III  
WebSphere Application Server SWAT team, IBM US

## Become a published author

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

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will 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](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review IBM Redbooks publications form found at:  
[ibm.com/redbooks](http://ibm.com/redbooks)
- ▶ Send your comments in an e-mail to:  
[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)
- ▶ Mail your comments to:  
IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400



# Introduction to JMS application problem determination

This chapter describes the first steps you take to diagnose a JMS related problem in a WebSphere Application Server V6.1 environment. It helps you determine which message provider is in use, how to find and collect relevant information, and how to find problem determination information in the rest of this IBM Redpaper that is applicable to your particular situation.

## 1.1 Identify your messaging provider type

This publication addresses problems that can occur when using two of the messaging providers supported by WebSphere Application Server. To diagnose a problem, you must first determine which provider you are using.

### 1.1.1 Default messaging provider

The default messaging provider is a component of the WebSphere Application Server Version 6.1 that is installed by default. You can configure it to support JMS messaging as part of a service integration bus.

For a detailed description of WebSphere default messaging concepts and architecture, refer to Part 2 of *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

### 1.1.2 WebSphere MQ as the JMS provider

Alternatively, you can configure WebSphere Application Server Version 6.1 to use WebSphere MQ as a messaging (JMS) provider.

For more information about WebSphere MQ as a JMS provider, see *Configuring JMS resources for the WebSphere MQ messaging provider* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tmj\\_admrm.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tmj_admrm.html)

### 1.1.3 Default messaging provider interoperoperation with WebSphere MQ

You can also configure WebSphere Application Server default messaging to interoperate with your existing WebSphere MQ infrastructure.

There are various forms of interoperation, and understanding the difference is important.

For help with understanding the interoperation choices, see:

- ▶ *Managing WebSphere Application Server Version 5 JMS use of WebSphere Application Server Version 6 messaging resources* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.express.doc/tasks/tjq0000\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.express.doc/tasks/tjq0000_.html)

- ▶ *Interoperating with WebSphere MQ using a WebSphere MQ link at:*  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.zseries.doc/tasks/tjc9999\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.zseries.doc/tasks/tjc9999_.html)
- ▶ *Interoperating with WebSphere MQ on z/OS using WebSphere MQ server at:*  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.zseries.doc/tasks/tjfp0031\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.zseries.doc/tasks/tjfp0031_.html)
- ▶ *Learning about messaging with WebSphere Application Server at:*  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tm\\_learn.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tm_learn.html)

## 1.2 Default messaging provider problem determination

The *service integration bus* is responsible for asynchronously transferring messages from message producing applications to queues and topic spaces. *Message points* within the messaging engines are the physical location of the destination. From the message points, the messages are transferred to the consuming application.

The first step in problem determination is to understand the bus and messaging engine topology that are used in your environment. For example, your topology might be:

- ▶ One service integration bus and one messaging engine.
- ▶ One service integration bus with clustering of messaging engines. Determine if the cluster is configured for high availability or workload management.
- ▶ One service integration bus with multiple messaging engines.
- ▶ One service integration bus with a foreign bus.

The type of topology that is used might be a decision point during the problem determination process. Use this information to identify the messaging engines involved in your application flow.

### 1.2.1 Identify your messaging engine topology

Messages are held on message points associated with the messaging engine. A message point can be a:

- ▶ Queue point
- ▶ Publication point
- ▶ Mediation point

When there is only one messaging engine within a service integration bus, both the producer and consumer applications can be connected to that single messaging engine only where the message point of the queue or topic space is located. This simple case is illustrated (for point-to-point messaging) in Figure 1-1.

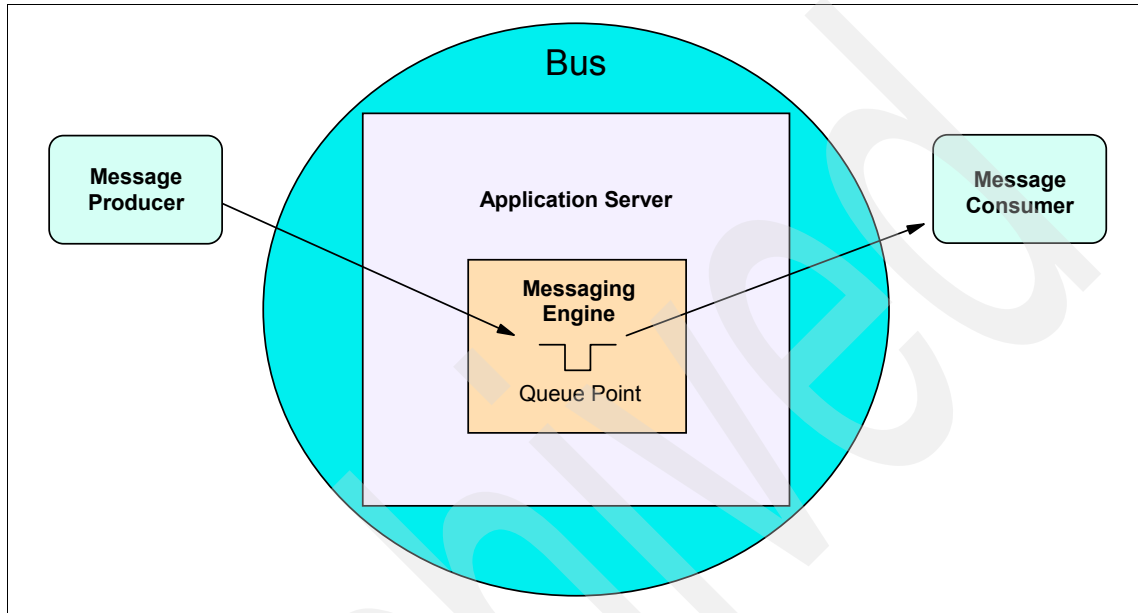


Figure 1-1 Single messaging engine topology

When using the default messaging provider in a distributed environment, you can configure multiple bus members and messaging engines for a single bus.

In addition, you can configure clusters as members of the bus, each with a scope for multiple messaging engines. In a *highly available (HA) configuration*, one messaging engine is active on one of the application servers in the cluster at any given time. If the messaging engine or application server fails, the messaging engine is started on another application server in the cluster. In a *workload management (WLM) configuration*, each application server in the cluster has an active messaging engine.

This more complex case is illustrated in Figure 1-1.

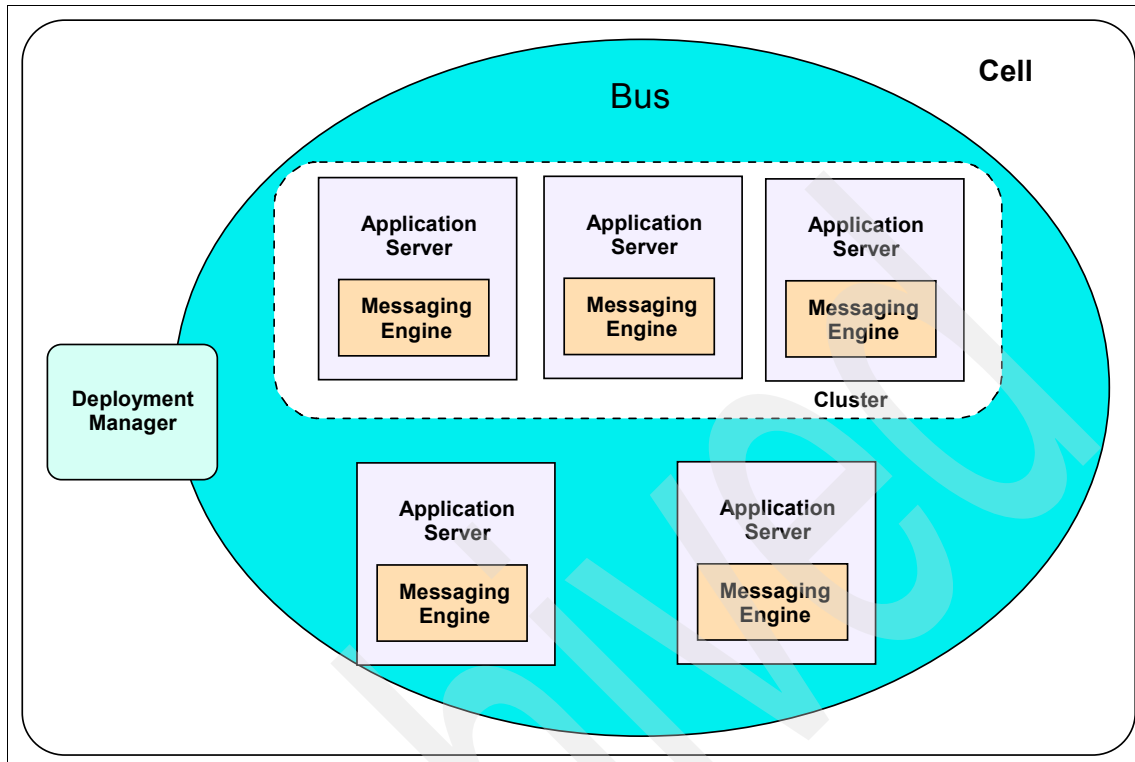


Figure 1-2 Multiple messaging engine topology

If your topology includes multiple messaging engines in the same bus, the producing and consuming applications can be connected to different messaging engines. And, in the case of point-to-point messaging, the producing and consuming applications can be connected to a messaging engine that is remote to the location of the actual message point.

This topology is illustrated in Figure 1-3 on page 6.

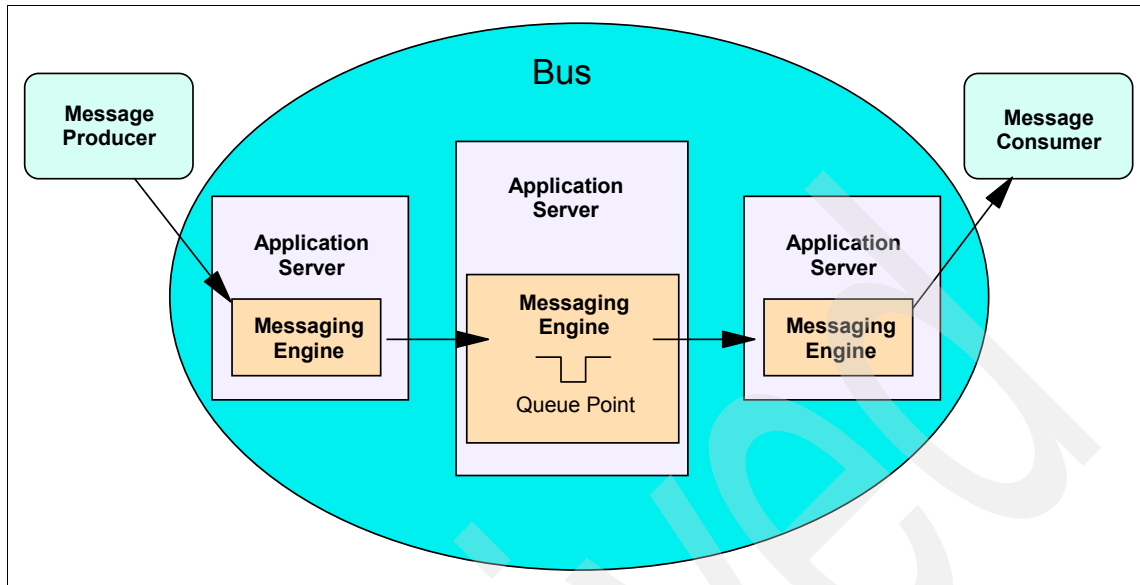


Figure 1-3 Message producer and message consumer connected to different messaging engines

You can configure a bus to connect to and exchange messages with other messaging infrastructures. You define the remote messaging infrastructure to the bus as a *foreign bus*, which can be another service integration bus in the same cell or in a different cell. A foreign bus can also be a WebSphere MQ infrastructure, as shown in Figure 1-4 on page 7.

The foreign bus represents the remote bus type. To identify a specific bus, you need either a WebSphere MQ link or service integration bus link.



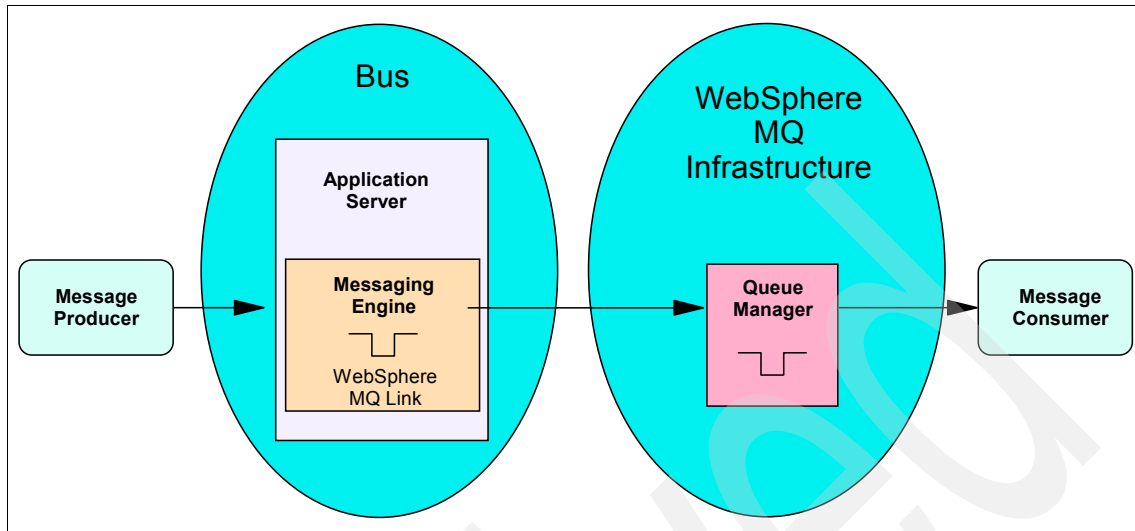


Figure 1-4 WebSphere MQ as a foreign bus

## 1.2.2 Determine the messaging engine connected to an application

If your application uses a JMS JNDI connection factory, inspect its properties to identify where it will connect.

In the WebSphere administrative console:

1. Select **Resources** → **JMS**.
2. Select the type of connection factory you are using.
3. Set the appropriate scope.
4. Click on the connection factory name to open the details page.
5. The connection properties display in the “Connection” area.
6. If a bus member or messaging engine name is defined in the Target field, you can use this to narrow the possible messaging engines to which the application can connect.

Unless the Target significance field is set to *Required*, the messaging engine specified might not necessarily become the configured target. Any messaging engine in the bus can be used.

Alternatively, you can use one of these methods to identify the exact messaging engine to which the application connects at run time:

- ▶ Call the `toString()` method of the connection object. This method includes a reference to the connected messaging engine.

- ▶ Enable the SIBJms\_External trace for a JMS application and inspect the output for a reference to the connected messaging engine.

### 1.2.3 Determine your messaging engine status

If you suspect that you have a default messaging problem, first make sure that the appropriate messaging engines are running.

The simplest method to check the status of the messaging engines is to use the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.

You see a list of messaging engines and their status, as illustrated in Figure 1-5.

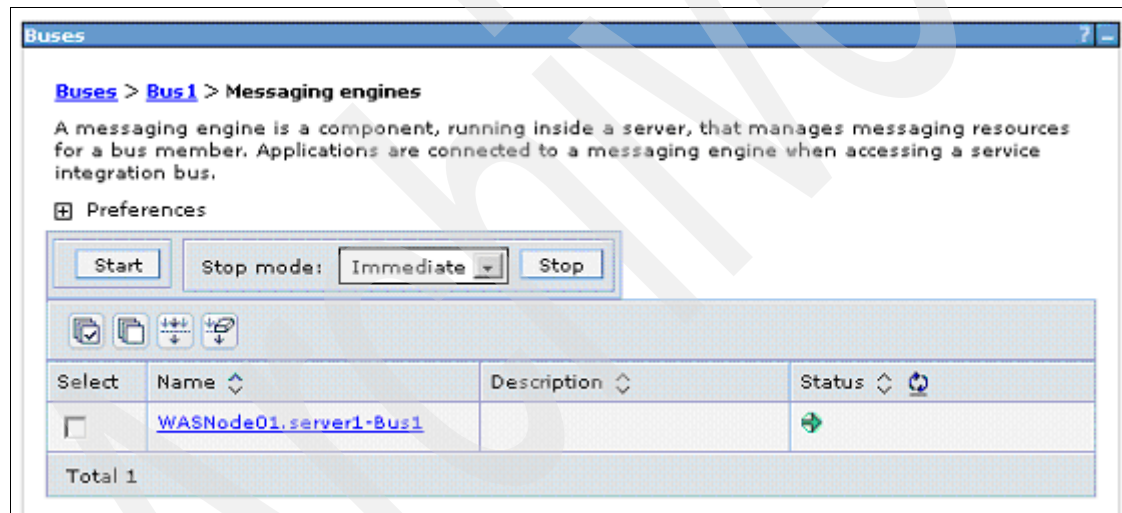


Figure 1-5 Messaging engine status

One of four messaging engine states displays in the Status column of this administrative console's messaging engine panel:

- ▶ A light green arrow indicates that the messaging engine has entered the *Transitive* starting state.
- ▶ A dark green arrow indicates that the messaging engine is in *Started* state.
- ▶ A red cross indicates that the messaging engine is in *Stopped* state.

- ▶ A grey circle with a line through it indicates that the messaging engine is in *Unavailable* state.

You can start or stop the messaging engine by checking the box to the left of the messaging engine and clicking the appropriate button.

If you are stopping the messaging engine, select a stop mode first. An *immediate* stop stops the messaging engine when all messaging operations that are being carried out at the time of the stop request are complete. A *force* stop stops the messaging engine without allowing messaging operations to complete and applications are forcibly disconnected.

## 1.2.4 Finding queued messages

To track messages that might be held in message points, you need to understand your topology. In a simple topology with one messaging engine, looking for messages is simply a matter of viewing the message points on the messaging engine.

If your topology includes multiple messaging engines in the same bus, the producing and consuming applications can be connected to different messaging engines. And, in the case of point-to-point messaging, the producing and consuming applications can be connected to a messaging engine that is remote to the location of the actual message point. Finding messages can involve tracking messages across messaging engines by looking at the remote message points. In Version 6.1, if you have multiple buses in the topology, messages that are queued for a remote bus cannot be viewed on message points.

To find queued messages, you first need to locate all of the message points that might contain messages and then check them for messages.

The following steps show how to check for messages in a point-to-point messaging context. However, you can use similar steps in a publish/subscribe environment; simply substitute references to *queue* and *queue point* with *topic space* and *subscription*, respectively.

To check for messages:

1. If the producing application is a JMS application, use the JMS JNDI destination definition to find the service integration bus destination to which it maps. Use the administrative console to:
  - a. Select **Resources** → **JMS**.
  - b. Select the destination type of **queue**.
  - c. Click on the queue name to open the details page.

- d. The service integration bus destination will be in the **Queue name** field in the Connection area.
2. Determine if the service integration bus destination is an alias:
  - a. Select **Service integration** → **Buses**.
  - b. Click the bus name to open the details page.
  - c. Click **Destinations**.

The type column indicates if the destination is an alias destination.

If this is the case:

    - i. Click on the destination name.
    - ii. The service integration bus destination is referenced under the General properties heading as the **Target identifier**.

Use the target destination, and repeat this step until a queue destination is reached.
3. Check all queue points of the queue destination for messages.

If multiple queue points exist for a given queue, the queue is owned by a clustered messaging engine. In such cases, perform these steps on each of the queue points:

  - a. Check each queue point for messages.
  - b. Where messages are found, click on the message to display its details. If the message is not one you are looking for, continue to the next step.
4. If the queue is mediated, repeat step 3 for the mediation points associated with the queue. If messages are queued for mediation, see Chapter 16, “Mediation problem determination” on page 229.

If you cannot locate your message, check that no applications are consuming messages from the message points and, therefore, removing the messages before they can be viewed in the administrative console.

The following sections explain how messages flow across message points and how to look at the message points involved in the flow.

### **Message points**

When a message is held on a message point, you can view it using the administrative console, as shown in Figure 1-6 on page 11:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.

4. Click the messaging engine name to open the details page.
5. Select the **Runtime** tab.

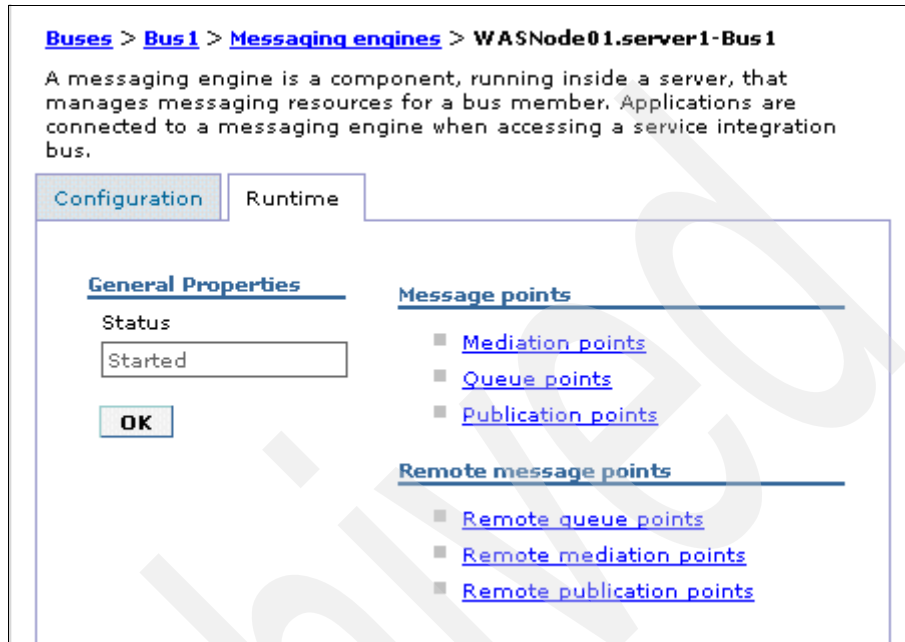


Figure 1-6 Messaging engine runtime information

6. To view messages held on a message point (as illustrated in Figure 1-7), select the message point type, for example, **Queue points**.
7. Click on the message point name.
8. Click **Messages**.

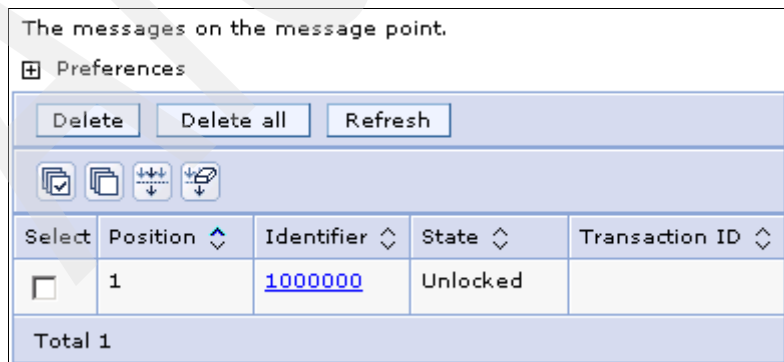


Figure 1-7 Messages held on a message point

The value in the Identifier column does not uniquely identify the message within the service integration bus and has no relation to the JMS message identifier supplied to the application. However, if you click on the identifier, you see the message details, including the API Message properties. These properties contain the identifier that can be obtained by the application from the JMS Message object and the system message ID that uniquely identifies the message within the service integration bus.

### **Remote message points**

The mechanisms that are used by the service integration bus to asynchronously transfer messages between messaging engines in a manner that maintains the high levels of reliability and recoverability (dependent on the message's chosen quality of service) are known as *remote message points*. These are runtime objects, created dynamically by each messaging engine.

There are three types of remote message points, one for each of the corresponding message point types: queue, publication, and mediation (see Figure 1-6 on page 11).

A remote message point on a messaging engine identifies the host messaging engine and its message point. The remote message point manages the state of messages flowing to and from the message point. You can think of it as a proxy to the real message point on a remote messaging engine.

### ***Finding messages queued for transfer***

Each remote message point maintains a queue of messages that are currently waiting to be transferred to the remote messaging engine that hosts the message point. It has information about the current state of those messages being sent or received (to be consumed by local consuming applications).

To view this information, use the administrative console to:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.
4. Click the messaging engine name to open the details page.
5. Select the **Runtime** tab.
6. Click on the type of remote message point.
7. Click on the remote message point name.

Figure 1-8 on page 13 illustrates this for A\_Queue, a queue-type message point.



Figure 1-8 Remote queue point

### Known remote message points

Each active remote message point has a corresponding partner on the remote messaging engine with which it is communicating. These partner objects are referred to as *known remote message points*. You can view them on the messaging engine that owns the message point; on the **Runtime** tab for the message point, look below **Additional Properties**.

Figure 1-9 on page 14 shows known remote queue points for A\_Queue, a queue-type message point.

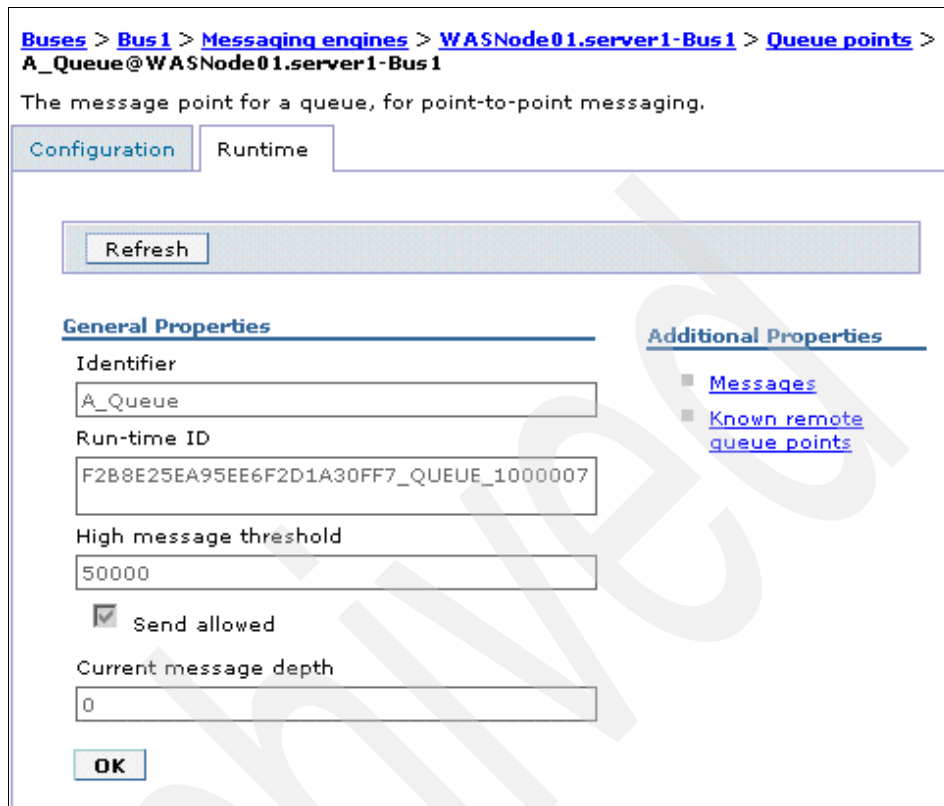


Figure 1-9 Known remote queue points

### Point-to-point messaging queue points

Figure 1-10 on page 15 shows the remote queue points and known remote queue points listed for three messaging engines (ME).



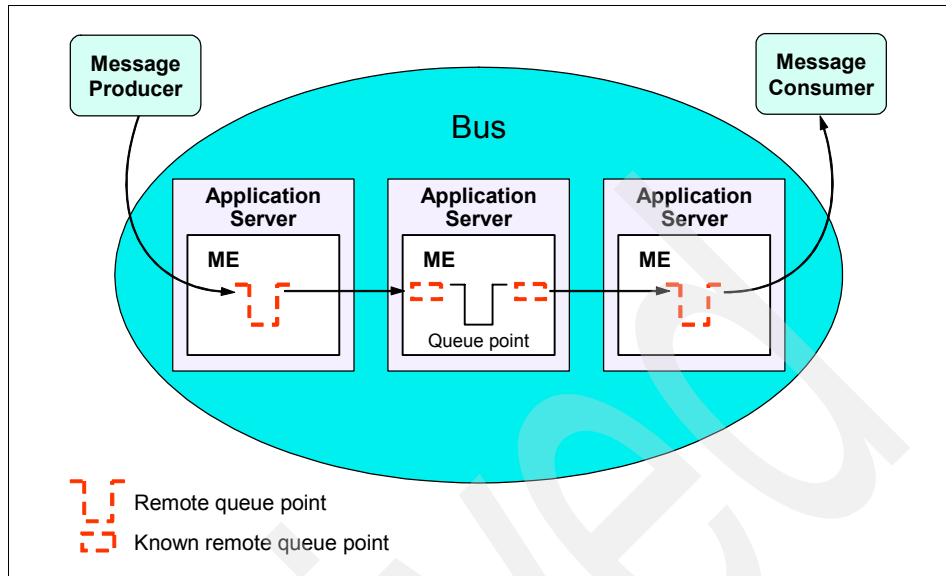


Figure 1-10 Remote queues in point-to-point messaging

The known remote queue points maintain information (Figure 1-11) on which messages have been received from the remote queue point and any current, in-flight messages about to be delivered. They also maintain a history of messages that are currently being consumed on the remote messaging engine.

Buses

[Buses](#) > [Bus1](#) > [Messaging engines](#) > [WASNode01.server1-Bus1](#) > [Queue points](#) > [A.Queue@WASNode01.server1-Bus1](#) > **Known remote queue points**

The remote messaging engines that have remote producers or consumers connected to this queue point.

⊞ Preferences

Select	Messaging engine	Current inbound messages	Inbound messages received	Current messages requests	Completed messages requests
<input type="checkbox"/>	<a href="#">WASNode01.ClusterMember02-Bus1</a>	0	1	0	0
Total		1			

Figure 1-11 Known remote queue point information for example

In a healthy running system, the state of a remote queue point and its partner known queue message point should be in-step and probably empty (or very nearly empty) of queued messages. However, the introduction of multiple places where messages might be queued obviously increases the potential number of places where messages can be held up on the way to ensure the reliable asynchronous delivery of messages. Although these messages are not lost, they can become stuck due to some external influence. Therefore, it is sometimes necessary to inspect this runtime information to either diagnose problems in message delivery or to simply monitor the current state of the system.

### **Publish/subscribe messaging queue points**

Similar to the remote queue points used to transfer messages from one messaging engine to another for point-to-point messages, *remote publication points* are used to transfer messages from one messaging engine to another for publish/subscribe messaging.

Each messaging engine in a bus has a *publication point* that represents a topic space. Any application that produces messages for a topic space produces them to the publication point on the connected messaging engine. This message is then forwarded to each subscription that is interested in this message. If any of those subscriptions are located on another messaging engine in the bus, a remote publication point is used to transfer those messages to the remote messaging engine's publication point and, from there, it is forwarded to the subscription.

If subscriptions exist on more than one remote messaging engine, the message is forwarded to multiple remote publication points, as illustrated in Figure 1-12 on page 17.

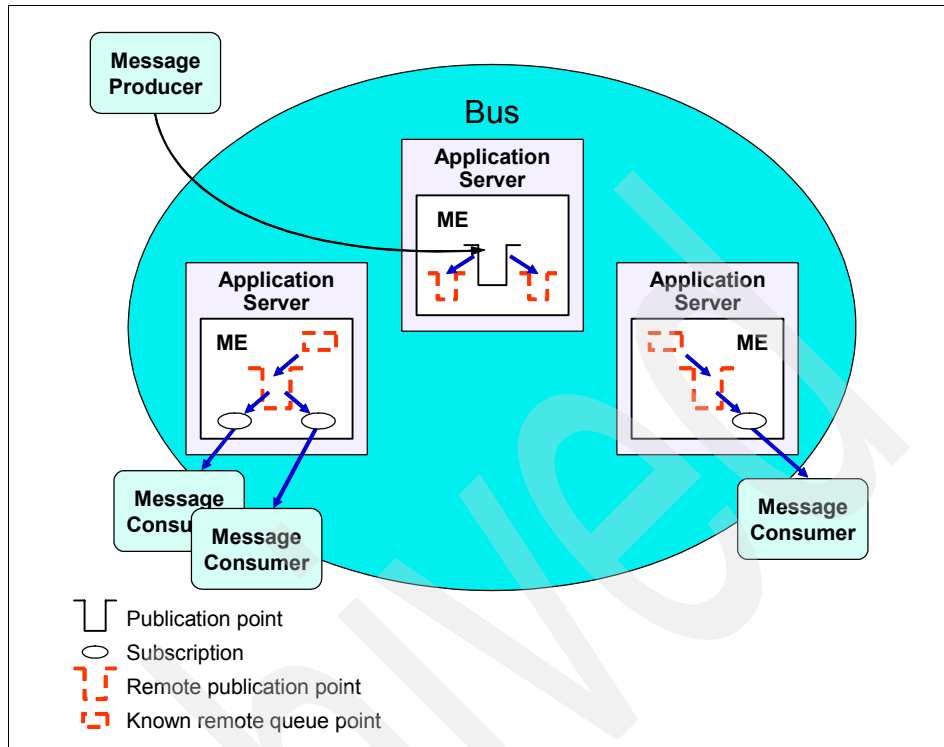


Figure 1-12 Publish/subscribe topology

For further information about messaging between messaging engines in a Network Deployment environment, see *Remote message points* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjo\\_remote\\_msg\\_pts.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjo_remote_msg_pts.html)

## 1.2.5 Finding lost messages

If messages are not being consumed by the application and they are not queued where you expect them, you can do several things to discover the reason.

If you have multiple buses, messages might not be viewable in the administrative console and will, therefore, be very hard to track down. Contact IBM technical support for assistance.

### Prevent applications from consuming messages on a queue

First, identify all possible applications consuming messages from the queue, including aliases to the queue. If you cannot identify all possible applications, you

can disallow all consumers on the queue to prevent applications from consuming messages by setting the **Receive allowed** property of the queue destination on the bus.

To disallow consumers on the queue:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations**.
4. Click on the destination name where you wish to disallow consumers.
5. Clear the **Receive allowed** check box.

You might need to resend messages and repeat the above steps to check for queued messages. If you are still unable to locate your messages, proceed to the next step.

### **Check the message reliability**

If messages have a reliability of *best effort nonpersistent*, the messages might have been discarded due to system resource constraints. This is one drawback of using best effort nonpersistent messages. Consider changing the reliability to *express nonpersistent* and resending the messages.

More information about message reliability can be found in *Message reliability levels* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjj9000\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjj9000_.html)

### **Check for expiration time**

You should check whether the sending application sets an expiration time in the message. If this time has elapsed, then the message will have been removed from the queue.

### **Check the exception destination**

The message might have been moved to an exception destination. Check the configuration of the queue to determine its exception destination policy:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations**.
4. Click on the destination name.

The exception destination policy is specified under the Exception destination heading.

Finally, if a specific destination has been identified, check that destination for the message. Where the system exception destination is being used, check the system exception destination for each messaging engine that hosts a queue point. If the message is found on one of the above destinations, display the message details to find the cause for the exception. Resolve the problem identified.

## 1.3 Collect diagnostics

After you understand your bus and messaging engine topology, determining the specific problem involves inspecting diagnostic information generated by the applications, application servers, and messaging engines in the topology.

Continue your diagnostic approach by collecting these diagnostic data:

- ▶ Application log

When an application receives an exception from a JMS API call, the full stack trace of this exception, including all linked exceptions can be the most useful starting point for problem determination.

When an exception is caught by application code, it is the responsibility of that application to produce the full stack trace of that exception. For example, the application can call the `printStackTrace` method on a caught exception to produce a full stack trace to the standard error location.

The standard error location is as follows:

- If the application is running in the client container, the stack trace is sent to the standard error of the `launchClient` console.
- If the application is running in an application server, the stack trace is sent to the `SystemErr` log of the application server.

An application hosted in an application server (such as a message-driven bean (MDB)) might choose not to catch an exception, or to re-throw it to its container. The container then logs the exception to the JVM™ logs of the application server. It might be necessary to check both the `SystemOut` and `SystemErr` JVM logs in this case.

- ▶ Application server JVM log files for the messaging engines

The messaging engines, and other application server components, within your topology log messages to the JVM logs of the application server within

which they are running. These messages can describe expected events, such as messaging engines starting and stopping, as well as error conditions.

Collect the log for each server where the applications and messaging engines can run. In a high availability configuration, the messaging engine is active on only one cluster member at a time, but it can be started on any member. In a WLM configuration, there might be multiple messaging engines active simultaneously on multiple cluster members.

If you are collecting diagnostic data to send to IBM technical support, see 18.2, “Contact IBM” on page 262.

### 1.3.1 Collecting JVM logs

JVM logs, often referred to as application server or SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager).

To find them, look in the these locations:

- ▶ On WebSphere Application Server V6.x for z/OS:

The JVM logs are located in the address space output. Usually a section labeled SYSOUT contains diagnostic data from the JVM that runs in the servant region.

- ▶ On WebSphere Application Server V6.x (distributed and i5/OS®):

The JVM log files are, by default, named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:

- *profile\_root*/logs/*server\_name*/SystemOut.log
- *profile\_root*/logs/*server\_name*/SystemErr.log

The location of application server logs is configurable:

- Select **Troubleshooting** → **Logs and Trace** in the navigation bar.
- Click on the server name.
- Select **JVM logs**.

This page shows the location of the log file.

If the error occurred some time ago, you might need to check older versions of the log. For example:

SystemOut\_05.06.07\_10.28.48.log

If the error occurs for an application running in a cluster and you do not know the specific server where the error occurred, you might need to collect SystemOut and SystemErr logs from all the active servers in the cluster.

### **JVM log settings**

For information about JVM log settings that affect file rotation, log location, and other SystemOut and SystemErr log behavior, see this Information Center section, *Java virtual machine (JVM) log settings*, at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/utrb\\_jvmlogs.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/utrb_jvmlogs.html)

### **Logs in the z/OS environment**

For the WebSphere on z/OS architecture an application server consists of a control region (CR) address space, and one or more servant region (SR) address spaces. An application server might also include a third type of address space called the Control Region Adjunct (CRA). The CRA is the address space for any messaging engines that might exist for the application server.

Messaging diagnostic information is located in the Servant Region (SR) and Control Region Adjunct (CRA) job logs.

For detailed information about diagnosing problems on the z/OS platform, see *WebSphere for z/OS Problem Determination Means and Tools*, REDP-6880.

## **1.4 Guide to chapters**

After you have gathered the basic information about your messaging configuration, you can approach problem determination several ways using the information in this IBM Redpaper.

To find the correct information:

1. If your messaging engine will not start, start with Chapter 2, “Messaging engine problem determination” on page 35.
2. If you know the error messages being produced, scan the tables in 1.5, “Guide to chapters by message” on page 22 for your message and see the chapter indicated.
3. If your application is receiving exceptions, scan the table in 1.6, “Guide to chapters by exception” on page 28 and see the chapter indicated.

4. If no error messages are being produced, or you are not sure, scan the list of symptoms addressed in each chapter in “Guide to chapters by symptom” on page 30.
5. If you feel sure your symptoms are not addressed, see Chapter 18, “The next step” on page 261 for information about contacting IBM technical support.

## 1.5 Guide to chapters by message

If you have already isolated an error message from an application server, and are unsure if you have a messaging problem, use Table 1-1 through Table 1-21 on page 28 as a reference to find problem determination activities described in this IBM Redpaper.

Table 1-1 CWSIA messages

Specific message	Chapter
CWSIA0004E	▶ Chapter 17, “Default messaging provider security” on page 245
CWSIA0006E	▶ Chapter 17, “Default messaging provider security” on page 245
CWSIA0059E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIA0062E	▶ Chapter 6, “JMS application problem determination” on page 81 ▶ Chapter 15, “Foreign bus problem determination” on page 209
CWSIA0063E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIA0067E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIA0069E	▶ Chapter 6, “JMS application problem determination” on page 81 ▶ Chapter 17, “Default messaging provider security” on page 245
CWSIA0085E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIA0086E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIA0087W	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIA0090E	▶ Chapter 6, “JMS application problem determination” on page 81 ▶ Chapter 17, “Default messaging provider security” on page 245
CWSIA0241E	▶ Chapter 2, “Messaging engine problem determination” on page 35 ▶ Chapter 6, “JMS application problem determination” on page 81



Table 1-2 CWSIC messages

Specific message	Chapter
CWSIC3080E	▶ Chapter 13, "WebSphere MQ link problem determination" on page 179
CWSIC3011E	▶ Chapter 13, "WebSphere MQ link problem determination" on page 179
CWSIC3096I	▶ Chapter 13, "WebSphere MQ link problem determination" on page 179
CWSIC3108E	▶ Chapter 13, "WebSphere MQ link problem determination" on page 179
CWSIC1001E	▶ Chapter 2, "Messaging engine problem determination" on page 35 ▶ Chapter 6, "JMS application problem determination" on page 81
CWSIC8002E	▶ Chapter 11, "WebSphere MQ server problem determination" on page 161

Table 1-3 CWSID messages

Specific message	Chapter
CWSID0027E	▶ Chapter 2, "Messaging engine problem determination" on page 35 ▶ Chapter 8, "Clustering problem determination" on page 119
CWSID0035E	▶ Chapter 8, "Clustering problem determination" on page 119

Table 1-4 CWSII messages

Specific message	Relevant topics
CWSII0205W	▶ Chapter 17, "Default messaging provider security" on page 245
CWSII0219W	▶ Chapter 15, "Foreign bus problem determination" on page 209
CWSII0211 through CWSII0240	▶ Chapter 17, "Default messaging provider security" on page 245
CWSII0242 through CWSII0259	▶ Chapter 17, "Default messaging provider security" on page 245

Table 1-5 CWSIJ messages

Specific message	Chapter
CWSIJ0063E	▶ Chapter 2, "Messaging engine problem determination" on page 35 ▶ Chapter 6, "JMS application problem determination" on page 81

Table 1-6 CWSIK messages

Specific message	Relevant topics
CWSIK0018E	▶ Chapter 17, "Default messaging provider security" on page 245
CWSIK0025E	▶ Chapter 6, "JMS application problem determination" on page 81
CWSIK0033E	▶ Chapter 6, "JMS application problem determination" on page 81
CWSIK0101W	▶ Chapter 16, "Mediation problem determination" on page 229
CWSIK0102E	▶ Chapter 16, "Mediation problem determination" on page 229
CWSIK0103E	▶ Chapter 16, "Mediation problem determination" on page 229
CWSIK0104E	▶ Chapter 13, "WebSphere MQ link problem determination" on page 179

Table 1-7 CWSIP messages

Specific message	Relevant topics
CWSIP0309E	▶ Chapter 17, "Default messaging provider security" on page 245
CWSIP0771I	▶ Chapter 16, "Mediation problem determination" on page 229
CWSIP0773I	▶ Chapter 16, "Mediation problem determination" on page 229
CWSIP0775I	▶ Chapter 16, "Mediation problem determination" on page 229
CWSIP0811W	▶ Chapter 11, "WebSphere MQ server problem determination" on page 161.
CWSIP0815W	▶ Chapter 11, "WebSphere MQ server problem determination" on page 161.

Table 1-8 CWSIS messages

Specific message	Chapter
CWSIS0002E	▶ Chapter 2, "Messaging engine problem determination" on page 35 ▶ Chapter 8, "Clustering problem determination" on page 119
CWSIS0002E with CWSISxxxxE	▶ Chapter 4, "Message store with data store persistence" on page 49
CWSIS0002E with CWSOMxxxxE	▶ Chapter 5, "File store problem determination" on page 69
CWSIS1002E	▶ Chapter 5, "File store problem determination" on page 69
CWSIS1501E	▶ Chapter 4, "Message store with data store persistence" on page 49 ▶ Chapter 8, "Clustering problem determination" on page 119
CWSIS1514E	▶ Chapter 4, "Message store with data store persistence" on page 49

Specific message	Chapter
CWSIS1519E	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> <li>▶ Chapter 8, "Clustering problem determination" on page 119</li> </ul>
CWSIS1522E	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> </ul>
CWSIS1524E	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> <li>▶ Chapter 8, "Clustering problem determination" on page 119</li> </ul>
CWSIS1535E	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> <li>▶ Chapter 8, "Clustering problem determination" on page 119</li> </ul>
CWSIS1538I	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> </ul>
CWSIS1545I	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> </ul>
CWSIS1546I	<ul style="list-style-type: none"> <li>▶ Chapter 4, "Message store with data store persistence" on page 49</li> </ul>
CWSIS1573E through CWSIS1576E	<ul style="list-style-type: none"> <li>▶ Chapter 5, "File store problem determination" on page 69</li> </ul>

Table 1-9 CWSIT messages

Specific message	Chapter
CWSIT0006E	<ul style="list-style-type: none"> <li>▶ Chapter 2, "Messaging engine problem determination" on page 35</li> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0007W	<ul style="list-style-type: none"> <li>▶ Chapter 2, "Messaging engine problem determination" on page 35</li> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0016E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0019E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0022E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0029I	<ul style="list-style-type: none"> <li>▶ Chapter 7, "Messaging in a multiple messaging engine environment" on page 109</li> </ul>
CWSIT0057E	<ul style="list-style-type: none"> <li>▶ Chapter 15, "Foreign bus problem determination" on page 209</li> </ul>
CWSIT0067E	<ul style="list-style-type: none"> <li>▶ Chapter 15, "Foreign bus problem determination" on page 209</li> </ul>
CWSIT0086E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> <li>▶ Chapter 15, "Foreign bus problem determination" on page 209</li> </ul>
CWSIT0088E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0089E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>
CWSIT0090E	<ul style="list-style-type: none"> <li>▶ Chapter 6, "JMS application problem determination" on page 81</li> </ul>

Specific message	Chapter
CWSIT0092E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIT0102E	▶ Chapter 6, “JMS application problem determination” on page 81
CWSIT0104E	▶ Chapter 6, “JMS application problem determination” on page 81

Table 1-10 CWSIV messages

Specific message	Relevant topics
CWSIV0775W	▶ Chapter 9, “Message-driven beans problem determination” on page 145

Table 1-11 CWSIQ messages

Specific message	Relevant topics
CWSIQ0017E	▶ Chapter 13, “WebSphere MQ link problem determination” on page 179

Table 1-12 CWSIZ messages

Specific message	Relevant topics
CWSIZ0002E CWSIZ0011E CWSIZ0020E CWSIZ0021E CWSIZ0045E CWSIZ0056E CWSIZ0057E CWSIZ0058E CWSIZ0059E	▶ Chapter 16, “Mediation problem determination” on page 229

Table 1-13 CWSJ messages

Specific message	Relevant topics
CWSJP9999E CWSJP0023E CWSJP0002E	▶ Chapter 11, “WebSphere MQ server problem determination” on page 161
CWSJR1181E CWSJR1192E	▶ Chapter 9, “Message-driven beans problem determination” on page 1456

Table 1-14 CWSOM messages

Specific message	Relevant topics
CWSOM1017E	▶ Chapter 5, “File store problem determination” on page 69
CWSOM1042E	▶ Chapter 5, “File store problem determination” on page 69

Table 1-15 CNTR messages

Specific message	Relevant topics
CNTR0020E	▶ Chapter 9, “Message-driven beans problem determination” on page 145

Table 1-16 CWPk messages

Specific message	Relevant topics
CWPkI0022E	▶ Chapter 17, “Default messaging provider security” on page 245

Table 1-17 DCSV messages

Message prefix	Reference
DCSVxxxxx	▶ <i>WebSphere Application Server V6.1: Workload Management Problem Determination</i> , REDP-4308

Table 1-18 HMGR messages

Message prefix	Reference
HMGRxxxxx	▶ <i>WebSphere Application Server V6.1: Workload Management Problem Determination</i> , REDP-4308

Table 1-19 J2CA messages

Specific message	Relevant topics
J2CA0164E	▶ Chapter 9, “Message-driven beans problem determination” on page 145
J2CA0052E	▶ Chapter 9, “Message-driven beans problem determination” on page 145
J2CA0137E	▶ Chapter 9, “Message-driven beans problem determination” on page 145

Table 1-20 WMSG messages

Specific message	Relevant topics
WMSG0902E	▶ Chapter 12, “WebSphere MQ configuration” on page 173
WMSG1603E	▶ Chapter 11, “WebSphere MQ server problem determination” on page 161 ▶ Chapter 12, “WebSphere MQ configuration” on page 173
WMSG1604E	▶ Chapter 12, “WebSphere MQ configuration” on page 173.
WMSG1605E	▶ Chapter 11, “WebSphere MQ server problem determination” on page 161 ▶ Chapter 12, “WebSphere MQ configuration” on page 173

Table 1-21 Other product messages

Specific message	Chapter
AMQ9202 AMQ9519 AMQ9520 AMQ9534 AMQ9558	▶ Chapter 13, “WebSphere MQ link problem determination” on page 179
DSRA8000E	▶ Chapter 4, “Message store with data store persistence” on page 49
TCPC0003E	▶ Chapter 2, “Messaging engine problem determination” on page 35

## 1.6 Guide to chapters by exception

Table 1-22 contains key exceptions that are discussed in the following chapters. When a message occurs followed by an exception, the message is considered the key indicator. Search for messages in 1.5, “Guide to chapters by message” on page 22 first. Use Table 1-22 if you do not find your key message.

Table 1-22 Guide to chapters by exception

Exception	Chapter
ClassNotFoundException + database error	▶ Chapter 8, “Clustering problem determination” on page 119
com.ibm.db2.jcc.b.SqlException	▶ Chapter 4, “Message store with data store persistence” on page 49
com.ibm.db2.jcc.t2zos.y+Storage Allocation Error	▶ Chapter 4, “Message store with data store persistence” on page 49

com.ibm.ejs.jms.listener.MDBInvalidConfigException	▶ Chapter 9, “Message-driven beans problem determination” on page 145
com.ibm.mq.MQException	▶ Chapter 11, “WebSphere MQ server problem determination” on page 161
com.ibm.mqservices.MQInternalException	▶ Chapter 10, “WebSphere MQ and MDBs” on page 157 ▶ Chapter 14, “JMS application with WebSphere MQ problem determination” on page 203
com.ibm.ws.sib.msgstore.PersistenceException	▶ Chapter 5, “File store problem determination” on page 69
com.ibm.ws.sib.comms.server.ObjectStoreFullException	▶ Chapter 6, “JMS application problem determination” on page 81
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException	▶ Chapter 8, “Clustering problem determination” on page 119
com.ibm.wsspi.sib.core.exception.SIRollbackException	▶ Chapter 5, “File store problem determination” on page 69
java.io.FileNotFoundException	▶ Chapter 5, “File store problem determination” on page 69
java.lang.ClassNotFoundException + <i>database error</i>	▶ Chapter 4, “Message store with data store persistence” on page 49
java.lang.Exception: De-reference of JMS provider's Reference failed - check provider is on classpath	▶ Chapter 12, “WebSphere MQ configuration” on page 173
java.sql.SQLException	▶ Chapter 8, “Clustering problem determination” on page 119
javax.jms.JMSEException + CWSIA0241E	▶ Chapter 6, “JMS application problem determination” on page 81
javax.jms.InvalidDestinationException	▶ Chapter 15, “Foreign bus problem determination” on page 209
javax.jms.JMSEException + <i>MQ reason code</i>	▶ Chapter 14, “JMS application with WebSphere MQ problem determination” on page 203
javax.jms.JMSEExceptions + WebSphere MQ return code 2046	▶ Chapter 12, “WebSphere MQ configuration” on page 173

java.lang.UnsatisfiedLinkError: mqjbnd05 (Not found in java.library.path)	<ul style="list-style-type: none"> <li>▶ See PK40371: MQ_INSTALL_VERSION Environment Variable needed for support for BINDINGS mode connection to WebSphere MQ 5.3 at:  <a href="http://www-1.ibm.com/support/docview.wss?uid=swg1PK40371">http://www-1.ibm.com/support/docview.wss?uid=swg1PK40371</a></li> </ul>
javax.naming.NameNotFoundException	<ul style="list-style-type: none"> <li>▶ Chapter 9, “Message-driven beans problem determination” on page 145</li> <li>▶ Chapter 10, “WebSphere MQ and MDBs” on page 157</li> </ul>
javax.jms.InvalidDestinationException	<ul style="list-style-type: none"> <li>▶ Chapter 10, “WebSphere MQ and MDBs” on page 157</li> </ul>
com.ibm.ejs.jms.listener.MDBInvalidConfigurationException	<ul style="list-style-type: none"> <li>▶ Chapter 10, “WebSphere MQ and MDBs” on page 157</li> </ul>
JMSSecurityException	<ul style="list-style-type: none"> <li>▶ Chapter 17, “Default messaging provider security” on page 245</li> </ul>
SILimitExceededException	<ul style="list-style-type: none"> <li>▶ Chapter 7, “Messaging in a multiple messaging engine environment” on page 109</li> </ul>
SSLHandshakeException	<ul style="list-style-type: none"> <li>▶ Chapter 17, “Default messaging provider security” on page 245</li> </ul>
Transaction rollback exceptions	<ul style="list-style-type: none"> <li>▶ Chapter 4, “Message store with data store persistence” on page 49</li> <li>▶ Chapter 5, “File store problem determination” on page 69</li> </ul>

## 1.7 Guide to chapters by symptom

Table 1-23 on page 31 contains key symptoms that are discussed in the following chapters. If you have not identified an error message or exception, use this table to identify your symptom.



Table 1-23 Guide to chapters by symptom

Chapter	Symptoms
Chapter 2, “Messaging engine problem determination” on page 35	<ul style="list-style-type: none"> <li>▶ A messaging engine fails to start</li> <li>▶ A messaging engine is active but fails to accept work</li> <li>▶ In z/OS, a Control Region Adjunct (CRA) abends</li> </ul>
Chapter 4, “Message store with data store persistence” on page 49	<ul style="list-style-type: none"> <li>▶ Messaging engine fails to start</li> <li>▶ Messaging engine is active but fails to accept work</li> <li>▶ Error messages with a prefix of CWSIS. The message text indicates a problem with a data store, data source, or database.</li> <li>▶ ClassNotFoundException that includes a database error message</li> <li>▶ Rollback exceptions in the messaging application</li> <li>▶ com.ibm.db2.jcc.b.SqlException</li> </ul>
Chapter 5, “File store problem determination” on page 69	<ul style="list-style-type: none"> <li>▶ Messaging engine fails to start.</li> <li>▶ Applications experience transaction rollbacks</li> <li>▶ java.io.FileNotFoundException exception</li> <li>▶ com.ibm.ws.sib.msgstore.PersistenceException</li> <li>▶ com.ibm.wsspi.sib.core.exception.SIRollbackException</li> </ul>
Chapter 6, “JMS application problem determination” on page 81	<ul style="list-style-type: none"> <li>▶ A JMS application is unable to create a connection to a service integration bus.</li> <li>▶ A JMS application is unable to produce messages to the WebSphere Application Server default messaging provider</li> <li>▶ JMS application unable to consume messages from the WebSphere Application Server default messaging provider.</li> <li>▶ javax.jms.JMSException exceptions</li> <li>▶ com.ibm.ws.sib.comms.server.ObjectStoreFullException exceptions</li> <li>▶ Lost messages</li> </ul>
Chapter 7, “Messaging in a multiple messaging engine environment” on page 109	<ul style="list-style-type: none"> <li>▶ Messages are not found when expected on the queue</li> <li>▶ Messages are not consumed from the queue when expected</li> <li>▶ Application fails with an SILimitExceededException failure while sending messages to a queue.</li> <li>▶ Application fails with an SILimitExceededException failure while sending messages to a topic space.</li> </ul>
Chapter 8, “Clustering problem determination” on page 119	<ul style="list-style-type: none"> <li>▶ One or more messaging engines do not start.</li> <li>▶ Non-delivery or orphaning of messages.</li> <li>▶ Multiple messaging engines are started on one cluster member while no messaging engines are started on others.</li> <li>▶ The messaging engine is not starting on the preferred server.</li> <li>▶ Response messages appear on a different partition than the request message (messages appear in an unexpected place).</li> <li>▶ Orphan messages after a failover.</li> <li>▶ Messages are not consumed from a partitioned destination.</li> <li>▶ Messages do not arrive on a specified partition of a partitioned destination.</li> </ul>

<p>Chapter 9, “Message-driven beans problem determination” on page 145</p>	<ul style="list-style-type: none"> <li>▶ A message-driven bean application fails to start.</li> <li>▶ A message-driven bean fails to connect to a messaging engine.</li> <li>▶ A message-driven bean fails to consume messages.</li> <li>▶ A message-driven bean fails during processing.</li> <li>▶ Lost messages</li> </ul>
<p>Chapter 10, “WebSphere MQ and MDBs” on page 157</p>	<ul style="list-style-type: none"> <li>▶ The MDB application does not start.</li> <li>▶ The MDB fails to connect to the MQ queue.</li> <li>▶ The MDB application starts, but is not driven by messages on the WebSphere MQ destination.</li> <li>▶ <code>com.ibm.mqservices.MQInternalException</code> exceptions</li> <li>▶ <code>javax.naming.NameNotFoundException</code> for the destination</li> <li>▶ <code>javax.jms.InvalidDestinationException</code> for the destination</li> <li>▶ <code>com.ibm.ejs.jms.listener.MDBInvalidConfigException</code> for the destination</li> </ul>
<p>Chapter 11, “WebSphere MQ server problem determination” on page 161</p>	<ul style="list-style-type: none"> <li>▶ Messages are not processed via the WebSphere MQ server. Messages might appear to be lost.</li> <li>▶ Unable to produce messages to a WebSphere MQ server destination.</li> <li>▶ Unable to consume messages from a WebSphere MQ destination</li> <li>▶ JMS messages arrive but appear corrupt</li> <li>▶ The application is unable to connect to the WebSphere MQ server</li> </ul>
<p>Chapter 12, “WebSphere MQ configuration” on page 173</p>	<ul style="list-style-type: none"> <li>▶ JMS objects such as queue connection factories are defined, but do not appear in JNDI.</li> <li>▶ <code>java.lang.Exception: De-reference of JMS provider's Reference failed - check provider is on classpath</code></li> <li>▶ Client application gets <code>JMSExceptions</code> when attempting to run in bindings mode with a stack trace containing the WebSphere MQ return code 2046.</li> </ul>
<p>Chapter 13, “WebSphere MQ link problem determination” on page 179</p>	<ul style="list-style-type: none"> <li>▶ The sender channel from WebSphere MQ Link fails to start.</li> <li>▶ The receiver channel from WebSphere MQ fails to start</li> <li>▶ Message flow problem from the service integration bus to WebSphere MQ.</li> <li>▶ Message flow problem from WebSphere MQ to the service integration bus</li> <li>▶ Messages flow across the WebSphere link but appear corrupt</li> </ul>

<p>Chapter 14, “JMS application with WebSphere MQ problem determination” on page 203</p>	<ul style="list-style-type: none"> <li>▶ A JMS application receives a JMSEException when attempting to connect to WebSphere MQ.</li> <li>▶ A JMS application receives a JMSEException while attempting to create a producer for a given WebSphere MQ destination.</li> <li>▶ A JMS application receives a JMSEException while attempting to send a message.</li> <li>▶ A JMS application appears to have sent the message successfully, but the message never arrives on the expected WebSphere MQ destination.</li> <li>▶ A JMS Application receives a JMSEException when attempting to create a consumer on for given WebSphere MQ destination.</li> </ul>
<p>Chapter 15, “Foreign bus problem determination” on page 209</p>	<ul style="list-style-type: none"> <li>▶ A messaging engine fails to start after configuring a foreign bus link</li> <li>▶ A foreign bus link fails to start.</li> <li>▶ JMS application gets an exception attempting to produce messages on a foreign destination.</li> </ul>
<p>Chapter 16, “Mediation problem determination” on page 229</p>	<ul style="list-style-type: none"> <li>▶ Messages are not being consumed the application.</li> <li>▶ Messages are being consumed, but are unmediated.</li> <li>▶ Messages are being mediated incorrectly.</li> <li>▶ Messages are mediated, but slowly.</li> </ul>
<p>Chapter 17, “Default messaging provider security” on page 245</p>	<ul style="list-style-type: none"> <li>▶ Authentication problems connecting to the bus.</li> <li>▶ Authorization problems connecting to the bus.</li> <li>▶ Authorization problems accessing destinations.</li> <li>▶ SSL problems connecting to the bus.</li> </ul>

Archived



## Messaging engine problem determination

Messaging engine problems usually fall into two categories: problems with starting the messaging engine and problems with connecting to the messaging engine. User applications designed to use the default messaging provider start independently from the messaging engine subsystem. However, they might become inactive or appear hung if the messaging engine is not running.

This chapter helps you distinguish between these two problem types, and it discusses in more detail problems with starting the messaging engine.

For information about problems that occur when the application attempts to connect to an active messaging engine, see Chapter 6, “JMS application problem determination” on page 81

## 2.1 Introduction

A messaging engine is a component of the WebSphere Application Server providing messaging functionality within a Service Integration Bus. For the application server to be messaging capable (using WebSphere default messaging) and to enable it to process messages locally, you must configure a messaging engine for it, and that messaging engine must be in a *Started* state.

For information about messaging engine configuration, see *Configuring Messaging Engines at:*

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjj0050\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjj0050_.html)

### 2.1.1 Symptoms of a messaging engine problem

Generally, messaging engine problem determination can be categorized into the following two areas:

- ▶ Messaging engine start-up problems
- ▶ Application to messaging engine connectivity

Symptoms addressed in this chapter include:

- ▶ A messaging engine fails to start
- ▶ A messaging engine is active but fails to accept work
- ▶ In z/OS, a Control Region Adjunct (CRA) abends

## 2.2 Verify the messaging engine is started

The first step in diagnosing any messaging problem is to determine the state of the messaging engines on the bus. Using 1.2.3, “Determine your messaging engine status” on page 8, determine which of the following states applies to your message engine.

The states and what they mean are:

- ▶ Unavailable

The message engine cannot be started. Messaging engine initialization is dependent on the bus members' application servers running the SIB Service (Service Integration Bus Service). While this service might be enabled at any

time, and it is automatically enabled for newly added bus members, it is initialized during server start-up.

In a newly-configured messaging engine, the new messaging engine might remain in the Unavailable state until you either start or restart its application servers.

- ▶ Stopped

A Stopped state might not be an indication of a problem. For example, an administrator might have stopped the messaging engine to perform an administrative task.

You can restart the messaging engine from the administrative console; or, you can wait for it to automatically start during the next application server restart.

- ▶ Started

In the messaging engine is in Started state, the underpinning messaging configuration is operational. Your next step in diagnosing the problem is to determine whether the application was able to connect to the messaging engine.

## 2.3 Analyze diagnostics

Problems encountered during messaging engine start-up are written to the application server log files. Problems encountered when connecting to a messaging engine are captured and reported by the JMS application. To collect this documentation, see 1.3, “Collect diagnostics” on page 19.

To analyze the problem:

- ▶ If you experienced a Control Region Adjunct (CRA) abend in z/OS, see 2.4.2, “Incorrect Control Region Adjunct (CRA) ownership (z/OS)” on page 42.
- ▶ Analyze the SystemOut log for the application server running the messaging engine to determine if the messaging engine started correctly.
- ▶ If the messaging engine started correctly, examine the logs associated with the application to determine if it was able to connect to the messaging engine.

This chapter addresses some specific error messages and their root causes. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261.

## 2.3.1 Analyze SystemOut log for the messaging engine

In the SystemOut log, search for error messages beginning with CWS, TCPC, DCSV, or HMGR. Start with the most recent log entries first and work, in order, back through older entries. Pay attention to all CWSI messages, which contain information about the messaging engine startup.

Look for any of these messages:

▶ CWSID0016I

Indicates messaging engine state changes, including *Joined*, *Starting*, *Started*, *Stopping*, *Stopped*. An application can connect to a messaging engine only when the engine is in the *Started* state.

▶ CWSIS0002E

Indicates an error occurred while starting the messaging engine. Messages encapsulated within this error and error messages that follow provide more information about the problem.

- A CWSIS0002E message with an embedded CWSIS message indicates a *message store* problem. The embedded message text indicates a problem with a data store, data source, or database.

See Chapter 3, “Message store problem determination” on page 45.

- A CWSIS0002E message with an embedded CWSOM message indicates a *file store configuration* problem. The CWSOM error message indicates a problem with a file store or log file.

See Chapter 5, “File store problem determination” on page 69.

▶ TCPC0003E

The following message during messaging engine startup indicates a *port conflict*. If this occurs, the messaging engine will start but will not accept work. See Section 2.4.1, “Service integration port conflicts” on page 42.

TCPC0003E: TCP Channel SIB\_TCP\_JFAP initialization failed. The socket bind failed for host \* and port 7276. The port may already be in use.

▶ DCSVxxxx and HMGRxxxx messages

Another common cause of a messaging failing to start is problems with the distributed environment, usually reflected in DCS (distribution and consistency services) messages in the SystemOut log. For help with these types of problems see *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308.



## Normal messaging engine start-up

To evaluate a messaging engine start-up problem, you need to be familiar with how a normal start-up appears in the SystemOut log. Example 2-1 is an extract from an application server log file; it shows the key entries associated with normal messaging engine start-up when the messaging engine is configured to use a file store for persistence.

### *Example 2-1 Messaging engine start up with file store*

---

```
WsServerImpl A WSVR0001I: Server server1 open for e-business
:
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Joined.
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Starting.
:
SibMessage I [:] CWSIS1566I: A single previous owner was found in
the messaging engine's file store, ME_UUID=1D842649652FA294,
INC_UUID=0E040E040F8E07CD
SibMessage I [:] CWSIS1563I: The messaging engine,
ME_UUID=1D842649652FA294, INC_UUID=0D800D800FA453BF, has acquired an
exclusive lock on the file store.
:
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Started.
```

---

Example 2-2 is an extract from an application server log file that shows the key entries associated with normal messaging engine start-up where the messaging engine is configured to use a data store for persistence. See Chapter 3, “Message store problem determination” on page 45 for more information about messaging engine persistence options.

### *Example 2-2 Messaging engine data store*

---

```
SibMessage I [:] CWSID0021I: Configuration reload is enabled for
bus Bus1.
SibMessage I [:] CWSIS1568I: Messaging engine
WASNode01.server1-Bus1 is using a data store.
SibMessage A [:] CWSIC2001I: Messaging connections are being
accepted.
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Joined.
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Starting.
```

```
SibMessage I [:] CWSIS1538I: The messaging engine,
ME_UUID=628531B848F5CBFF, INC_UUID=07FA07FAFAFCFDA2, is attempting to
obtain an exclusive lock on the data store.
SibMessage I [:] CWSIS1543I: No previous owner was found in the
messaging engines data store.
SibMessage I [:] CWSIS1537I: The messaging engine,
ME_UUID=628531B848F5CBFF, INC_UUID=07FA07FAFAFCFDA2, has acquired an
exclusive lock on the data store.
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Started.
```

---

### **Problems during messaging engine startup**

When a messaging engine fails to start, the messages shown in Example 2-3 generally occur and are commonly grouped together. The resulting messaging engine state is Unavailable.

#### *Example 2-3 Messaging engine startup problem*

---

```
... CWSIS0002E: The messaging engine encountered an exception while
starting. Exception: ...
:
CWSID0027E: Messaging engine <MessagingEngineName> cannot be restarted
because a serious error has been reported.
CWSID0016I: Messaging engine <MessagingEngineName> is in state Stopped.
CWSID0016I: Messaging engine <MessagingEngineName> is in state Joined.
```

---

The CWSIS0002E message provides the primary source of problem determination information for messaging engine startup problems. This is a generic message; however, the encapsulated message text normally contains information that can help you identify the root cause of the problem.

If the messages indicate the messaging engine started correctly, continue with section 2.3.2, “Analyze the application logs” on page 40 to determine if the application was able to connect to the messaging engines.

## **2.3.2 Analyze the application logs**

JMS applications that use the default messaging provider connect to a messaging engine, whether they run inside the application server or externally in a client container. This process is abstracted with the use of a connection factory, and the initial connection request, or *bootstrap*, might simply serve to redirect the application to another messaging engine on the bus.

Find errors that occur when the application connects to the messaging engine in the application logs:

- ▶ If the messaging engine experiences port conflicts, it starts but does not accept work. You might see these messages:

CWSIT0007W: It is not possible to contact the bootstrap server at localhost:7276:BootstrapBasicMessaging because of exception: com.ibm.websphere.sib.exception.SIResourceException:

CWSIC1001E: A client attempted to connect with a remote messaging engine (localhost:7276 - BootstrapBasicMessaging) but the connection cannot be completed. Ensure the messaging engine is started: exception com.ibm.ws.sib.jfapchannel.JFapConnectFailedException:

CWSIJ0063E: A network connection to host name 127.0.0.1, port 7276 cannot be established... javax.jms.JMSEException:

CWSIA0241E: An exception was received during the call to the method JmsManagedConnectionFactoryImpl.createConnection: com.ibm.websphere.sib.exception.SIResourceException:

CWSIT0006E: It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: [localhost:7276:BootstrapBasicMessaging].

For more on port conflicts see 2.4.1, “Service integration port conflicts” on page 42.

- ▶ When the connection process for a J2EE application fails, a generic JMSEException is generated on a createConnection, createQueueConnection or createTopicConnection:

javax.jms.JMSEException: CWSIA0241E

If you see this type of exception, see Chapter 6, “JMS application problem determination” on page 81.

## 2.4 Causes and solutions

Common conditions that can prevent a messaging engine from starting or that can keep it from accepting work include:

- ▶ Service integration port conflicts
- ▶ Incorrect Control Region Adjunct (CRA) ownership (z/OS)

## 2.4.1 Service integration port conflicts

When a messaging engine is started but does not accept work, look for indications of a port conflict during startup. You might see a message such as the following in the SystemOut log for the messaging engine server:

```
TCPC0003E: TCP Channel SIB_TCP_JFAP initialization failed. The socket bind failed for host * and port 7276. The port may already be in use.
```

A JMS client might experience this error:

*Example 2-4 JMS client symptom of a port conflict*

---

```
[:] CWSIT0007W: It is not possible to contact the bootstrap server at localhost:7276:BootstrapBasicMessaging because of exception: com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E: A client attempted to connect with a remote messaging engine (localhost:7276 - BootstrapBasicMessaging) but the connection cannot be completed. Ensure the messaging engine is started: exception com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A network connection to host name 127.0.0.1, port 7276 cannot be established...  
javax.jms.JMSEException: CWSIA0241E: An exception was received during the call to the method JmsManagedConnectionFactoryImpl.createConnection: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: [localhost:7276:BootstrapBasicMessaging].
```

---

This is a configuration error. If you see this message, review the ports assigned to the application server to ensure that there are no conflicts with the following ports:

- ▶ Service Integration Bus (SIB) port
- ▶ SIB secure port
- ▶ SIB MQ interoperability port
- ▶ SIB MQ interoperability secure port

If you suspect that there might be a port conflict, you can use a command, such as `netstat`, to view the ports that are currently in use.

## 2.4.2 Incorrect Control Region Adjunct (CRA) ownership (z/OS)

An installation configuration error can result in not assigning the Adjunct region its own CRA1 ID for the Started Task. This result is not apparent until the CRA

region is required. The symptom of this problem is a CRA abends due to incorrect ownership.

Figure 2-1 illustrates correct ownership for the task BBOS1A (last line). For an incorrectly created CRA region you would see ownership by the Servant Region (SR); that is, ASSR1.

NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO
	BBOWTR	BBOWTR	BBOWTR			NS	FF	236	0.00	0.00	
	BBOS1	BBOS1	BBOCTL	STC00120	ASCR1	NS	EF	46T	0.00	0.00	
	BBODMNB	BBODMNB	BBODAEMN	STC00122	WSDMNCR1	NS	EF	3636	0.00	0.00	
	BBOS1S	BBOS1S	BBOSR	STC00125	ASSR1	IN	EF	13T	0.00	4478.2	
	BBOS1A	BBOS1A	BBOSR	STC00124	CRA1	IN	EF	13T	0.00	9816.0	

Figure 2-1 BBOS1A ownership

If you determine that the CRA is abending due to incorrect ownership:

1. Enter the ISPF customization dialog in option 6. For example:  
**EX 'HLQ.SBBOCLIB(BBOWSTR)' 'APPL(BBO) LANG(ENUS) ' )**
2. Choose the option for **Create stand-alone Application Server nodes**.
3. Load your application server saved configuration customization variables.
4. Go into **Define Variables** and select the **Server Customization** option.
5. Page through the variables until you find the place where you can select the user ID and procedure for the CRA.
6. Change the user ID to CRA1.
7. Choose the option to generate the customization jobs, and then save the variables.
8. Rerun job BBOCBRAJ, which generates the RACF® commands used to create the user ID and started task definitions for the CRA. These are found in member BBOWBRAK.
9. Extract the commands relevant to the CRA from member BBOWBRAK.
10. If you had not created the ID before, you would run all the commands relevant to CRA, changing the RDEFINE to RALTER on the started task definition. For example, see Example 2-5.

*Example 2-5 Create the user ID*

---

```
ADDUSER CRA1 DFLTGRP(WSSR1) OMVS(UID(2433)
HOME(/var/WebSphere/home/WSSR1) PROGRAM(/bin/sh)) NAME('WAS APPSVR
ADJUNCT') NOPASSWORD NOOIDCARD

ALU CRA1 OMVS(FILEPROC(10000))
```

```
CONNECT CRA1 GROUP(WSCFG1)
PERMIT CB.*.WCLxxx.* CLASS(SERVER) ID(CRA1) ACC(READ)
```

```
PERMIT CB.*.WCLxxxADJUNCT.* CLASS(SERVER) ID(CRA1) ACC(READ)
RALTERSTARTEDprocname.*STDATA(USER(CRA1)GROUP(WSCFG1)TRACE(YES))
SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

---

If you already had defined the ID, you just need to run the command in Example 2-6.

*Example 2-6 Alter the user ID*

---

```
RALTERSTARTEDprocname.*STDATA(USER(CRA1)GROUP(WSCFG1)TRACE(YES))
SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

---

## 2.5 Validate the solution

Restart the messaging engine and bus member (if applicable). Check the messaging engine state to make sure it starts.

## Message store problem determination

The message store holds the state of your WebSphere Application Server default messaging system, both persistently on a storage device and in memory at runtime. During configuration of the WebSphere Application Server, you can choose between these two configuration options for storing persistent message data:

- ▶ **File store.** Introduced in WebSphere V6.1, this mechanism uses flat files on a local or remote file system to store all persistent data.
- ▶ **Data store.** This method lets you use an existing relational database management system (RDBMS), such as IBM DB2®, to store all persistent data.

Problems might occur when a messaging engine attempts to access its persistent message store. The first step in any message store problem investigation is to identify the type of message store persistence in use.

## 3.1 Determine message store type

The simplest method to check the message store type in use is by using the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.
4. Click on the messaging engine name.
5. On the right side of this page, click **Message store** to open the configuration page for the message store and its persistence mechanism.

The type of message store is listed at the top of this page. For example:

Service integration → Buses → *Bus\_name* → Messaging engines → *Messaging\_Engine\_Name* → *Message\_Store\_Type*

Where *Message\_Store\_Type* is listed as either **File store** or **Data store**.

Alternatively, you can determine the message store persistence type by examining the SystemOut log file. For a given messaging engine, one of the following entries is present in the application server log file:

- ▶ SibMessage I [:] CWSIS1569I: Messaging engine *Messaging\_Engine\_Name* is using a file store.
- ▶ SibMessage I [:] CWSIS1568I: Messaging engine *Messaging\_Engine\_Name* is using a data store.

Message store problems are written to the SystemOut log files. If an unexpected exception is caught during the startup of your message store, it will generally be enclosed as part of a CWSIS0002E error message. For example:

SibMessage E [:] CWSIS0002E: The messaging engine encountered an exception while starting. Exception:

The contents of this message can help you (or IBM service personnel) determine the root cause of the problem. Messages and exceptions such as this one are used as reference points throughout this material to enable quicker determination of the problem you are experiencing. Always check your logs for messages and exception information before starting the problem determination exercise so you have the best possible chance of success.



## 3.2 Where to go from here

If your messaging engine has encountered a problem with its message store and it is configured to use *file store persistence*, see Chapter 5, “File store problem determination” on page 69.

If your messaging engine has encountered a problem with its message store and is configured to use *data store persistence*, see Chapter 4, “Message store with data store persistence” on page 49.

Archived

Archived



## Message store with data store persistence

This chapter addresses problems with messaging engines that use a data store for message persistence.

If you are using a file store, see Chapter 5, “File store problem determination” on page 69. If you are not sure which mechanism you are using for persisting messages, see 3.1, “Determine message store type” on page 46.

## 4.1 Introduction to data stores

A data store uses an existing RDBMS to store all persistent data used by the WebSphere Application Server default messaging system. It lets you incorporate your messaging data into an existing data storage infrastructure.

These resources can help you understand data stores:

- ▶ For a description of data stores, see *Data stores* at:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.doc/concepts/cjm0410\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.doc/concepts/cjm0410_.html)
- ▶ For a detailed description of the data store locking, see *Data store exclusive access locking* at:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/concepts/cjm0450\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/concepts/cjm0450_.html)
- ▶ For information about setting up a data store in DB2 for z/OS see Chapter 9 of *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850.  
<http://www.redbooks.ibm.com/abstracts/sg246850.html>
- ▶ For explanation of a specific DB2 z/OS error messages searching for the message in the IBM Information Management Software for z/OS Solutions Information Center at:  
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>

### 4.1.1 Symptoms of a data store problem

Symptoms of a problem with the messaging engine data store include:

- ▶ Messaging engine fails to start
- ▶ Messaging engine is active but fails to accept work
- ▶ Error messages with a prefix of CWSIS. The message text indicates a problem with a data store, data source, or database.
- ▶ `ClassNotFoundException` that includes a database error message.
- ▶ Rollback exceptions in the messaging application
- ▶ `com.ibm.db2.jcc.b.SqlException`

## 4.2 Verify system integrity

If you are using a database as the messaging data store and you are having problems that appear related to the data store, first verify the following information, which are described in more detail below:

- ▶ Database status
- ▶ Database configuration
- ▶ Database connectivity
- ▶ Messaging engine state

### 4.2.1 Verify database status

A messaging engine configured to use a data store relies on connectivity to an underlying RDBMS. Your RDBMS is separate from your WebSphere system (unless you use the default Derby JDBC™ provider in a single server), and it might be in a disjointed state. In any case, if you suspect connectivity to the database could be the problem you should check the RDBMS to make sure that it:

- ▶ It is running satisfactorily
- ▶ It has network connectivity to your WebSphere server machine
- ▶ It is set up to accept connections from your WebSphere server

### 4.2.2 Verify database configuration

Consider these requirements regarding your data stores:

- ▶ The underlying database that is used by a data store must have a page size of at least 4K to hold the tables created by the system.
- ▶ DB2 for z/OS does not support the **create tables automatically** option. On DB2 for z/OS, the DDL for your messaging engine tables must be created using the sibDDLGenerator tool, and then run by your database administrator to manually create the required tables before starting the messaging engine for the first time.

For further information about setting up a data store in DB2 for z/OS, see Chapter 9 of *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850.

## 4.2.3 Verify connectivity to the database

You can test the connection to the database using the administrative console. This test is helpful in discovering network connectivity problems to the database and configuration problems if this is a newly defined database.

For information about testing connections using the administrative console, see *Testing a connection with the administrative console* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tdat\\_testcon.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tdat_testcon.html)

The **Test connection** button in the administrative console is not a 100% reliable method of testing connectivity to your database. The authentication values of the data source might not match those of the messaging engine and environment variables might need adjusting. For more information, see *Test connection service* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cdat\\_testcon.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cdat_testcon.html)

## 4.2.4 Verify the messaging engine is started

Use the process in 1.2.3, “Determine your messaging engine status” on page 8 to determine the state of your messaging engine:

- ▶ If the messaging engine is not in Started state, attempt to start it. If the messaging engine will not start, see 4.3, “Messaging engine fails to start” on page 52
- ▶ If the messaging engine is in Started state, see 4.4, “Messaging engine not accepting work” on page 65

## 4.3 Messaging engine fails to start

A messaging engine that fails to start is often the first visible symptom of a data store configuration problem. Look in the SystemOut log for indications of why the messaging engine fails.

### 4.3.1 Analyze the SystemOut log for the messaging engine

Look for any CWSISxxxx messages. In particular, pay attention to error messages, but information and warning messages are useful as well. Look for

any `ClassNotFoundException` errors that include the data source or JDBC driver for the messaging data store.

The following list contains specific error messages that apply to data store problems. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261.

### Database connectivity errors

The following errors indicate problems with database connectivity:

- ▶ If you see either of these messages, see 4.3.2, “Data source not found” on page 55:
  - `SibMessage I [:] CWSIS1514E`: A data source JNDI name has not been specified.
  - `SibMessage I [:] CWSIS1524E`: Data source `jdbc/com.ibm.ws.sib/testNode01.server1-default` not found.
- ▶ If you see this message, either in the SystemOut log or when you try a Test Connection function, see 4.3.3, “Data source `ClassNotFoundException`” on page 57:
  - **`java.lang.ClassNotFoundException`** for a data source, for example:  
`java.lang.ClassNotFoundException: DSRA8000E: No jar or zip files found in JDBC_DRIVER_PATH`
- ▶ If you see this message, see 4.3.4, “Data source connection pool constraint” on page 57.
  - `SibMessage I [:] CWSIS1522E`: The messaging engine's request for a database connection timed out.

### Database locking errors

To ensure data integrity in a data store the messaging engine takes an exclusive lock on one of its tables at startup. If this lock cannot be obtained the messaging engine will not start.

For more detailed information about data store exclusive access locking please read the *Data store exclusive access locking* in the Information Center at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjm0450\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjm0450_.html)

These error messages indicate problems with database locking:

- ▶ CWSIS1519E indicates a locking problem; however, the root cause might vary depending on whether you see this message after a failover has occurred or at initial startup of the messaging engine:
  - SibMessage I [:] **CWSIS1519E**: Messaging engine testNode01.server1-default cannot obtain the lock on its data store, which ensures it has exclusive access to the data.

If you see this message and *no failover has occurred*, see 4.3.5, “Messaging engine fails to obtain a lock” on page 58.

If you see this message *after a failover situation*, see 4.3.6, “Messaging engine fails to obtain lock on failover” on page 58.
- ▶ If you see CWSIS1535E, see 4.3.7, “Messaging engine unique ID does not match data store” on page 59:
  - SibMessage E [:] **CWSIS1535E**: The messaging engine's unique id does not match that found in the data store.

This message is usually accompanied by these messages:

SibMessage I [:] **CWSIS1538I**: The messaging engine, ME\_UUID=ME2, INC\_UUID=INC1, is attempting to obtain an exclusive lock on the data store.

SibMessage I [:] **CWSIS1545I**: A single previous owner was found in the messaging engine's data store, ME\_UUID=ME1, INC\_UUID=INC1

### Errors specific to DB2 for z/OS

Some common errors are specific to using DB2 for z/OS to host a data store. You might need to work closely with your DBA and SYSPROG when working through these problems.

If you are using DB2 for z/OS to host the data store, look for these messages in the SystemOut log:

#### *Example 4-1 DB2 for z/OS data source exception*

---

**CWSIS0002E**: The messaging engine encountered an exception while starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException:  
**CWSIS1501E**: The data source has produced an unexpected exception:  
***exception\_information***

---



Information specific to the problem is contained in the *exception\_information* of the CWSIS1501E message. The following are possible exceptions:

- ▶ If you see this exception, see 4.3.8, “Data source exception: SQLCODE -471” on page 61:

```
com.ibm.db2.jcc.b.SqlException: DB2 SQL error: SQLCODE: -471,  
SQLSTATE: 55023, SQLERRMC: SYSIBM.SQLTABLES;00E7900C
```

- ▶ If you see this exception, see 4.3.9, “Data source exception: unavailable resource” on page 62:

```
com.ibm.db2.jcc.b.SqlException: UNSUCCESSFUL EXECUTION CAUSED BY AN  
UNAVAILABLE RESOURCE. REASON 00E7009A, TYPE OF RESOURCE 100, AND  
RESOURCE NAME TEMP DATABASE
```

- ▶ If you see this exception, see 4.3.10, “Data source exception: failure to load native library” on page 63:

```
com.ibm.db2.jcc.b.SqlException: Failure in loading T2 native library  
db2jcct2DSRA0010E: SQL State = null, Error Code = -99, 999
```

- ▶ If you see this exception, see 4.3.11, “Data source exception: storage allocation error” on page 64:

```
com.ibm.db2.jcc.t2zos.y:  
[IBM/DB2] [T2zos/2.5.48] T2zosPreparedStatement.readPrepareDescribeOut  
put_.processDescribeOutput:1563:Storage Allocation Error at  
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink.readDataFromPer  
sistence(AbstractItemLink.java:2487) at  
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink._restoreItem(Ab  
stractItemLink.java:639)
```

## 4.3.2 Data source not found

Access to your chosen RDBMS is provided to the message store through a JDBC data source defined in your application server configuration. If this data source cannot be found by the message store at runtime, there are two error messages that might be output to notify you of the problem.

The symptoms of this problem are that the messaging engine will not start, and one of these messages appears in SystemOut:

- ▶ SibMessage I [:] CWSIS1514E: A data source JNDI name has not been specified.
- ▶ SibMessage I [:] CWSIS1524E: Data source *data\_source\_name* not found.

## CWSIS1514E

This message refers to the data source field on the administrative console panel for your message store.

### **Solution**

To check the value in the data source field for the messaging engine to make sure that it is not missing:

1. Select **Service integration** → **Buses** and click the bus name to open the details page.
2. Click **Messaging engines** and click on the name of the messaging engine to open the details page.
3. Click **Message Store** under Additional Properties.
4. Validate that the JDBC name listed under the **Data source JNDI name** heading matches the data source you have configured.

## CWSIS1524E

This message indicates that the message store cannot find the data source name that it has been provided within the JNDI name space.

The most common cause for this problem is that the data source for the messaging data store is not defined at a scope visible to the messaging engine.

For example, if the messaging engine is in `cell3/node2/server1`, the data source might be defined in either `server1` or `node2` or `cell3` to be visible for use by the data store.

Within a highly available environment, there are additional factors to consider when you create data sources, as you might have a messaging engine that can run on more than one server.

For example, if you have a single messaging engine which can fail over between two servers: `cell3/node2/server1` and `cell3/node2/server2`, you need to define the data source at a scope visible to both servers. In this case, a suitable scope could be either `node2` or `cell3`. A common mistake is to create the data source with a scope of `server1`. Initially, the configuration will appear to work, when the messaging engine starts on `server1`. However, when the messaging engine fails-over to `server2`, `server2` will not be able to locate the appropriate data source.

### **Solution**

Ensure that the definition of the data source is in a scope that is visible to the messaging engine (or engines) that are configured to use it.

### 4.3.3 Data source ClassNotFoundException

The symptoms of this problem are that the messaging engine will not start and you receive a data source ClassNotFoundException message. This exception in Figure 4-1 can also appear when you try a Test Connection to the database from the administrative console.

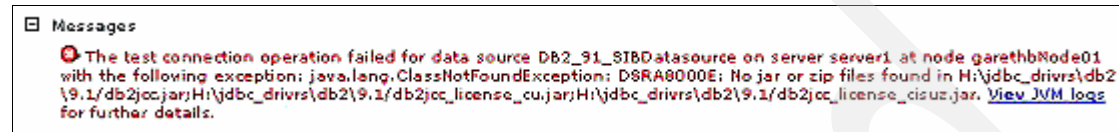


Figure 4-1 ClassNotFoundException on Test Connection

This problem is commonly caused by the content or scoping of your WebSphere Application Server environment variables.

#### Solution

Validate the paths referenced in the exception information to help narrow the problem. For example the following path is part of the default value for a new DB2 JDBC provider:

```
 ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
```

If the value of  `${DB2UNIVERSAL_JDBC_DRIVER_PATH}`  is not set then the application server will not be able to load the classes required to run any data source configured to use this provider. In cases where the value is set correctly, you must check to see that it is defined at a scope visible to the server in which the provider is defined.

The simplest way to check a WebSphere Application Server environment variable is to use the administrative console. Select **Environment** → **WebSphere Variables**.

### 4.3.4 Data source connection pool constraint

The symptoms of this problem are that the messaging engine will not start and the following message appears in SystemOut:

```
SibMessage I [:] CWSIS1522E: The messaging engine's request for a database connection timed out.
```

This message indicates that the messaging engine has encountered a limitation in the current configured size of the connection pool used by the data source.

## Solution

The recommended minimum size of connection pool for your messaging engine's data source is at least 50. This setting allows a large number of concurrent threads to carry out persistent messaging work.

When increasing the size of your connection pool make sure to check your RDBMS documentation to ensure your database supports the required number of concurrent sessions.

For more information, see *Tuning the JDBC data source of a messaging engine* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.iseries.doc/tasks/tjm0230\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.iseries.doc/tasks/tjm0230_.html)

### 4.3.5 Messaging engine fails to obtain a lock

In this situation, the messaging engine fails to start with CWSIS1519E.

In a high availability configuration (the default for clustered messaging engines) where you have multiple instances of a messaging engine configured to fail over, a race could occur between two instances trying to get the lock on their shared data. In this case, the failing instance produces the following message when it fails to acquire its lock:

```
SibMessage I [:] CWSIS1519E: Messaging engine
testNode01.server1-default cannot obtain the lock on its data store,
which ensures it has exclusive access to the data.
```

## Solution

Check that no other instances of the messaging engine are running in the cluster.

### 4.3.6 Messaging engine fails to obtain lock on failover

When a messaging engine fails over, the new messaging engine instance (ME2) attempts to gain its exclusive lock on the database tables as normal. However, the previous instance of the messaging engine (ME1) might not have failed in a controlled manner, which can lead to the lock that it held not being released correctly. Therefore, when ME2 tries to connect to the tables, the pre-existing lock from ME1 might stop it from getting a lock. In this case, you see this message in your logs:

```
SibMessage I [:] CWSIS1519E: Messaging engine
testNode01.server1-default cannot obtain the lock on its data store,
which ensures it has exclusive access to the data.
```

The usual reason for ME1's lock to last longer than the messaging engine instance is the database waiting for the connection between ME1 and the database to timeout.

For example, suppose that the dropped connection timeout is set to five minutes in the database server but it only takes two minutes for ME2 to start after a failover. This situation would leave a time period of three minutes in which ME2 cannot yet gain a lock on the database tables, even though ME1 no longer exists.

This situation is only a minor problem because ME2 continues to try and gain the lock until the database releases the lock that ME1 held, and things can continue as normal.

If, however, the default timeout values in your database system are not small, you can see how easily this outage could become a major one while ME2 is waiting for the lock to be released.

Another version of this problem has been seen when running a database server in a UNIX® environment (including Linux®). In this case, it is not the database timeout which is causing the problem; it is the operating system level TCP\_KEEPAIVE timeout value. The default for this value can be in multiples of hours which would have a major affect on your systems ability to fail over in a timely fashion.

### **Solution**

Check that the connection timeout values in your database server configuration match your expectations regarding failover time. Also check the values for the operating systems TCP timeouts to make sure they allow failover in a timely fashion.

## **4.3.7 Messaging engine unique ID does not match data store**

The symptoms of this problem are that the messaging engine will not start and the following message appears in SystemOut:

```
SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not match that found in the data store.
```

This problem generally occurs when a messaging engine detects that it does not have exclusive ownership of a data store.

There are two potential causes: ME\_UUID mis-match or INC\_UUID mis-match.

## ME\_UUID mis-match

Example 4-2 illustrates the typical error message reported in the SystemOut log files:

### *Example 4-2 ME\_UUID mis-match*

---

```
SibMessage I [:] CWSIS1538I: The messaging engine, ME_UUID=ME2,
INC_UUID=INC1, is attempting to obtain an exclusive lock on the data
store.
SibMessage I [:] CWSIS1545I: A single previous owner was found in the
messaging engine's data store, ME_UUID=ME1, INC_UUID=INC1
SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not
match that found in the data store. ME_UUID=ME2, ME_UUID(DB)=ME1
```

---

Referring to the messages in Example 4-2, you see that the ME\_UUIDs quoted in the CWSIS1535 message do not match. Messaging engine ME2 is trying to take an exclusive lock on tables that are owned by ME1. This will never succeed.

The most likely causes for this problem are failing to clean up the data from a deleted messaging engine or miss-configuration. One of these conditions is likely occurring:

- ▶ A new messaging engine (ME2) is attempting to use the same database and schema information as a previously deleted messaging engine (ME1), without the data for ME1 being deleted.
- ▶ ME2 is attempting to use the same schema name as ME1 in the same database.
- ▶ ME2 is mistakenly using the data source defined for ME1.

## Solution

Make sure that the data source you have configured for your data store references the correct database instance and that the schemas being used by each messaging engine are unique.

## INC\_UUID mis-match

Example 4-3 illustrates the typical error message reported in the application server log files.

### *Example 4-3 INC-UUID mis-match*

---

```
SibMessage I [:] CWSIS1538I: The messaging engine, ME_UUID=ME1,
INC_UUID=INC2, is attempting to obtain an exclusive lock on the data
store.
SibMessage I [:] CWSIS1545I: A single previous owner was found in the
messaging engine's data store, ME_UUID=ME1, INC_UUID=INC1
```

```
SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not
match that found in the data store. ME_UUID=ME1, INC_UUID=INC1,
ME_UUID(DB)=ME1, INC_UUID(DB)=INC2
```

---

Referring to the messages in Example 4-3 on page 60, you see that the INC\_UUIDs quoted in the CWSIS1535E message do not match. An INC\_UUID is used to identify a particular instance of a messaging engine. For example if a messaging engine is configured to fail over between two servers, as in the example given here, there are one ME\_UUID (ME1) and two INC\_UUIDs (INC1 and INC2). To make sure that only one of these messaging engines can access the data store at any one time, the INC\_UUID is checked to determine which instance is currently in control.

The likely cause of this problem is a drop in communication between the database and the messaging engine. This would lead to:

1. ME1,INC1 starts and gets a lock on its data store.
2. ME1,INC1 loses its connection to the database and hence its lock.
3. ME1,INC1 retries to get its lock, but the database is still down.
4. ME1,INC2 is started by the HA manager to replace ME1,INC1.
5. ME1,INC2 find the database is available and acquires the lock.
6. ME1,INC1 also finds the database is now available but notices that another instance has acquired the lock and outputs CWSIS1535.

The root cause of this problem is the loss of connectivity to the database.

### **Solution**

Check your network connectivity and database server logs to determine why the problem occurred.

## **4.3.8 Data source exception: SQLCODE -471**

**This information is for DB2 for z/OS only.**

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 4-4.

*Example 4-4 Data source exception: SQLCODE -471*

---

```
CWSIS0002E: The messaging engine encountered an exception while
starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException:
```

**CWSIS1501E:** The data source has produced an unexpected exception:  
**com.ibm.db2.jcc.b.SqlException: DB2 SQL error: SQLCODE: -471, SQLSTATE: 55023, SQLERRMC: SYSIBM.SQLTABLES;00E7900C**

---

This problem can occur if you have configured your messaging engine to use the DB2 Universal JDBC Driver to connect to DB2 on z/OS. The JDBC driver needs to execute stored procedures that are not available.

### **Solution**

Work with your database administrator to ensure the following requirements have been met:

- ▶ DSNUTILS is configured to use WLM.
- ▶ DSNUTILS is running APF authorized.
- ▶ The WLM environment you are trying to use is defined to WLM and is in an available state.
- ▶ DB2 has RACF authority to use WLM.

Refer to the following resources for further guidance on this problem:

- ▶ *Code 00E7900C*  
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.mc/msgs/code00e7900c.html>
- ▶ *DB2-supplied stored procedures -Invoking utilities as a stored procedure (DSNUTILS)*  
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.ugref/db2stpr.htm>

## **4.3.9 Data source exception: unavailable resource**

### **DB2 for z/OS only**

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 4-5.

*Example 4-5 Data source exception: unavailable resource*

---

**CWSIS0002E:** The messaging engine encountered an exception while starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException:



**CWSIS1501E:** The data source has produced an unexpected exception:  
**com.ibm.db2.jcc.b.SqlException: UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON 00E7009A, TYPE OF RESOURCE 100, AND RESOURCE NAME TEMP DATABASE**

---

This problem indicates that either the TEMP database is not defined or that its page size is not suitable for your use.

### **Solution**

Define a suitable TEMP database for the DB2 Universal JDBC driver so that it will work correctly with DB2 for z/OS.

Work with the database administrator to ensure the following has been done:

- ▶ A TEMP database is created for each member of a data sharing group.
- ▶ The TEMP database has a 32k page size by defining it in a 32K buffer pool.

Refer to the following resources for further guidance on this problem:

- ▶ *Code 00E7009A*

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.mc/msgs/code00e7009a.html>

## **4.3.10 Data source exception: failure to load native library**

### **DB2 for z/OS only**

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 4-6:

*Example 4-6 Data source exception: unavailable resource*

---

**CWSIS0002E:** The messaging engine encountered an exception while starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException:

**CWSIS1501E:** The data source has produced an unexpected exception:  
**com.ibm.db2.jcc.b.SqlException: Failure in loading T2 native library db2jcc2DSRA0010E: SQL State = null, Error Code = -99, 999**

---

The problem is that the DB2 native libraries cannot be found by the DB2 JDBC driver.

## Solution

The DB2 libraries for your WebSphere Application Server installation might have been placed in the linklist or //STEPLIB DD concatenation for the WebSphere Application Server for z/OS address spaces that will use JDBC. In some installations a combination of techniques are used.

If the DB2 libraries have been placed in the linklist or steplib, refer to this resource for further guidance on this problem, *Data access problems - DB2 database*:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rtrb\\_dsaccess3.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rtrb_dsaccess3.html)

If you are still experiencing the problem, update the PROCs for the servant address spaces to include the DB2 libraries.

If your installation does not have SDSNEXIT, SDSNLOAD, and SDSNLOD2 in the linklist, then you must update the //STEPLIB DD concatenation for the servant address space with the missing libraries.

### 4.3.11 Data source exception: storage allocation error

#### DB2 for z/OS only

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 4-7.

*Example 4-7 Data source exception: unavailable resource*

---

**CWSIS1501E:** The data source has produced an unexpected exception:  
**com.ibm.db2.jcc.t2zos.y:**  
**[IBM/DB2] [T2zos/2.5.48] T2zosPreparedStatement.readPrepareDescribeOutput**  
**\_:processDescribeOutput:1563:Storage Allocation Error** at  
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink.readDataFromPersistence(AbstractItemLink.java:2487) at  
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink.\_restoreItem(AbstractItemLink.java:639)

---

When using the DB2 Universal JDBC driver in type 2 mode, the driver can allocate excessive amounts of memory in order to hold result sets, exhausting the available storage space.

## Solution

You can avoid this error by performing one of the following steps:

- ▶ Set the JDBC driver custom property `fullyMaterializeLobData` to `false`. In the administrative console:
  - a. Select **Resources** → **JDBC** → **JDBC Providers**.
  - b. Click the name of the JDBC provider you have configured.
  - c. Select **Data sources**.
  - d. Click the names of the data source you have configured.
  - e. Select **Custom properties**.
  - f. Click **fullyMaterializeLobData** and ensure the value is set to **false**.
- ▶ Or, use the Universal JDBC provider of type 4 driver.

Restart the application server after making your change.

## 4.4 Messaging engine not accepting work

A messaging engine will stop accepting work if the exclusive lock on its data store has been lost. The messaging engine will refuse any further work until it has successfully reacquired the lock.

If your messaging engine stops accepting work during normal run time, you will start to see rollback exceptions in your applications and application server log files.

### 4.4.1 Analyze SystemOut

Analyze SystemOut for the messaging engine. Look for any CWSISxxxx messages. In particular, pay attention to error messages, but information and warning messages are useful as well.

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261. If you see these messages, see 4.4.2, “Active messaging engine loses the lock on the data store” on page 66:

- ▶ `SibMessage I [:] CWSIS1546I: The messaging engine, ME_UUID=82EC72EC0F3555B6, INC_UUID=743A743A0F35629F, has lost an existing lock or failed to gain an initial lock on the data store.`

- ▶ SibMessage I [:] CWSIS1538I: The messaging engine, ME\_UUID=82EC72EC0F3555B6, INC\_UUID=743A743A0F35629F, is attempting to obtain an exclusive lock on the data store.
- ▶ SibMessage I [:] CWSIS1519E: Messaging engine testNode01.server1-default cannot obtain the lock on its data store, which ensures it has exclusive access to the data.

## 4.4.2 Active messaging engine loses the lock on the data store

If the messaging engine stops accepting work during normal run time, you see messages like those in Example 4-8. These messages indicate that a messaging engine has lost the exclusive lock on the data store at run time. The messaging engine stops accepting work and will refuse any further work until it has successfully reacquired the lock. You might see rollback exceptions in the messaging application logs. If the messaging engine continues trying to get the lock, you see many repeating occurrences of these messages.

*Example 4-8 Database lock lost during runtime*

---

SibMessage I [:] **CWSIS1546I: The messaging engine, ME\_UUID=82EC72EC0F3555B6, INC\_UUID=743A743A0F35629F, has lost an existing lock or failed to gain an initial lock on the data store.**

SibMessage I [:] **CWSIS1538I: The messaging engine, ME\_UUID=82EC72EC0F3555B6, INC\_UUID=743A743A0F35629F, is attempting to obtain an exclusive lock on the data store.**

---

In a failover configuration, the problem can be caused by a temporary glitch in the network that caused a failover messaging engine to start and to take a lock on the database. In this case, the original messaging engine should produce the message shown in Example 4-9.

*Example 4-9 CWSIS1519 at failover*

---

SibMessage I [:] **CWSIS1519E: Messaging engine testNode01.server1-default cannot obtain the lock on its data store, which ensures it has exclusive access to the data.**

---

### Solution

This problem is most likely caused by connectivity problems to the database, such as a network or database server outage. Resolve these problems. If network connectivity and the database server are both functional, check the rest of the messaging engines in your failover cluster to see which is currently holding

the lock on the database. The instance that is successfully holding the lock should have started and be accepting work.

## 4.5 Validate solution

Restart the messaging engine or bus member. Check the messaging engine state to ensure it has started and is accepting work.

Archived

Archived

## File store problem determination

WebSphere Application Server Version 6.1 introduced a new persistence option, the *file store*, for the default messaging provider message store. This option uses flat files, on a local or remote file system, to store all persistent data used by the WebSphere Application Server default messaging system. File store is the default persistence mechanism for any new messaging engines you create.

This chapter addresses problems with messaging engines that use a file store for message persistence. If you are using data store, see Chapter 4, “Message store with data store persistence” on page 49. If you are not sure which mechanism you are using for persisting messages, see 3.1, “Determine message store type” on page 46.

## 5.1 Introduction to using file stores

Proper planning and configuration of the file store can prevent problems from occurring. This section discusses issues related to the use of file stores.

For a description of the file store and its configuration, see *File stores* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjm0002\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjm0002_.html)

### 5.1.1 Symptoms of a file store problem

Symptoms of a problem with the messaging engine file store include:

- ▶ Messaging engine fails to start.
- ▶ Applications experience transaction rollbacks.
- ▶ The SystemOut log includes a `java.io.FileNotFoundException` message.
- ▶ The SystemOut log includes a `com.ibm.ws.sib.msgstore.PersistenceException` message.
- ▶ The SystemOut log includes a `com.ibm.wsspi.sib.core.exception.SIRollbackException` message.

### 5.1.2 Sizing your file store files

There are three files which make up a working file store:

- ▶ Log: default size 100MB
- ▶ Permanent Store: default sizes min=200MB, max=500MB
- ▶ Temporary Store: default sizes min=200MB, max=500MB

In most cases running out of space in your file store files can be corrected by tuning the sizes of files in your system to match the requirements of your specific workload. The following sections discuss scenarios to consider when estimating the size of files you will need.

#### **Send and receive large numbers of messages**

If your workload consists of a high throughput of messages then overall storage capacity should not be a priority. Any message in the system is not likely to last long and so it might not even make it to disk before being consumed. In this kind of scenario you need to consider how many messages are likely to be on disk at any one point in time and to ensure that your files are big enough to manage. To determine the correct values for your file sizes in this kind of scenario, you



might get the best results by experimenting with the system under expected workloads.

### **Allow messages to build up for processing later (batch)**

In some systems there might be a requirement to accept messages over a period of time without consuming them to be processed at the end of the day (for example, product stock updates). In these cases, the size of your file store files needs to match or exceed the storage requirements for the amount of message data you expect to build up.

For example, if you expect up to 1000 messages to be received, each with a size below 1 MB, you would need up to  $1000 \times 1 \text{ MB} = 1000 \text{ MB}$  of space in your store files. In practice, the size required is more than 1000 MB because space is reserved in your store files for system purposes. A good estimate for this extra space is the size of your log file. For example, if your log file is 100 MB you would estimate 1100 MB for your store file maximum size.

Factoring in some variance in your expected workload when you calculate your file sizes; overestimating to some extent might be a safer, more reliable approach. Because disk storage is relatively cheap it is better to have a working system with unused file space than to have a system come down due to lack of space. Also, if all of your messages use a persistent reliability, you only need to set your permanent store file size limits to this value. If you do not make use of the non-persistent reliabilities (or vice versa), there is no need to have your temporary store file this large.

### **Message buildup during a messaging engine failover**

A messaging system that consists of producers and consumers that are on separate remote messaging engines requires some consideration when planning for failover.

For example, suppose you have three servers:

- ▶ server1: Producer application and Messaging Engine 1
- ▶ server2: Consumer application and Messaging Engine 2
- ▶ server3: Failover server for server2/Messaging Engine 2

Standard behavior in this scenario would see messages being routed from the producer application on server1 through Messaging Engine 1 to Messaging Engine 2 on server2 and then to the consumer application. If, however, server2 fails over to server3, the messages being produced by the application on server1 build up in Messaging Engine 1 waiting for Messaging Engine 2 to come up. In the period of time in which Messaging Engine 2 is in the process of failing over and the producing application is still running, a buildup occurs in Messaging

Engine 1; you need to consider this build up when you plan your file store file sizes.

To estimate the amount of extra space you need in your file store files in this scenario, consider the size and throughput of messages being created on server1.

For example, if server1 is producing 1000 messages a second at a size of 10 KB, your system is creating  $1000 \times 10 \text{ K} \times 60 = 600000 \text{ K}$  message data each minute. So, for each minute that the failover of server2 takes, you have a build up of approximately 600 MB of data in server1. The normal production and consumption cycle of your file store does not need a large amount of file space (due to the short life of the messages). However, if you need to allow server2 three minutes to fail over, you must plan on your store files being able to hold more than 1800 MB of data. Again, you need to estimate this final value based on your expected failover time for server2, which you will determine through experimentation at development time.

### 5.1.3 File system requirements for file store

The file store relies heavily upon the capabilities of the underlying file system. In order to set up a reliable system, there are some very simple but important functions and guarantees that must be provided. In this section, we outline the basic requirements that a file store has on the file system it is using, and we give reasons for their importance. Because there are a large number of file system implementations available (NTFS, NFS, HPFS, SAN, and so on) we cannot discuss implementation specifics; however, by describing these requirements, we hope to provide you with the ability to determine if the file system you choose is suitable for your file store files.

#### **Exclusive read/write file locking**

To ensure exclusive access to the data stored in your file store files, your file systems need to support locking of files for read/write access. This requirement is especially important in high availability (HA) and clustering scenarios where your files might be accessed remotely, and the possibility that two servers might try to access them simultaneously. If the correct locking semantics are not supported by the file system hosting your file store files, two server processes could modify the data stored in them and, therefore, produce in an inconsistent state for both servers.

#### **Disk force on write**

When data is written to disk by the file store, it is written as part of the user transaction and it has a direct affect on the outcome of that transaction. To achieve the correct resolution of the user's transaction, you need to be

guaranteed at write time that the data that is written has made it to the physical disk.

For example, if the data is buffered in memory at write time, then the file store returns from the transaction successfully (commit) believing that the data is safe. If the buffered data is then lost because of a system failure, the system is inconsistent and the data for that transaction is lost.

This is very dangerous behavior which destroys the reliability of your messaging system. To avoid this situation, the file system used to support your file store files must ensure that when a `java.nio.channels.FileChannel.force()` is called, all waiting data is written to the physical disk (or, in the case of hardware-assisted file systems, at least guaranteed to make it to disk in the case of failure).

### **Accurate reporting of file space allocation**

Another area that affects the reliability of your file store is accurate allocation of file system space at allocation time. If the file store allocates 100 MB of space on disk, it must be able to rely on the availability of the full 100 MB to accurately guarantee the success of writing data to disk.

For example, when a file store allocates 100 MB of file space it is expecting an exclusively allocated piece of disk that is 100 MB in size to be available to it. If the file system checks the free space on the disk successfully but does not allocate the space for writing solely by the file store, another process could use up some of the space that makes up the 100 MB allocated by the file store. If the file store tries to write to disk, expecting there to be plenty of space left in the 100 MB it allocated, but fails because of lack of space, not only has a failure occurred but the file store now has no accurate way of gauging how much free space is available for use.

A problem like this leads to an inconsistency between the disk and the file store which will result in the system being unable to fulfill new requests reliably.

### **Do not use file system level compression**

Similar to the problem described above, using compression on your file system stops the file store from accurately measuring the amount of free space available in its files. Because the data written to disk is a compressed version of what the file store holds, the size of the data differs (at a variable rate), making any space calculations inaccurate.

## 5.2 Collect diagnostics

In addition to the SystemOut log files for the messaging engine and the application log collected in 1.3, “Collect diagnostics” on page 19, collect the first failure data capture (FFDC) logs, which are located in:

*profile\_root/logs/ffdc*

Each log file name is prefixed with the originating server name.

The FFDC logs are primarily intended for use by IBM Service. However, you might find it useful to search this repository as part of your problem determination exercise.

## 5.3 Analyze diagnostics

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261:

1. Inspect the SystemOut log for CWSIS0002E with a nested CWSOM1017E message as in Example 5-1.

*Example 5-1 CWSOM1017: Unable to access file store location*

---

**SibMessage E [:] CWSIS0002E:** The messaging engine encountered an exception while starting. Exception:  
com.ibm.ws.sib.msgstore.PersistenceException: Unexpected exception caught starting persistent message store: **CWSOM1017E: ObjectManagerState=ObjectManagerState(X:\Log)/Stopped ManagedObject(0/0)/Ready/26202620(ObjectManagerState) caught exception=X:\Log(Exception) trying to locate or create log file name={2}(String)**.

---

This message indicates a messaging engine is not able to access the file system locations that have been provided for use by the file store files.

Reasons why you could receive this message are:

- Incorrect path. the locations provided do not exist.
- User account not authorized. The user account under which WebSphere runs does not have authorization to read and write to the file system location.

2. To further narrow the appropriate action, check the application server FFDC log for an exception that matches the message seen in SystemOut:
  - **java.io.FileNotFoundException: path (The system cannot find the path specified.)**  
If you see this type of exception indicating the system could not find the path, note the path and see 5.4.1, “Incorrect path” on page 76.
  - **java.io.FileNotFoundException: path (Access is denied.)**  
If you see this type of exception indicating access was denied, see 5.4.2, “User account not authorized” on page 77.
3. Inspect the SystemOut log for CWSIS0002E with a nested CWSOM1042E message as in Example 5-2.

*Example 5-2 Messaging engine exception: CWSOM1042E*

---

```
SibMessage E [:] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: Unexpected exception
caught starting persistent message store: CWSOM1042E:
ObjectStore=AbstractObjectStore(F:\\PermanentStore)/69946994(ObjectStor
e) was asked to allocate space for
ManagedObject=ManagedObject(null/null)/Constructed/22f422f4(ManagedObje
ct) when it was full.
```

---

The CWSOM1042E message indicates a space constraint for the file store has prevented the messaging engine from starting.

If you see this message, see 5.4.3, “File store has a space constraint during startup” on page 78.

4. Inspect the SystemOut log for CWSIS1002E with a nested message in the range of CWSIS1573E - CWSIS1576E. You might also see CWSM1042.

These messages indicate that the file store experienced a space constraint during runtime. The CWSIS157xE message provides more detailed information about the file that failed as in Example 5-3.

*Example 5-3 CWSIS157xE*

---

```
SibMessage E [:] CWSIS1573E: The file store's log file is full.
SibMessage E [:] CWSIS1574E: The file store's permanent store file is
full.
SibMessage E [:] CWSIS1575E: The file store's temporary store file is
full.
```

---

If you see CWSIS1002E and any of these messages, see 5.4.4, “File store has a space constraint during runtime” on page 78.

## 5.4 Causes and solutions

The initial barrier to a working message file store is the successful creation of the files that store all persistent data. The most common reasons a messaging engine is unable to create file store files at startup are:

- ▶ Incorrect path
- ▶ User account not authorized

Given that space in the file store files is finite (even when the store files are set to “unlimited” your disk space is finite), it is likely that you will run out of space when you first experiment with tuning the size of your files to meet your requirements. The symptom is the messaging engine fails to start or an application is unable to complete a transaction. The most common reasons are:

- ▶ File store has a space constraint during startup.
- ▶ File store has a space constraint during runtime.

### 5.4.1 Incorrect path

The primary symptom of a messaging engine unable to access the file store location is that the messaging engine fails to start and a CWSOM1017E message is logged to SystemOut. The corresponding entry in the FFDC log looks like Figure 5-4.

*Example 5-4 PersistenceException: CWSOM1017E*

---

```
Exception = com.ibm.ws.sib.msgstore.PersistenceException
Source = com.ibm.ws.sib.msgstore.impl.MessageStoreImpl.start
probeid = 755
Stack Dump = com.ibm.ws.sib.msgstore.PersistenceException: Unexpected
exception caught starting persistent message store: CWSOM1017E:
ObjectManagerState=ObjectManagerState(X:\Log)/Stopped
ManagedObject(0/0)/Ready/26202620(ObjectManagerState) caught
exception=X:\Log(Exception) trying to locate or create log file
name={2}(String).
at
com.ibm.ws.sib.msgstore.persistence.objectManager.PersistentMessageStor
eImpl.start
at com.ibm.ws.sib.msgstore.impl.MessageStoreImpl.start
at com.ibm.ws.sib.admin.impl.JsMessagingEngineImpl.start
at com.ibm.ws.sib.admin.impl.HAManagerMessagingEngineImpl.activate
at com.ibm.ws.sib.admin.impl.JsActivationThread.run
Caused by: com.ibm.ws.objectManager.NonExistentLogFileException:
CWSOM1017E: ObjectManagerState=ObjectManagerState(X:\Log)/Stopped
```

```
ManagedObject(0/0)/Ready/26202620(ObjectManagerState) caught
exception=X:\Log(Exception) trying to locate or create log file
name={2}(String).
at com.ibm.ws.objectManager.ObjectManagerState.<init>
at com.ibm.ws.objectManager.ObjectManager.initialise
at com.ibm.ws.objectManager.ObjectManager.<init>
at
com.ibm.ws.sib.msgstore.persistence.objectManager.PersistentMessageStor
eImpl.start
... 4 more
Caused by: java.io.FileNotFoundException: X:\Log (The system cannot
find the path specified.)
```

---

The root cause of this exception is in the message for the `FileNotFoundException`: the system cannot find the path specified.

### **Solution**

Review the values you have set in the file store configuration to ensure that they refer to valid file system locations.

## **5.4.2 User account not authorized**

This problem is characterized by the following FFDC entry:

*Example 5-5 FileNotFoundException: Access is denied*

---

```
Caused by: java.io.FileNotFoundException: X:\Log (Access is denied.)
at java.io.RandomAccessFile.<init>
at java.io.RandomAccessFile.<init>
at com.ibm.ws.objectManager.ObjectManagerState.<init>
```

---

The root cause of this exception is in the message for the `FileNotFoundException`. The messaging engine is unable to create the necessary files because of security restrictions on the file system locations provided.

### **Solution**

Ensure that the user account under which your WebSphere installation is running has read/write access to the file system you have specified in the file store configuration.

### 5.4.3 File store has a space constraint during startup

If the file system you are using to store your file store files does not have the space required to create or grow the files as defined by the file store configuration parameters, then your messaging engine can fail to start.

The symptoms of this problem are that the messaging engine does not start and message CWSIS0002 and a nested CWSOM1042 are logged to SystemOut.

*Example 5-6 Messaging engine exception: CWSOM1042E*

---

```
SibMessage E [:] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: Unexpected exception
caught starting persistent message store: CWSOM1042E:
ObjectStore=AbstractObjectStore(F:\\PermanentStore)/69946994(ObjectStor
e) was asked to allocate space for
ManagedObject=ManagedObject(null/null)/Constructed/22f422f4(ManagedObje
ct) when it was full.
```

---

In this case, a lack of space was first noticed when trying to create the PermanentStore file; however, this problem could equally apply to the TemporaryStore or the log file.

#### **Solution**

Review the file store configuration to ensure there is enough disk space available at the file store location to store at least the minimum expected size of the files.

### 5.4.4 File store has a space constraint during runtime

If the file store system faces a space constraint, because of either disk space requirements or configured size limitations, you will experience problems with messaging applications during run time.

The symptoms of this problem include message CWSIS1002 with nested message in the range (CWSIS1573 to CWSIS1576) in the SystemOut log for the messaging engine.

The first sign of reaching the limit of one of your file store files will most likely be experienced by a messaging application in your WebSphere environment. When the application attempts to commit a transaction containing some messaging work, the transaction will roll back because of a lack of space in a file.



The following exception stack is an example for such a problem. This exception is seen by the user application and it is also written to the server's FFDC directory.

*Example 5-7 SIRollbackException: CWSIS1002E*

---

```
com.ibm.wsspi.sib.core.exception.SIRollbackException: CWSIS1002E: An
unexpected exception was caught during transaction completion.
at
com.ibm.ws.sib.msgstore.transactions.MSDelegatingLocalTransaction.commit
at com.ibm.ws.sib.msgstore.test.cache.RegisterMessage.testRegister
at com.ibm.js.test.LoggingTestCase.runTest
Caused by: com.ibm.ws.sib.msgstore.PersistenceFullException:
CWSIS1575E: The file store's temporary store file is full.
at
com.ibm.ws.sib.msgstore.persistence.objectManager.BatchingContextImpl.insert
at com.ibm.ws.sib.msgstore.task.AddTask.persist
at
com.ibm.ws.sib.msgstore.persistence.objectManager.PersistentMessageStoreImpl.commit
at
com.ibm.ws.sib.msgstore.transactions.MSDelegatingLocalTransaction.commit
Caused by: com.ibm.ws.objectManager.ObjectStoreFullException:
CWSOM1042E:
ObjectStore=AbstractObjectStore(X:\Build\SIB\ws\code\sib.msgstore.impl\
build/filestore\TemporaryStore)/3eb83eb8(ObjectStore) was asked to
allocate space for
ManagedObject=ManagedObject(null/null)/Constructed/2ca62ca6(Persistable
SlicedData[ BINARYDATA ])(ManagedObject) when it was full.
```

---

In the case above, the application transaction was rolled back because the store file that is needed to store the message on disk was full. In this example, the store file is clearly stated as the temporary store, which means that it was an ExpressNonPersistent or ReliableNonPersistent message being committed at the time.

When either of these asynchronous reliability levels is used, you might see another message that can alert you to a lack of space in your file store files:

```
SibMessage E [:] CWSIS1576E: The dispatcher is full.
```

## **Solution**

When you see that your system is constrained by a lack of free space in any of its file store files, review the current size limits to determine if they are suitable for your expected workload.

Check that there is sufficient space on the disk for your store files to grow to the upper limit you have set in the configuration.

### **5.4.5 Validate solution**

If the problem prevented the messaging engine from starting, restart the messaging engine, server or cluster and verify that it starts. Check that the error message is no longer being generated and written to the SystemOut logs.

If you were experiencing a space problem with the file stores, restart the messaging engine and the messaging applications. Check that the error messages are no longer being generated and written to the SystemOut logs.

## JMS application problem determination

This chapter describes how to diagnose and resolve problems that occur with JMS applications that use the default messaging provider. Problems include the inability to connect to a service integration bus and problems with sending (producing) or receiving (consuming) messages.

For more information about connecting applications to a service integration bus see *Connecting applications to a service integration bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001_.html)

## 6.1 Symptoms of a JMS application problem

Symptoms addressed in this chapter include:

- ▶ A JMS application is unable to create a connection to a service integration bus.
- ▶ A JMS application is unable to produce messages to the WebSphere Application Server default messaging provider
- ▶ JMS application unable to consume messages from the WebSphere Application Server default messaging provider.
- ▶ You see `javax.jms.JMSException` messages in the application or SystemOut logs.
- ▶ You see `com.ibm.ws.sib.comms.server.ObjectStoreFullException` messages in the application or SystemOut logs.
- ▶ Messages are lost.

## 6.2 Analyze diagnostics

The primary symptoms of a JMS application problem are found in the application messages. You might also find additional information in the SystemOut log for the messaging engine involved in the connection attempt.

### 6.2.1 Verify the messaging engine is started

If you expected the client application to connect to a specific messaging engine, first check that the messaging engine and the server on which it is running are started.

To determine the messaging engine to which the application is attempting to connect, see 1.2.2, “Determine the messaging engine connected to an application” on page 7.

To determine the state of a messaging engine, see 1.2.3, “Determine your messaging engine status” on page 8.

If the messaging engine is not started and will not start, you need to diagnose the cause of that problem first. See Chapter 2, “Messaging engine problem determination” on page 35.

## 6.2.2 Analyze application exception messages

Check the application log for an exception stack trace.

Search for `javax.jms.JMSEException` and determine what the application was attempting to do when it received this exception. Start with the `JMSEException` and note the messages for each of the linked exceptions until you reach the bottom. Then, you will understand the path the problem took through the system code:

1. If the exception occurred on a `createConnection`, `createQueueConnection`, or `createTopicConnection`:
  - If your client is *not* running on an application server, see 6.3, “JMS application outside the server container cannot connect to bus” on page 84.
  - If your client *is* running on an application server, see 6.4, “JMS application in server container cannot connect to the bus” on page 92.
2. If the exception occurred on a `createProducer`, `createQueueSender`, or `createTopicPublisher`, or when the application attempted to produce a message (calling `producer.send`), see 6.5, “JMS application unable to produce messages” on page 97.
3. If the exception occurred when the application attempted to create a consumer for a given destination by calling `createConsumer`, `createQueueReceiver`, or `createTopicSubscriber`, see 6.7, “JMS application unable to consume messages” on page 103.
4. If the exception occurred when the application attempted to consume a message by calling `consumer.receive`, or if the application was running in the client container and an exception was received on the `Connection` `ExceptionListener` when using a `MessageListener`, see 6.7, “JMS application unable to consume messages” on page 103.
5. If you do not see errors, and the application seems to have successfully sent the message but it never reaches the expected destination, see 6.6, “Message produced but does not arrive at destination” on page 102.
6. If you do not see errors, and the application seems to have successfully created a consumer for the destination but does not receive any messages (that is, calls to `receive` return `null`, or a `MessageListener` is not invoked) see 6.7, “JMS application unable to consume messages” on page 103.

## 6.3 JMS application outside the server container cannot connect to bus

When a JMS application runs outside of the application server (for example, in the J2EE client container or as a thin client in a J2SE™ environment) and the application calls the `createConnection` method to create a connection to the service integration bus, the following steps take place. (These steps are transparent to the application.):

1. A connection is created to one of the application servers in the cell that hosts the appropriate service integration bus. The server that is used is called the *bootstrap server*, and it must have the SIB Service enabled on it.
2. After the connection to a bootstrap server is established, the client application can query the bootstrap server to determine where to go to connect to a messaging engine on the specified bus. This query might also include additional constraints (called *target properties*) specified by the user on the JMS ConnectionFactory. These constraints limit the set of messaging engines on the bus to which the application will be allowed to connect; for example, they might specify connecting to any messaging engine inside a specified cluster bus member.
3. When a suitable target messaging engine has been found by the query, the messaging engine might be on a different server than the bootstrap server in which the query was executed. In this case, the application is redirected by the bootstrap server; it disconnects from the bootstrap server and creates a new connection to the server that hosts the target messaging engine. If the target messaging engine is hosted by the bootstrap server, the original connection is used instead of unnecessarily closing it and creating a new one.

To diagnosis a failure to connect to the bus, determine if:

1. The application was able to contact a bootstrap server.
2. The bootstrap server query successfully located a target messaging engine that matched the user specified target properties.
3. A redirect was necessary (the target messaging engine was on a different server) and if the redirect was successful.

### 6.3.1 Symptoms of a connection problem

The symptom of this type of problem is most apparent in the application messages.

The J2EE client application throws a `java.jms.JMSEException` when attempting to connect to the bus. The `JMSEException` is thrown by one of the following:

- ▶ `createConnection`
- ▶ `createQueueConnection`
- ▶ `createTopicConnection`

Error messages following the exception indicate the problem.

## 6.3.2 Analyze the exception in the application log

To diagnose the problem, first examine the `JMSEException` and its associated linked exceptions that were caught by the application. See the listing in Example 6-1 on page 86.

Search for the following messages:

- ▶ **javax.jms.JMSEException: CWSIA0241E:** An exception was received during the call to the method `JmsManagedConnectionFactoryImpl.createConnection:`  
`com.ibm.websphere.sib.exception.SIResourceException`

Note the full content of the message and continue searching the log for error messages.

- ▶ If you see any of these messages, see 6.3.3, “Unable to contact the bootstrap server” on page 88:
  - **CWSIT0006E:** It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: {0}
  - **CWSIT0007W:** It is not possible to contact the bootstrap server at {0} because of exception: {1}.
  - **CWSIJ0063E:** A network connection to host name {0}, port {1} cannot be established.
- ▶ If you see this message, see “Server does not permit client applications to connect” on page 89:
  - **CWSIT0090E:** A bootstrap request was made to bus {0} using channel chain {1}. Use of this chain is not permitted by bus {0}.
- ▶ If you see any these messages, see 6.3.4, “No messaging engine matches the target properties” on page 90:
  - **CWSIT0019E:** No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}

- **CWSIT0088E:** There are currently no messaging engines in bus {0} running. Additional failure information: {1}
- **CWSIT0102E:** A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.  
CWSIT0102E would be included as a cause of CWSIT0019E or CWSIT0088E (in the {2} or {1} insert position respectively).
- ▶ If you see this message, see 6.3.5, “Redirect to messaging engine fails” on page 91:
  - **CWSIT0092E:** An attempt was made to connect to bus {0} using channel chain {1}. Use of this chain is not permitted by bus {0}.

## Example

A JMSException will resemble Example 6-1.

### Example 6-1 JMSException CWSIA0241E

---

Caused by: **javax.jms.JMSException: CWSIA0241E:** An exception was received during the call to the method `JmsManagedConnectionFactoryImpl.createConnection:`  
`com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E:` It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to:  
`[localhost:7289:BootstrapSecureMessaging].`

at  
`com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManagedConnectionFactoryImpl.java:243)`  
 at  
`com.ibm.ws.sib.api.jms.impl.JmsQueueConnectionFactoryImpl.createQueueConnection(JmsQueueConnectionFactoryImpl.java:149)`  
 at  
`com.ibm.ws.sib.tools.explorer.helper.JMSHelper.getConnectionToME(JMSHelper.java:588)`  
 ... 3 more

Caused by: `com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E:` It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to:  
`[localhost:7289:BootstrapSecureMessaging]`

at  
`com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.generateException(TrmSICoreConnectionFactoryImpl.java:928)`  
 at  
`com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.remoteBootstrap(TrmSICoreConnectionFactoryImpl.java:637)`



```

    at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.createConnection(TrmSICoreCo
nnectionFactoryImpl.java:298)
    at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.createConnection(TrmSICoreCo
nnectionFactoryImpl.java:213)
    at
com.ibm.ws.sib.api.jmsra.impl.JmsJcaConnectionFactoryImpl.createCoreConnection(JmsJca
ConnectionFactoryImpl.java:672)
    at
com.ibm.ws.sib.api.jmsra.impl.JmsJcaConnectionFactoryImpl.createConnection(JmsJcaConn
ectionFactoryImpl.java:559)
    at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManag
edConnectionFactoryImpl.java:212)
    ... 5 more
Caused by: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0007W: It is not
possible to contact the bootstrap server at localhost:7289:BootstrapSecureMessaging
because of exception: com.ibm.websphere.sib.exception.SIResourceException:
CWSIC1001E: A client attempted to connect with a remote messaging engine
(localhost:7289 - BootstrapSecureMessaging) but the connection cannot be completed.
Ensure the messaging engine is started: exception
com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A network
connection to host name 127.0.0.1, port 7289 cannot be established...
    at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.contactBootstrapService(TrmS
ICoreConnectionFactoryImpl.java:752)
    at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.remoteBootstrap(TrmSICoreCon
nectionFactoryImpl.java:580)
    ... 10 more
Caused by: com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E: A client
attempted to connect with a remote messaging engine (localhost:7289 -
BootstrapSecureMessaging) but the connection cannot be completed. Ensure the
messaging engine is started: exception
com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A network
connection to host name 127.0.0.1, port 7289 cannot be established..
    at
com.ibm.ws.sib.comms.client.ClientSideConnection.connect(ClientSideConnection.java:35
0)
    at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.contactBootstrapService(TrmS
ICoreConnectionFactoryImpl.java:682)
    ... 11 more

```

```
Caused by: com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A
network connection to host name 127.0.0.1, port 7289 cannot be established.
    at
com.ibm.ws.sib.jfapchannel.impl.octracker.ConnectionDataGroup.connect(ConnectionDataG
roup.java:480)
:
```

---

In this example, the application could not contact the bootstrap server (CWSIT0006E) because it could not open a socket to the bootstrap server at the specified host and port (CWSIJ0063E).

### 6.3.3 Unable to contact the bootstrap server

The symptoms of this problem are:

- ▶ **CWSIT0006E**: It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: {0}
- ▶ **CWSIT0007W**: It is not possible to contact the bootstrap server at {0} because of exception: {1}.
- ▶ **CWSIJ0063E**: A network connection to host name {0}, port {1} cannot be established.

If the connection attempt failed because it was unable to contact the bootstrap server, you must resolve the problem that caused the failure. In most cases this problem is caused by one of the following situations.

#### **Provider endpoints in the connection factory were not set**

The provider endpoints field of the ConnectionFactory that defines where the bootstrap servers are located has not been set. Therefore, the provider endpoint has been defaulted, but the defaulted value does not point to a server.

#### **Solution**

Configure the provider endpoints field to contain the *host:port:chain* of one or more bootstrap servers as described in the following Information Center topic *Configuring a connection to a non-default bootstrap server* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjn0033\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjn0033_.html)

#### **Bootstrap server is not started**

The bootstrap servers are not started; therefore, a connection to the specified endpoint cannot be established.

## **Solution**

Ensure that the application server that you are expecting the client application to connect is running.

## **Server does not permit client applications to connect**

In some cases (particularly where security is enabled) the server might not permit client applications to connect using certain types of transport chain. For example, when bus security is enabled (the default) the use of the BootstrapBasicMessaging chain is not permitted.

The symptom for this problem is:

```
CWSIT0090E: A bootstrap request was made to bus {0} using channel chain {1}. Use of this chain is not permitted by bus {0}.
```

## **Solution**

Perform *one* of the following actions to resolve the problem:

- ▶ Modify the application connection factory to use a permitted chain (the preferred approach).
- ▶ Modify the server configuration to permit the use of the requested chain. For information about the use of permitted chains see the Information Center topic, *Transport security in service integration bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjr0490\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjr0490_.html)

## **Messaging engine port conflict**

If the messaging engine experienced a port conflict at startup, the application can experience the messages shown in Example 6-2.

*Example 6-2 JMS client symptom of a port conflict*

---

```
[:] CWSIT0007W: It is not possible to contact the bootstrap server at localhost:7276:BootstrapBasicMessaging because of exception: com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E: A client attempted to connect with a remote messaging engine (localhost:7276 - BootstrapBasicMessaging) but the connection cannot be completed. Ensure the messaging engine is started: exception com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A network connection to host name 127.0.0.1, port 7276 cannot be established...
```

```
javax.jms.JMSEException: CWSIA0241E: An exception was received during the call to the method
```

```
JmsManagedConnectionFactoryImpl.createConnection:
```

```
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It was
```

not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: [localhost:7276:BootstrapBasicMessaging].

---

This problem is due to a configuration error. If you see this message, review the ports assigned to the application server to ensure that there are no conflicts with the following ports:

- ▶ Service Integration Bus (SIB) port
- ▶ SIB secure port
- ▶ SIB MQ interoperability port
- ▶ SIB MQ interoperability secure port

If you suspect that there might be a port conflict, use a command such as **netstat** to view the ports that are currently in use.

### 6.3.4 No messaging engine matches the target properties

The symptoms of this problem are:

- ▶ **CWSIT0019E:** No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}
- ▶ **CWSIT0102E:** A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.
- ▶ **CWSIT0088E:** There are currently no messaging engines in bus {0} running. Additional failure information: {1}

CWSIT0102 would normally be seen as a cause of CWSIT0019 or CWSIT0088 (in the {2} or {1} insert positions, respectively).

If the application can contact to a bootstrap server, the process moves to the next step. The application queries the bootstrap server to locate a target messaging engine. These symptoms occur when it was not possible to locate a messaging engine that matched the target properties specified in the ConnectionFactory.

#### **Solution**

Check the target properties on the connection factory to ensure that they accurately describe the conditions that you wish to place on the connection. The properties to check include the BusName, Target, TargetType, TargetSignificance, TargetTransportChain and ConnectionProximity, and are quoted in the CWSIT0019E error message.

For information about the use of target properties see the Information Center topic, *Connecting applications to a service integration bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001_.html)

### ***ConnectionProximity property***

A CWSIT0102E error message indicates a matching messaging engine has been ruled out due to the proximity constraint.

The ConnectionProximity property is taken relative to the bootstrap server. If the proximity is set to Server and the bootstrap server does not contain a messaging engine matching the target properties, a connection will never be created.

### ***Messaging engine not running***

If you are satisfied that the target properties accurately describe the set of messaging engines that you wish to have considered for connection, ensure that one or more of those messaging engines is running. You might observe the error message CWSIT0088E if none of the messaging engines are running.

For information about determining and changing the state of a messaging engine, see 1.2.3, “Determine your messaging engine status” on page 8.

## **6.3.5 Redirect to messaging engine fails**

The symptom of this problem is this message:

```
CWSIT0092E: An attempt was made to connect to bus {0} using channel chain {1}. Use of this chain is not permitted by bus {0}.
```

If the bootstrap server query successfully located a target messaging engine that matched the target properties, the application will have received instructions from the bootstrap server to close its current connection and redirect to another server (bus member) in the cell that is hosting the messaging engine that was chosen for connection.

A failure at this point is very rare, but it can happen if the information returned by the query has since become out of date; for example, the server hosting the messaging engine has crashed, or the messaging engine has failed over from that server to another server in the cluster at the same time as the client application was attempting to connect. Failures of this type are transient and can usually be resolved by waiting a short time and then retrying the connection attempt.

One example of a non-transient failure at this point is the case in which the targetTransportChain is not permitted by the target server, as exposed in the

CWSIT0092E error message. You can resolve this by changing the application connection factory to use a permitted chain (preferred), or by modifying the server configuration to permit the requested chain. For information about the use of permitted chains, see the Information Center topic *Transport security in service integration bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjr0490\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjr0490_.html)

### 6.3.6 Validate the solution

The solution to your connection problem can be validated by re-running the client application so that it attempts to connect to the bus after you have made the necessary changes. The application should now connect successfully. If it does not do so, re-evaluate the exception information that is received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the connection from being made.

In some circumstances (for example, modifying the chains permitted by the server), you might need to restart the application servers in order for changes to take effect.

## 6.4 JMS application in server container cannot connect to the bus

A JMS application running inside a server container is part of a J2EE artifact such as an EJB™ or servlet (that is, an application running in the EJB container or the Web container, respectively). If your problem relates to receiving messages using an MDB, refer to Chapter 9, “Message-driven beans problem determination” on page 145.

Applications running inside a server do not follow the same bootstrap process as those running outside the server container (described in 6.3, “JMS application outside the server container cannot connect to bus” on page 84) for contacting a bootstrap server, because they are already running in that environment. The logic used to find an appropriate messaging engine to connect to is the same for both types of clients after the bootstrap server has been contacted; the system uses the target properties specified in the application ConnectionFactory to determine the set of messaging engines that are active and acceptable for the conditions requested by the application. Failures to connect from inside the server container (or from the client environment after the bootstrap server has been successfully contacted) are usually due to the relevant messaging engines

not being active, or because the target properties specified by the application do not match any of the messaging engines on the bus.

### 6.4.1 Symptoms of a connection problem

The symptom of this type of problem are most apparent in the application messages. The J2EE client application throws a `java.jms.JMSEException` when attempting to connect to bus. The `JMSEException` is thrown by one of the following:

- ▶ `createConnection`
- ▶ `createQueueConnection`
- ▶ `createTopicConnection`

Error messages following the exception indicate the problem.

### 6.4.2 Analyze the exception in the application log

To diagnose the problem, first examine the `JMSEException` and its associated linked exceptions that were caught by the application. This examination will often be enough to tell you why the connection attempt was not successful.

Search for these messages:

- ▶ `JMSEException` thrown by `createConnection`, `createQueueConnection`, or `createTopic` connection.

```
javax.jms.JMSEException: CWSIA0241E: An exception was received during the call to the method JmsManagedConnectionFactoryImpl.createConnection: com.ibm.websphere.sib.exception.SIResourceException
```

Note the full content of the message and continue searching the log for error messages.

- ▶ If you see this message, see 6.4.4, “Bus name does not exist” on page 95.  
**CWSIT0086E:** Bus {0} not found
- ▶ If you see these messages, see 6.4.3, “No messaging engine matches the target properties” on page 94:
  - **CWSIT0019E:** No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}
  - **CWSIT0102E:** A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.  
CWSIT0102E would normally be seen as a cause of CWSIT0019E.

- ▶ If you see this message, see 6.4.5, “No messaging engines active” on page 95.
 

**CWSIT0088E:** There are currently no messaging engines in bus {0} running. Additional failure information: {1}
- ▶ If you see this message, see 6.4.6, “Matching messaging engine found but not started” on page 96.
 

**CWSIT0104E:** The client attempted to connect to the messaging engine {0} on bus {1} but the connection could not be created because the messaging engine is not started.
- ▶ If you see any of these messages, see 6.4.7, “Security authorization failure” on page 96:
  - **CWSIT0016E:** The user ID {0} failed authentication in bus {1}.  
The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.
  - **CWSIT0022E:** The user ID {0} failed authentication in bus {1}.  
The user ID used to create an intra-bus messaging engine connection failed to be authenticated by the remote messaging engine.
  - **CWSIT0089E:** The user ID {0} failed authentication in bus {1}.  
The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.

### 6.4.3 No messaging engine matches the target properties

The symptoms of this problem are these messages:

- ▶ **CWSIT0019E:** No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}
- ▶ **CWSIT0102E:** A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.

The problem is that one or more messaging engines are running but none match the specified target properties.

#### Solution

Check the target properties to ensure that they are correct, or start a messaging engine that matches the existing target properties. For example, a simple spelling



mistake in the target field can prevent a connection from being made under any circumstances.

For information about the use of target properties see the Information Center topic, *Connecting applications to a service integration bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001_.html)

### ***ConnectionProximity property***

A CWSIT0102E error message indicates a matching messaging engine has been ruled out due to the proximity constraint.

When the application is running in a server environment, the ConnectionProximity property is taken relative to the server in which the application is running. CWSIT0102E occurs if the ConnectionProximity is set to Server and the local server does not contain a messaging engine.

## **6.4.4 Bus name does not exist**

The symptom of this problem is this message:

**CWSIT0086E:** Bus {0} not found

The bus name that was specified does not exist in the configuration.

### **Solution**

Modify the bus name property of the connection factory to refer to a bus that has been defined.

## **6.4.5 No messaging engines active**

The symptom of this problem is this message:

**CWSIT0088E:** There are currently no messaging engines in bus {0} running. Additional failure information: {1}

There are no messaging engines currently active in the specified bus.

### **Solution**

Start one or more messaging engines in the bus.

## 6.4.6 Matching messaging engine found but not started

The symptom of this problem is this message:

**CWSIT0104E:** The client attempted to connect to the messaging engine {0} on bus {1} but the connection could not be created because the messaging engine is not started.

A matching messaging engine was found in the local process but it is not in started state.

### Solution

Start the messaging engine and then retry the application.

## 6.4.7 Security authorization failure

The failure to connect is often caused by a security authorization or authentication constraint which might be revealed to the application as one of these messages.

Symptoms of an authorization problem include:

- ▶ **CWSIT0016E:** The user ID {0} failed authentication in bus {1}. The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.
- ▶ **CWSIT0022E:** The user ID {0} failed authentication in bus {1}. The user ID used to create an intra-bus messaging engine connection failed to be authenticated by the remote messaging engine.
- ▶ **CWSIT0089E:** The user ID {0} failed authentication in bus {1}. The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.

This problem is usually due to an incorrect user ID or password specified for the connection to the remote messaging engine.

### Solution

Ensure the user ID is authorized to connect to the bus.

See 17.5, "Authorization problems connecting to the bus" on page 255.

## 6.4.8 Validate the solution

Resolve the problem described in the exception received by the application and then retry the application connection attempt. In some circumstances (for example, where you have modified JNDI resources in order to fix the problem) you might need to restart the application servers for changes to take effect.

If the application does not successfully connect after you have made the appropriate changes, reevaluate the exception information that is now received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the connection from being made.

## 6.5 JMS application unable to produce messages

This topic describes the problems that you might encounter when trying to produce, or send, a message from within a JMS application. It is assumed that the application has successfully connected to the bus.

### 6.5.1 Symptoms that an application is unable to produce messages

The symptoms of this type of problem are most apparent in the application messages.

- ▶ The JMS application throws a `java.jms.JMSException` when attempting to create a producer for a given destination. It throws the `JMSException` in one of the following methods:
  - `createProducer`
  - `createQueueSender`
  - `createTopicPublisherError`

The messages following the exception indicate the problem.

- ▶ The JMS application throws a `java.jms.JMSException` when attempting to send a message (that is, calling `producer.send`).
- ▶ The application seems to have successfully sent the message but it never reaches the expected destination. No exception is received.

### 6.5.2 Analyze diagnostics

First, examine the `JMSException` and its associated linked exceptions that were caught by the application. This examination is often enough to tell you why the problem occurred.

Also, review the application server log files for the application process (server or client container) to see if there are any related entries around the time that the attempt was made to send the message.

## Analyze the application log

Search for these messages:

- ▶ If you see this message, see 6.5.4, “Cannot attach to the requested destination” on page 99.  
**CWSIA0062E:** Failed to create a MessageProducer for {0}
- ▶ If you see this message, see 6.5.5, “Insufficient authorization to create a producer for the queue” on page 100.  
**CWSIA0069E:** The user does not have authorization to carry out this operation. See the linked exception for details.
- ▶ If you see this exception, see 6.5.6, “Application not closing producer objects” on page 100.  
`com.ibm.ws.sib.comms.server.ObjectStoreFullException`
- ▶ If you see any of this messages, see 6.5.7, “Current system state problems” on page 101.
  - **CWSIA0063E:** Failed to send to {0}
  - **CWSIK0025E:** The destination {0} on messaging engine {1} is not available because the high limit for the number of messages for this destination has already been reached.
  - **CWSIK0033E:** The message reliability value {0} is greater than the destination reliability value {1} for destination {2} on messaging engine {3}.
- ▶ Look for this message:  
**CWSIA0067E:** An exception was received during the call to the method {1}: {0}.  
Most other errors have CWSIA0067E as the top level message, and a detailed description of the problem is provided in the linked exceptions. In those situations, follow the problem diagnosis instructions for the type of error described in the linked exception.

If you do not see error messages, it is likely that the application successfully produced the message. If the message, however, does not arrive at the destination, see 6.6, “Message produced but does not arrive at destination” on page 102.

### 6.5.3 Causes and solutions

The most likely causes for failing to send a message are listed below along with references to the sections that describe how to resolve the problems.

Failures that occur while trying to create a JMS MessageProducer, QueueSender, or TopicPublisher include:

- ▶ CWSIA0062E - See 6.5.4, “Cannot attach to the requested destination” on page 99.
- ▶ CWSIA0069E - See 6.5.5, “Insufficient authorization to create a producer for the queue” on page 100.
- ▶ CWSIA0055E - See 6.5.6, “Application not closing producer objects” on page 100.

Failures that occur while trying to send a message (that is, calling `producer.send()`) include:

- ▶ CWSIA0069E - See 6.5.5, “Insufficient authorization to create a producer for the queue” on page 100.
- ▶ CWSIA0063E - See 6.5.7, “Current system state problems” on page 101. These problems include:
  - CWSIK0025E - “Queue is full” on page 101
  - CWSIK0033E - “Destination maximum reliability lower than message reliability” on page 101.

If the application seems to have successfully sent the message but it never reaches the expected destination, see 6.6, “Message produced but does not arrive at destination” on page 102.

### 6.5.4 Cannot attach to the requested destination

The symptom of this problem is this message:

```
CWSIA0062E: Failed to create a MessageProducer for {0}
```

CWSIA0062E indicates that it was not possible to attach to the requested destination. The linked exception provides more details on the exact reason for the failure:

- ▶ This problem is usually because the requested destination has not been defined or an error was made in the JNDI destination objects.
- ▶ A variation on this problem is that the producer is being created for a temporary destination (usually to send a reply message) but the temporary

destination has been deleted, either because the originating application has explicitly asked for it to be deleted or because the originating application has disconnected, causing the temporary destination to be automatically deleted.

- ▶ In rare cases this problem might also indicate that the producer was expecting to connect to a queue and the destination is actually a topic space, or the other way around.

### 6.5.5 Insufficient authorization to create a producer for the queue

The symptom of this problem is this message:

**CWSIA0069E:** The user does not have authorization to carry out this operation. See the linked exception for details.

CWSIA0069E indicates that the application does not have authorization to create a producer for the requested queue.

#### **Solution**

The linked exception provides further details.

If the message does not provide enough information to resolve the problem, see 17.6, “Authorization problems accessing a destination” on page 258 for further problem determination information.

### 6.5.6 Application not closing producer objects

The symptom of this problem is this message:

**com.ibm.ws.sib.comms.server.ObjectStoreFullException**

In long running or high turnover scenarios a badly behaved application might receive a linked exception containing a `com.ibm.ws.sib.comms.server.ObjectStoreFullException`. This exception indicates that the application has created thousands of producer objects without closing any of them. This exception is usually preceded by CWSIA0055E errors in the application server logs of the appropriate environment.

#### **Solution**

Modify the application so that it closes the `MessageProducer` object after it has finished using it.

## 6.5.7 Current system state problems

The symptoms of this problem include these messages:

- ▶ **CWSIA0063E:** Failed to send to {0}
- ▶ **CWSIK0025E:** The destination {0} on messaging engine {1} is not available because the high limit for the number of messages for this destination has already been reached.
- ▶ **CWSIK0033E:** The message reliability value {0} is greater than the destination reliability value {1} for destination {2} on messaging engine {3}.

Errors related to the current state of the system are reported as CWSIA0063E which contains a linked exception providing detailed information about what went wrong.

### Queue is full

CWSIK0025E indicates that the destination is not available because it has reached the limit of messages that it has been configured to store; that is, the queue is full. The corrective action is to start the consuming application to cause messages to be consumed from the appropriate destination. After some messages have been consumed from the destination the sending application can send messages again.

**Note:** This message can occur for other reasons than the queue is full; for example, the queue is not found. The problem has been reported to IBM technical support.

### Destination maximum reliability lower than message reliability

CWSIK0033E occurs when the service integration bus destination object has been configured with a maximum reliability which is lower than the reliability of the message that the producer is attempting to send.

## 6.5.8 Validate the solution

Resolve the problem described in the exception received by the application, and trigger the application so that it attempts to send another message. In some circumstances (for example, if you have made modifications to a JNDI resource such as a Queue object) you might need to restart the application servers so that changes take effect.

If the application does not successfully send the message after you have made the appropriate changes, re-evaluate the exception information that is now

received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the send from being successful.

## 6.6 Message produced but does not arrive at destination

In the event that the message was produced by the application without error but did not arrive at the destination, you can take the following actions to check for common problems.

**Note:** It is possible that an error is produced but either not caught and displayed by the application, or it occurred on a remote system, such as a foreign bus. Ensure that your application is catching and displaying all `JMSEExceptions`.

### 6.6.1 Verify the transaction was committed

If the message was sent under a transaction, verify that the transaction has been committed. To determine if a message is still locked under a transaction, look at the runtime panel for the destination in question (for example the queue point) and view the available messages. A message marked `locked` is being processed under a transaction by another consumer.

If the transaction has not been committed, the message is not yet available for consumption by other applications. If the message is locked under a transaction, it has not been committed yet. If the transaction has been committed, the message is no longer visible on the queue.

In the server environment, control of the transaction is often handled by the container in which the application is running (for example, when using container-managed transactions). However, it might be under the control of the application (bean-managed transactions). In the client container or thin client environments, the transaction is always controlled by the application using the `session.commit` and `session.rollback` methods.

### 6.6.2 Verify the message was sent to the correct destination

It is surprisingly common for a message to be sent to a different destination than was expected.



Check which JNDI destination object is used by the application, and then check the contents of that destination object to ensure it matches your expectations.

### 6.6.3 Check for the message on the exception destination

Check the exception destination to see if your message has been moved there for some reason. If this is the case, the exception header will provide details about why it was moved there.

For information about determining if a the message is on an exception destination, see “Check the exception destination” on page 18.

### 6.6.4 Find the message

To determine how far through the system the message has traversed, see 1.2.4, “Finding queued messages” on page 9.

## 6.7 JMS application unable to consume messages

This topic describes some of the problems you might encounter when trying to receive a message using a JMS application. It is assumed that the application has successfully connected to the bus.

### 6.7.1 Symptoms that an application is unable to consume messages

The symptom of this type of problem will be most apparent in the application messages.

- ▶ The JMS application throws a `java.jms.JMSEException` when attempting to create a consumer for a given destination. The `JMSEException` is thrown by one of the following methods:
  - `createConsumer`
  - `createQueueReceiver`
  - `createTopicSubscriber`

The messages following the exception indicate the problem.

- ▶ The JMS application throws a `java.jms.JMSEException` when attempting to consume a message (that is, calling `consumer.receive`). Alternatively, if the application is running in the client container, an exception is received on the `ConnectionExceptionListener` when using a `MessageListener`.

- ▶ The application seems to have successfully created a consumer for the destination but does not receive any messages (that is, calls to `receive` return `null`, or a `MessageListener` is not invoked). No exception is received.

## 6.7.2 Verify integrity

If the JMS application has received a `JMSException`, the exception provides the primary source of diagnostic information for resolving this type of problem. In most cases, the cause is obvious when you examine the exception messages that caused the failure.

If the application did not receive an exception, but a call to `consumer.receive` returns `null`, there might still be a problem if you expect the message to be there.

**Note:** A call to `consumer.receive` returns `null` if there was no message available. The method returns successfully (with no exception) but you do not have a message to process. This result can be expected in some situations (for example, the producer might not have sent a message since the last time you checked). However, it could also indicate a problem.

Check the following possibilities first:

- ▶ Was the message successfully sent by the producing application?  
Look for a `JMSException` in the application log that indicates a problem producing messages (see 6.2, “Analyze diagnostics” on page 82). If these types of errors exist, address them first.
- ▶ Has the JMS connection been started?  
The JMS specification requires that messages not be delivered to consumers (synchronous or asynchronous) until the connection has been started. A `CWSIA0087W` warning message is returned to the application.  
If you see this message, modify your application to start the connection before calling `receive`.
- ▶ Did the message reach the destination from which the consumer is trying to receive the message?  
Messages can often be rerouted by business logic in mediations or administratively defined forward routing paths. Check that the producer and consumer are configured to produce and consume in the appropriate locations. In the case of queues or durable subscriptions, you can check the status by stopping the consuming application and using the runtime administration panel view of the queue point or subscription to view the available messages.

- ▶ If consuming from a queue, has the message already been consumed by another application or by a different part of the same application?

This assessment usually requires a level of knowledge about what is going on in the system to identify. For example, the administrator would know whether other applications consume from the same destination.

See 1.2.4, “Finding queued messages” on page 9 for help in tracking down missing messages.

An advanced user might wish to turn on the SIB message trace to help with the identification process. To enable the trace using the administrative console, select **Troubleshooting** → **Logs and Traces** → *server\_name* → **Change Log Details Levels**. On the Runtime tab, add `SIBMessageTrace=all=enabled` to the Groups, and click **Apply**.

- ▶ Has the message been locked by another application that is part way through processing it?

To determine if this is the case, look at the runtime panel for the destination in question (for example the queue point) and view the available messages. A message marked locked is being processed under a transaction by another consumer.

- ▶ Check the exception destination to see if the message that was sent by the producer had a problem that meant it could not be sent to the target destination. In this case the message exception header provides details about what went wrong.
- ▶ The message also might not be available if it was sent with a reliability which caused it to be discarded (for example, if the message was sent as non-persistent and the server holding it was restarted). The message might have been discarded, even if the server was not restarted, if it was sent with a reliability of `BestEffortNonPersistent` and the system had constrained resources.

### 6.7.3 Analyze diagnostics

First, examine the `JMSException` and its associated linked exceptions that were caught by the application. This examination will often be enough to tell you why the attempt to consume a message was not successful.

Also, review the application server log files for the application process (server or client container) to see if there are any related entries around the time that the attempt was made to receive the message.

## Analyze the application log

Search for the following messages:

- ▶ If you see this message, 6.7.4, “Cannot attach to the destination” on page 106.  
**CWSIA0086E:** Failed to create a MessageConsumer for {0}
- ▶ If you see this message, 6.7.5, “Not authorized for the requested queue” on page 107.  
**CWSIA0090E:** The user does not have authorization to carry out this operation. See the linked exception for details.
- ▶ If you see these messages, 6.7.6, “Application not closing consumer objects” on page 107.  
**com.ibm.ws.sib.comms.server.ObjectStoreFullException**
- ▶ If you see this message, examine the other associated messages.  
**CWSIA0085E:** An exception was received during the call to the method {1}: {0}.  
Most other errors generated when receiving messages have CWSIA0085 as the top level message, and detailed description of the problem is provided in the linked exceptions

### 6.7.4 Cannot attach to the destination

The symptom of this problem is this message:

**CWSIA0086E:** Failed to create a MessageConsumer for {0}

CWSIA0086E indicates that it was not possible to attach to the requested destination. The linked exception provides more details on the exact reason for the failure and indicates what needs to be done to resolve it.

This problem is usually because the requested destination has not been defined (or an error was made in the JNDI Destination object).

A variation on this problem occurs when the consumer is being created for a temporary destination (usually to consume a reply message) but the temporary destination has been deleted by the application. Alternatively, this application might not be permitted to consume from the temporary destination. The JMS specification requires that only the application that creates a temporary destination can consume from it.

In rare cases this problem might also indicate that the consumer was expecting to connect to a queue and the destination is a topic space, or the other way around.

### 6.7.5 Not authorized for the requested queue

The symptom of this problem is this message:

CWSIA0090E: The user does not have authorization to carry out this operation. See the linked exception for details.

CWSIA0090E indicates that the application does not have authorization to create a consumer for the requested queue. The linked exception provides further details.

If the message does not provide enough information to resolve the problem, see 17.6, “Authorization problems accessing a destination” on page 258 for more problem determination information.

### 6.7.6 Application not closing consumer objects

The symptom of this problem is the following exception:

**`com.ibm.ws.sib.comms.server.ObjectStoreFullException`**

In long running or high turnover scenarios, a badly behaved application might receive a linked exception containing a `com.ibm.ws.sib.comms.server.ObjectStoreFullException`. This exception indicates that the application has created thousands of consumer objects without closing any of them. This is usually preceded by CWSIA0059E messages in the J2EE application log, client console or application server log, depending upon where the application is running. In this situation, modify the application to close the `MessageConsumer` object after it has finished using it.


### 6.7.7 Validate the solution

Resolve the problem described in the exception received by the application, and trigger the application attempt to consume the message again. In some circumstances (for example, when you have modified JNDI resources to resolve the problem), you might need to restart the application servers for changes to take effect.

If the application does not successfully send the message after you have made the appropriate changes, re-evaluate the exception information that is now received by the application to determine whether the original problem has

continued to occur, or whether a new problem is preventing the send from being successful.

Archived



## Messaging in a multiple messaging engine environment

Chapter 6, “JMS application problem determination” on page 81 discusses problems that prevent a JMS application from connecting to a service integration bus and problems that arise when an application attempts to send or receive messages using the default messaging provider.

This chapter covers JMS application problems that are specific to an environment that includes multiple messaging engines. In this type of topology, the producing and consuming applications might be connected to different messaging engines and, in the case of point-to-point messaging, might be connected to a messaging engine that is remote to the location of the message point. This chapter discusses problems specific to this type of messaging topology.

## 7.1 Symptoms of problems in a multiple messaging engine environment

The following symptoms are specific to networking in an environment with multiple messaging engines. If you experience any of these symptoms, see the referenced section:

- ▶ Messages are not found when expected on the queue.  
Messages have been produced to a destination by a messaging application but, the messages cannot be seen on the destination's message points or arrive after an unexpectedly long period of time.  
See 7.3, "Messages are lost or are slow to arrive" on page 111.
- ▶ Messages are not consumed from the queue when expected.  
A consuming application does not receive any messages even when the messages can be seen on the queue's queue points.  
See 7.4, "Messages are not consumed or consumed slowly" on page 113.
- ▶ Application fails with an `SIMPLimitExceededException` failure while sending messages to a queue.  
See 7.5, "SIMPLimitExceededException failure while sending messages to a queue" on page 114.
- ▶ Application fails with an `SIMPLimitExceededException` failure while sending messages to a topic space.  
See 7.6, "SIMPLimitExceededException failure while sending messages to a topic space" on page 116.

If these symptoms do not match your problem or if you are not sure, continue to 7.2, "Analyze diagnostics" on page 110 to collect and analyze log files.

## 7.2 Analyze diagnostics

Scan each log for indications of an error, as described below.

### 7.2.1 Analyze the application log

In the application log, look for `JMSEExceptions`.



Specifically, scan for `SIMPLimitExceededException`. If you find this exception, see the appropriate section for your environment:

- ▶ 7.5, “`SIMPLimitExceededException` failure while sending messages to a queue” on page 114
- ▶ 7.6, “`SIMPLimitExceededException` failure while sending messages to a topic space” on page 116.

If you do not find this message but you find other `JMSEExceptions`, see 1.6, “Guide to chapters by exception” on page 28.

## 7.2.2 Analyze SystemOut for the messaging engines

For messages to be transmitted between remote queue points and the local queue point (and hence messaging engines), the respective messaging engines must communicate with each other. The connectivity status between messaging engines within a bus is reported in the respective messaging engine’s application server log files.

In the messaging engine `SystemOut` log, scan for error messages with a prefix of `CWSI`:

- ▶ If you see `CWSIT0029I`, see 7.3, “Messages are lost or are slow to arrive” on page 111.
- ▶ If you see error messages other than this, check the list of messages in 1.5, “Guide to chapters by message” on page 22.

## 7.3 Messages are lost or are slow to arrive

If an application is connected to a different messaging engine than the one on which the identified message points are located, messages might be in transit between messaging engines.

For each possible messaging engine that the producer could have connected to, check the state of the remote queue point on the messaging engine.

Examine the message points for the messaging engine. If the messaging engine has been used to produce messages to the queue point in question, there will be a remote queue point, which identifies it (and the messaging engine hosting that queue point).

If the queue point is clustered, there might be multiple remote queue points.

For each remote queue point:

1. Check to see if there are any messages queued for transmission.

Check the current outbound messages count on the remote queue point panel. If this value is greater than zero:

- a. Verify the messaging engine identified as hosting the message point is running. If it is not, start the messaging engine and check that the messages arrive.
- b. Check that the two messaging engines are able to communicate.

Examine the application server log files and search for occurrences of the message code **CWSIT0029I**. If you find it, check the message body to see if the messaging engines match those hosting any queue points that interest you. For example:

```
CWSIT0029I: The connection for messaging engine  
Messaging_Engine_1 in bus Bus_Name to messaging engine  
Messaging_Engine_2 stopped.
```

This message generally implies some kind of network connectivity problem. Resolving connectivity issues enables the messages to be delivered to the queue point.

- c. Select the remote message point and then select outbound messages.

Examine the state of the first message.

If it is in the *committing* state, the transaction used to produce the message has not been successfully committed, which blocks this first message and subsequent messages from being transmitted. Resolve any problem with the associated transaction.

If the state is *pending send*, the target messaging engine cannot be contacted. Check the application server log files to ensure that the messaging engine is running and that no communication errors have been reported.

2. If no messages are queued for transmission, but the number of outbound messages sent is greater than zero, then messages have been successfully sent from this messaging engine to the message point. Reconfirm that the message has not arrived and been removed from the queue point (for example, consumed by an application, expired, or moved to an exception destination).

## 7.4 Messages are not consumed or consumed slowly

When you think messages are being consumed slowly or not at all, your first step in diagnosing the problem is to find and examine the messages:

1. For each queue point containing messages, display the messages.

If messages are available but are in a *locked* state, resolve why they are not available:

- If a message is locked with a transaction ID, then the message has been deleted under a transaction that has not yet been committed.
- If a message is locked with no transaction ID and stays in the locked state for a significant period of time, it is likely that it is in the process of being consumed by an application connected to another messaging engine.

If no available messages were found, see 7.3, “Messages are lost or are slow to arrive” on page 111.

2. If messages are found and the destination consists of more than one queue point, determine if messages are found on all queue points or a subset of the queue points.

A consumer will only consume from a single message point. A consumer might be attached to a message point that does not contain any messages. If a clustered queue is being used, you must ensure that all queue points have an active consumer to be sure that messages will be consumed.

For more information about this behavior see *Workload sharing with queue destinations* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjt0014\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjt0014_.html)

If an *application is connected to a messaging engine that owns one of the messaging points*, the application always consumes from only that queue point. If no messages are available on the queue point, then the application has no messages to consume. If messages are available, see 6.7, “JMS application unable to consume messages” on page 103 for general JMS consumer problem determination information.

If an *application is connected to a messaging engine that does not own a queue point*, the messaging engine attaches to any one of the remote queue points. If the application is running, you can see which queue points are attached by inspecting the remote queue points on the messaging engine where the application is connected:

1. Determine which messaging engine to which the application is connected.
2. Identify the remote queue points. If this messaging engine is currently, or was recently, being used to consume messages from the message point on a remote messaging engine, there will be a remote queue point for it. If the message point is clustered, there might be multiple remote message points.

3. Perform the following for each remote queue point:

- a. If the consuming application is currently waiting for a message, one of the remote message points has at least one current message request listed.

If none are found, the consuming application is not connected to this messaging engine or is not actively consuming. Check the application for errors.

Otherwise, if a message request is found, note the selector associated with the request. View the message point that corresponds to the remote message point to see if there are available messages that would match the selector. It is likely that there are no messages available on this particular message point suitable for the consuming application.

If there are messages, proceed to the next step.

- b. If the consuming application is not currently waiting for a message or even connected it is harder to diagnose a possible reason for failing to consume messages.

In this case, the completed message request property of a remote message point might be used to give an indication of which message point was being consumed from. Use this request property to inspect the message point to see if it contains any available messages. If it does not, it is likely that this is why the consumer did not receive any messages. Otherwise, if possible, re-run the consuming application and follow the previous steps.

## 7.5 **SIMPLimitExceededException failure while sending messages to a queue**

This section discusses steps to take to diagnose the problem when a producing application's send of a message to a *queue* fails with an

`SIMPLimitExceededException`. If the application is a JMS application, this exception will be wrapped by a `JMSEException`.

## 7.5.1 Examine the queue point message depths

This error implies that the queue point (or remote queue point) for which the application is producing has exceeded its high message threshold.

To identify where the problem lies:

1. Determine the following information:
  - The messaging engine to which your application is connected.
  - The appropriate queue and, where applicable, any mediation points.
2. Check the current message depth of each queue point.

If the current depth is at or above the high message threshold, the queue point is full. Reasons why this might occur are that:

- a. No consumer is actively consuming from this queue point.  
Ensure that each queue point has an active consumer.
- b. The active consumers cannot consume messages at a rate equal to the rate at which the producers are producing them. You can resolve this in one of two ways:
  - Increase the speed of the consumer applications; for example, increase the number of threads used by an MDB or connect consuming applications directly to this messaging engine.
  - Reduce the speed of the producing applications.

If one or more of the queue points has not reached the high message threshold, the failing producing application might be connected to a messaging engine that does not own the queue point for the destination. A remote queue point has been created by this messaging engine and the messages have been built up here:

1. Identify which messaging engines the producing application was or is connected to.
2. For each messaging engine, identify the remote queue points.
3. Check the current outbound message value of the remote queue point. If this value is high or it has reached the high message threshold of the messaging engine, check the following areas:
  - a. Determine if the outbound messages sent value is increasing over time. Refresh the panel after a few seconds to see if the number changes:

- i. If the `outbound messages sent` value is *not increasing*, make sure the messaging engine that owns the queue point is running. If not, start it. If it fails to start, see Chapter 2, “Messaging engine problem determination” on page 35.

After the messaging engine is started, the outbound messages should drain from the remote queue point into the queue point. If the messages are not sent, examine the application server log files and search for occurrences of message CWSIT0029I. If you find this message, check the message body to see if the messaging engines match those hosting any queue points that interest you.

For example:

```
CWSIT0029I: The connection for messaging engine
Messaging_Engine_1 in bus Bus_Name to messaging engine
Messaging_Engine_2 stopped.
```

This message generally implies a network connectivity problem. Resolving connectivity issues enables the messages to be delivered to the queue point.

- ii. If the `outbound messages sent` value is *increasing*, messages are successfully being transmitted to the corresponding queue point. The reason for the large number of outbound messages is that the rate of message production is faster than they can be transmitted. This situation might be due to a slow network connection between messaging engines or to other network traffic interfering with this connection’s performance. Improving the network connection should increase message throughput. If it does not, the message production rate must be reduced to prevent the backlog.

## 7.6 SIMPLimitExceededException failure while sending messages to a topic space

This section discusses steps to take to diagnose the problem when a producing application’s send of a message to a *topic space* fails with an `SIMPLimitExceededException`. If the application is a JMS application, this exception will be wrapped by a `JMSException`.

### 7.6.1 Examine the publication point message depths

This error implies that the publication point for which that the application is producing has exceeded its high message threshold. This situation occurs if you have one or more subscriptions with a large number of messages queued on it.

These subscriptions are either application defined subscriptions or system level subscriptions, made by publication points on neighboring messaging engines.

Application subscriptions build up messages if no application is currently consuming messages from them, or if their consumption rate is slower than the message production rate.

Neighboring publication point subscriptions might build up messages if the neighboring messaging engine is not running, there is a problem communicating with it, or the message production rate is too fast for the network connection.

Neighboring publication points are represented on a local messaging engine as a remote publication point. The remote publication points transmit messages reliably from publishing applications connected to one messaging engine to subscriptions created on different messaging engines.

To diagnose which of these is causing the problem:

1. Determine the topic space for which the application is producing messages.
2. Identify the messaging engine to which the producing application is connected.
3. Display the publication point for the topic space on the connected messaging engine.
4. Select **Subscriptions** to list all current subscriptions.
5. Select each subscription to display its current message depth.

A subscription with a number approaching the high message threshold of the publication point might be causing further production of messages to fail. If you find that this is the case, determine the reason for the high number of messages. The reason for this result is most likely one of these conditions:

- No application is actively consuming the messages.

Start an application or delete the subscription (if it is a durable subscription).

- The consuming application cannot consume at the same rate as the producing applications are producing messages.

Reduce the message rate of the producer.

6. If no local subscriptions have a backlog of messages, check for any remote publication points on the messaging engine.
7. Check the current outbound messages sent count for each remote publication point. If it is high or has reached the high message threshold of the messaging engine, check the following conditions:

- a. If the outbound messages sent value is *not increasing*, make sure the messaging engine that owns the publication point is running. If not, start it. If it fails to start, see Chapter 2, “Messaging engine problem determination” on page 35.

After the messaging engine is started, the outbound messages should drain from the remote publication point into the publication point and subscriptions. If the messages are not sent, examine the application server log files and search for occurrences of message CWSIT0029I. If you find this message, check the message body to see if the messaging engines match those hosting any queue points that interest you.

For example:

```
CWSIT0029I: The connection for messaging engine
Messaging_Engine_1 in bus Bus_Name to messaging engine
Messaging_Engine_2 stopped.
```

This message generally implies a network connectivity problem. Resolving connectivity issues enables the messages to be delivered to the queue point.

- b. If the outbound messages sent value is *increasing*, messages are successfully being transmitted to the corresponding publication point. The reason for the large number of outbound messages is that the rate of message production is faster than they can be transmitted. This condition can be due to a slow network connection between messaging engines or to other network traffic interfering with this connection’s performance. Improving the network connection should increase message throughput. If it does not, the message production rate must be reduced to prevent the backlog.





## Clustering problem determination

This chapter walks you through the process of debugging WebSphere Application Server cluster problems associated with the default messaging provider.

## 8.1 Introduction to clustering for messaging

When configuring a clustered environment for messaging using the default messaging provider, an installation can follow one of these two approaches:

- ▶ Configure and use a cluster in a *highly available (HA) configuration*. In this configuration, only one messaging engine is active on one of the cluster members at any given time.
- ▶ Configure and use a cluster in a *workload management (WLM) configuration*. In this configuration, a number of active messaging engines equal to the number of cluster members are active simultaneously, each specifically pinned to a cluster member.

The decision for the style of the messaging cluster is made at configuration time, after the WebSphere cluster and cluster members have been created.

**Note:** A topic space cannot be workload-balanced in a WLM cluster.

For more information about messaging engine clustering, see:

- ▶ *Configure a Service Integration Bus in a network deployment environment*  
<http://www.ibm.com/developerworks/webservices/library/ws-sibus/>
- ▶ *Configure platform messaging and Web Services Gateway for a clustered environment, Part 1: Cluster enhancements for WebSphere Application Server Version 6.1 on z/OS*  
<http://www.ibm.com/developerworks/webservices/library/ws-clusterenv1/index.html>
- ▶ *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

### 8.1.1 Symptoms of problems specific to clustering

The problems involved in the use of a clustering for the default messaging provider fall into two categories: configuration problems and runtime problems:

- ▶ The primary symptom of a configuration problem is that one or more messaging engines fail to start.

- ▶ The primary symptom of a runtime problem is the non-delivery or orphaning of messages. Another symptom is messaging engines starting in the wrong location.

**Note:** Some clustering problems might have symptoms of data store errors, specifically message CWSIS1546I. This problem can occur when two instances of a messaging engine attempt to connect to the same data store; the first successfully obtains the lock, and the second receives multiple CWSIS errors. This behavior is intentional. The data store prevents your messaging engine from starting in two places; so the data store prevents the potential result of corrupted messaging data. For more information about data store locking see Chapter 4, “Message store with data store persistence” on page 49.

## 8.2 Analyze diagnostics

In an HA configuration, although only one messaging engine is active at a time, it can run on any cluster member. You might need to look at the log for each cluster member to determine where that messaging engine is located.

In a WLM configuration, there is one active messaging engine per cluster member. You need to look at the SystemOut log for each member to determine the reason for a specific messaging engine not starting.

Analyze each SystemOut log to determine where the messaging engine attempted to start and if it was successful.

For each of the cluster members, view the SystemOut log until you find the log entries with the messaging engine. Look for messages that indicate a messaging engine has attempted to start and failed. In a WLM configuration, one messaging engine per cluster member should be started.

Search for messages with the following format:

- ▶ CWSIDxxxxI
- ▶ CWSIDxxxxE
- ▶ CWSISxxxxE

### 8.2.1 Determine if the messaging engine started normally

The log for a messaging engine normal start will look similar to Example 8-1 on page 122:

*Example 8-1 Messaging engine start up with file store*

---

```
WsServerImpl A WSVR0001I: Server server1 open for e-business
:
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Joined.
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Starting.
:
SibMessage I [:] CWSIS1566I: A single previous owner was found in
the messaging engine's file store, ME_UUID=1D842649652FA294,
INC_UUID=0E040E040F8E07CD
SibMessage I [:] CWSIS1563I: The messaging engine,
ME_UUID=1D842649652FA294, INC_UUID=0D800D800FA453BF, has acquired an
exclusive lock on the file store.
:
SibMessage I [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Started.
```

---

**Note:** If the messaging engines appear to have started normally on all applicable cluster members, see “Message delivery problems” on page 123.

A start up that failed will have error messages of the format CWSISxxxxE and CWSIDxxxxE. The key message is:

**CWSIS0002E:** The messaging engine encountered an exception while starting. Exception: {0}

The exception information provides a reason why the messaging engine has failed to start.

Example 8-2 shows an example set of these types of messages.

*Example 8-2 SystemOut log with messaging engine error*

---

```
[...]CWSID0016I: Messaging engine ClusterHA.000-Bus1 is in state
Starting.
```

```
[...]CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: CWSIS1524E: Data
source, jdbc/HA, not found.
```

[...]CWSID0035E: Messaging engine ClusterHA.000-Bus1 cannot be started; detected error reported during com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()

[...]CWSID0027E: Messaging engine ClusterHA.000-Bus1 cannot be restarted because a serious error has been reported.

[...]CWSID0016I: Messaging engine ClusterHA.000-Bus1 is in state Stopped.

[...]CWSID0016I: Messaging engine ClusterHA.000-Bus1 is in state Joined.

---

**Note:** If the messaging engine did not start properly, see “Messaging engine fails to start” on page 124.

## Message delivery problems

If the messaging engines appear to have started normally but you are having issues with message delivery or with the location of the messaging engine, look for the following symptoms.

Common problems that can occur in an HA or WLM configuration are:

- ▶ The messaging engine is not starting on the preferred server. In a WLM configuration, multiple messaging engines are started on one cluster member while no messaging engines are started on others. If either of these conditions exist, see 8.4.1, “Core group policy is incorrect or no preferred server defined” on page 134.
- ▶ Response messages appear on a different partition than the request message (messages appear in an unexpected place). If this appears to be your problem, see 8.4.2, “A permanent reply queue is in use” on page 138.
- ▶ Orphan messages after a failover. The failover might have been successful or not. See 8.4.4, “Orphaned messages after failover” on page 139.

Common problems that are specific to a WLM configuration are:

- ▶ Messages are not consumed from a partitioned destination. See 8.4.6, “Orphan messages on a partitioned destination” on page 142.
- ▶ Messages do not arrive on a specified partition of a partitioned destination. See 8.4.7, “Cluster weights incorrectly specified” on page 143.

If none of these symptoms fits your problem, see Chapter 1, “Introduction to JMS application problem determination” on page 1 to see if any other problem types might apply, or see Chapter 18, “The next step” on page 261 to gather documentation and contact IBM support.

## Messaging engine fails to start

The failure of a messaging engine to start usually indicates a configuration problem.

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261:

- ▶ If you find any of the following exception information, see 8.3.1, “JDBC provider or data source configuration error” on page 125.

CWSIS1524E: Data source, *data\_source*, not found

CWSIS1501E: The data source has produced an unexpected exception:  
java.sql.SQLException: Database '*database\_name*' not found.com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: Database, *database\_name*, DB2 Authentication Error

CWSIS1501E: The data source has produced an unexpected exception:  
java.sql.SQLException: Failed to start database '*database*', see the next exception for details.

- ▶ If you find this exception, see 8.3.3, “The file store directories were created incorrectly” on page 132.

File IO error: Permission Denied

- ▶ The following messages can indicate a configuration problem with the table schema, or can indicate a problem obtaining a lock during failover.

[...]CWSIS1535E: The messaging engine's unique id does not match that found in the data store.

CWSIS1519E: Messaging engine TestCluster.001-bus1 cannot obtain the lock on its data store, which ensures it has exclusive access to the data.

If you see these messages and do not believe you have a failover situation, or if you are not sure, see 8.3.4, “Table schema names are not unique” on page 133.

If you see these messages and a messaging engine does not start after a failover, see 8.4.3, “Data store is locked” on page 138.

- ▶ If you find the following exception information, see 8.3.2, “Environment variables in the classpath are scoped incorrectly” on page 130.

com.ibm.ws.sib.msgstore.MessageStoreRuntimeException:  
ClassNotFoundException *class\_name* e.g a com.ibm.db2.jcc class

## 8.3 Clustering configuration problems

The following are some of the more common configuration errors that can prevent the messaging engine from starting.

### 8.3.1 JDBC provider or data source configuration error

Messages CWSIS1524E and CWSIS1501E usually indicate a problem that can be traced to a configuration error for the messaging data store JDBC provider or data source. Common mistakes are:

- ▶ The JDBC provider is configured at the wrong scope.
- ▶ The message store database does not exist.
- ▶ The authentication alias has not been specified correctly.
- ▶ The JNDI name is specified incorrectly.

The specific cause might be apparent from the exception messages. If so, you can go directly to one of these topics. If not, check each possibility.

#### The JDBC provider is configured at the wrong scope

The JDBC provider and data source defined for the messaging engine data store must be defined at the *Cell* or *Cluster* scope so they are visible to all cluster members. If defined at the node level, only the cluster members on that node will be able to use them.

This problem is indicated by the messages shown in Example 8-3.

*Example 8-3 Messaging engine exception: Data source not found*

---

```
[...] CWSIS0002E: The messaging engine encountered an exception while starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: CWSIS1524E: Data
source, DataSource, not found.
```

---

You can resolve this problem by defining the JDBC provider for the messaging engine data store at the correct scope.

In an HA configuration, there will be one messaging engine for the cluster. In a WLM configuration, there will be multiple messaging engines for the cluster, each with its own distinct data store; therefore, you need to check the configuration for each JDBC provider and data source.

Using the administrative console:

1. Select **Resources** → **JMS** → **Data sources**.
2. Select **All scopes**.

Identify the data sources for all messaging engines in the cluster and the scopes to which they are each defined.

In an HA configuration, there is one messaging engine that can start on any cluster member. Ensure the data source and JDBC provider are defined at the Cluster or Cell scope so that each member of the cluster has access to that same definition.

In a WLM configuration, each cluster member can have an active messaging engine. Ensure the data source for each messaging engine's message store is at a scope available to the cluster member and that it is unique.

3. If necessary, create a new JDBC provider and data source at the correct scope:
  - a. Select the correct scope.
  - b. Select **New**.
  - c. Create a new JDBC provider (if necessary) and data source similar to the original resources, but with a new name and JNDI name.
4. Update the messaging engine with the new data source JNDI name:
  - a. Select **Cluster** → *cluster\_name*.
  - b. Select **Messaging Engine** → *Messaging\_engine*.
  - c. Select **Message Store**.
  - d. Update the data source JNDI name to refer to the newly defined data source.
  - e. Save all changes and restart the cluster.

### **The message store database does not exist**

If the database does not exist or if the data source has been configured with the incorrect database name, access to the database fails and the messaging engine cannot start.

This problem shows up as a statement in the SystemOut log as in Example 8-4 on page 127 suggesting the database cannot be found.



*Example 8-4 Messaging engine exception: database not found*

---

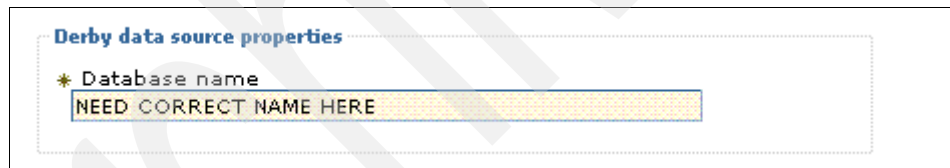
[...] **CWSIS0002E:** The messaging engine encountered an exception while starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException:  
**CWSIS1501E: The data source has produced an unexpected exception:  
java.sql.SQLException: Database 'SazFillin' not found.**

---

Check the data source properties to make sure that the database name in the configuration matches the database name.

Using the administrative console:

1. Select **Resources** → **JDBC** → **Data sources**.
2. Select **All scopes**.
3. Identify the data sources for each messaging engine in the cluster. In an HA configuration, there is one messaging engine that can start on any cluster member. In a WLM configuration, there are multiple messaging engines, up to one per cluster member.
4. For each data source:
  - a. Click the data source name to open the configuration page.
  - b. Verify the database name is set to the correct database (Figure 8-1).



*Figure 8-1 Database name setting in the data source*

5. Save all changes and restart the cluster.

### **The authentication alias has not been specified correctly**

This problem is specific to using a remote DB2 installation as the relational database for the messaging data store. A JAAS authentication alias associated with the data source is required for a connection to be made to the database.

If the JAAS authentication alias is not correct or is not set, access to the database fails. The symptom is a DB2 authentication error in the SystemOut log, similar to Example 8-5 on page 128.

[...] **CWSIS0002E:** The messaging engine encountered an exception while starting. Exception:  
**com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: Database, TestDB, DB2 Authentication Error**

---

The solution is to configure or correct the JAAS authentication alias.

Using the administrative console:

1. Select **Resources** → **JDBC** → **Data sources**.
2. Identify the data sources for each messaging engine in the cluster. In an HA configuration, there is one messaging engine that can start on any cluster member. In a WLM configuration, there are multiple messaging engines, up to one per cluster member.
3. For each data source:
  - a. Click the data source name to open the configuration page.
  - b. Click **JAAS-J2C authentication Data**.
  - c. Either select the JAAS authentication alias of interest and check the input data is correct, or define a new JAAS authentication alias by selecting **New**.
4. If a new JAAS authentication alias is defined, you need to associate it with the messaging engines for the cluster.
  - a. Select **Servers** → **Clusters** → *cluster\_name*.
  - b. Select **Messaging engines** → *messaging\_engine*.
  - c. Select **Message store**.
  - d. Use the drop-down to select the authentication alias, as shown in Figure 8-2 on page 129.

General Properties		Related Items
UUID	<input type="text" value="79D34C2152DE5A87"/>	
* Data source JNDI name	<input type="text" value="TestDataSource"/>	<a href="#">JAAS - J authent data</a>
Schema name	<input type="text" value="TestSchema"/>	
Authentication alias	<input type="text" value="(none)"/> <input type="text" value="(none)"/> <input type="text" value="BLADE202CellManager01/FBLAlias"/>	

Figure 8-2 Select the authentication alias

5. Save all changes and restart the cluster.

### The JNDI name is specified incorrectly

The JNDI name specified in the data source must match that specified in the messaging engine configuration. If it does not match, the exception shown in Example 8-6 will occur.

*Example 8-6 Messaging engine exception: Failed to start database*

---

```
[...]CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: CWSIS1524E: Data
source, DataSource, not found.

[...]CWSIS0002E: The messaging engine encountered an exception while
starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException:
CWSIS1501E: The data source has produced an unexpected exception:
java.sql.SQLException: Failed to start database
'c:/was61/profiles/AppSrv01/installedApps/BLADE202Cell101/DefaultApplica
tion.ear/DefaultDB', see the next exception for details.
```

---

To resolve this problem, check and correct (if necessary) the data source properties for each messaging engine.

Using the administrative console:

1. Select **Resources** → **JDBC** → **Data sources**
2. Set the scope to **All Scopes**.

You see a list of all the data sources defined at various scopes, similar to the list shown in Figure 8-3.

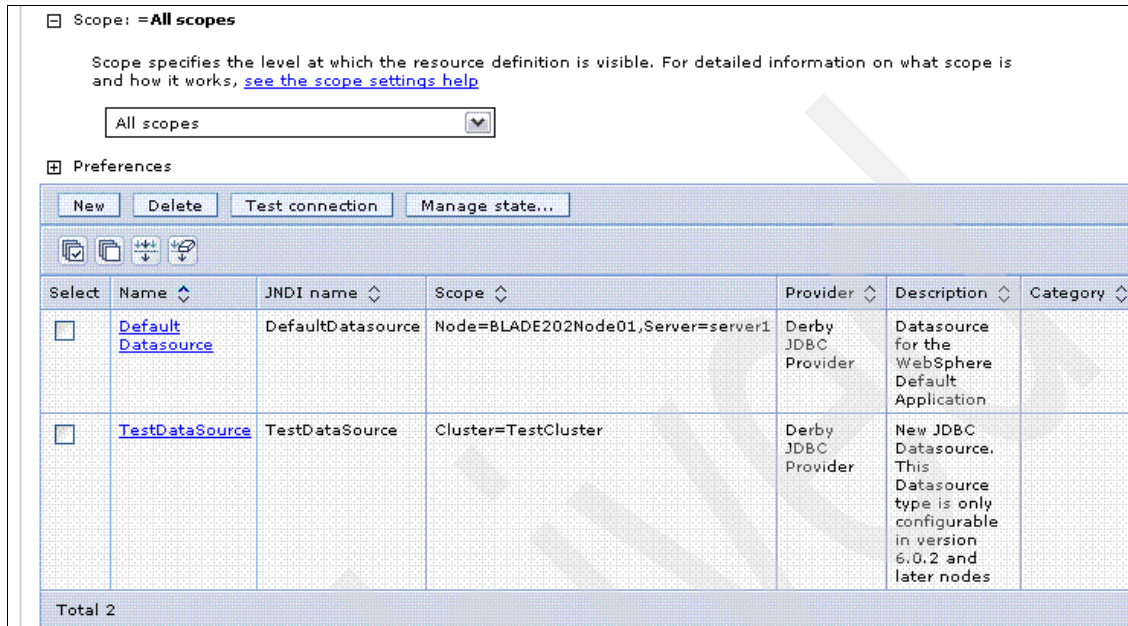


Figure 8-3 List of data sources

3. Make a note of the data source name and JNDI name for each cluster member messaging engine. If there is not a data source defined for each cluster member, you need to define them.
4. Next, use the information you collected to make sure the messaging engine has been configured with the correct JNDI name. Select **Servers** → **Clusters** → *cluster\_name*.
5. Select **Messaging engines** → *messaging\_engine*.
6. Select **Message store**.
7. Ensure the data source JNDI name is correct for that messaging engine.
8. Repeat this procedure for each cluster member that has its own database.
9. Save all changes, and restart the cluster.

### 8.3.2 Environment variables in the classpath are scoped incorrectly

Problems can occur when the JDBC provider for a messaging engine data store uses WebSphere variables in the classpath definition, and the variable does not exist or it is defined at the wrong scope.

This problem occurs when drivers for the JDBC provider are stored in different locations on the systems hosting the cluster members. If the variable is only defined at the Cell or Cluster scope, it will be incorrect for at least one of the cluster members.

This problem shows up as a statement in the SystemOut log as shown in Example 8-7.

*Example 8-7 Messaging engine exception: ClassNotFoundException*

---

```
[...] CWSIS0002E: The messaging engine encountered an exception while starting. Exception:  
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException:  
ClassNotFoundException <name of class> (for example, com.ibm.db2.jcc  
class)
```

---

To resolve this problem, define the WebSphere variable correctly on each cluster member.

For example, a configuration using a DB2 universal database as a data store needs the location of the DB2 JDBC drivers defined for each cluster member. Therefore, you need to define the DB2UNIVERSAL\_JDBC\_DRIVER\_PATH variable.

To define this variable using the administrative console:

1. Select **Environment** → **WebSphere Variables**.
2. Select the correct scope from the drop-down.

You see a list of variables as shown in Figure 8-4 on page 132.

☐ Scope: Cell=**BLADE202Cell01**, Node=**BLADE202Node01**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#)

Node=BLADE202Node01 ▼

☐ Preferences

New Delete

☑ ☐ ⚙️ ↻

Select	Name ↕	Value ↕
<input type="checkbox"/>	<a href="#">APP_INSTALL_ROOT</a>	\${USER_INSTALL_ROOT}/installedApps
<input type="checkbox"/>	<a href="#">CONNECTJDBC JDBC DRIVER PATH</a>	
<input type="checkbox"/>	<a href="#">CONNECTOR_INSTALL_ROOT</a>	\${USER_INSTALL_ROOT}/installedConnectors
<input type="checkbox"/>	<a href="#">DB2390 JDBC DRIVER PATH</a>	
<input type="checkbox"/>	<a href="#">DB2UNIVERSAL JDBC DRIVER NATIVEPATH</a>	
<input type="checkbox"/>	<a href="#">DB2UNIVERSAL JDBC DRIVER PATH</a>	

Figure 8-4 WebSphere environment variables for DB2

3. Verify and update the variables as needed to ensure they have the correct value. For example, DB2UNIVERSAL\_JDBC\_DRIVER\_PATH must have the value of the location of the DB2 JDBC drivers on the cluster member of interest.

If the variable does not exist at the scope, create it by clicking **New** and completing the Name and Value fields.

4. Save all changes and restart the cluster.

### 8.3.3 The file store directories were created incorrectly

If a file store is used for messaging engine persistence, problems can occur if the log and storage directories are created by the user rather than by the messaging engine. A messaging engine creates the directories it needs as it starts up.

This problem shows up as a statement in the SystemOut log with the following message text:

```
File IO error: Permission Denied
```

To solve this problem, remove the directories, which you can find in the WebSphere directory structure:

```
app_server_root/Profiles/cluster_profile/Filestores/com.ibm.ws.sib/
```

Restart the cluster and the messaging engines will create the directories it needs on start up.

### 8.3.4 Table schema names are not unique

If a single database is used to house data store tables for multiple messaging engine tables, each messaging engine must be assigned a unique schema name to avoid two messaging engines attempting to obtain a lock on the same table set.

This problem shows up in the SystemOut log as shown in Example 8-8.

*Example 8-8 Messaging engine exception: cannot obtain lock on data store*

---

```
[...]CWSIS1535E: The messaging engine's unique id does not match that found in the data store.
```

```
CWSIS1519E: Messaging engine TestCluster.001-bus1 cannot obtain the lock on its data store, which ensures it has exclusive access to the data.
```

---

To correct this problem, create unique schema names for each cluster member.

Using the administrative console:

1. Select **Service Integration** → **Buses**.
2. Click on the bus name.
3. Click **Messaging Engines**.
4. Click on the messaging engine name.
5. Click **Message Store**.
6. Make sure the Schema name box is filled with a unique schema name as shown in Figure 8-5 on page 134.


<b>General Properties</b>	
UUID	79D34C2152DE5A87
* Data source JNDI name	TestDataSource
Schema name	TestSchema
Authentication alias	(none) 

Figure 8-5 Messaging engine data store schema name

7. Repeat this procedure for each of the messaging engines, making sure all schema names are unique.
8. Save the configuration changes and restart the cluster.

## 8.4 Cluster runtime problems

This section discusses problems specific to clustering of messaging engines where the messaging engines start correctly but there is a problem with message delivery.

### 8.4.1 Core group policy is incorrect or no preferred server defined

To exert some control over the location of the messaging engine, you can define a preferred server in the core group policy. The messaging engine will start on the preferred server if the cluster member is active.

If a core group policy is defined incorrectly with invalid or non-matching criteria, the WLM manager randomly assigns the messaging engines to cluster members, resulting in the messaging engines being started on random cluster members.

#### Defining a preferred server

First, define a core group policy that can be associated with a messaging engine. The default messaging provider includes a DefaultCoreGroup that can be extended for cluster configurations.



To create a core group policy using the administrative console:

1. Select **Servers** → **Core Groups** → **Core group settings**.
2. Click on the core group name.
3. In the Configuration tab, select **Policies**.
4. Click **New** to define a new policy (Figure 8-6 on page 136):
  - a. Select **One of N** as the policy type.

The One of N policy keeps one member of the high availability group active at all times. It is used by groups that desire singleton failover. In an HA configuration, there is only one messaging engine.

- b. Enter a name for the policy.
- c. If you want failback to be active (that is, the messaging engine fails back when the preferred server comes online after a failover has taken place), select the failback box, as shown in Figure 8-6 on page 136.

**Core groups**

[Core groups](#) > [DefaultCoreGroup](#) > [Policies](#) > [New](#) > [New](#)

Use this page to define a One of N policy. This type of policy keeps one group member active at

Configuration

**General Properties**

\* Name  
Cluster ME Policy

\* Policy type  
One of N policy

Description

\* Is alive timer  
0 seconds

Quorum

Failback

The additional properties will not be available u  
properties for this item are applied or saved.

**Additional Properties**

- Custom properties
- Match criteria
- Preferred servers

Figure 8-6 Core group policy for high availability and failback

Next, you define the match criteria that associates a messaging engine with the policy.

Using the administrative console:

1. Select **Servers** → **Core groups** → **Core group settings**.
2. Click on the core group.
3. In the configuration tab, select **Policies**.
4. Click on the policy name.
5. Select **Match Criteria**.
6. Click **New**.

- Enter the name/value pair information. The Name and Value fields together specify a messaging engine or group of messaging engines, it will be linked to this policy. Name/value pairs that you can use shown in Table 8-1.

Table 8-1 Match criteria name/value pairs

Name	Value	Messaging engines the policy will match
type	WSAF_SIB	Any messaging engine
WSAF_SIB_MESSAGING_ENGINE	The name of your messaging engine.	A particular messaging engine
WSAF_SIB_BUS	The name of your bus	All messaging engines in a particular bus
IBM_hc	The name of your cluster	All messaging engines in a particular cluster

In this instance, you need to use the WSAF\_SIB\_MESSAGING\_ENGINE name/value pair, as shown in Figure 8-7.

Core groups

[Core groups](#) > [DefaultCoreGroup](#) > [Policies](#) > [Default SIBus Policy](#) > [Match criteria](#) > [New](#)

Use this page to specify a name-value pair that is used as part of the match criteria for a policy. It must match an attribute that is part of the name of the high availability group associated with the

Configuration

**General Properties**

\* Name  
WSAF\_SIB\_MESSAGING\_ENGINE

\* Value  
TestCluster.001-bus1

Description  
Match Criteria for Messaging engine 001 in cluster

Figure 8-7 Name / value pair

Finally, specify the preferred server for the policy:

- Select **Servers** → **Core groups** → **Core group settings**.

2. Click on the core group name.
3. In the configuration tab select **Policies**.
4. Click on the policy name.
5. Click **Preferred servers**.
6. Select the cluster member on which the messaging engine should start , and use **Add** to move the cluster member to the preferred server list (on the right).  
  
Multiple servers can be in the list but the order is important. There is an implicit stronger preference for a server that appears earlier in the list.
7. Save the changes and restart the cluster.

## 8.4.2 A permanent reply queue is in use

New in WebSphere V6.1 is the ability to create and assign a permanent reply queue for JMS response messages. This functionality, while supported in a single server environment, is not supported in a clustered environment because the response message currently cannot be guaranteed to use the same partition of the destination as the request message.

## 8.4.3 Data store is locked

When a messaging engine tries to fail over and cannot complete the process, a problem can occur with the data store lock. If the messaging engine is taking over from another that was running on a different cluster member, the database might not have released the database locks that comprise the datastore lock.

This problem shows up as a statement in the SystemOut log suggesting that a lock cannot be obtained on the data store as in Example 8-9.

*Example 8-9 Failover fails due to data store lock*

---

[...] **CWSIS1535E**: The messaging engine's unique id does not match that found in the data store.

**CWSIS1519E**: Messaging engine TestCluster.001-bus1 cannot obtain the lock on its data store, which ensures it has exclusive access to the data.

---

To resolve this problem, use the administration tools for the chosen relational database management system to examine the locks on the SIBOWNER table. If the database is still holding locks after the failure of a cluster member, release

the locks and attempt to restart the messaging engine on the appropriate cluster member, using the administrative console.

If a foreign bus is defined it is also affected by the failover process. After the messaging engine is restarted, the foreign bus should restart. For more information about foreign bus problem determination, see Chapter 15, “Foreign bus problem determination” on page 209.

#### 8.4.4 Orphaned messages after failover

An orphaned message is one that exists on a destination but no application consumes it. You would normally determine that you have orphaned messages by monitoring the number of messages sent as opposed to the number of (expected) messages received. You might also notice from the administrative console that the message depth of a queue is not zero when you would expect it to be.

Reasons that messages can be orphaned after a failover include:

- ▶ If a messaging engine fails over from one active cluster member to another active cluster member, messages keep flowing; however, if failover cannot take place, persistent messages can become orphaned.

To receive the orphaned messages, the messaging engine needs to be started on one of the cluster members. At this point, a client can connect and consume the messages. This state is applicable to all destination points.

To start a messaging engine, see 1.2.3, “Determine your messaging engine status” on page 8.

- ▶ If a foreign bus is defined, there might be orphaned messages due to one end of the link not failing over. For more information about foreign bus problem determination, see Chapter 15, “Foreign bus problem determination” on page 209.
- ▶ Messages can also be orphaned after a successful failover if the client is not configured correctly. If cluster members are spread across multiple machines in an HA configuration, any clients need to be able to connect to all of the cluster members.

To solve this problem, create an endpoint list.

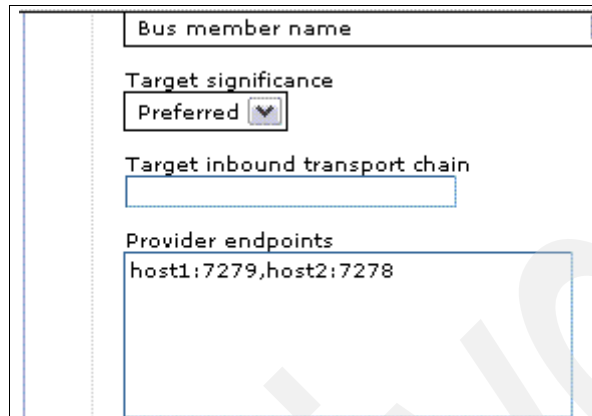
##### **Creating an endpoint list**

To create an endpoint list using the administrative console:

1. Select **Resources** → **JMS**.
2. Select **Connection Factories**.

3. Select the **connection\_factory\_name** that the client is trying to use to connect to the cluster.

You can enter a comma separated list of provider endpoints into the provider endpoints box as shown in Figure 8-8.



The screenshot shows a configuration window with the following fields:

- Bus member name: [ ]
- Target significance: Preferred [v]
- Target inbound transport chain: [ ]
- Provider endpoints: host1:7279,host2:7278

Figure 8-8 Provider endpoints

The list takes the form *host:port:chain,host:port:chain*. Enter all of the cluster members that are in a highly available group:

- *host* is the box where a particular cluster member is located.
  - *port* is the SIB\_ENDPOINT\_ADDRESS or SIB\_ENDPOINT\_SECURE\_ADDRESS for the cluster member (see **Servers** → **server\_name** → **Ports**).
  - *chain* is the target transport chain and is optional in this list.
4. Save your changes and restart cluster.

### 8.4.5 Core group policy is not correct or no preferred server defined

If a core group policy is incorrectly defined with invalid or non matching criteria, the WLM manager randomly assigns the messaging engines to cluster members. In a WLM configuration, this can result in multiple messaging engines being started on a single cluster member, while leaving other cluster members with no messaging engine.

The solution is to pin the messaging engines to cluster members using core group policies. The messaging engines will then start on the defined cluster members if the cluster members are active.

Some WLM clusters have highly available messaging engines in the cluster. The HA messaging engines have a preferred server defined instead of being pinned to a cluster member; other messaging engines that are not HA are pinned.

To define a preferred server, see 8.4.1, “Core group policy is incorrect or no preferred server defined” on page 134. The process to pin a messaging engine to a cluster member in a WLM configuration is similar; however, make sure to select the **Use preferred servers only** box when you define a One of N policy, as shown in Figure 8-9 on page 142.

This definition pins the messaging engine to the preferred server so that it cannot fail over. If the cluster member goes offline, the messaging engine is not available until the cluster member comes back online. You can also use a static policy to pin a messaging engine.

The WLM cluster has multiple messaging engines, so you need to perform the process described above for each messaging engine.

**Core groups**

[Core groups](#) > [DefaultCoreGroup](#) > [Policies](#) > **HA ME Policy**

Use this page to define a One of N policy. This type of policy keeps one group member active at

Configuration

---

General Properties	Additional Properties
* Name <input type="text" value="HA ME Policy"/>	<input type="checkbox"/> <a href="#">Custom property</a>
* Policy type <input type="text" value="One of N policy"/>	<input type="checkbox"/> <a href="#">Match criteria</a>
Description <input type="text"/>	<input type="checkbox"/> <a href="#">Preferred servers</a>
* Is alive timer <input type="text" value="0"/> seconds	
<input type="checkbox"/> Quorum	
<input type="checkbox"/> Failback	
<input checked="" type="checkbox"/> Preferred servers only	

Figure 8-9 Select Preferred servers only

## 8.4.6 Orphan messages on a partitioned destination

In a WLM configuration, with multiple messaging engines in a cluster, a queue destination becomes partitioned. The number of partitions depends upon the number of messaging engines because every messaging engine in the cluster has a queue point. If the destination is a topic space, it has a publication point localized to every messaging engine in the cluster. For more information about destination points, see 1.2.4, “Finding queued messages” on page 9.

Partitions cannot communicate with other partitions. A client must be connected to each cluster member to collect all messages from all partitions.



If an active cluster member goes offline, persistent messages being routed through the cluster member can remain on the destination point until the cluster member and messaging engine becomes active again. After the cluster member becomes active again, the client can reconnect and consume the messages.

### 8.4.7 Cluster weights incorrectly specified

If the weights are incorrectly specified in a WLM cluster configuration some messages might not arrive on a specified partition of a partitioned destination. If a weight of 0 is specified for a particular cluster member, messages will not flow to the destination point on this cluster member. This configuration variable is not an attribute of messaging; it is an attribute of the server.

For information about specifying cluster weights, see:

- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *Clusters and workload management* at:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun\\_srvgrp.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_srvgrp.html)

Archived

## Message-driven beans problem determination

This chapter discusses common problems you might encounter when using a message-driven bean (MDB) configured with WebSphere Application Server default messaging.

If you are using MDBs to consume messages from a WebSphere MQ destination, see Chapter 10, “WebSphere MQ and MDBs” on page 157.

## 9.1 Introduction to message-driven beans

Message-driven beans (MDBs) are EJBs that act as message consumers. An MDB always runs inside an application server, and it connects to a messaging engine to consume messages. MDB applications are invoked by WebSphere when messages arrive on the queue the MDB has been configured to consume from.

When an MDB application starts:

1. The activation specification is read and validated. If the activation specification fails validation the MDB will not start.
2. The MDB attempts to connect to a messaging engine in the bus that has been specified on the activation specification. If there are no messaging engines available then one of the following occurs:
  - If the MDB is running on a server that is a bus member which the activation specification references, then it will wait for the messaging engine on that same server to start and connect to it.
  - If the MDB is running on a server that is not a bus member of the specified bus, then it will retry every 30 seconds to connect to a messaging engine that might be running on a remote server.
3. The MDB creates an asynchronous consumer on the configured destination.
4. The consumer informs the MDB application whenever a message is available for consumption by calling its `onMessage` method.

### For more information

For more information about activation specification settings, see:

- ▶ *JMS activation specification settings* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/sibjmsresources/SIBJMSActivationSpec\\_DetailForm.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/sibjmsresources/SIBJMSActivationSpec_DetailForm.html)

- ▶ If your MDB is attempting to consume messages from a remote WebSphere Application Server Cell, some additional configuration steps are necessary.

See *Configuring the core group bridge service* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun\\_ha\\_coregroupbridge.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_ha_coregroupbridge.html)

- ▶ *Message ordering considerations*

You can configure MDBs to process multiple messages in parallel or to have batches of messages sent for performance benefits. However, if message

ordering is important then you must have messages processed one at a time, by a single MDB instance. To achieve this behavior, you need to carefully select the activation specification properties.

See *JMS activation specification settings* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/sibjmsresources/SIBJMSActivationSpec\\_DetailForm.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/sibjmsresources/SIBJMSActivationSpec_DetailForm.html)

- ▶ MDBs and cluster considerations

For more information, see Section 9.5.2, *MDBs and clusters* in *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304

### 9.1.1 Symptoms of a messaging-driven bean problem

Symptoms addressed in this chapter include:

- ▶ A message-driven bean application fails to start.
- ▶ A message-driven bean fails to connect to a messaging engine.
- ▶ A message-driven bean fails to consume messages.
- ▶ A message-driven bean fails during processing.
- ▶ Messages are lost.

## 9.2 Verify integrity

If you are having problems related to a messaging-driven bean application, do the following steps before proceeding with problem determination:

- ▶ Verify the MDB application is started.
- ▶ Verify that messages can be sent to the target destination.
- ▶ Verify the MDB is consuming messages.

### 9.2.1 Verify the MDB application is started

To diagnose the problem, first determine the state of the deployed MDB application. The simplest method to check the state is using the administrative console:

1. Select **Applications** → **Enterprise Applications**.

You see a list of deployed applications along with their status. If the MDB application has not started, a red cross displays in the corresponding status column. If the application is started, then a green arrow displays.

2. If the application is not started, select the box to the left of the application and click **Start**.

If the MDB application fails to start, see 9.3, “Analyze the SystemOut log for the MDB server” on page 149.

## 9.2.2 Verify that messages can be sent to the target destination

Verify that messages can be produced for the target destination. For example, create a test application to produce messages that are sent to the destination.

## 9.2.3 Verify the MDB is consuming messages

If the MDB fails to handle a message, the messaging engine attempts to redeliver the message based on the destination properties of the destination. To display these properties, use the administrative console to:

1. Select **Service Integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations** to display a list of queue and topic destinations.
4. Select the appropriate queue destination from the displayed list.

If the message cannot be processed, delivery is reattempted until the maximum threshold for failed deliveries is met. At this point, the messaging engine attempts to deliver the message to the exception destination. By default, this address uses a destination that is generic for that messaging engine. The naming format of the default exception destination is:

```
SYSTEM.Exception.Destination.Messaging_Engine_Name
```

Depending on your configuration, a unique exception destination might have been defined for the queue in question.

By default, no application server log messages are produced when an MDB processes a received message. With this in mind, it is advisable that you add some form of logging to an application so that you can recognize that messages are being processed successfully.

## 9.3 Analyze the SystemOut log for the MDB server

Examine the application server log files for any exceptions. The most common cause of problems is a configuration problem with the activation specification the MDB is configured to use.

If the SystemOut log includes:

- ▶ J2CA0164E, see 9.4.1, “Activation specification destination name incorrect” on page 150.
- ▶ J2CA0052E, see 9.4.2, “Unable to locate the activation specification” on page 151.
- ▶ J2CA0137E, collect any error messages following (CWSJRxxxxE) and see 9.4.3, “Invalid activation specification” on page 151.
- ▶ The following messages, see 9.4.4, “Listener port used for default messaging resources” on page 152:

```
Unable to start MDB Listener QueueReceiver, JMSDestination
Destination_JNDI_Name :
com.ibm.ejs.jms.listener.MDBInvalidConfigException: Cannot deploy an
MDB against a listener port specifying a default messaging JMS
resource.
```

- ▶ CWSIV0775W, see 9.4.5, “MDB unable to connect to a messaging engine” on page 153:
- ▶ CNTR0020E, this message provides the primary source of problem determination information for MDB processing problems.

See 9.4.6, “MDB does not handle exception appropriately” on page 153.

If the MDB is not consuming messages but you find no exception information in the SystemOut log, see 9.4.7, “MDB started but inactive: No errors in the logs” on page 154.

## 9.4 Causes and solutions

Generally, MDB problem determination can be categorized into the following areas:

- ▶ A message-driven bean application fails to start.

This failure is generally caused by a configuration problem. For possible causes, see:

- 9.4.1, “Activation specification destination name incorrect” on page 150.

- 9.4.2, “Unable to locate the activation specification” on page 151.
- 9.4.3, “Invalid activation specification” on page 151.
- ▶ A message-driven bean fails to consume messages. See:
  - 9.4.4, “Listener port used for default messaging resources” on page 152.
  - “Activation specification used for WebSphere MQ resources” on page 151.
  - 9.4.7, “MDB started but inactive: No errors in the logs” on page 154.
- ▶ A message-driven bean fails to connect to a messaging engine.
 

See 9.4.5, “MDB unable to connect to a messaging engine” on page 153.
- ▶ A message-driven bean fails during processing.
 

See 9.4.6, “MDB does not handle exception appropriately” on page 153.

### 9.4.1 Activation specification destination name incorrect

The symptom of this problem is this message in Example 9-1.

*Example 9-1 Activation specification error J2CA0164E*

---

```

ActivationSpe E   J2CA0164E: The lookup of the Destination with JNDI
Name JNDI_Name failed. The Destination is required by the Activation
Specification. The lookup failed due to exception
javax.naming.NameNotFoundException: Context: Scope_Path, name: JNDI
Name: First component in name JNDI_Name not found. [Root exception is
org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0]
  
```

---

Check that the destination name is correct and that the activation specification and destination JMS resources have been bound into JNDI.

To find JMS resources, look in **Resources** → **JMS** → *destination\_type*. Select the destination to view the JNDI name, which must match the destination JNDI name specified in the activation specification for the destination. The destination might also be specified at deploy time or by the application developer in the configuration file for MDB (inside the EAR file).

To find activation specifications, look in **Resources** → **JMS** → **Activation specifications**.



## 9.4.2 Unable to locate the activation specification

The symptom of this problem is this message in Example 9-2.

*Example 9-2 Activation specification error J2CA0052E*

---

```
ActivationSpe E   J2CA0052E: The lookup of the Activation Specification
with JNDI Name JNDI_Name failed due to the following exception:
javax.naming.NameNotFoundException: Context: Scope_Path, name:
JNDI_Name : First component in name JNDI_Name not found. [Root
exception is org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0]
```

---

Check that the activation specification that the MDB has been configured to use exists. Either the MDB was configured to use an incorrect activation specification or the activation specification has not been created.

## 9.4.3 Invalid activation specification

Message J2CA0137E indicates an invalid property has been specified for the activation specification. This message contains nested messages that provide more information about the failure. See Example 9-3.

*Example 9-3 Invalid activation specification property*

---

```
ActivationSpe E   J2CA0137E: The ActivationSpec validate() method
failed with an InvalidPropertyException. The ActivationSpec is
ActivationSpecification_JNDI_Name, which belongs to the installed
ResourceAdapter RA_Name and is associated with the MDB Application
MDB_Application_Name. See the following list of failed properties
along with their values: Property_Name. The exception
is:javax.resource.spi.InvalidPropertyException: CWSJR1181E: The JMS
activation specification has invalid values - the reason(s) for failing
to validate the JMS activation specification are: Nested_Exception(s)
```

---

This message occurs when a problem with the configuration has been detected. See the nested exception for more information, and update the activation specification property information.

### Activation specification used for WebSphere MQ resources

One specific example of this type of error can occur when an MDB is configured to consume from a WebSphere MQ destination using an activation specification.

Example 9-4 on page 152 illustrates the error that can occur.

*Example 9-4 Invalid use of activation specification*

---

ActivationSpec E **J2CA0137E:** The ActivationSpec validate() method failed with an InvalidPropertyException. The ActivationSpec is Activation\_Specification\_JNDI\_Name, which belongs to the installed ResourceAdapter RA\_Name and is associated with the MDB Application MDB\_Application\_Name. See the following list of failed properties along with their values: destination Destination\_Name. The exception is: javax.resource.spi.InvalidPropertyException: **CWSJR1181E:** The JMS activation specification has invalid values - the reason(s) for failing to validate the JMS activation specification are: [**CWSJR1192E: JMS activation specs using a destination type of queue must have a destination of type [com.ibm.websphere.sib.api.jms.JmsQueue] but the destination passed was of type [com.ibm.mq.jms.MQQueue]**]

---

Ensure that the MDB is configured with an activation specification and is pointing to a destination of the correct type. In this example, the destination reference is wrong. An activation specification is being used so the destination should refer to a service integration bus destination.

#### **9.4.4 Listener port used for default messaging resources**

A listener port has been configured for use with the default messaging resources. This problem is characterized by these messages present in the application server log files in Example 9-5.

*Example 9-5 Unable to start MDB listener*

---

**Unable to start MDB Listener** QueueReceiver, JMSDestination  
*Destination\_JNDI\_Name :*  
com.ibm.ejs.jms.listener.MDBInvalidConfigException: Cannot deploy an MDB against a listener port specifying a default messaging JMS resource.

---

MDBs that use default messaging must be configured to use an activation specification and not a listener port. Likewise, MDBs that are configured to consume messages from a WebSphere MQ messaging destination must be configured to use a listener port and not an activation specification.

## 9.4.5 MDB unable to connect to a messaging engine

The MDB is unable to create a connection to a messaging engine in the specified bus as in Example 9-6.

*Example 9-6 MDB unable to connect to a messaging engine*

---

```
SibMessage W [:] CWSIV0775W: The creation of a connection for
destination Destination_Name on bus Bus_Name for endpoint activation
Endpoint_Activation_Details failed with exception
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0086E: Bus
Bus_Name not found
```

---

If this problem occurs:

- ▶ Ensure the bus name in the activation specification is correct and is pointing to the correct bus for your configuration. You can find activation specifications under **Resources** → **JMS** → **Activation specifications**.
- ▶ Ensure the servers hosting the messaging engines in the specified bus are started.
- ▶ Ensure that the target messaging engines on the remote servers are started.

You might find additional diagnostic information in nested exceptions.

## 9.4.6 MDB does not handle exception appropriately

When an MDB is given a message to consume, the MDB takes on the responsibility of providing logging until the MDB's `onMessage` method completes. If an exception is thrown by the MDB, the application server catches the exception and traces it to the log files.

The message shown in Example 9-7 indicates an exception occurred in the MDB but was not handled.

*Example 9-7 Exception occurs in an MDB*

---

```
[...] 00000066 ExceptionUtil E CNTR0020E: EJB threw an unexpected
(non-declared) exception during invocation of method "onMessage" on
bean
"BeanId(PackageReceivedEAR#PackageReceived.jar#PackageReceived, null)".
Exception data: java.lang.RuntimeException: MDB error at
receiver.PackageReceivedBean.onMessage(Unknown Source)
```

```
at  
com.ibm.ejs.container.MessageEndpointHandler.invokeMdbMethod(MessageEnd  
pointHandler.java:990)
```

---

It is the responsibility of the MDB application to handle exceptions.

The solution in this case is to modify the MDB application to handle exception cases.

### **Message not restored to queue after MDB failure**

The MDB transaction-type set in the EJB module deployment descriptor can be container-managed or bean-managed. A bean-managed transaction is responsible for committing or rolling back the transaction. It does this with the `UserTransaction` interface.

If the MDB is bean managed and it does not correctly cope with transactions and exceptions, messages be “lost.”

For more information, see Chapter 8 of *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

## **9.4.7 MDB started but inactive: No errors in the logs**

If a messaging engine for the desired bus is defined on the local server, then the MDB only attempts to connect to the local messaging engine. If the local messaging engine is not currently running, the MDB will silently wait for it to start.

If an MDB is deployed to a cluster, the MDB application will start on all servers within the cluster, but only the MDB instance on the server where the messaging engine is active will consume messages. During failover, you might experience a period of time in which the MDB is waiting for a local messaging engine to start. Several defects in this area were fixed in fix pack 6.1.0.9 (and 6.0.2.23) for WebSphere Application Server.

## 9.5 Validate the solution

If you have made changes to the WebSphere Application Server JMS or JNDI resources, you must restart the application server processes for changes to take effect.

Verify the MDB application is started and that the MDB application is processing messages correctly.

Archived

Archived



## WebSphere MQ and MDBs

This topic covers some of the problems that you might encounter with a message-driven bean configured to consume messages from a WebSphere MQ destination using a listener port.

## 10.1 Symptoms of problems with MDBs and listener ports

Symptoms of a problem with MDBs that consume messages from a WebSphere MQ destination using a listener port include:

- ▶ The MDB application does not start.
- ▶ The MDB fails to connect to the WebSphere MQ queue.
- ▶ The MDB application starts, but is not driven by messages on the WebSphere MQ destination.
- ▶ The SystemOut log contains `com.ibm.mqservices.MQInternalException` messages.
- ▶ The SystemOut log contains `javax.naming.NameNotFoundException` for the destination.
- ▶ The SystemOut log contains `javax.jms.InvalidDestinationException` for the destination.
- ▶ The SystemOut log contains `com.ibm.ejs.jms.listener.MDBInvalidConfigException` for the destination.

## 10.2 Verify system integrity

To ensure the environment is properly set up, work with the WebSphere MQ administrator:

- ▶ Ensure the target WebSphere MQ queue manager is available.
- ▶ Ensure messages can be produced for the target destination; for example, use WebSphere MQ Explorer.

## 10.3 Analyze the application server log

Examine the application server log for the MDB application; look for error messages prefixed with `WMSG`. The nested exception contains an error message indicating the reason for the failure to start the listener port.



If the root cause is a WebSphere MQ problem, a WebSphere MQ reason code is included in the linked exception information. For example:

```
:Caused by: com.ibm.mqservices.MQInternalException: MQJE001: An
MQException occurred: Completion Code 2, Reason 2059
MQJE011: Socket connection attempt refused
```

**Tip:** The `mqrc reason_code` command on a WebSphere MQ system returns a brief description of the given reason code.

These exceptions might occur:

- ▶ JMSDestination *jms/theQ*: javax.naming.NameNotFoundException  
The JMS destination provided to the listener port was not found in JNDI. Ensure that the destination JNDI name is specified correctly.
- ▶ JMSDestination *jms/theQ*: javax.jms.InvalidDestinationException  
A JMS queue that is not of type WebSphere MQ messaging provider has been specified on the listener port.  
Ensure that the destination JNDI name provided to the listener port resolves to a WebSphere MQ destination and that duplicate objects have not been defined.
- ▶ JMSDestination *mqq*:  
com.ibm.ejs.jms.listener.MDBInvalidConfigException  
The connection factory JNDI provided to the listener port is pointing to a connection factory that is not of type WebSphere MQ. Ensure that the connection factory JNDI name provided resolves to a WebSphere MQ connection factory and that duplicate objects have not been defined.
- ▶ An MQ reason code is attached  
This message implies that the listener port has at least made a connection attempt to WebSphere MQ. See Chapter 14, “JMS application with WebSphere MQ problem determination” on page 203.

## 10.4 Validate the solution

If you made changes to the WebSphere Application Server JMS or JNDI resources, you must restart the application server processes for changes to take effect. If changes are made to WebSphere MQ queue manager objects only, an application server restart is not normally required.

Validate the solution by checking the application server logs to ensure that the error messages no longer occur.

The presence of message WMSG0042I indicates that the listener port in question has started successfully.



## WebSphere MQ server problem determination

WebSphere MQ server functionality was added in Version 6.1 of WebSphere Application Server. It provides another means of interoperating between WebSphere MQ and the default messaging provider in the application server.

This chapter discusses problems that can occur that are specific to the use of WebSphere MQ server.

## 11.1 Introduction to WebSphere MQ server configurations

WebSphere MQ server provides a means for defining the connection attributes for a client-based connection to WebSphere MQ V6 for z/OS queue managers and queue sharing groups so that a service integration bus might interconnect with them.

You can find more information about the WebSphere MQ server can be found in the following WebSphere Application Server version 6.1 Information Center topic, *Working with WebSphere MQ server* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjfp0014\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjfp0014_.html)

## 11.2 Verify system integrity

The first thing to check if you experience problems with a WebSphere MQ server configuration is to ensure that the WebSphere MQ server is enabled. If not, messages can appear to be “lost”.

In the administrative console:

1. Select **Servers** → **WebSphere MQ servers**.
2. Select the server definition you wish to test.
3. Click **Test connection**.

If the connection is not enabled, see 11.4, “WebSphere MQ server not enabled” on page 164.

Even if the test indicates the connection is enabled, this test only checks that the deployment manager process has connectivity. It is possible that the node or server might not be configured correctly (MQ\_INSTALL\_ROOT problem) at run time, or that there is a firewall or network problem.

## 11.3 Symptoms of problems

Symptom of problems that can occur while using WebSphere MQ server include:

- ▶ Messages are not processed by WebSphere MQ server. Messages might appear to be lost.

There are no queue points for a WebSphere MQ server connection. The message transport is synchronous. The messages either go or an error will be thrown.

If the message is sent fire-and-forget, the message could end up on the exception destination for the messaging engine where the connection was made.

Specific messages that indicate this type of problem include WMSG1605E, CWSJP9999E, and WMSG1603E.

If any of these symptoms describe your problem, see 11.4, “WebSphere MQ server not enabled” on page 164.

- ▶ Unable to produce messages to a WebSphere MQ server destination.

The application receives an exception while attempting to create a producer for a given destination or while attempting to send a message.

Examples of messages that can accompany the exception for this type of problem include CWSIP0811W, CWSIC8002E, CWSJP0023E.

If this is the case, see 11.5, “Unable to produce messages to a WebSphere MQ server-hosted destination” on page 166.

- ▶ Unable to consume messages from a WebSphere MQ destination.

The application receives an exception while attempting to create a consumer for a given destination or while attempting to send a message.

Examples of messages that can accompany the exception for this type of problem include CWSIP0815W and CWSJP0002E.

If this is the case, see 11.6, “Unable to consume messages from a WebSphere MQ destination” on page 168.

- ▶ JMS messages arrive but appear corrupt.

Messages arriving on WebSphere MQ destinations do not appear to have maintained standard/user specific JMS headers.

Messages cannot be selected based on JMSXUserID or JMSCorrelationID because they have changed or contain additional spaces.

If this is the case, see 11.7, “JMS messages arrive but appear corrupt” on page 170.

- ▶ The application is unable to connect.

An application using a destination provided by the WebSphere MQ server connects only to the service integration bus. The application must use the correct type of connection factory. Using the wrong type of connection will produce a `JMSException`.

If you are experiencing connectivity problems:

- a. Ensure that you are using a connection factory for the default messaging provider. Using the administrative console:
  - i. Select **Resources** → **JMS** → **connection factory type**
  - ii. Validate the provider column for your connection factory  
Check for duplicates such as JMS resources defined with the same name for WebSphere MQ and default messaging providers.
- b. Ensure the WebSphere MQ server is enabled.

If neither of these is the problem, you are probably experiencing a generic JMS connectivity issue. See Chapter 6, “JMS application problem determination” on page 81.

## 11.4 WebSphere MQ server not enabled

An application server hosting a messaging engine that is used to connect to the WebSphere MQ server must have access to the required minimum level of the WebSphere MQ client.

A reference to the WebSphere MQ client code is provided by the WebSphere Application Server environment variable `MQ_INSTALL_ROOT`, and is defined at the node scope. By default, this variable points to the client that is provided with the application server, and in most cases this client is sufficient. However, you might need to modify this if your configuration is to run in bindings mode.

If the `MQ_INSTALL_ROOT` variable references the wrong location or the referenced WebSphere MQ client is not a supported level, then the WebSphere MQ server functionality is disabled.

### 11.4.1 Analyze SystemOut for the messaging engine

Examine the application server log files for error messages prefixed with `CWSJP` and `WMSG`.

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261:

- ▶ If you see this message, see 11.4.2, “MQ JMS client not detected” on page 165.  
**WMSG1605E:** It was not possible to detect the MQ JMS client at the specified path {0}.
- ▶ If you see this message, see 11.4.3, “WebSphere MQ client version not supported” on page 165.  
**CWSJP9999E:** WebSphere MQ Server functionality cannot operate with the level of WebSphere MQ client found MQ\_Client\_Level\_Found WebSphere MQ Server functionality has been disabled.
- ▶ If you see this message, see 11.4.4, “Previous version registered” on page 166.  
**WMSG1603E:** An internal error occurred. It was not possible to register the WebSphere MQ JMS client with the application server due to exception {0}.

## 11.4.2 MQ JMS client not detected

The symptom of this problem is the following message:

**WMSG1605E:** It was not possible to detect the MQ JMS client at the specified path {0}.

Ensure that the WebSphere Application Server MQ\_INSTALL\_ROOT variable is set correctly and that the WebSphere Application Server process has the correct access permissions.

## 11.4.3 WebSphere MQ client version not supported

The symptom of this problem is this message:

**CWSJP9999E:** The version of WebSphere MQ Client referenced by the WebSphere Application Server MQ\_INSTALL\_ROOT variable does not meet the minimum supported level of WebSphere MQ.

Check the system requirements for WebSphere Application Server 6.1 to ensure that your combination of WebSphere Application Server and WebSphere MQ versions is supported for your environment.

See *System Requirements for WebSphere Application Server V6.1* at:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27007651>

If it is not possible to apply the specified maintenance then you should restore the MQ\_INSTALL\_ROOT variable to its original setting of `${WAS_INSTALL_ROOT}/lib/WMQ`.

#### 11.4.4 Previous version registered

The symptom of this problem is this message:

**WMSG1603E:** An internal error occurred. It was not possible to register the WebSphere MQ JMS client with the application server due to exception {0}.

A problem has occurred when loading the client because a previous version was already registered. This generally occurs when changing between versions of the client.

To resolve this problem:

1. Ensure no processes are running for the given WebSphere Application Server profile.
2. Run the `osgiCfgInit` command from the bin directory of the given WebSphere Application Server profile.
3. Restart the WebSphere Application Server nodes and servers for the given profile.

#### 11.4.5 Validate the solution

If you changed the WebSphere MQ\_INSTALL\_ROOT variable or modified access permissions, restart each application server that hosts a messaging engine that interacts with the WebSphere MQ server.

After the restart, check the application server logs to ensure no error messages prefixed CWSJP or WMSG are reported.

### 11.5 Unable to produce messages to a WebSphere MQ server-hosted destination

This topic examines common problems encountered when attempting to produce messages to a destination hosted on a WebSphere MQ server.



## 11.5.1 Collect diagnostics

In addition to gathering information from the application log and the SystemOut messaging engine (see 1.3, “Collect diagnostics” on page 19), you need to collect the information from the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.

## 11.5.2 Analyze the application log

First, examine the exception in the application log; specifically, look at the final **Caused by:** section. The error will look similar to Example 11-1.

*Example 11-1 Errors indicating problem producing messages*

---

**Caused by:** com.ibm.wsspi.sib.core.exception.SISessionDroppedException:  
**CWSIP0811W:** A message could not be sent to destination MQQ assigned to WebSphere MQ Server bus member QIU0-ghbus. The WebSphere MQ completion code was 2. The WebSphere MQ reason code was 2053.

at  
com.ibm.ws.sib.comms.common.JFAPCommunicator.parseSingleException(JFAPCommunicator.java:1874)  
... 55 more

**Caused by:** com.ibm.websphere.sib.exception.SIErrorException:  
**CWSIC8002E:** An internal error occurred. An unknown or unexpected exception was thrown by the core A

PI: exception  
com.ibm.ws.sib.remote.mq.exceptions.CorruptMQSessionException:  
**CWSJP0023E:** A Websphere MQ error was encountered when using the Websphere MQ queue GHQ on the Websphere MQ queue manager Name :QIU0, Host :winmvsu0, Port :1481, Channel :MQClient, Bus :ghbus, BM Name :QIU0-ghbus, BM Uuid :FDA1D736CCFAB7FC, Auth alias :, Transport :OutboundBasicWMQClient: com.ibm.mq.MQException: MQJE001: Completion Code 2, Reason 2053.

at  
com.ibm.ws.sib.comms.common.JFAPCommunicator.parseSingleException(JFAPCommunicator.java:1980)  
... 55 more

**Caused by:** com.ibm.websphere.sib.exception.SIErrorException:  
**CWSIC8002E:** An internal error occurred. An unknown or unexpected exception was thrown by the core A  
PI: exception com.ibm.mq.MQException: MQJE001: Completion Code 2, Reason 2053.

---

Examine the exception information and the nested exception information contained in the **Caused by** sections:

- ▶ If a WebSphere MQ reason code is given as the root cause, then the failure is likely be the result of a WebSphere MQ problem (as in the case above).

If this is the case, see Chapter 14, “JMS application with WebSphere MQ problem determination” on page 203.

- ▶ Otherwise, the problem is likely to reside within the service integration bus.

If this is the case, see Chapter 11, “WebSphere MQ server problem determination” on page 161. Specifically, look at section 6.5, “JMS application unable to produce messages” on page 97.

## 11.6 Unable to consume messages from a WebSphere MQ destination

The JMS application might be unable to consume messages from a destination hosted on a WebSphere MQ server.

This topic examines common problems encountered when attempting to consume messages from a destination hosted on a WebSphere MQ server.

### 11.6.1 Collect diagnostics

In addition to the application log and messaging engine SystemOut already collected in 1.3, “Collect diagnostics” on page 19, look in the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.

### 11.6.2 Analyze diagnostics

Search for and examine the `JMSEException` for indications of the problem. Specifically, examine the message included in the final **Caused by**: section. It will look something like Example 11-2.

*Example 11-2 Stack trace in application log containing the exception*

---

```
Stack Dump =  
com.ibm.ws.sib.processor.impl.exceptions.RMQSessionDroppedException:  
CWSIP0815W: A message could not be received from destination MQQ  
assigned to WebSphere MQ Server bus member QIU0-ghbus. The WebSphere MQ  
completion code was 2. The WebSphere MQ reason code was 2016.
```

```

    at
com.ibm.ws.sib.processor.impl.mqproxy.RMQCursor.next(RMQCursor.java:208
)
    at
com.ibm.ws.sib.processor.impl.RMQAsynchThread$AsynchRunnable.runAsynchC
onsumer(RMQAsynchThread.java:553)
    at
com.ibm.ws.sib.processor.impl.RMQAsynchThread$AsynchRunnable.run(RMQAsy
nchThread.java:429)
    at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1469)
Caused by:
com.ibm.ws.sib.remote.mq.exceptions.CorruptRMQSessionException:
CWSJP0002E: An internal messaging error occurred in
com.ibm.ws.sib.remote.mq.impl.AbstractRMQSession.processException,
1:952:1.54 due to an exception com.ibm.mq.MQException: MQJE001:
Completion Code 2, Reason 2016
    at
com.ibm.ws.sib.remote.mq.impl.AbstractRMQSession.processSessionExceptio
n(AbstractRMQSession.java:957)
    at
com.ibm.ws.sib.remote.mq.impl.AbstractRMQSession.processSessionExceptio
n(AbstractRMQSession.java:897)
    at
com.ibm.ws.sib.remote.mq.impl.BaseCursor.next(BaseCursor.java:223)
    at
com.ibm.ws.sib.processor.impl.mqproxy.RMQCursor.next(RMQCursor.java:135
)
    ... 3 more
Caused by: com.ibm.mq.MQException: MQJE001: Completion Code 2, Reason
2016
    at com.ibm.mq.MQQueue.getMsg2(MQQueue.java:1071)

```

---

Examine the exception and nested exception information contained in the **Caused by** sections:

- ▶ If a WebSphere MQ reason code is given as the root cause, then the failure is likely the result of a WebSphere MQ problem (as in the case above).  
If this is the case, see 14.5, “JMS application is unable to consume messages” on page 207.
- ▶ Otherwise, the problem is likely to reside within the service integration bus.  
If this is the case, see Chapter 11, “WebSphere MQ server problem determination” on page 161. Specifically, look at section 6.7, “JMS application unable to consume messages” on page 103.

## 11.7 JMS messages arrive but appear corrupt

This topic examines the reasons that messages arrive at the intended destination but appear to be missing information or have incorrect data in them.

### 11.7.1 Collect diagnostics

Locate the header and payload information of the message that appears incorrect.

Work with the WebSphere MQ administrator to find and examine the message.

### 11.7.2 Analyze diagnostics

Examine the message headers and payload to determine if the message is incorrect:

- ▶ If the message is missing custom JMS headers, see 11.7.3, “WebSphere MQ RFH2 headers not enabled” on page 170.
- ▶ If the JMSXUserID field is either padded with spaces or has been truncated, see 11.8, “JMSXUserID field padded with spaces or truncated” on page 171.
- ▶ If the JMSCorrelationID field is padded with spaces or truncated, see 11.8.1, “JMSCorrelationID field padded with spaces or truncated” on page 172.

### 11.7.3 WebSphere MQ RFH2 headers not enabled

If the custom JMS headers are not propagated, the likely reason for this is that the definition of the destination within the service integration bus does not have the WebSphere MQ RFH2 headers enabled.

If you are expecting messages to arrive on WebSphere MQ queues with these headers included, then you must ensure that the **Include an RFH2 message header** property is checked for the WebSphere MQ queue point you are using.

In the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Select **Destinations**.
4. Click the destination configured to use the WebSphere MQ server.
5. Select **queue points**.

6. Click on the name of the WebSphere MQ queue point.
7. Select the **Include an RFH2 message header when sending messages to WebSphere MQ** check box (as illustrated in Figure 11-1).

The screenshot shows the configuration page for a WebSphere MQ queue point. The breadcrumb navigation is: Buses > gibus > Destinations > MQQ > WebSphere MQ queue points > MQQ@QIU0-gibus. Below the breadcrumb, it says "The message point for a WebSphere queue." and "Configuration". The "General Properties" section contains the following fields and options:

- Identifier: MQQ@QIU0-gibus
- UUID: F7293B23A11DB804
- Target UUID: BA68F887BDF3F1661DBA1C11
- \* WebSphere MQ queue name: GHQ
- Inbound nonpersistent reliability: Reliable nonpersistent
- Inbound persistent reliability: Assured persistent
- Include an RFH2 message header when sending messages to WebSphere MQ

At the bottom, there are buttons for Apply, OK, Reset, and Cancel.

Figure 11-1 Setting to include an RFH2 message header

## 11.8 JMSXUserID field padded with spaces or truncated

If the JMSXUserID field is either padded with spaces or has been truncated, the MQMD is restricted to 8 bytes; therefore, a maximum of 8 characters can be used.

If the field is padded with spaces, then this is because the current implementation copies the user ID from an MQ message when it brings the message into the service integration bus. A direct copy of the 8 byte field is performed so that the user ID appears in the service integration bus as 8 characters. If the field was originally shorter, then it is padded with spaces. The same is true for messages flowing from the service integration bus to WebSphere MQ.

This behavior is a limitation in the current release; therefore, you need to consider it when you develop applications that use this field.

### **11.8.1 JMSCorrelationID field padded with spaces or truncated**

The JMSCorrelationID is processed in much the same way as the JMSXUserID (described above). The field is made to fit the size provided; if the field is shorter than the field length it is padded with 0s. This behavior is a limitation in the current release; therefore, you need to consider it when you develop applications that use this field.

### **11.8.2 Validate the solution**

If you made changes to the WebSphere MQ server definition or JNDI resources, you must restart the application servers that interact with the WebSphere MQ server for the changes to take effect.

Send messages and ensure the expected message properties are now correctly propagated.



## WebSphere MQ configuration

This chapter discusses problem determination techniques and possible root causes for problems when using the WebSphere MQ messaging provider.

## 12.1 Identify symptoms

When applications running in an application server use the WebSphere MQ messaging provider, the server must have access to the WebSphere MQ client code. These symptoms are an indication that the WebSphere MQ messaging provider is not enabled:

- ▶ JMS objects such as queue connection factories are defined, but do not appear in JNDI.
- ▶ WMSGxxxxE error messages appear in the system logs indicating that WebSphere MQ JMS binders have been disabled.

If your application is running in an application server, see 12.2, “WebSphere MQ messaging provider not enabled” on page 174.

WebSphere MQ JMS applications running outside of the application server require access to the WebSphere MQ client binaries. These symptoms are an indication that the WebSphere messaging client cannot resolve the WebSphere MQ client binaries:

- ▶ This exception is reported when running client application:  

```
java.lang.Exception: De-reference of JMS provider's Reference failed  
- check provider is on classpath
```
- ▶ Client application gets JMSEExceptions when attempting to run in bindings mode with a stack trace containing the WebSphere MQ return code 2046.

If your application is running outside of an application server, see 12.3, “Messaging client fails to resolve the WebSphere MQ libraries” on page 177.

## 12.2 WebSphere MQ messaging provider not enabled

For WebSphere Application Server to utilize WebSphere MQ as a messaging provider, it must access the WebSphere MQ JMS client code. This is resolved as part of the WebSphere Application Server install process. The WebSphere MQ code is located under *app\_server\_root/lib/WMQ* and is referenced by the WebSphere Variable `MQ_INSTALL_ROOT` at the node scope.

Depending on your WebSphere Application Server configuration and WebSphere MQ infrastructure, it might become necessary to change the `MQ_INSTALL_ROOT` variable. The most common reasons for this are:

- ▶ To point to a different version of the WebSphere MQ client



- ▶ To utilize WebSphere MQ queue managers running on the same hardware as the WebSphere Application Server environment, that is, run in bindings mode.

If the install directory is set incorrectly, or there are permission problems on the target directory, then the WebSphere MQ JMS binders will be disabled preventing the use of WebSphere MQ as a JMS provider.

## 12.2.1 Analyze SystemOut

Examine the SystemOut log for the server running the application. Look for error messages prefixed with WMSG:

- ▶ If an error was encountered during the loading the WebSphere MQ Client, you see this message in the log:  
**WMSG0902E:** The WebSphere MQ JMS Binders have been disabled as either the WebSphere MQ Client has not been installed, or the MQ\_INSTALL\_ROOT variable has not been set.
- ▶ If you see this message, see 12.2.2, “MQ JMS client not detected” on page 175.  
**WMSG1605E:** It was not possible to detect the MQ JMS client at the specified path {0}.
- ▶ If you see this message, see 12.2.3, “WebSphere MQ Client version not supported” on page 176.  
**WMSG1604E:** It was not possible to use the specified WebSphere MQ JMS client at path {0} because it is at version {1}, the minimum supported version is {2}.
- ▶ If you see this message, see 12.2.4, “Previous version registered” on page 176.  
**WMSG1603E:** An internal error occurred. It was not possible to register the WebSphere MQ JMS client with the application server due to exception {0}.

## 12.2.2 MQ JMS client not detected

The symptom of this problem is this message:

**WMSG1605E:** It was not possible to detect the MQ JMS client at the specified path {0}.

Ensure that the WebSphere Application Server MQ\_INSTALL\_ROOT variable is set correctly and that the WebSphere Application Server process has the correct access permissions.

### 12.2.3 WebSphere MQ Client version not supported

The symptom of this problem is this message:

WMSG1604E: It was not possible to use the specified WebSphere MQ JMS client at path {0} because it is at version {1}, the minimum supported version is {2}.

The version of WebSphere MQ Client referenced by the WebSphere Application Server MQ\_INSTALL\_ROOT variable does not meet the minimum supported level of WebSphere MQ.

Check the System Requirements for WebSphere Application Server 6.1 to ensure that your combination of WebSphere Application Server and WebSphere MQ versions is supported for your environment:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651>

### 12.2.4 Previous version registered

The symptom of this problem is this message:

WMSG1603E: An internal error occurred. It was not possible to register the WebSphere MQ JMS client with the application server due to exception {0}.

A problem has occurred when loading the client as a previous version was already registered. This generally occurs when changing between versions of the client, to resolve this problem perform this steps:

1. Ensure no processes are running for the given WebSphere Application Server profile.
2. Run the **osgiCfgInit** command from the bin directory of the given WebSphere Application Server profile.
3. Restart the WebSphere Application Server nodes and servers for the given profile.

## 12.2.5 Validate the solution

Whenever the WebSphere MQ\_INSTALL\_ROOT variable is changed or access permissions modified, it will be necessary to restart the application server processes for the changes to take effect. After the restart, check the application server logs to ensure no error messages prefixed WMSG are reported.

## 12.3 Messaging client fails to resolve the WebSphere MQ libraries

WebSphere MQ JMS applications running outside of the application server require access to the WebSphere MQ client binaries. For the WebSphere Application Server client container, the location of the WebSphere MQ client is specified within the `${WAS_INSTALL_ROOT}/bin/launchClient` script.

### 12.3.1 Examine the application messages

Problems will be reported by the J2SE application client. Messages are written to the prompt where the `launchClient` command was run.

Examine the application messages for exceptions. Look for these messages:

- ▶ `java.lang.Exception: De-reference of JMS provider's Reference failed - check provider is on classpath`
- ▶ Exceptions when attempting to run in **bindings** mode with a stack trace that contains the WebSphere MQ return code 2046.

These messages indicate that the WebSphere messaging client cannot resolve the WebSphere MQ client binaries. If you are experiencing these symptoms, open the `${WAS_INSTALL_ROOT}/bin/launchClient` script in an editor, and check that the `JMS_PATH` references the correct location for the WebSphere MQ JMS jar files to be used:

- ▶ If the `JMS_PATH` is referencing the incorrect location, update the path to point to the correct location of the MQ JMS jar files that you want to use.
- ▶ If the `JMS_PATH` is referencing the correct location, but there is still a problem, the most likely cause is that the user launching the application does not have the correct access permissions. Update the user permissions and rerun the client application.

It is not necessary to restart the application server for changes to the `${WAS_INSTALL_ROOT}/bin/launchClient` script to take effect.

Archived



## WebSphere MQ link problem determination

This chapter identifies problem determination activities for the WebSphere MQ link component of the default messaging provider in WebSphere Application Server V6.1.

## 13.1 Introduction to WebSphere MQ link

A WebSphere MQ link enables a service integration bus to interconnect with a WebSphere MQ queue manager. Depending on the chosen configuration, the WebSphere MQ link can support both point-to-point and publish-subscribe messaging paradigms.

For a detailed description of the WebSphere MQ link and its configuration options, see *How the WebSphere MQ link interoperates with WebSphere MQ* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.express.doc/concepts/cjc1002\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.express.doc/concepts/cjc1002_.html)

Common problems with the WebSphere MQ link can be categorized into these three areas:

- ▶ Sender channels fail to start.
- ▶ Message flow problems.
- ▶ Corrupt messages.

## 13.2 Verify integrity

The first step in diagnosing the problem is to do this verify the integrity of the system:

- ▶ Verify the WebSphere MQ link is running.
- ▶ Verify the WebSphere MQ link sender channel is running.
- ▶ Verify the WebSphere MQ link receiver channel is running.

For information about the various states of the MQ link, the states of the sender and receiver channels, and their meanings, see *States of the WebSphere MQ link and its channels* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjc0003\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjc0003_.html)

### 13.2.1 Verify the WebSphere MQ link is running

Check the state of the WebSphere MQ link with the administrative console as follows:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.

2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**.
4. Check the Status column.

A Running status indicates a network connection has been established between the application server and queue manager. Messages will be transmitted.

A Standby status indicates the sender channel is not network-connected to its WebSphere MQ counterpart receiver channel. It is waiting for a message to send before attempting to establish a connection.

If the link is in a Stopped state, attempt to start it by selecting the link and clicking **Start**.

### 13.2.2 Verify the WebSphere MQ link sender channel is running

Check the state of the sender channel with the administrative console as follows:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check
4. Select the **Sender Channel**.
5. Check the Status column.

If the channel is in Stopped state, attempt to start it. If it is in Retry state, work with the WebSphere MQ administrator to ensure the WebSphere MQ resources are available.

### 13.2.3 Verify the WebSphere MQ link receiver channel is running

To view or modify the receiver channel status, in the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check.

4. Select **Receiver Channel**.
5. Check the Status column.

If the channel is in Stopped state, attempt to start it.

## 13.3 Identify symptoms

Find the symptoms that most closely resembles the symptoms you are experiencing:

- ▶ The sender channel from WebSphere MQ link fails to start:
  - The sender channel is in state Retry.
  - Attempts to start the WebSphere MQ link sender channel manually result in error similar to this:  
**CWSIQ0017E:** The sender channel Sender\_Channel\_Name for MQLink MQLink\_Name has failed to establish a connection with the remote host Host\_Name/IP\_Address because the remote listener for port Port\_Number is not available.
  - Messages that are targeted at the WebSphere MQ queue manager are backing up on the WebSphere MQ link sender channel.

If you are experiencing similar symptoms, see 13.4, “Sender channel fails to start” on page 183.

- ▶ Receiver channel from WebSphere MQ fails to start.
  - Attempts to manually start the WebSphere MQ sender channel result in the channel going into a Retry state.
  - Messages that are targeted at the service integration bus start to back up on the WebSphere MQ sender channel.

If you are experiencing similar symptoms, see 13.5, “Receiver channel fails to start” on page 187.

- ▶ Message flow problem from the service integration bus to WebSphere MQ:
  - *No message flow:* JMS messages are sent successfully but do not arrive on the WebSphere MQ destination. Examination of the appropriate channels confirms they are in a Running state, but that the message count is not incrementing.
  - *Messages apparently lost:* JMS messages are sent successfully but do not arrive on the WebSphere MQ destination. Examination of the appropriate channels confirms they are in a Running state and that the message count for the channel shows that messages have successfully flowed across the channel.



If you are experiencing similar symptoms, see 13.6, “Encountering a message flow problem from the service integration bus to WebSphere MQ” on page 189.

- ▶ Message flow problem from WebSphere MQ to the service integration bus:
  - *No message flow*: JMS messages are sent successfully but do not arrive on the service integration bus destination. Examination of the appropriate channels confirms they are in Running state, but that the message count is not incrementing.
  - *Messages lost*: JMS messages are sent successfully but do not arrive on the service integration bus destination. Examination of the appropriate channels confirms they Running state and that the message count for the channel shows that messages have successfully flowed across the channel.

If you are experiencing similar symptoms, see 13.7, “Encountering a message flow problem from WebSphere MQ to the service integration bus” on page 195.

- ▶ Messages flow across the WebSphere link but appear corrupt.

Messages arriving on WebSphere MQ destinations do not appear to have maintained standard or user-specific JMS headers.

If you are experiencing similar symptoms, see 13.8, “Messages flow across the WebSphere link but appear corrupt” on page 198.

## 13.4 Sender channel fails to start

This topic examines some of the common configuration issues that prevent the WebSphere MQ link sender channel from starting. The sender channel is required to enable communication from a service integration bus to WebSphere MQ.

### 13.4.1 Collect diagnostics

In addition to the SystemOut log for the application server hosting the messaging engine configured with the WebSphere MQ link (see 1.3, “Collect diagnostics” on page 19), locate the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.

## 13.4.2 Analyze SystemOut

Examine the application server log files for the server hosting the messaging engine configured with the WebSphere MQ link. Look for error messages with prefixes CWSIC and CWSIQ:

- ▶ If you see this message, see 13.4.3, “Cannot contact listener” on page 184.  
**CWSIQ0017E:** The sender channel {0} for MQLink {1} has failed to establish a connection with the remote host {2} because the remote listener for port {3} is not available.
- ▶ If you see this message, see 13.4.4, “Channel not available” on page 185.  
**CWSIC3108E:** The WebSphere MQ link {0} ended because channel {1} is not currently available on the remote system.
- ▶ If you see this message, see 13.4.5, “Attempt to send message failed” on page 185.  
**CWSIC3080E:** The WebSphere MQ link {0} has ended because the remote queue manager {1} cannot receive a message.
- ▶ If you see this message, see 13.4.6, “Message sequence number error” on page 186.  
**CWSIC3011E:** The WebSphere MQ link {0} and the remote queue manager do not agree on the next message sequence number. A message with sequence number {1} has been sent when sequence number {2} was expected.

## 13.4.3 Cannot contact listener

The symptom for this problem is this message:

**CWSIQ0017E:** The sender channel {0} for MQLink {1} has failed to establish a connection with the remote host {2} because the remote listener for port {3} is not available.

Error message CWSIQ0017E indicates that it is not possible to connect to the listener on the host and port provided.

To resolve the problem:

- ▶ Ensure that the host and port displayed in the error message are correct. Modify them in the WebSphere MQ link sender channel as required.
- ▶ Work with your WebSphere MQ administrator to ensure that an associated listener is running on the specified port.
- ▶ Verify that the connection attempt is not blocked by a firewall.

## 13.4.4 Channel not available

The symptom for this problem is this message:

**CWSIC3108E:** The WebSphere MQ link {0} ended because channel {1} is not currently available on the remote system.

Error message CWSIC3108E indicates that the channel name specified is not available on the WebSphere MQ queue manager.

To resolve the problem:

- ▶ Verify that the channel name is correct (and uses the correct case).
- ▶ Work with your WebSphere MQ administrator to ensure that the receiver channel is defined and in Started state on the remote queue manager.

If these steps fail to resolve the problem, examine the WebSphere MQ queue manager logs for error codes that give a clearer indication of the problem. The most common MQ error codes are:

- ▶ AMQ9534: Channel *Channel\_Name* is currently not enabled.

This message indicates that the channel has been placed in Stopped state.

Start the channel manually from WebSphere MQ to allow communication to begin. Investigate why the channel is stopped.

- ▶ AMQ9519: Channel *Channel\_Name* not found

This message indicates that the channel has not been defined or that the channel name entered within the WebSphere MQ link sender is incorrect.

Define the required receiver channel or adjust the channel name provided for the WebSphere MQ link sender accordingly.

## 13.4.5 Attempt to send message failed

The symptom for this problem is this message:

**CWSIC3080E:** The WebSphere MQ link {0} has ended because the remote queue manager {1} cannot receive a message.

CWSIC3080E indicates that an attempt to send a message to WebSphere MQ failed.

This problem can occur when a message is sent to WebSphere MQ but WebSphere MQ could not forward the message to the requested destination. In addition, it could not place the message on the queue managers dead-letter-queue. The channel is in doubt about which messages have been committed by WebSphere MQ for the unit of work that it has sent. The In doubt

setting on the channel is set to true, which prevents WebSphere Application Server from sending any more messages to WebSphere MQ until it is resolved. The message is retained on the WebSphere MQ link sender channel. So see the In doubt value, in the administrative console, look on the Runtime tab of the sender channel.

To diagnose this problem, work with the WebSphere MQ administrator. The diagnosis process is similar to the one you would use if messages could not be sent from WebSphere MQ to WebSphere Application Server. For more information, see 13.7, “Encountering a message flow problem from WebSphere MQ to the service integration bus” on page 195.

### 13.4.6 Message sequence number error

The symptom for this problem is this message:

**CWSIC3011E:** The WebSphere MQ link {0} and the remote queue manager do not agree on the next message sequence number. A message with sequence number {1} has been sent when sequence number {2} was expected.

Error message CWSIC3011E indicates that, for some reason, the two ends of the channel do not agree on the next message sequence number.

This problem might be the result of deleting and recreating one end of the channel pairing. If this is the case, reset the sequence number on the WebSphere MQ link sender channel.

To check the state of the WebSphere MQ link sender channel with the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check.
4. Select the **Sender Channel**.
5. Select the box next to the sender channel, and click **Reset**.

**Note:** If you are using WebSphere MQ NPM\_SPEED settings, message CWSIC3011E might also occur as a result of a WebSphere Application Server defect. This problem is reported in APAR PK49927.

## 13.4.7 Validate the solution

If you made changes to the WebSphere MQ link sender channel definition, restart the application server configuration so that they take effect.

Start the sender channel and ensure that the channel is in the Running state.

## 13.5 Receiver channel fails to start

This topic examines some of the common configuration issues that can be encountered when attempting to start WebSphere MQ sender channels to enable communication from WebSphere MQ to a service integration bus.

### 13.5.1 Collect diagnostics

Locate the WebSphere MQ queue manager error logs. Work with your WebSphere MQ administrator to collect the appropriate logs for analysis for your platform.

### 13.5.2 Analyze the WebSphere MQ queue manager logs

Examine the WebSphere MQ queue manager error log files and check for message codes: AMQ9520, AMQ9558, and AMQ9202:

- ▶ If you see this message, see 13.5.3, “Channel not defined remotely” on page 187.  
**AMQ9520:** Channel not defined remotely
- ▶ If you see this message, see 13.5.4, “Remote channel not available” on page 188.  
**AMQ9558:** Remote Channel is not currently available
- ▶ If you see this message, see 13.5.5, “Remote host not available” on page 188.  
**AMQ9202:** Remote host Host Name not available, retry later

### 13.5.3 Channel not defined remotely

The symptom for this problem is this message:

**AMQ9520:** Channel not defined remotely

This message indicates that the WebSphere MQ link channel name, defined on the WebSphere MQ sender, cannot be located on the remote (WebSphere Application Server) system.

Ensure the channel name is correct, observing any capitalization, and that the appropriately named WebSphere MQ link receiver channel is defined for the WebSphere MQ link configuration.

### 13.5.4 Remote channel not available

The symptom for this problem is this message:

**AMQ9558:** Remote Channel is not currently available

This message indicates that the channel on the remote (WebSphere Application Server) system is defined but is not available. This problem might be due to the channel being placed in the Stopped state or because insufficient resources are available.

Ensure the WebSphere MQ link receiver is not in Stopped state. If it is, start it.

If the channel is not stopped, check the application server logs for further indications of the reason why the channel is unavailable.

### 13.5.5 Remote host not available

The symptom for this problem is this message:

**AMQ9202:** Remote host Host Name not available, retry later

This message indicates that it was not possible to connect to the service integration bus on the host and port provided under the connection name (conname) attribute of the WebSphere MQ sender channel definition.

Verify that the host and port are correct. The port number should match the port defined for the SIB\_MQ\_ENDPOINT\_ADDRESS on the appropriate application server. The default for this port is 5558.

To check the ports for the application servers in your configuration use the administrative console to:

1. Select **Servers** → **Application Servers**.
2. Click the server name on which your WebSphere MQ link is defined, to open the details page.
3. Click to expand the Ports sub-section under the Communications heading.

4. Note the `SIB_MQ_ENDPOINT_ADDRESS` port number.

### 13.5.6 Validate the solution

If you made changes to the WebSphere MQ link definition or JNDI resources, restart the application server so that they take effect.

Ask the WebSphere MQ administrator to start the WebSphere MQ sender channel.

## 13.6 Encountering a message flow problem from the service integration bus to WebSphere MQ

This topic examines some of the issues that can affect message flow from the service integration bus using a WebSphere MQ link to WebSphere MQ destinations.

### 13.6.1 Determine if the message has reached the WebSphere MQ link

Determine how far through the system the message has progressed. Using the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check.
4. Select **Sender Channel**.
5. Click on the name of the WebSphere MQ sender channel.
6. Select the **Runtime** tab.

You see the panel that displays the runtime state of the sender channel, as illustrated in Figure 13-1 on page 190.

Runtime State	
Number of messages sent	1
Buffers sent	4
Buffers received	4
Bytes sent	2402
Bytes received	212
Last message send time	12:56 PM
Last message send date	5/5/07
Remaining short retry starts	10
Remaining long retry starts	999999999

Figure 13-1 Sender channel runtime state

Check the **Number of messages sent** field. This field should increment when messages are sent.

## 13.6.2 Collect diagnostics

In addition to the SystemOut log for the server hosting the messaging engine configured with the WebSphere MQ link, and the application log, you need the following information:

- ▶ Examine the exception destination for the messaging engine hosting the MQ link. For more information about exception destinations, see *Exception destinations* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjo0004\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjo0004_.html)



**Note:** If the **number of messages sent** count on the sender channel is not incrementing, the messages are not reaching WebSphere MQ and you do not need diagnostic data from WebSphere MQ.

- ▶ Locate the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.
- ▶ Examine the dead letter queue on the appropriate queue managers on the WebSphere MQ system.

### 13.6.3 Analyze diagnostics

If the messages on the sender channel are *incrementing*, the problem is occurring on the WebSphere MQ side. See 13.6.5, “WebSphere MQ errors” on page 193.

If the messages on the sender channel are *not incrementing*, the messages are not making it as far as the WebSphere MQ link. Continue analyzing the diagnostics.

If you are producing the messages using a messaging engine that is not hosting the MQ link, you could be experiencing a problem related to inter- messaging engine communication. See Chapter 7, “Messaging in a multiple messaging engine environment” on page 109.

#### Analyze SystemOut

Search the SystemOut log for:

**CWSIC3096I:** While sending message to queue manager *Queue\_Manager\_Name* down WebSphere MQ link *MQLink\_Name* one or more messages were put to the exception destination.

This message indicates that the message could not be sent across the MQ Link and that the message has been placed on the exception destination.

In general, this error message is accompanied by another error message indicating the cause, which is also added into the message when routed to the exception destination.

If this message is also present, see 13.6.4, “Message is too large” on page 192.

**CWSIK0104E:** The message is too big to be handled by remote MQ; sending to exception destination: message size {0}, maximum message size {1}.

## Analyze the application log

Examine the application log for exception information that might help you determine the cause. The initial exception received by the application (deployed server side) resides here. This exception should help narrow the JMS resources involved in the operation which failed.

### 13.6.4 Message is too large

The symptom for this problem is this message:

**CWSIK0104E:** The message is too big to be handled by remote MQ; sending to exception destination: message size {0}, maximum message size {1}.

A common reason a message is routed to the exception destination when sending to WebSphere MQ is because the message is too large to be processed by the MQ system. It is bigger than allowed by the maximum message size negotiated by the sender/receiver channel pairing.

You can increase this size can be increased for larger messages. However, you must adjust the size on both sides of the channel.

In WebSphere Application Server, the maximum message size setting is configured against the WebSphere MQ link instead of on the channel. To configure this setting, from the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you want to check.

You see the WebSphere MQ link configuration panel, as illustrated in Figure 13-2 on page 193.

The screenshot shows a configuration window titled "Connection" with the following fields:

- \* Foreign bus name:** A dropdown menu showing "QIU0".
- \* Queue manager name:** A text input field containing "ghbus".
- Batch size:** A text input field containing "50".
- Maximum message size:** A text input field containing "4194304" followed by the unit "bytes".

Figure 13-2 Adjusting the maximum message size

To change the maximum message size on WebSphere MQ, work with your WebSphere MQ administrator to update the WebSphere MQ channel configuration.

### 13.6.5 WebSphere MQ errors

If the sender channel number of messages sent is incrementing, messages are successfully making it to the WebSphere MQ link.

Work with your WebSphere MQ administrator to check the appropriate queue manager's dead letter queue and examine the WebSphere MQ error logs to determine where the message has been routed.

In general, if the message is persistent, it will either be on a transmission queue pending a send to another queue manager as part of a multi-hop configuration, or it will have been placed upon the dead letter queue.

If the message has been placed on the dead letter queue, check the dead letter header for an indication of the reason the message was placed there.

Common reasons are as follows:

- ▶ MQRC\_Q\_FULL

The target destination has already reached its maximum capacity and cannot receive any more messages.

Investigate why the WebSphere MQ queue has become full and, if appropriate, increase the capacity of the queue.

- ▶ MQRC\_UNKNOWN\_OBJECT\_NAME

The queue name specified is not defined in the local WebSphere MQ queue manager. The destination that was used is included as part of the dead letter header information.

Ensure that it is correct. Either adjust the JMS/alias definition for the destination in the originating application server configuration, or define the queue in the target queue manager.

► MQRC\_UNKNOWN\_REMOTE\_Q\_MGR

The message was sent specifying a target queue manager that is not known by the receiving queue manager.

The primary reason this problem occurs is that the foreign bus does not have the same name as the remote queue manager; for example, the queue manager name is QIU1 but the foreign bus is named MQSystem. If this is the case, messages that flow from the service integration bus to the queue manager will have the queue manager name set to MQSystem which is not known by the receiving queue manager.

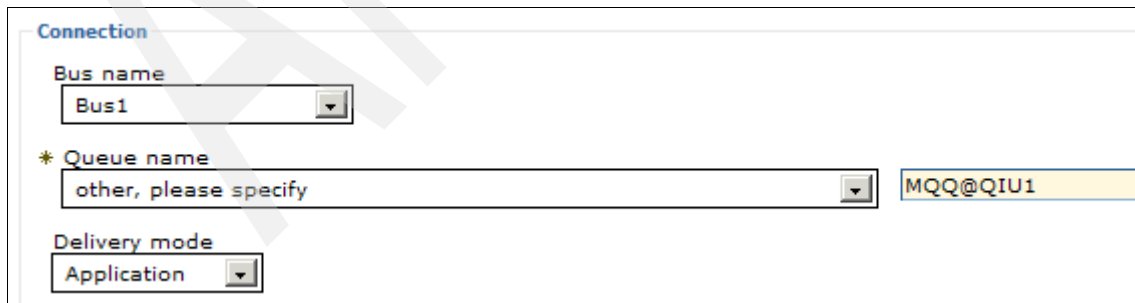
To overcome this problem, use an @ notation in the queue name specified in the JMS queue definition to define the specific target queue manager. See the example shown in Figure 13-3.

The format is:

*queue\_name@queue\_manager*

For example, in the administrative console:

- Select **Resources** → **JMS** → **Queues**.
- Click on the JMS queue name.
- In the Connection area, specify the bus name where the MQ link is defined.
- Select **other, please specify** for the Queue name, and enter the queue name as in Figure 13-3.



The screenshot shows a configuration window titled "Connection". It contains three main sections:

- Bus name:** A dropdown menu with "Bus1" selected.
- \* Queue name:** A dropdown menu with "other, please specify" selected. To its right is a text input field containing "MQQ@QIU1".
- Delivery mode:** A dropdown menu with "Application" selected.

Figure 13-3 Specifying the target queue with queue manager

### 13.6.6 Validate the solution

If you made changes to the WebSphere MQ link definition or JNDI resources, restart your application server configuration.

Send messages and ensure they arrive on the target WebSphere MQ destination.

## 13.7 Encountering a message flow problem from WebSphere MQ to the service integration bus

This topic examines some of the issues that can affect message flow from WebSphere MQ to service integration bus destinations via the WebSphere MQ link.

### 13.7.1 Verify integrity

Work with the WebSphere MQ administrator to:

- ▶ Determine how far through the WebSphere MQ system the message has progressed.
- ▶ Check the runtime status of the WebSphere MQ sender channel.
- ▶ Determine if the messages counter is incremented when messages are sent.

### 13.7.2 Collect diagnostics

Collect the following information:

- ▶ In the application server SystemOut log files, determine:
  - The server hosting the messaging engine configured with the WebSphere MQ link.
  - The server hosting the application which is attempting to receive messages.
  - The servers hosting messaging engines involved in routing the message.

For information about collecting the SystemOut log, see 1.3, “Collect diagnostics” on page 19.

- ▶ Examine the exception destination for the messaging engines involved in your messaging flow and hosting the WebSphere MQ link. For more information about exception destinations, see *Exception destinations* at:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjo0004\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjo0004_.html)
- ▶ Locate the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.
- ▶ Examine the dead letter queue on the appropriate queue managers on the WebSphere MQ system.

### 13.7.3 Analyze diagnostics

To analyze diagnostics:

- ▶ If the WebSphere MQ sender channel messages counter is *not incrementing*, the messages are not being transmitted across the channel. Examine the WebSphere MQ queue manager errors logs and dead letter queue to determine the location of the message and the reason it cannot be sent.

If you see message MQRC\_UNKNOWN\_REMOTE\_Q\_MGR, see 13.7.4, “Target queue manager is not known” on page 197.

- ▶ If the WebSphere MQ sender channel messages counter is *incrementing*, messages are being sent across the MQ link but, for some reason, they are *stuck* or *lost* in the service integration bus.

Examine the application server logs and the exception destinations for any applicable messaging engines to determine the location of the message and the reason the message has not been delivered.

Common problems include:

- The target destination does not exist so the message has been routed to the exception destination.
- The target destination is full, and the message has been routed to the exception destination.
- The target destination is on a messaging engine that cannot be contacted, and the message is held on a remote queue point until the message can be delivered.
- The target destination specifies a bus that does not exist and the message has been routed to the exception destination.

If you are producing the messages using a messaging engine that is not hosting the MQ link, you could be experiencing a problem related to inter-messaging

engine communication. See Chapter 7, “Messaging in a multiple messaging engine environment” on page 109.

### 13.7.4 Target queue manager is not known

A common problem, particularly when using request-reply, is that the target queue manager is not known by the local queue manager. This problem, characterized by message MQRC\_UNKNOWN\_REMOTE\_Q\_MGR, might be due to a WebSphere MQ link configuration error.

The **Queue manager name** attribute (shown in Figure 13-4 on page 198) on the WebSphere MQ link represents the name of the queue manager that WebSphere Application Server represents. The WebSphere Application Server must specify the same value on WebSphere MQ as the remote queue manager, using the WebSphere administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines**. Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link.

**Connection**

- \* Foreign bus name: QIUO
- \* Queue manager name: ghbus
- Batch size: 50
- Maximum message size: 4194304 bytes
- Heartbeat interval: 300 seconds
- Sequence wrap: 999999999
- Adoptable
- Initial state: Started

Figure 13-4 Defining the queue manager name for the foreign bus

### 13.7.5 Validate the solution

If you have made changes to the WebSphere MQ link definition or JNDI resources, you must restart your application server configuration so that they take effect. To confirm the solution to the message flow problem, resend additional messages and ensure they arrive on the target service integration bus destination.

## 13.8 Messages flow across the WebSphere link but appear corrupt

This topic examines reasons that messages arrive at the intended destination but appear to have missing information or incorrect data in them.



## 13.8.1 Collect diagnostics

Collect the header and payload information of the message that appears incorrect.

The consuming application usually has an error when processing the header information of the message.

The approach for analyzing the message depends on the direction of transport:

- ▶ WebSphere MQ to WebSphere Application Server

Look at the message on the exception destination. Alternatively, modify the consuming application to output the content of the message to a log, for example. There is a small chance the message might be rolled-back to a queue point.

- ▶ WebSphere Application Server to WebSphere MQ

You need to work with the WebSphere MQ administrator to recover the content of the message.

## 13.8.2 Analyze diagnostics

Examine the message headers and payload to determine if the message is incorrect.

If the message is missing JMS specific headers, such as JMSCorrelationID or custom defined user headers, see 13.8.3, “MQRFH not included” on page 199.

## 13.8.3 MQRFH not included

By default, the MQRFH is not included when sending messages from a service integration bus to WebSphere across a WebSphere MQ link. As a result, any JMS specific headers such as JMSCorrelationID and custom defined user headers are not included in the message.

**Note:** This is not the case when messages flow into the service integration bus from WebSphere MQ, because all messages are converted into JMS messages.

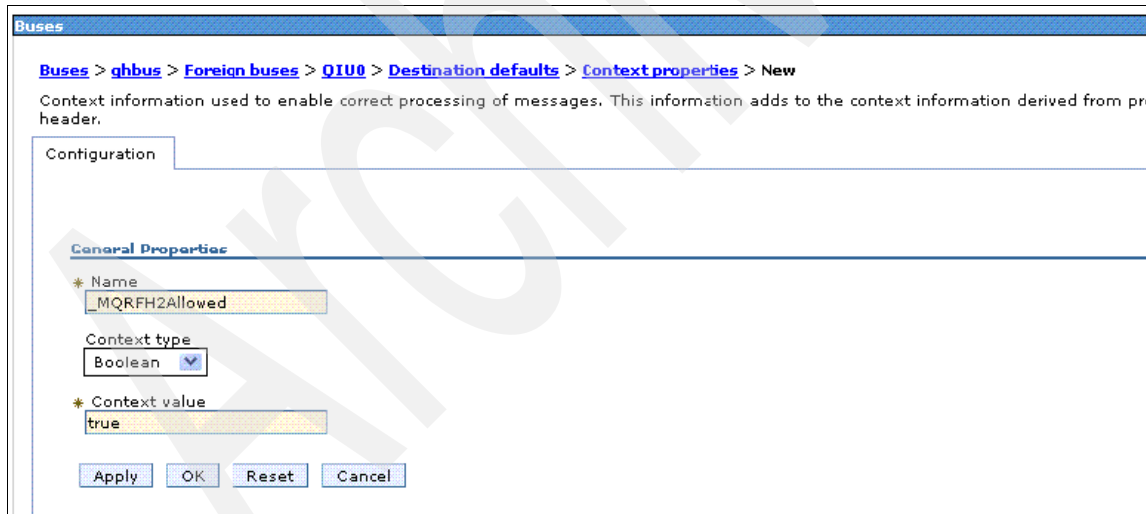
To include the JMS headers on outbound messages, you must set the `_MQRFH2Allowed` context property. You can set it for individual alias destinations, foreign destinations, or on the WebSphere MQ link destination

default so that all messages sent across the WebSphere MQ link will include the JMS headers. You must set this property with a Boolean value of true.

Using the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Foreign buses**. Click the foreign bus name to open the details page.
3. Click **Destination defaults**.
4. Click **Context properties**.
5. Click **New**:
  - Enter `_MQRFH2Allowed` for the Name.
  - Select Boolean for the Context type.
  - Enter `true` for the Context value.
6. Click **OK**.

Figure 13-5 shows the panel you use to set the property.



The screenshot shows a web browser window with the title 'Buses'. The breadcrumb navigation is 'Buses > qhbus > Foreign buses > QIU0 > Destination defaults > Context properties > New'. Below the breadcrumb is a description: 'Context information used to enable correct processing of messages. This information adds to the context information derived from pr header.' The main content area is titled 'Configuration' and contains a section for 'General Properties'. Under 'General Properties', there are three fields: 'Name' with the value 'MQRFH2Allowed', 'Context type' with a dropdown menu set to 'Boolean', and 'Context value' with the value 'true'. At the bottom of the form are four buttons: 'Apply', 'OK', 'Reset', and 'Cancel'.

Figure 13-5 Enable MQRFH headers

## 13.9 Validate the solution

If you made changes to the WebSphere MQ link definition or to JNDI resources, restart the application server so that they take effect.

Send messages and ensure that the expected message properties are now correctly propagated.

Archived

Archived

# JMS application with WebSphere MQ problem determination

This chapter investigates some of the common problems encountered when an application attempts to do one of the following:

- ▶ Create a connection to WebSphere MQ using the WebSphere MQ messaging provider.
- ▶ Produce (or send) a message to a WebSphere MQ destination within a JMS application.
- ▶ Consume a message from a WebSphere MQ destination.

## 14.1 Identify symptoms

The symptom that a JMS application is *unable to connect to WebSphere MQ* is that the application receives a `JMSEException` when it attempts the connection. If this symptom fits your problem, see 14.3, “JMS application is unable to create a connection” on page 205.

Symptoms that an application is *unable to produce messages* to WebSphere MQ include:

- ▶ A JMS application receives a `JMSEException` while attempting to create a producer for a given WebSphere MQ destination.
- ▶ A JMS application receives a `JMSEException` while attempting to send a message.
- ▶ A JMS application appears to have sent the message successfully, but the message never arrives on the expected WebSphere MQ destination.

If these symptoms fit your problem, see 14.4, “JMS application is unable to produce messages” on page 206.

The symptom that an application is *unable to consume messages* from WebSphere MQ is that the application receives a `JMSEException` when attempting to create a consumer on a given WebSphere MQ destination. If this symptom fits your problem, see 14.5, “JMS application is unable to consume messages” on page 207.

## 14.2 Analyze the application log

Examine the application server logs for JMS exceptions. If the root cause is a WebSphere MQ problem, look for a WebSphere MQ reason code should be included in the linked exception information.

To determine the cause of the problem, you need to identify the WebSphere MQ reason code. For example, in the section of stack trace shown below, the WebSphere MQ reason code is 2059.

```
:Caused by: com.ibm.mqservices.MQInternalException: MQJE001: An
MQException occurred: Completion Code 2, Reason 2059
MQJE011: Socket connection attempt refused
:
```

**Tip:** Use the `mqrc reason_code` command on a WebSphere MQ system to get a brief description of the given reason code.

The following sections discuss some of the more common problems you might find.

## 14.3 JMS application is unable to create a connection

The following are some of the more common WebSphere MQ reason codes you might experience:

▶ Reason code 2035

The user ID being provided on the connection attempt is not authorized on the WebSphere MQ system. To resolve this issue:

- Provide an authentication alias to be used by the connection that is an authorized user on the WebSphere MQ System.
- Provide a valid user ID/password on the client application's call to the `createConnection()` method.

▶ Reason code 2058

The queue manager name provided on the connection attempt does not match the queue manager to which it has been connected.

Two reasons this error can occur are:

- Although the connection factory is configured with the correct host, port, and queue manager name, the connection method has been set to **bindings**. This is the default when defining an WebSphere MQ connection factory. The connection attempt is made locally.

If trying to connect to a remote system, ensure the connection method is set to **client**.

- The connection has successfully been made, but the queue manager name provided does not match the queue manager name with which the WebSphere MQ listener is associated.

Ensure the queue manager name in the connection factory is correct, including the correct case.

▶ Reason code 2059

The queue manager to which you are attempting to connect is not available.

Examine the WebSphere MQ queue manager error log for an indication of the root cause. If the reason for the failure is not apparent, check the following:

- If the listener is running but the queue manager itself is not started, ensure the queue manager is running.

- If the connection is trying to use a client connection but the listener is not running or the port provided on the connection factory is incorrect, ensure that the listener is running and the port is correct.

## 14.4 JMS application is unable to produce messages

The following are some of the more common WebSphere MQ reason codes you might experience:

- ▶ Failures that occur when creating a producer for the given WebSphere MQ destination:
  - Reason code 2035  
The user ID is not authorized to perform this action.  
Ensure the correct user id is being used and that the correct security settings are in place for the WebSphere MQ queue manager.
  - Reason code 2051  
The destination you are attempting to send to is put disabled.  
Investigate why the queue might be disabled, and modify the configuration appropriately.
  - Reason code 2085  
The destination to which you are attempting to connect does not exist.  
The first line of the exception contains a message similar to this:  
**MQJMS2008:** failed to open MQ queue Queue\_Name  
Ensure that the queue name is correct and exists within the target WebSphere MQ queue manager.
  - Reason code 2087  
The destination to which you are attempting to send is located on a remote queue manager and has been specified in the JMS queue definition; however, the local queue manager is not configured to route messages to the remote queue manager.  
Work with the WebSphere MQ administrator to ensure that the configuration is set up to route messages between WebSphere MQ queue managers.
- ▶ A failure that occurs when sending a message produces reason code 2053.  
The destination to which you are attempting to send already contains the maximum number of messages.



First, investigate why the queue is full and then identify appropriate actions to resolve the problem. For example, you might need to increase the maximum depth of the queue.

If the application appears to have sent the message but it does not reach the target destination, the following are possible causes:

- ▶ Message on dead letter queue.

Check the WebSphere MQ queue manager's dead letter queue (if one is defined) to see if the message has been moved there. If this is the case, then examine the dead letter header for more information to explain why the message was moved.

- ▶ Target destination cannot be reached.

If the target destination was hosted on a remote queue manager, ensure that the channels between queue managers are running.

## 14.5 JMS application is unable to consume messages

The following are some of the more common WebSphere MQ reason codes you might experience:

- ▶ Reason code 2016

The destination from which you are attempting to receive is inhibited.

Investigate why the queue might be inhibited and modify the configuration appropriately.

- ▶ Reason code 2035

The user ID being used is not authorized to perform this action.

Ensure the correct user ID is being used and that the correct security settings are in place for the WebSphere MQ queue manager.

- ▶ Reason code 2085

The destination to which you are attempting to connect does not exist. In such a case, the first line of the exception contains a message similar this:

**MQJMS2008:** failed to open MQ queue Queue\_Name

Ensure that the queue name is correct and exists within the target WebSphere MQ queue manager.

## 14.6 Validate the solution

If you made changes to the WebSphere Application Server JMS or JNDI resources, restart the application server so that changes take effect. If changes were only made to WebSphere MQ queue manager objects, you do not normally need to restart the application server.

Run the application and ensure the problem has been resolved.



## Foreign bus problem determination

This chapter identifies common problems that can occur when setting up a foreign bus link using the default messaging provider. These problems are related to configuration errors when defining a new foreign bus link or when configuring an application to use the link.

## 15.1 Introduction to foreign buses

A foreign bus is a representation of another service integration bus, providing messaging functionality from one service integration bus to another. A foreign bus link can only be used for send operations. You cannot receive messages from a remote bus across a foreign bus link.

For a detailed description of a foreign bus, see *Foreign buses* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.iseries.doc/concepts/cjj0030\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.iseries.doc/concepts/cjj0030_.html)

For examples that use foreign buses, see 15.5, “Foreign bus link usage examples” on page 213.

### 15.1.1 Identify symptoms

Common symptoms of a foreign bus problem include:

- ▶ A messaging engine fails to start after configuring a foreign bus link.
- ▶ A foreign bus link fails to start.
- ▶ A JMS application gets an exception while attempting to produce messages on a foreign destination.

If the messaging engine and foreign bus link both start, you might be experiencing application problems.

## 15.2 Verify system integrity

The first step in diagnosing the problem is to verify the status of the messaging engine and the foreign bus link.

### 15.2.1 Verify the status of the foreign bus link

The first step in diagnosing any foreign bus problem is to determine if the foreign bus links on the buses are active. To check the status of the foreign bus links from the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.

4. Click the messaging engine name to open the details page.
5. Click **Service integration bus links** (under Additional Properties heading).

Ensure the foreign bus link is in a Started state (a solid green arrow icon). If the link is not started, attempt to start it by selecting the box to the left of the link and clicking **Start**.

If the link will not start, you might have a configuration problem.

## 15.2.2 Verify the messaging engine has started

A problem with foreign bus configuration can prevent the messaging engine from starting. Use the instructions in 1.2.3, “Determine your messaging engine status” on page 8 to ensure the messaging engine has started.

If the messaging engine is not started, attempt to start it. If the link will not start, you might have a configuration problem.

## 15.3 Analyze diagnostics

Problems with a foreign bus configuration are usually indicated by messages in the application server SystemOut log for the messaging engines. If the messaging engine or link to the foreign bus failed to start, you see messages that indicate the problem in this log.

If the messaging engine and foreign bus both appear to be successfully started, the problem might be related to the application.

### 15.3.1 Analyze SystemOut for the messaging engines

A foreign bus is a link between two messaging engines in two service integration buses. Analyze the SystemOut log for the application servers hosting both messaging engines.

Examine the SystemOut logs for these error messages:

- ▶ If you see this message, the name of the bus you defined in the foreign bus configuration does not match the name of the remote bus.  
**CWSIT0057E:** The inter-bus connection Foreign\_Bus\_Link\_Name failed in the remote messaging engine on host Host\_Name with reason:  
**CWSIT0086E:** Bus Bus\_Name not found.  
Modify the foreign bus configuration to reference the correct remote bus name.
- ▶ If you see this message, the name of the foreign bus link you defined in this bus was not the same as the name of the foreign bus link on the other bus, or the name of the messaging engine specified in the link is incorrect.  
**CWSIT0057E:** The inter-bus connection Foreign\_Bus\_Link\_Name failed in the remote messaging engine on host Host\_Name with reason:  
**CWSIT0067E:** Inter-bus connection Foreign\_Bus\_Link\_Name in bus Remote\_Bus\_Name is not available.  
Ensure that the name of the foreign bus link is the same in both buses, or correct the messaging engine name.
- ▶ If you see this message, you most likely have a configuration error related to the JMS resources used by the application.  
**CWSIA0062E:** Failed to create a MessageProducer for {0}  
Examine the application log for further information about the error.
- ▶ If you see this message, you have a security access error.  
**CWSII0219W:** The bus bus1 denied the user Bus1User access to send a message to the foreign bus bus2.  
See 15.5.4, “Securing foreign bus environments” on page 222 for an example of the correct way to configure the security for the foreign bus connection.

### 15.3.2 Analyze the application log

Examine the application log for exceptions creating the queue sender:

- ▶ If you see this message, you might have a configuration error.  
Exception caught while creating the queue sender for queue jms/q :  
javax.jms.InvalidDestinationException  
One possible cause of this error is that you have defined the JMS queue to point to the foreign destination on the local bus. It should point to the real destination on the foreign bus.

For an example of this type of configuration error, see 15.5.2, “Configuring the JMS queue to point to the correct bus” on page 216.

If you found an error message other than those listed and the information included in the messages was not sufficient to determine the root cause of the message, see Chapter 18, “The next step” on page 261.

## 15.4 Validate the solution

Restart the application servers and check that the messaging engines and foreign bus links start. Rerun the JMS application and ensure that messages can now be produced.

## 15.5 Foreign bus link usage examples

The following sections provide examples of two areas of foreign bus usage that are commonly configured incorrectly and can cause problems.

### 15.5.1 Establishing a foreign bus

The following example illustrates how a foreign bus configuration is established between two existing buses (Bus1 and Bus2), each with a single messaging engine, and is used as a reference throughout this topic.

An application has been deployed into an application server scope where Bus1 is defined as an existing local bus. The application needs to access resources on another bus, Bus2. To fulfill this requirement, you need to update both existing configurations to provide direct access between them. The following steps show how this might be achieved.

On Bus1:

1. Create a foreign bus definition, which has the same name as the remote bus (in this case, Bus2). See Figure 15-1 on page 214.

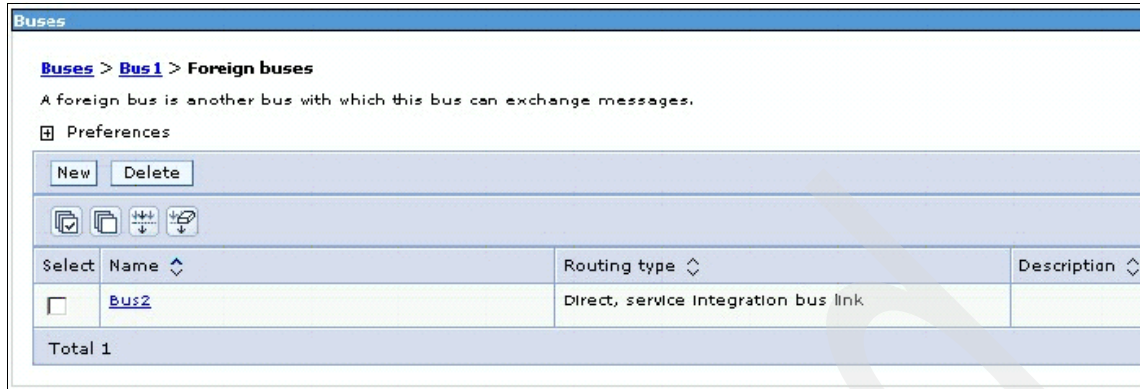


Figure 15-1 Foreign bus definition on Bus1

Note: You can also define other routing properties for the foreign bus (on both buses) for security and topic space mapping. In this example, they are left unchanged.

2. Define a foreign bus link, (for example, named `SameBusLink`), that has `Bus2` as its foreign bus name and a remote messaging engine definition with the name of a messaging engine in `Bus2`. See Figure 15-2.

**Tip:** Copy the name of the messaging engine into the clipboard before creating this link.



Figure 15-2 Foreign bus link

On Bus2:

1. Create a foreign bus definition. Use the same name as the remote bus, in this case, `Bus1`. See Figure 15-3 on page 215.



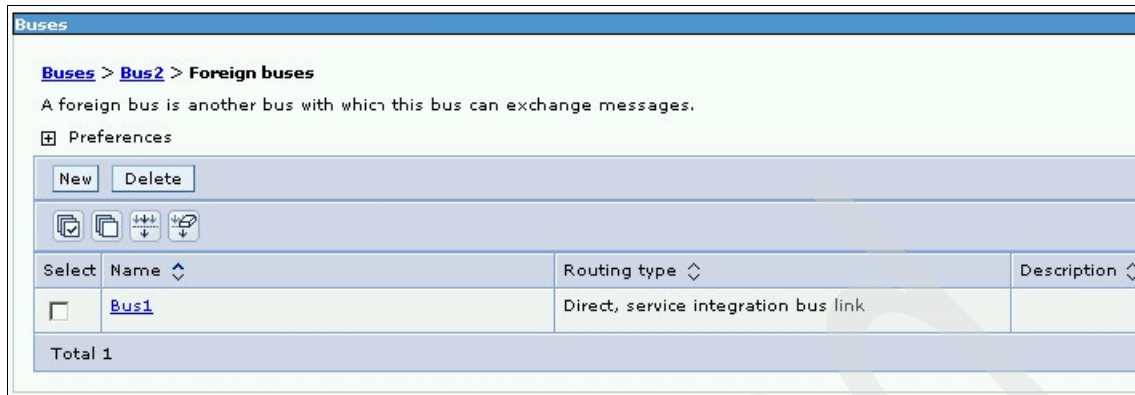


Figure 15-3 Foreign bus on Bus2

2. Define a foreign bus link with the following properties:
  - Foreign bus link name: SameBusLink
  - Foreign bus: Bus1
  - Remote messaging engine: The name of a messaging engine in Bus1. See Figure 15-4.



Figure 15-4 Foreign bus link

In point-to-point messaging, you can only send messages to a remote queue; the remote queue cannot receive messages.

A foreign bus is really a link between a messaging engine in one bus and a messaging engine in another. To enable the application server to process messages across to the foreign bus, you must configure a foreign bus, and both messaging engines and both foreign bus links must be started. When the bus link is started on one bus, both sides of the link usually become active.

For a detailed description of configuring foreign buses, see *Configuring the properties of a foreign bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.iseries.doc/tasks/tjj0078\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.iseries.doc/tasks/tjj0078_.html)

Deleting and recreating or modifying the properties on only one side of a foreign bus link can cause problems. You should delete, and then recreate the complete foreign bus link configuration with the correct properties. Then, restart the application servers so that the foreign bus configuration changes take effect.

## 15.5.2 Configuring the JMS queue to point to the correct bus

Configuration errors are the most common type of problem experienced when attempting to use a foreign destination to produce messages to a queue on a remote bus using a JMS client and JMS queue definitions.

Perhaps the most common problem is configuring the JMS queue to point to the incorrect destination.

**Note:** The JMS queue must point to the real destination on the remote (foreign) bus, not the foreign destination on the local bus.

For example, consider the situation in which you have an application that wants to produce messages to Bus2Queue. Bus2Queue resides on bus2 and is defined as a foreign destination to Bus1.

Figure 15-5 on page 217 shows the definition for Bus2Queue from the viewpoint of Bus1.

**Buses**

**Buses > Bus1 > Destinations**

A bus destination is defined on a service integration bus, and is hosted by one or more locations within the bus. Applications, producers, consumers, or both to exchange messages.

Preferences

New Delete Mediate Unmediate

Select	Identifier	Bus	Type
<input type="checkbox"/>	<a href="#">AliasToWrong</a>	Bus1	Alias
<input type="checkbox"/>	<a href="#">Bus1Queue</a>	Bus1	Queue
<input type="checkbox"/>	<a href="#">Bus1TopicSpace</a>	Bus1	Topic space
<input type="checkbox"/>	<a href="#">Bus2Queue</a>	Bus2	Foreign
<input type="checkbox"/>	<a href="#">Default.Topic.Space</a>	Bus1	Topic space
<input type="checkbox"/>	<a href="#">WrongNameForRemoteQ</a>	Bus2	Foreign
<input type="checkbox"/>	<a href="#">_SYSTEM.Exception.Destination.BLADE202Node01.server1-Bus1</a>	Bus1	Queue

Total 7

Figure 15-5 Bus2Queue defined as a foreign destination to Bus1

Figure 15-6 shows the configuration of Bus2Queue from the viewpoint of Bus2. Bus2Queue is a queue on Bus2.

**Buses**

**Buses > Bus2 > Destinations**

A bus destination is defined on a service integration bus, and is hosted by one or more locations within the bus. Applications, producers, consumers, or both to exchange messages.

Preferences

New Delete Mediate Unmediate

Select	Identifier	Bus	Type
<input type="checkbox"/>	<a href="#">Bus2Queue</a>	Bus2	Queue

Figure 15-6 Bus2Queue defined as a topic space on Bus2

### Correct configuration

Figure 15-7 on page 218 shows the correct configuration for JMS queue `.jms/q` and destination `Bus2Queue`. The JMS queue references the real destination on the remote (foreign) bus, `Bus2`.

**Queues > AnyOldName**

A JMS queue is used as a destination for point-to-point messaging. Use JMS queue destination administrative objects to manage the queue.

Configuration

---

**General Properties**

**Administration**

Scope  
Node=BLADE202Node01,Server=server1

Provider  
Default messaging provider

\* Name  
AnyOldName

\* JNDI name  
jms/q

Description

---

**Connection**

Bus name  
Bus2

\* Queue name  
Bus2Queue

Delivery mode  
Application

Figure 15-7 Correct configuration for a queue on a foreign destination

### Incorrect configuration

Figure 15-8 on page 219 illustrates an incorrect configuration for JMS queue jms/q and destination Bus2Queue. The JMS queue is bound to a destination that does not exist on Bus1 because it only references a foreign destination.

**Queues** > **AnyOldName**

A JMS queue is used as a destination for point-to-point messaging. Use JMS queue destination administrative objects to messaging provider.

Configuration

---

**General Properties**

**Administration**

Scope  
Node=BLADE202Node01,Server=server1

Provider  
Default messaging provider

\* Name  
AnyOldName

\* JNDI name  
jms/q

Description

---

**Connection**

Bus name  
Bus1

\* Queue name  
other, please specify Bus2Queue

Delivery mode  
Application

Figure 15-8 Incorrect: JMS queue points to the destination on the local bus

When a JMS application attempts to produce messages on this JMS queue, the following error message is written to the application server log files.

**CWSIA0062E:** Failed to create a MessageProducer for queue://<Remote\_Bus\_Name+Destination\_Name>?busName=Local\_Bus\_Name at:

### 15.5.3 Configuring topic space mappings

Topic space mapping enables subscribers on a local topic space to receive messages that have been published to a remote topic space using the foreign bus link. The publisher and subscribers on both buses must both share the same topic name.

For more detailed information about topic space mappings, see *Configuring topic space mappings between service integration buses* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.doc/tasks/tjj0707\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.doc/tasks/tjj0707_.html)

In this example, assume that:

- ▶ Messages are published to a remote topic space called Bus2TopicSpace on bus2.
- ▶ Messages are subscribed to from a local topic space called Bus1TopicSpace on bus1.
- ▶ The topic name is Fruit.
- ▶ Messages are sent using JMS clients.

The mapping is defined in the local bus, bus1, as a routing property of the foreign bus, bus2, as shown in Figure 15-9.

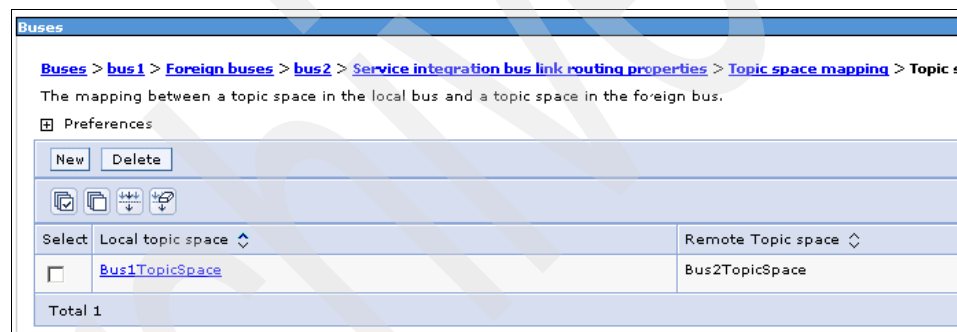


Figure 15-9 Bus1TopicSpace mapped to Bus2TopicSpace

The JMS topic definitions in Figure 15-10 on page 221 are provided for reference. On the left, you see the definitions for Bus1Fruit; on the right, Bus2Fruit.

Administration	Administration
Scope Node=BLADE202Node01,Server=server1	Scope Node=BLADE202Node01,Server=server1
Provider Default messaging provider	Provider Default messaging provider
* Name Bus1Fruit	* Name Bus2Fruit
* JNDI name jms/Bus1Fruit	* JNDI name jms/Bus2Fruit
Description	Description
Connection	Connection
Topic name Fruit	Topic name Fruit
Bus name bus1	Bus name bus2
* Topic space Bus1TopicSpace	* Topic space Bus2TopicSpace
JMS delivery mode Application	JMS delivery mode Application

Figure 15-10 Topic definitions on bus1 and bus2

Using the WebSphere Application Server administrative console, to determine whether messages have flowed across the foreign bus link:

1. Select **Servers** → **Application servers**.
2. Click the server name hosting bus2 (for this example, server2).
3. Click **Messaging engines**.
4. Click the messaging engine name to open the details page.
5. Click the **Runtime** tab.
6. Under the Remote message points heading, click **Remote publication points**.

This panel shown in Figure 15-11 on page 222 displays.

Remote Publication Points

Remote publication points that are producing messages to publication points on remote messaging engines.

References

Delete all messages

Identifier	Message engine	Current outbound messages	Outbound messages sent
<a href="#">Bus2TopicSpace</a>	BLADE202Node01.server1-bus1	0	10

Figure 15-11 Remote publication points for server2

The **Outbound messages sent** column reflects the number of messages that have flowed across the link.

## 15.5.4 Securing foreign bus environments

This topic builds on the WebSphere Application Server V6.1 Information Center task, *Securing access to a foreign bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/tasks/tjj0002S\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/tasks/tjj0002S_.html)

The following steps provide an example of securing a foreign bus configuration using point-to-point messaging.

The local bus is referred to as Bus1 and the remote (foreign) bus is Bus2. It assumes two user IDs (one for each bus) have been created,; Bus1User and Bus2User, respectively.

To secure a foreign bus:

1. Enable administrative security, and configure a user repository. In this example, both the local bus and the remote (foreign) bus use the same repository.

For more information about performing this task see *Administrative security* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/csec\\_global.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/csec_global.html)

2. Stop your WebSphere Application Server configuration.



3. Enable bus security for both Bus1 and Bus2 as documented in *Messaging security* at:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/concepts/cjr0420\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/concepts/cjr0420_.html)
4. Give Bus1User the bus connector role for Bus1. Using the administrative console:
  - a. Select **Service integration** → **Buses**.
  - b. Click on the link under the Security heading for Bus1.
  - c. Select **Users and groups in the bus connector role**.
  - d. Click **New**, and add user Bus1User.
5. Use the previous steps to give Bus2User the bus connector role for Bus2.
6. Create an authentication alias for the foreign bus link; for example, FBLAlias setup for user Bus1User:
  - a. Select **Service integration** → **Buses**.
  - b. Click on the link under the Security heading for Bus1.
  - c. Select **JAAS - J2C authentication data**.
  - d. Click **New** and add the alias.
7. Update the foreign bus definition for Bus2 on Bus1:
  - a. Select the authentication alias to be used to authenticate to Bus2.
  - b. Add the Target inbound transport chain and Bootstrap endpoints to use the secure chain and port, as shown in Figure 15-12 on page 224.

This procedure defines the link so it connects to Bus2 using a particular messaging engine and the secure inbound transport chain. It will bootstrap using the port and chain supplied.

Port 7287 is the SIB\_ENDPOINT\_SECURE\_ADDRESS port for the application server hosting the messaging engine on Bus2 (host blade202).

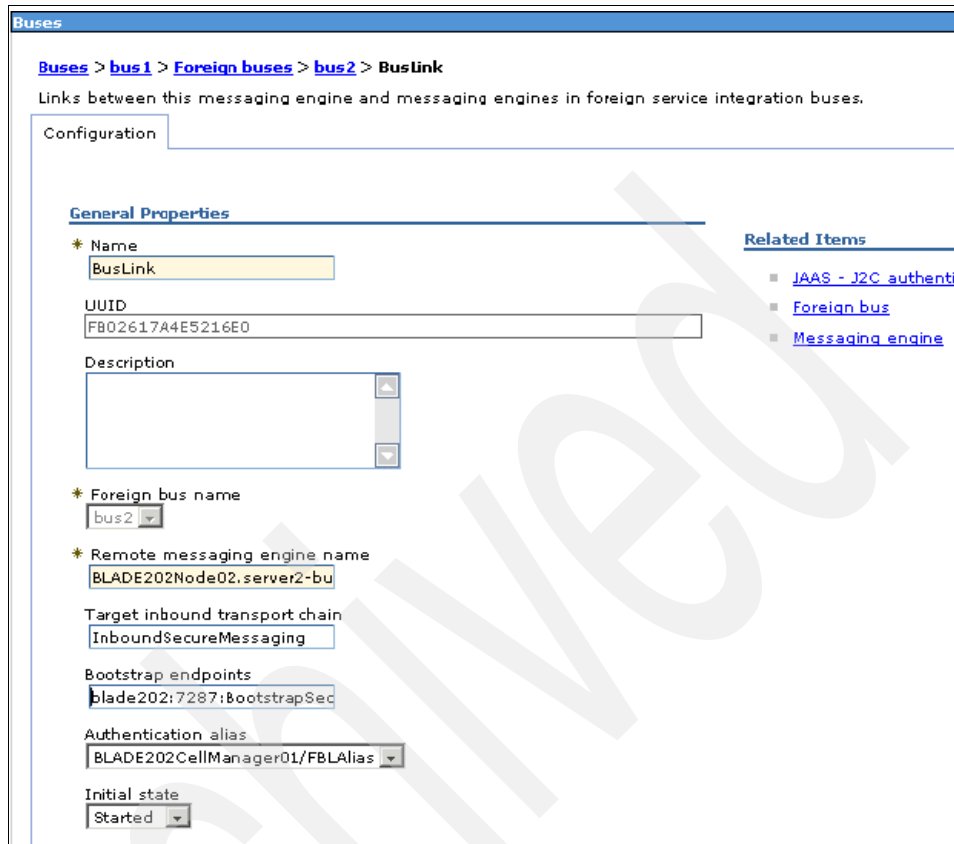


Figure 15-12 Securing the foreign bus link on Bus1

8. Update the foreign bus definition for Bus1 on Bus2:
  - a. Select the authentication alias to be used to authenticate to Bus1.
  - b. Add the target inbound transport chain and bootstrap endpoints to use the secure chain and port, as shown in Figure 15-13 on page 225.

This will define the link so it connects to Bus1 using a particular messaging engine and the secure inbound transport chain. It will bootstrap using the port and chain supplied.

Port 7286 is the SIB\_ENDPOINT\_SECURE\_ADDRESS port for the application server hosting the messaging engine on Bus1 (host blade202).

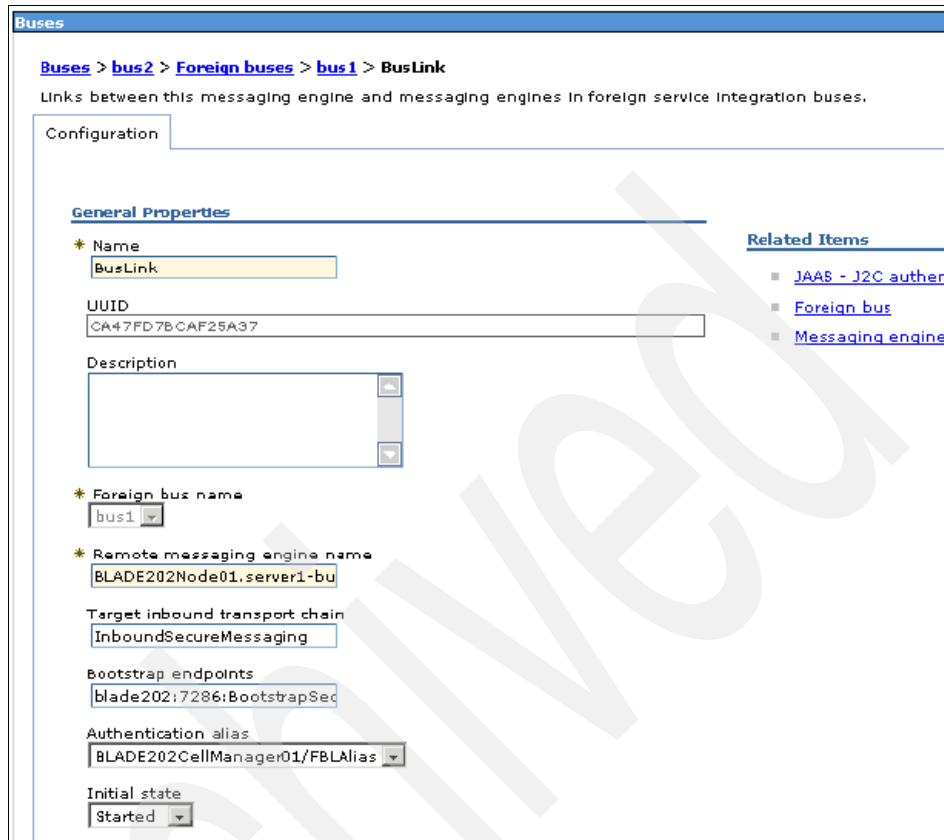


Figure 15-13 Securing the foreign bus link on Bus2

9. For Bus1, set the foreign bus, Bus2, routing properties to use the outbound user ID of Bus2User:
  - a. Select **Service integration** → **Buses**.
  - b. Click on the bus name, **Bus1**.
  - c. Select **Foreign buses**.
  - d. Click **Bus2**.
  - e. Select **Service integration bus link routing properties**, and set the properties as shown in Figure 15-14 on page 226.

**Buses**

[Buses](#) > [bus1](#) > [Foreign buses](#) > [bus2](#) > **Service integration bus link routing properties**

The routing properties for a service integration bus link to a foreign service integration bus.

Configuration

General Properties	Additional Properties
Name <input type="text" value="bus1:bus2"/>	<ul style="list-style-type: none"> <li><a href="#">Topic space mapping</a></li> </ul>
UUID <input type="text" value="EA35A36241A2A17E15638515"/>	
Inbound user ID <input type="text"/>	
Outbound user ID <input type="text" value="Bus2User"/>	

Figure 15-14 Service integration bus link routing properties

10. Using similar steps for Bus2, set the foreign bus, Bus1, routing properties to use the outbound user ID of Bus1User, as shown in Figure 15-15.

**Buses**

[Buses](#) > [bus2](#) > [Foreign buses](#) > [bus1](#) > **Service integration bus link routing properties**

The routing properties for a service integration bus link to a foreign service integration bus.

Configuration

General Properties	Additional Properties
Name <input type="text" value="bus1:bus2"/>	<ul style="list-style-type: none"> <li><a href="#">Topic space mapping</a></li> </ul>
UUID <input type="text" value="F74C5DA4C91F2A90E8C04DA7"/>	
Inbound user ID <input type="text"/>	
Outbound user ID <input type="text" value="Bus1User"/>	

Figure 15-15 Service integration bus link routing properties

11. Set the JMS queue connection factories to use the authentication alias. For example, see Figure 15-16 on page 227.

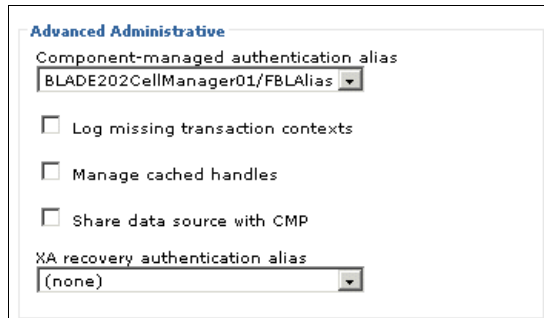


Figure 15-16 Queue connection factories → Default messaging provider

12. For applications running in a WebSphere Application Server client container, you might need to use the `clientconfig.bat` tool to add an authentication alias so the application might access the secured resources.

For more information, see *Starting the Application Client Resource Configuration Tool and opening an EAR file at:*

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/ucli\\_tstartacrct.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/ucli_tstartacrct.html)

13. Restart the application server, and check that the foreign bus links start.

At this point, if you attempted to put messages to the remote queue on Bus2 you would see an error message similar to the following, because you must also define the SIB access role types for the resources you are trying to use.

**CWSII0219W:** The bus bus1 denied the user Bus1User access to send a message to the foreign bus bus2.

In this example, the application sends messages to the remote queue Bus2Queue on a remote (foreign) bus, Bus2. Depending on the configuration you choose, you can use one of two ways to achieve this behavior:

- ▶ If there are no foreign or alias destinations configured for the destination on Bus2, Bus1User can send messages using the foreign bus link. Because the outbound user has been specified, you must give Bus1User authority to put (or send) messages to Bus2Queue. For this example, grant privileges using the wsadmin interface:

```
(Using Jacl) $AdminTask addUserToForeignBusRole {-bus bus1 -foreignBus bus2 -role sender -user Bus1User}
```

```
(Using Jython) AdminTask.addUserToForeignBusRole('[-bus bus1 -foreignBus bus2 -role Sender -user Bus1User]')
```

- ▶ If a foreign or alias destination has been defined, you must give Bus1User authority to send messages to the remote destination using this foreign destination. For this example, using the wsadmin interface to grant the privileges:

```
(Using Jacl) $AdminTask addUserToDestinationRole {-bus bus1 -type ForeignDestination -foreignBus bus2 -destination Bus2Queue -role sender -user Bus1User}
```

```
(Using Jython) AdminTask.addUserToDestinationRole('[-bus bus1 -type ForeignDestination -foreignBus bus2 -destination Bus2Queue -role sender -user Bus1User]')
```

After restarting all processes, the user is enabled to send messages across the foreign bus link with security enabled.

Where SSL chains are used, it is necessary to exchange SSL certificates. For detailed information about how to perform this task, see *Controlling which foreign buses can link to your bus* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjr0420\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjr0420_.html)



## Mediation problem determination

Various problems can occur during the mediation of messages within the messaging bus of the WebSphere Application Server. If you experience problems with the mediation of messages, use this chapter to diagnose the problem.

This chapter walks you through the process of debugging problems with the mediation of messages.

## 16.1 Introduction to mediation

Service integration bus destinations can be configured as mediated destinations. When this is done, a new mediation point is associated with the destination.

The mediation process is:

1. When a message is sent to a mediated destination it is added to the mediation point.
2. The messaging engine takes the message from the mediation point and passes it to a mediation.
3. After the mediation has completed, the message is, as directed by the mediation, processed in one of these ways:
  - Added to the destination
  - Deleted
  - Sent to another destination
  - Sent to the exception destination associated with the destination

This process happens without any information logged to the application server SystemOut log, unless the implementation of the mediation logs it.

For more information about message mediation see the WebSphere Application Server V6.1 Information Center topic *Mediations* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjp9999\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjp9999_.html)

## 16.2 Identify symptoms

Mediation problems discussed are categorized by their major symptom. The first step is to determine which of the following symptoms fit your problem:

- ▶ Messages are not being consumed by the application.
- ▶ Messages are being consumed, but are unmediated.
- ▶ Messages are mediated incorrectly.
- ▶ Messages are mediated, but slowly.



## 16.3 Messages are not being consumed by the application

There are two primary reasons that an application does not consume mediated messages:

- ▶ Messages are queued on the correct destination but are waiting to be mediated.
- ▶ Messages are being sent to the wrong destination.

### 16.3.1 Determine if messages are queued on the mediation point

First, determine if messages are queued on the mediation point waiting to be mediated. To examine the current message depth at the mediation point, using the administrative console:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Destinations**.
4. Click on the destination for the messages.
5. Click **Mediation Points**.
6. Click on the relevant mediation point (you might need to examine each of the mediation points in turn).
7. Click the **Runtime** tab.
8. Examine the message depth to see if it looks unusually large or is increasing faster than the messages are being handled.

If the message depth continually increases with no indication any messages are being processed, see 16.6, “Messages are waiting to be mediated” on page 233.

If the message depth both increases and decreases, but messages are arriving faster than they are being handled, see 16.8, “Messages are mediated, but slowly” on page 241.

### 16.3.2 Determine if messages are queued at the wrong destination

If no messages appear queued to the mediation point and they are not being consumed by the application, the messages might be at the wrong destination, because either the mediation put them on the wrong destination or the messages are at an exception destination.

If you have this situation, see 16.9, “Messages are arriving at the wrong destination” on page 243.

## 16.4 Messages are being consumed, but are unmediated

If messages are being consumed by an application, but the results of that message consumption indicate that:

- ▶ The required message *mediation is not being performed*, see 16.5, “Mediation is not performed” on page 232.
- ▶ The required message *mediation is being performed incorrectly*, see 16.7, “Messages are mediated incorrectly” on page 238.

If messages are being mediated correctly, but the *arrival rate of messages at that destination is lower than expected*, see 16.8, “Messages are mediated, but slowly” on page 241.

## 16.5 Mediation is not performed

If messages are arriving at the application unmediated:

- ▶ Verify the destination is mediated.
- ▶ Verify the message is being routed to the mediated destination.

### 16.5.1 Verify the destination is mediated

Use the administrative console to verify the configuration of the relevant destinations:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations**.  
This page lists the destinations defined on the bus and shows any mediations associated with the destination.

#### **Solution**

If no mediation is associated with the destination, you need to configure it. Select the destination and click **Mediate**.

## 16.5.2 Verify the message is being routed to the mediated destination

The use of *forward routing paths* is a common reason that a mediation is not performed. The forward routing path is a property of the message that can be used by the producing application to determine a route for the message. As a result, a message, once in the system, might be routed differently than expected. A mediation can be used to manipulate the forward and reverse routing paths of a message.

For more information about forward routing path capabilities see *Destination routing paths* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjo0005\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjo0005_.html)

### Solution

Ensure the correct destination is being mediated and that the routing path is correct for your configuration.

The mediated destination is usually the destination from which the application is expecting to consume messages. For a more complex application design that uses the forward routing path capabilities, you need to examine the application design to determine which destinations are mediated.

One option is to inject a message and trace it through the system. An alternative is to use the administrative console to stop the message points, starting them one at a time to watch the message manually traverse the path.

If you make changes to application server resources, you need to restart the application server for those changes to take effect.

## 16.6 Messages are waiting to be mediated

If messages are not being consumed by any application but appear to be waiting to be mediated, collect and analyze the diagnostic data to determine what is causing the messages to wait.

## 16.6.1 Verify the mediation point is operational

You can use the administrative console to verify that the mediation point is operational (that is, that messages at the mediation point will be passed to the mediation code to be mediated):

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Destinations**.
4. Click on the destination name.
5. Click **Mediation Points** in the Message points.

This page lists the mediation points of the destination and shows their current status, which should be **Started**.

If it is any other state, attempt to start it by selecting the mediation point and clicking **Start**.

There are several reasons the mediation point might be stopped when there are no problems:

- ▶ The mediation point was stopped by an operator and not restarted.
- ▶ The initial state has been configured as **Stopped**.
- ▶ The mediation point has been deleted and then redefined. The previous deletion has not yet completed. After the previous deletion has completed, the mediation point will automatically start when the server has started, assuming that the configured initial state for the new mediation point is **Started**.

If the mediation point was stopped and you were able to start it, verify the messages are flowing to the application.

If the mediation point was stopped and you are not sure why it was stopped, or if it fails to start, continue to 16.6.2, “Verify the mediation application is started” on page 234 to collect diagnostics.

If the mediation point was started and this does not appear to be the reason messages are waiting, continue to 16.6.2, “Verify the mediation application is started” on page 234 to collect diagnostics.

## 16.6.2 Verify the mediation application is started

To view the status of the application select **Applications** → **Enterprise applications** in the administrative console. If the application is not started, start it by selecting the box to the left of the application and clicking **Start**.

### 16.6.3 Analyze SystemOut

If the previous system checks have not resolved the problem, you need to analyze the SystemOut log for the bus member to which the queue and mediation points have been assigned. If the bus member is a cluster, you might need to repeat these steps for each server in the cluster.

If the mediation point is not in the Started state, messages are not being processed. You should examine the SystemOut log file for the following messages:

- ▶ **CWSIP0771I:** Administrator action  
An administrator has manually stopped the mediation point. Start the mediation point manually.
- ▶ **CWSIP0773I:** Using configured initial state  
The mediation point has been configured with an initial state of Stopped. Either start the mediation point manually, or change the initial state of the mediation point to be Started and restart the messaging engine or application server.
- ▶ **CWSIP0775I:** Waiting for the deletion of another mediation point on this destination.  
The mediation point has been deleted and then redefined. The previous deletion has not yet completed. After the previous deletion has completed the mediation point will automatically start when the server has started, assuming that the configured initial state for the new mediation point is Started.
- ▶ **CWSIZ0002E:** The mediation named *Mediation\_Name* that is attached to destination *Destination\_Name* is defined to use mediation handler list *Mediation\_Handler\_List\_Name*. However this handler list does not exist.  
See 16.6.4, “Mediation is not configured correctly” on page 237.
- ▶ **CWSIZ0011E:** The mediation named *Mediation\_Name* attached to destination *Destination\_Name* could not process the message *Message\_ID* because the application *Mediation\_Application\_Name* has not started.  
Start the application and the mediation point will automatically start to process messages.
- ▶ **CWSIZ0020E:** The mediation authentication alias for the bus called *Bus\_Name* could not be located.  
or

**CWSIZ0021E:** The mediation authentication alias mediator for the bus called *Bus\_Name* could not be resolved.

or

**CWSIZ0045E:** The mediation authentication alias mediator for the bus called *Bus\_Name* could not be authenticated.

See 16.6.5, “Invalid mediation authentication alias” on page 237.

- ▶ **CWSIZ0056E:** The mediation named *Mediation\_Name* attached to destination *Destination\_Name* could not process the message *Message\_ID* because the next destination *Next\_Destination* cannot accept the message due to exception `com.ibm.ws.sib.processor.exceptions.SIMPLimitExceededException`.

The message, after the mediation code completed, has a forward routing path. However the message could not be sent along that forward routing path. The mediated message was not delivered to its next destination due to problems with the next destination.

See Chapter 7, “Messaging in a multiple messaging engine environment” on page 109 for more problem determination guidance.

- ▶ **CWSIZ0057E:** The mediation named *Mediation\_Name* attached to destination *Destination\_Name* could not process the message *Message\_ID* because the message cannot be made available to consumers due to exception ...

The mediation has completed successfully, but when the system attempted to make the message available to the consumers it was unable to do so. The exception information should provide more information about the problem. Resolve any problems that might exist with the next destination and restart the mediation.

- ▶ **CWSIZ0058E:** The mediation named *Mediation\_Name* attached to destination *Destination\_Name* could not process the message *Message\_ID* because the connection to the messaging engine has been lost as reported by exception `com.ibm.ws.sib.jfapchannel.JfapConnectionBrokenException`.

The mediation has been configured to use a mediation execution point that is remote from the queue point of the destination. The mediation has completed successfully; however, when the system attempted to return the message to the messaging engine, the system reported that the connection was no longer available.

The most common problem is that the messaging engine is unavailable or is stopping. Validate that all messaging engines are in the correct state. Resolve the problem and restart the mediation.

- ▶ **CWSIZ0059E:** The mediation named *Mediation\_Name* that is attached to destination *Destination\_Name* is defined to use mediation handler list *Mediation\_List\_Name*. However no handler lists exist.

See 16.6.4, “Mediation is not configured correctly” on page 237.

If none of these symptoms apply, but it appears that the mediation is not being performed, the mediation might be running slowly. To investigate this, see 16.8, “Messages are mediated, but slowly” on page 241.

## 16.6.4 Mediation is not configured correctly

Use the administrative console to verify that the mediation has been configured correctly:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Mediations** in the Destination resources.
4. Click on the mediation name.

Pay specific attention to the mediation handler list name. This case sensitive name must match exactly the handler list name that was defined when the mediation was developed. When the handler list cannot be found, but other handler lists can be located, the system generates a **CWSIZ0052I** message which is placed in the logs for the relevant server. You can use this message to verify the spelling of the handler list name.

An example of this message is:

**CWSIZ0052I:** Mediation handler lists defined in the server are [SimpleMediationHandler].

If the mediation appears to be correctly configured but the mediation is not being performed, see 16.5, “Mediation is not performed” on page 232.

## 16.6.5 Invalid mediation authentication alias

Use the administrative console to verify that the mediation authentication alias has been configured correctly:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Security** in the Additional Properties.

Check that the correct authentication alias has been selected and that the user ID and password for that authentication alias are correct.

### 16.6.6 Validate the solution

If you made changes to the application server resources, restart the application server so that the changes take effect. To validate the solution to your mediation problem, send a test message to the system and verifying that it was correctly mediated.

## 16.7 Messages are mediated incorrectly

If messages are being consumed by an application but the results of that message consumption indicate that the required message mediation is being incorrectly performed, you need to trace the message mediation to determine if the mediation code is operating as expected.

### 16.7.1 Collect diagnostics

To verify that the mediation code is operating as expected, take a trace to determine how the mediation modifies the message and what action is taken with the message after the mediation has completed.

The trace must be for the server of the bus member to which the queue and mediation points have been assigned. If the bus member is a cluster, you might need to repeat these steps for each server in the cluster.

Using the administrative console:

1. Select **Troubleshooting** → **Logs and Traces** in the navigation bar.
2. Click on the name of the server where the mediation is running.
3. Click **Change Log Details Levels**.
4. Select the **Runtime** tab.
5. Add `SIBMessageTraceContentsMediations=all` to the Groups in the large text box. (Use a colon (:)) to separate clauses in this box.)
6. Click **Apply**.

Recreate the problem; then, turn off the trace by resetting the trace string.



## 16.7.2 Analyze the trace

By default, the trace is stored in `${SERVER_LOG_ROOT}/trace.log`.

The trace log contains records similar to Example 16-1.

**Note:** The timestamps in the example have been replaced with ellipsis (...).

### *Example 16-1 Mediation trace*

---

```
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] CWSIZ1000I: A message
with ID ID:c0aa5975e6450404189fb1e1110a134f00000000000000001 and System
Message ID 56F8B38EDBCBB8E9_500003 has been delivered to mediation
Modifying attached to destination Modify.
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] Discriminator
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] ForwardRoutingPath
= []
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] ReverseRoutingPath
= []
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] Priority
= 4
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] Reliability
= ReliablePersistent
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] TimeToLive
= 0
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] RemainingTimeToLive
= -1
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] ReplyDiscriminator
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] ReplyPriority
= 4
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] ReplyReliability
= None
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] ReplyTimeToLive
= 0
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] RedeliveredCount
= 0
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] Userid
=
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMSType
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMSXAppID
= Service Integration Bus
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMSXDeliveryCount
= 1
```

```

[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMSXGroupID
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMSXGroupSeq
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Format
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Feedback
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_PutApplType
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_Report_Exception = null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_Report_Expiration = null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Report_COA
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Report_COD
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Report_PAN
= false
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Report_NAN
= false
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_Report_Pass_Msg_ID = false
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_Report_Pass_Correl_ID = false
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_Report_Discard_Msg = false
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_Last_Msg_In_Group = null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_PutDate
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_PutTime
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Encoding
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_Character_Set
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] JMS_IBM_ExceptionReason
= null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_ExceptionMessage = null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_ExceptionTimestamp = null

```

```

[...] 00000034 MediationMess 3 [MyBus:s1-MyBus]
JMS_IBM_ExceptionProblemDestination = null
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] Format = JMS:text
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] (DataGraph) {
(DataObject: JmsTextBody) {
    data=(DataObject: DataType1) {
        value=(Data)'test'
    }
}
}
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] CWSIZ1001I: A message
with ID ID:c0aa5975e6450404189fb1e1110a134f00000000000000001 and System
Message ID 56F8B38EDBCBB8E9_500003 has been sent on the forward routing
path [] by mediation Modifying attached to destination Modify.
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] TimeToLive
= 10000
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] RemainingTimeToLive
= 10000
[...] 00000034 MediationMess 3 [MyBus:s1-MyBus] level = gold

```

---

The trace log shows the contents of the message before the mediation executes, the action that was taken after the mediation completed, and, if the message was not deleted, what the mediation changed in the message.

### Solution

Review the entries to ensure that the actions taken are what you expect. If not, correct and redeploy the mediation application.

## 16.7.3 Validate the solution

To validate the solution to your mediation problem, send a test message to the system and verify that it was correctly mediated.

## 16.8 Messages are mediated, but slowly

Messages are being consumed by an application, but the arrival rate of messages at that destination is lower than expected.

### 16.8.1 Configure for performance

A mediation will be a performance bottleneck if the mediation is unable to process messages as fast as the messages are delivered to the mediation point.

When this bottleneck occurs, the message depth at the mediation point will steadily increase as messages are queued waiting for the mediation to run.

If the mediation point seems to be a bottleneck for traffic, check the configuration to ensure you have taken advantage of performance features.

### **Allow concurrent message mediation**

If the mediation was designed so that different instances of the mediation can be working on different messages simultaneously (and message ordering is not a requirement), the mediation can be configured to permit this behavior. To verify that the mediation has been configured to permit concurrent message mediation, in administrative console:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Mediations** in the Destination Resources.
4. Click on the relevant mediation.
5. Ensure that the **Allow concurrent mediation** box is selected to permit concurrent mediation.

### **Increase thread pool size**

If concurrent mediation is permitted and the mediation is still a bottleneck but there is spare capacity on the server where the mediation is running, you can increase the size of the mediation thread pool to permit more concurrent mediations to run. To examine and modify the mediation thread pool in the administrative console:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Messaging Engines** in Topology.
4. Click on the relevant messaging engine. You might need to examine each messaging engine in turn.
5. Click **Mediation thread pool** in the Additional properties.

## **16.8.2 Validate the solution**

Restart the server and send test messages to the system. Verify that these are correctly mediated with improved throughput.

## 16.9 Messages are arriving at the wrong destination

If messages arrive at the wrong destination, you need to investigate why the messages arrived at that destination.

### 16.9.1 Examine the messages

To find messages that might be arriving at the wrong destination:

1. Stop any applications that might be consuming the messages.
2. Send a test message.
3. Use the administrative console to find and examine the message.
4. Select **Service Integration** → **Buses** in the navigation bar and select the bus.
5. Click **Destinations**. For each destination that could have the message:
  - a. Click on the destination name.
  - b. Click **Queue Points** and then click the queue point name.
  - c. Click the Runtime tab.
  - d. Click **Messages**. If the message is there, click on the message.

The **exception destination reason** field of the message is either blank or it contains a message:

- If the exception destination reason field is *blank*, the mediation (or sending application) has explicitly sent the message to the wrong destination.
- If the **exception destination reason** field is *not empty*, the message has been sent to the destination because a problem was encountered, and this destination is defined as the exception destination.

Examine the reason and see Exception destination reasons to determine why the message was rerouted.

#### Exception destination reasons

The reason codes provide information about the problem; either something is wrong with the mediation code or something is wrong with the message itself. Use this information to decide whether you need to examine the mediation logic or message, and what to look for. If the problem is with the message, trace the mediation to see what is happening. For information about tracing the mediation, see 16.7, “Messages are mediated incorrectly” on page 238:

- ▶ **CWSIK0101W:** The message *Message\_ID* has been re-routed to the exception destination *Exception\_Destination\_Name* because the mediation *Mediation\_Name* at destination *Destination\_Name* caused the exception `java.lang.NullPointerException`.

The mediation sent the message to the exception destination.

- ▶ **CWSIK0102E:** The message *Message\_ID* cannot progress further along its forward routing path after being mediated by mediation *Mediation\_Name* at destination *Destination\_Name* because the message does not conform to the message format `JMS:text`.

The mediation modified the message in such a way that the contents of the message no longer matches the schema of the message.

- ▶ **CWSIK0103E:** The message *Message\_ID* cannot progress further along its forward routing path after being mediated by mediation *Mediation\_Name* at destination *Destination\_Name* because the message is not well formed.

The mediation modified the message in such a way that that the message can no longer be serialized.

## 16.10 Search online support

If you are sure the problem is in the mediation process, there are tasks that you can do before contacting IBM support. First, review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on this IBM support page:

<http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&q=mediation&Go.x=0&Go.y=0>



## Default messaging provider security

This chapter discusses possible configuration problems that can occur when you configure security for the default messaging provider.

## 17.1 Introduction

The goal of messaging security is to allow trusted clients to perform messaging operations, and to prevent non-trusted clients from performing similar messaging operations. Various problems can occur when attempting to use messaging that are related to the security policy preventing access. These problems generally manifest themselves in one of two ways:

- ▶ An application cannot connect to the bus.
- ▶ An application, which is connected to the bus, cannot send or receive messages.

For more information about messaging security, see these additional resources:

- ▶ *Service integration bus security* at:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjr9999\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjr9999_.html)
- ▶ *WebSphere MQ security* at:  
[http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/fa12730\\_.htm](http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/fa12730_.htm)

## 17.2 Identify symptoms

The potential problem areas related to messaging security might be classified as:

- ▶ Authentication problems connecting to the bus

When connecting to a bus the first thing that occurs when security is enabled is to identify the client. This identity is then used for subsequent authorization checks. The identification process, commonly known as *authentication*, involves taking an identity (or user name) and a claim (or password) and checking that the claim matches the identity. If they do, the identity is confirmed. Authentication problems are typically caused by an invalid identity claim (for example, the password has expired or does not match the password stored in the user repository).

For information about configuring the cell to contact the user repository, see the WebSphere Application Server Information Center topic, *Authenticating users* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec\\_authusers.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_authusers.html)



- ▶ Authorization problems connecting to the bus

*Authorization* is the process of determining if an identified user is allowed to access a resource. An authorization failure does not necessarily indicate a problem (for example, when a user is not supposed to have access).

When the application fails authorization for the bus, it is prevented from accessing the messaging resources.

- ▶ Authorization problems accessing a destination

When the application fails authorization for a destination resource, the requesting application is not allowed access to the destination.

- ▶ SSL problems connecting to the bus

In WebSphere Application Server V6.1, a change was made to the use of certificates for the application server. Prior to V6.1, each application server came with the same certificate, by default.

When a profile is created in V6.1, a set of keys, specific to that server, is created. When a node is federated, the federation process ensures that trust is established within the cell. However, trust is not automatically established for clients or for cross cell communications.

## 17.3 Analyze diagnostics

Security problems that occur with the default messaging provider usually produce a message in the SystemOut log for the application server that is running the messaging engine, and an exception in the application log.

Look at both the top level `JMSecurityException` or `JMSEException` as well as any linked exceptions. The linked exceptions provide additional documentation and debug aids. They are subject to change without notice so you should not rely on them for program flow.

The following list contains specific errors addressed by this chapter. If you find an error message other than those listed and the information included in the message is not enough to determine the root cause of the message, see Chapter 18, “The next step” on page 261:

- ▶ This message in the SystemOut for the messaging engine indicates an authentication failure:

**CWSII0205W:** The bus *Bus\_name* could not authenticate the user *User*

The client JMS application receives a `JMSSecurityException` with this message:

**CWSIA0004E:** The authentication for the supplied user name *User* and the associated password was not successful

If you find these indicators, see 17.4, “Authentication problems connecting to the bus” on page 250.

- ▶ Another example of an authentication failure is this message:

**CWSII0212W:** The bus *Bus\_name* denied an anonymous user access to the bus.

This message indicates that a client has attempted to connect without providing any credentials.

If you find this message, see 17.4, “Authentication problems connecting to the bus” on page 250.

- ▶ This message in the SystemOut for the messaging engine indicates an authorization failure for the bus:

**CWSII0211W:** The bus *Bus\_name* denied the user *User* access to the bus.

The client JMS application receives a `JMSSecurityException` with this message:

**CWSIA0006E:** The authorization for the supplied user name was not successful.

If you find these indicators, see 17.5, “Authorization problems connecting to the bus” on page 255.

- ▶ Authorization problems accessing a destination produce messages in the range of CWSII0213-CWSII0240 and CWSII0242-CWSII0259, inclusive. The messages you see depend on what the client application was attempting to do:

- A failure to *send* a message results in these messages:

In SystemOut for the messaging engine server:

**CWSII0213W:** The bus *Bus\_Name* denied the user *User* access to send messages to the destination *Destination\_Name*.

The client JMS application receives a `JMSSecurityException`, which contains this message:

**CWSIA00069E:** The user does not have authorization to carry out this operation. See the linked exception for details.

The linked exception contains:

**CWSIK0018E:** Send access to destination *Destination\_Name* was denied for user with subject *User\_Name*.

- A failure to *receive* a message results in these messages:

In SystemOut for the messaging engine server:

**CWSII0214W:** The bus *Bus\_Name* denied the user *User\_Name* access to receive messages from the destination *Destination\_Name*.

The client JMS application receives a JMSSecurityException with this message:

**CWSIA0090E:** The user does not have authorization to carry out this operation. See the linked exception for details.

The linked exception contains:

**CWSIP0309E:** Receive access from destination *Destination\_Name* was denied for user with subject *User\_Name*.

If you see these indications, see 17.6, “Authorization problems accessing a destination” on page 258.

- ▶ If the failure occurs in an application server, this message is written to the application sever log files:

**CWPKI0022E:** SSL HANDSHAKE FAILURE: A signer with SubjectDN "CN=pdServer.itso.ibm.com, O=IBM, C=US" was sent from target host:port "pdServer.itso.ibm.com:7287". The signer may need to be added to local trust store "/opt/HDR61/profiles/jestred\_na\_1/config/cells/pdCell/trust.p12" located in SSL configuration alias "NodeDefaultSSLSettings" loaded from SSL configuration file "security.xml". The extended error message from the SSL handshake exception is: "No trusted certificate found".

In the client, a JMSEException is seen. More diagnostic information is contained in the linked exceptions. If the problem is a trust store issue, you see an SSLHandshakeException with the message:

No trusted certificate found

Problems related to the use of SSL are outside the scope of this problem determination guide. However, problems arising from trust store are common, and there are a number of useful WebSphere Information Center documents.

For information about configuring trust stores, see:

- *Retrieving signers from a remote SSL port* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec\\_sslretrievesignersport.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_sslretrievesignersport.html)

- *Adding a signer certificate to a keystore* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec\\_ssladdsignercert.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_ssladdsignercert.html)

- *Exchanging signer Certificates at:*

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec\\_ssl exchangesigncerts.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_ssl exchangesigncerts.html)

For information about the **retrieveSigners** command used by the client, see:

*Retrieving signers using retrieveSigners utility at the client at:*

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec\\_sslretrieveclient.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_sslretrieveclient.html)

## 17.4 Authentication problems connecting to the bus

The two primary reasons for an authentication failure to the bus are:

- ▶ The user ID does not exist.
- ▶ The password specified is not correct.

The resolution to the problem depends on the following factors:

- ▶ Is the JMS application running in an application server or a client?
- ▶ Is the JMS application making use of container or application provided authentication?

### 17.4.1 Solution for an application in an application server

To resolve this problem depends on whether container or application authentication is being used. To determine which type of authentication is being used, you need to look at the application configuration in the administration console:

1. Select **Applications**.
2. Select **Manage Applications**.
3. Click on the application name that is experiencing the failure.

Under the Resources heading, look for a **Resource references** link. If this link is not available, then your application does not have any resource references defined, which means one of two things:

- The application selected does not use JMS.

If the application does not use JMS, identify the application that needs correction and start over.

- The application selected does not use resource references.

If it does use JMS, then it is doing a direct JNDI lookup. In this case see “Resolving the problem when using application authentication” on page 252.

Click the **Resource references** link and you see a page similar to that shown in Figure 17-1. The login configuration for each resource reference is shown in the table in a red box. In this example, the authentication mode used is Container. The word Container is replaced with Application if application-based authentication is to be used.

Specify authentication method:

None  
 Use default method (many-to-one mapping)  
 Authentication data entry  
  
 Use custom login configuration  
 Application login configuration

Apply

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
<input type="checkbox"/>	Sample JMS Application	abvt.sib.ABVTMDB	abvt_sib_ejb.jar,META-INF/ejb-jar.xml	jms/mdb/ABVT_CF	ABVT_CF Browse...	Resource authorization: Container Authentication method: None

OK Cancel

Figure 17-1 Manage resource references

### Resolving the problem when using container authentication

Figure 17-2 on page 252 shows an example where no authentication alias has been configured.

Specify authentication method:

None  
 Use default method (many-to-one mapping)  
 Authentication data entry  
  
 Use custom login configuration  
 Application login configuration

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
<input type="checkbox"/>	Sample JMS Application	abvt.sib.ABVTMDB	abvt_sib_ejb.jar,META-INF/ejb-jar.xml	jms/mdb/ABVT_CF	<input type="text" value="ABVT_CF"/> <input type="button" value="Browse..."/>	Resource authorization: Container Authentication method: None

Figure 17-2 Configuring Container-managed authentication

To configure the authentication alias to be used:

1. Select the check box for the resource reference.
2. Select **Use default method (many-to-one mapping)** and select an authentication alias from the pull-down list.
3. Click **Apply** to make the change to the resource reference.
4. Click **OK**.

### Resolving the problem when using application authentication

Application authentication is designed for cases in which the application is providing the credential. The application typically provides the credentials in the code. Often, these credentials are thought of as hard coded, although the application might obtain them through some application specific mechanism. If this is the case, consult with the application author to resolve this problem.

Alternatively, WebSphere Application Server provides an additional convenience mechanism for providing credentials called the *component-managed authentication alias*. It is specified on the JMS connection factory and is only used if application authentication is specified and the zero arguments variant of `createConnection()` is used.

It is not good practice to rely on a component-managed authentication alias. The authentication alias is shared by all users of the connection factory who are using application authentication (and direct JNDI lookups).

If you want to supply the credentials on the JMS connection factory, use container-managed authentication instead.

If you are relying on the component-managed authentication alias, you can configure it on the connection factory definitions. You can work with one of these definitions on the menu by expanding **Resources** → **JMS** and selecting one of the connection factories: queue or topic. If the connection factory is for the default messaging provider, look to the bottom of the connection factory definition page at a section titled **Component-managed authentication alias**, and select an authentication alias from the drop-down box; for example, see Figure 17-3.

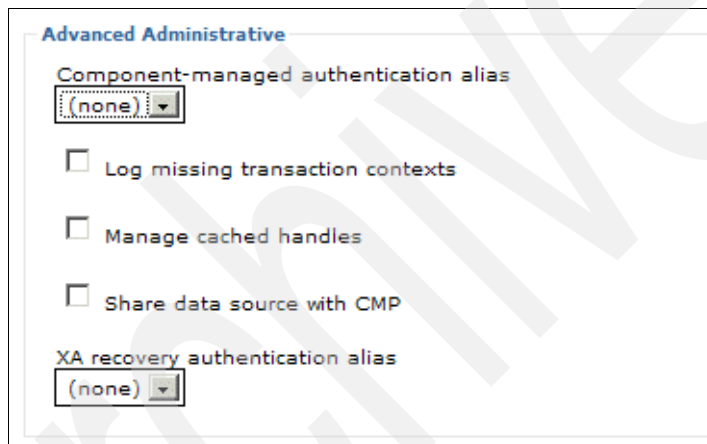
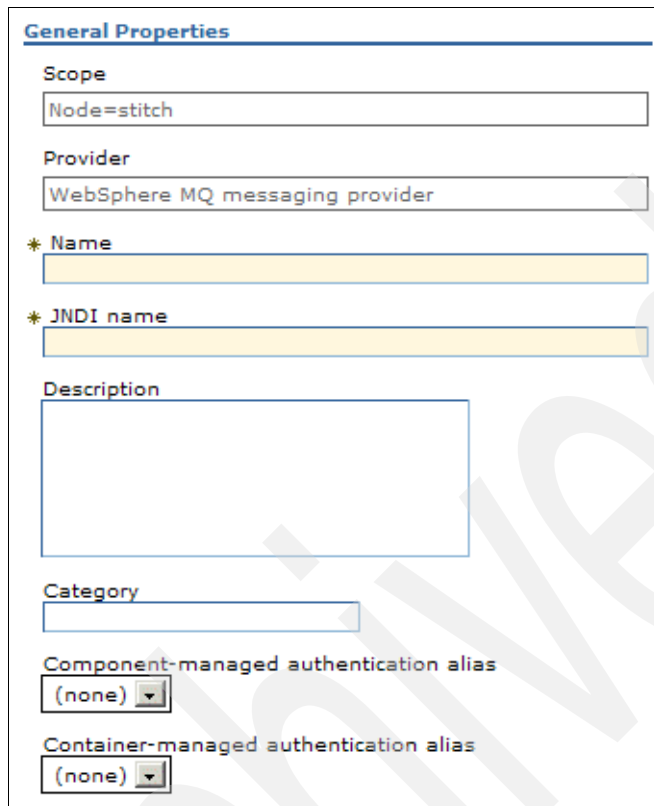


Figure 17-3 Default messaging provider component-managed alias

If the connection factory is for the WebSphere MQ messaging provider, use the configuration setting near the top of the properties titled **Component-managed authentication alias**; for example, see Figure 17-4 on page 254.



**General Properties**

Scope

Provider

\* Name

\* JNDI name

Description

Category

Component-managed authentication alias

Container-managed authentication alias

Figure 17-4 WebSphere MQ messaging provider component-managed alias

**Note:** The use of component-managed authentication aliases is strongly discouraged. Using it is a sign that the application performing the lookup should be using a resource-reference with container authentication and an application update should be performed instead.

### Verifying the authentication alias user ID and password

If an authentication alias has been configured (and it is the correct one), you next need to check the configuration of the authentication alias:

1. Select **Security** → **Secure administration, applications, and infrastructure**.
2. Java Authentication and Authorization Service, and select **J2C authentication data**. This page shows a list of authentication aliases.
3. Select the alias that was being used. You see a window similar to Figure 17-5 on page 255.



The screenshot shows a dialog box titled "General Properties". It contains four input fields, each preceded by an asterisk (\*):  
- "Alias" with the text "pdAlias"  
- "User ID" with the text "pdUser"  
- "Password" with the text "\*\*\*\*\*"  
- "Description" which is empty.  
At the bottom of the dialog are four buttons: "Apply", "OK", "Reset", and "Cancel".

Figure 17-5 Configure authentication alias

4. Correct the user name and password, click **OK** and save the configuration.
5. Restart the application server.

The correct credentials will be used to connect to the messaging provider.

## 17.4.2 Resolving the problem in a client application

The credentials for a client application are typically provided by calling the `createConnection()` method with a user ID and password. How these are provided is application specific. Contact the application author for further guidance to diagnose this problem.

## 17.4.3 Validate solution

To validate the solution to your security problem rerun the application. If you made changes to the WebSphere Application Server JMS or JNDI resources, you need to restart the application server processes for changes to take effect.

## 17.5 Authorization problems connecting to the bus

The primary reason for an authorization failure when connecting to the bus is that the user, or a group that includes the user, does not have the bus connector role for that bus.

You can configure a user to have the bus connector role using the administrative console or the `wsadmin` scripting tool.

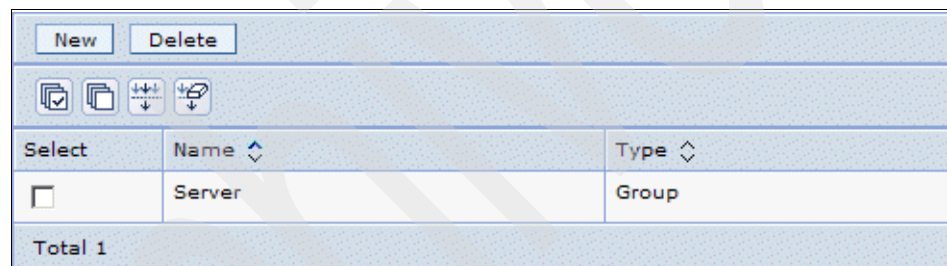
Best practice is to assign roles using groups rather than users directly.

## Resolving the problem using the administrative console

From the administrative console:

1. Select **Service Integration** → **Buses**.
2. Click on the bus name to which the client requires access.
3. Select **Security**.
4. Select **Users and groups in the bus connector role**.

This page lists the users and groups in the bus connector role. By default, this list only contains a single group, the Server group. This convention gives application servers authority to connect to the bus, but not the individual applications. The default is shown in Figure 17-6.



Select	Name	Type
<input type="checkbox"/>	Server	Group

Total 1

Figure 17-6 List of users and groups in bus connector role

You can modify this list by either deleting or adding users or groups.

5. To give a user the bus connector role, click **New**.

A new panel displays that lets you add users and groups (along with the special Everyone, AllAuthenticated, and Server groups) to the bus connector role.

For illustration purposes in this example, user `pdUser` is granted direct access to the bus.

The settings are shown in Figure 17-7 on page 257.

**General Properties**

**Bus Connector Role**

Group name  
 User name   
 Server - Allow servers to connect to the bus  
 All Authenticated - Allow all authenticated users to connect to the bus  
 Everyone - Allow unauthenticated users to connect to the bus

Figure 17-7 Adding a user to the bus connector role

- Click **OK** to add the user to the bus connector role. The result is shown in Figure 17-8.

Select	Name	Type
<input type="checkbox"/>	Server	Group
<input type="checkbox"/>	pdUser	User
Total 2		

Figure 17-8 List of users and groups in the bus connector role

- Save and synchronize the configuration. You do not need to restart the server restart.

### Resolving the problem using the wsadmin scripting tool

Alternatively, you can use the `wsadmin` scripting tool to configure a user for the bus connector role. There are an extensive set of commands for configuring bus security.

For information about commands for configuring the bus connector role, see the WebSphere Application Server version 6.1 Information Center, *Administering bus connector roles* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjr0100\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjr0100_.html)

To modify the bus connector role, load the **wsadmin** tool and enter:

```
(Using Jacl) $AdminTask addUserToBusConnectorRole {-bus pdBus -user pdUser}
```

```
(Using Jython) AdminTask.addUserToBusConnectorRole(['-bus pdBus -user pdUser'])
```

## 17.6 Authorization problems accessing a destination

The primary reason for an authorization failure on a destination is that the user is not in the relevant role for the destination being accessed. A user is in a role if he or she, or a group to which the user belongs, has been configured with that role.

There are several roles that allow types of access. These are shown in Table 17-1.

*Table 17-1 Messaging roles and description*

Role	Description
Bus connector	Role required to permit the user to connect to the messaging bus.
Creator	Role required to permit the user to create temporary destinations within the name space.
Sender	Role required to permit the user to send a message to a resource (e.g. a Queue, a Foreign bus)
Receiver	Role required to permit the user to receive a message from a resource, for example, a queue or topic.
Browser	Role required to permit the user to browse a message on a resource for example, a queue or topic.

Table 17-2 on page 259 shows which resources the roles are applicable to. The resources shown are not destinations; they are resource types used in the **wsadmin** commands. The Queue resource type is also used for administering roles for Port, Web service, and temporary destination prefixes.

Table 17-2 Which roles are applicable to what resources

	Local Bus	Foreign Bus	Queue	Topic Space	Alias	Foreign Dest	Default	Topic Space Root	Topic
Bus connector	X								
Sender		X	X	X	X	X	X	X	X
Receiver			X	X	X		X	X	X
Browser			X		X		X		
Creator			X				X		

### Resolving the problem using the wsadmin scripting tool

The *wsadmin* tool is the only way to administer destination security roles. For detailed instructions on the commands and how they are used, see the Information Center topic *Administering authorization permissions* at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjr0380\\_.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/tasks/tjr0380_.html)

Best practice is to assign roles using groups rather than users directly. However, for illustration purposes, in the following examples, the user `pdUser` is granted access to the bus.

To resolve a problem *sending* messages, use this command:

```
$AdminTask addUserToDestinationRole {-type Queue -bus pdBus
                                     -destination pdQueue -role Sender -user pdUser}
```

To resolve the problem *receiving* messages, use this command:

```
(Using Jacl) $AdminTask addUserToDestinationRole {-type Queue -bus pdBus
                                                  -destination pdQueue -role Receiver -user pdUser}
```

```
(Using Jython) AdminTask.addUserToDestinationRole('[-type Queue -bus
pdBus -destination pdQueue -role Receiver -group pdPUser]')
```

If using groups, the commands are similar. For example:

```
(Using Jacl) $AdminTask addGroupToDestinationRole {-type Queue -bus pdBus  
-destination pdQueue -role Receiver -group pdPeople}
```

```
(Using Jython) AdminTask.addGroupToDestinationRole('[-type Queue -bus  
pdBus -destination pdQueue -role Receiver -group pdPeople]')
```

### 17.6.1 Validate the solution

To validate the solution to your security problem, rerun the application. If you made changes to the WebSphere Application Server JMS or JNDI resources, you need to restart the application server processes for changes to take effect.



## The next step

This chapter contains information about online resources that you will find useful in diagnosing problems and includes links to the MustGather information that you will need before contacting IBM technical support.

## 18.1 Online resources

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other things that can go wrong. If you are still experiencing a problem, you can use the following resources to help debug the problem further:

- ▶ Further explanation and description of appropriate user actions can be found by searching for each error message code in the WebSphere Application Server, version 6.1 Information Center.

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>

### 18.1.1 WebSphere MQ resources

If you were unable to resolve the problem using the instructions above then you can use the following resources to help debug the problem further.

- ▶ The WebSphere MQ library can be found at:

<http://www-306.ibm.com/software/integration/wmq/library/>

- The WebSphere MQ System Administration Guide contains information about examining the WebSphere MQ error logs.
- the WebSphere MQ Messages Reference contains a detailed description of WebSphere MQ reason codes and suggested actions

- ▶ WebSphere MQ reason codes can be found at:

<http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/amqzao/amqzao0m.htm>

## 18.2 Contact IBM

Before contacting IBM technical support you should examine the relevant MustGather documents and collect the appropriate documentation for your problem.

For problems using the *default messaging provider* see:

- ▶ Mustgather: Service Integration Technology

<http://www-1.ibm.com/support/docview.wss?uid=swg21266769>

For problems using the WebSphere MQ JMS provider see the following:



- ▶ MustGather: Documentation required by WebSphere MQ - Java and JMS problems

<http://www-1.ibm.com/support/docview.wss?uid=swg21176943>

Archived

Archived

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 266. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304
- ▶ *WebSphere for z/OS Problem Determination Means and Tools*, REDP-6880
- ▶ *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308

## Online resources

These Web sites are also relevant as further information sources:

- ▶ WebSphere Application Server V6.1 Information Center  
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>
- ▶ IBM Information Management Software for z/OS Solutions Information Center  
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>
- ▶ *Configure a Service Integration Bus in a network deployment environment*  
<http://www-128.ibm.com/developerworks/webservices/library/ws-sibus/>

- ▶ *Configure platform messaging and Web Services Gateway for a clustered environment, Part 1: Cluster enhancements for WebSphere Application Server Version 6.1 on z/OS*  
<http://www-128.ibm.com/developerworks/webservices/library/ws-clusterenv1/index.html>
- ▶ System Requirements for WebSphere Application Server V6.1  
<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651>
- ▶ WebSphere Application Server Support page  
<http://www-306.ibm.com/software/webservers/appserv/was/support/>
- ▶ WebSphere MQ Information Center  
<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp>
- ▶ MustGather: Service Integration Technology  
<http://www-1.ibm.com/support/docview.wss?uid=swg21199330>
- ▶ MustGather: Documentation required by WebSphere MQ - Java and JMS problems  
<http://www-1.ibm.com/support/docview.wss?uid=swg21176943>
- ▶ The WebSphere MQ library  
<http://www-306.ibm.com/software/integration/wmq/library/>
- ▶ WebSphere MQ reason codes  
<http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/amqzao/amqzao0m.htm>

## How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

Archived

Archived

# Index

## Symbols

\_MQRFH2Allowed 199

## A

activation specification 146–147, 150–152  
addUserToDestinationRole 259  
AMQ9202 188  
AMQ9202. 187  
AMQ9519 185  
AMQ9520 187  
AMQ9534 185  
AMQ9558 187–188  
application authentication 252  
authentication 128, 246, 250–251  
authentication alias 127–128, 205, 223, 237, 252–254  
authentication data 223  
authorization 96, 100, 246–248, 258

## B

batch 71  
BBOCBRAJ 43  
BBOS1A 43  
BBOWBRAK 43  
BBOWSTRT 43  
bean-managed transaction 102  
best effort nonpersistent 18  
BestEffortNonPersistent 105  
bootstrap endpoint 224  
bootstrap process 92  
bootstrap server 41, 84, 86, 88, 90–92  
BootstrapBasicMessaging 41–42, 89–90  
bus connector role 255–256, 258

## C

ClassNotFoundException 53, 57, 124, 131  
classpath 174  
clientconfig.bat 227  
cluster 111, 113, 119–120, 126–128, 130, 132, 138–140, 142, 147  
cluster weight 143  
CNTR0020E 153

com.ibm.db2.jcc 131  
com.ibm.db2.jcc.b.SqlException 50, 55, 62–63  
com.ibm.db2.jcc.t2zos.y 55, 64  
com.ibm.ejs.jms.listener.MDBInvalidConfigException 158–159  
com.ibm.mq.MQException 169  
com.ibm.mqservices.MQInternalException 158–159, 204  
com.ibm.websphere.sib.exception.SIResourceException 41  
com.ibm.ws.sib.comms.server.ObjectStoreFullException 98, 100, 106–107  
com.ibm.ws.sib.jfapchannel.JFapConnectFailedException 41  
com.ibm.ws.sib.msgstore.MessageStoreRuntime-Exception 124, 128–129, 131  
com.ibm.ws.sib.msgstore.PersistenceException 76  
com.ibm.wsspi.sib.core.exception.SIRollbackException 79  
commit 73  
committing state 112  
component-managed authentication alias 252–253  
concurrent message mediation 242  
Connection ExceptionListener 83  
connection factory 7, 140, 159, 164, 174, 205, 226, 252–253  
ConnectionExceptionListener 103  
ConnectionFactory 84, 88, 92  
ConnectionProximity 91, 95  
consumer.receive 103–104  
container authentication 251, 254  
container-managed transactions 102  
control region 21  
Control Region Adjunct 21, 42  
Control Region Adjunct abend 36–37  
core group 135–136, 138  
core group bridge service 146  
core group policy 134, 140  
corrupt messages 163, 198  
createConnection 83, 85, 93, 205, 252  
createConnection() 255  
createConsumer 83, 103  
createProducer 83, 97

createQueueConnection 83, 85, 93  
 createQueueReceiver 83, 103  
 createQueueSender 83, 97  
 createTopicConnection 83, 85, 93  
 createTopicPublisher 83  
 createTopicPublisherError 97  
 createTopicSubscriber 83, 103  
 CWPKI0022E 249  
 CWSIA0004E 248  
 CWSIA0006E 248  
 CWSIA0055E 99–100  
 CWSIA0059E 107  
 CWSIA0062E 98–99, 212, 219  
 CWSIA0063E 98–99, 101  
 CWSIA0067E 98  
 CWSIA0069E 98–100, 248  
 CWSIA0085E 106  
 CWSIA0086E 106  
 CWSIA0090E 106–107, 249  
 CWSIA0241E 41–42, 85–86, 89, 93  
 CWSIC1001E 41–42, 87, 89  
 CWSIC3011E 184, 186  
 CWSIC3080E 184–185  
 CWSIC3096I 191  
 CWSIC3108E 184–185  
 CWSIC8002E 163, 167  
 CWSID0016I 38–39, 122  
 CWSID0027E 123  
 CWSID0035E 123  
 CWSII0205W 247  
 CWSII0211W 248  
 CWSII0212W 248  
 CWSII0214W 249  
 CWSII0219W 212, 227  
 CWSIJ0063E 41, 85, 87–89  
 CWSIK0018E 248  
 CWSIK0025E 98–99, 101  
 CWSIK0033E 98–99, 101  
 CWSIK0101W 244  
 CWSIK0102E 244  
 CWSIK0103E 244  
 CWSIK0104E 191–192  
 CWSIP0309E 249  
 CWSIP0771I 235  
 CWSIP0773I 235  
 CWSIP0775I 235  
 CWSIP0811W 163, 167  
 CWSIP0815W 163, 168  
 CWSIQ0017E 182, 184  
 CWSIS0002E 38, 40, 46, 54, 61–63, 74–75, 78,  
 122, 125, 127–129, 131  
 CWSIS1002 78  
 CWSIS1002E 75, 79  
 CWSIS1501E 62–64, 124–125, 127, 129  
 CWSIS1514E 53, 55–56  
 CWSIS1519E 54, 58, 66, 124, 133, 138  
 CWSIS1522E 53, 57  
 CWSIS1524E 53, 55–56, 124–125, 129  
 CWSIS1535E 54, 59–61, 124, 133, 138  
 CWSIS1538I 54, 66  
 CWSIS1545I 54  
 CWSIS1546I 65–66  
 CWSIS1568I 46  
 CWSIS1569I 46  
 CWSIS1573 78  
 CWSIS1573E 75  
 CWSIS1574E 75  
 CWSIS1575E 75, 79  
 CWSIS1576 78  
 CWSIS1576E 75, 79  
 CWSIT0006E 41–42, 85–86, 88–89  
 CWSIT0007W 41–42, 85, 87–89  
 CWSIT0016E 94, 96  
 CWSIT0019E 85, 90, 93–94  
 CWSIT0022E 94, 96  
 CWSIT0029I 111–112, 116, 118  
 CWSIT0057E 212  
 CWSIT0067E 212  
 CWSIT0086E 93, 95, 212  
 CWSIT0088E 86, 90–91, 94–95  
 CWSIT0089E 94, 96  
 CWSIT0090E 85, 89  
 CWSIT0092E 86, 91–92  
 CWSIT0102E 86, 90–91, 93–95  
 CWSIT0104E 94, 96  
 CWSIV0775W 153  
 CWSIZ0002E 235  
 CWSIZ0011E 235  
 CWSIZ0020E 235  
 CWSIZ0021E 236  
 CWSIZ0045E 236  
 CWSIZ0052I 237  
 CWSIZ0056E 236  
 CWSIZ0057E 236  
 CWSIZ0058E 236  
 CWSIZ0059E 237  
 CWSJP0002E 163, 169  
 CWSJP0023E 163, 167



CWSJP9999E 163, 165  
CWSJR1181E 151–152  
CWSJR1192E 152  
CWSM1042 75  
CWSOM1017E 74, 76  
CWSOM1042E 75, 78–79

## D

data sharing group 63  
data source 58, 125–129  
data source JNDI name 126  
data store 39, 45–46, 49–54, 56, 59–61, 65–66, 133  
data store lock 138  
database  
    configuration 51  
    connectivity 52  
    connectivity errors 53  
    locking errors 53  
    status 51  
DB2 authentication error 127  
DB2 for z/OS 51, 54, 61–64  
db2jcc.jar 57  
DB2UNIVERSAL\_JDBC\_DRIVER\_PATH  
131–132  
dead letter queue 191, 193, 196, 207  
DefaultCoreGroup 134  
disk force on write 72  
DSNUTILS 62  
DSRA8000E 53  
durable subscription 104

## E

endpoint list 139  
exception destination 18, 103, 105, 112, 148, 163, 192, 196, 244  
exception destination policy 19  
exception destination reason 243  
exception header 103, 105  
exclusive access locking 53  
exclusive lock 39, 66  
expired message 112  
express nonpersistent 18  
ExpressNonPersistent 79

## F

failover 54, 58–59, 61, 66, 71–72, 135, 139

FFDC 74–75  
File IO error 124, 133  
file locking 72  
file space allocation 73  
File store 46  
file store 39, 45, 47, 49, 69–72, 74–76, 79–80, 132  
    incorrect path 76  
    requirements 72  
    size 70  
    space constraint 78  
    space constraint 78  
file system level compression 73  
FileNotFoundException 75  
foreign bus 3, 6–7, 139, 194, 198, 200, 209–215, 220, 222–226  
foreign bus link 33, 209–212, 214–216, 219, 221, 223–225, 227–228  
forward routing path 104, 233

## H

HA configuration 4, 120, 125–128, 139  
high message threshold 117  
highly available configuration 4, 120

## I

IBM\_hc 137  
In doubt setting 185  
INC\_UUID 59–60  
in-flight message 15  
InvalidPropertyException 152

## J

J2CA0052E 151  
J2CA0137E 151–152  
J2CA0164E 149–150  
JAAS 127–128, 223  
java.io.FileNotFoundException 75, 77  
java.jms.JMSEException 85, 97, 103  
java.lang.ClassNotFoundException 53  
java.lang.Exception 174, 177  
java.lang.NullPointerException 244  
java.nio.channels.FileChannel.force() 73  
javax.jms.InvalidDestinationException 158–159, 212  
javax.jms.JMSEException 41–42, 83, 85–86, 89, 93  
javax.naming.NameNotFoundException 150–151, 158–159

JDBC provider 125–126, 131  
JMS header 163, 170, 183  
JMS queue 216, 218, 226  
JMS\_PATH 177  
JMSCorrelationID 163, 170, 172  
JMSException 104, 110, 115–116, 163, 174, 204, 247, 249  
JMSSecurityException 247–249  
JMSSecurityException. 248  
JMSXUserID 163, 170–171  
JNDI name 129–130, 150, 159  
JVM log 20–21  
JVM logs 20

## K

known remote message points 13  
known remote queue point 14–15

## L

listener port 149, 152, 158–159  
lock 58, 66, 133  
locked message 102, 105, 113  
logs  
    JVM 20  
lost messages 17, 111, 147, 154, 162, 182–183

## M

maximum message size 192  
MDB 29, 32, 115, 145–147, 149–155, 157–158  
MDB Listener 152  
MDB listener 149  
ME\_UUID 59–61  
mediated destination 230, 233  
mediated message 236  
mediation 229  
mediation handler list 237  
mediation point 3, 10, 115, 230–231, 234–235, 238, 241–242  
mediation thread pool 242  
message delivery problem 123  
message depth 115, 139, 231  
message point 9–10  
message reliability 18, 101  
message state 112–113  
message store 46, 126, 128  
message-driven bean 145–147, 149, 157  
MessageListener 83, 103–104

MessageProducer 99–100  
messaging engine 8  
    cluster 134  
    failover 71  
    fails to accept work 36, 65  
    fails to start 36, 40, 52, 70, 120, 124  
    normal start 39  
    not started 96  
    port conflict 89  
    starts in the wrong location 121  
    state 36  
MQ interoperability port 90  
MQ\_INSTALL\_ROOT 162, 164–166, 174–176  
MQException 159, 204  
MQJE001 159, 169, 204  
MQJE011 204  
MQJMS2008 206–207  
mqrc 159  
MQRC\_Q\_FULL 193  
MQRC\_UNKNOWN\_OBJECT\_NAME 193  
MQRC\_UNKNOWN\_REMOTE\_Q\_MGR 194, 196–197  
MQRFH 199

## N

native library 63  
netstat 42, 90

## O

orphan message 121  
orphan messages 123  
orphaned message 139  
orphaned messages 139  
osgiCfglnit 166, 176  
outbound messages sent 115–118, 222

## P

partition 123, 138, 142  
partitioned destination 123  
pending state 112  
performance 241  
permanent store 70  
PermanentStore file 78  
port conflict 42, 89  
preferred serve 137  
preferred server 123, 134–135, 138, 141  
producer.send 83, 99

provider endpoint 88, 140  
publication point 3, 16, 117, 142  
publication point message depth 116

## Q

queue manager 158, 167–168, 183, 187, 193–194, 205  
queue name 194  
queue point 3, 10–11, 14, 243  
queue point message depth 115  
queued messages 9  
QueueSender 99

## R

RACF 62  
RACLIST 44  
reason code 2016 207  
reason code 2035 205–207  
reason code 2051 206  
reason code 2053 206  
reason code 2058 205  
reason code 2059 205  
reason code 2085 206–207  
reason code 2087 206  
receive allowed property 18  
receiver channel 181–182, 185, 187–188, 192  
Redbooks Web site 266  
    Contact us xvi  
ReliableNonPersistent 79  
remote message point 12–13, 112  
remote messaging engine 215  
remote publication point 16  
remote queue point 14, 16, 111–112, 114  
remote topic space 219–220  
reply queue 138  
response message 138  
reverse routing path 233  
RFH2 header 170  
rollback exception 65

## S

schema 133  
security 212, 214, 222–223, 228, 245–247  
selector 114  
sender channel 180–184, 186–189, 191, 196  
sequence number error 186  
Servant Region 21

Server group 256  
service integration bus 3  
SIB access role types 227  
SIB message trace 105  
SIB MQ interoperability port 42  
SIB MQ interoperability secure port 42, 90  
SIB port 42, 90  
SIB secure port 42  
SIB secure port SIB 90  
SIB Service 84  
SIB\_ENDPOINT\_ADDRESS 140  
SIB\_ENDPOINT\_SECURE\_ADDRESS 140, 223–224  
SIB\_MQ\_ENDPOINT\_ADDRESS 188–189  
SIB\_TCP\_JFAP 42  
sibDDLGenerator 51  
SIBJms\_External trace 8  
SIBMessageTraceContentsMediations 238  
SIBOWNER table 138  
SIBErrorException 167  
SIMPLimitExceededExceptio 110  
SIMPLimitExceededException 111, 116  
SIMPLimitExceededExcpetion 115  
SISessionDroppedException 167  
SQLCODE  
    -471 55  
SQLCODE -471 61  
SSL 247, 249  
SSL chain 228  
SSLHandshakeException 249  
state 8  
static policy 141  
stop mode 9  
storage allocation error 55  
subscription 16, 117  
SYSTEM.Exception.Destination 148  
SystemErr 20  
SystemOut 20

## T

T2 native library 55, 63  
target significance 7  
TCP\_KEEPALIVE 59  
TCP0003E 38, 42  
TEMP database 63  
temporary destination 99  
temporary store 70, 79  
TemporaryStore 78

thread pool size 242  
topic space 4, 116, 142, 214, 219–220  
TopicPublisher  
    99  
toString() 7  
trace 8, 238–239, 241  
transaction 102  
transaction ID 113  
transport chain 140  
trust store 249

## U

UserTransaction 154

## W

WebSphere MQ Explorer 158  
WebSphere MQ Link 180–183  
WebSphere MQ link 179–181, 183–185, 189,  
192–193, 195–199, 201  
WebSphere MQ messaging provider 173–174  
WebSphere MQ reason code 168  
WebSphere MQ Server 161–164, 166  
WLM configuration 4, 120, 126–128, 141  
WMSG0042I 160  
WMSG0902E 175  
WMSG1603E 163, 165–166, 175–176  
WMSG1604E 175–176  
WMSG1605E 163, 165, 175  
workload management configuration 4, 120  
WSAF\_SIB 137  
WSAF\_SIB\_BUS 137  
WSAF\_SIB\_MESSAGING\_ENGINE 137





# WebSphere Application Server V6.1: JMS Problem Determination



**Understand the default messaging and WebSphere MQ providers**

This paper provides problem determination information for WebSphere® Application Server V6.1 JMS application users. It discusses problem diagnosis for JMS applications using the default messaging provider and the WebSphere MQ provider.

**Locate and analyze messages and exceptions**

This publication includes information for WebSphere Application Server V6.1 on distributed platforms and z/OS.

**Diagnose your own messaging problems**

**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](http://ibm.com/redbooks)